Graylog Documentation

Release 3.1.0

Graylog, Inc.

Oct 26, 2021

GRAYLOG

NOTE: There are multiple options for reading this documentation. See link to the lower left.

ARCHITECTURAL CONSIDERATIONS

There are a few rules of thumb when scaling resources for Graylog:

- Graylog nodes should have a focus on CPU power. These also serve the user interface to the browser.

- Elasticsearch nodes should have as much RAM as possible and the fastest disks you can get. Everything depends on I/O speed here.

- MongoDB is storing meta information and configuration data and doesn't need many resources.

Also keep in mind that ingested messages are **only** stored in Elasticsearch. If you have data loss in the Elasticsearch cluster, the messages are gone - except if you have created backups of the indices.

1.1 Minimum setup

This is a minimum Graylog setup that can be used for smaller, non-critical, or test setups. None of the components are redundant, and they are easy and quick to setup.
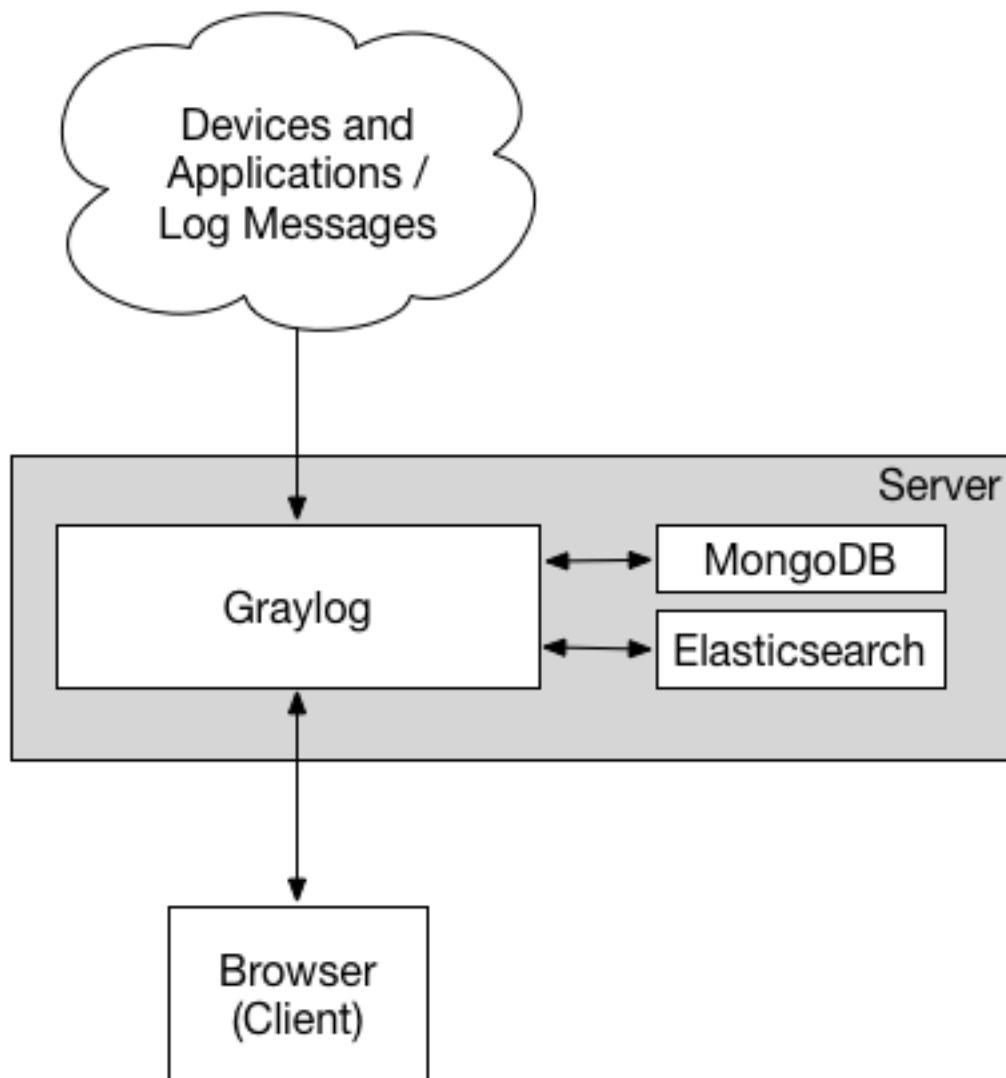
Our *Virtual Machine Appliances* are using this design by default, deploying nginx as *frontend proxy*.

1.2 Bigger production setup

This is a setup for bigger production environments. It has several Graylog nodes behind a load balancer distributing the processing load.

The load balancer can ping the Graylog nodes via HTTP on the Graylog REST API to check if they are alive and take dead nodes out of the cluster.
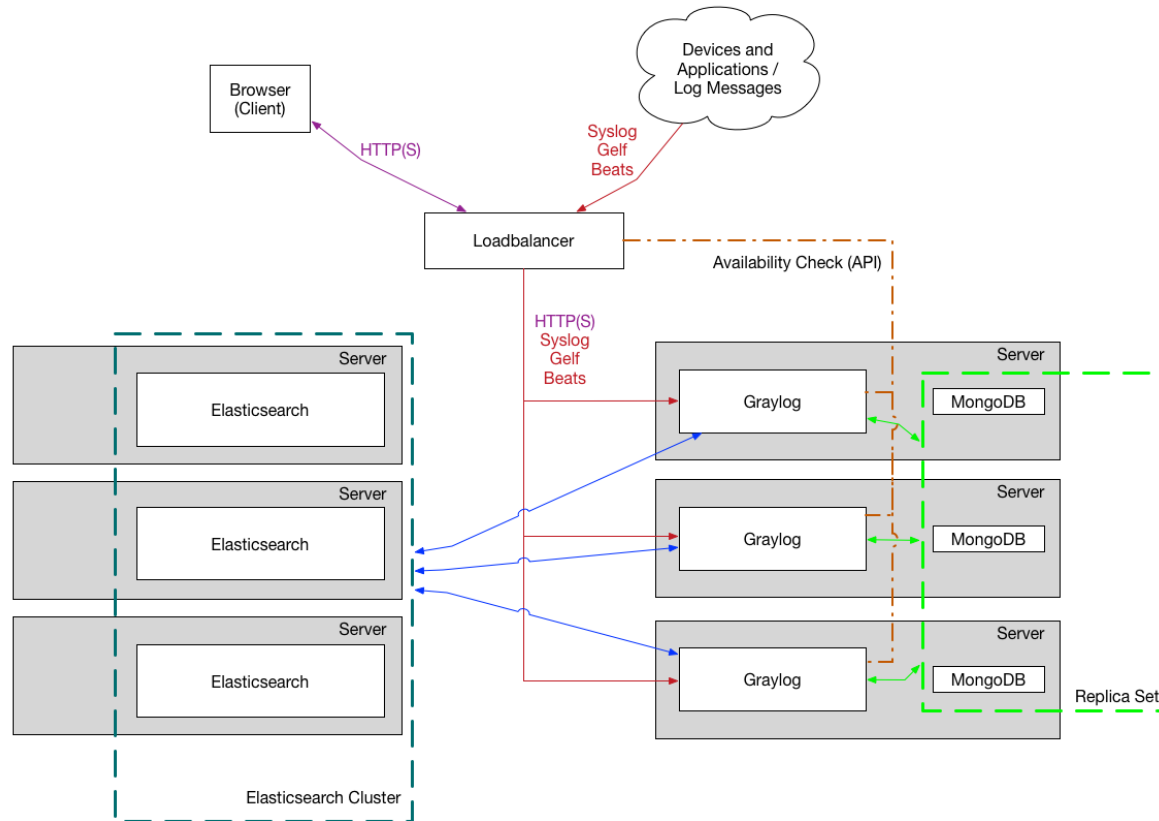
How to plan and configure such a setup is covered in our *Multi-node Setup guide*.

Some guides on the Graylog Marketplace also offer some ideas how you can use RabbitMQ (AMQP) or Apache Kafka to add some queueing to your setup.

1.3 Graylog Architecture Deep Dive

If you are really interested in the Graylog architecture at a more detailed level - whether you want to understand more for planning your architecture design, performance tuning, or just because you love stuff like that, our cheeky engineering team has put together this deep architecture guide. It's not for the faint at heart, but we hope you love it.

GETTING STARTED

This guide is designed for first time users and is intended to give enough key information to get Graylog installed and configured initially. Each section links to additional details on the topic.

Graylog is a very flexible solution. It can be deployed in many different ways. For those who would like to do an initial lab evaluation of Graylog, we recommend starting with the virtual machine appliances.

Virtual Appliances are the fastest way to get started. However, since the Virtual Appliances are generally not suitable for use in production, **they should be used strictly for proof of concept, evaluations or lab environments**. Users should plan to pick one of the other, more flexible installation methods for a production deployment.

If you need assistance planning and building your logging environment we offer professional support that can work with you.

2.1 Planning Your Log Collection

We know you are eager to get Graylog installed and working, but we ask that you take a few moments to review this section and plan your deployment appropriately. Proper planning will make the difference between a useful solution that meets a variety of stakeholder needs and a complicated mess that drains resources and provides little value. There are many factors you must consider when designing a log management solution.

2.1.1 Strategies

Even in a small organization, modern environments produce a lot of log data. Not long ago, 500 MB per day was considered a normal volume of logs for a small shop. Today, 5GB per day is not unusual for a small environment. A large environment can produce a thousand times more than that.

Assuming an average event size of 500k, 5GB per day equates to 125 log events every second, some 10.8 million events per day. With that much information being generated, you will need a strategy to manage it effectively. There are two major approaches.

Minimalist

"Doing the needful"

The Minimalist Strategy proceeds from a "Default No" position when deciding which events to collect. What that means is you don't collect any log unless it is required for an identified business use case. This strategy has some advantages, it keeps licensing and storage costs down, by reducing the volume of collected events. It also minimizes the "noise" produced by extraneous events, allowing analysts to focus on events that have maximum value. Finally, it improves system and query efficiency, improving performance overall.

Maximalist

"Collect it all, let Graylog sort it out."

The Maximalist strategy is to collect all events that are produced by any source. The thinking goes, all log data is potentially valuable, especially for forensics. Collecting it all and keeping it forever guarantees you will have it if you need it. However, this strategy is often not practical, due to budgetary or other constraints. The cost of this strategy can be prohibitive, since many more technical and human resources must be devoted to collection, processing and storage of event data. There is a performance penalty associated with keeping extremely large data sets online that must be considered as well.

2.1.2 Use Cases

"What do you want to do with event data?"

Use cases should inform most decisions during the planning phase. Some of these decisions include determining the event sources from which you must collect, how you will collect from these sources, how much of each event type to store, how events should be enriched and how long to retain the data.

Use case, broadly defined, means the technical steps necessary to achieve a technical and/or business outcome. An easier way to think about it is that a use case is a description of what you want to do with an event log once you've collected it. Use cases are often categorized to group like activities. Some common use case categories are Security, Operations, and DevOps. An example of a Security use case might be monitoring user logins to critical resources. An operations use case might monitor network or hardware performance, while DevOps use cases would focus on real-time application layer monitoring or troubleshooting.

2.1.3 Event Log Sources

"What logs do you need to collect?"

In an environment where seemingly everything generates event logs, it can be difficult to know what to collect. In most cases, selection of event sources should be driven by the use cases you have identified. For example, if the use case is monitoring of user logins to critical resources, the event sources selected should be only those of the critical resources in question. Perhaps the LDAP directory server, Local servers, firewalls, network devices, and key applications.

Some other potential event sources by category.

Security

- Firewalls

- Endpoint Security (EDR, AV, etc.)

- Web Proxies/Gateways

- LDAP/Active Directory

- IDS

- DNS

- DHCP

- Servers

- Workstations

- Netflow

Ops

- Applications

- Network Devices

- Servers

- Packet Capture/Network Recorder

- DNS

- DHCP

- Email

DevOps

- Application Logs

- Load Balancer Logs

- Automation System Logs

- Business Logic

2.1.4 Collection method

"How will you collect it?"

After a list of event sources has been determined, the next step is to decide the method of collection for each source. Although many hardware and software products support common methods such as sending log data via syslog, many do not. It is critical to understand what method each event source uses and what resources that may require. For example, if a log shipper will be required to read logs from a local file on all servers, a log shipper must be selected and tested prior to deployment. In other cases, proprietary API's or software tools must be employed and integrated.

In some cases, changes to the event sources themselves (security devices, network hardware or applications) may be required. Additional planning is often required to deploy and maintain these collection methods over time.

Graylog supports many input types out of the box. More inputs are available in the Graylog Marketplace. At the time of writing, Graylog supports the following:

- Syslog (TCP, UDP, AMQP, Kafka)

- GELF (TCP, UDP, AMQP, Kafka, HTTP)

- AWS (AWS Logs, FlowLogs, CloudTrail)

- Beats/Logstash

- CEF (TCP, UDP, AMQP, Kafka)

- JSON Path from HTTP API

- Netflow (UDP)

- Plain/Raw Text (TCP, UDP, AMQP, Kafka)

The Graylog Marketplace is the central directory of add-ons for Graylog. It contains plugins, content packs, GELF libraries and more content built by Graylog developers and community members.
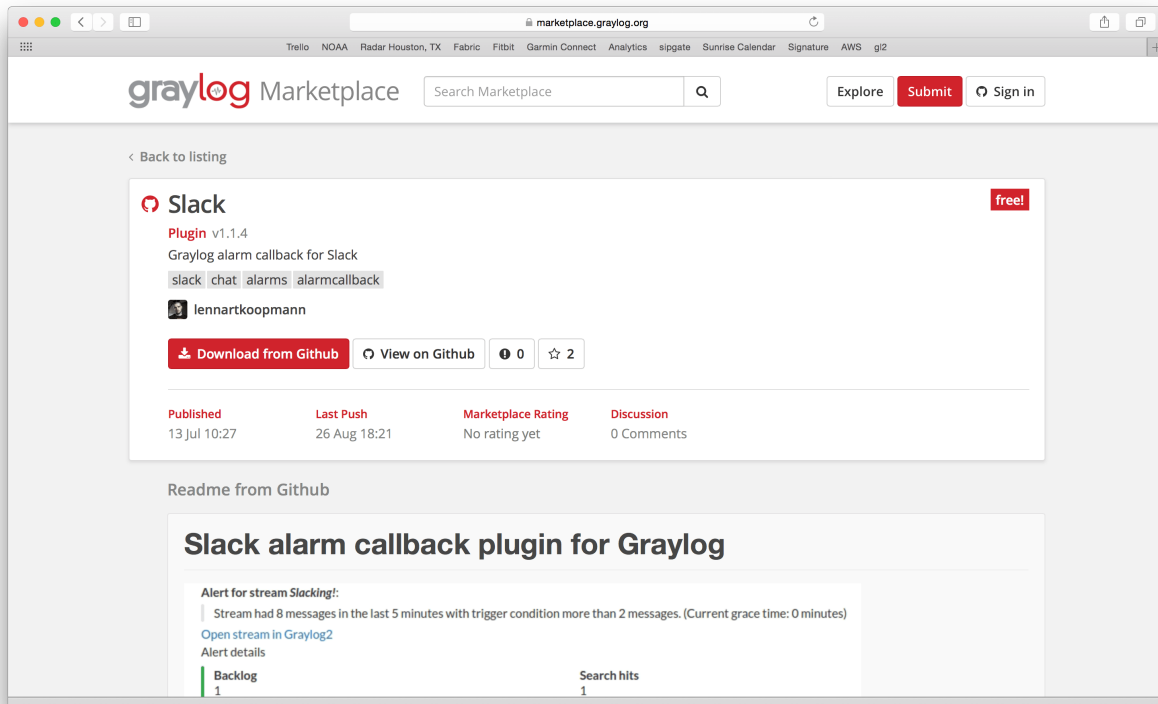
2.1.5 Users

"Who will use the solution?"

The most important user-related factor to consider is the number of users. If the number is large, or if many users will be querying the data simultaneously, you may want to take that into consideration when designing an architecture.

The users' level of skill should be considered. Less technical users may require more pre-built content, such as dashboards. They may also require more training.

Consideration should also be paid as to what event sources each user group should have access. As in all questions of access control, the principle of least privilege should apply.

Some typical user groups include:

- Security Analysts
- Engineers
- Management
- Help Desk

2.1.6 Retention

"How long will you keep the data?"

A key question when planning your log management system is log retention. There are two ways event log data may be retained, online or archived. Online data is stored in Elasticsearch and is searchable through the Graylog GUI. Archived data is stored in a compressed format, either on the Graylog server or on a network file share. It is still searchable, via GREP for example, but must be reconstituted in Graylog in order to be searchable through the GUI again.

Some regulatory frameworks require retention of event log data for a prescribed period. In the absence of a clear requirement, the question becomes one of balancing the cost of retention (storage) versus the utility of having historical data. There is no single answer, as each situation is different.

Most Graylog customers retain 30-90 days online (searchable in Elasticsearch) and 6-13 months of archives.

Calculating Storage Requirements

Like most data stores, Elasticsearch reacts badly when it consumes all available storage. In order to prevent this from happening, proper planning and monitoring must be performed.

Many variables affect storage requirements, such as how much of each message is kept, whether the original message is retained once parsing is complete, and how much enrichment is done prior to storage.

A simple rule of thumb for planning storage is to take your average daily ingestion rate, multiply it by the number of days you need to retain the data online, and then multiply that number by 1.3 to account for metadata overhead. (GB/day x Ret. Days x 1.3 = storage req.).

Elasticsearch makes extensive use of slack storage space in the course of it's operations. Users are strongly encouraged to exceed the minimum storage required for their calculated ingestion rate. When at maximum retention, Elasticsearch storage should not exceed 75% of total space.

2.2 Download & Install Graylog

Graylog can be deployed in many different ways, You should download whatever works best for you. For those who would like to do an initial lab evaluation of Graylog, we recommend starting with the virtual machine appliances.

Virtual Appliances are definitely the fastest way to get started. However, since the virtual appliances are generally not suitable for use in production, **they should be used strictly for proof of concept, evaluations or lab environments**.

The virtual appliances are also completely unsecured. No hardening has been done and all default services are enabled.

For production deployments users should select and deploy one of the other, more flexible, installation methods.

2.2.1 Operating System Packages

Graylog may be installed on the following operating systems.

- Ubuntu
- Debian
- RHEL/CentOS
- SLES

Most customers use package tools like DEB or RPM to install the Graylog software. Details are included in the section, *Operating System Packages*.

2.2.2 Configuration Management

Customers who prefer to deploy graylog via configuration management tools may do so. Graylog currently supports *Chef, Puppet, Ansible*.

2.2.3 Containers

Graylog supports Docker for deployment of Graylog, MongoDB and Elasticsearch. Installation and configuration instructions may be found on the *Docker* installation page.

2.2.4 Virtual Appliances

Virtual Appliances may be downloaded from virtual appliance download page If you are unsure what the latest stable version number is, take a look at our release page.

Supported Virtual Appliances

- OVA

- AWS-AMI

Deployment guide for *Virtual Machine Appliances*.

Deployment guide for *Amazon Web Services*.

Virtual Appliance Caveats

Virtual appliances are not suitable for production deployment out of the box. They do not have sufficient storage, nor do they offer capabilities like index replication that meet high availability requirements.

The virtual appliances are not hardened or otherwise secured. Use at your own risk and apply all security measures required by your organization.

2.3 Initial Configuration

Once the application is installed, there are a few items that must be configured before Graylog may be started for the first time. Both the Graylog `server.conf` and Elasticsearch `elasticsearch.yml` files are configuration files that contain key details needed for intial configuration.

This guide will provide you with the essential settings to get Graylog up and running. There are many other important settings in these files and we encourage you to review them once you are up and running. For more details, please see *server.conf*.

Note: If you are using the virtual appliance, please skip this section and go directly to *Connect to the Web Console*.

2.3.1 server.conf

The file `server.conf` is the Graylog configuration file. The default location for `server.conf` is: `/etc/graylog/server/server.conf`.

Note: All default file locations are listed *here*.

- **Entries are generally expected to be a single line of the form, one of the following:**

    - `propertyName=propertyValue`

    - `propertyName:propertyValue`

- **White space that appears between the property name and property value is ignored, so the following are equivalent:**

    - `name=Stephen`

    - `name = Stephen`

- White space at the beginning of the line is also ignored.

- Lines that start with the comment characters ! or # are ignored. Blank lines are also ignored.

- The property value is generally terminated by the end of the line.

- White space following the property value is not ignored, and is treated as part of the property value.

- The characters newline, carriage return, and tab can be inserted with characters \n, \r, and \t, respectively.

General Properties

- **is_master = true**

    – If you are running more than one instances of Graylog server you must designate (only) one `graylog-server` node as the master. This node will perform periodical and maintenance actions that slave nodes won't.

- **password_secret = <secret>**

    – You MUST set a secret that is used for password encryption and salting. The server will refuse to start if this value is not set. Use at least 64 characters. If you run multiple `graylog-server` nodes, make sure you use the same `password_secret` for all of them!

Note: Generate a secret with for example `pwgen -N 1 -s 96`.

- **root_username = admin**

    – The default root user is named **admin**.

- **root_password_sha2 = <SHA2>**

    – A SHA2 hash of the password you will use for your initial login. Insert a SHA2 hash generated with `echo -n "Enter Password:  " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d" " -f1` and you will be able to log in to the web interface with username **admin** and password **yourpassword**.

Caution: You MUST specify a hash password for the root user (which you only need to initially set up the system and in case you lose connectivity to your authentication backend). This password cannot be changed using the API or via the web interface. If you need to change it, modify it in this file.

Web Properties

- **http_bind_address = 127.0.0.1:9000**

    – The network interface used by the Graylog HTTP interface.

    – This address and port is used by default in the `http_publish_uri`

- **http_publish_uri = http://127.0.0.1:9000/**

    – Web interface listen URI.

    – The HTTP URI of this Graylog node which is used by all clients using the Graylog web interface.

Elasticsearch Properties

- **elasticsearch_hosts = http://node1:9200,http://user:password@node2:19200**
  - List of Elasticsearch hosts Graylog should connect to.
  - Need to be specified as a comma-separated list of valid URIs for the http ports of your elasticsearch nodes.
  - If one or more of your elasticsearch hosts require authentication, include the credentials in each node URI that requires authentication.
  - Default: `http://127.0.0.1:9200` You may retain the default setting only if Elasticsearch is installed on the same host as the Graylog server.

MongoDB

- **mongodb_uri = mongodb://...**
  - MongoDB connection string. Enter your MongoDB connection and authentication information here.
  - See https://docs.mongodb.com/manual/reference/connection-string/ for details.
  - **Examples:**
    * Simple: `mongodb_uri = mongodb://localhost/graylog`
    * Authenticate against the MongoDB server: `mongodb_uri = mongodb://grayloguser:secret@localhost:27017/graylog`
    * Use a replica set instead of a single host: `mongodb_uri = mongodb://grayloguser:secret@localhost:27017,localhost:27018,localhost:27019/graylog?replicaSet=rs01`

Outgoing HTTP

- **http_proxy_uri =**
  - HTTP proxy for outgoing HTTP connections
- **http_non_proxy_hosts =**
  - A list of hosts that should be reached directly, bypassing the configured proxy server.
  - This is a list of patterns separated by ",". The patterns may start or end with a "*" for wildcards.
  - Any host matching one of these patterns will be reached through a direct connection instead of through a proxy.

2.3.2 elasticsearch.yml

`Elasticsearch.yml` is the Elasticsearch configuration file. The default location for elasticsearch.yml is: `/etc/elasticsearch/elasticsearch.yml`.

Several values must be properly configured in order for elasticsearch to work properly.

- **cluster.name: graylog**
  - This value may be set to anything the customer wishes, though we recommend using "graylog".
  - This value must be the same for every Elasticsearch node in a cluster.

- **network.host: 172.30.4.105**

    – By default, Elasticsearch binds to loopback addresses only (e.g. 127.0.0.1). This is sufficient to run a single development node on a server.

    – In order to communicate and to form a cluster with nodes on other servers, your node will need to bind to a non-loopback address.

- **http.port: 9200**

    – Port Elasticsearch will listen on. We recommend using the default value.

- **discovery.zen.ping.unicast.hosts: ["es01.acme.org", "es02.acme.org"]**

    – Elasticsearch uses a custom discovery implementation called "Zen Discovery" for node-to-node clustering and master election. To form a cluster with nodes on other servers, you have to provide a seed list of other nodes in the cluster that are likely to be live and contactable.

    – May be specified as IP address or FQDN.

2.4 Connect to the Web Console

Open a browser and navigate to the URL `http://xxx.xxx.xxx.xxx:9000`, substituting the IP of your graylog server. You should see a Graylog login page similar to the screenshot below.

If using the VM appliance, log in using `admin` for both the username and password. If using either container or OS versions of Graylog, log in as admin and use the password from which you derived the password secret when installing Graylog.
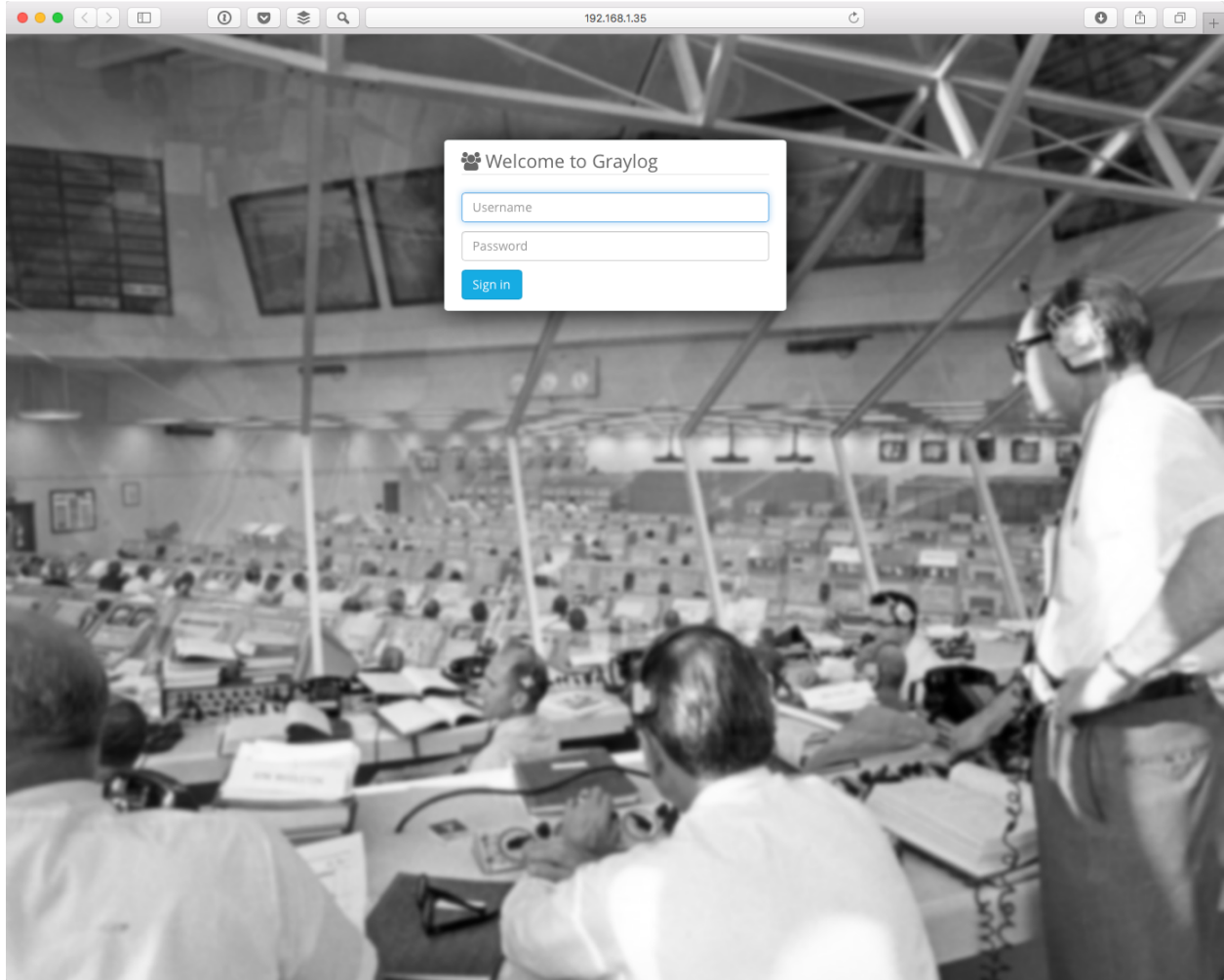
Logging in will get you to a "Getting Started" screen. But, if you are reading this, then you've already found the "Getting Started Guide", so just keep going.

Also, feel free to dismiss the guide or keep it for later.
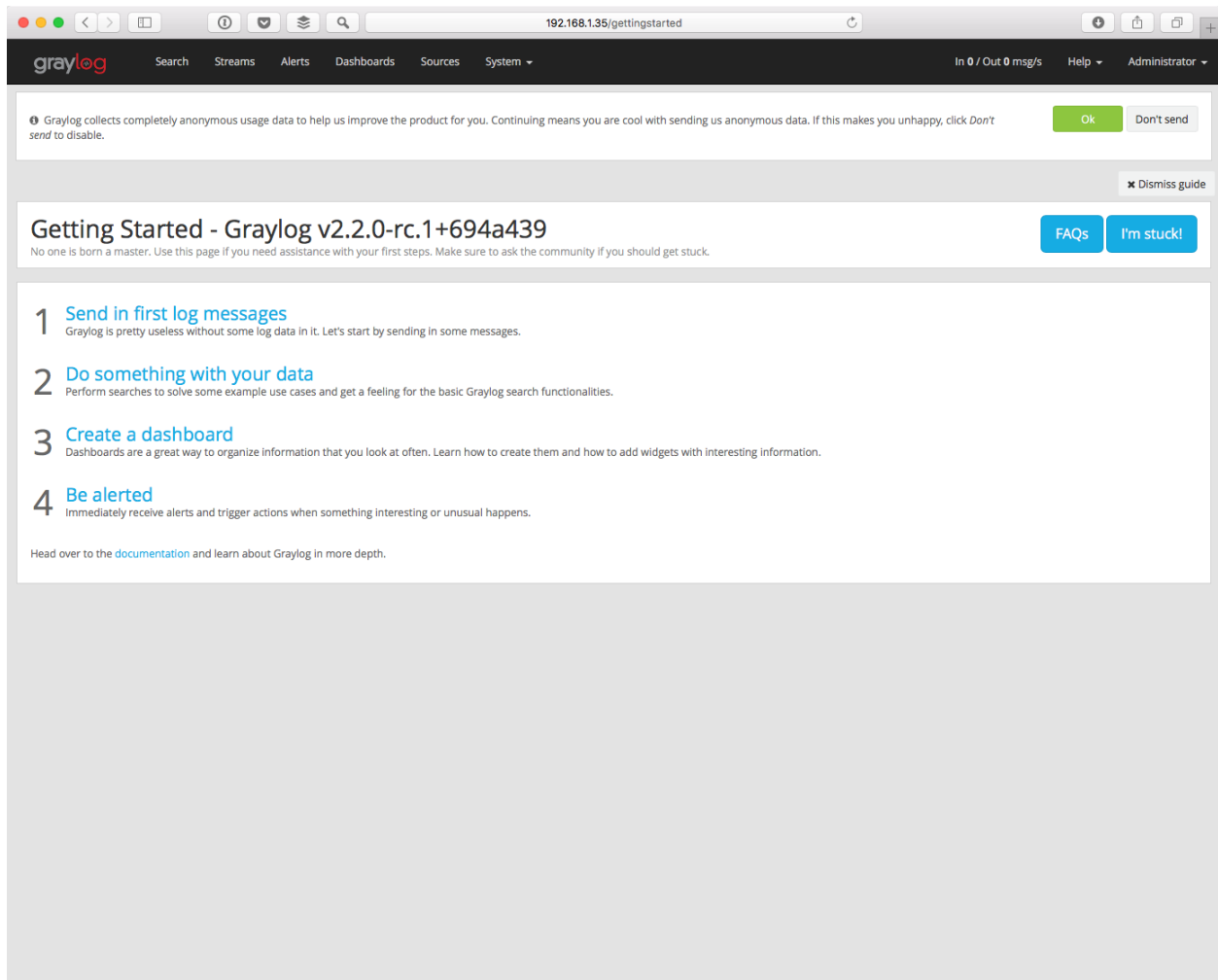
2.5 Explore Graylog

Once messages are being received, you may want to poke around and explore a bit. There are several pages available, though not all pages may be visible to all users, depending on individual permissions. The following is a brief description of each page's purpose and function.

2.5.1 Streams

Streams are a core feature of Graylog and may be thought of as a form of tagging for incoming messages. Streams are a mechanism used to route messages into categories in real-time. Stream rules instruct Graylog which message to route into which streams.

Streams have many uses. First, they are used to route data for storage into an index. They are also used to control access to data, route messages for parsing, enrichment or other modification and determine which messages will be archived.

Streams may be used in conjunction with Alerts to notify users or otherwise respond when a message meets a set of conditions.

Messages may belong to one or to multiple streams. For additional detail, please see *Streams*.

2.5.2 Searches

The Graylog Search page is the interface used to search logs directly. Graylog uses a simplified syntax, very similar to Lucene. Relative or absolute time ranges are configurable from drop down menus. Searches may be saved or visualized as dashboard widgets that may be added directly to dashboards from within the search screen.

Users may configure their own views and may choose to see either summary or complete data from event messages.

For additional detail, please see *Searching*.

2.5.3 Dashboards

Graylog Dashboards are visualizations or summaries of information contained in log events. Each dashboard is populated by one or more widget, Widgets visualize or summarize event log data with data derived from field values such as counts, averages, or totals. Trend indicators, charts, graphs, and maps are easily created to visualize the data.

Dashboard widgets and dashboard layouts are configurable. Dashboard access is controlled via Graylog's role based access control. Dashboards may be imported and exported via content packs.

For additional detail, please see *Dashboards*.

2.5.4 Alerts

Alerts are composed of two related elements, alert conditions and alert notifications. Alert conditions are tied to streams and may be based on the content of a field, the aggregated value of a field, or message count thresholds. An alert notification triggers when a condition is met, typically sending an email or HTTP call back to an analyst or another system.

Additional output types may also be created via plugins. Alerts may be imported and exported via content packs.

For additional detail, please see *Alerts*.

2.5.5 System

Overview

The Overview page displays information relating to the administration of the Graylog instance. It contains information on system notifications, system job status, ingestion rates, Elasticsearch cluster health, indexer failures, Time configuration and the system event messages.

Configuration

The Configuration page allows users to set options or variables related to searches, message processors and plugins.

Nodes

The Nodes page contains summary status information for each Graylog node. Detailed health information and metrics are available from buttons displayed on this page.

Inputs

Usually the first thing configured after initial system setup, Inputs are used to tell Graylog on which port to listen or how to go and retrieve event logs. The Inputs page allows users to create and configure new inputs, to manage extractors, to start and stop inputs, get metrics for each input and to add static fields to incoming messages.

Outputs

Outputs are used to define methods of forwarding data to remote systems, including port, protocol and any other required information. Out of the box, Graylog supports STDOUT and GELF outputs, but users may write their own and more are available in the *Graylog Marketplace*.

Authentication

The Authentication page is used to configure Graylog's authentication providers and manage the active users of this Graylog cluster. Graylog supports LDAP or Active Directory for both authentication and authorization.

Content Packs

Content packs accelerate the set-up process for a specific data source. A content pack can include inputs/extractors, streams, dashboards, alerts and pipeline processors.

Any program element created within Graylog may be exported as Content Packs for use on other systems. These may be kept private by the author, for use in quick deployment of new nodes internally, or may be shared with the community via the Graylog Marketplace. For example, users may create custom Inputs, Streams, Dashboards, and Alerts to support a security use case. These elements may be exported in a content pack and then imported on a newly installed Graylog instance to save configuration time and effort.

Users may download content packs created and shared by other users via the *Graylog Marketplace*. User created content packs are not supported by Graylog, but instead by their authors.

List of Elements Supported in Content Packs

- Inputs
- Grok Patterns
- Outputs
- Streams
- Dashboards
- Lookup Tables
- Lookup Caches
- Lookup Data Adapters

Indices

An Index is the basic unit of storage for data in Elasticsearch. Index sets provide configuration for retention, sharding, and replication of the stored data.

Values, like retention and rotation strategy, are set on a per index basis, so different data may be subjected to different handling rules.

For more details, please see *Index model*.

Sidecars

Graylog created the Sidecar agent to manage fleets of log shippers like Beats or NXLog. These log shippers are used to collect OS logs from Linux and Windows servers. Log shippers are often the simplest way to read logs written locally to a flat file and send them to a centralized log management solution. Graylog supports management of any log shipper as a backend.

For more details, please see *Graylog Sidecar*.

Pipelines

Graylog's Processing Pipelines are a powerful feature that enables user to run a rule, or a series of rules, against a specific type of event. Tied to streams, pipelines allow for routing, blacklisting, modifying and enriching messages as they flow through Graylog. Basically, if you want to parse, change, convert. add to, delete from or drop a message, Pipelines are the place to do it.

For more details, please see *Processing Pipelines*.

2.6 Collect Messages

Once Graylog and associated components are running, the next step is to begin collecting logs.

The first step is to create an input. Inputs define the method by which Graylog collects logs. Out of the box, Graylog supports multiple methods to collect logs, including:

- Syslog (TCP, UDP, AMQP, Kafka)
- GELF(TCP, UDP, AMQP, Kafka, HTTP)
- AWS - AWS Logs, FlowLogs, CloudTrail
- Beats/Logstash
- CEF (TCP, UDP, AMQP, Kafka)
- JSON Path from HTTP API
- Netflow (UDP)
- Plain/Raw Text (TCP, UDP, AMQP, Kafka)

2.6.1 Content packs

Additional inputs may be installed via content packs. Content packs are bundles of Graylog input, extractor, stream, dashboard, and output configurations that can provide full support for a data source. Some content packs are shipped with Graylog by default and some are available from the website. Content packs that were downloaded from the Graylog Marketplace can be imported using the Graylog web interface.

You can load and even create your own content packs from the `System / Content Packs` page of the Graylog web interface.

2.6.2 Create an Input

To create an Input, open the `System / Inputs` page in the top menu, click the arrow in the drop down field, select your input type and click green button labled *Launch new input*.

Usually, the default settings are correct, but you may change any that you wish. Some input types may require authentication or other information specific to that source.

Note: If Graylog is not running as root, you will not have the option of using ports lower than 1024 for inputs. Sending devices may need to be reconfigured. Since best practice dictates that applications should not be run as root, customers who cannot change the event source are encouraged to use a load balancer, or other external means, to perform port translation.

Save the input. It will start automatically.

If your event source is already configured to send events to the port you selected, in the case of *push* event sources like Syslog or CEF, you should start to recieve messages within a few seconds.

Check out *Sending in log data* if you'd like to learn more about the supported options for ingesting messages into Graylog.

2.6.3 Verify Messages Are Being Collected

Once you have an input defined, you will want to verify that you are receiving messages on that input. Check the *Throughput / Metrics* section to the right of your input. You should see the *NetworkIO* values start to climb, showing the amount of data consumed on this input.

Click on the *Show received messages* button next to the input you just created. You should now see the messages received on this input.
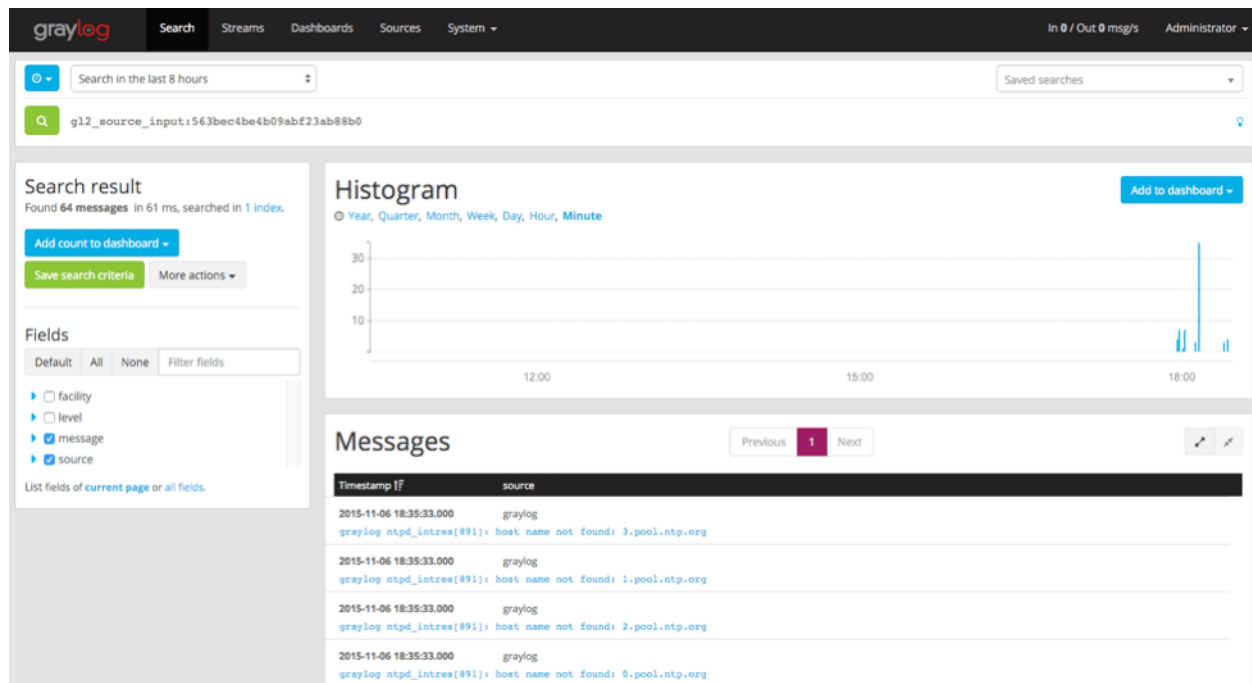
If you click on *Sources* in the top menu, you will see a nice overview of all devices, servers or applications that are sending data into Graylog and how many messages has received from each source. Initially, you may not see much on this page. However, once you start sending data from more systems, their hostnames or IP addresses will also be listed on this page.
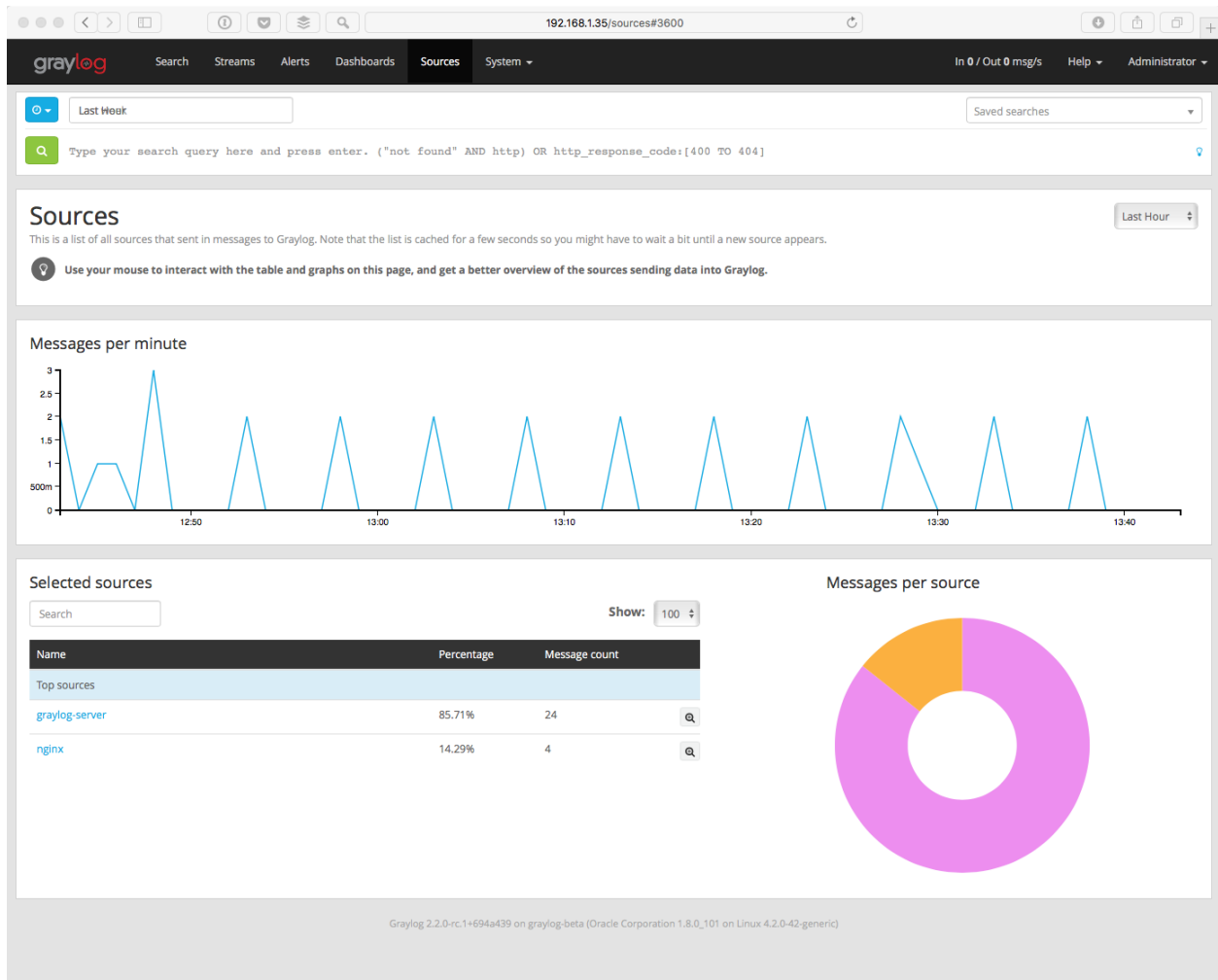
Skip the following section if you are all good.

If You Don't Have Messages

1. Check to see that you made the proper entries in the input configuration described above.

2. Check the configuration at the event source and make sure that it matches the ports and other options defined in the input. For example, if you changed the port for a Syslog UDP input to 5014, be sure the sending device has that same port defined.

3. Check to see if traffic is coming to the defined port. You can use the tcpdump command to do this:

```
$ sudo tcpdump -i lo host 127.0.0.1 and udp port 5014
```

4. Check to see if the server is listening on the host:

```
$ sudo netstat -peanut | grep ":5014"
```

If you still have issues, connect to our community support or get in touch with us via the professional support offering.

INSTALLING GRAYLOG

Modern server architectures and configurations are managed in many different ways. Some people still put new software somewhere in `opt` manually for each server while others have already jumped on the configuration management train and fully automated reproducible setups.

Graylog can be installed in many different ways so you can pick whatever works best for you. We recommend to start with the *virtual machine appliances* for the fastest way to get started and then pick one of the other, more flexible installation methods to build an easier to scale setup.

This chapter is explaining the many ways to install Graylog and aims to help choosing the one that fits your needs.

3.1 Virtual Machine Appliances

3.1.1 Pre-Considerations

Please run this appliance always in a separated network that is isolated from the internet. Read also the notes about production *readiness*!

3.1.2 Download

Download the OVA image. If you are unsure what the latest version number is, take a look at our release page.

3.1.3 Run the image

You can run the OVA in many systems like VMware or Virtualbox. In this example we will guide you through running the OVA in the free Virtualbox on OSX.

In Virtualbox select *File -> Import appliance*:
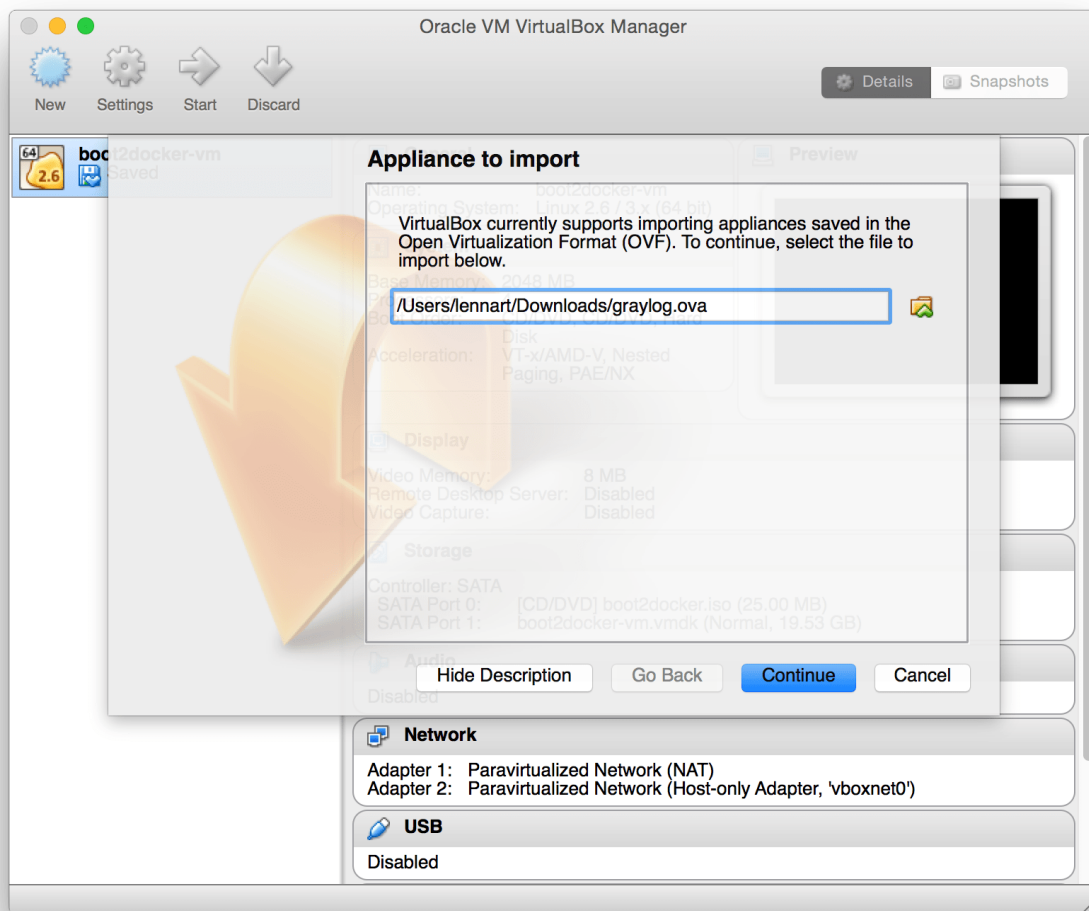
Hit *Continue* and keep the suggested settings on the next page as they are. Make sure that you have enough RAM and CPUs on your local machine. You can lower the resources the virtual machine will get assigned but we recommend to not lower it to ensure a good Graylog experience. In fact you might have to raise it if you plan to scale out later and send more messages into Graylog.

Press *Import* to finish loading the OVA into Virtualbox:
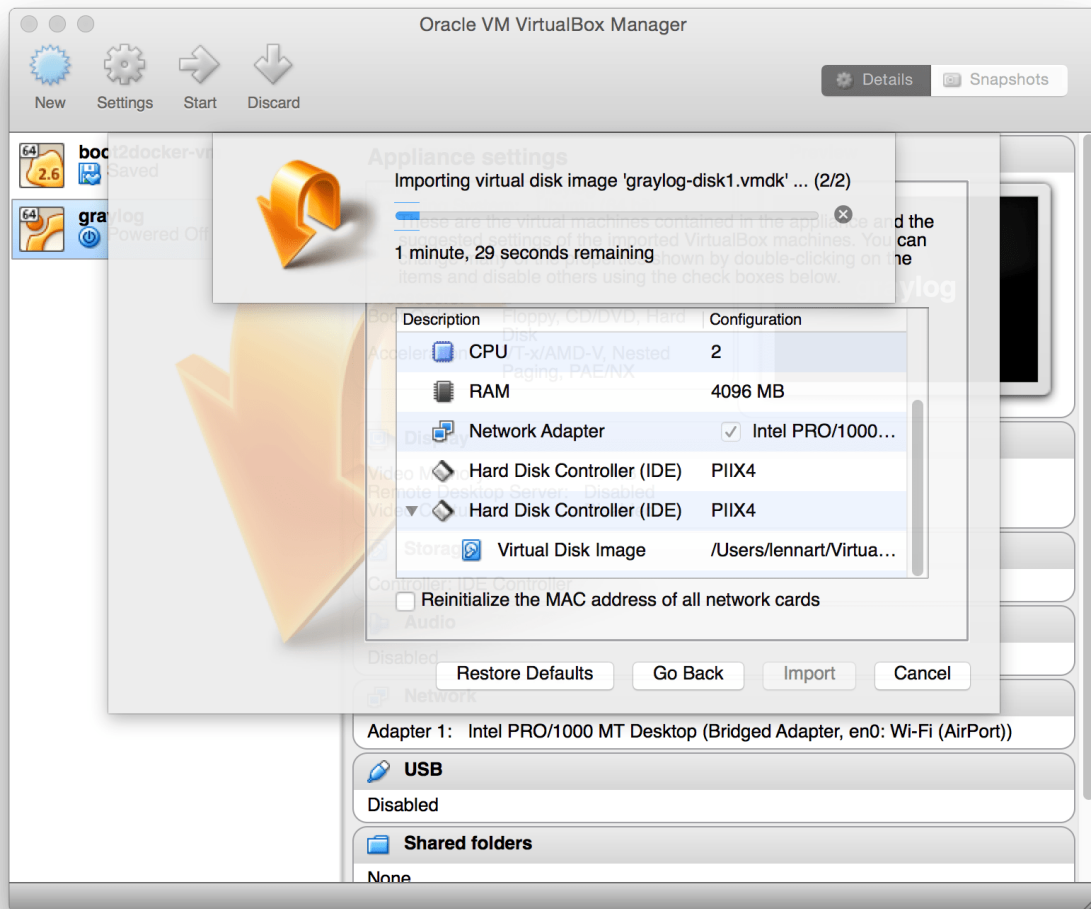
You can now start the VM and should see a login shell like this when the boot completed:
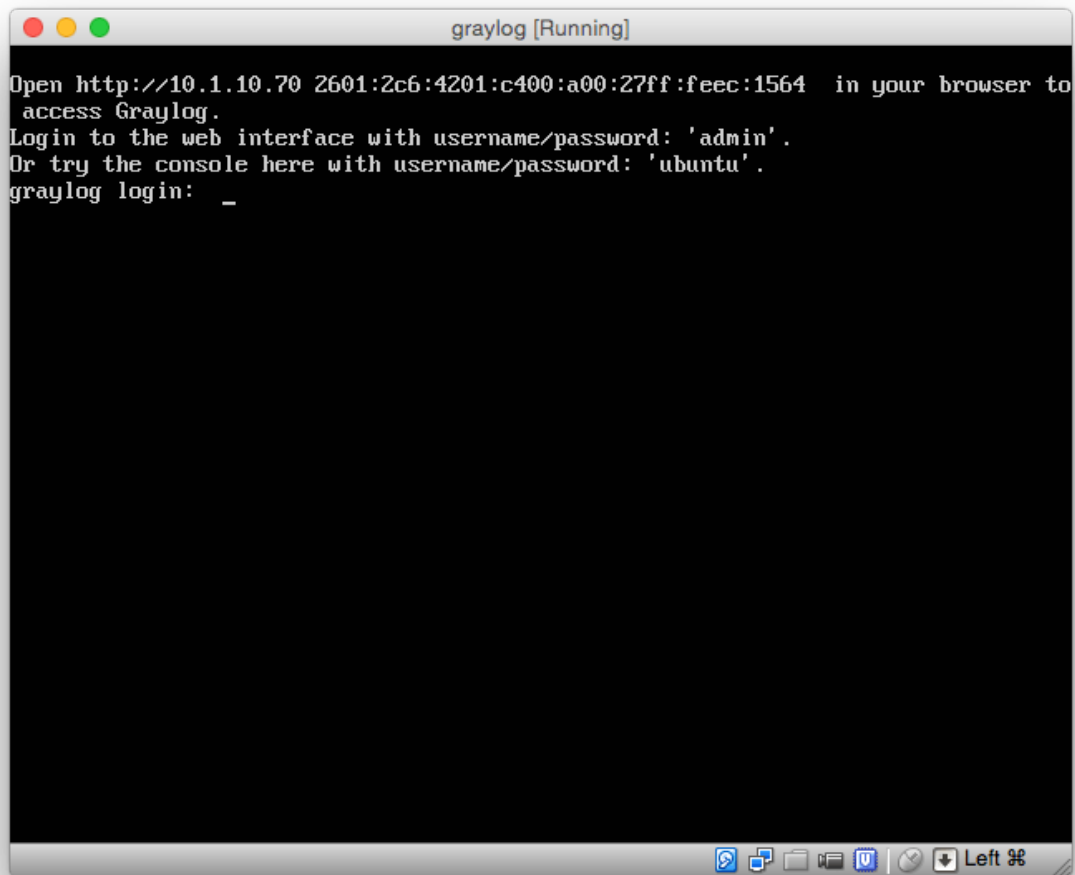
Note: If you don't have a working DHCP server for your virtual machine, you will get the error message:

"Your appliance came up without a configured IP address. Graylog is probably not running correctly!"

In this case, you have to login and edit configuration files under `/etc/netplan/` in order to setup a fixed IP address. Then create the file `/var/lib/graylog-server/firstboot` and reboot.

3.1.4 Logging in

You can log into the shell of the operating system of the appliance with the user *ubuntu* and the password *ubuntu*. You should of course change those credentials.

The web interface is reachable on port 80 at the IP address of your virtual machine. The login prompt of the shell is showing you this IP address, too. (See screenshot above).

The standard user for the web interface is *admin*, the password is shown in the console of the virtual server on the first boot.

3.1.5 Configuration

Please check the *Graylog configuration file* documentation, if you need to further customize your appliance.

3.1.6 VMWare ESXi

> **Warning:** The appliances are build technically with VirtualBox. However most ESXi versions are able to import and run the appliance but be prepared for unexpected troubles especially during import.

If you are using the appliance on a VMWare host, you might want to install the hypervisor tools:

```
sudo apt-get install -y open-vm-tools
```

3.1.7 Update OVA to latest Version

2.x It is not possible to upgrade previous OVAs to Graylog 3.0.0.

3.x Starting with Graylog 3.0.0, OVAs use the Operating System packages, so you can upgrade your appliance by following *this update guide*.

3.1.8 Production readiness

The Graylog appliance is not created to provide a production ready solution. It is build to offer a fast and easy way to try the software itself and not wasting time to install Graylog and it components to any kind of server.

3.2 Operating System Packages

Until configuration management systems made their way into broader markets and many datacenters, one of the most common ways to install software on Linux servers was to use operating system packages. Debian has `DEB`, Red Hat has `RPM` and many other distributions are based on those or come with their own package formats. Online repositories of software packages and corresponding package managers make installing and configuring new software a matter of a single command and a few minutes of time.

Graylog offers official `DEB` and `RPM` package repositories. The packages have been tested on the following operating systems:

- Ubuntu 16.04, 18.04

- Debian 8, 9

- RHEL/CentOS 6, 7

The repositories can be set up by installing a single package. Once that's done, the Graylog packages can be installed via `apt-get` or `yum`. The packages can also be downloaded with a web browser at https://packages.graylog2.org/ if needed.

3.2.1 Prerequisites

Make sure to install and configure the following software before installing and starting any Graylog services:

- Java (>= 8)
- Elasticsearch (5.x or 6.x)
- MongoDB (3.6 or 4.0)

Caution: Graylog 3 **does not** work with Elasticsearch 7.x!

Caution: Graylog 3 **does not** work with MongoDB 4.2!

3.2.2 DEB / APT

Download and install graylog-3.1-repository_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ sudo apt-get install apt-transport-https
$ wget https://packages.graylog2.org/repo/packages/graylog-3.1-repository_latest.deb
$ sudo dpkg -i graylog-3.1-repository_latest.deb
$ sudo apt-get update
$ sudo apt-get install graylog-server
```

Hint: If you want the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to install them now. The following install all official provided packages by Graylog at the same time: `sudo apt-get install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

After the installation completed successfully, Graylog can be started with the following commands. Make sure to use the correct command for your operating system.

OS	Init System	Command
Debian 7	SysV	`sudo service graylog-server start`
Debian 8 & 9, Ubuntu 16.04, 18.04	systemd	`sudo systemctl start graylog-server`

The packages are configured to **not** start any Graylog services during boot. You can use the following commands to start Graylog when the operating system is booting.

OS	Init System	Command
Debian 7	SysV	`sudo update-rc.d graylog-server defaults 95 10`
Debian 8 & 9, Ubuntu 16.06, 18.04	systemd	`sudo systemctl enable graylog-server`

Update to latest version

If you've been using the repository package to install Graylog before, it has to be updated first. The new package will replace the repository URL, without which you will only be able to get bugfix releases of your previously installed version of Graylog.

The update basically works like a fresh installation:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-3.1-repository_latest.deb
$ sudo dpkg -i graylog-3.1-repository_latest.deb
$ sudo apt-get update
$ sudo apt-get install graylog-server
```

> **Caution:** If you have the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to update them together with the Graylog server package. The following command updates all official provided packages by Graylog at the same time: `sudo apt-get install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

Manual Repository Installation

If you don't like to install the repository DEB to get the repository configuration onto your system, you can do so manually (although we don't recommend to do that).

First, add the Graylog GPG keyring which is being used to sign the packages to your system.

> **Hint:** We assume that you have placed the GPG key into `/etc/apt/trusted.gpg.d/`.

Now create a file `/etc/apt/sources.list.d/graylog.list` with the following content:

```
deb https://packages.graylog2.org/repo/debian/ stable 3.1
```

3.2.3 RPM / YUM / DNF

Download and install graylog-3.1-repository_latest.rpm via `rpm(8)`:

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-3.1-repository_
↪latest.rpm
$ sudo yum install graylog-server
```

> **Hint:** If you want the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to install them now. The following install all official provided packages by Graylog at the same time: `sudo yum install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

After the installation completed successfully, Graylog can be started with the following commands. Make sure to use the correct command for your operating system.

OS	Init System	Command
CentOS 6	SysV	`sudo service graylog-server start`
CentOS 7	systemd	`sudo systemctl start graylog-server`

The packages are configured to **not** start any Graylog services during boot. You can use the following commands to start Graylog when the operating system is booting.

OS	Init System	Command
CentOS 6	SysV	`sudo update-rc.d graylog-server defaults 95 10`
CentOS 7	systemd	`sudo systemctl enable graylog-server`

Update to latest version

If you've been using the repository package to install Graylog before, it has to be updated first. The new package will replace the repository URL, without which you will only be able to get bugfix releases of your previously installed version of Graylog.

The update basically works like a fresh installation:

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-3.1-repository_
↪latest.rpm
$ sudo yum clean all
$ sudo yum install graylog-server
```

Running `yum clean all` is required because YUM might use a stale cache and thus might be unable to find the latest version of the `graylog-server` package.

> **Caution:** If you have the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to update them together with the Graylog server package. The following command updates all official provided packages by Graylog at the same time: `sudo yum install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

Manual Repository Installation

If you don't like to install the repository RPM to get the repository configuration onto your system, you can do so manually (although we don't recommend to do that).

First, add the Graylog GPG key which is being used to sign the packages to your system.

> **Hint:** We assume that you have placed the GPG key into `/etc/pki/rpm-gpg/RPM-GPG-KEY-graylog`.

Now create a file named `/etc/yum.repos.d/graylog.repo` with the following content:

```
[graylog]
name=graylog
baseurl=https://packages.graylog2.org/repo/el/stable/3.1/$basearch/
gpgcheck=1
repo_gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-graylog
```

3.2.4 Step-by-step guides

Ubuntu installation

This guide describes the fastest way to install Graylog on Ubuntu 18.04 LTS. All links and packages are present at the time of writing but might need to be updated later on.

> **Warning:** This setup should not be done on publicly exposed servers. This guide **does not cover** security settings!

Prerequisites

Taking a minimal server setup as base will need this additional packages:

```
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt-get install apt-transport-https openjdk-8-jre-headless uuid-runtime pwgen
```

If you get an the error *Unable to locate package* you likely need to enable the universe repository with the following command, at which point you can run the previous commands again:

```
$ sudo add-apt-repository universe
```

MongoDB

The official MongoDB repository provides the most up-to-date version and is the recommended way of installing MongoDB[1]:

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv␣
→9DA31620334BD75D9DCB49F368818C72E52529D4
$ echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0␣
→multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
$ sudo apt-get update
$ sudo apt-get install -y mongodb-org
```

The last step is to enable MongoDB during the operating system's startup:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable mongod.service
$ sudo systemctl restart mongod.service
```

---

[1] For e.g. corporate proxies and other non-free environments you can use a keyserver approach via wget. `wget -qO- 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0x9DA31620334BD75D9DCB49F368818C72E52529D4' | sudo apt-key add -`

Elasticsearch

Graylog can be used with Elasticsearch 6.x, please follow the installation instructions from the Elasticsearch installation guide:

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
$ echo "deb https://artifacts.elastic.co/packages/oss-6.x/apt stable main" | sudo tee␣
↪-a /etc/apt/sources.list.d/elastic-6.x.list
$ sudo apt-get update && sudo apt-get install elasticsearch-oss
```

Make sure to modify the Elasticsearch configuration file (`/etc/elasticsearch/elasticsearch.yml`) and set the cluster name to `graylog` additionally you need to uncomment (remove the # as first character) the line, and add `action.auto_create_index:   false` to the configuration file:

```
cluster.name: graylog
action.auto_create_index: false
```

After you have modified the configuration, you can start Elasticsearch:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable elasticsearch.service
$ sudo systemctl restart elasticsearch.service
```

Graylog

Now install the Graylog repository configuration and Graylog itself with the following commands:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-3.1-repository_latest.deb
$ sudo dpkg -i graylog-3.1-repository_latest.deb
$ sudo apt-get update && sudo apt-get install graylog-server
```

Hint: If you want the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to install them now. The following install all official provided packages by Graylog at the same time: `sudo apt-get install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

Follow the instructions in your `/etc/graylog/server/server.conf` and add `password_secret` and `root_password_sha2`. These settings are mandatory and without them, Graylog will not start!

You need to use the following command to create your `root_password_sha2`:

```
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d"
↪" -f1
```

To be able to connect to Graylog you should set `http_bind_address` to the public host name or a public IP address of the machine you can connect to. More information about these settings can be found in *Configuring the web interface*.

Note: If you're operating a single-node setup and would like to use HTTPS for the Graylog web interface and the Graylog REST API, it's possible to use *NGINX or Apache as a reverse proxy*.

The last step is to enable Graylog during the operating system's startup:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable graylog-server.service
$ sudo systemctl start graylog-server.service
```

The next step is to *ingest messages* into your Graylog and extract the messages with *extractors* or use *the Pipelines* to work with the messages.

Multiple Server Setup

If you plan to have multiple server taking care of different roles in your cluster *like we have in this big production setup* you need to modify only a few settings. This is covered in our *Multi-node Setup guide*. The *default file location guide* will give you the file you need to modify in your setup.

Debian installation

This guide describes the fastest way to install Graylog on Debian Linux 9 (Stretch). All links and packages are present at the time of writing but might need to be updated later on.

> **Warning:** This setup should not be done on publicly exposed servers. This guide **does not cover** security settings!

Prerequisites

If you're starting from a minimal server setup, you will need to install these additional packages:

```
$ sudo apt update && sudo apt upgrade
$ sudo apt install apt-transport-https openjdk-8-jre-headless uuid-runtime pwgen␣
→dirmngr
```

MongoDB

The official MongoDB repository provides the most up-to-date version and is the recommended way of installing MongoDB[1]:

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv␣
→9DA31620334BD75D9DCB49F368818C72E52529D4
$ echo "deb http://repo.mongodb.org/apt/debian stretch/mongodb-org/4.0 main" | sudo␣
→tee /etc/apt/sources.list.d/mongodb-org-4.0.list
$ sudo apt-get update
$ sudo apt-get install -y mongodb-org
```

The last step is to enable MongoDB during the operating system's startup:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable mongod.service
$ sudo systemctl restart mongod.service
```

---

[1] For e.g. corporate proxies and other non-free environments you can use a keyserver approach via wget. `wget -qO- 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0x9DA31620334BD75D9DCB49F368818C72E52529D4' | sudo apt-key add -`

Elasticsearch

Graylog can be used with Elasticsearch 6.x, please follow the installation instructions from the Elasticsearch installation guide:

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
$ echo "deb https://artifacts.elastic.co/packages/oss-6.x/apt stable main" | sudo tee
↪-a /etc/apt/sources.list.d/elastic-6.x.list
$ sudo apt update && sudo apt install elasticsearch-oss
```

Make sure to modify the Elasticsearch configuration file (`/etc/elasticsearch/elasticsearch.yml`) and set the cluster name to `graylog` additionally you need to uncomment (remove the # as first character) the line, and add `action.auto_create_index:   false` to the configuration file:

```
cluster.name: graylog
action.auto_create_index: false
```

After you have modified the configuration, you can start Elasticsearch:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable elasticsearch.service
$ sudo systemctl restart elasticsearch.service
```

Graylog

Now install the Graylog repository configuration and Graylog itself with the following commands:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-3.1-repository_latest.deb
$ sudo dpkg -i graylog-3.1-repository_latest.deb
$ sudo apt update && sudo apt install graylog-server
```

Hint: If you want the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to install them now. The following install all official provided packages by Graylog at the same time: `sudo apt-get install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

Follow the instructions in your `/etc/graylog/server/server.conf` and add `password_secret` and `root_password_sha2`. These settings are mandatory and without them, Graylog will not start!

You need to use the following command to create your `root_password_sha2`:

```
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d"
↪" -f1
```

To be able to connect to Graylog you should set `http_bind_address` to the public host name or a public IP address of the machine you can connect to. More information about these settings can be found in *Configuring the web interface*.

Note: If you're operating a single-node setup and would like to use HTTPS for the Graylog web interface and the Graylog REST API, it's possible to use *NGINX or Apache as a reverse proxy*.

The last step is to enable Graylog during the operating system's startup:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable graylog-server.service
$ sudo systemctl start graylog-server.service
```

The next step is to *ingest messages* into your Graylog and extract the messages with *extractors* or use *the Pipelines* to work with the messages.

Multiple Server Setup

If you plan to have multiple server taking care of different roles in your cluster *like we have in this big production setup* you need to modify only a few settings. This is covered in our *Multi-node Setup guide*. The *default file location guide* will give you the file you need to modify in your setup.

CentOS installation

This guide describes the fastest way to install Graylog on CentOS 7. All links and packages are present at the time of writing but might need to be updated later on.

> **Warning:** This setup should not be done on publicly exposed servers. This guide **does not cover** security settings!

Prerequisites

Taking a minimal server setup as base will need this additional packages:

```
$ sudo yum install java-1.8.0-openjdk-headless.x86_64
```

If you want to use `pwgen` later on you need to Setup EPEL on your system with `sudo yum install epel-release` and install the package with `sudo yum install pwgen`.

MongoDB

Installing MongoDB on CentOS should follow the tutorial for RHEL and CentOS from the MongoDB documentation. First add the repository file `/etc/yum.repos.d/mongodb-org.repo` with the following contents:

```
[mongodb-org-4.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.0.asc
```

After that, install the latest release of MongoDB with `sudo yum install mongodb-org.`

Additionally, run these last steps to start MongoDB during the operating system's boot and start it right away:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable mongod.service
$ sudo systemctl start mongod.service
```

Elasticsearch

Graylog can be used with Elasticsearch 6.x, please follow the installation instructions from the Elasticsearch installation guide.

First install the Elastic GPG key with `rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch` then add the repository file `/etc/yum.repos.d/elasticsearch.repo` with the following contents:

```
[elasticsearch-6.x]
name=Elasticsearch repository for 6.x packages
baseurl=https://artifacts.elastic.co/packages/oss-6.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

followed by the installation of the latest release with `sudo yum install elasticsearch-oss`.

Make sure to modify the Elasticsearch configuration file (`/etc/elasticsearch/elasticsearch.yml`) and set the cluster name to `graylog` additionally you need to uncomment (remove the # as first character) the line, and add `action.auto_create_index:    false` to the configuration file:

```
cluster.name: graylog
action.auto_create_index: false
```

After you have modified the configuration, you can start Elasticsearch:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable elasticsearch.service
$ sudo systemctl restart elasticsearch.service
```

Graylog

Now install the Graylog repository configuration and Graylog itself with the following commands:

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-3.1-repository_
↪latest.rpm
$ sudo yum install graylog-server
```

Hint: If you want the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to install them now. The following install all official provided packages by Graylog at the same time: `sudo yum install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

Follow the instructions in your `/etc/graylog/server/server.conf` and add `password_secret` and `root_password_sha2`. These settings are mandatory and without them, Graylog will not start!

You need to use the following command to create your `root_password_sha2`:

```
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d"
↪" -f1
```

To be able to connect to Graylog you should set `http_bind_address` to the public host name or a public IP address of the machine you can connect to. More information about these settings can be found in *Configuring the web interface*.

Note: If you're operating a single-node setup and would like to use HTTPS for the Graylog web interface and the Graylog REST API, it's possible to use *NGINX or Apache as a reverse proxy*.

The last step is to enable Graylog during the operating system's startup:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable graylog-server.service
$ sudo systemctl start graylog-server.service
```

The next step is to *ingest messages* into your Graylog and extract the messages with *extractors* or use *the Pipelines* to work with the messages.

SELinux information

Hint: We assume that you have `policycoreutils-python` installed to manage SELinux.

If you're using SELinux on your system, you need to take care of the following settings:

- Allow the web server to access the network: `sudo setsebool -P httpd_can_network_connect 1`

- **If the policy above does not comply with your security policy, you can also allow access to each port individually:**

    - Graylog REST API and web interface: `sudo semanage port -a -t http_port_t -p tcp 9000`

    - Elasticsearch (only if the HTTP API is being used): `sudo semanage port -a -t http_port_t -p tcp 9200`

- Allow using MongoDB's default port (27017/tcp): `sudo semanage port -a -t mongod_port_t -p tcp 27017`

If you run a single server environment with *NGINX or Apache proxy*, enabling the Graylog REST API is enough. All other rules are only required in a multi-node setup. Having SELinux disabled during installation and enabling it later, requires you to manually check the policies for MongoDB, Elasticsearch and Graylog.

Hint: Depending on your actual setup and configuration, you might need to add more SELinux rules to get to a running setup.

Further reading

- https://www.nginx.com/blog/nginx-se-linux-changes-upgrading-rhel-6-6/
- https://wiki.centos.org/HowTos/SELinux
- https://wiki.centos.org/TipsAndTricks/SelinuxBooleans
- http://www.serverlab.ca/tutorials/linux/administration-linux/troubleshooting-selinux-centos-red-hat/
- https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/SELinux_Users_and_Administrators_Guide/
- https://www.digitalocean.com/community/tutorials/an-introduction-to-selinux-on-centos-7-part-1-basic-concepts

Multiple Server Setup

If you plan to have multiple server taking care of different roles in your cluster *like we have in this big production setup* you need to modify only a few settings. This is covered in our *Multi-node Setup guide*. The *default file location guide* will give you the file you need to modify in your setup.

SLES installation

This guide describes the fastest way to install Graylog on SLES 12 SP3. All links and packages are present at the time of writing but might need to be updated later on.

> **Warning:** This setup should not be done on publicly exposed servers. This guide **does not cover** security settings!

Prerequisites

The following patterns are required for a minimal setup (see SLES 12 SP3 Deployment Guide):

```
- Base System
- Minimal System (Appliances)
- YaST configuration packages
```

> **Warning:** This Guide assumes that the firewall is disabled and communication is possible to the outside world.

Assuming a minimal setup, you have to install the Java runtime environment:

```
$ sudo zypper install java-1_8_0-openjdk
```

MongoDB

Installing MongoDB on SLES should follow the tutorial for SLES from the MongoDB documentation. Add the GPG key and the repository before installing MongoDB:

```
$ sudo rpm --import https://www.mongodb.org/static/pgp/server-4.0.asc
$ sudo zypper addrepo --gpgcheck "https://repo.mongodb.org/zypper/suse/12/mongodb-org/
↪4.0/x86_64/" mongodb
$ sudo zypper -n install mongodb-org
```

In order to automatically start MongoDB on system boot, you have to activate the MongoDB service by running the following commands:

```
$ sudo chkconfig mongod on
$ sudo systemctl daemon-reload
$ sudo systemctl restart mongod.service
```

Elasticsearch

Graylog can be used with Elasticsearch 6.x, please follow the installation instructions from the Elasticsearch installation guide.

First install the Elastic GPG key with `rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch` then add the repository file `/etc/zypp/repos.d/elasticsearch.repo` with the following contents:

```
[elasticsearch-6.x]
name=Elasticsearch repository for 6.x packages
baseurl=https://artifacts.elastic.co/packages/oss-6.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

followed by the installation of the latest release with `sudo zypper install elasticsearch-oss`.

Make sure to modify the Elasticsearch configuration file (`/etc/elasticsearch/elasticsearch.yml`) and set the cluster name to `graylog` additionally you need to uncomment (remove the # as first character) the line, and add `action.auto_create_index:    false` to the configuration file:

```
cluster.name: graylog
action.auto_create_index: false
```

In order to automatically start Elasticsearch on system boot, you have to activate the Elasticsearch service by running the following commands:

```
$ sudo chkconfig elasticsearch on
$ sudo systemctl daemon-reload
$ sudo systemctl restart elasticsearch.service
```

Graylog

First install the Graylog GPG Key with `rpm --import https://packages.graylog2.org/repo/debian/keyring.gpg` then add the repository file `/etc/zypp/repos.d/graylog.repo` with the following content:

```
[graylog]
name=graylog
baseurl=https://packages.graylog2.org/repo/el/stable/3.1/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-graylog
```

After that, install the latest release with `sudo zypper install graylog-server`.

Hint: If you want the *Integrations Plugins* or the *Enterprise Plugins* installed, you need to install them now. The following install all official provided packages by Graylog at the same time: `sudo zypper install graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

Make sure to follow the instructions in your `/etc/graylog/server/server.conf` and add `password_secret` and `root_password_sha2`. These settings are mandatory and without them, Graylog will not start!

You can use the following command to create your `password_secret`:

```
cat /dev/urandom | base64 | cut -c1-96 | head -1
```

You need to use the following command to create your `root_password_sha2`:

```
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d"
↪" -f1
```

To be able to connect to Graylog you should set `http_bind_address` to the public host name or a public IP address of the machine you can connect to. More information about these settings can be found in *Configuring the web interface*.

Note: If you're operating a single-node setup and would like to use HTTPS for the Graylog web interface and the Graylog REST API, it's possible to use *NGINX or Apache as a reverse proxy*.

The last step is to enable Graylog during the operating system's startup:

```
$ sudo chkconfig graylog-server on
$ sudo systemctl daemon-reload
$ sudo systemctl start graylog-server.service
```

The next step is to *ingest messages* into your new Graylog Cluster and extract the messages with *extractors* or use *the Pipelines* to work with the messages.

Cluster Setup

If you plan to have multiple servers assuming different roles in your cluster *like we have in this big production setup* you need to modify only a few settings. This is covered in our *Multi-node Setup guide*. The *default file location guide* lists the locations of the files you need to modify.

3.2.5 Feedback

Please file a bug report in the GitHub repository for the operating system packages if you run into any packaging related issues.

If you found this documentation confusing or have more questions, please open an issue in the Github repository for the documentation.

3.3 Chef, Puppet, Ansible

The DevOps movement turbocharged market adoption of the newest generation of configuration management and orchestration tools like Chef, Puppet or Ansible. Graylog offers official scripts for all three of them:

- https://supermarket.chef.io/cookbooks/graylog2
- https://forge.puppet.com/graylog/graylog
- https://galaxy.ansible.com/Graylog2/graylog-ansible-role

3.4 Docker

3.4.1 Requirements

You will need a fairly recent version of Docker.

We will use the following Docker images in this chapter:

- Graylog: graylog/graylog
- MongoDB: mongo
- Elasticsearch: docker.elastic.co/elasticsearch/elasticsearch

3.4.2 Quick start

If you want to checkout Graylog on your local desktop without any further customization, you can run the following three commands to create the necessary environment:

```
$ docker run --name mongo -d mongo:3
$ docker run --name elasticsearch \
    -e "http.host=0.0.0.0" \
    -e "ES_JAVA_OPTS=-Xms512m -Xmx512m" \
    -d docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
$ docker run --name graylog --link mongo --link elasticsearch \
    -p 9000:9000 -p 12201:12201 -p 1514:1514 \
    -e GRAYLOG_HTTP_EXTERNAL_URI="http://127.0.0.1:9000/" \
    -d graylog/graylog:3.1
```

> **Warning:** All configuration examples are created to run on the local computer. Should those be used on external servers, adjust GRAYLOG_HTTP_EXTERNAL_URI and add GRAYLOG_HTTP_PUBLISH_URI and GRAYLOG_HTTP_EXTERNAL_URI according to the *server.conf documentation*.

How to get log data in

You can create different kinds of inputs under *System / Inputs*, however you can only use ports that have been properly mapped to your docker container, otherwise data will not go through.

For example, to start a Raw/Plaintext TCP input on port 5555, stop your container and recreate it, while appending -p 5555:5555 to your docker run command:

```
$ docker run --link mongo --link elasticsearch \
    -p 9000:9000 -p 12201:12201 -p 1514:1514 -p 5555:5555 \
    -e GRAYLOG_HTTP_EXTERNAL_URI="http://127.0.0.1:9000/" \
    -d graylog/graylog:3.1
```

Similarly, the same can be done for UDP by appending -p 5555:5555/udp.

After that you can send a plaintext message to the Graylog Raw/Plaintext TCP input running on port 5555 using the following command:

```
$ echo 'First log message' | nc localhost 5555
```

Settings

Graylog comes with a default configuration that works out of the box but you have to set a password for the admin user and the web interface needs to know how to connect from your browser to the Graylog REST API.

Both settings can be configured via environment variables (also see *Configuration*):

```
-e GRAYLOG_ROOT_PASSWORD_
↪SHA2=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
-e GRAYLOG_HTTP_EXTERNAL_URI="http://127.0.0.1:9000/"
```

In this case you can login to Graylog with the username and password admin.

Generate your own admin password with the following command and put the SHA-256 hash into the GRAYLOG_ROOT_PASSWORD_SHA2 environment variable:

```
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d"
↪" -f1
```

All these settings and command line parameters can be put in a docker-compose.yml file, so that they don't have to be executed one after the other.

Example Version 2:

```
version: '2'
services:
  # MongoDB: https://hub.docker.com/_/mongo/
  mongodb:
    image: mongo:3
  # Elasticsearch: https://www.elastic.co/guide/en/elasticsearch/reference/6.x/docker.
↪html
```

(continues on next page)

```
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
  environment:
    - http.host=0.0.0.0
    - transport.host=localhost
    - network.host=0.0.0.0
    - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
  ulimits:
    memlock:
      soft: -1
      hard: -1
  mem_limit: 1g
# Graylog: https://hub.docker.com/r/graylog/graylog/
graylog:
  image: graylog/graylog:3.1
  environment:
    # CHANGE ME (must be at least 16 characters)!
    - GRAYLOG_PASSWORD_SECRET=somepasswordpepper
    # Password: admin
    - GRAYLOG_ROOT_PASSWORD_
↪SHA2=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
    - GRAYLOG_HTTP_EXTERNAL_URI=http://127.0.0.1:9000/
  links:
    - mongodb:mongo
    - elasticsearch
  depends_on:
    - mongodb
    - elasticsearch
  ports:
    # Graylog web interface and REST API
    - 9000:9000
    # Syslog TCP
    - 1514:1514
    # Syslog UDP
    - 1514:1514/udp
    # GELF TCP
    - 12201:12201
    # GELF UDP
    - 12201:12201/udp
```

Example Version 3:

```
version: '3'
services:
  # MongoDB: https://hub.docker.com/_/mongo/
  mongo:
    image: mongo:3
    networks:
      - graylog
  # Elasticsearch: https://www.elastic.co/guide/en/elasticsearch/reference/6.x/docker.
↪html
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
    environment:
      - http.host=0.0.0.0
      - transport.host=localhost
      - network.host=0.0.0.0
```

```
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    deploy:
      resources:
        limits:
          memory: 1g
    networks:
      - graylog
  # Graylog: https://hub.docker.com/r/graylog/graylog/
  graylog:
    image: graylog/graylog:3.1
    environment:
      # CHANGE ME (must be at least 16 characters)!
      - GRAYLOG_PASSWORD_SECRET=somepasswordpepper
      # Password: admin
      - GRAYLOG_ROOT_PASSWORD_
→SHA2=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
      - GRAYLOG_HTTP_EXTERNAL_URI=http://127.0.0.1:9000/
    networks:
      - graylog
    depends_on:
      - mongo
      - elasticsearch
    ports:
      # Graylog web interface and REST API
      - 9000:9000
      # Syslog TCP
      - 1514:1514
      # Syslog UDP
      - 1514:1514/udp
      # GELF TCP
      - 12201:12201
      # GELF UDP
      - 12201:12201/udp
networks:
  graylog:
    driver: bridge
```

After starting all three Docker containers by running `docker-compose up`, you can open the URL `http://127.0.0.1:9000` in a web browser and log in with username `admin` and password `admin` (make sure to change the password later). Change `GRAYLOG_HTTP_EXTERNAL_URI=` to your server IP if you run Docker remotely.

3.4.3 Configuration

Every configuration option can be set via environment variables. Simply prefix the parameter name with `GRAYLOG_` and put it all in upper case.

For example, setting up the SMTP configuration for sending Graylog alert notifications via email, the `docker-compose.yml` would look like this:

```
version: '2'
services:
```

```
mongo:
  image: "mongo:3"
  # Other settings [...]
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
  # Other settings [...]
graylog:
  image: graylog/graylog:3.1
  # Other settings [...]
  environment:
    GRAYLOG_TRANSPORT_EMAIL_ENABLED: "true"
    GRAYLOG_TRANSPORT_EMAIL_HOSTNAME: smtp
    GRAYLOG_TRANSPORT_EMAIL_PORT: 25
    GRAYLOG_TRANSPORT_EMAIL_USE_AUTH: "false"
    GRAYLOG_TRANSPORT_EMAIL_USE_TLS: "false"
    GRAYLOG_TRANSPORT_EMAIL_USE_SSL: "false"
```

Another option would be to store the configuration file outside of the container and edit it directly.

Custom configuration files

Instead of using a long list of environment variables to configure Graylog (see *Configuration*), you can also overwrite the bundled Graylog configuration files.

The bundled configuration files are stored in `/usr/share/graylog/data/config/` inside the Docker container.

Create the new configuration directory next to the `docker-compose.yml` file and copy the default files from GitHub:

```
$ mkdir -p ./graylog/config
$ cd ./graylog/config
$ wget https://raw.githubusercontent.com/Graylog2/graylog-docker/3.1/config/graylog.
→conf
$ wget https://raw.githubusercontent.com/Graylog2/graylog-docker/3.1/config/log4j2.xml
```

The newly created directory `./graylog/config/` with the custom configuration files now has to be mounted into the Graylog Docker container.

This can be done by adding an entry to the volumes section of the `docker-compose.yml` file:

```
version: '2'
services:
  mongodb:
    image: mongo:3
    # Other settings [...]
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
    # Other settings [...]
  graylog:
    image: graylog/graylog:3.1
    # Other settings [...]
    volumes:
      # Mount local configuration directory into Docker container
      - ./graylog/config:/usr/share/graylog/data/config
```

> **Warning:** Graylog is running as USER graylog with the ID `1100` in Docker. That ID need to be able to read the configuration files you place into the container.

Reading individual configuration settings from files

The Graylog Docker image supports reading individual configuration settings from a file. This can be used to secure configuration settings with Docker secrets or similar mechanisms.

This has the advantage, that configuration settings containing sensitive information don't have to be added to a custom configuration file or into an environment variable in plaintext.

The Graylog Docker image checks for the existence of environment variables with the naming scheme `GRAYLOG_<CONFIG_NAME>__FILE` on startup and expects the environment variable to contain the absolute path to a readable file.

For example, if the environment variable `GRAYLOG_ROOT_PASSWORD_SHA2__FILE` contained the value `run/secrets/root_password_hash`, the Graylog Docker image would use the contents of `/run/secrets/root_password_hash` as value for the `root_password_sha2` configuration setting.

Docker secrets

> **Note:** Docker secrets are only available in Docker Swarm services starting with Docker 1.13. Please refer to Manage sensitive data with Docker secrets for more details.

Example for using Docker secrets in a Docker Swarm service:

```
# Create SHA-256 hash of our password
$ echo -n 'password' | sha256sum | awk '{ print $1 }'
5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

# Create a Docker secret named "root_password_hash"
$ printf '5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8' | \
  docker secret create root_password_hash -
nlujwooo5uu6z0m91bmve79uo

$ docker secret ls
ID                          NAME                    DRIVER          CREATED       ␣
↪    UPDATED
nlujwooo5uu6z0m91bmve79uo   root_password_hash                      34 seconds ago ␣
↪    34 seconds ago

# Create Docker Swarm service named "graylog" with access
# to the secret named "root_password_hash"
$ docker service create --name graylog \
 --secret root_password_hash \
 -e GRAYLOG_ROOT_PASSWORD_SHA2__FILE=/run/secrets/root_password_hash \
 -p 9000:9000 graylog/graylog:3.1
mclk5gm39ingk51s869dc0htz
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
```

(continues on next page)

```
$ docker service ls
ID                    NAME              MODE          REPLICAS        IMAGE␣
→          PORTS
mclk5gm39ing     graylog         replicated     1/1                         ␣
→graylog:latest     *:9000->9000/tcp
```

3.4.4 Persisting data

In order to make the recorded data persistent, you can use external volumes to store all data. In case of a container restart, this will simply re-use the existing data from the former instances.

Using Docker volumes for the data of MongoDB, Elasticsearch, and Graylog, the `docker-compose.yml` file looks as follows:

```yaml
version: '2'
services:
  # MongoDB: https://hub.docker.com/_/mongo/
  mongodb:
    image: mongo:3
    volumes:
      - mongo_data:/data/db
  # Elasticsearch: https://www.elastic.co/guide/en/elasticsearch/reference/6.x/docker.
→html
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
    volumes:
      - es_data:/usr/share/elasticsearch/data
    environment:
      - http.host=0.0.0.0
      - transport.host=localhost
      - network.host=0.0.0.0
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    mem_limit: 1g
  # Graylog: https://hub.docker.com/r/graylog/graylog/
  graylog:
    image: graylog/graylog:3.1
    volumes:
      - graylog_journal:/usr/share/graylog/data/journal
    environment:
      # CHANGE ME (must be at least 16 characters)!
      - GRAYLOG_PASSWORD_SECRET=somepasswordpepper
      # Password: admin
      - GRAYLOG_ROOT_PASSWORD_
→SHA2=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
      - GRAYLOG_HTTP_EXTERNAL_URI=http://127.0.0.1:9000/
    links:
      - mongodb:mongo
      - elasticsearch
    depends_on:
      - mongodb
      - elasticsearch
```

```
    ports:
      # Graylog web interface and REST API
      - 9000:9000
      # Syslog TCP
      - 1514:1514
      # Syslog UDP
      - 1514:1514/udp
      # GELF TCP
      - 12201:12201
      # GELF UDP
      - 12201:12201/udp
# Volumes for persisting data, see https://docs.docker.com/engine/admin/volumes/
↪volumes/
volumes:
  mongo_data:
    driver: local
  es_data:
    driver: local
  graylog_journal:
    driver: local
```

Start all services with exposed data directories:

```
$ docker-compose up
```

3.4.5 Plugins

In order to add plugins you can build a new image based on the existing graylog/graylog Docker image with the needed plugin included or you add a volume that points to the locally downloaded plugin file.

New Docker image

Simply create a new Dockerfile in an empty directory with the following contents:

```
FROM graylog/graylog:3.1
RUN wget -O /usr/share/graylog/plugin/graylog-plugin-auth-sso-3.0.0.jar https://
↪github.com/Graylog2/graylog-plugin-auth-sso/releases/download/3.0.0/graylog-plugin-
↪auth-sso-3.0.0.jar
```

Build a new image from the new `Dockerfile` (also see docker build):

```
$ docker build -t graylog-with-sso-plugin .
```

In this example, we created a new image with the SSO plugin installed. From now on reference to the newly built image instead of graylog/graylog.

The `docker-compose.yml` file has to reference the new Docker image:

```
version: '2'
services:
  mongo:
    image: "mongo:3"
    # Other settings [...]
  elasticsearch:
```

```
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
    # Other settings [...]
  graylog:
    image: graylog-with-sso-plugin
    # Other settings [...]
```

Volume-mounted plugin

Instead of building a new docker image, you can also add additional plugins by mounting them directly and individually into the `plugin` folder of the original Docker image. This way, you don't have to create a new docker image every time you want to add a new plugin (or remove an old one).

Simply create a `plugin` folder, download the plugin(s) you want to install into it and mount each file as an additional volume into the docker container:

```
$ mkdir -p ./graylog/plugin
$ wget -O ./graylog/plugin/graylog-plugin-auth-sso-2.3.0.jar https://github.com/
↪Graylog2/graylog-plugin-auth-sso/releases/download/2.3.0/graylog-plugin-auth-sso-2.
↪3.0.jar
```

The `docker-compose.yml` file has to reference the new Docker image:

```
version: '2'
services:
  mongo:
    image: "mongo:3"
    # Other settings [...]
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
    # Other settings [...]
  graylog:
    image: graylog/graylog:3.1
    # Other settings [...]
    volumes:
      # Mount local plugin file into Docker container
      - ./graylog/plugin/graylog-plugin-auth-sso-2.3.0.jar:/usr/share/graylog/plugin/
↪graylog-plugin-auth-sso-2.3.0.jar
```

You can add as many of these links as you wish in your `docker-compose.yml` file. Simply restart the container and docker will recreate the graylog container with the new volumes included:

```
$ docker-compose restart
```

3.4.6 Kubernetes automatic master selection

Running Graylog in Kubernetes opens the challenge to set the `is_master=true` setting only for one node in the cluster. The problem can be solved by calculating the name of the pod if Graylog is running in a stateful set with the following environment variable:

```
env:
- name: POD_NAME
  valueFrom:
    fieldRef:
      fieldPath: metadata.name
```

For a stateful set, the name of the first pod in a cluster always ends with `-0`. See the Documentation about stateful set . The master selection mechanism in docker-entrypoint.sh file does the following:

- Examine if Graylog is running inside Kubernetes

- Verify that the pod name ends in `-0`

- Set `is_master=true` for this container

3.4.7 Nomad automatic master selection

When running Graylog in a Nomad cluster, you must ensure that only one node in the cluster has the setting `is_master=true` in the server.conf file.

Whether the container is running on Nomad may be identified with an environmetal check on NO-MAD_ALLOC_INDEX. Should that variable be set to 0, the container will set Graylog to `is_master=true`. If the variable is set to anything other than 0 , it will set Graylog to `is_master=false`.

3.4.8 Troubleshooting

- In case you see warnings regarding open file limit, try to set ulimit from the outside of the container:

```
$ docker run --ulimit nofile=64000:64000 ...
```

- The `devicemapper` storage driver can produce problems with Graylogs disk journal on some systems. In this case please pick another driver like `aufs` or `overlay`.

3.4.9 Testing a beta version

> **Caution:** We only recommend running pre-release versions if you are an experienced Graylog user and know what you are doing.

You can also run a pre-release (alpha, beta, or release candidate) version of Graylog using Docker.

The pre-releases are tagged in the graylog/graylog Docker image.

See the available tags for the Graylog image on Docker Hub and pick an alpha/beta/rc tag like this:

```
$ docker run --link mongo --link elasticsearch -p 9000:9000 -p 12201:12201 -p
→1514:1514 \
    -e GRAYLOG_HTTP_BIND_ADDRESS="127.0.0.1:9000" \
    -d graylog/graylog:3.1.0-beta.3-1
```

3.5 Amazon Web Services

3.5.1 AMIs

Select your AMI and AWS Region.

3.5.2 Usage

- Click on *Launch instance* for your AWS region to start Graylog into.
- Choose an instance type **with at least 4GB memory**.
- Finish the wizard and spin up the VM.
- Open port 80 and 22 in the applied security group to access the web interface.
- Login to the instance via SSH as user *ubuntu* to see web login credentials.
- additionally open more ports for ingesting log data, like 514 for syslog or 12201 for the GELF protocol.

Open *http://<private ip>* in your browser to access the Graylog web interface. Default username is *admin* with the password shown on the first SSH login.

3.5.3 Networking

Your browser needs access to port 80 for reaching the web interface. Make sure that a security group is opening that port. On the appliance a NGINX instance is used as proxy to simplify network access. Take a look in the configuration */etc/nginx/sites-available/default* for further fine tuning.

3.5.4 HTTPS

In order to enable HTTPS for the web interface port 443 needs also be open. The configuration can be done with NGINX as well. See *Using HTTPS* for a full reference.

3.5.5 Basic configuration

The Graylog appliance is based on Ubuntu system packages so all configuration changes can be done analog to the rest of this documentation. See *Configuring Graylog*

3.5.6 Production readiness

The Graylog appliance is not created to provide a production ready solution. It is build to offer a fast and easy way to try the software itself and not wasting time to install Graylog and it components to any kind of server.

3.6 Microsoft Windows

Unfortunately there is no supported way to run Graylog on Microsoft Windows operating systems even though all parts run on the Java Virtual Machine. We recommend to run the *virtual machine appliances* on a Windows host. It should be technically possible to run Graylog on Windows but it is most probably not worth the time to work your way around the cliffs.

Should you require running Graylog on Windows, you need to disable the message journal in `graylog-server` by changing the following setting in the `graylog.conf`:

```
message_journal_enabled = false
```

Due to restrictions of how Windows handles file locking the journal will not work correctly.

Please note that this impacts Graylog's ability to buffer messages, so we strongly recommend running Graylog on Linux. Consider a Linux virtual machine on a Windows host. Graylog setups on Windows are no fun and not officially supported.

3.7 Manual Setup

3.7.1 Graylog server on Linux

Prerequisites

Graylog depends on MongoDB and Elasticsearch to operate, please refer to *the system requirements* for details.

Downloading and extracting the server

Download the tar archive from the download pages and extract it on your system:

```
~$ tar xvfz graylog-VERSION.tgz
~$ cd graylog-VERSION
```

Configuration

Now copy the example configuration file:

```
~# cp graylog.conf.example /etc/graylog/server/server.conf
```

You can leave most variables as they are for a first start. All of them should be well documented.

Configure at least the following variables in `/etc/graylog/server/server.conf`:

- **`is_master = true`**

    - Set only one `graylog-server` node as the master. This node will perform periodical and maintenance actions that slave nodes won't. Every slave node will accept messages just as the master nodes. Nodes will fall back to slave mode if there already is a master in the cluster.

- **`password_secret`**

    - You must set a secret that is used for password encryption and salting here. The server will refuse to start if it's not set. Generate a secret with for example `pwgen -N 1 -s 96`. If you run multiple `graylog-server` nodes, make sure you use the same `password_secret` for all of them!

- **root_password_sha2**

    - A SHA2 hash of a password you will use for your initial login. Set this to a SHA2 hash generated with `echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d" " -f1` and you will be able to log in to the web interface with username **admin** and password **yourpassword**.

- **elasticsearch_hosts**

    - List of Elasticsearch hosts Graylog should connect to.

- **mongodb_uri**

    - Enter your MongoDB connection and authentication information here.

Starting the server

You need to have Java installed. Running the OpenJDK is totally fine and should be available on all platforms. For example on Debian it is:

```
~$ apt-get install openjdk-8-jre
```

Start the server:

```
~$ cd bin/
~$ ./graylogctl start
```

The server will try to write a `node_id` to the `graylog-server-node-id` file. It won't start if it can't write there because of for example missing permissions.

See the startup parameters description below to learn more about available startup parameters. Note that you might have to be *root* to bind to the popular port 514 for syslog inputs.

You should see a line like this in the debug output of Graylog successfully connected to your Elasticsearch cluster:

```
2013-10-01 12:13:22,382 DEBUG: org.elasticsearch.transport.netty - [graylog-server]
→connected to node [[Unuscione, Angelo][thN_gIBkQDm2ab7k-2Zaaw][inet[/10.37.160.
→227:9300]]]
```

You can find the logs of Graylog in the directory `logs/`.

Important: All systems running Graylog must have synchronised system time. We strongly recommend to use [NTP](#) or similar mechanisms on all machines of your Graylog infrastructure.

Supplying external logging configuration

Graylog is using [Apache Log4j 2](#) for its internal logging and ships with a [default log configuration file](#) which is embedded within the shipped JAR.

In case you need to modify Graylog's logging configuration, you can supply a Java system property specifying the path to the configuration file in your start script (e. g. `graylogctl`).

Append this before the `-jar` parameter:

```
-Dlog4j.configurationFile=file:///path/to/log4j2.xml
```

Substitute the actual path to the file for the `/path/to/log4j2.xml` in the example.

In case you do not have a log rotation system already in place, you can also configure Graylog to rotate logs based on their size to prevent the log files to grow without bounds using the [RollingFileAppender](#).

One such example `log4j2.xml` configuration is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration packages="org.graylog2.log4j" shutdownHook="disable">
  <Appenders>
      <RollingFile name="RollingFile" fileName="/tmp/logs/graylog.log"
                   filePattern="/tmp/logs/graylog-%d{yyyy-MM-dd}.log.gz">
        <PatternLayout>
          <Pattern>%d %-5p: %c - %m%n</Pattern>
        </PatternLayout>
        <!-- Rotate logs every day or when the size exceeds 10 MB (whichever comes
→first) -->
        <Policies>
          <TimeBasedTriggeringPolicy modulate="true"/>
          <SizeBasedTriggeringPolicy size="10 MB"/>
        </Policies>
        <!-- Keep a maximum of 10 log files -->
        <DefaultRolloverStrategy max="10"/>
      </RollingFile>

      <Console name="STDOUT" target="SYSTEM_OUT">
          <PatternLayout pattern="%d %-5p: %c - %m%n"/>
      </Console>

      <!-- Internal Graylog log appender. Please do not disable. This makes internal
→log messages available via REST calls. -->
      <Memory name="graylog-internal-logs" bufferSize="500"/>
  </Appenders>
  <Loggers>
      <Logger name="org.graylog2" level="info"/>
      <Logger name="com.github.joschi.jadconfig" level="warn"/>
      <Logger name="org.apache.directory.api.ldap.model.message.BindRequestImpl"
→level="error"/>
      <Logger name="org.elasticsearch.script" level="warn"/>
      <Logger name="org.graylog2.periodical.VersionCheckThread" level="off"/>
      <Logger name="com.joestelmach.natty.Parser" level="warn"/>
      <Logger name="kafka.log.Log" level="warn"/>
      <Logger name="kafka.log.OffsetIndex" level="warn"/>
      <Logger name="org.apache.shiro.session.mgt.AbstractValidatingSessionManager"
→level="warn"/>
      <Root level="warn">
          <AppenderRef ref="STDOUT"/>
          <AppenderRef ref="RollingFile"/>
          <AppenderRef ref="graylog-internal-logs"/>
      </Root>
  </Loggers>
</Configuration>
```

Command line (CLI) parameters

There are a number of CLI parameters you can pass to the call in your `graylogctl` script:

- `-h`, `--help`: Show help message

- `-f CONFIGFILE`, `--configfile CONFIGFILE`: Use configuration file `CONFIGFILE` for Graylog; default: `/etc/graylog/server/server.conf`

- `-d`, `--debug`: Run in debug mode

- `-l`, `--local`: Run in local mode. Automatically invoked if in debug mode. Will not send system statistics, even if enabled and allowed. Only interesting for development and testing purposes.

- `-p PIDFILE`, `--pidfile PIDFILE`: Set the file containing the PID of graylog to `PIDFILE`; default: `/tmp/graylog.pid`

- `-np`, `--no-pid-file`: Do not write PID file (overrides `-p`/`--pidfile`)

- `--version`: Show version of Graylog and exit

Problems with IPv6 vs. IPv4?

If your Graylog node refuses to listen on IPv4 addresses and always chooses for example a `http_bind_address` like `:::9000` you can tell the JVM to prefer the IPv4 stack.

Add the `java.net.preferIPv4Stack` flag in your `graylogctl` script or from wherever you are calling the `graylog.jar`:

```
~$ sudo -u graylog java -Djava.net.preferIPv4Stack=true -jar graylog.jar
```

Create a message input and send a first message

Log in to the web interface on port 9000 (e.g. `http://127.0.0.1:9000`) and navigate to *System -> Inputs*.
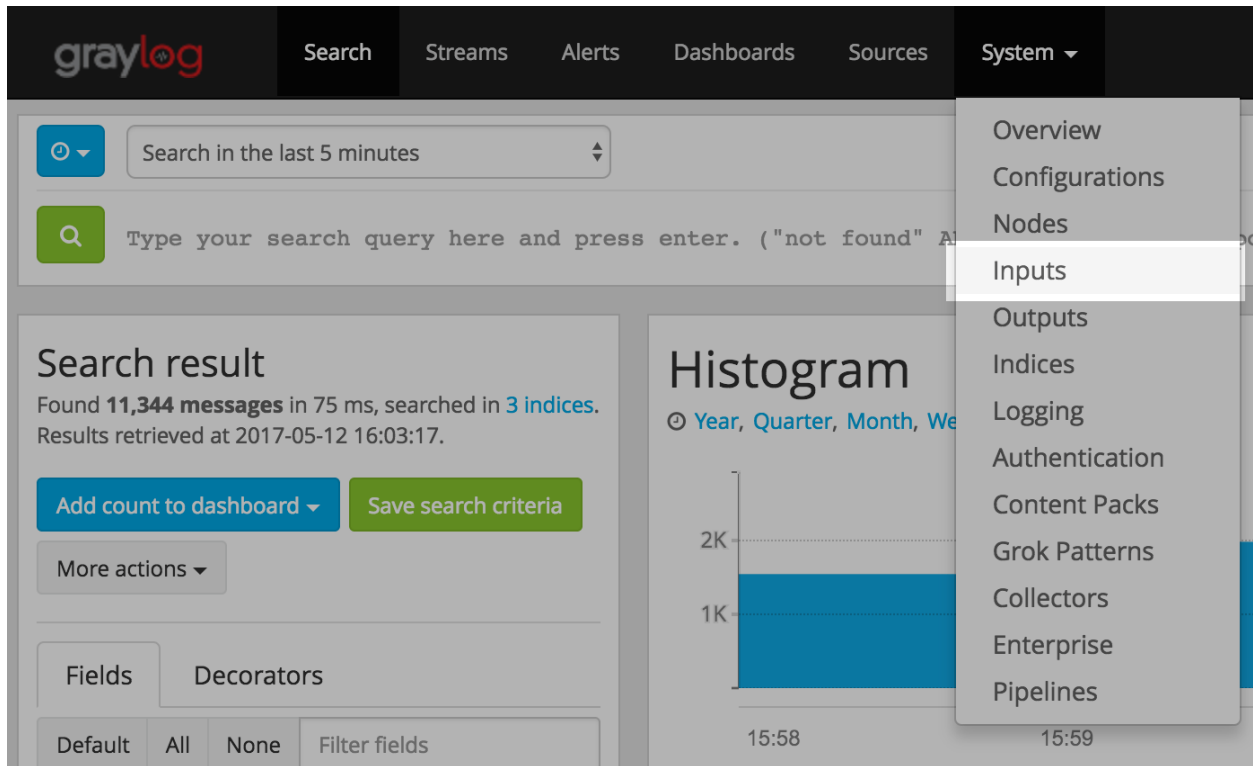
Launch a new *Raw/Plaintext UDP* input, listening on `127.0.0.1` on port `9099`. There's no need to configure anything else for now.

The list of running inputs on that node should show you your new input right away.

Let's send a message in:

```
echo "Hello Graylog, let's be friends." | nc -w 1 -u 127.0.0.1 9099
```

This has sent a short string to the raw UDP input you just opened. Now search for *friends* using the search bar on the top and you should already see the message you just sent in. Click on it in the table and see it in detail:
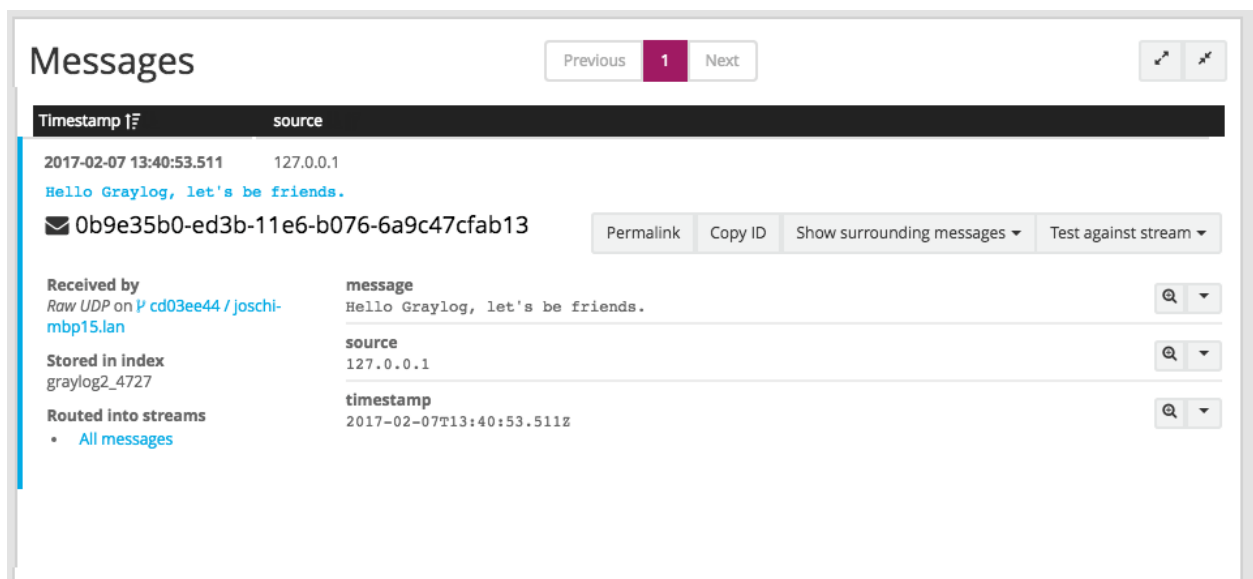
You have just sent your first message to Graylog! Why not spawn a syslog input and point some of your servers to it? You could also create some user accounts for your colleagues.

3.8 System requirements

The Graylog server application has the following prerequisites:

- Some modern Linux distribution (Debian Linux, Ubuntu Linux, or CentOS recommended)
- Elasticsearch 5 or 6
- MongoDB 3.6 or 4.0
- Oracle Java SE 8 (OpenJDK 8 also works; latest stable update is recommended)

Caution: Graylog prior to 2.3 **does not** work with Elasticsearch 5.x!

Caution: Graylog 3.x **does not** work with Elasticsearch 7.x!

Hint: Graylog 3.x does include first compatibility with Java 11 and we welcome people that test this.

Hint: Graylog 3.x does work with MongoDB 4.2 when this runs in compatibility mode 4.0

UPGRADING GRAYLOG

When upgrading from a previous version of Graylog you follow the previous used installation method (ex. from image or package) using the new version numbers.

The following Upgrade notes should be read carefully before you start the upgrade process. Breaking changes and dependency upgrades are documented in those upgrade notes.

You should always follow minor versions when updating across multiple versions to make sure necessary migrations are run correctly. The upgrade notes are always written coming from the stable release before.

4.1 Upgrading to Graylog 2.0.x

4.1.1 Elasticsearch 2.x

The embedded Elasticsearch node being used by Graylog has been upgraded to Elasticsearch 2.x which includes some breaking changes. Graylog 2.x does not work with Elasticsearch 1.x anymore and cannot communicate with existing Elasticsearch 1.x clusters.

Please see Breaking changes in Elasticsearch 2.x for details.

The blog article Key points to be aware of when upgrading from Elasticsearch 1.x to 2.x also contains interesting information about the upgrade path from Elasticsearch 1.x to 2.x.

Multicast Discovery

Multicast discovery has been removed from Elasticsearch 2.x (although it is still provided as an Elasticsearch plugin for now).

To reflect this change, the `elasticsearch_discovery_zen_ping_unicast_hosts` now has to contain the address of at least one Elasticsearch node in the cluster which Graylog can connect to.

Default network host

The network interface which Elasticsearch binds to (`elasticsearch_network_host`) has been changed to localhost (i. e. `127.0.0.1` or `::1`); see Network changes/Bind to localhost.

If Elasticsearch is not running on the same machine, `elasticsearch_network_host` must be set to a host name or an IP address which can be accessed by the other Elasticsearch nodes in the cluster.

Index range types

Note: This step needs to be performed before the update to Elasticsearch 2.x!

Some Graylog versions stored meta information about indices in elasticsearch, alongside the messages themselves. Since Elasticsearch 2.0 having multiple types with conflicting mappings is no longer possible, which means that the `index_range` type must be removed before upgrading to Elasticsearch 2.x.

Find out if your setup is affected by running (replace `$elasticsearch` with the address of one of your Elasticsearch nodes) `curl -XGET $elasticsearch:9200/_all/_mapping/index_range; echo`

If the output is `{}` you are not affected and can skip this step.

Otherwise, you need to delete the `index_range` type, Graylog does not use it anymore.

As Graylog sets older indices to read-only, first we need to remove the write block on those indices. Since we'll be working with Elasticsearch's JSON output, we recommend installing the `jq` utility which should be available on all popular package managers or directly at [GitHub](#).

```
for i in `curl -s -XGET $elasticsearch:9200/_all/_mapping/index_range | jq -r "keys[]
↪"`; do
    echo -n "Updating index $i: "
    echo -n "curl -XPUT $elasticsearch:9200/$i/_settings -d '{\"index.blocks.read_
↪only\":false, \"index.blocks.write\":false}' : "
    curl -XPUT $elasticsearch:9200/$i/_settings -d '{"index.blocks.read_only":false,
↪"index.blocks.write":false}'
    echo
done
```

The output for each of the curl commands should be `{"acknowledged":true}`. Next we have to delete the `index_range` mapping. We can perform this via the next command.

Note: We strongly recommend to perform this on a single index before running this bulk command. This operation can be expensive to perform if you have a lot of affected indices.

```
for i in `curl -s -XGET $elasticsearch:9200/_all/_mapping/index_range | jq -r "keys[]
↪"`; do
    echo -n "Updating index $i: "
    curl -XDELETE $elasticsearch:9200/$i/index_range
    echo
done
```

It is not strictly necessary to set the indices back to read only, but if you prefer to do that, note the index names and commands during the first step and change the `false` into `true`.

Graylog Index Template

Graylog applies a custom index template to ensure that the indexed messages adhere to a specific schema.

Unfortunately the index template being used by Graylog 1.x is incompatible with Elasticsearch 2.x and has to be removed prior to upgrading.

In order to delete the index template the following curl command has to be issued against on of the Elasticsearch nodes:

```
curl -X DELETE http://localhost:9200/_template/graylog-internal
```

Graylog will automatically create the new index template on the next startup.

Dots in field names

One of the most important breaking changes in Elasticsearch 2.x is that field names may not contain dots anymore.

Using the Elasticsearch Migration Plugin might help to highlight some potential pitfalls if an existing Elasticsearch 1.x cluster should be upgraded to Elasticsearch 2.x.

4.1.2 MongoDB

Graylog 2.x requires MongoDB 2.4 or newer. We recommend using MongoDB 3.x and the WiredTiger storage engine.

When upgrading from MongoDB 2.0 or 2.2 to a supported version, make sure to read the Release Notes for the particular version.

4.1.3 Log4j 2 migration

Graylog switched its logging backend from Log4j 1.2 to Log4j 2.

Please refer to the Log4j Migration Guide for information on how to update your existing logging configuration.

4.1.4 Dead Letters feature removed

The Dead Letters feature, which stored messages that couldn't be indexed into Elasticsearch for various reasons, has been removed.

This feature has been disabled by default. If you have enabled the feature the configuration file, please check the `dead_letters_enabled` collection in MongoDB and remove it afterwards.

4.1.5 Removed configuration settings

Index Retention and Rotation Settings

In 2.0.0 the index rotation and retention settings have been moved from the Graylog server config file to the database and are now configurable via the web interface.

The old settings from the `graylog.conf` or `/etc/graylog/server/server.conf` will be migrated to the database.

> **Warning:** When you upgrade from a 1.x version and you modified any rotation/retention settings, please make sure you **KEEP** your old settings in the config file so the migration process will add your old settings to the database! Otherwise the retention process will use the default settings and might remove a lot of indices.

Overview

Some settings, which have been deprecated in previous versions, have finally been removed from the Graylog configuration file.

Table 1: Removed configuration settings

Setting name	Replacement
`mongodb_host`	`mongodb_uri`
`mongodb_port`	`mongodb_uri`
`mongodb_database`	`mongodb_uri`
`mongodb_useauth`	`mongodb_uri`
`mongodb_user`	`mongodb_uri`
`mongodb_password`	`mongodb_uri`
`elasticsearch_node_name`	`elasticsearch_node_name_prefix`
`collector_expiration_threshold`	(moved to collector plugin)
`collector_inactive_threshold`	(moved to collector plugin)
`rotation_strategy`	UI in web interface (System/Indices)
`retention_strategy`	UI in web interface (System/Indices)
`elasticsearch_max_docs_per_index`	UI in web interface (System/Indices)
`elasticsearch_max_size_per_index`	UI in web interface (System/Indices)
`elasticsearch_max_time_per_index`	UI in web interface (System/Indices)
`elasticsearch_max_number_of_indices`	UI in web interface (System/Indices)
`dead_letters_enabled`	None

4.1.6 Changed configuration defaults

For better consistency, the defaults of some configuration settings have been changed after the project has been renamed from *Graylog2* to *Graylog*.

Table 2: Configuration defaults

Setting name	Old default	New default
`elasticsearch_cluster_name`	`graylog2`	`graylog`
`elasticsearch_node_name`	`graylog2-server`	`graylog-server`
`elasticsearch_index_prefix`	`graylog2`	`graylog`
`elasticsearch_discovery_zen_ping_unicast_hosts`	empty	`127.0.0.1:9300`
`elasticsearch_discovery_zen_ping_multicast_enabled`	`true`	`false`
`mongodb_uri`	`mongodb://127.0.0.1/graylog2`	`mongodb://localhost/graylog`

4.1.7 Changed prefixes for configuration override

In the past it was possible to override configuration settings in Graylog using environment variables or Java system properties with a specific prefix.

For better consistency, these prefixes have been changed after the project has been renamed from *Graylog2* to *Graylog*.

Table 3: Configuration override prefixes

Override	Old prefix	New prefix	Example
Environment variables	`GRAYLOG2_`	`GRAYLOG_`	`GRAYLOG_IS_MASTER`
System properties	`graylog2.`	`graylog.`	`graylog.is_master`

4.1.8 REST API Changes

The output ID key for the list of outputs in the `/streams/*` endpoints has been changed from `_id` to `id`.

```
{
  "id": "564f47c41ec8fe7d920ef561",
  "creator_user_id": "admin",
  "outputs": [
    {
      "id": "56d6f2cce45e0e52d1e4b9cb", // ==> Changed from `_id` to `id`
      "title": "GELF Output",
      "type": "org.graylog2.outputs.GelfOutput",
      "creator_user_id": "admin",
      "created_at": "2016-03-02T14:03:56.686Z",
      "configuration": {
        "hostname": "127.0.0.1",
        "protocol": "TCP",
        "connect_timeout": 1000,
        "reconnect_delay": 500,
        "port": 12202,
        "tcp_no_delay": false,
        "tcp_keep_alive": false,
        "tls_trust_cert_chain": "",
        "tls_verification_enabled": false
      },
      "content_pack": null
    }
  ],
  "matching_type": "AND",
  "description": "All incoming messages",
  "created_at": "2015-11-20T16:18:12.416Z",
  "disabled": false,
  "rules": [],
  "alert_conditions": [],
  "title": "ALL",
  "content_pack": null
}
```

4.1.9 Web Interface Config Changes

The web interface has been integrated into the Graylog server and was rewritten in React. Therefore configuring it has changed fundamentally since the last version(s). Please consult *Web interface* for details.

Please take note that the `application.context` configuration parameter present in Graylog 1.x (and earlier) is not existing anymore. The web interface can currently only be served without a path prefix.

4.2 Upgrading to Graylog 2.1.x

4.2.1 HTTPS Setup

Previous versions of Graylog were automatically generating a private key/certificate pair for HTTPS if either the private key or the certificate (or both) for `rest_tls_key_file`, `rest_tls_cert_file`, `web_tls_key_file`, or `web_tls_cert_file` couldn't be read. While this feature is very comfortable for inexperienced users, it has lots of serious drawbacks like very weak key sizes (only 1024 bits), being untrusted by all TLS libraries used by web browsers and other client software (because they are self-signed and not included in the system's CA/trust store), and problems with inter-node communications with other Graylog nodes.

Due to those shortcomings, the feature has been removed completely. Users need to use proper certificates or generate their own self-signed certificates and configure them with the appropriate settings, see *Using HTTPS* for reference.

4.2.2 Web Interface Listener

Graylog 2.0.x has been using separate listeners for the REST API and the web interface by default. The Graylog REST API on `http://127.0.0.1:12900`, the Graylog web interface on `http://127.0.0.1:9000`. Beginning with Graylog 2.1.0 it is possible to run both the REST API and the web interface on the same host/port-combination and this is now the default. This means that the REST API is now running on `http://127.0.0.1:9000/api/` by default and the web interface is now running on `http://127.0.0.1:9000/`. Furthermore, all requests going to `http://127.0.0.1:9000/api/` requesting a content-type of `text/html` or `application/xhtml+xml` are redirected to the web interface, therefore making it even easier to set up Graylog and use it behind proxies, expose it externally etc.

Please take note that you can still run the REST API and the web interface on two separate listeners. If you are running a Graylog 2.0.x configuration specifying `web_listen_uri` explicitly and you want to keep that, you do not have to change anything.

Please also take note, that when you have configured `rest_listen_uri` and `web_listen_uri` to run on the same host/port-combination, the following configuration directives will have no effect:

- `web_enable_tls`, `web_tls_cert_file`, `web_tls_key_file`, `web_tls_key_password` (These will depend on the TLS configuration of the REST listener).

- `web_enable_cors`, `web_enable_gzip`, `web_thread_pool_size`, `web_max_initial_line_length`, `web_max_header_size` (Those will depend on the corresponding settings of the REST listener).

4.2.3 Internal Metrics to MongoDB

Previous versions of Graylog included a (long deprecated) metrics reporter for writing internal metrics into MongoDB in a fixed interval of 1 second.

This feature has been removed completely and can be optionally pulled in by using the Graylog Metrics Reporter Plugins.

4.2.4 Configuration file changes

Network settings

The network settings in the Graylog configuration file (`rest_listen_uri`, `rest_transport_uri`, and `web_listen_uri`) are now using the default ports for the HTTP (80) and HTTPS (443) if no custom port was given. Previously those settings were using the custom ports 12900 (Graylog REST API) and 9000 (Graylog web interface) if no explicit port was given.

Examples:

Configuration setting	Old effective URI	New effective URI
`rest_listen_uri = http://127.0.0.1:12900/`	`http://127.0.0.1:12900/`	`http://127.0.0.1:12900/`
`rest_listen_uri = http://127.0.0.1/`	`http://127.0.0.1:12900/`	`http://127.0.0.1:80/`
`rest_listen_uri = https://127.0.0.1/`	`https://127.0.0.1:12900/`	`https://127.0.0.1:443/`

Collector Sidecar

The network changes are reflected in the Sidecar configuration as well and should be adopted. However it's still possible to use the old API port by setting it explicitly. In case a mass deployment is too hard to change, just run the following to switch back to the old REST API port (OVA based installation):

```
sudo graylog-ctl set-listen-address --service rest --address http://0.0.0.0:12900
sudo graylog-ctl reconfigure
```

4.2.5 Graylog REST API

Removed resources

Original resource	Replacement
`/system/buffers`	`/system/metrics/org.graylog2.buffers.input.size`
	`/system/metrics/org.graylog2.buffers.input.usage`
	`/system/metrics/org.graylog2.buffers.process.size`
	`/system/metrics/org.graylog2.buffers.process.usage`
	`/system/metrics/org.graylog2.buffers.output.size`
	`/system/metrics/org.graylog2.buffers.output.usage`
`/system/buffers/classes`	None

Removed index rotation/retention settings from "/system/configuration"

The index rotation and retention settings have been moved to MongoDB in Graylog 2.0.0 but the representation of the old configuration options was still present in the `/system/configuration` resource.

In order to stay in sync with the actual configuration file, the following values have been removed:

- `rotation_strategy`
- `retention_strategy`
- `elasticsearch_max_docs_per_index`
- `elasticsearch_max_size_per_index`
- `elasticsearch_max_time_per_index`
- `elasticsearch_max_number_of_indices`

The retention and rotation configuration settings can be retrieved using the following resources:

- `/system/indices/rotation/config`
- `/system/indices/retention/config`

4.2.6 For Plugin Authors

Between Graylog 2.0.x and 2.1.0 we also made changes to the Plugin API. These include:

- Removing `org.graylog2.plugin.streams.Stream#getAlertCondition`, as it was faulty and not easily replaceable with a working version without breaking our separation of models and persistence services.

If you are maintaining a plugin that was originally written for Graylog 1.x or 2.0.x, you need to make sure that your plugin is still compiling and working under Graylog 2.1.x or adapt it if necessary.

UI Plugins

The new app prefix feature requires some changes in UI plugins to make them work with that.

- `import webpackEntry from 'webpack-entry';` needs to be added at the very top of the `src/web/index.jsx` file
- The `Routes.pluginRoute()` function needs to be used instead of a literal string to build URLs for links and buttons

Please check the *updated plugins documentation* for details.

4.2.7 Changed Elasticsearch Cluster Status Behavior

In previous versions Graylog stopped indexing into the current write index if the *Elasticsearch cluster status* turned RED. Since Graylog 2.1.0 only checks the status of the current write index when it tries to index messages.

If the current write index is GREEN or YELLOW, Graylog will continue to index messages even though the overall cluster status is RED. This avoids Graylog downtimes when doing Elasticsearch maintenance or when older indices have problems.

4.2.8 Changes in message field values trimming

Previous versions of Graylog were trimming message field values inconsistently, depending on the codec used. We have changed that behaviour in Graylog 2.1.0, so all message field values are trimmed by default. This means that leading or trailing whitespace of every field is removed during ingestion.

Important: This change will break your existing stream rules, extractors, and Drool rules if you are expecting leading or trailing white spaces in them. Please adapt them so they do not require those white spaces.

4.3 Upgrading to Graylog 2.2.x

4.3.1 Email Alarm Callback

Previous versions of Graylog created an implicit email alarm callback if no explicit callback existed for a stream.

Due to the extensive rework done in alerting, this behavior has been modified to be explicit, and more consistent with other entities within Graylog: from now on **there will not be a default alarm callback**.

To simplify the transition for people relying on this behavior, we have added a migration step that will create an email alarm callback for each stream that has alert conditions, has alert receivers, but has no associated alarm callbacks.

With to the introduction of email templates in 0.21, the `transport_email_subject_prefix` config setting became unused. It is now being removed completely. In early versions it was used to add a prefix to the generated subject of alerting emails. Since 0.21 it is possible to define a complete template used for the generation of alert email subjects.

4.3.2 Alert Notifications (previously known as Alarm Callbacks)

Graylog 2.2.0 introduces some changes in alerting. Alerts have now states, helping you to know in an easier way if something requires your attention.

These changes also affect the way we send notifications: Starting in Graylog 2.2.0, alert notifications are only executed **once**, just when a new alert is triggered. As long as the alert is unresolved or in grace period, **Graylog will not send further notifications**. This will help you reducing the noise and annoyance of getting notified way too often when a problem persists for a while.

If you are using Graylog for alerting, please take a moment to ensure this change will not break any of your processes when an alert occurs.

4.3.3 Default stream/Index Sets

With the introduction of index sets, and the ability to change a stream's write target, the default stream needs additional information, which is calculated when starting a new Graylog 2.2 master node.

It requires recalculation of the index ranges of the default stream's index set, which when updating from pre-2.2 versions is stored in the `graylog_` index. This is potentially expensive, because it has to calculate three aggregations across every open index to detect which streams are stored in which index.

Please be advised that this necessary migration can put additional load on your cluster.

> **Warning:** Make sure that all rotation and retention strategy plugins you had installed in 2.1 are updated to a version that is compatible with 2.2 before you start the Graylog 2.2 version for the first time. (e.g. Graylog Enterprise) This is needed so the required data migrations will run without problems.

> **Warning:** The option to remove a message from the default stream is currently not available when using the pipeline function *route_to_stream*. This will be fixed in a subsequent bug fix release. Please see the corresponding Github issue.

4.3.4 RotationStrategy & RetentionStrategy Interfaces

The Java interfaces for `RetentionStrategy` and `RotationStrategy` changed in 2.2. The `#rotate()` and `#retain()` methods are now getting an `IndexSet` as first parameter.

This only affects you if you are using custom rotation or retention strategies.

4.3.5 Changes in Exposed Configuration

The exposed configuration settings on the `/system/configuration` resource of the Graylog REST API doesn't contain the following (deprecated) Elasticsearch-related settings anymore:

- `elasticsearch_shards`
- `elasticsearch_replicas`
- `index_optimization_max_num_segments`
- `disable_index_optimization`

4.3.6 Changes in Split & Count Converter

The behavior of the split & count converter has been changed to that it resembles typical `split()` functions.

Previously, the split & count converter returned 0, if the split pattern didn't occur in the string. Now it will return 1.

Examples:

String	Split Pattern	Old Result	New Result
<empty>	–	0	0
foo	–	0	1
foo-bar	–	2	2

4.3.7 Graylog REST API

Streams API

Due to the introduction of index sets, the payload for creating, updating and cloning of streams now requires the `index_set_id` field. The value for this needs to be the ID of an existing index set.

Affected endpoints:

- `POST /streams`
- `PUT /streams/{streamId}`
- `POST /streams/{streamId}/clone`

4.4 Upgrading to Graylog 2.3.x

4.4.1 Graylog switches to Elasticsearch HTTP client

In all prior versions, Graylog used the Elasticsearch node client to connect to an Elasticsearch cluster, which was acting as a client-only Elasticsearch node. For compatibility reasons of the used binary transfer protocol, the range of Elasticsearch versions Graylog could connect to was limited. For more information and differences between the different ways to connect to Elasticsearch, you can check the Elasticsearch documentation.

Starting with version 2.3.0, we are switching over to using a lightweight HTTP client, which is almost version-agnostic. The biggest change is that it does not connect to the Elasticsearch native protocol port (defaulting to 9300/tcp), but the Elasticsearch HTTP port (defaulting to 9200/tcp).

Due to the differences in connecting to the Elasticsearch cluster, configuring Graylog has changed. These configuration settings have been removed:

```
elasticsearch_cluster_discovery_timeout
elasticsearch_cluster_name
elasticsearch_config_file
elasticsearch_discovery_initial_state_timeout
elasticsearch_discovery_zen_ping_unicast_hosts
elasticsearch_http_enabled
elasticsearch_network_bind_host
elasticsearch_network_host
elasticsearch_network_publish_host
elasticsearch_node_data
elasticsearch_node_master
elasticsearch_node_name_prefix
elasticsearch_path_data
elasticsearch_path_home
elasticsearch_transport_tcp_port
```

The following configuration options are now being used to configure connectivity to Elasticsearch:

Config Setting	Type	Comments	Default
`elasticsearch_connect_timeout`	Duration	Timeout when connection to individual Elasticsearch hosts	`10s` (10 Seconds)
`elasticsearch_hosts`	List<URIs>	Comma-separated list of URIs of Elasticsearch hosts	`http://127.0.0.1:9200`
`elasticsearch_idle_timeout`	Duration	Timeout after which idle connections are terminated	`-1s` (Never)
`elasticsearch_max_total_connections`	int	Maximum number of total Elasticsearch connections	`20`
`elasticsearch_max_total_connections_per_route`	int	Maximum number of Elasticsearch connections per route/host	`2`
`elasticsearch_max_retries`	int	Maximum number of retries for requests to Elasticsearch	`2`
`elasticsearch_socket_timeout`	Duration	Timeout when sending/receiving from Elasticsearch connection	`60s` (60 Seconds)
`elasticsearch_discovery_enabled`	boolean	Enable automatic Elasticsearch node discovery	`false`
`elasticsearch_discovery_filter`	String	Filter by node attributes for the discovered nodes	empty (use all nodes)
`elasticsearch_discovery_frequency`	Duration	Frequency of the Elasticsearch node discovery	`30s` (30 Seconds)

In most cases, the only configuration setting that needs to be set explicitly is `elasticsearch_hosts`. All other configuration settings should be tweaked only in case of errors.

> **Warning:** The automatic node discovery does not work if Elasticsearch requires authentication, e. g. when using Shield (X-Pack).

> **Caution:** Graylog does not react to externally triggered index changes (creating/closing/reopening/deleting an index) anymore. All of these actions need to be performed through the Graylog REST API in order to retain index consistency.

Special note for upgrading from an existing Graylog setup with a new Elasticsearch cluster

If you are upgrading the Elasticsearch cluster of an existing Graylog setup without migrating the indices, your Graylog setup contains stale index ranges causing nonexisting index errors upon search/alerting. To remediate this, you need to manually trigger an index range recalculation for all index sets once. This is possible using the web interface using the System->Indices functionality or by using the REST API using the `/system/indices/ranges/<index set id>/rebuild` endpoint.

4.4.2 Graylog REST API

Rotation and Retention strategies

The deprecated HTTP resources at `/system/indices/rotation/config` and `/system/indices/retention/config`, which didn't work since Graylog 2.2.0, have been removed.

These settings are part of the index set configuration and can be configured under `/system/indices/index_sets`.

Stream List Response structure does not include *in_grace* field anymore

The response to `GET /streams`, `GET /streams/<id>` & `PUT /streams/<id>` does not contain the `in_grace` field for configured alert conditions anymore.

The value of this flag can be retrieved using the `GET /alerts/conditions` endpoint, or per stream using the `GET /streams/<streamId>/alerts/conditions` endpoint.

4.5 Upgrading to Graylog 2.4.x

You can upgrade from Graylog 2.3.x to Graylog 2.4.x without the need to change the configuration of your Graylog server.

4.5.1 More plugins shipped by default

The following Graylog plugins are now shipped as part of the Graylog server release.

- AWS Plugin - https://github.com/Graylog2/graylog-plugin-aws
- Threat Intelligence Plugin - https://github.com/Graylog2/graylog-plugin-threatintel
- NetFlow Plugin - https://github.com/Graylog2/graylog-plugin-netflow
- CEF Plugin - https://github.com/Graylog2/graylog-plugin-cef

> **Warning:** Make sure you remove all previous versions of these plugins from your `plugin/` folder before starting the new Graylog version!

4.6 Upgrading to Graylog 2.5.x

4.6.1 Protecting against CSRF, HTTP header required

Using the Graylog server API requires all clients sending non-GET requests to include a custom HTTP header (`X-Requested-By`). The value of the header is not important, but it's presence is, as all requests without it will be ignored and will return a 400 error.

This is important for people using scripts that modify Graylog in any way through the REST API. We already adapted Graylog web interface and our plugins, so if you don't use any scripts or 3rd party products to access Graylog, you don't have to do anything else.

If you are using the Graylog Sidecar, you either have to use Graylog version 2.5.1 or update the Sidecar to version 0.1.7. That version is using the correct CSRF headers for HTTP requests against the Graylog server API.

4.6.2 Elasticsearch 6 changes

There is a breaking change in Elasticsearch 6 that may affect some queries on your searches and dashboards:

Before Elasticsearch 6, queries for keyword fields were split by whitespaces and combined with `OR` operators resulting, for example, in `type:(ssh login)` and `type:(ssh OR login)` being equivalent. This is no longer the case in Elasticsearch 6 and now those queries are different: the former looking for the `ssh login` value, the second for either `ssh` or `login` values.

Please ensure to look for those queries in your Graylog setup before upgrading to Elasticsearch 6 and add the `OR` operators where needed.

4.7 Upgrading to Graylog 3.0.x

4.7.1 Elasticsearch Version Requirements

Graylog 3.0 drops support for Elasticsearch versions before 5.6.x. That means you have to upgrade Elasticsearch to at least version 5.6.13 before upgrading Graylog to version 3.0. Make sure to read the Elasticsearch upgrade guides before doing that.

4.7.2 Simplified HTTP interface configuration

Graylog used to have a lot of different settings regarding the various HTTP interfaces it provides, namely the Graylog REST API and the Graylog web interface.

This mostly originates from the fact that Graylog used to consist of two components before Graylog 2.0.0, a server component and a separate web interface.

The changes in this release finally merge the HTTP listeners for the Graylog REST API and web interface into a single HTTP listener, which should make the initial configuration of Graylog simpler and reduce errors caused by conflicting settings.

The path of the Graylog REST API is now hard-coded to `/api`, so if you're still using the legacy URI on port 12900/tcp or have been using a custom path (via the `rest_listen_uri` or `rest_transport_uri` settings), you'll have to update the URI used to access the Graylog REST API.

This might also affect your **Graylog Collector Sidecars**. Make sure to check each `collector_sidecar.yml` and update the `server_url` accordingly.

If you are using a reverse proxy in front of Graylog (like nginx) and configured it to set the `X-Graylog-Server-URL` HTTP header, you have to remove the `api/` suffix because that is now the default. (as mentioned above)

Example:

```
# This nginx setting in Graylog <3.0 ...
header_upstream X-Graylog-Server-URL http://{host}/api

# ... needs to be changed to the following with Graylog 3.0
header_upstream X-Graylog-Server-URL http://{host}/
```

For a more detailed description of the new HTTP settings, please consult the annotated Graylog configuration file.

Overview of removed Graylog REST API settings:

Removed Setting	New Setting	Default
`rest_listen_uri`	`http_bind_address`	`127.0.0.1:9000`
`rest_transport_uri`	`http_publish_uri`	`http://` `$http_bind_address/`
`web_endpoint_uri`	`http_external_uri`	`$http_publish_uri`
`rest_enable_cors`	`http_enable_cors`	`true`
`rest_enable_gzip`	`http_enable_gzip`	`true`
`rest_max_header_size`	`http_max_header_size`	`8192`
`rest_max_initial_line_length`	`http_max_initial_line_length`	`4096`
`rest_thread_pool_size`	`http_thread_pool_size`	`16`
`rest_enable_tls`	`http_enable_tls`	`false`
`rest_tls_cert_file`	`http_tls_cert_file`	Empty
`rest_tls_key_file`	`http_tls_key_file`	Empty
`rest_tls_key_password`	`http_tls_key_password`	Empty

Overview of removed Graylog web interface settings:

Removed Setting	New Setting	Default
`web_enable`	None	
`web_listen_uri`	`http_bind_address`	`127.0.0.1:9000`
`web_enable_cors`	`http_enable_cors`	`true`
`web_enable_gzip`	`http_enable_gzip`	`true`
`web_max_header_size`	`http_max_header_size`	`8192`
`web_max_initial_line_length`	`http_max_initial_line_length`	`4096`
`web_thread_pool_size`	`http_thread_pool_size`	`16`
`web_enable_tls`	`http_enable_tls`	`false`
`web_tls_cert_file`	`http_tls_cert_file`	Empty
`web_tls_key_file`	`http_tls_key_file`	Empty
`web_tls_key_password`	`http_tls_key_password`	Empty

4.7.3 Plugins merged into the Graylog server

Starting with Graylog 3.0.0, the following official plugins were merged into the Graylog server:

- Beats Input

- CEF Input

- Collector Plugin

- Enterprise Integration Page

- Map Widget

- NetFlow Input

- Pipeline Processor

That means these plugins are not available as separate plugins anymore. If you manually update your Graylog installation (without using operating system packages), make sure to remove all old plugin files from the plugin_dir folder.

The old issues in these repositories are still available for reference but new issues should only be created in the Graylog server issue tracker.

The following HTTP API paths changed due to the plugin merge:

Old Path	New Path
`/plugins/org.graylog.plugins.map/mapdata`	`/search/mapdata`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/pipeline`	`/system/pipelines/` `pipeline`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/pipeline/parse`	`/system/pipelines/` `pipeline/parse`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/rule`	`/system/pipelines/rule`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/rule/functions`	`/system/pipelines/rule/` `functions`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/rule/multiple`	`/system/pipelines/rule/` `multiple`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/rule/parse`	`/system/pipelines/rule/` `parse`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/connections`	`/system/pipelines/` `connections`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/connections/to_stream`	`/system/pipelines/` `connections/to_stream`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/connections/to_pipeline`	`/system/pipelines/` `connections/to_pipeline`
`/plugins/org.graylog.plugins.pipelineprocessor/` `system/pipelines/simulate`	`/system/pipelines/` `simulate`

4.7.4 New "bin_dir" and "data_dir" configuration parameters

We introduced two new configuration parameters related to file system paths.

- `bin_dir` config option points to the directory that contains scripts like `graylogctl`.
- `data_dir` option configures the base directory for Graylog server state.

Please check the updated default `graylog.conf` configuration file for required changes to your existing file.

4.7.5 Removed support for Drools-based filters

For a long time, Graylog allowed to use Drools to filter messages. Unfortunately, using Drools to perform complex filter logic came with a performance penalty and wasn't as flexible as we would have liked it to be.

Starting with Graylog 3.0.0, the support for Drools-based message filters has been removed from Graylog. The `rules_file` configuration setting has been removed accordingly.

We recommend migrating the Drools-based logic to Processing Pipelines.

Drools-based blacklist

Graylog provided undocumented blacklist-functionality based on Drools. This blacklist could only be modified via the Graylog REST API on the `/filters/blacklist` resource.

If you've been using this functionality, you'll have to migrate these blacklist rules to the Processing Pipelines.

To check if you're using the Drools-based blacklist in Graylog prior to version 3.0.0, you can run the following command:

```
# curl -u admin:password -H 'Accept: application/json' 'http://graylog.example.com/
→api/filters/blacklist?pretty=true'
```

String-based blacklist rule

Old blacklist rule:

```
{
    "id" : "54e30000123412341234001",
    "type" : "string",
    "name" : "String Blacklist",
    "description" : "Drop messages based on case-insensitive string comparison",
    "fieldName" : "custom_field",
    "pattern" : "EXAMPLE pattern",
    "creator_user_id" : "admin",
    "created_at" : "2018-04-04T12:00:00.000Z"
}
```

New pipeline rule:

```
rule "string-blacklist"
when
  has_field("custom_field") &&
  lowercase(to_string($message.custom_field)) == "example pattern"
then
  drop_message();
end
```

See also:

- has_field()

- lowercase()

- drop_message()

Regex-based blacklist rule

Old blacklist rule:

```
{
    "id" : "54e30000123412341234002",
    "type" : "regex",
    "name" : "Regex Blacklist",
    "description" : "Drop messages based on regular expression",
    "fieldName" : "custom_field",
```

(continues on next page)

```
  "pattern" : "^EXAMPLE.*",
  "creator_user_id" : "admin",
  "created_at" : "2018-04-04T12:00:00.000Z"
}
```

New pipeline rule:

```
rule "regex-blacklist"
when
  has_field("custom_field") &&
  regex("^EXAMPLE.*", to_string($message.custom_field)).matches == true
then
  drop_message();
end
```

See also:

- has_field()

- regex()

- drop_message()

IP Range-based blacklist rule

Old blacklist rule:

```
{
  "id" : "54e30001234123412340003",
  "type" : "iprange",
  "name" : "IP Blacklist",
  "description" : "Drop messages based on IP address",
  "fieldName" : "custom_field",
  "pattern" : "192.168.0.0/16",
  "creator_user_id" : "admin",
  "created_at" : "2018-04-04T12:00:00.000Z"
}
```

New pipeline rule:

```
rule "ip-blacklist"
when
  has_field("custom_field") &&
  cidr_match("192.168.0.0/16", to_ip($message.custom_field))
then
  drop_message();
end
```

See also:

- has_field()

- to_ip()

- cidr_match()

- drop_message()

4.7.6 Changed metrics name for stream rules

The name of the metrics for stream rules have been changed to include the stream ID which helps identifying the actual stream they are related to.

Old metric name:

```
org.graylog2.plugin.streams.StreamRule.${stream-rule-id}.executionTime
```

New metric name:

```
org.graylog2.plugin.streams.Stream.${stream-id}.StreamRule.${stream-rule-id}.
↪executionTime
```

4.7.7 Email alarm callback default settings

The defaults of the configuration settings for the email alarm callback with regard to encrypted connections have been changed.

Setting	Old default	New default
transport_email_use_tls	false	true
transport_email_use_ssl	true	false

Furthermore, it's not possible anymore to enable both settings (SMTP with STARTTLS and SMTP over SSL) at the same time because this led to errors at runtime when Graylog tried to upgrade the connection to TLS with STARTTLS in an already existing SMTPS connection.

Most SMTP services prefer SMTP with STARTTLS to provide an encrypted connection.

4.7.8 Collector Sidecar is deprecated

Graylog 3.0 comes with a new Sidecar implementation. We still support the old **Collector Sidecars**, which can be found in the `System / Collectors (legacy)` menu entry. For more information check the *Sidecar documentation* and the *Upgrade guide*.

4.7.9 Legacy Content Packs

The implementation of content packs where fundamentally reworked. Parameters were added and checks implemented to give the user better usability. This rework did come with the cost that old content packs might not work any longer and stop the new content packs from loading. If the content packs page does not finish loading we recommend to remove the old content packs from your MongoDB. For that, please connect to your MongoDB shell and remove all content packs with the following command:

```
> db.content_packs.deleteMany({})
```

This command will only remove the content packs, it will not remove the installed configurations.

4.7.10 Elasticsearch 6 changes

There is a breaking change in Elasticsearch 6 that may affect some queries on your searches and dashboards:

Before Elasticsearch 6, queries for keyword fields were split by whitespaces and combined with `OR` operators resulting, for example, in `type:(ssh login)` and `type:(ssh OR login)` being equivalent. This is no longer the case in Elasticsearch 6 and now those queries are different: the former looking for the `ssh login` value, the second for either `ssh` or `login` values.

Please ensure to look for those queries in your Graylog setup before upgrading to Elasticsearch 6 and add the `OR` operators where needed.

4.8 Upgrading to Graylog 3.1.x

Overview

- *Upgrading to Graylog 3.1.x*
    - *Views & Extended Search*
        - *HTTP API Changes*
        - *Configuration File Changes*
    - *Alerts*
        - *Alarm Callback Plugins*
        - *Alert Condition Plugins*
        - *Deprecated HTTP API Endpoints*
        - *Configuration File Changes*

4.8.1 Views & Extended Search

The Views and Extended Search feature has been open-sourced in this version. (except the support for parameters) It was only accessible in Graylog Enterprise in Graylog 3.0.

HTTP API Changes

The following Views related HTTP API paths changed due to the migration to open source:

Old Path	New Path
`/api/plugins/org.graylog.plugins.enterprise/views`	`/api/views`
`/api/plugins/org.graylog.plugins.enterprise/search`	`/api/views/search`

Configuration File Changes

The following views related configuration file settings changed due to the migration to open source:

Removed Setting	New Setting
`enterprise_search_maximum_search_age`	`views_maximum_search_age`

4.8.2 Alerts

The old Alerts system has been replaced by an Alerts & Events system in Graylog 3.1.

Existing alerts will be automatically migrated to the new system when the Graylog 3.1 server starts for the first time. The migration will log the number of migrated legacy alert conditions and alarm callbacks:

```
2019-08-05T10:36:06.404Z INFO  [V20190722150700_LegacyAlertConditionMigration]␣
→Migrated <2> legacy alert conditions and <2> legacy alarm callbacks
```

Alarm Callback Plugins

The new Alerts & Events system is supporting the execution of legacy Alarm Callback plugins for now. We recommend to switch event definitions over to event notifications, though. At some point in the future support for legacy Alarm Callback plugins will be removed. More information for plugin developers can be found on the *Event Notifications* page.

Note: Please note, that the data sent via a legacy Alarm Callback might be slightly different than via the old Alerts system. If you've built automation on top of alerts, you might want to check that everything still works after the migration.

Alert Condition Plugins

The new Alerts & Events system is *not* supporting the execution of legacy Alert Condition plugins. Old alerts using the internal alert conditions are automatically migrated to new event definitions. (message count, field value and field content value conditions) Custom alert condition plugins cannot be migrated and need to be rewritten as event definitions in the new system.

Deprecated HTTP API Endpoints

The following HTTP API endpoints are deprecated and will be removed in a future release:

Deprecated API Endpoints
`/api/streams/{streamid}/alerts/{alertId}/history`
`/api/streams/{streamid}/alerts/{alertId}/history`
`/api/alerts/callbacks`
`/api/alerts/callbacks/types`
`/api/alerts/callbacks/{alarmCallbackId}/test`
`/api/alerts/conditions`
`/api/alerts/conditions/types`
`/api/streams/alerts`
`/api/streams/alerts/paginated`
`/api/streams/alerts/{alertId}`
`/api/streams/{streamid}/alarmcallbacks`
`/api/streams/{streamid}/alarmcallbacks/available`
`/api/streams/{streamid}/alarmcallbacks/alarmCallbackId`
`/api/streams/{streamid}/alerts/conditions`
`/api/streams/{streamid}/alerts/conditions/test`
`/api/streams/{streamid}/alerts/conditions/{conditionId}`
`/api/streams/{streamid}/alerts/conditions/{conditionId}/test`
`/api/streams/{streamid}/alerts`
`/api/streams/{streamid}/alerts/check`
`/api/streams/{streamid}/alerts/paginated`
`/api/streams/{streamid}/alerts/receivers`
`/api/streams/{streamid}/alerts/sendDummyAlert`

The deprecated API endpoints don't have a one to one mapping to new ones, but the following Endpoints can be used to manage Event Definitions and Event Notifications as well as search for events:

New Events API Endpoints
`/api/events/definitions`
`/api/events/entity_types`
`/api/events/notifications`
`/api/events/search`

Configuration File Changes

The following alerting related configuration file settings changed in this release:

Setting	Status	Description
`alert_check_interval`	re-moved	Was part of the old and now removed alerts system.
`processing_status_persist_interval`	added	Related to the new alerts system.
`processing_status_update_threshold`	added	Related to the new alerts system.
`processing_status_journal_write_rate_threshold`	added	Related to the new alerts system.
`default_events_index_prefix`	added	Related to the new alerts system.
`default_system_events_index_prefix`	added	Related to the new alerts system.

See *server configuration page* for details on the new settings.

4.9 Upgrading Graylog Originally Installed from Image

2.x It is not possible to upgrade previous OVAs to Graylog 3.0.0.

3.x Starting with Graylog 3.0.0, OVAs use the Operating System packages, so you can upgrade your appliance by following *this update guide*.

4.10 Upgrading Graylog Originally Installed from Package

If the current installation was installed using a package manager (ex. yum, apt), update the repository package to the target version, and use the system tools to upgrade the package. For .rpm based systems *this update guide* and for .deb based systems *this update guide* should help.

4.11 Upgrading Elasticsearch

Since Graylog 2.3 Elasticsearch 5.x is supported. This Graylog version supports Elasticsearch 2.x and 5.x. It is recommended to update Elasticsearch 2.x to the latest stable 5.x version, after you have Graylog 2.3 or later running. This Elasticsearch upgrade does not need to be made during the Graylog update.

When upgrading from Elasticsearch 2.x to Elasticsearch 5.x, make sure to read the upgrade guide provided by Elastic. The Graylog *Elasticsearch configuration documentation* contains information about the compatible Elasticsearch version. After the upgrade you must *rotate the indices once manually*.

Graylog 2.5 is the first Graylog version that supports Elasticsearch 6, the upgrade is recommended as soon as possible but might need more attention and include the need to reindex your data. Make sure to check *our Elasticsearch 6 upgrade notes* for this and other requirements.

When upgrading from Elasticsearch 5.x to Elasticsearch 6.x, make sure to read the upgrade guide provided by Elastic. The Graylog *Elasticsearch configuration documentation* contains information about the compatible Elasticsearch version. After the upgrade you must *rotate the indices once manually*.

4.11.1 Elasticsearch 6 notes

Upgrades from Elasticsearch 2.x direct to the latest Elasticsearch 6.x are not supported. Only the upgrade from Elasticsearch 5.x is supported and covered by this document.

At first check if the data in Elasticsearch need to be re-indexed:

```
$ http :9200/_settings | \
 jq '[ path(.[] |select(.settings.index.version.created < "5000000"))[] ]'
```

The above example use the tools httpie and jq to query the Elasticsearch API and check if any indices are created with Elasticsearch prior to Version 5. If that returns any index names, you need to re-index your data to make them work with Elasticseach 6.

Upgrading to Elasticsearch 6 is always a full-cluster-restart and all breaking changes need to checked carefully. Once started their is no going back or downgrade possible.

The Elasticsearch breaking changes notes contain a complete list of changes in Elasticsearch that should be checked against your configuration. The most notable is the cluster name is no longer allowed in *path.data* (see breaking changes in Elasticsearch 6) and the release of Elasticseach OSS Packages.

Upgrade without re-index

When no re-index is needed the easiest way is to follow the elastic upgrade guide for Elasticsearch this gives all needed commands.

Upgrade with re-index

First a brief overview what steps need to be performed followed by the list of commands. Once you started the process of reindex your data you need to finish all steps to get a working Graylog and Elasticsearch cluster again.

1. Upgrade to Graylog latest 2.4 release (2.4.6 at time of writing)

2. **Upgrade ES to the latest 5.6.x on all cluster nodes. (5.6.14 as of this writing)**

    - See: https://www.elastic.co/guide/en/elasticsearch/reference/5.6/setup-upgrade.html

3. **Wait until all shards are initialized after updating ES**

    - If the active write index is still a 2.x ES index, a manual index rotation needs to be triggered

4. Upgrade to Graylog 2.5 (2.5.1 at time of writing)

5. Update the index template for every index set to the latest ES 6 one by using the Graylog HTTP API. (otherwise a user has to rotate the active write index just to install the latest index template)

6. **Check which indices have been created with ES 2.x and need re-indexing**

    - **For each index to re-index do the following**

        - Check that index is not the active write index

        - Create a re-index target index: <index-name>_reindex (e.g. graylog_0_reindex) (with correct settings for shards and replicas)

        - Check that mapping and settings of the new index are correct

        - Start re-index task in ES (using requests_per_second URL param and size param in the payload to avoid overloading the ES cluster)

        - Check progress of re-index task and wait until it is done

        - Check that the document counts in the old and the new index match

        - Delete old index

        - Recreate the old index: <index-name> (e.g. graylog_0) (with correct settings for shards and replicas)

        - Check that mapping and settings of the new index are correct

        - Start re-index task in ES to re-index documents back into the old index (using requests_per_second URL param and size param in the payload to avoid overloading the ES cluster)

        - Check that the document counts in the old and the new index match

        - Recreate Graylog index ranges for the old index

        - Delete temporary re-index target index (e.g. graylog_0_reindex)

7. Delete old re-index tasks from ES

8. Upgrade to the latest ES 6.x version. (6.5.1 as of this writing)

Detailed list of commands

Note: This is not a copy&paste tutorial and you need to read and adjust the commands to your local needs. We use the tools httpie and jq in the following commands.

Prerequisites

Check ES versions of all nodes

The ES version needs to be the same on all ES nodes in the cluster before we can start the re-index process!:

```
http ":9200/_cat/nodes?v&h=name,ip,version"
```

Check that all shards are initialized ("green")

All shards need to be initialized before we can start the re-index process.:

```
http ":9200/_cat/indices?h=health,status,index" | grep -v '^green'
```

Update Graylog index templates in Elasticsearch

The index templates that Graylog writes to Elasticsearch need to be updated before we can start the re-index process.:

```
http post :9000/api/system/indexer/indices/templates/update x-requested-by:httpie
```

Collect indices that need a re-index to work with ES 6

All indices which have not been created with ES 5 need to be re-index to work with ES 6. (or deleted if they are not needed anymore. . .):

```
http :9200/_settings | jq '[ path(.[] | select(.settings.index.version.created <
→"5000000"))[] ]'
```

Re-Index commands for every index

The following commands need to be executed for every index that needs to be re-indexed. Replace the graylog_0 index name in the examples below with the index name you are currently working on.

Check if index is an active write index

We should never re-index the active write target because that index is actively written to. If the active write index is still a 2.x ES index, a manual index rotation needs to be triggered.:

```
http :9200/*_deflector/_alias | jq 'keys'
```

Create new index

The new index needs to be created before it can be used as a re-index target. The request needs to include the correct settings for the number of shards and replicas. These settings can be different for each index set! (actual settings can be found in the Graylog "System / Indices" page for each index set):

```
http put :9200/graylog_0_reindex settings:='{"number_of_shards":4,"number_of_replicas
↪":0}'
```

Check mapping and index settings

Use these commands to check if the settings and index mapping for the new index are correct.:

```
http :9200/graylog_0_reindex/_mapping
http :9200/graylog_0_reindex/_settings
```

Start re-index process

This command starts the actual re-index process. It will return a task ID that can be used to check the progress of the re-index task in Elasticsearch.

The size value in the payload is the batch size that will be used for the re-index process. It defaults to 1000 and can be adjusted to tune the re-indexing process.:

```
http post :9200/_reindex wait_for_completion==false source:='{"index":"graylog_0",
↪"size": 1000}' dest:='{"index":"graylog_0_reindex"}'
```

The re-index API supports the requests_per_second URL parameter to throttle the re-index process. This can be useful to make sure that the re-index process doesn't take too much resources. See this document for an explanation on how the parameter works: https://www.elastic.co/guide/en/elasticsearch/reference/6.0/docs-reindex.html#_url_parameters_3:

```
http post :9200/_reindex wait_for_completion==false requests_per_second==500 source:='
↪{"index":"graylog_0","size": 1000}' dest:='{"index":"graylog_0_reindex"}'
```

Wait for the re-index to complete and check re-index progress

The re-index progress can be checked with the following command using the task ID that has been returned by the re-index request.:

```
http :9200/_tasks/<task-id>
```

Compare documents in the old and new index

Before we continue, we should check that all documents have been re-indexed into the new index by comparing the document counts.:

```
http :9200/graylog_0/_count
http :9200/graylog_0_reindex/_count
```

Delete old index

Now delete the old index so we can recreate it for re-indexing.:

```
http delete :9200/graylog_0
```

Recreate old index

Recreate the old index again so we can use it as a re-index target. The request needs to include the correct settings for the number of shards and replicas. These settings can be different for each index set! (actual settings can be found in the Graylog "System / Indices" page for each index set):

```
http put :9200/graylog_0 settings:='{"number_of_shards":4,"number_of_replicas":0}'
```

Check mapping and index settings

Use these commands to check if the settings and index mapping for the recreated index are correct.:

```
http :9200/graylog_0/_mapping
http :9200/graylog_0/_settings
```

Start re-index process for old index

This command starts the re-index process to move back the documents into the old index. It will return a task ID that can be used to check the progress of the re-index task in Elasticsearch.

The size value in the payload is the batch size that will be used for the re-index process. It defaults to 1000 and can be adjusted to tune the re-indexing process.:

```
http post :9200/_reindex wait_for_completion==false source:='{"index":"graylog_0_
→reindex","size": 1000}' dest:='{"index":"graylog_0"}'
```

The re-index API supports the requests_per_second URL parameter to throttle the re-index process. This can be useful to make sure that the re-index process doesn't take too much resources. See this document for an explanation on how the parameter works: https://www.elastic.co/guide/en/elasticsearch/reference/6.0/docs-reindex.html#_url_parameters_3:

```
http post :9200/_reindex wait_for_completion==false requests_per_second==500 source:='
→{"index":"graylog_0_reindex","size": 1000}' dest:='{"index":"graylog_0"}'
```

Compare documents in the old and new index

Before we continue, we should check that all documents have been re-indexed into the re-created old index by comparing the document counts with the temporary index.:

```
http :9200/graylog_0/_count
http :9200/graylog_0_reindex/_count
```

Create index range for the recreated index

Graylog needs to know about the recreated index by creating an index range for it.:

```
http post :9000/api/system/indices/ranges/graylog_0/rebuild x-requested-by:httpie
```

Delete temporary re-index target index

The temporary re-index target index can now be deleted because we don't use it anymore.:

```
http delete :9200/graylog_0_reindex
```

Cleanup

The re-index process leaves some tasks in Elasticsearch that need to be cleaned up automatically.

Find completed re-index tasks for deletion

Execute the following command to get all the tasks we should remove.:

```
http :9200/.tasks/_search | jq '[.hits.hits[] | select(._source.task.action ==
→"indices:data/write/reindex" and ._source.completed == true) | {"task_id": ._id,
→"description": ._source.task.description}]'
```

Remove completed re-index tasks

Execute the following command for every completed task ID. Re-Index Commands:

```
http delete :9200/.tasks/task/<task-id>
```

CHAPTER

FIVE

CONFIGURING GRAYLOG

5.1 server.conf

The file `server.conf` is the Graylog configuration file.

Note: Check *Default file locations* to locate it in your installation.

It has to use ISO 8859-1/Latin-1 character encoding. Characters that cannot be directly represented in this encoding can be written using Unicode escapes as defined in Java SE Specifications, using the u prefix. For example, u002c.

- **Entries are generally expected to be a single line of the form, one of the following:**

    - `propertyName=propertyValue`

    - `propertyName:propertyValue`

- **White space that appears between the property name and property value is ignored, so the following are equivalent:**

    - `name=Stephen`

    - `name = Stephen`

- White space at the beginning of the line is also ignored.

- Lines that start with the comment characters ! or # are ignored. Blank lines are also ignored.

- The property value is generally terminated by the end of the line. White space following the property value is not ignored, and is treated as part of the property value.

- A property value can span several lines if each line is terminated by a backslash (\) character. For example:

```
targetCities=\
        Detroit,\
        Chicago,\
        Los Angeles
```

This is equivalent to `targetCities=Detroit,Chicago,Los Angeles` (white space at the beginning of lines is ignored).

- The characters newline, carriage return, and tab can be inserted with characters \n, \r, and \t, respectively.

- The backslash character must be escaped as a double backslash. For example:

```
path=c:\\docs\\doc1
```

5.1.1 Properties

General

- `is_master = true`

  - If you are running more than one instances of Graylog server you have to select only one `graylog-server` node as the master. This node will perform periodical and maintenance actions that slave nodes won't.

  - Every slave node will accept messages just as the master nodes. Nodes will fall back to slave mode if there already is a master in the cluster.

- `node_id_file = /etc/graylog/server/<node-id>`

  - The auto-generated node ID will be stored in this file and read after restarts. It is a good idea to use an absolute file path here if you are starting Graylog server from init scripts or similar.

- `password_secret = <secret>`

  - You MUST set a secret that is used for password encryption and salting. The server will refuse to start if it's not set. Use at least 64 characters. If you run multiple `graylog-server` nodes, make sure you use the same `password_secret` for all of them!

  Note: Generate a secret with for example `pwgen -N 1 -s 96`

- `root_username = admin`

  - The default root user is named **admin**.

- `root_password_sha2 = <SHA2>`

  - A SHA2 hash of a password you will use for your initial login. Set this to a SHA2 hash generated with `echo -n "Enter Password:  " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d" " -f1` and you will be able to log in to the web interface with username **admin** and password **yourpassword**.

  > **Caution:** You MUST specify a hash password for the root user (which you only need to initially set up the system and in case you lose connectivity to your authentication backend). This password cannot be changed using the API or via the web interface. If you need to change it, modify it in this file.

- `root_email = ""`

  - The email address of the root user. Default is empty.

- `root_timezone = UTC`

  - The time zone setting of the root user. See this list of valid time zones. Default is UTC.

- `bin_dir = bin`

  - This directory contains binaries that are used by the Graylog server. (relative or absolute)

- `data_dir = data`

  - This directory is used to store Graylog server state. (relative or absolute)

- `plugin_dir = plugin`

  - Set plugin directory here (relative or absolute)

Web & REST API

- **http_bind_address = 127.0.0.1:9000**

    – The network interface used by the Graylog HTTP interface.

    – This network interface must be accessible by all Graylog nodes in the cluster and by all clients using the Graylog web interface.

    – If the port is omitted, Graylog will use port 9000 by default.

- **http_publish_uri = http://$http_bind_address/**

    – The HTTP URI of this Graylog node which is used to communicate with the other Graylog nodes in the cluster and by all clients using the Graylog web interface.

    – The URI will be published in the cluster discovery APIs, so that other Graylog nodes will be able to find and connect to this Graylog node.

    – This configuration setting has to be used if this Graylog node is available on another network interface than $http_bind_address, for example if the machine has multiple network interfaces or is behind a NAT gateway.

    – This configuration setting *must not* be configured to a wildcard address!

    – If `http_bind_address` contains a wildcard IPv4 address (0.0.0.0), `http_publish_uri` will be filled with the first non-loopback IPv4 address of this machine instead.

- **http_external_uri = $http_publish_uri**

    – The public URI of Graylog which will be used by the Graylog web interface to communicate with the Graylog REST API.

    – The external Graylog URI usually has to be specified, if Graylog is running behind a reverse proxy or load-balancer and it will be used to generate URLs addressing entities in the Graylog REST API (see $http_bind_address).

    – When using Graylog Collector, this URI will be used to receive heartbeat messages and must be accessible for all collectors.

    – This setting can be overriden on a per-request basis with the "X-Graylog-Server-URL" HTTP request header.

- **http_enable_cors = true**

    – Enable CORS headers for HTTP interface.

    – This is necessary for JS-clients accessing the server directly.

    – If these are disabled, modern browsers will not be able to retrieve resources from the server.

- **http_enable_gzip = true**

    – This compresses API responses and therefore helps to reduce overall round trip times.

- **http_max_header_size = 8192**

    – The maximum size of the HTTP request headers in bytes.

- **http_thread_pool_size = 16**

    – The size of the thread pool used exclusively for serving the HTTP interface.

- **http_enable_tls = false**

    – This secures the communication with the HTTP interface with TLS to prevent request forgery and eavesdropping.

5.1. server.conf   **95**

- **http_tls_cert_file = /path/to/graylog.crt**

    - The X.509 certificate chain file in PEM format to use for securing the HTTP interface.

- **http_tls_key_file = /path/to/graylog.key**

    - The PKCS#8 private key file in PEM format to use for securing the HTTP interface.

- **http_tls_key_password = secret**

    - The password to unlock the private key used for securing the HTTP interface. (if key is encrypted)

- **trusted_proxies = 127.0.0.1/32, 0:0:0:0:0:0:0:1/128**

    - Comma separated list of trusted proxies that are allowed to set the client address with X-Forwarded-For header. May be subnets, or hosts.

Elasticsearch

- **elasticsearch_hosts = http://node1:9200,http://user:password@node2:19200**

    - List of Elasticsearch hosts Graylog should connect to.

    - Need to be specified as a comma-separated list of valid URIs for the http ports of your elasticsearch nodes.

    - If one or more of your elasticsearch hosts require authentication, include the credentials in each node URI that requires authentication.

    - Default: `http://127.0.0.1:9200`

- **elasticsearch_connect_timeout = 10s**

    - Maximum amount of time to wait for successfull connection to Elasticsearch HTTP port.

    - Default: 10 seconds

- **elasticsearch_socket_timeout = 60s**

    - Maximum amount of time to wait for reading back a response from an Elasticsearch server.

    - Default: 60 seconds

- **elasticsearch_idle_timeout = -1s**

    - Maximum idle time for an Elasticsearch connection. If this is exceeded, this connection will be tore down.

    - Default: infinity

- **elasticsearch_max_total_connections = 20**

    - Maximum number of total connections to Elasticsearch.

    - Default: 20

- **elasticsearch_max_total_connections_per_route = 2**

    - Maximum number of total connections per Elasticsearch route (normally this means per elasticsearch server).

    - Default: 2

- **elasticsearch_max_retries = 2**

    - Maximum number of times Graylog will retry failed requests to Elasticsearch.

    - Default: 2

- **elasticsearch_discovery_enabled = false**

    - Enable automatic Elasticsearch node discovery through Nodes Info, see Elasticsearch Reference » Cluster APIs » Nodes Info.

    - Default: `false`

> **Warning:** Automatic node discovery does not work if Elasticsearch requires authentication, e. g. with Shield.

> **Warning:** This setting must be false on AWS Elasticsearch Clusters (the hosted ones) and should be used carefully. In case of trouble with connections to ES this should be the first option to be disabled. See *Automatic node discovery* for more details.

- **elasticsearch_discovery_filter = rack:42**

    - Filter for including/excluding Elasticsearch nodes in discovery according to their custom attributes, see Elastic Search Reference » Cluster APIs » Node Specification.

    - Default: empty

- **elasticsearch_discovery_frequency = 30s**

    - Frequency of the Elasticsearch node discovery.

    - Default: 30 seconds

- **elasticsearch_compression_enabled = false**

    - Enable payload compression for Elasticsearch requests.

    - Default: false

Rotation

> **Attention:** The following settings identified with *!* in this section have been moved to the database in Graylog 2.0. When you upgrade, make sure to set these to your previous 1.x settings so they will be migrated to the database!

- **rotation_strategy = count** *!*

    - Graylog will use multiple indices to store documents in. You can configured the strategy it uses to determine when to rotate the currently active write index.

    - It supports multiple rotation strategies: - `count` of messages per index, use `elasticsearch_max_docs_per_index`   - `size` per index, use `elasticsearch_max_size_per_index`

    - valid values are `count`, `size` and `time`, default is `count`.

- **elasticsearch_max_docs_per_index = 20000000** *!*

    - (Approximate) maximum number of documents in an Elasticsearch index before a new index is being created, also see no_retention and `elasticsearch_max_number_of_indices`.

    - Configure this if you used `rotation_strategy = count` above.

- **elasticsearch_max_size_per_index = 1073741824** *!*

- – (Approximate) maximum size in bytes per Elasticsearch index on disk before a new index is being created, also see `no_retention` and `` `elasticsearch_max_number_of_indices` ``. Default is 1GB.

  – Configure this if you used `rotation_strategy = size` above.

- **elasticsearch_max_time_per_index = 1d** *!*

  – (Approximate) maximum time before a new Elasticsearch index is being created, also see `no_retention` and `elasticsearch_max_number_of_indices`. Default is 1 day.

  – Configure this if you used `rotation_strategy = time` above.

  – Please note that this rotation period does not look at the time specified in the received messages, but is using the real clock value to decide when to rotate the index!

  – **Specify the time using a duration and a suffix indicating which unit you want:**

    * `1w` = 1 week

    * `1d` = 1 day

    * `12h` = 12 hours

  – Permitted suffixes are: `d` for day, `h` for hour, `m` for minute, `s` for second.

- **elasticsearch_max_number_of_indices = 20** *!*

  – How many indices do you want to keep?

- **retention_strategy = delete** *!*

  – Decide what happens with the oldest indices when the maximum number of indices is reached.

  – **The following strategies are availble:**

    * `delete` - Deletes the index completely (Default)

    * `close` - Closes the index and hides it from the system. Can be re-opened later.

- **elasticsearch_disable_version_check = true**

  – Disable checking the version of Elasticsearch for being compatible with this Graylog release.

> **Warning:** Using Graylog with unsupported and untested versions of Elasticsearch may lead to data loss!

- **no_retention = false**

  – Disable message retention on this node, i. e. disable Elasticsearch index rotation.

> **Attention:** The following settings identified with *!!* have been moved to the database in Graylog 2.2.0. When you upgrade, make sure to set these to your previous settings so they will be migrated to the database. This settings are read **once** at the very first startup to be the initial settings in the database.

- **elasticsearch_shards = 4** *!!*

  – The number of shards for your indices. A good setting here highly depends on the number of nodes in your Elasticsearch cluster. If you have one node, set it to `1`.

- **elasticsearch_replicas = 0** *!!*

    - The number of replicas for your indices. A good setting here highly depends on the number of nodes in your Elasticsearch cluster. If you have one node, set it to `0`.

    Note: `elasticsearch_shards` and `elasticsearch_replicas` only applies to newly created indices.

- **elasticsearch_index_prefix = graylog** *!!*

    - Prefix for all Elasticsearch indices and index aliases managed by Graylog.

- **elasticsearch_template_name = graylog-internal** *!!*

    - Name of the Elasticsearch index template used by Graylog to apply the mandatory index mapping.

    - Default: graylog-internal

- **elasticsearch_analyzer = standard** *!!*

    - Analyzer (tokenizer) to use for message and full_message field. The "standard" filter usually is a good idea.

    - All supported analyzers are: standard, simple, whitespace, stop, keyword, pattern, language, snowball, custom

    - Elasticsearch documentation: https://www.elastic.co/guide/en/elasticsearch/reference/5.6/analysis.html

    - Note that this setting only takes effect on newly created indices.

- **disable_index_optimization = false** *!!*

    - Disable the optimization of Elasticsearch indices after index cycling. This may take some load from Elasticsearch on heavily used systems with large indices, but it will decrease search performance. The default is to optimize cycled indices.

- **index_optimization_max_num_segments = 1** *!!*

    - Optimize the index down to <= index_optimization_max_num_segments. A higher number may take some load from Elasticsearch on heavily used systems with large indices, but it will decrease search performance. The default is 1.

- **allow_leading_wildcard_searches = false**

    - Do you want to allow searches with leading wildcards? This can be extremely resource hungry and should only be enabled with care.

    - See also: *Searching*

- **allow_highlighting = false**

    - Do you want to allow searches to be highlighted? Depending on the size of your messages this can be memory hungry and should only be enabled after making sure your Elasticsearch cluster has enough memory.

- **elasticsearch_request_timeout = 1m**

    - Global request timeout for Elasticsearch requests (e. g. during search, index creation, or index time-range calculations) based on a best-effort to restrict the runtime of Elasticsearch operations.

    - Default: 1m

- **`elasticsearch_index_optimization_timeout = 1h`**
  - Global timeout for index optimization (force merge) requests.
  - Default: 1h

- **`elasticsearch_index_optimization_jobs = 20`**
  - Maximum number of concurrently running index optimization (force merge) jobs.
  - If you are using lots of different index sets, you might want to increase that number.
  - Default: 20

- **`index_ranges_cleanup_interval = 1h`**
  - Time interval for index range information cleanups. This setting defines how often stale index range information is being purged from the database.
  - Default: 1h

- **`output_batch_size = 500`**
  - Batch size for the Elasticsearch output. This is the maximum (!) number of messages the Elasticsearch output module will get at once and write to Elasticsearch in a batch call. If the configured batch size has not been reached within `output_flush_interval` seconds, everything that is available will be flushed at once. Remember that every output buffer processor manages its own batch and performs its own batch write calls. (`outputbuffer_processors` variable)

- **`output_flush_interval = 1`**
  - Flush interval (in seconds) for the Elasticsearch output. This is the maximum amount of time between two batches of messages written to Elasticsearch. It is only effective at all if your minimum number of messages for this time period is less than `output_batch_size *` `outputbuffer_processors`.

- `output_fault_count_threshold = 5`

- **`output_fault_penalty_seconds = 30`**
  - As stream outputs are loaded only on demand, an output which is failing to initialize will be tried over and over again. To prevent this, the following configuration options define after how many faults an output will not be tried again for an also configurable amount of seconds.

- `processbuffer_processors = 5`

- **`outputbuffer_processors = 3`**
  - The number of parallel running processors.
  - Raise this number if your buffers are filling up.

- `outputbuffer_processor_keep_alive_time = 5000`

- `outputbuffer_processor_threads_core_pool_size = 3`

- `outputbuffer_processor_threads_max_pool_size = 30`

- **`udp_recvbuffer_sizes = 1048576`**
  - UDP receive buffer size for all message inputs (e. g. SyslogUDPInput).

- **`processor_wait_strategy = blocking`**
  - Wait strategy describing how buffer processors wait on a cursor sequence. (default: sleeping)
  - **Possible types:**

  * `yielding` - Compromise between performance and CPU usage.

  * `sleeping` - Compromise between performance and CPU usage. Latency spikes can occur after quiet periods.

  * `blocking` - High throughput, low latency, higher CPU usage.

  * `busy_spinning` - Avoids syscalls which could introduce latency jitter. Best when threads can be bound to specific CPU cores.

- **ring_size = 65536**

  - Size of internal ring buffers. Raise this if raising `outputbuffer_processors` does not help anymore.

  - For optimum performance your LogMessage objects in the ring buffer should fit in your CPU L3 cache.

  - Must be a power of 2. (512, 1024, 2048, . . .)

- `inputbuffer_ring_size = 65536`

- `inputbuffer_processors = 2`

- `inputbuffer_wait_strategy = blocking`

- **message_journal_enabled = true**

  - Enable the disk based message journal.

- **message_journal_dir = data/journal**

  - The directory which will be used to store the message journal. The directory must me exclusively used by Graylog and must not contain any other files than the ones created by Graylog itself.

Attention: If you create a seperate partition for the journal files and use a file system creating directories like 'lost+found' in the root directory, you need to create a sub directory for your journal. Otherwise Graylog will log an error message that the journal is corrupt and Graylog will not start.

- `message_journal_max_age = 12h`

- **message_journal_max_size = 5gb**

  - Journal hold messages before they could be written to Elasticsearch.

  - For a maximum of 12 hours or 5 GB whichever happens first.

  - During normal operation the journal will be smaller.

- **message_journal_flush_age = 1m**

  - This setting allows specifying a time interval at which we will force an fsync of data written to the log. For example if this was set to 1000 we would fsync after 1000 ms had passed.

- **message_journal_flush_interval = 1000000**

  - This setting allows specifying an interval at which we will force an fsync of data written to the log. For example if this was set to 1 we would fsync after every message; if it were 5 we would fsync after every five messages.

- **message_journal_segment_age = 1h**

  - This configuration controls the period of time after which Graylog will force the log to roll even if the segment file isn't full to ensure that retention can delete or compact old data.

- `message_journal_segment_size = 100mb`

> **Attention:** When the journal is full and it keeps receiving messages, it will start dropping messages as a FIFO queue: The first dropped message will be the first inserted and so on (and not some random).

- **`async_eventbus_processors = 2`**

    - Number of threads used exclusively for dispatching internal events. Default is 2.

- **`lb_recognition_period_seconds = 3`**

    - How many seconds to wait between marking node as DEAD for possible load balancers and starting the actual shutdown process. Set to 0 if you have no status checking load balancers in front.

- **`lb_throttle_threshold_percentage = 95`**

    - Journal usage percentage that triggers requesting throttling for this server node from load balancers. The feature is disabled if not set.

- `stream_processing_timeout = 2000`

- **`stream_processing_max_faults = 3`**

    - Every message is matched against the configured streams and it can happen that a stream contains rules which take an unusual amount of time to run, for example if its using regular expressions that perform excessive backtracking.

    - This will impact the processing of the entire server. To keep such misbehaving stream rules from impacting other streams, Graylog limits the execution time for each stream.

    - The default values are noted below, the timeout is in milliseconds.

    - If the stream matching for one stream took longer than the timeout value, and this happened more than "max_faults" times that stream is disabled and a notification is shown in the web interface.

Note: Since 0.21 the Graylog server supports pluggable output modules. This means a single message can be written to multiple outputs. The next setting defines the timeout for a single output module, including the default output module where all messages end up.

- **`output_module_timeout = 10000`**

    - Time in milliseconds to wait for all message outputs to finish writing a single message.

- **`stale_master_timeout = 2000`**

    - Time in milliseconds after which a detected stale master node is being rechecked on startup.

- **`shutdown_timeout = 30000`**

    - Time in milliseconds which Graylog is waiting for all threads to stop on shutdown.

MongoDB

- **mongodb_uri = mongodb://...**

    - MongoDB connection string. Enter your MongoDB connection and authentication information here.

    - See https://docs.mongodb.com/manual/reference/connection-string/ for details.

    - Take notice that +-signs in the username or password need to be replaced by `%2B`.

    - **Examples:**

        * Simple: `mongodb://localhost/graylog`

        * Authenticate against the MongoDB server: `mongodb_uri = mongodb://grayloguser:secret@localhost:27017/graylog`

        * Use a replica set instead of a single host: `mongodb://grayloguser:secret@localhost:27017,localhost:27018,localhost:27019/graylog?replicaSet=rs01`

        * DNS Seedlist is set as `mongodb+srv://server.example.org/graylog`.

- **mongodb_max_connections = 1000**

    - Increase this value according to the maximum connections your MongoDB server can handle from a single client if you encounter MongoDB connection problems.

- **mongodb_threads_allowed_to_block_multiplier = 5**

    - Number of threads allowed to be blocked by MongoDB connections multiplier. Default: 5

    - If `mongodb_max_connections` is 100, and `mongodb_threads_allowed_to_block_multiplier` is 5, then 500 threads can block. More than that and an exception will be thrown.

    - http://api.mongodb.com/java/current/com/mongodb/MongoOptions.html#threadsAllowedToBlockForConnectionMultiplier

Email

- `transport_email_enabled = false`

- `transport_email_hostname = mail.example.com`

- `transport_email_port = 587`

- `transport_email_use_auth = true`

- **transport_email_use_tls = true**

    - Enable SMTP with STARTTLS for encrypted connections.

- **transport_email_use_ssl = false**

    - Enable SMTP over SSL (SMTPS) for encrypted connections.

> **Attention:** Make sure to enable only *one* of these two settings because most (or all) SMTP services only support one of the encryption mechanisms on the same port. Most SMTP services support SMTP with STARTTLS while SMTPS is deprecated on most SMTP services. Setting both to `false` is needed when you want to sent via unencrypted connection.

- `transport_email_auth_username = you@example.com`

- transport_email_auth_password = secret

- transport_email_subject_prefix = [graylog]

- transport_email_from_email = graylog@example.com

- **transport_email_web_interface_url = https://graylog.example.com**

    - Specify this to include links to the stream in your stream alert mails.

    - This should define the fully qualified base url to your web interface exactly the same way as it is accessed by your users.

HTTP

- **http_connect_timeout = 5s**

    - The default connect timeout for outgoing HTTP connections.

    - Values must be a positive duration (and between 1 and 2147483647 when converted to milliseconds).

    - Default: 5s

- **http_read_timeout = 10s**

    - The default read timeout for outgoing HTTP connections.

    - Values must be a positive duration (and between 1 and 2147483647 when converted to milliseconds).

    - Default: 10s

- **http_write_timeout = 10s**

    - The default write timeout for outgoing HTTP connections.

    - Values must be a positive duration (and between 1 and 2147483647 when converted to milliseconds).

    - Default: 10s

- **http_proxy_uri =**

    - HTTP proxy for outgoing HTTP connections

Attention: If you configure a proxy, make sure to also configure the "http_non_proxy_hosts" option so internal HTTP connections with other nodes does not go through the proxy.

- **http_non_proxy_hosts =**

    - A list of hosts that should be reached directly, bypassing the configured proxy server.

    - This is a list of patterns separated by ";". The patterns may start or end with a "*" for wildcards.

    - Any host matching one of these patterns will be reached through a direct connection instead of through a proxy.

Processing Status

Note: The settings for processing status usually don't need to be tweaked.

- **processing_status_persist_interval = 1s**

    - The server is writing processing status information to the database on a regular basis. This setting controls how often the data is written to the database.

    - Values must be a positive duration and cannot be less than one second.

    - Default: 1s (one second)

- **processing_status_update_threshold = 1m**

    - Configures the threshold for detecting outdated processing status records. Any records that haven't been updated in the configured threshold will be ignored.

    - Values must be a positive duration and cannot be less than one second.

    - Default: 1m (one minute)

- **processing_status_journal_write_rate_threshold= 1**

    - Configures the journal write rate threshold for selecting processing status records. Any records that have a lower one minute rate than the configured value might be ignored. (dependent on number of messages in the journal)

    - Values must be a positive duration.

    - Default: 1

Script alert notification

- **integrations_web_interface_uri = https://graylog.example.com**

    - Specify this to include a search page link (that displays relevant alert messages) in the script arguments or standard in JSON.

    - This should define the fully qualified base url to your web interface exactly the same way as it is accessed by your users.

    - Default: none

- **integrations_scripts_dir = /usr/share/graylog-server/scripts**

    - An absolute or relative path where scripts are permitted to be executed from.

    - If specified, this overrides the default location (see the *File Locations* document.

Others

- **gc_warning_threshold = 1s**

    - The threshold of the garbage collection runs. If GC runs take longer than this threshold, a system notification will be generated to warn the administrator about possible problems with the system. Default is 1 second.

- **ldap_connection_timeout = 2000**

    - Connection timeout for a configured LDAP server (e. g. ActiveDirectory) in milliseconds.

- **disable_sigar = false**

    - Disable the use of SIGAR for collecting system stats.

- **dashboard_widget_default_cache_time = 10s**

    - The default cache time for dashboard widgets. (Default: 10 seconds, minimum: 1 second)

- **proxied_requests_thread_pool_size = 32**

    - For some cluster-related REST requests, the node must query all other nodes in the cluster. This is the maximum number of threads available for this. Increase it, if `/cluster/*` requests take long to complete.

    - Should be `http_thread_pool_size * average_cluster_size` if you have a high number of concurrent users.

- **default_events_index_prefix = gl-events**

    - The default index prefix for graylog events.

- **default_system_events_index_prefix = gl-system-events**

    - The default index prefix for graylog system events.

5.2 Web interface

When your Graylog instance/cluster is up and running, the next thing you usually want to do is check out our web interface, which offers you great capabilities for searching and analyzing your indexed data and configuring your Graylog environment. Per default you can access it using your browser on `http://<graylog-server>:9000/`.

5.2.1 Overview

The Graylog web interface was rewritten in JavaScript for 2.0 to be a client-side single-page browser application. This means its code is running solely in your browser, fetching all data via HTTP(S) from the REST API of your Graylog server.

Note: The HTTP address must be accessible by everyone using the web interface. This means that Graylog *must* listen on a public network interface *or* be exposed to one using a proxy, NAT or a load balancer!

5.2.2 Configuration Options

If our default settings do not work for you, there is a number of options in the Graylog server configuration file which you can change to influence its behavior:

Setting	Default	Explanation
`http_bind_address`	`127.0.0.1:9000`	The network interface used by the Graylog HTTP interface.
`http_publish_uri`	If not set, `http://$http_bind_address` will be used.	The HTTP URI of this Graylog node which is used to communicate with the other Graylog nodes in the cluster and by all clients using the Graylog web interface.
`http_external_uri`	If `$http_publish_uri` not set, `$http_publish_uri` will be used.	The public URI of Graylog which will be used by the Graylog web interface to communicate with the Graylog REST API. Graylog web interface.
`http_enable_cors`	`true`	This is necessary for JS-clients accessing the server directly. If disabled, modern browsers will not be able to retrieve resources from the server.
`http_enable_gzip`	`true`	Serve web interface assets using compression to reduce overall roundtrip times.
`http_max_header_size`	`8192`	The maximum size of the HTTP request headers in bytes.
`http_thread_pool_size`	`16`	The size of the thread pool used exclusively for serving the HTTP interface.
`http_enable_tls`	`false`	This secures the communication with the HTTP interface with TLS to prevent request forgery and eavesdropping.
`http_tls_cert_file`	(no default)	The X.509 certificate chain file in PEM format to use for securing the HTTP interface.
`http_tls_key_file`	(no default)	The PKCS#8 private key file in PEM format to use for securing the HTTP interface.
`http_tls_key_password`	(no default)	The password to unlock the private key used for securing the HTTP interface. (only needed if the key is encryped)

5.2.3 How does the web interface connect to the Graylog server?

The web interface is fetching all information it is showing from the REST API of the Graylog server. Therefore it needs to connect to it using HTTP(S). There are several ways how you can define which way the web interface connects to the Graylog server. The URI used by the web interface is determined in this exact order:

- If the HTTP(S) client going to the web interface port sends a `X-Graylog-Server-URL` header, which contains a valid URL, then this is overriding everything else.

- If `http_external_uri` is defined in the Graylog configuration file, this is used if the aforementioned header is not set.

- If `http_publish_uri` is defined in the Graylog configuration file, this is used if the aforementioned `http_external_uri` is not set.

- If none of the above are defined, `http://$http_bind_address` is used.

The web interface assets (e.g. the `index.html`, CSS and JavaScript files) are accessible at the URI root (`/` by default) and the REST API endpoints are accessible at the `/api` path.

Example:

Setting `http_bind_address` to `10.0.0.1:9000` configures the Graylog server with the following URLs.

- Web interface: `http://10.0.0.1:9000/`

- REST API: `http://10.0.0.1:9000/api/`

5.2.4 Browser Compatibility

Writing the web interface as a single-page application is a challenging task. We want to provide the best possible experience to everyone, which often means using modern web technology only available in recent browsers, while keeping a reasonable compatibility with old and less-capable browsers. These browsers are officially supported in Graylog 3.0:

Browser	OS	Minimum Version
Chrome	Windows, OS X, Linux	50
Firefox	Windows, OS X, Linux	45 / 38 ESR
Internet Explorer	Windows	11
Microsoft Edge	Windows	25
Safari	OS X	9

Please take into account that you need to enable JavaScript in order to use Graylog web interface.

5.2.5 Making the web interface work with load balancers/proxies

If you want to run a load balancer/reverse proxy in front of Graylog, you need to make sure that:

- The HTTP port of the load balancer/reverse proxy is accessible for clients
- The HTTP address for the Graylog server is properly set (as explained in *How does the web interface connect to the Graylog server?*), so it is resolvable and accessible for the load balancer/reverse proxy.
- If you use SSL, your certificates must be valid and trusted by your clients.

Note: To help you with your specific environment, we show some example configuration use cases.

For the configuration use cases below we assume the following:

- Your Graylog server configuration contains `http_bind_address = 127.0.0.1:9000`
- The hostname for the setup is `graylog.example.org`
- The IP address for that hostname is `192.168.0.10`

Using a Layer 3 load balancer (forwarding TCP Ports)

1. Configure your load balancer to forward connections going to `192.168.0.10:80` to `127.0.0.1:9000`.
2. Start the Graylog server as usual.
3. Access the web interface on `http://graylog.example.org`.
4. Read up on *Using HTTPS*.

NGINX

Proxy web interface and API traffic using HTTP:

```
server
{
    listen 80 default_server;
    listen [::]:80 default_server ipv6only=on;
    server_name graylog.example.org;

    location / {
      proxy_set_header Host $http_host;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Server $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Graylog-Server-URL http://$server_name/;
      proxy_pass       http://127.0.0.1:9000;
    }
}
```

NGINX can be used for SSL Termination, you would only need to modify the `server listen` directive and add all Information about your certificate.

If you are running multiple Graylog Server you might want to use HTTPS/SSL to connect to the Graylog Servers (on how to Setup read *Using HTTPS*) and use HTTPS/SSL on NGINX. The configuration for TLS certificates, keys and ciphers is omitted from the sample config for brevity's sake.

Proxy web interface and API traffic using HTTPS (TLS):

```
server
{
    listen       443 ssl http2;
    server_name graylog.example.org;
    # <- your SSL Settings here!

    location /
    {
      proxy_set_header Host $http_host;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Server $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Graylog-Server-URL https://$server_name/;
      proxy_pass       http://127.0.0.1:9000;
    }
}
```

If you want to serve serveral different applications under one domain name, you can also serve the Graylog web interface using a path prefix.

Proxy web interface and API traffic under a path prefix using HTTP:

```
server
{
    listen 80 default_server;
    listen [::]:80 default_server ipv6only=on;
    server_name applications.example.org;

    location /graylog/
```

(continues on next page)

```
    {
      proxy_set_header Host $http_host;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Server $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Graylog-Server-URL http://$server_name/graylog/;
      rewrite           ^/graylog/(.*)$  /$1  break;
      proxy_pass        http://127.0.0.1:9000;
    }
}
```

This makes your Graylog setup available under the following URLs:

- Web interface: `http://applications.example.org/graylog/`

- REST API: `http://applications.example.org/graylog/api/`

Apache httpd 2.x

Proxy web interface and API traffic using HTTP:

```
<VirtualHost *:80>
    ServerName graylog.example.org
    ProxyRequests Off
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>

    <Location />
        RequestHeader set X-Graylog-Server-URL "http://graylog.example.org/"
        ProxyPass http://127.0.0.1:9000/
        ProxyPassReverse http://127.0.0.1:9000/
    </Location>

</VirtualHost>
```

Proxy web interface and API traffic using HTTPS (TLS):

```
<VirtualHost *:443>
    ServerName graylog.example.org
    ProxyRequests Off
    SSLEngine on
    # <- your SSL Settings here!

    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>

    <Location />
        RequestHeader set X-Graylog-Server-URL "https://graylog.example.org/"
        ProxyPass http://127.0.0.1:9000/
        ProxyPassReverse http://127.0.0.1:9000/
    </Location>

</VirtualHost>
```

HAProxy 1.6

Proxy web interface and API traffic using HTTP:

```
frontend http
    bind 0.0.0.0:80

    option forwardfor
    http-request add-header X-Forwarded-Host %[req.hdr(host)]
    http-request add-header X-Forwarded-Server %[req.hdr(host)]
    http-request add-header X-Forwarded-Port %[dst_port]
    acl is_graylog hdr_dom(host) -i -m str graylog.example.org
    use_backend    graylog if is_graylog

backend graylog
    description     The Graylog Web backend.
    http-request set-header X-Graylog-Server-URL http://graylog.example.org/
    use-server graylog_1
    server graylog_1 127.0.0.1:9000 maxconn 20 check
```

Multiple Backends (roundrobin) with Health-Check (using HTTP):

```
frontend graylog_http
    bind *:80
    option forwardfor
    http-request add-header X-Forwarded-Host %[req.hdr(host)]
    http-request add-header X-Forwarded-Server %[req.hdr(host)]
    http-request add-header X-Forwarded-Port %[dst_port]
    acl is_graylog hdr_dom(host) -i -m str graylog.example.org
    use_backend    graylog

backend graylog
    description     The Graylog Web backend.
    balance roundrobin
    option httpchk HEAD /api/system/lbstatus
    http-request set-header X-Graylog-Server-URL http://graylog.example.org/
    server graylog1 192.168.0.10:9000 maxconn 20 check
    server graylog2 192.168.0.11:9000 maxconn 20 check
    server graylog3 192.168.0.12:9000 maxconn 20 check
```

5.3 Load balancer integration

When running multiple Graylog servers a common deployment scenario is to route the message traffic through an IP load balancer. By doing this we can achieve both a highly available setup, as well as increasing message processing throughput, by simply adding more servers that operate in parallel.

5.3.1 Load balancer state

However, load balancers usually need some way of determining whether a backend service is reachable and healthy or not. For this purpose Graylog exposes a load balancer state that is reachable via its REST API.

There are two ways the load balancer state can change:

- due to a lifecycle change (e.g. the server is starting to accept messages, or shutting down)

- due to manual intervention via the REST API

Note: In the following examples we assume that the Graylog REST API is available on the URI path `/api/` (e. g. `http://graylog.example.com/api/`).

To query the current load balancer status of a Graylog instance, all you need to do is to issue a HTTP call to its REST API:

```
GET /api/system/lbstatus
```

The status knows three different states, `ALIVE`, `THROTTLED` and `DEAD`, which is also the `text/plain` response of the resource. Additionally, the same information is reflected in the HTTP status codes: If the state is `ALIVE` the return code will be `200 OK`, for `THROTTLED` it will be `429 (too many request)` and for `DEAD` it will be `503 Service unavailable`. This is done to make it easier to configure a wide range of load balancer types and vendors to be able to react to the status.

The resource is accessible without authentication to make it easier for load balancers to access it.

To programmatically change the load balancer status, an additional endpoint is exposed:

```
PUT /api/system/lbstatus/override/alive
PUT /api/system/lbstatus/override/dead
PUT /api/system/lbstatus/override/throttled
```

Only authenticated and authorized users are able to change the status, in the currently released Graylog version this means only admin users can change it.

5.3.2 Graceful shutdown

Often, when running a service behind a load balancer, the goal is to be able to perform zero-downtime upgrades, by taking one of the servers offline, upgrading it, and then bringing it back online. During that time the remaining servers can take the load seamlessly.

By using the load balancer status API described above one can already perform such a task. However, it would still be guesswork when the Graylog server is done processing all the messages it already accepted.

For this purpose Graylog supports a graceful shutdown command, also accessible via the web interface and API. It will set the load balancer status to `DEAD`, stop all inputs, turn on messages processing (should it have been disabled manually previously), and flush all messages in memory to Elasticsearch. After all buffers and caches are processed, it will shut itself down safely.

5.3.3 Web Interface

It is possible to use the Graylog web interface behind a load balancer for high availability purposes.

Note: Take care of the configuration you need *with a proxy setup*, as it will *not* work out of the box.

5.4 Using HTTPS

We highly recommend securing your Graylog installation using SSL/TLS to make sure that no sensitive data is sent over the wire in plain text. To make this work, you need to enable the `http_enable_tls` setting in your Graylog server configuration.

You also need to make sure that you have proper certificates in place, which are valid and trusted by the clients.

Note: If you're operating a single-node setup and would like to use HTTPS for the Graylog web interface and the Graylog REST API, it's possible to use *NGINX or Apache as a reverse proxy*.

5.4.1 Things to consider

You have multiple options to ensure that your connection is secure and safe. The first would be to create a self-signed certificate, add that to the previously copied java keystore and use this keystore with your Graylog java options. Since you will need to do this for every certificate and every trust store, this quickly becomes unmanageable in a clustered architecture. Each node needs to trust all certificates from all other nodes.

The second option would be to create your own certificate authority. You only add the certificate authority once to the key store and all certificates that are created with this authority will be trusted.

The same can be done if you have already your own certificate authority, you only need the certificates and keys in the format that can be used with Graylog. Add the certificate authority key to the keystore and all certificates that are signed by this certificate authority will be trusted. Same when you pay for certificates or use a free Certificate authority like let's encrypt to get the server certificates.

Just add the certificate authority to the keystore and all certificates are trusted.

5.4.2 Certificate/Key file format

When you are configuring TLS, you need to make sure that your certificate/key files are in the right format, which is X.509 for certificates and PKCS#8 for the private keys. Both must to be stored in PEM format.

5.4.3 Creating a self-signed private key/certificate

Create a file named `openssl-graylog.cnf` with the following content (customized to your needs):

```
[req]
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no

# Details about the issuer of the certificate
[req_distinguished_name]
C = US
ST = Some-State
L = Some-City
O = My Company
OU = My Division
CN = graylog.example.com

[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

# IP addresses and DNS names the certificate should include
# Use IP.### for IP addresses and DNS.### for DNS names,
# with "###" being a consecutive number.
[alt_names]
IP.1 = 203.0.113.42
DNS.1 = graylog.example.com
```

Create PKCS#5 private key and X.509 certificate:

```
$ openssl version
OpenSSL 0.9.8zh 14 Jan 2016
$ openssl req -x509 -days 365 -nodes -newkey rsa:2048 -config openssl-graylog.cnf -
→keyout pkcs5-plain.pem -out cert.pem
Generating a 2048 bit RSA private key
..........................+++
.+++
writing new private key to 'pkcs5-plain.pem'
-----
```

Convert PKCS#5 private key into a *unencrypted* PKCS#8 private key:

```
$ openssl pkcs8 -in pkcs5-plain.pem -topk8 -nocrypt -out pkcs8-plain.pem
```

Convert PKCS#5 private key into an *encrypted* PKCS#8 private key (using the passphrase `secret`):

```
$ openssl pkcs8 -in pkcs5-plain.pem -topk8 -out pkcs8-encrypted.pem -passout␣
→pass:secret
```

5.4.4 Converting a PKCS #12 (PFX) file to private key and certificate pair

PKCS #12 key stores (PFX files) are commonly used on Microsoft Windows. This needs to be done only if you have to convert PKCS #12 Keys to be used with Graylog.

In this example, the PKCS #12 (PFX) file is named `keystore.pfx`:

```
$ openssl pkcs12 -in keystore.pfx -nokeys -out graylog-certificate.pem
$ openssl pkcs12 -in keystore.pfx -nocerts -out graylog-pkcs5.pem
$ openssl pkcs8 -in graylog-pkcs5.pem -topk8 -out graylog-key.pem
```

The resulting `graylog-certificate.pem` and `graylog-key.pem` can be used in the Graylog configuration file.

5.4.5 Converting an existing Java Keystore to private key/certificate pair

This section describes how to export a private key and certificate from an existing Java KeyStore in JKS format. This is needed if you want to export the certificates from the Java KeyStore.

The starting point is an existing Java KeyStore in JKS format which contains a private key and certificate which should be used in Graylog:

```
$ keytool -list -v -keystore keystore.jks -alias graylog.example.com
Enter keystore password:
Alias name: graylog.example.com
Creation date: May 10, 2016
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=graylog.example.com, OU=Unknown, O="Graylog, Inc.", L=Hamburg, ST=Hamburg,
→C=DE
Issuer: CN=graylog.example.com, OU=Unknown, O="Graylog, Inc.", L=Hamburg, ST=Hamburg,
→C=DE
Serial number: 2b33832d
Valid from: Tue May 10 10:02:34 CEST 2016 until: Mon Aug 08 10:02:34 CEST 2016
Certificate fingerprints:
      MD5:  8A:3D:9F:ED:69:93:1B:6C:E3:29:66:EA:82:8D:42:BE
      SHA1: 5B:27:92:25:46:36:BC:F0:82:8F:9A:30:D8:50:D0:ED:32:4D:C6:A0
      SHA256:
→11:11:77:F5:F6:6A:20:A8:E6:4A:5D:B5:20:21:4E:B8:FE:B6:38:1D:45:6B:ED:D0:7B:CE:B8:C8:BC:DD:B4:FB
      Signature algorithm name: SHA256withRSA
      Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: AC 79 64 9F A1 60 14 B9   51 F4 F5 0B B3 B5 02 A5  .yd..`..Q.......
0010: B8 07 DC 7B                                        ....
]
]
```

The Java KeyStore in JKS format has to be converted to a PKCS#12 keystore, so that OpenSSL can work with it:

```
$ keytool -importkeystore -srckeystore keystore.jks -destkeystore keystore.p12 -
↪deststoretype PKCS12
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
Entry for alias graylog.example.com successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or
↪cancelled
```

After the keystore has been successfully converted into PKCS#12 format, OpenSSL can export the X.509 certificate with PEM encoding:

```
$ openssl pkcs12 -in keystore.p12 -nokeys -out graylog-certificate.pem
Enter Import Password:
MAC verified OK
```

The private key can only be exported in PKCS#5 format with PEM encoding:

```
$ openssl pkcs12 -in keystore.p12 -nocerts -out graylog-pkcs5.pem
Enter Import Password:
MAC verified OK
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Graylog currently only supports PKCS#8 private keys with PEM encoding, so OpenSSL has to convert it into the correct format:

```
$ openssl pkcs8 -in graylog-pkcs5.pem -topk8 -out graylog-key.pem
Enter pass phrase for graylog-pkcs5.pem:
Enter Encryption Password:
Verifying - Enter Encryption Password:
```

The working directory should now contain the PKCS#8 private key (`graylog-key.pem`) and the X.509 certificate (`graylog-certificate.pem`) to be used with Graylog:

```
$ head graylog-key.pem graylog-certificate.pem
==> graylog-key.pem <==
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIE6TAbBgkqhkiG9w0BBQMwDgQIwMhLa5bw9vgCAggABIIEyN42AeYJJNBEiqhI
mWqJDot4Jokw2vB4abcIJ5Do4+7tjtMrecVRCDSvBZzjkXjnbumBHEoxexe5f0/z
wgq6f/UDyTM3uKYQTG91fcqTyMDUlo3Wc8OqSqsNehOAQzA7hMCehqgNJHO0Zfny
EFvrXHurJWi4eA9vLRup86dbm4Wp3o8pmjOLduXieHfcgVtm5jfd7XfL5cRFS8kS
bSFH4v8xDxLNaJmKkKl9gPCACMRbO9nGk/Z9q9N8zkj+xG9lxlNRMX51SRzg20E0
nyyKTb39tJF35zjroB2HfiFWyrPQ1uF6yGoroGvu0L3eWosjBLjdRs0eBgjJCm5P
ic9zSVqMH6/4CPKJqvB97vP4QhpYcr9jlYJsbn6Zg4OIELpM00VLvp0yU9tqTuRR
TDPYAlNMLZ2RrV52CEsh3zO21WHM7r187x4WHgprDFnjkXf02DrFhgCsGwkEQnb3
vj86q13RHhqoXT4W0zugvcv2/NBLMv0HNQBAfEK3X1YBmtQpEJhwSxeszA1i7CpU


==> graylog-certificate.pem <==
Bag Attributes
    friendlyName: graylog.example.com
    localKeyID: 54 69 6D 65 20 31 34 36 32 38 36 37 38 32 33 30 39 32
subject=/C=DE/ST=Hamburg/L=Hamburg/O=Graylog, Inc./OU=Unknown/CN=graylog.example.com
issuer=/C=DE/ST=Hamburg/L=Hamburg/O=Graylog, Inc./OU=Unknown/CN=graylog.example.com
-----BEGIN CERTIFICATE-----
MIIDkTCCAnmgAwIBAgIEKzODLTANBgkqhkiG9w0BAQsFADB5MQswCQYDVQQGEwJE
RTEQMA4GA1UECBMHSGFtYnVyZzEQMA4GA1UEBxMHSGFtYnVyZzEWMBQGA1UEChMN
```

```
R3JheWxvZywgSW5jLjEQMA4GA1UECxMHVW5rbm93bjEcMBoGA1UEAxMTZ3JheWxv
Zy5leGFtcGxlLmNvbTAeFw0xNjA1MTAwODAyMzRaFw0xNjA4MDgwODAyMzRaMHkx
```

The resulting PKCS#8 private key (`graylog-key.pem`) and the X.509 certificate (`graylog-certificate.pem`) can now be used to enable encrypted connections with Graylog by enabling TLS for the Graylog REST API and the web interface in the Graylog configuration file:

```
# Enable HTTPS support for the HTTP interface.
# This secures the communication with the HTTP interface with TLS to prevent request
↪forgery and eavesdropping.
http_enable_tls = true

# The X.509 certificate chain file in PEM format to use for securing the HTTP
↪interface.
http_tls_cert_file = /path/to/graylog-certificate.pem

# The PKCS#8 private key file in PEM format to use for securing the HTTP interface.
http_tls_key_file = /path/to/graylog-key.pem

# The password to unlock the private key used for securing the HTTP interface. (if
↪key is encrypted)
http_tls_key_password = secret
```

5.4.6 Sample files

This section shows the difference between following private key formats with samples. It will help you to identify between the following private key formats and provides samples.

PKCS#5 plain private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIBOwIBAAJBANxtmQ1Kccdp7HBNt8zgTai48Vv617bj4SnhkcMN99sCQ2Naj/sp
[...]
NiCYNLiCawBbpZnYw/ztPVACK4EwOpUy+u19cMB0JA==
-----END RSA PRIVATE KEY-----
```

PKCS#8 plain private key:

```
-----BEGIN PRIVATE KEY-----
MIIBVAIBADANBgkqhkiG9w0BAQEFAASCAT4wggE6AgEAAkEA6GZN0rQFKRIVaPOz
[...]
LaLGdd9G63kLg85eldSy55uIAXsvqQIgfSYaliVtSbAgyx1Yfs3hJ+CTpNKzTNv/
Fx80EltYV6k=
-----END PRIVATE KEY-----
```

PKCS#5 encrypted private key:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,E83B4019057F55E9

iIPs59nQn4RSd7ppch9/vNE7PfRSHLoQFmaAjaF0DxjV9oucznUjJq2gphAB2E2H
[...]
y5IT1MZPgN3LNkVSsLPWKo08uFZQdfu0JTKcn7NPyRc=
-----END RSA PRIVATE KEY-----
```

PKCS#8 encrypted private key:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIBpjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIU9Y9p2EfWucCAggA
[...]
IjsZNp6zmlqf/RXnETsJjGd0TXRWaEdu+XOOyVyPskX2177X9DUJoD31
-----END ENCRYPTED PRIVATE KEY-----
```

5.4.7 Adding a self-signed certificate to the JVM trust store

Graylog nodes inside a cluster need to communicate with each other using the Graylog REST API. When using HTTPS for the Graylog REST API, the X.509 certificate must be *trusted* by the JVM trust store (similar to the trusted CA bundle in an operating system), otherwise communication will fail.

Important: If you are using different X.509 certificates for each Graylog node, you have to add *all of them* into the JVM trust store of each Graylog node.

The default trust store of an installed Java runtime environment can be found at `$JAVA_HOME/jre/lib/security/cacerts`. In order not to "pollute" the official trust store, we make a copy of it which we will use with Graylog instead:

```
$ cp -a "${JAVA_HOME}/jre/lib/security/cacerts" /path/to/cacerts.jks
```

After the original key store file has been copied, we can add the self-signed certificate (`cert.pem`, see *Creating a self-signed private key/certificate*) to the key store (the default password is `changeit`):

```
$ keytool -importcert -keystore /path/to/cacerts.jks -storepass changeit -alias␣
→graylog-self-signed -file cert.pem
Owner: CN=graylog.example.com, O="Graylog, Inc.", L=Hamburg, ST=Hamburg, C=DE
Issuer: CN=graylog.example.com, O="Graylog, Inc.", L=Hamburg, ST=Hamburg, C=DE
Serial number: 8c80134cee556734
Valid from: Tue Jun 14 16:38:17 CEST 2016 until: Wed Jun 14 16:38:17 CEST 2017
Certificate fingerprints:
      MD5:  69:D1:B3:01:46:0D:E9:45:FB:C6:6C:69:EA:38:ED:3E
      SHA1: F0:64:D0:1B:3B:6B:C8:01:D5:4D:33:36:87:F0:FB:10:E1:36:21:9E
      SHA256:␣
→F7:F2:73:3D:86:DC:10:22:1D:14:B8:5D:66:B4:EB:48:FD:3D:74:89:EC:C4:DF:D0:D2:EC:F8:5D:78:49:E7:2F
      Signature algorithm name: SHA1withRSA
      Version: 3

Extensions:

[Other details about the certificate...]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

To verify that the self-signed certificate has indeed been added, it can be listed with the following command:

```
$ keytool -keystore /path/to/cacerts.jks -storepass changeit -list | grep graylog-
→self-signed -A1
graylog-self-signed, Jun 14, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):␣
→F0:64:D0:1B:3B:6B:C8:01:D5:4D:33:36:87:F0:FB:10:E1:36:21:9E
```

The printed certificate fingerprint (SHA1) should match the one printed when importing the self-signed certificate.

In order for the JVM to pick up the new trust store, it has to be started with the JVM parameter `-Djavax.net.ssl.trustStore=/path/to/cacerts.jks`. If you've been using another password to encrypt the JVM trust store than the default `changeit`, you additionally have to set the JVM parameter `-Djavax.net.ssl.trustStorePassword=secret`.

Most start and init scripts for Graylog provide a `JAVA_OPTS` variable which can be used to pass the `javax.net.ssl.trustStore` and (optionally) `javax.net.ssl.trustStorePassword` system properties.

Note: The default location to change the JVM parameter depends on your installation type and is documented *with all other default locations*.

Warning: Without adding the previously created Java keystore to the JVM parameters, Graylog won't be able to verify any self-signed certificates or custom CA certificates.

5.4.8 Disabling specific TLS ciphers and algorithms

Since Java 7u76 it is possible to disable specific TLS algorithms and ciphers for secure connections.

In order to disable specific TLS algorithms and ciphers, you need to provide a properties file with a list of disabled algorithms and ciphers. Take a look at the example security.properties in the Graylog source repository.

For example, if you want to disable all algorithms except for TLS 1.2, the properties file has to contain the following line:

```
jdk.tls.disabledAlgorithms=SSLv2Hello, SSLv3, TLSv1, TLSv1.1
```

If additionally you want to disable DSA/RSA key sizes lower than 2048 bits and EC key sizes lower than 160 bits, the properties file has to contain the following line:

```
jdk.tls.disabledAlgorithms=SSLv2Hello, SSLv3, TLSv1, TLSv1.1, EC keySize < 160, RSA
↪keySize < 2048, DSA keySize < 2048
```

To load the properties file into a JVM, you have to pass it to Java using the `java.security.properties` system property:

```
java -Djava.security.properties=/path/to/security.properties -jar /path/to/graylog.
↪jar server
```

Most start and *init scripts for Graylog* provide a `JAVA_OPTS` variable which can be used to pass the `java.security.properties` system property.

Further reading

- https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html#DisabledAlgorithms

- http://www.oracle.com/technetwork/java/javase/7u76-relnotes-2389087.html

- http://bugs.java.com/bugdatabase/view_bug.do?bug_id=7133344

- https://tersesystems.com/2014/01/13/fixing-the-most-dangerous-code-in-the-world/

5.5 Multi-node Setup

This guide doesn't provide a step-by-step tutorial for building a multi-node Graylog cluster but does simply give some advice for questions that might arise during the setup.

It's important for such a project that you understand each step in the setup process and do some planning upfront. Without a proper roadmap of all the things you want to achieve with a Graylog cluster, you will be lost on the way.

Graylog should be the last component you install in this setup. Its dependencies, namely MongoDB and Elasticsearch, have to be up and running first.

Important: This guide doesn't include instructions for running a multi-node Graylog cluster in an untrusted network. We assume that the connection between the hosts is trusted and doesn't have to be secured individually.

5.5.1 Prerequisites

Every server which is part of this setup should have the software requirements installed to run the targeted software. All software requirements can be found in the installation manual.

We highly recommend that the system time on all systems is kept in sync via NTP or a similar mechanism. Needless to say that DNS resolution must be working, too. Because everything is a freaking DNS problem.

In order to simplify the installation process, the servers should have a working Internet connection.

5.5.2 MongoDB replica set

We recommend to deploy a MongoDB replica set.

MongoDB doesn't have to run on dedicated servers for the workload generated by Graylog, but you should follow the recommendations given in the MongoDB documentation about architecture. Most important is that you have an odd number of MongoDB servers in the replica set.

In most setups, each Graylog server will also host an instance of MongoDB which is part of the same replica set and shares the data with all other nodes in the cluster.

Note: To avoid unauthorized access to your MongoDB database, the MongoDB replica set should be setup with authentication.

The correct order of working steps should be as follows:

1. Create the replica set (`rs01`)

2. Create the database (`graylog`)

3. Create a user account for accessing the database, which has the roles `readWrite` and `dbAdmin`.

If your MongoDB needs to be reachable over network you should set the IP with `bind_ip` in the configuration.

5.5.3 Elasticsearch cluster

The Elasticsearch setup documentation should help you to install Elasticsearch with a robust base configuration.

It is important to name the Elasticsearch cluster not simply named *elasticsearch* to avoid accidental conflicts with Elasticsearch nodes using the default configuration. Just choose anything else (we recommend *graylog*), because this is the default name and any Elasticsearch instance that is started in the same network will try to connect to this cluster.

The Elasticsearch servers need one IP that can be reached over network set in `network.host` and some participants of the cluster in `discovery.zen.ping.unicast.hosts`. That is enough to have a minimal cluster setup.

When you secure your Elasticsearch with User Authentication you need to add credentials to the Graylog configuration to be able to use the secured Elasticsearch cluster with Graylog.

5.5.4 Graylog Multi-node

After the installation of Graylog, you should take care that only one Graylog node is configured to be master with the configuration setting `is_master = true`.

The HTTP settings configured in `http_bind_address` (or `http_publish_uri`) must be accessable for all Graylog nodes of the cluster.

Graylog to MongoDB connection

The `mongodb_uri` configuration setting must include all MongoDB nodes forming the replica set, the name of the replica set, as well as the previously configured user account with access to the replica set. The configuration setting is a normal MongoDB connection string.

Finally, the MongoDB connection string in the Graylog configuration file should look like this:

```
mongodb_uri = mongodb://USERNAME:PASSWORD@mongodb-node01:27017,mongodb-node02:27017,
↪mongodb-node03:27017/graylog?replicaSet=rs01
```

Graylog to Elasticsearch connection

Graylog will connect to the Elasticsearch REST API.

To avoid issues with the connection to the Elasticsearch cluster you should add some of the network addresses of the Elasticsearch nodes to `elasticsearch_hosts`.

Graylog web interface

It's possible to use a *loadbalancer* in front of all Graylog servers, please refer to *Making the web interface work with load balancers/proxies* for more details.

Depending on your setup, it's possible to either use a hardware loadbalancer for TLS/HTTPS termination, a *reverse proxy*, or to simply enable it *in the Graylog node*.

5.5.5 Scaling

Each component in this multi-node setup can be scaled on the individual needs.

Depending on the amount of messages ingested and how long messages should be available for direct search, the Elasticsearch cluster will need most of the resources on your setup.

Keep an eye on the Metrics of each part of the cluster. One option is to use telegraf to fetch important metrics and store them in your favorite metric system (e. g. Graphite, Prometheus or Influx).

Elasticseach Metrics and some administration can be done with Elastic HQ or Cerebro. Those will help you to understand the Elasticsearch cluster health and behavior.

Graylog Metrics can be monitored with the Graylog Metrics Reporter plugins which are able to send the internal Graylog metrics to your favorite metrics collector (e. g. Graphite or Prometheus).

Up until today, we have almost never faced the issue that the MongoDB replica set needed special attention. But of course you should still monitor it and store its metrics - just to be sure.

5.5.6 Troubleshooting

- After every configuration change or service restart, watch the logfile of the applications you have worked on. Sometimes other log files can also give you hints about what went wrong. For example if you're configuring Graylog and try to find out why the connection to the MongoDB isn't working, the MongoDB logs can help to identify the problem.

- If *HTTPS has been enabled for the Graylog REST API*, it need to be setup for the Graylog web interface, too.

5.6 Elasticsearch

We strongly recommend to use a dedicated Elasticsearch cluster for your Graylog setup.

If you are using a shared Elasticsearch setup, a problem with indices unrelated to Graylog might turn the cluster status to YELLOW or RED and impact the availability and performance of your Graylog setup.

5.6.1 Elasticsearch versions

Starting with version 2.3, Graylog uses the HTTP protocol to connect to your Elasticsearch cluster, so it does not have a hard requirement for the Elasticsearch version anymore. We can safely assume that any version starting from 2.x is working.

Graylog version	Elasticsearch version
1.2.0-1.2.1	1.7.1
1.3.0-1.3.3	1.7.3
1.3.4	1.7.5
2.0.0	2.3.1
2.0.1-2.0.3	2.3.2
2.1.0-2.1.3	2.3.5
2.2.0-2.2.3	2.4.4
2.3.x-2.4.x	2.4.x, 5.6.x
2.5.x	2.4.x, 5.6.x, 6.8.x
3.0-3.1	5.6.x, 6.8.x

Caution: Graylog 3.x **does not** work with Elasticsearch 7.x!

Note: Graylog works fine with the Amazon Elasticsearch Service using Elasticsearch 5 or 6.

Note: Graylog works fine with the Elastic Cloud using Elasticsearch 5 or 6.

5.6.2 Configuration

Caution: As Graylog has switched from an embedded Elasticsearch node client to a lightweight HTTP client in version 2.3, please check the *upgrade notes* how to migrate your configuration if you are switching from an earlier version.

Graylog

The most important setting to make a successful connection is a list of comma-separated URIs to one or more Elasticsearch nodes. Graylog needs to know the address of at least one other Elasticsearch node given in the `elasticsearch_hosts` setting. The specified value should at least contain the scheme (`http://` for unencrypted, `https://` for encrypted connections), the hostname or IP and the port of the HTTP listener (which is `9200` unless otherwise configured) of this node. Optionally, you can also specify an authentication section containing a user name and a password, if either your Elasticsearch node uses Shield/X-Pack or Search Guard, or you have an intermediate HTTP proxy requiring authentication in between the Graylog server and the Elasticsearch node. Additionally you can specify an optional path prefix at the end of the URI.

A sample specification of `elasticsearch_hosts` could look like this:

```
elasticsearch_hosts = http://es-node-1.example.org:9200/foo,https://
↪someuser:somepassword@es-node-2.example.org:19200
```

Caution: Graylog assumes that all nodes in the cluster are running the same versions of Elasticsearch. While it might work when patch-levels differ, we highly encourage to keep versions consistent.

> **Warning:** Graylog does not react to externally triggered index changes (creating/closing/reopening/deleting an index) anymore. All of these actions need to be performed through the Graylog REST API in order to retain index consistency.

Available Elasticsearch configuration tunables

The following configuration options are now being used to configure connectivity to Elasticsearch:

Config Setting	Type	Comments	Default
`elasticsearch_connect_timeout`	Duration	Timeout when connection to individual Elasticsearch hosts	`10s` (10 Seconds)
`elasticsearch_hosts`	List<URI>	Comma-separated list of URIs of Elasticsearch hosts	`http://127.0.0.1:9200`
`elasticsearch_idle_timeout`	Duration	Timeout after which idle connections are terminated	`-1s` (Never)
`elasticsearch_max_total_connections`	int	Maximum number of total Elasticsearch connections	`20`
`elasticsearch_max_total_connections_per_route`	int	Maximum number of Elasticsearch connections per route/host	`2`
`elasticsearch_socket_timeout`	Duration	Timeout when sending/receiving from Elasticsearch connection	`60s` (60 Seconds)
`elasticsearch_discovery_enabled`	boolean	Enable automatic Elasticsearch node discovery	`false`
`elasticsearch_discovery_filter`	String	Filter by node attributes for the discovered nodes	empty (use all nodes)
`elasticsearch_discovery_frequency`	Duration	Frequency of the Elasticsearch node discovery	`30s` (30 Seconds)
`elasticsearch_compression_enabled`	boolean	Enable GZIP compression of Elasticseach request payloads	`false`

Automatic node discovery

> **Caution:** Authentication with the Elasticsearch cluster will not work if the automatic node discovery is being used.

> **Caution:** Automatic node discovery does not work when using the Amazon Elasticsearch Service because Amazon blocks certain Elasticsearch API endpoints.

Graylog uses automatic node discovery to gather a list of all available Elasticsearch nodes in the cluster at runtime and distribute requests among them to potentially increase performance and availability. To enable this feature, you need to set the `elasticsearch_discovery_enabled` to `true`. Optionally, you can define the a filter allowing to selectively include/exclude discovered nodes (details how to specify node filters are found in the Elasticsearch cluster documentation) using the `elasticsearch_discovery_filter` setting, or tuning the frequency of the node discovery using the `elasticsearch_discovery_frequency` configuration option.

Configuration of Elasticsearch nodes

Control access to Elasticsearch ports

If you are not using Shield/X-Pack or Search Guard to authenticate access to your Elasticsearch nodes, make sure to restrict access to the Elasticsearch ports (default: 9200/tcp and 9300/tcp). Otherwise the data is readable by anyone who has access to the machine over network.

Open file limits

Because Elasticsearch has to keep a lot of files open simultaneously it requires a higher open file limit that the usual operating system defaults allow. **Set it to at least 64000 open file descriptors.**

Graylog will show a notification in the web interface when there is a node in the Elasticsearch cluster which has a too low open file limit.

Read about how to raise the open file limit in the corresponding 5.x / 6.x documentation pages.

Heap size

It is strongly recommended to raise the standard size of heap memory allocated to Elasticsearch. Just set the `ES_HEAP_SIZE` environment variable to for example `24g` to allocate 24GB. We recommend to use around 50% of the available system memory for Elasticsearch (when running on a dedicated host) to leave enough space for the system caches that Elasticsearch uses a lot. But please take care that you don't cross 32 GB!

Merge throttling

> As of ES 6.2 Merge Throttling settings have been deprecated. (https://www.elastic.co/guide/en/ elasticsearch/reference/6.2/breaking_60_settings_changes.html)

Elasticsearch is throttling the merging of Lucene segments to allow extremely fast searches. This throttling however has default values that are very conservative and can lead to slow ingestion rates when used with Graylog. You would see the message journal growing without a real indication of CPU or memory stress on the Elasticsearch nodes. It usually goes along with Elasticsearch INFO log messages like this:

```
now throttling indexing
```

When running on fast IO like SSDs or a SAN we recommend to increase the value of the `indices.store.throttle.max_bytes_per_sec` in your `elasticsearch.yml` to 150MB:

```
indices.store.throttle.max_bytes_per_sec: 150mb
```

Play around with this setting until you reach the best performance.

Tuning Elasticsearch

Graylog is already setting specific configuration for every index it is managing. This is enough tuning for a lot of use cases and setups.

More detailed information about the configuration of Elasticsearch can be found in the official documentation.

5.6.3 Avoiding split-brain and shard shuffling

Split-brain events

Elasticsearch sacrifices consistency in order to ensure availability, and partition tolerance. The reasoning behind that is that short periods of misbehaviour are less problematic than short periods of unavailability. In other words, when Elasticsearch nodes in a cluster are unable to replicate changes to data, they will keep serving applications such as Graylog. When the nodes are able to replicate their data, they will attempt to converge the replicas and to achieve *eventual consistency*.

Elasticsearch tackles the previous by electing master nodes, which are in charge of database operations such as creating new indices, moving shards around the cluster nodes, and so forth. Master nodes coordinate their actions actively with others, ensuring that the data can be converged by non-masters. The cluster nodes that are not master nodes are not allowed to make changes that would break the cluster.

The previous mechanism can in some circumstances fail, causing a **split-brain event**. When an Elasticsearch cluster is split into two sides, both thinking they are the master, data consistency is lost as the masters work independently on the data. As a result the nodes will respond differently to same queries. This is considered a catastrophic event, because the data from two masters can not be rejoined automatically, and it takes quite a bit of manual work to remedy the situation.

Avoiding split-brain events

Elasticsearch nodes take a simple majority vote over who is master. If the majority agrees that they are the master, then most likely the disconnected minority has also come to conclusion that they can not be the master, and everything is just fine. This mechanism requires at least 3 nodes to work reliably however, because one or two nodes can not form a majority.

The minimum amount of master nodes required to elect a master must be configured manually in `elasticsearch.yml`:

```
# At least NODES/2+1 on clusters with NODES > 2, where NODES is the number of master
→nodes in the cluster
discovery.zen.minimum_master_nodes: 2
```

The configuration values should typically for example:

Master nodes	minimum_master_nodes	Comments
1	1	
2	1	With 2 the other node going down would stop the cluster from working!
3	2	
4	3	
5	3	
6	4	

Some of the master nodes may be *dedicated master nodes*, meaning they are configured just to handle lightweight operational (cluster management) responsibilities. They will not handle or store any of the cluster's data. The function of such nodes is similar to so called *witness servers* on other database products, and setting them up on dedicated witness sites will greatly reduce the chance of Elasticsearch cluster instability.

A dedicated master node has the following configuration in `elasticsearch.yml`:

```
node.data: false
node.master: true
```

Shard shuffling

When cluster status changes, for example because of node restarts or availability issues, Elasticsearch will start automatically rebalancing the data in the cluster. The cluster works on making sure that the amount of shards and replicas will conform to the cluster configuration. This is a problem if the status changes are just temporary. Moving shards and replicas around in the cluster takes considerable amount of resources, and should be done only when necessary.

Avoiding unnecessary shuffling

Elasticsearch has couple configuration options, which are designed to allow short times of unavailability before starting the recovery process with shard shuffling. There are 3 settings that may be configured in `elasticsearch.yml`:

```
# Recover only after the given number of nodes have joined the cluster. Can be seen
↪as "minimum number of nodes to attempt recovery at all".
gateway.recover_after_nodes: 8
# Time to wait for additional nodes after recover_after_nodes is met.
gateway.recover_after_time: 5m
# Inform ElasticSearch how many nodes form a full cluster. If this number is met,
↪start up immediately.
gateway.expected_nodes: 10
```

The configuration options should be set up so that only *minimal* node unavailability is tolerated. For example server restarts are common, and should be done in managed manner. The logic is that if you lose large part of your cluster, you probably should start re-shuffling the shards and replicas without tolerating the situation.

5.6.4 Custom index mappings

Sometimes it's useful to not rely on Elasticsearch's dynamic mapping but to define a stricter schema for messages.

Note: If the index mapping is conflicting with the actual message to be sent to Elasticsearch, indexing that message will fail.

Graylog itself is using a default mapping which includes settings for the `timestamp`, `message`, `full_message`, and `source` fields of indexed messages:

```
$ curl -X GET 'http://localhost:9200/_template/graylog-internal?pretty'
{
  "graylog-internal" : {
    "order" : -2147483648,
    "template" : "graylog_*",
    "settings" : { },
    "mappings" : {
```

(continues on next page)

```
      "message" : {
        "_ttl" : {
          "enabled" : true
        },
        "_source" : {
          "enabled" : true
        },
        "dynamic_templates" : [ {
          "internal_fields" : {
            "mapping" : {
              "index" : "not_analyzed",
              "type" : "string"
            },
            "match" : "gl2_*"
          }
        }, {
          "store_generic" : {
            "mapping" : {
              "index" : "not_analyzed"
            },
            "match" : "*"
          }
        } ],
        "properties" : {
          "full_message" : {
            "analyzer" : "standard",
            "index" : "analyzed",
            "type" : "string"
          },
          "streams" : {
            "index" : "not_analyzed",
            "type" : "string"
          },
          "source" : {
            "analyzer" : "analyzer_keyword",
            "index" : "analyzed",
            "type" : "string"
          },
          "message" : {
            "analyzer" : "standard",
            "index" : "analyzed",
            "type" : "string"
          },
          "timestamp" : {
            "format" : "yyyy-MM-dd HH:mm:ss.SSS",
            "type" : "date"
          }
        }
      }
    },
    "aliases" : { }
  }
}
```

In order to extend the default mapping of Elasticsearch and Graylog, you can create one or more custom index mappings and add them as index templates to Elasticsearch.

Let's say we have a schema for our data like the following:

Field Name	Field Type	Example
`http_method`	string	GET
`http_response_code`	long	200
`ingest_time`	date	2016-06-13T15:00:51.927Z
`took_ms`	long	56

This would translate to the following additional index mapping in Elasticsearch:

```
"mappings" : {
  "message" : {
    "properties" : {
      "http_method" : {
        "type" : "string",
        "index" : "not_analyzed"
      },
      "http_response_code" : {
        "type" : "long"
      },
      "ingest_time" : {
        "type" : "date",
        "format": "strict_date_time"
      },
      "took_ms" : {
        "type" : "long"
      }
    }
  }
}
```

The format of the `ingest_time` field is described in the Elasticsearch documentation about the format mapping parameter. Also make sure to check the Elasticsearch documentation about Field datatypes.

In order to apply the additional index mapping when Graylog creates a new index in Elasticsearch, it has to be added to an index template. The Graylog default template (`graylog-internal`) has the lowest priority and will be merged with the custom index template by Elasticsearch.

> **Warning:** If the default index mapping and the custom index mapping cannot be merged (e. g. because of conflicting field datatypes), Elasticsearch will throw an exception and won't create the index. So be extremely cautious and conservative about the custom index mappings!

Creating a new index template

Save the following index template for the custom index mapping into a file named `graylog-custom-mapping.json`:

```
{
  "template": "graylog_*",
  "mappings" : {
    "message" : {
      "properties" : {
        "http_method" : {
          "type" : "string",
          "index" : "not_analyzed"
```

(continues on next page)

```
      },
      "http_response_code" : {
        "type" : "long"
      },
      "ingest_time" : {
        "type" : "date",
        "format": "strict_date_time"
      },
      "took_ms" : {
        "type" : "long"
      }
    }
  }
 }
}
```

Finally, load the index mapping into Elasticsearch with the following command:

```
$ curl -X PUT -d @'graylog-custom-mapping.json' -H 'Content-Type: application/json'
↪'http://localhost:9200/_template/graylog-custom-mapping?pretty'
{
  "acknowledged" : true
}
```

Every Elasticsearch index created from that time on, will have an index mapping consisting of the original `graylog-internal` index template and the new `graylog-custom-mapping` template:

```
$ curl -X GET 'http://localhost:9200/graylog_deflector/_mapping?pretty'
{
  "graylog_2" : {
    "mappings" : {
      "message" : {
        "_ttl" : {
          "enabled" : true
        },
        "dynamic_templates" : [ {
          "internal_fields" : {
            "mapping" : {
              "index" : "not_analyzed",
              "type" : "string"
            },
            "match" : "gl2_*"
          }
        }, {
          "store_generic" : {
            "mapping" : {
              "index" : "not_analyzed"
            },
            "match" : "*"
          }
        } ],
        "properties" : {
          "full_message" : {
            "type" : "string",
            "analyzer" : "standard"
          },
```

　　　　　　　　　　　　　　　　　　　　　　　　　Chapter 5. Configuring Graylog

```
          "http_method" : {
            "type" : "string",
            "index" : "not_analyzed"
          },
          "http_response_code" : {
            "type" : "long"
          },
          "ingest_time" : {
            "type" : "date",
            "format" : "strict_date_time"
          },
          "message" : {
            "type" : "string",
            "analyzer" : "standard"
          },
          "source" : {
            "type" : "string",
            "analyzer" : "analyzer_keyword"
          },
          "streams" : {
            "type" : "string",
            "index" : "not_analyzed"
          },
          "timestamp" : {
            "type" : "date",
            "format" : "yyyy-MM-dd HH:mm:ss.SSS"
          },
          "took_ms" : {
            "type" : "long"
          }
        }
      }
    }
  }
}
```

Note: When using different index sets every index set can have its own mapping.

Deleting custom index templates

If you want to remove an existing index template from Elasticsearch, simply issue a `DELETE` request to Elasticsearch:

```
$ curl -X DELETE 'http://localhost:9200/_template/graylog-custom-mapping?pretty'
{
  "acknowledged" : true
}
```

After you've removed the index template, new indices will only have the original index mapping:

```
$ curl -X GET 'http://localhost:9200/graylog_deflector/_mapping?pretty'
{
  "graylog_3" : {
    "mappings" : {
```

```
  "message" : {
    "_ttl" : {
      "enabled" : true
    },
    "dynamic_templates" : [ {
      "internal_fields" : {
        "mapping" : {
          "index" : "not_analyzed",
          "type" : "string"
        },
        "match" : "gl2_*"
      }
    }, {
      "store_generic" : {
        "mapping" : {
          "index" : "not_analyzed"
        },
        "match" : "*"
      }
    } ],
    "properties" : {
      "full_message" : {
        "type" : "string",
        "analyzer" : "standard"
      },
      "message" : {
        "type" : "string",
        "analyzer" : "standard"
      },
      "source" : {
        "type" : "string",
        "analyzer" : "analyzer_keyword"
      },
      "streams" : {
        "type" : "string",
        "index" : "not_analyzed"
      },
      "timestamp" : {
        "type" : "date",
        "format" : "yyyy-MM-dd HH:mm:ss.SSS"
      }
    }
  }
}
}
```

Additional information on Elasticsearch Index Templates can be found in the official Elasticsearch Template Documentation

---

Note: Settings and index mappings in templates are only applied to new indices. After adding, modifying, or deleting an index template, you have to manually rotate the write-active indices of your index sets for the changes to take effect.

Rotate indices manually

Select the desired index set on the `System / Indices` page in the Graylog web interface by clicking on the name of the index set, then select "Rotate active write index" from the "Maintenance" dropdown menu.

5.6.5 Cluster Status explained

Elasticsearch provides a classification for the cluster health.

The cluster status applies to different levels:

- **Shard level** - see status descriptions below
- **Index level** - inherits the status of the worst shard status
- **Cluster level** - inherits the status of the worst index status

That means that the Elasticsearch cluster status can turn red if a single index or shard has problems even though the rest of the indices/shards are okay.

Note: Graylog checks the status of the current write index while indexing messages. If that one is GREEN or YELLOW, Graylog will continue to write messages into Elasticsearch regardless of the overall cluster status.

Explanation of the different status levels:

RED

The RED status indicates that some or all of the primary shards are not available.

In this state, no searches can be performed until all primary shards have been restored.

YELLOW

The YELLOW status means that all of the primary shards are available but some or all shard replicas are not.

When the index configuration include replication with a count that is equal or higher than the number of nodes, your cluster cannot become green. In most cases, this can be solved by adding another Elasticsearch node to the cluster or by reducing the replication factor of the indices.

GREEN

The cluster is fully operational. All primary and replica shards are available.

5.7 Index model

5.7.1 Overview

Graylog is transparently managing one or more sets of Elasticsearch indices to optimize search and analysis operations for speed and low resource consumption.

To enable managing indices with different mappings, analyzers, and replication settings Graylog is using so-called index sets which are an abstraction of all these settings.

Each index set contains the necessary settings for Graylog to create, manage, and fill Elasticsearch indices and handle index rotation and data retention for specific requirements.

Graylog is maintaining an index alias per index set which is always pointing to the current write-active index from that index set. There is always exactly one index to which new messages are written until the configured rotation criterion (number of documents, index size, or index age) has been met.

A background task continuously checks if the rotation criterion of an index set has been met and a new index is created and prepared when that happens. Once the index is ready, the index alias is atomically switched to it. That means that all Graylog nodes can write messages into the alias without even knowing what the currently write-active index of the index set is.
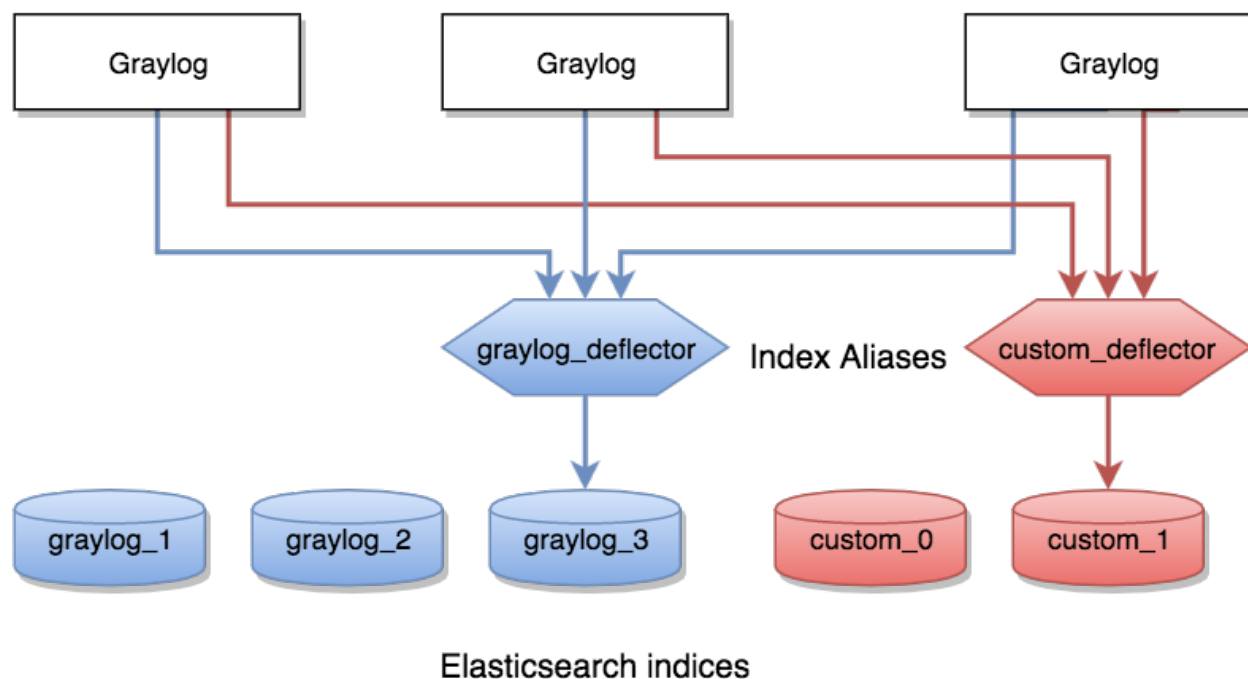
Almost every read operation is performed with a given time range. Because Graylog is writing messages sequentially into Elasticsearch it can keep information about the time range each index covers. It selects a lists of indices to query when having a time range provided. If no time range was provided, it will search in all indices it knows.

Read Path
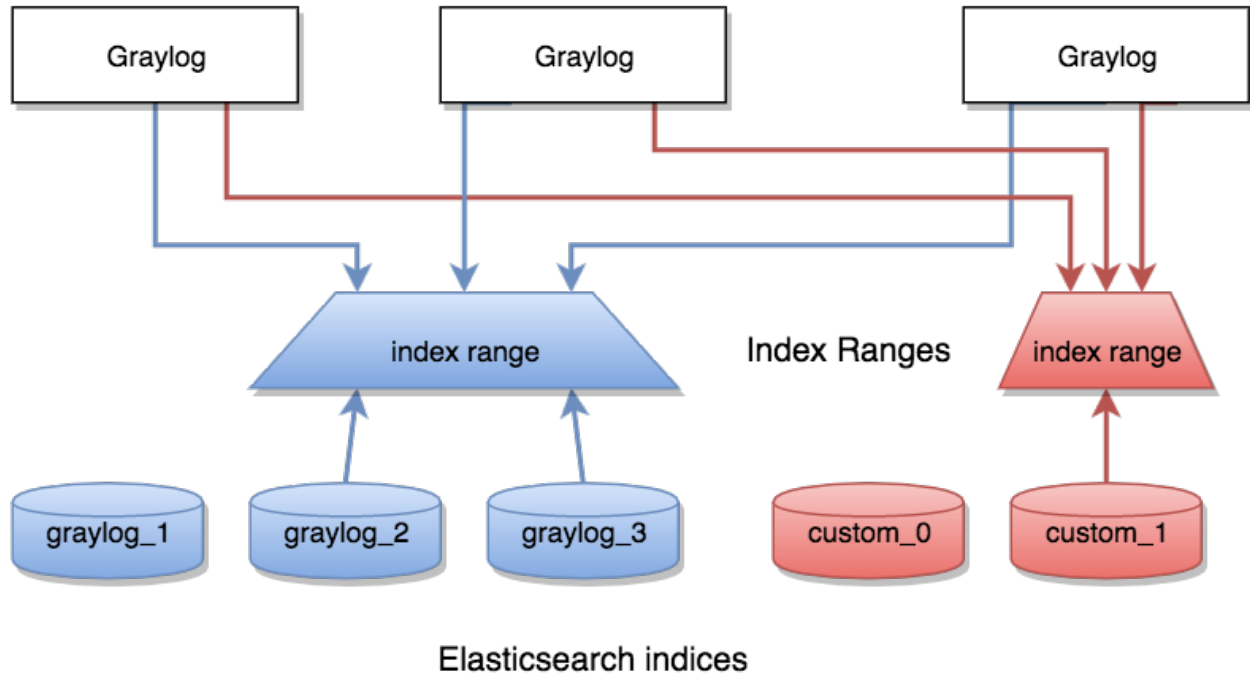
Eviction of indices and messages

There are configuration settings for the maximum number of indices Graylog is managing in a given index set.

Depending on the configured retention strategy, the oldest indices of an index set will automatically be closed, deleted, or exported when the configured maximum number of indices has been reached.

The deletion is performed by the Graylog master node in a background thread which is continuously comparing the number of indices with the configured maximum:

```
INFO : org.graylog2.indexer.rotation.strategies.AbstractRotationStrategy - Deflector
→index <graylog_95> should be rotated, Pointing deflector to new index now!
INFO : org.graylog2.indexer.MongoIndexSet - Cycling from <graylog_95> to <graylog_96>.
INFO : org.graylog2.indexer.MongoIndexSet - Creating target index <graylog_96>.
INFO : org.graylog2.indexer.indices.Indices - Created Graylog index template "graylog-
→internal" in Elasticsearch.
INFO : org.graylog2.indexer.MongoIndexSet - Waiting for allocation of index <graylog_
→96>.
INFO : org.graylog2.indexer.MongoIndexSet - Index <graylog_96> has been successfully
→allocated.
INFO : org.graylog2.indexer.MongoIndexSet - Pointing index alias <graylog_deflector>
→to new index <graylog_96>.
INFO : org.graylog2.system.jobs.SystemJobManager - Submitted SystemJob <f1018ae0-dcaa-
→11e6-97c3-6c4008b8fc28> [org.graylog2.indexer.indices.jobs.
→SetIndexReadOnlyAndCalculateRangeJob]
INFO : org.graylog2.indexer.MongoIndexSet - Successfully pointed index alias <graylog_
→deflector> to index <graylog_96>.
```

5.7.2 Index Set Configuration

Index sets have a variety of different settings related to how Graylog will store messages into the Elasticsearch cluster.

- **Title**: A descriptive name of the index set.

- **Description**: A description of the index set for human consumption.

- **Index prefix**: A unique prefix used for Elasticsearch indices managed by the index set. The prefix must start with a letter or number, and can only contain letters, numbers, _, - and +. The index alias will be named accordingly, e. g. `graylog_deflector` if the index prefix was `graylog`.

- **Analyzer**: (default: `standard`) The Elasticsearch analyzer for the index set.

- **Index shards**: (default: 4) The number of Elasticsearch shards used per index.

- **Index replicas**: (default: 0) The number of Elasticsearch replicas used per index.

- **Max. number of segments**: (default: 1) The maximum number of segments per Elasticsearch index after index optimization (force merge), see Segment Merging for details.

- **Disable index optimization after rotation**: Disable Elasticsearch index optimization (force merge) after index rotation. Only activate this if you have serious problems with the performance of your Elasticsearch cluster during the optimization process.

Index rotation

- **Message count**: Rotates the index after a specific number of messages have been written.
- **Index size**: Rotates the index after an approximate size on disk (before optimization) has been reached.
- **Index time**: Rotates the index after a specific time (e. g. 1 hour or 1 week).

Index Rotation Configuration
Graylog uses multiple indices to store documents in. You can configure the strategy it uses to determine when to rotate the currently active write index.

Select rotation strategy

Index Message Count   ✕ ▾

Max documents per index

20000000

Maximum number of documents in an index before it gets rotated

Index retention

- **Delete**: Delete indices in Elasticsearch to minimize resource consumption.
- **Close**: Close indices in Elasticsearch to reduce resource consumption.
- **Do nothing**
- **Archive**: Commercial feature, see *Archiving*.

Index Retention Configuration
Graylog uses a retention strategy to clean up old indices.

Select retention strategy

Delete Index   ✕ ▾

Max number of indices

20

Maximum number of indices to keep before **deleting** the oldest ones

5.7.3 Maintenance

Keeping the index ranges in sync

Graylog will take care of calculating index ranges automatically as soon as a new index has been created.

In case the stored metadata about index time ranges has run out of sync, Graylog will notify you in the web interface. This can happen if an index was deleted manually or messages from already "closed" indices were removed.

The system will offer you to just re-generate all time range information. This may take a few seconds but is an easy task for Graylog.

You can easily re-build the information yourself after manually deleting indices or doing other changes that might cause synchronization problems:

```
$ curl -XPOST http://127.0.0.1:9000/api/system/indices/ranges/rebuild
```

This will trigger a system job:

```
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Recalculating index ranges.
INFO : org.graylog2.system.jobs.SystemJobManager - Submitted SystemJob <9b64a9d0-dcac-
→11e6-97c3-6c4008b8fc28> [org.graylog2.indexer.ranges.RebuildIndexRangesJob]
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Recalculating index ranges
→for index set Default index set (graylog2_*): 5 indices affected.
INFO : org.graylog2.indexer.ranges.MongoIndexRangeService - Calculated range of
→[graylog_96] in [7ms].
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Created ranges for index
→graylog_96: MongoIndexRange{id=null, indexName=graylog_96, begin=2017-01-
→17T11:49:02.529Z, end=2017-01-17T12:00:01.492Z, calculatedAt=2017-01-17T12:00:58.
→097Z, calculationDuration=7, streamIds=[000000000000000000000001]}
[...]
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Done calculating index
→ranges for 5 indices. Took 44ms.
INFO : org.graylog2.system.jobs.SystemJobManager - SystemJob <9b64a9d0-dcac-11e6-97c3-
→6c4008b8fc28> [org.graylog2.indexer.ranges.RebuildIndexRangesJob] finished in 46ms.
```

Manually rotating the active write index

Sometimes you might want to rotate the active write index manually and not wait until the configured rotation criterion for in the latest index has been met, for example if you've changed the index mapping or the number of shards per index.

You can do this either via an HTTP request against the REST API of the Graylog master node or via the web interface:

```
$ curl -XPOST http://127.0.0.1:9000/api/system/deflector/cycle
```


Triggering this job produces log output similar to the following lines:

```
INFO : org.graylog2.rest.resources.system.DeflectorResource - Cycling deflector for
→index set <58501f0b4a133077ecd134d9>. Reason: REST request.
INFO : org.graylog2.indexer.MongoIndexSet - Cycling from <graylog_97> to <graylog_98>.
INFO : org.graylog2.indexer.MongoIndexSet - Creating target index <graylog_98>.
INFO : org.graylog2.indexer.indices.Indices - Created Graylog index template "graylog-
→internal" in Elasticsearch.
INFO : org.graylog2.indexer.MongoIndexSet - Waiting for allocation of index <graylog_
→98>.
INFO : org.graylog2.indexer.MongoIndexSet - Index <graylog_98> has been successfully
→allocated.
INFO : org.graylog2.indexer.MongoIndexSet - Pointing index alias <graylog_deflector>
→to new index <graylog_98>.
INFO : org.graylog2.system.jobs.SystemJobManager - Submitted SystemJob <024aac80-dcad-
→11e6-97c3-6c4008b8fc28> [org.graylog2.indexer.indices.jobs.
→SetIndexReadOnlyAndCalculateRangeJob]
```

(continues on next page)

```
INFO : org.graylog2.indexer.MongoIndexSet – Successfully pointed index alias <graylog_
↪deflector> to index <graylog_98>.
INFO : org.graylog2.indexer.retention.strategies.
↪AbstractIndexCountBasedRetentionStrategy – Number of indices (5) higher than limit␣
↪(4). Running retention for 1 index.
INFO : org.graylog2.indexer.retention.strategies.
↪AbstractIndexCountBasedRetentionStrategy – Running retention strategy [org.graylog2.
↪indexer.retention.strategies.DeletionRetentionStrategy] for index <graylog_94>
INFO : org.graylog2.indexer.retention.strategies.DeletionRetentionStrategy – Finished␣
↪index retention strategy [delete] for index <graylog_94> in 23ms.
```

5.8 Backup

When it comes to backup in a Graylog setup it is not easy to answer. You need to consider what type of backup will suit your needs.

Your Graylog Server setup and settings are easy to backup with a MongoDB dump and a filesystem backup of all configuration files.

The data within your Elasticsearch Cluster can take the advantage of the Snapshot and Restore function that are offered by Elasticsearch.

5.8.1 Disaster recovery

To be able to restore Graylog after a total System crash you need the Graylog `server.conf` file - to be exact you need the key you used for `password_secret` in the configuration. The second important part is the MongoDB. This database contains all configuration. Possible options how-to backup MongoDB can be found at the MongoDB documentation.

If you need to restore log data, you can do this using the archiving feature of Graylog enterprise or any other elasticsearch backup and restore option. It is not enough to copy the data directories of your Elasticsearch nodes, you might not be able to restore from that.

Elasticsearch and MongoDB are databases, for both you should implement the ability to make a data dump and restore that - if you need want to be able to restore the current state.

5.9 Default file locations

Each installation flavor of Graylog will place the configuration files into a specific location on the local files system. The goal of this section is to provide a short overview about the most common and most important default file locations.

5.9.1 DEB package

This paragraph covers Graylog installations on Ubuntu Linux, Debian Linux, and Debian derivates installed with the *DEB package*.

Graylog

	File system path
Configuration	`/etc/graylog/server/server.conf`
Logging configuration	`/etc/graylog/server/log4j2.xml`
Plugins	`/usr/share/graylog-server/plugin`
Binaries	`/usr/share/graylog-server/bin`
Scripts	`/usr/share/graylog-server/scripts`
JVM settings	`/etc/default/graylog-server`
Message journal files	`/var/lib/graylog-server/journal`
Log Files	`/var/log/graylog-server/`

Elasticsearch

Note: These are only the most common file locations. Please refer to the Elasticsearch documentation for a comprehensive list of default file locations.

	File system path
Configuration	`/etc/elasticsearch`
JVM settings	`/etc/default/elasticsearch`
Data files	`/var/lib/elasticsearch/data`
Log files	`/var/log/elasticsearch/`

MongoDB

	File system path
Configuration	`/etc/mongod.conf`
Data files	`/var/lib/mongodb/`
Log files	`/var/log/mongodb/`

5.9.2 RPM package

This paragraph covers Graylog installations on Fedora Linux, Red Hat Enterprise Linux, CentOS Linux, and other Red Hat Linux derivates installed with the *RPM package*.

Graylog

	File system path
Configuration	`/etc/graylog/server/server.conf`
Logging configuration	`/etc/graylog/server/log4j2.xml`
Plugins	`/usr/share/graylog-server/plugin`
Binaries	`/usr/share/graylog-server/bin`
Scripts	`/usr/share/graylog-server/scripts`
JVM settings	`/etc/sysconfig/graylog-server`
Message journal files	`/var/lib/graylog-server/journal`
Log Files	`/var/log/graylog-server/`

Elasticsearch

Note: These are only the most common file locations. Please refer to the Elasticsearch documentation for a comprehensive list of default file locations.

	File system path
Configuration	`/etc/elasticsearch`
JVM settings	`/etc/sysconfig/elasticsearch`
Data files	`/var/lib/elasticsearch/`
Log files	`/var/log/elasticsearch/`

MongoDB

	File system path
Configuration	`/etc/mongod.conf`
Data files	`/var/lib/mongodb/`
Log files	`/var/log/mongodb/`

5.10 Graylog REST API

The functionality Graylog REST API is very comprehensive; even the Graylog web interface is exclusively using Graylog REST API to interact with the Graylog cluster.

To connect to the Graylog REST API with a web browser, just add `api/api-browser` to your current `http_publish_uri` setting or use the **API browser** button on the nodes overview page (*System / Nodes* in the web interface).

For example if your Graylog REST API is listening on `http://192.168.178.26:9000/api/`, the API browser will be available at `http://192.168.178.26:9000/api/api-browser/`.

Note: The customized version of Swagger UI used by Graylog does currently only work in Google Chrome and Firefox.

5.10.1 Using the API browser

After providing the credentials (username and password), you can browse all available HTTP resources of the Graylog REST API.

5.10.2 Interacting with the Graylog REST API

While having a graphical UI for the Graylog REST API is perfect for interactive usage and exploratory learning, the real power unfolds when using the Graylog REST API for automation or integrating Graylog into another system, such as monitoring or ticket systems.

Naturally, the same operations the API browser offers can be used on the command line or in scripts. A very common HTTP client being used for this kind of interaction is curl.

Note: In the following examples, the username `GM` and password `superpower` will be used to demonstrate how to work with the Graylog REST API running at `http://192.168.178.26:9000/api`.

Warning: Since Graylog 2.5.0, all non-GET API requests **must include and set a value** for the `X-Requested-By` HTTP header. This is needed to prevent CSRF attacks.

The following command displays Graylog cluster information as JSON, exactly the same information the web interface is displaying on the *System / Nodes* page:

```
curl -u GM:superpower -H 'Accept: application/json' -X GET 'http://192.168.178.
↪26:9000/api/cluster?pretty=true'
```

The Graylog REST API will respond with the following information:

```
{
  "71ab6aaa-cb39-46be-9dac-4ba99fed3d66" : {
    "facility" : "graylog-server",
    "codename" : "Smuttynose",
    "node_id" : "71ab6aaa-cb39-46be-9dac-4ba99fed3d66",
    "cluster_id" : "3adaf799-1551-4239-84e5-6ed939b56f62",
    "version" : "2.1.1+01d50e5",
    "started_at" : "2016-09-23T10:39:00.179Z",
    "hostname" : "gm-01-c.fritz.box",
    "lifecycle" : "running",
    "lb_status" : "alive",
    "timezone" : "Europe/Berlin",
    "operating_system" : "Linux 3.10.0-327.28.3.el7.x86_64",
    "is_processing" : true
  },
  "ed0ad32d-8776-4d25-be2f-a8956ecebdcf" : {
    "facility" : "graylog-server",
    "codename" : "Smuttynose",
    "node_id" : "ed0ad32d-8776-4d25-be2f-a8956ecebdcf",
    "cluster_id" : "3adaf799-1551-4239-84e5-6ed939b56f62",
    "version" : "2.1.1+01d50e5",
    "started_at" : "2016-09-23T10:40:07.325Z",
    "hostname" : "gm-01-d.fritz.box",
    "lifecycle" : "running",
    "lb_status" : "alive",
    "timezone" : "Europe/Berlin",
    "operating_system" : "Linux 3.16.0-4-amd64",
    "is_processing" : true
  },
  "58c57924-808a-4fa7-be09-63ca551628cd" : {
    "facility" : "graylog-server",
```

(continues on next page)

```
    "codename" : "Smuttynose",
    "node_id" : "58c57924-808a-4fa7-be09-63ca551628cd",
    "cluster_id" : "3adaf799-1551-4239-84e5-6ed939b56f62",
    "version" : "2.1.1+01d50e5",
    "started_at" : "2016-09-30T13:31:39.051Z",
    "hostname" : "gm-01-u.fritz.box",
    "lifecycle" : "running",
    "lb_status" : "alive",
    "timezone" : "Europe/Berlin",
    "operating_system" : "Linux 4.4.0-36-generic",
    "is_processing" : true
}
```

Creating and using Access Token

For security reasons, using the username and password directly on the command line or in some third party application is undesirable.

To prevent having to use the clear text credentials, Graylog allows to create access tokens which can be used for authentication instead.

Note: Users require the permissions `users:tokenlist`, `users:tokencreate`, and `users:tokenremove` to manage their access tokens. Please check the documentation on *Permission system* for more information. Also note that users, even administrators, may only manage their *own* tokens.

The following example will create an access token named `agents` for the user `graylog-sidecar`:

- Navigate to the users configuration menu `System / Authentication`.

- Select the user you want to create a token for and click on `Edit tokens`.

- Give the token a name and create it.

- You should see now the token in the list.

Either by unchecking the hide option or by copying the token to the clipboard you can access the token. The received access token can now be used as username in a request to the Graylog REST API using Basic Auth together with the literal password `token`.

When an access token is no longer needed, it can be delete on the Graylog UI via the `Delete` button.

Creating and using Session Token

While access tokens can be used for permanent access, session tokens will expire after a certain time. The expiration time can be adjusted in the user's profile.

Getting a new session token can be obtained via `POST` request to the Graylog REST API. Username and password are required to get a valid session ID. The following example will create an session token for the user `GM`:

```
curl -i -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -H
→'X-Requested-By: cli' 'http://192.168.178.26:9000/api/system/sessions' -d '{
→"username":"GM", "password":"superpower", "host":""}'
```

The response will include the session token in the field `session_id` and the time of expiration:

```
{
    "valid_until" : "2016-10-24T16:08:57.854+0000",
    "session_id" : "cf1df45c-53ea-446c-8ed7-e1df64861de7"
}
```

The received token can now be used as username in a request to the Graylog REST API using Basic Auth together with the literal password `session`.

Now a `curl` command to get a list of access tokens would look as follows:

```
curl -u cf1df45c-53ea-446c-8ed7-e1df64861de7:session -H 'Accept: application/json' -X
→GET 'http://192.168.178.26:9000/api/cluster?pretty=true'
```

SECURING GRAYLOG

To secure your Graylog setup, you should not use one of our pre-configured images, create your own unique installation where you understand each component and secure the environment by design. Expose only the services that are needed and secure them whenever possible with TLS/SSL and some kind of authentication. Do not use the pre-created appliances for critical production environments.

On the Graylog appliances MongoDB and Elasticsearch is listening on the external interface. This makes the creation of a cluster easier and demonstrates the way Graylog works. Never run this in an insecure network.

When using Amazon Web Services and our pre-configured AMI, never open all ports in the security group. Do not expose the server to the internet. Access Graylog only from within your VPC. Enable encryption for the communication.

6.1 Default ports

All parts of one Graylog installation will communicate over network sockets. Depending on your setup and number of nodes this might be exposed or can be bound to localhost. The *SELinux* configuration is already covered in our step-by-step guide for CentOS Linux.

Table 1: Default network communication ports

Component	Port
Graylog (web interface / API)	9000 (tcp)
Graylog to Elasticsearch	9200 (tcp)
Elasticsearch node communication	9300 (tcp)
MongoDB	27017 (tcp)

Each setup is unique in the requirements and ports might be changed by configuration, but you should limit who is able to connect to which service. In the *architecture description* you can see what components need to be exposed and communicate with each other.

6.2 Configuring TLS ciphers

When running Graylog in untrusted environments such as the Internet, we strongly recommend to use SSL/TLS for all connections.

It's possible to *disable unsafe or deprecated TLS ciphers* in Graylog. When using *nginx or Apache httpd* for SSL termination the Mozilla SSL Configuration Generator will help to create a reasonably secure configuration for them.

6.3 Security related topics

6.3.1 Generating Graylog certificates and keys with Microsoft AD CS

In order to really make your Graylog installation "your own" Graylog, you will need to add TLS certificates issued and trusted by your own organization. Many organizations rely upon Microsoft's ADCS (Active Directory Certificate Services) for the issuance of their internal certificates. Here, we will explain the basic requirements and workflow of setting up all keys and certificates for a Graylog stack.

In these examples we will assume a Graylog cluster, consisting of:

- One Graylog front-end server.

- Three Graylog data receiver hosts (clustered).

- Three ElasticSearch instances (clustered).

- SearchGuard, used to apply TLS/SSL for free on ElasticSearch.

- Three MongoDB instances (clustered as *replica set*)

Required certificates

In order to provide your full cluster with all required certificates, we'll need to make all of the following keypairs and certificates. For each certificate you'll need to gather the relevant hostnames, DNS aliases and IP addresses, because we want the certificates to work for all of these.

Table 2: Required certificates and key files

Goal	Subject	Subject alt. names	Filetype
MongoDB 1	CN=hostname1.mydomain.local	dns:hostname1 dns:hostname1.mydomain.local dns: graylogmongoalias.mydomain.local dns:graylogmongoalias ip:192.168.100.101	PEM PKCS#12 (cert+key in one)
MongoDB 2	CN=hostname2.mydomain.local	dns:hostname2 dns:hostname12mydomain.local dns: graylogmongoalias.mydomain.local dns:graylogmongoalias ip:192.168.100.102	PEM PKCS#12 (cert+key in one)
MongoDB 3	CN=hostname3.mydomain.local	dns:hostname3 dns:hostname3.mydomain.local dns: graylogmongoalias.mydomain.local dns:graylogmongoalias ip:192.168.100.103	PEM PKCS#12 (cert+key in one)
Graylog frontend	CN=hostname4.mydomain.local	dns:hostname4 dns:hostname4.mydomain.local dns:graylogguialias.mydomain.local dns:graylogguialias ip:192.168.100.104	PEM (certificate) PKCS#8 (key file)
Graylog receiver 1	CN=hostname5.mydomain.local	dns:hostname5 dns:hostname5.mydomain.local dns:graylogreceiveralias.mydomain.local dns: graylogreceiveralias ip:192.168.100.105	PEM (certificate) PKCS#8 (key file)
Graylog receiver 2	CN=hostname6.mydomain.local	dns:hostname6 dns:hostname6.mydomain.local dns:graylogreceiveralias.mydomain.local dns: graylogreceiveralias ip:192.168.100.106	PEM (certificate) PKCS#8 (key file)
Graylog receiver 3	CN=hostname7.mydomain.local	dns:hostname7 dns:hostname7.mydomain.local dns:graylogreceiveralias.mydomain.local dns: graylogreceiveralias ip:192.168.100.107	PEM (certificate) PKCS#8 (key file)
Search-Guard Admin	CN=searchguardadmin,O=yourorganization		PEM (certificate) PKCS#8 (key file)
Elastic-Search 1	CN=hostname8.mydomain.local	dns:hostname8 dns:hostname8.mydomain.local dns: graylogelasticalias.mydomain.local dns:graylogelasticalias ip:192.168.100.108	PEM (certificate) PKCS#8 (key file)
Elastic-Search 2	CN=hostname9.mydomain.local	dns:hostname9 dns:hostname9.mydomain.local dns: graylogelasticalias.mydomain.local dns:graylogelasticalias ip:192.168.100.109	PEM (certificate) PKCS#8 (key file)
Elastic-Search 3	CN=hostname9.mydomain.local	dns:hostname9 dns:hostname9.mydomain.local dns: graylogelasticalias.mydomain.local dns:graylogelasticalias ip:192.168.100.109	PEM (certificate) PKCS#8 (key file)

Graylog stack certificate template

The certificates for the Graylog stack and all of its components need some pretty specific settings. In order to achieve these, you will need to define a new certificate template in ADCS.

Be careful:

- Defining a new certicate template will require elevated privileges in your Active Directory domain.

- PKI and certificates are a matter of trust! Do not break your organization's *Certificate Policy* or its *Certificate practice Statement*. Stick to your standard procedures and do not simply start messing with the PKI!

Defining the new template is done through the ADCS management tool "*Certification Authority*".

1. Duplicate the default ADCS WebServer template, rename it to your liking.

2. General tab:

    1. Set the name to something recognizable, for example "Graylog Stack Template".

    2. The software will automatically generate the internal name, which removes all spaces: "GraylogStack-Template".

3. Cryptography tab:

    1. **Provider** is the *Key Storage Provider*

    2. **Requests can use any provider available on the subject's computer** is true

    3. **Algorithm** is *RSA 2048*

    4. **Request hash** is *SHA256*

    5. **Use alternate signature hash** must be set to false.

4. Extensions tab:

    1. **Application policies** is set to both *server auth* as well as *client auth*.

5. Request handling tab:

Note: If you are going to be generating all the keypairs on your issuing CA or on another management station, then you will need to add the following as well, which will allow you to export the keypair for migration to the Graylog stack servers.

---

    1. **Allow the private key to be exported** is set to *Yes*.

Generating the keypair and certificates - preparation

The following instructions assume that you generate all the keypairs on a Windows administrative workstation, or on the issuing CA itself (meaning, you'll need that extra "*Allow the private key to be exported*" flag). You can of course generate all keys on the Graylog stack servers and then simply submit the CSR (certificate signing request) to the CA.

The .INF input file for the *certreq* command would look similar to the following. Note that we are referring to the template by the internal name, which does not have whitespace!:

```
[Version]
signature="$Windows NT$"
[NewRequest]
Subject="CN=hostname5.mydomain.local"
HashAlgorithms=SHA256
Keyalgorithm=RSA
KeyLength=2048
Exportable=True
MachineKeySet=True
[RequestAttributes]
CertificateTemplate="GraylogStackTemplate"
[Extensions]
2.5.29.17="{text}"
_continue_="dns=hostname5&"
_continue_="dns=hostname5.mydomain.local&"
_continue_="dns=graylogreceiveralias.mydomain.local&"
_continue_="dns=graylogreceiveralias&"
_continue_="ipaddress=192.168.100.105&"
```

If you're one of the edge-cases where you will be using an older *Internet Explorer* to talk to the IP address of the host, as opposed to the hostname or its alias, you will need to add:

```
_continue_="dns=192.168.100.105&"
```

For some reason IExplore ignores the *ipaddress* field of the SAN (subject alternative name).

The above is only one of the needed .INF files; you will need one for each keypair being generated! So adjust all relevant fields and save each .INF file separately.

Generating the keypair and certificates - execution

As said, we're assuming that you're generating the keypairs on your Windows administration station. If you're generating the keypairs on the Graylog Linux hosts, then you will need to use different instructions.

For each of the .INF files that we built, we will run commands like the following (assuming that the files are all in D:secretsgraylog):

```
certreq -new D:\secrets\graylog\hostname5-graylogreceiver.inf D:\secrets\graylog\
→hostname5-graylogreceiver.req
certreq -submit D:\secrets\graylog\hostname5-graylogreceiver.req
```

This gives you a request ID, for example "531". Ask one of your PKI administrators to approve the request, for example:

```
certutil -resubmit 531
```

Afterwards you can continue:

```
certreq -retrieve 531 D:\secrets\graylog\hostname5-graylogreceiver.cer
certreq -accept D:\secrets\graylog\hostname5-graylogreceiver.cer
```

What all of this does is:

1. Generate a keypair by your specifications.

2. Generate a CSR for the keypair.

3. Submit the CSR to the issuing CA.

4. Approve the CSR on the issuing CA.

5. Export the signed certificate from the issuing CA.

6. Import the signed certificate into your current server's certificate store.

SearchGuard admin

SearchGuard is used to add TLS/SSL encryption onto ElasticSearch for free. The product requires that the admin-user authenticates using a keypair and certificate. The generation process is similar to the one above, except that you won't be adding SANs, because the user does not have DNS names or IP addresses. The subject name will understandably also be different (e.g. *CN=searchguardadmin,OU=yourteam,O=yourorganization*), but be warned that it must match exactly with the account name in the SearchGuard configuration.

Generating the keypair and certificates - conversion

We showed earlier (in the table above) that each part of the Graylog stack has specific requirements for the format and files that are used to submit the keypair and the certificate. We will need to convert everything we have right now, in order to make them usable.

> **Warning:** Key materials are very sensitive information! You should not leave them lying around! Once you have finished the setup of all keys and certificates on the Graylog stack, you must delete all the files we've put into D:secretsgraylog. Never leave key materials lying around!

Also, please use strong passwords on all PFX and PKCS files! Store these passwords safely, in a password vaulting application.

CA Chain

Each application requires that you provide the CA chain of your PKI, for inclusion in its trust store. The following assumes that you have one root CA and one issuing CA and that you've put their respective certificates in D:secretsgraylog:

```
openssl x509 -in rootca.crt -outform pem -out D:\secrets\graylog\rootca.pem
openssl x509 -in ca.crt -outform pem -out D:\secrets\graylog\ca.pem
type D:\secrets\graylog\rootca.pem > D:\secrets\graylog\cachain.pem
type D:\secrets\graylog\rootca.pem >> D:\secrets\graylog\cachain.pem
```

The resulting cachain.pem file can be used in all Graylog stack applications for inclusion in the trust store. You will probably need to run the file through **dos2unix** first though, to fix line endings.

MongoDB

For each of the keypairs we made we will need to repeat the following in Powershell (adjust all names accordingly):

```
Get-ChildItem -Path cert:\\LocalMachine\My | Select-String hostname3
```

This will return metadata of the certificate for MongoDB on hostname3. You will need the thumbprint string, which will look similar to "*5F98EBBFE735CDDAE00E33E0FD69050EF9220254*". Moving on:

```
$mypass = ConvertTo-SecureString -String "yoursafepassword" -Force -AsPlainText
Get-ChildItem -Path cert:\\LocalMachine\My\5F98EBBFE735CDDAE00E33E0FD69050EF9220254 |␣
→Export-PfxCertificate -FilePath D:\secrets\graylog\hostname3-mongodb.pfx -Password␣
→$mypass
openssl x509 -in D:\secrets\graylog\hostname3-mongodb.cer -outform pem -out D:\
→secrets\graylog\hostname3-mongodb.crt
openssl pkcs12 -in D:\secrets\graylog\hostname3-mongodb.pfx -nocerts -out D:\secrets\
→graylog\hostname3-mongodb.key
type D:\secrets\graylog\hostname3-mongodb.crt > D:\secrets\graylog\hostname3-mongodb.
→pem
D:\secrets\graylog\hostname3-mongodb.key >> D:\secrets\graylog\hostname3-mongodb.pem
```

Finally, edit the PEM file D:secretsgrayloghostname3-mongodb.pem to remove all extraneous metadata and whitespaces. There should be nothing separating the === *END CERTIFICATE* === and the === *BEGIN PRIVATE KEY* === headers.

You may upload the PEM file to the relevant MongoDB server, where you will need to do one final conversion: use **dos2unix** to convert the line endings from Windows-type to Linux-type.

Graylog and ElasticSearch

For each of the keypairs we made we will need to repeat the following in Powershell (adjust all names accordingly):

```
Get-ChildItem -Path cert:\\LocalMachine\My | Select-String hostname5
```

This will return metadata of the certificate for MongoDB on hostname5. You will need the thumbprint string, which will look similar to "*5F98EBBFE735CDDAE00E33E0FD69050EF9220254*". Moving on:

```
$mypass = ConvertTo-SecureString -String "yoursafepassword" -Force -AsPlainText
Get-ChildItem -Path cert:\\LocalMachine\My\5F98EBBFE735CDDAE00E33E0FD69050EF9220254 |␣
↪Export-PfxCertificate -FilePath D:\secrets\graylog\hostname5-receiver.pfx -Password
↪$mypass
openssl x509 -in D:\secrets\graylog\hostname5-receiver.cer -outform pem -out D:\
↪secrets\graylog\hostname5-receiver.crt
openssl pkcs12 -in D:\secrets\graylog\hostname5-receiver.pfx -nocerts -out D:\secrets\
↪graylog\hostname5-receiver.key
openssl pkcs8 -in D:\secrets\graylog\hostname5-receiver.key -topk8 -out D:\secrets\
↪graylog\hostname5-receiver.pem
```

Finally, edit the CRT and PEM files to remove all extraneous metadata and whitespaces. There should be nothing before or after the **=== BEGIN** and **END ===** tags.

You may upload the PEM and CRT files to the relevant ElasticSearch or Graylog server, where you will need to do one final conversion: use **dos2unix** to convert the line endings from Windows-type to Linux-type.

6.3.2 Secured Graylog and Beats input

The goal of this guide is to have a secured Graylog interface, API and secure communication for Beats that are authenticated by certificate. This way only trusted sources are able to deliver messages into Graylog.

This is a structured document that contains only information already given at various location in this documentation. It should give the missing connection between the different parts of the documentation.

SSL/TLS prework

Create a CA with our shadowCA or use your already given CA. That is needed to create all certificates. The examples will take the given names from our shadowCA and reference to that only, please adjust this to your local needs. If in doubt check the shadowCA scripts what kind of certificate is created and used.

The CA certificate needs to be imported on all machines that are part of the setup using the documented steps. Depending on your Browser you might need to import the `.der` to your Browser to trust the CA. In addition the CA `.der` file is imported to a JVM Keystore that is used by Graylog.

Adding of .der to JVM Keystore

Graylog needs to know the CA that is used to verify the certificates. The prime advantage is that it only needs the CA certificate and not all known self-signed certificates in the setup.:

```
# test the .der file
keytool -v -printcert -file shadowCA.der

# copy cacert into Graylog Folder (ubuntu / debian and CENTOS openJDK )
[ -f /usr/lib/jvm/jre/lib/security/cacerts ] && cp /usr/lib/jvm/jre/lib/security/
↪cacerts /etc/graylog/server/cacerts.jks
[ -f /usr/lib/jvm/java-8-openjdk-amd64/jre/lib/security/cacerts ] && cp /usr/lib/jvm/
↪java-8-openjdk-amd64/jre/lib/security/cacerts /etc/graylog/server/cacerts.jks
```

(continues on next page)

```
# import CA .der into keystore
# will only work if the default password & user is not changed.
keytool -importcert -alias shadowCA -keystore /etc/graylog/server/cacerts.jks -
↪storepass changeit -file shadowCA.der
```

Custom JVM Keystore for Graylog

Modify the *JVM Setting* to include `-Djavax.net.ssl.trustStore=/etc/graylog/server/cacerts.jks` in the `GRAYLOG_JAVA_OPTS`.

Create certificates

Create certificates for each server, all hostnames and IPs that might be used later to connect from and to this server should be included in the certificates. See README of shadowCA for the possible options. The most common error is that the certificate name does not match the hostname that is used for the connection.

The shadowCA uses the same settings that can be found *in the SSL* documentation, but easy up the process.

Deploy and configure

Graylog

HTTPS

Place the `.key` and `.crt` file on your Graylog server in the configuration dir (/etc/graylog/server/) and add them to the Graylog server.conf. In addition change `http_enable_tls` to **true**. You might need to cover other settings in a multinode cluster or special setups - just read the comments of the settings inside of the server.conf.

When using the Sidecar, use the **https** URI in the *sidecar.yml*

After restart of Graylog the web interface and the API is served via https only. No automatic redirect from http to https is made.

TLS Beats Input

To enable TLS on the input, a certificate (and private key file) is needed. It can be the same or a different certificate as the one of your REST/web interface, as long as it matches all hostnames of your input. Just reference the files *TLS cert file* and *TLS private key file* in the Beats Input configuration and restart the input.

The ingesting client will verify the presented certificate against his know CA certificates, if that is successful communication will be establised using TLS.

Add client authentication to Beats input

Create one directory (`/etc/graylog/server/trusted_clients`) that will hold all client certificates you allow to connect to the beats input. This directory must be available on all Graylog server that have the input enabled. Write that path in the beats input configuration *TLS Client Auth Trusted Certs* and select **required** for the option *TLS client authentication*.

After this setting is saved only clients that provide a certificate that is trusted by the CA and is placed inside the configured directory (`/etc/graylog/server/trusted_clients`) can deliver messages to Graylog.

Beats Shipper

When using Beats configure a logstash output. The SSL configuration can be found as the second point in the description by elastic . This is:

```
output.logstash:
        hosts: ["graylog.example.org:5044"]
        ssl.certificate_authorities: ["/etc/ca.pem"]
        ssl.certificate: "/etc/client.crt"
        ssl.key: "/etc/client.key"
```

Place your previously created certificates on the server where you installed beats and adjust the configuration to your needs.

The certificate (`.crt`) file of the beats needs to be placed at the Graylog server in the configured directory for trusted clients only if you have enabled that feature at the beats input in Graylog and want client authentication.

6.3.3 Logging user activity

Graylog has been built using a client-server architecture model in which the user interface retrieves all data via a collection of REST APIs. Thus logging relevant user activity, in other words an access log, is simply a matter of enabling a built-in feature. It logs all requests to the Graylog REST API and produces an access log augmented by additional information, like the user name, the remote address, and the user agent.

Configuring the Access Log

The Access Log is configured by adding an appender and logger to the Log4j2 configuration file (`log4j2.xml`). The following example demonstrates required additions on top of the normal logging configuration:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration packages="org.graylog2.log4j" shutdownHook="disable">
    <Appenders>
        <!-- Simple appender that writes access log to specified file -->
        <File name="RestAccessLog" fileName="/var/log/graylog/server/restaccess.log"
↪append="true">
            <PatternLayout pattern="%d %-5p: %c - %m%n"/>
        </File>
    </Appenders>
    <Loggers>
        <!-- RestAccessLogFilter -->
        <Logger name="org.graylog2.rest.accesslog" level="debug" additivity="false">
                <AppenderRef ref="RestAccessLog" level="debug"/>
                <AppenderRef ref="STDOUT" level="info"/>
```

(continues on next page)

```
        </Logger>
    </Loggers>
</Configuration>
```

The resulting log entries will look similar to the following messages:

```
2016-06-08 18:21:55,651 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-]
→"GET streams" Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:46.0) Gecko/20100101␣
→Firefox/46.0 200 -1
2016-06-08 18:21:55,694 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-]
→"GET system/fields" Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:46.0) Gecko/20100101␣
→Firefox/46.0 200 -1
2016-06-08 18:21:55,698 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-]
→"GET system/fields" Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:46.0) Gecko/20100101␣
→Firefox/46.0 200 -1
2016-06-08 18:21:55,780 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-]
→"GET system/inputs" Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:46.0) Gecko/20100101␣
→Firefox/46.0 200 -1
2016-06-08 18:21:56,021 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-]
→"GET search/universal/relative?query=%2A&range=300&limit=150&sort=timestamp%3Adesc"␣
→Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:46.0) Gecko/20100101 Firefox/46.0 200 -1
```

X-Forwarded-For HTTP header support

If there is a proxy server, load balancer, or a network device which hides the client's IP address from Graylog, it can read the information from a supplied `X-Forwarded-For` HTTP request header. Most HTTP-capable devices support setting such a (semi-) standard HTTP request header.

Since overriding the client address from a externally supplied HTTP request header opens the door for spoofing, the list of trusted proxy servers which are allowed to provide the `X-Forwarded-For` HTTP request header, can be configured using the `trusted_proxies` setting in the Graylog configuration file (`graylog.conf`):

```
# Comma separated list of trusted proxies that are allowed to set the client address␣
→with X-Forwarded-For
# header. May be subnets, or hosts.
trusted_proxies = 127.0.0.1/32, 0:0:0:0:0:0:0:1/128
```

6.3.4 Using ModSecurity

ModSecurity is a popular open source web application firewall that can be used in conjuction with the Apache and Nginx web servers. When Graylog is configured behind a web server that uses ModSecurity, certain configuration changes must be made. The following examples are for version 2.x rules.

Some distributions (for example RHEL 7.x) ship with older rule sets that do not allow the MIME type `application/json` to be used in requests. This can be fixed by modifying the variable `tx.allowed_request_content_type`:

```
# Allow application/json
SecRule REQUEST_URI "@beginsWith /" \
  "id:'000001', \
  phase:1, \
  t:none, \
setvar:'tx.allowed_request_content_type=application/x-www-form-urlencoded|multipart/
→form-data|text/xml|application/xml|application/x-amf|application/json|application/
→octet-stream', \
```

```
    nolog, \
    pass"
```

Load balancers accessing `/system/lbstatus` rarely provide the ordinary HTTP headers `Host`, `Accept`, or `User-Agent`. The default rules disallow requests that are missing the mentioned headers. They should be explicitly allowed:

```
# Host header
SecRule REQUEST_URI "@beginsWith /system/lbstatus" \
  "id:'000002', \
  phase:2, \
  t:none, \
  ctl:ruleRemoveById=960008, \
  nolog, \
  pass"
# Accept header
SecRule REQUEST_URI "@beginsWith /system/lbstatus" \
  "id:'000003', \
  phase:2, \
  t:none, \
  ctl:ruleRemoveById=960015, \
  nolog, \
  pass"
# User agent header
SecRule REQUEST_URI "@beginsWith /system/lbstatus" \
  "id:'000004', \
  phase:2, \
  t:none, \
  ctl:ruleRemoveById=960009, \
  nolog, \
  Pass"
```

The HTTP verb DELETE is usually forbidden by default. It should be explicitly allowed for requests to `/api/`:

```
# Enable DELETE for /api/
SecRule REQUEST_URI "@beginsWith /api/" \
  "id:'000005', \
  phase:1, \
  t:none, \
  setvar:'tx.allowed_methods=GET HEAD POST OPTIONS DELETE', \
  nolog, \
  pass"
```

ModSecurity ships by default with strict rules against SQL injection. The query strings used in Graylog searches trigger those rules, breaking all search functionality. It should be noted that Graylog ships with no SQL based products. The offending rules can usually be safely removed, for example:

```
# Disable SQL injection rules
SecRuleRemoveById 981173
SecRuleRemoveById 960024
SecRuleRemoveById 981318
SecRuleRemoveById 981257
```

6.3.5 The URL Whitelist

There are certain components in Graylog which will perform outgoing HTTP requests. Among those, are event notifications and HTTP-based data adapters.

Allowing Graylog to interact with resources using arbitrary URLs may pose a security risk. HTTP requests are executed from Graylog servers and might therefore be able to reach more sensitive systems than an external user would have access to, including AWS EC2 metadata, which can contain keys and other secrets, Elasticsearch and others.

It is therefore advisable to restrict access by explicitly whitelisting URLs which are considered safe. HTTP requests will be validated against the Whitelist and are prohibited if there is no Whitelist entry matching the URL.

Configuring the Whitelist

The Whitelist configuration is located at `System/Configurations`. The Whitelist is enabled by default.

Disabling the Whitelist

If the security implications mentioned above are of no concern, the Whitelist can be completely disabled. When disabled, HTTP requests will not be restricted.

"Exact match" Whitelist entries

Whitelist entries of type `Exact match` contain a string which will be matched against a URL by direct comparison. If the URL is equal to this string, it is considered to be whitelisted.

"Regex" Whitelist entries

Whitelist entries of type `Regex` contain a regular expression. If a URL matches the regular expression, the URL is considered to be whitelisted. Graylog uses the Java Pattern class to evaluate regular expressions.

SENDING IN LOG DATA

A Graylog setup is pretty worthless without any data in it. This page explains the basic principles of getting your data into the system and also explains common fallacies.

7.1 What are Graylog message inputs?

Message inputs are the Graylog parts responsible for accepting log messages. They are launched from the web interface (or the REST API) in the *System -> Inputs* section and are launched and configured without the need to restart any part of the system.

7.2 Content packs

Content packs are bundles of Graylog input, extractor, stream, dashboard, and output configurations that can provide full support for a data source. Some content packs are shipped with Graylog by default and some are available from the website. Content packs that were downloaded from the Graylog Marketplace can be imported using the Graylog web interface.

You can load and even create own content packs from the *System -> Content Packs* section of your Graylog web interface.

7.3 Syslog

Graylog is able to accept and parse RFC 5424 and RFC 3164 compliant syslog messages and supports TCP transport with both the octet counting or termination character methods. UDP is also supported and the recommended way to send log messages in most architectures.

Many devices, especially routers and firewalls, do not send RFC compliant syslog messages. This might result in wrong or completely failing parsing. In that case you might have to go with a combination of *raw/plaintext* message inputs that do not attempt to do any parsing and *Extractors*.

Rule of thumb is that messages forwarded by `rsyslog` or `syslog-ng` are usually parsed flawlessly.

7.3.1 Sending syslog from Linux hosts

Sending syslog data from Linux hosts is described on the Graylog Marketplace.

7.3.2 Sending syslog from MacOS X hosts

Sending log messages from MacOS X syslog daemons is easy. Just define a `graylog-server` instance as UDP log target by adding this line in your `/etc/syslog.conf`:

```
*.* @graylog.example.org:514
```

Now restart `syslogd`:

```
$ sudo launchctl unload /System/Library/LaunchDaemons/com.apple.syslogd.plist
$ sudo launchctl load /System/Library/LaunchDaemons/com.apple.syslogd.plist
```

Important: If `syslogd` was running as another user you might end up with multiple `syslogd` instances and strange behavior of the whole system. Please check that only one `syslogd` process is running:

```
$ ps aux | grep syslog
lennart          58775   0.0  0.0  2432768    592 s004  S+    6:10PM   0:00.00 grep
→syslog
root             58759   0.0  0.0  2478772   1020  ??   Ss    6:09PM   0:00.01 /usr/
→sbin/syslogd
```

That's it! Your MacOS X syslog messages should now appear in your Graylog system.

7.4 GELF / Sending from applications

The Graylog Extended Log Format (GELF) is a log format that avoids the shortcomings of classic plain syslog and is perfect to logging from your application layer. It comes with optional compression, chunking and most importantly a clearly defined structure. There are dozens of GELF libraries for many frameworks and programming languages to get you started.

Read more about *GELF in the specification*.

7.4.1 GELF via HTTP

You can send in all GELF types via HTTP, including uncompressed GELF that is just a plain JSON string.

After launching a GELF HTTP input you can use the following endpoints to send messages:

```
http://graylog.example.org:[port]/gelf (POST)
```

Try sending an example message using curl:

```
curl -XPOST http://graylog.example.org:12202/gelf -p0 -d '{"short_message":"Hello
→there", "host":"example.org", "facility":"test", "_foo":"bar"}'
```

Both keep-alive and compression are supported via the common HTTP headers. The server will return a `202 Accepted` when the message was accepted for processing.

7.5 Using Apache Kafka as transport queue

Graylog supports Apache Kafka as a transport for various inputs such as GELF, syslog, and Raw/Plaintext inputs. The Kafka topic can be filtered by a regular expression and depending on the input, various additional settings can be configured.

Learn how to use rsyslog and Apache Kafka in the Sending syslog via Kafka into Graylog guide.

7.6 Using RabbitMQ (AMQP) as transport queue

Graylog supports AMQP as a transport for various inputs such as GELF, syslog, and Raw/Plaintext inputs. It can connect to any AMQP broker supporting AMQP 0-9-1 such as RabbitMQ.

Learn how to use rsyslog and RabbitMQ in the Sending syslog via AMQP into Graylog guide.

7.7 Microsoft Windows

Sending syslog data from Windows is described on the Graylog Marketplace.

7.8 Ruby on Rails

This is easy: You just need to combine a few components.

7.8.1 Log all requests and logger calls into Graylog

The recommended way to send structured information (i.e. HTTP return code, action, controller, . . . in additional fields) about every request and explicit `Rails.logger` calls is easily accomplished using the GELF gem and lograge. Lograge builds one combined log entry for every request (instead of several lines like the standard Rails logger) and has a Graylog output since version 0.2.0.

Start by adding Lograge and the GELF gem to your Gemfile:

```
gem "gelf"
gem "lograge"
```

Now configure both in your Rails application. Usually `config/environments/production.rb` is a good place for that:

```
config.lograge.enabled = true
config.lograge.formatter = Lograge::Formatters::Graylog2.new
config.logger = GELF::Logger.new("graylog.example.org", 12201, "WAN", { :host =>
↪"hostname-of-this-app", :facility => "heroku" })
```

This configuration will also send all explicit `Rails.logger` calls (e.g. `Rails.logger.error "Something went wrong"`) to Graylog.

7.8.2 Log only explicit logger calls into Graylog

If you don't want to log information about every request, but only explicit `Rails.logger` calls, it is enough to only configure the Rails logger.

Add the GELF gem to your Gemfile:

```
gem "gelf"
```

. . . and configure it in your Rails application. Usually `config/environments/production.rb` is a good place for that:

```
config.logger = GELF::Logger.new("graylog.example.org", 12201, "WAN", { :host =>
→"hostname-of-this-app", :facility => "heroku" })
```

7.8.3 Heroku

You need to apply a workaround if you want custom logging on Heroku. The reason for this is that Heroku injects an own logger (`rails_log_stdout`), that overwrites your custom one. The workaround is to add a file that makes Heroku think that the logger is already in your application:

```
$ touch vendor/plugins/rails_log_stdout/heroku_fix
```

7.9 Raw/Plaintext inputs

The built-in *raw/plaintext* inputs allow you to parse any text that you can send via TCP or UDP. No parsing is applied at all by default until you build your own parser using custom *Extractors*. This is a good way to support any text-based logging format.

You can also write *Plugins* if you need extreme flexibility.

7.10 JSON path from HTTP API input

The JSON path from HTTP API input is reading any JSON response of a REST resource and stores a field value of it as a Graylog message.

7.10.1 Example

Let's try to read the download count of a release package stored on GitHub for analysis in Graylog. The call looks like this:

```
$ curl -XGET https://api.github.com/repos/YourAccount/YourRepo/releases/assets/12345
{
  "url": "https://api.github.com/repos/YourAccount/YourRepo/releases/assets/12345",
  "id": 12345,
  "name": "somerelease.tgz",
  "label": "somerelease.tgz",
  "content_type": "application/octet-stream",
  "state": "uploaded",
  "size": 38179285,
```

(continues on next page)

```
  "download_count": 9937,
  "created_at": "2013-09-30T20:05:01Z",
  "updated_at": "2013-09-30T20:05:46Z"
}
```

The attribute we want to extract is `download_count` so we set the JSON path to `$.download_count`.

This will result in a message in Graylog looking like this:

You can use Graylog to analyze your download counts now.

7.10.2 JSONPath

JSONPath can do much more than just selecting a simple known field value. You can for example do this to select the first `download_count` from a list of releases where the field `state` has the value `uploaded`:

```
$.releases[?(@.state == 'uploaded')][0].download_count
```

. . . or only the first download count at all:

```
$.releases[0].download_count
```

You can learn more about JSONPath here.

7.11 Reading from files

Log files come in a lot of different flavors and formats, much more than any single program could handle.

To support this use case, we provide the *Sidecar* which acts as a supervisor process for other programs, such as nxlog and Filebeats, which have specifically been built to collect log messages from local files and ship them to remote systems like Graylog.

Of course you can still use any program supporting the GELF or syslog protocol (among others) to send your logs to Graylog.

7.12 Input Throttling

Throttling allows certain Graylog Inputs to slow their message intake rates (by temporarily pausing intake processing) if contention occurs in the Graylog Journal.

7.12.1 Graylog Inputs that support throttling

- AWS Flow Logs
- AWS Logs
- CEF AMQP Input
- CEF Kafka Input
- GELF AMQP
- GELF Kafka
- JSON path from HTTP API
- Raw/Plaintext AMQP
- Raw/Plaintext Kafka
- Syslog AMQP
- Syslog Kafka

7.12.2 Enabling throttling

To enable throttling for one of these inputs, edit it in *System > Inputs* and check the *Allow throttling this input* checkbox.

7.12.3 Throttling criteria

When enabled, the following criteria will be used to determine if throttling will occur:

1. If there are zero uncommitted entries in the Graylog Journal, throttling will not occur. No further checks will be performed.
2. Throttling will occur if the Journal has more than 100k uncommitted entries.
3. Throttling will occur if the Journal is growing in size rapidly (approximately 20k entries per second or greater).
4. Throttling will occur if the process ring buffer is full.
5. Nothing is currently being written to the Journal, throttling will not occur. No further checks will be performed.
6. Throttling will occur if the Journal is more than 90% full.
7. Throttling will occur if the Journal write rate is more than twice as high as the read rate.

GRAYLOG SIDECAR

Note: Graylog 3.0 comes with a new Sidecar implementation. We still support the old **Collector Sidecars**, which can be found in the `System / Collectors (legacy)` menu entry. In case you need to configure legacy **Collector Sidecar** please refer to the Graylog Collector Sidecar documentation. We encourage users to migrate to the new **Sidecar**, which is covered by this document.

Graylog Sidecar is a lightweight configuration management system for different log collectors, also called *Backends*. The Graylog node(s) act as a centralized hub containing the configurations of log collectors. On supported message-producing devices/hosts, Sidecar can run as a service (Windows host) or daemon (Linux host).
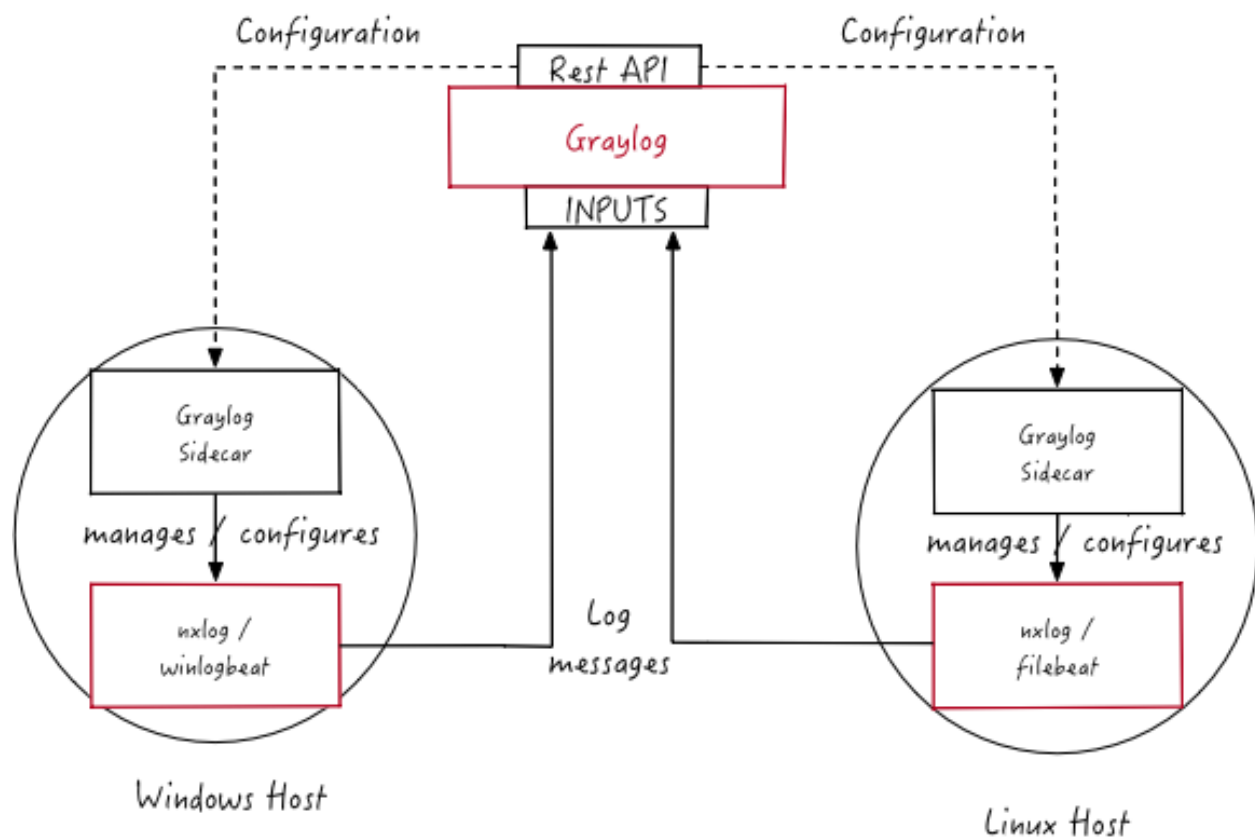
The log collector configurations are centrally managed through the Graylog web interface. Periodically, the Sidecar daemon will fetch all relevant configurations for the target, using the *REST API*. On its first run, or when a configuration

change has been detected, Sidecar will *generate* (render) relevant backend configuration files. Then it will start, or restart, those reconfigured log collectors.

8.1 Installation

Currently we provide pre-compiled packages on the Github releases page of the project. Once the Sidecar project is settled and matured we will add the packages to the DEB and YUM online repositories. To get the Sidecar working Download a package and install it on the target system.

Please follow the version matrix to pick the right package:

Sidecar version	Graylog server version
1.0.x	3.0.x
0.1.x	2.2.x,2.3.x,2.4.x,2.5.x,3.0.x
0.0.9	2.1.x

All following commands should be executed on the **remote machine** where you want to collect log data from.

8.1.1 Install the Sidecar

Ubuntu

```
$ sudo dpkg -i graylog-sidecar_1.0.0-1_amd64.deb
```

Edit the configuration (see *Configuration*) and activate the Sidecar as a system service:

```
$ vi /etc/graylog/sidecar/sidecar.yml

$ sudo graylog-sidecar -service install

[Ubuntu 14.04 with Upstart]
$ sudo start graylog-sidecar

[Ubuntu 16.04 and later with Systemd]
$ sudo systemctl start graylog-sidecar
```

CentOS

Install the RPM package on RedHat based systems

```
$ sudo rpm -i graylog-sidecar-1.0.0-1.x86_64.rpm
```

Edit the configuration (see *Configuration*) and activate the Sidecar as a system service:

```
$ vi /etc/graylog/sidecar/sidecar.yml

$ sudo graylog-sidecar -service install
$ sudo systemctl start graylog-sidecar
```

Windows

Use the Windows installer, it can be run interactively:

```
$ graylog_sidecar_installer_1.0.0-1.exe
```

Or in silent mode with:

```
$ graylog_sidecar_installer_1.0.0-1.exe /S -SERVERURL=http://10.0.2.2:9000/api -
→APITOKEN=yourapitoken
```

Optionally edit the configuration (see *Configuration*) and register the system service:

```
notepad.exe C:\Program Files\Graylog\sidecar\sidecar.yml

& "C:\Program Files\graylog\sidecar\graylog-sidecar.exe" -service install
& "C:\Program Files\graylog\sidecar\graylog-sidecar.exe" -service start
```

8.1.2 Install collectors

Next up, you can decide which collectors you want to use with your Sidecar and install them as well. We only cover the installation of the most common ones here, but you are free to use other collectors as well. Graylog contains default collector configurations for Filebeat, Winlogbeat and NXLog. But since you're able to define your own collector backends, there is nothing stopping you from running e.g. sysmon, auditd or packetbeat.

Beats on Linux

Install Filebeat or another Beats package by following the instructions on the official Filebeat download page.

Beats on Windows

The Windows Sidecar package already includes Filebeat and Winlogbeat. For other Beats packages follow the instructions on the official Beats download page.

NXLog on Ubuntu

Install the NXLog package from the official NXLog download page. Because the Sidecar takes control of stopping and starting NXlog it is necessary to stop all running instances of NXlog and unconfigure the default system service:

```
$ sudo /etc/init.d/nxlog stop
$ sudo update-rc.d -f nxlog remove
$ sudo gpasswd -a nxlog adm
$ sudo chown -R nxlog.nxlog /var/spool/nxlog
```

NXLog on CentOS

The same on a RedHat based system:

```
$ sudo service nxlog stop
$ sudo chkconfig --del nxlog
$ sudo gpasswd -a nxlog root
$ sudo chown -R nxlog.nxlog /var/spool/nxlog
```

NXlog on Windows

Install the NXLog package from the official download page and deactivate the system service. We just need the binaries installed on the system:

```
$ C:\Program Files (x86)\nxlog\nxlog -u
```

8.2 Sidecar Configuration

On the command line you can provide a path to the configuration file with the `-c` switch. The default configuration path on Linux systems is `/etc/graylog/sidecar/sidecar.yml` and `C:\Program Files\Graylog\sidecar\sidecar.yml` on Windows.

Most configuration parameters come with built-in defaults. The only parameters that need adjustment are `server_url` and `server_api_token`. You can get your API token by following the link on the *Sidecars Overview* page.

8.2.1 sidecar.yml Reference

Parameter	Description
server_url	URL to the Graylog API, e.g. `http://192.168.1.1:9000/api/`
server_api_token	The API token to use to authenticate against the Graylog server API. e.g `1jq26cssvc6rj4qac4bt9oeeh0p4vt5u5kal9jocl1g9mdi4og3n` The token is mandatory and needs to be configured.
node_id	The node ID of the sidecar. This can be a path to a file or an ID string. Example file path: `file:/etc/graylog/sidecar/node-id` Example ID string: `6033137e-d56b-47fc-9762-cd699c11a5a9` ATTENTION: Every sidecar instance needs a unique ID! Default: `file:/etc/graylog/sidecar/node-id`
node_name	Name of the Sidecar instance, will also show up in the web interface. The hostname will be used if not set.
update_interval	The interval in seconds the sidecar will fetch new configurations from the Graylog server Default: `10`
tls_skip_verify	This configures if the sidecar should skip the verification of TLS connections. Default: `false`
send_status	This controls the transmission of detailed sidecar information like collector status, metrics and log file lists. It can be disabled to reduce load on the Graylog server if needed. Default: `true`
list_log_files	Send a directory listing to Graylog and display it on the host status page, e.g. `/var/log`. This can also be a list of directories. Default: `[]`
cache_path	The directory where the sidecar stores internal data. Default: `/var/cache/graylog-sidecar`
collector_configuration_directory	The directory where the sidecar generates configurations for collectors. Default: `/var/lib/graylog-sidecar/generated`
log_path	The directory where the sidecar stores its logs. Default: `/var/log/graylog-sidecar`
log_rotate_max_file_size	The maximum size of the log file before it gets rotated. Default: `10MiB`
log_rotate_keep_files	The maximum number of old log files to retain.
collector_binaries_whitelist	A list of binaries which are allowed to be executed by the Sidecar. An empty list disables the white list feature. Default: `/usr/bin/filebeat, /usr/bin/packetbeat, /usr/bin/metricbeat, /usr/bin/heartbeat, /usr/bin/auditbeat, /usr/bin/journalbeat, /usr/share/filebeat/bin/filebeat, /usr/share/packetbeat/bin/packetbeat, /usr/share/metricbeat/bin/metricbeat, /usr/share/heartbeat/bin/heartbeat, /usr/share/auditbeat/bin/auditbeat, /usr/share/journalbeat/bin/journalbeat, /usr/bin/nxlog, /opt/nxlog/bin/nxlog`

8.2.2 First start

Once you installed the Sidecar package and started the service for the first time, you can verify that it shows up in the *Sidecars Overview* page. A new sidecar instance will not have any configurations assigned yet. Take the *Step-by-step guide* to create your first configuration.

8.2.3 Mode of Operation

When the Sidecar is assigned a configuration via the Graylog web interface, it will write a configuration file into the `collector_configuration_directory` directory for each collector backend. E.g. if you assigned a Filebeat collector you will find a `filebeat.yml` file in that directory. All changes have to be made in the Graylog web interface. Every time the Sidecar detects an update to its configuration it will rewrite the corresponding collector configuration file. Manually editing these files is not recommended.

Every time a collector configuration file is changed the collector process is restarted. The Sidecar takes care of the collector processes and reports the status back to the web interface

8.2.4 Sidecar Status

Each Sidecar instance is able to send status information back to Graylog. By enabling the option `send_status` metrics like load or the IP address of the host Sidecar is running on are sent. Also metrics that are relevant for a stable operation e.g. disk volumes over 75% utilization are included. Additionally with the `list_log_files` option a directory listing is displayed in the Graylog web interface. In that way an administrator can see which files are available for collecting. The list is periodically updated and files with write access are highlighted for easy identification. After enabling `send_status` or `send_status` + `list_log_files` go to the collector overview and click on one of them, a status page with the configured information will be displayed.

8.3 Step-by-step guide

We have prepared an example on how to configure the Sidecar using the Graylog web interface. The assumption is that we want to collect Apache logfiles and ship them with a Filebeat collector to a Beats input that is listening on Port 5044 on your Graylog Server.

- The first step is to create a Beats input where collectors can send data to. Click on `System / Inputs` and start a global Beats input on the listening address 0.0.0.0 and port 5044.

- Navigate to the Sidecars overview. In your Graylog web interface click on `System / Sidecars`.

- Navigate to the Sidecar `Configuration` page.

- Next we create a new configuration: We give the configuration a name and select `filebeat on Linux` as collector. (This collector definition is shipped with Graylog, and comes with a default configuration template). Most of the configuration defaults should work for you. However you need to change the `hosts:` setting and point it to your Beats input. You also might want to change the `paths:` to the location of your Apache logs. When done click `Create` to save your configuration.

- Next we need to assign our newly created configuration (and therefore the Filebeat collector) to our sidecar. Go to the `Collector Administration` page.

- You will see a list of sidecars and underneath them a list of collectors that could be assigned to them. Please note that collectors are assigned to sidecars by means of applying a collector configuration to the sidecar. Therefore, we first select the `filebeat` collector and then click on the `Configure` menu, where we can select the

`filebeat-conf` configuration we created earlier.
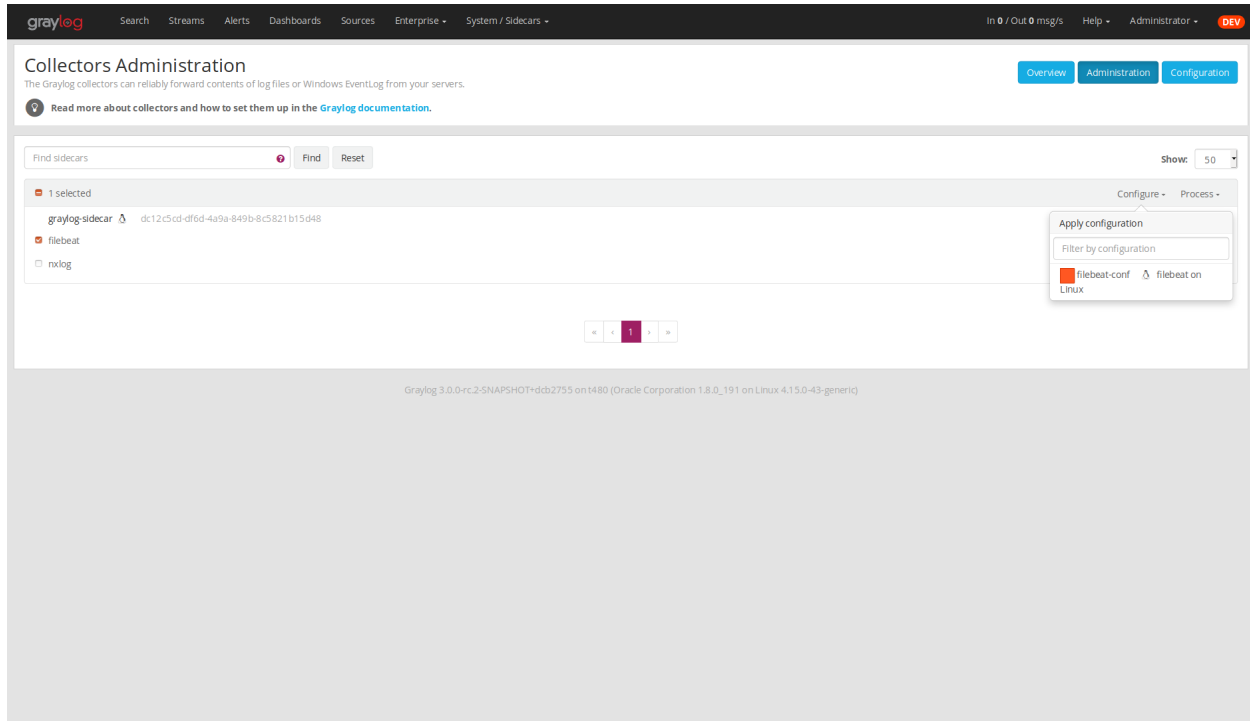
- Confirming the assignment, will directly push this configuration to your sidecar which will go and start the Filebeat collector with this configuration.
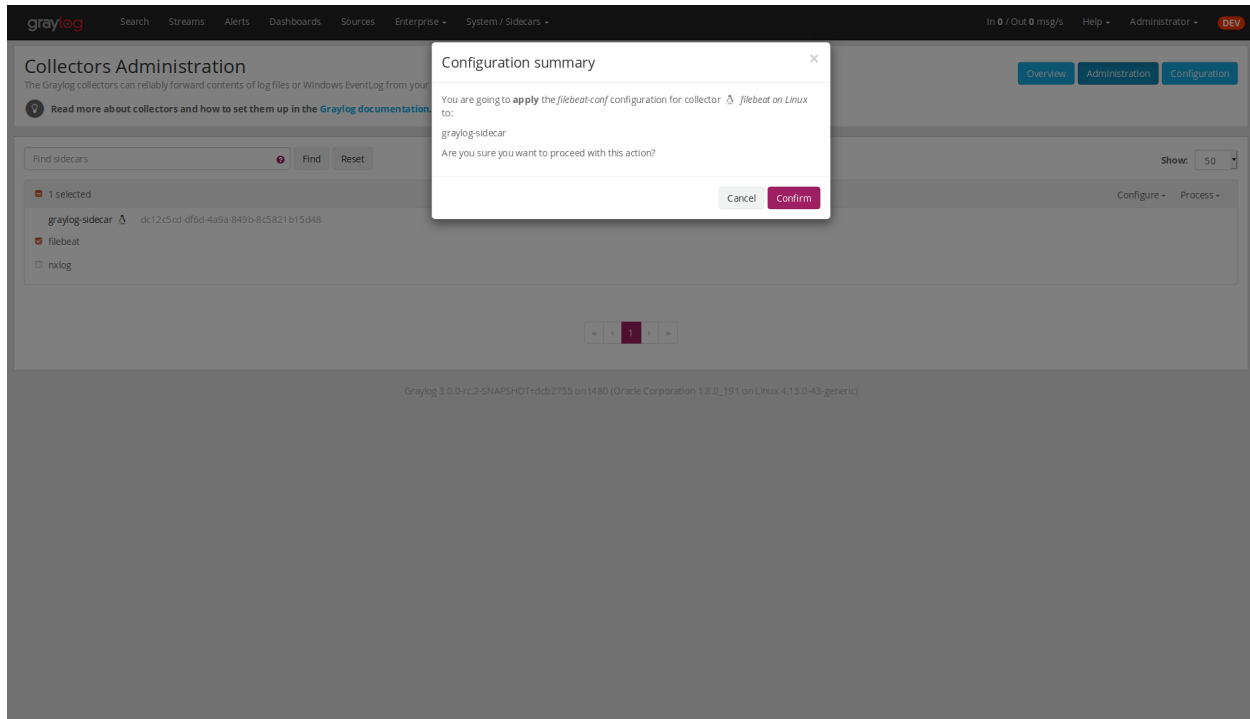
- If everything went fine, you should see the status `running` on the administration page.

- Congratulations your collector setup is working now! You can go back to the Sidecars overview and click on the `Show messages` button to search for logs that have been collected via your sidecar.

8.4 Creating a new Log Collector

Graylog comes with a few predefined log collectors which can be easily extended and changed to your needs. Let's assume you want your sidecar to run *rsyslogd(8)* for you.

- Navigate to the Sidecars overview. In your Graylog web interface click on `System / Sidecars`.

- Navigate to the Sidecar `Configuration` page.

- After we click on `Create Log Collector`, we are presented with the following page, where we have to fill out some fields for our new collector. We give the collector a unique name and select `Linux` and `Foreground Execution`. Given that you installed rsyslogd(8) under `/usr/sbin/rsyslogd` we configure the executable path accordingly. If you are using `Foreground Execution` make sure that the collector you are running does not daemonize itself. Otherwise the sidecar has no way of controlling the collector once it has forked off into the background. For rsyslogd we therefore provide `-n` as *Execute Parameter*. If your collector

supports configuration validation, it is advised to use it. This acts as a pre-check, so that sidecar won't restart a collector with a broken configuration. For rsyslogd the option to do a configuration check is `-N 1`. Optionally you can provide a *Default Template* which will be proposed once you create a configuration for this collector.

- Next up you can use your newly created collector by creating a configuration for it and assign it to a Sidecar. Please follow the *Step-by-step guide* accordingly.

- **Note**: Your Sidecar might refuse to start your collector, because it needs to be added to the `collector_binaries_whitelist` first. Please edit your *Configuration* and restart your Sidecar.

8.5 Using Configuration Variables

Configuration variables can contain arbitrary strings like the IP address of your Graylog server or the port of an input. The variables can then be used in multiple collector configurations, which avoids duplication and simplifies management.

To create a configuration variable go any `Collector Configuration` page:

On the right you'll find a box `Collector Configuration Reference` which contains *Runtime Variables* and *Variables*. Click on `Variables` and then `Create Variable` to receive the following modal:

In this example we replace the hard coded IP and Port from our Beats input with a new variable named `${user.BeatsInput}`:

We can now use this variable in all our configurations. If we ever need to change the IP/port of our input, we just change the variable.

8.5.1 Runtime Variables

Runtime variables contain runtime informations from each Sidecar that is requesting this configuration. An important example is the `${sidecar.nodeId}` variable. The collector configuration should contain an instruction to fill that variable in an extra field *gl2_source_collector*. This allows Graylog to relate messages to the Sidecar that produced them. (This is what makes the `Show messages` button on the Sidecars overview page work)

8.6 Secure Sidecar Communication

The Communication between Sidecar and Graylog will be secured if your API *uses SSL*.

To secure the communication between the Collector and Graylog you just need to mark `Enable TLS` in your Beats Input. Without giving additional Information, Graylog will now create a self-signed certificate for this Input. Now in the Sidecar Beats Output Configuration you just mark `Enable TLS Support` and `Insecure TLS connection`. After this is saved, the communication between Beats and Graylog will use TLS.

8.6.1 Certificate based client authentication

If you want Graylog to only accept data from authenticated Collectors please follow the steps at *Secured Graylog and Beats input*

8.7 Run Sidecar as non-root user

The default is that the Sidecar is started with the root user to allow access to all log files. But this is not mandatory. If you like to start it with a daemon user, proceed like the following:

- Create a daemon user e.g. `sidecar`

The Sidecar itself is accessing the following files and directories:

- `sidecar.yml` - /etc/graylog/sidecar/sidecar.yml

- `collector_configuration_directory` - /var/lib/graylog-sidecar/generated/

- `node_id` - /etc/graylog/sidecar/node-id

- `cache_path` - /var/cache/graylog-sidecar/

- `log_path` - /var/log/graylog-sidecar/

So to make these directories readable for the `sidecar` user, use:

- `chown -R sidecar /etc/graylog/sidecar`

- `chown -R sidecar /var/cache/graylog-sidecar`

- `chown -R sidecar /var/lib/graylog-sidecar`

- `chown -R sidecar /var/log/graylog-sidecar`

You can change all paths to different places in the file system. If you prefer to store all Sidecar data in the home directory of the `sidecar` user, just change the paths accordingly.

Now `systemd` needs to know that the Sidecar should be started with a non-root user. Open `/etc/systemd/system/collector-sidecar.service` with an editor and navigate to the `[Service]` section, add:

```
User=sidecar
Group=sidecar
```

To make use of these settings reload systemd:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart graylog-sidecar
```

Check the log files in `/var/log/graylog-sidecar` for any errors. Understand that not only the Sidecar but also all backends, like `filebeat`, will be started as `sidecar` user after these changes. So all log files that the backend should observe also need to be readable by the `sidecar` user. Depending on the Linux distribution there is usually an administrator group which has access to most log files. By adding the `sidecar` user to that group you can grant access fairly easy. For example on Debian/Ubuntu systems this group is called `adm` (see System Groups in Debian Wiki or Security/Privileges - Monitor system logs in Ubuntu wiki).

8.8 Upgrading from the Collector Sidecar

This guide describes how you can perform an upgrade from the deprecated **Collector Sidecars** (0.1.x) to the new **Sidecars** (1.x).

One major difference between the old and the new Sidecars, is that we replaced the UI based collector configuration approach with one where you can manage the plain text configuration of the collectors directly. This might seem like an inconvenience at first, but gives you the flexibility to configure any collector backend you want.

Additionally, the new Sidecars don't assign configurations based on tags anymore. Instead you have to assign configurations explicitly (see *Step-by-Step guide*).

8.8.1 1. Install New Sidecar

The new Sidecar has different paths and executable names, so it can coexist with the old one. Install the new Sidecar by following the *Installation instructions* and have your Sidecar running as described in *First Start*.

Note: In case you were using filebeat on Linux, please make sure to also install the official collector package, since the filebeat binary is not part of the Sidecar package anymore.

8.8.2 2. Migrate configuration

Next, we need to migrate the configuration that was previously rendered on each host by the **Collector Sidecar**, to a new **Collector Configuration**.

We recommend to use the *Sidecar Configuration Migrator*. However, retrieving the old configuration can also be done manually by fetching it from your host at the `/etc/graylog/collector-sidecar/generated/` directory.

8.8.3 3. Adopt configuration to Graylog 3.0

There are a few things that might need attention after an upgrade:

- Use *Runtime variables* for static fields

  The imported configuration contains instructions that add static fields which allows Graylog to relate messages to a Sidecar. You should replace the hardcoded values of `gl2_source_collector` and `collector_node_id` with runtime variables.

  In case of a Beats collector this would be:

  ```
  fields.gl2_source_collector: ${sidecar.nodeId}
  fields.collector_node_id: ${sidecar.nodeName}
  ```

- Migrate to the new Beats input

  Graylog 3.0 comes with a new Beats input. The former one was renamed to `Beats (deprecated)`. The new input handles fields a little different. Therefore you should define `fields_under_root:  true` for the new input to get the Graylog fields work.

8.8.4 4. Switch over to the new Sidecar

Once you're done creating a new configuration, you can assign it to your Sidecar (see *Step-by-Step guide*). If everything works as expected, make sure to uninstall the old **Collector Sidecar** to avoid collecting your logs twice.

8.8.5 Sidecar Configuration Migrator

The task of the Sidecar configuration migrator is to extract the configuration from existing **Collector Sidecars** and convert it into new **Sidecar** configurations.

This feature needs a **Collector Sidecar** with version 0.1.8 or greater. Please upgrade the instance you want to import configurations from, if necessary.

- Navigate to the Collectors (legacy) overview. In your Graylog web interface click on `System / Collectors (legacy)`.

- Click on the name of the Collector you want to import configurations from

- Click the `Import Configuration` button on a backend to import a configuration. If the import was successful, follow the link to create a new Sidecar configuration:
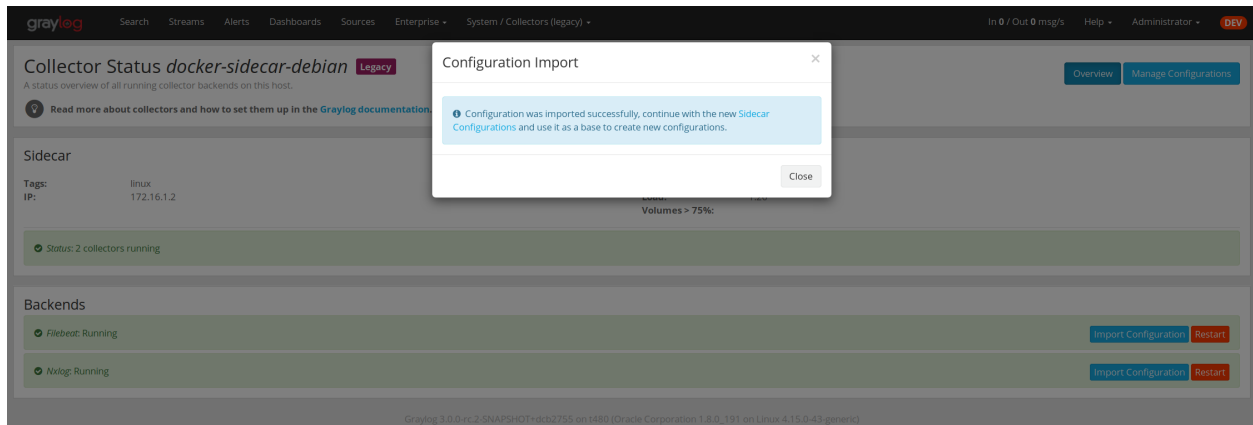
- After clicking on `Create Configuration` use the `Migrate` button underneath the configuration editor:
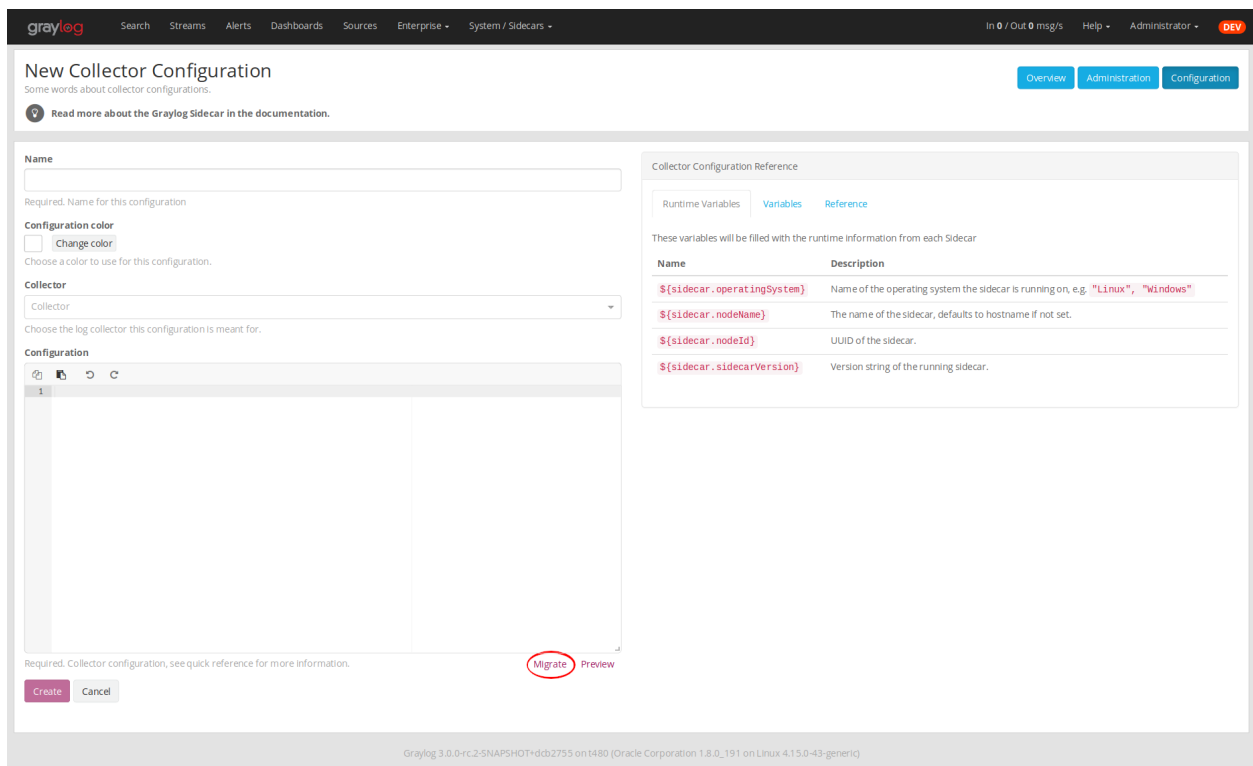
- A window opens up and lets you pick already imported configurations. Clicking `Apply` will paste the configuration into the editor. Afterwards you can edit and save the configuration as usual.
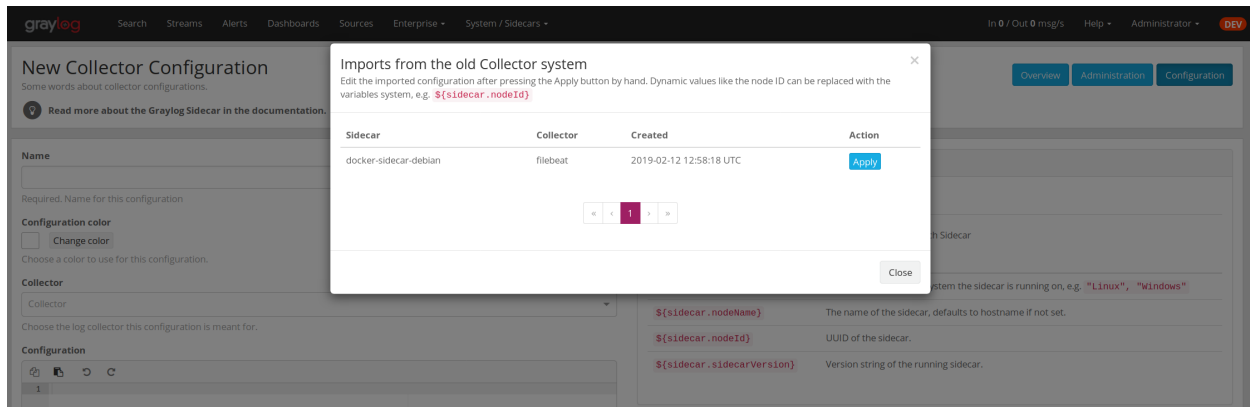
8.9 Sidecar Glossary

To understand the different parts of the Graylog Sidecar they are explained in the following section.

8.9.1 Configuration

A configuration is the representation of a log collector configuration file in the Graylog web interface. A configuration can be assigned to Sidecars, which also assigns the corresponding collector. You can have multiple configurations for a single log collector. However, you can not assign the same collector twice to a Sidecar.

8.9.2 Inputs

Inputs are the way how collectors ingest data. An input can be a log file that the collector should continuously read or a connection to the Windows event system that emits log events. An input is connected to an output, otherwise there would be no way of sending the data to the next hop. So first create an output and then associate one or many inputs with it.

8.10 Debug

The Sidecar is writing log files to the directory configured in `log_path`. One file for each backend, there you can check for general issues like file permissions or log transmission problems. The Sidecar itself is writing to `sidecar.log`. Problems like failed connection to the Graylog API can be found there.

You can also start the Sidecar in foreground and monitor the output of the process:

```
$ graylog-sidecar -debug
```

8.11 Uninstall

On Linux just uninstall the package, to perform an uninstall on Windows run:

```
& "C:\Program Files\Graylog\graylog-sidecar.exe" -service stop
& "C:\Program Files\Graylog\graylog-sidecar.exe" -service uninstall
```

8.12 Known Problems

Currently we know of two problems with NXLog:

- Since version 2.9.17 timestamps are transmitted without millisecond precision
- On Windows machines NXlog is not able to store its collector state so features like file tailing don't work correctly in combination with Sidecar. Use Sidecar version 0.1.0-alpha.1 or newer.

Known issue if you use a loadbalancer or firewall in front of Graylog's API:

- The Sidecar is using a persistent connection for API requests. Therefore it logs `408 Request Time-out` if the loadbalancer session or http timeout is lower than the configured `update_interval`.

SEARCHING

9.1 Search query language

9.1.1 Syntax

The search syntax is very close to the Lucene syntax. By default all message fields are included in the search if you don't specify a message field to search in.

Messages that include the term *ssh*:

```
ssh
```

Messages that include the term *ssh* or *login*:

```
ssh login
```

Messages that include the exact phrase *ssh login*:

```
"ssh login"
```

Messages where the field *type* includes *ssh*:

```
type:ssh
```

Messages where the field *type* includes *ssh* or *login*:

```
type:(ssh OR login)
```

Note: Elasticsearch 2.x and 5.x split queries on whitespace, so the query `type:(ssh login)` was equivalent to `type:(ssh OR login)`. This is no longer the case in Elasticsearch 6.0 and you must now include an `OR` operator between each term.

Messages where the field *type* includes the exact phrase *ssh login*:

```
type:"ssh login"
```

Messages that have the field *type*:

```
_exists_:type
```

Messages that do not have the field *type*:

```
NOT _exists_:type
```

Note: Elasticsearch 2.x allows to use `_missing_:type` instead of `NOT _exists_:type`. This query syntax has been removed in Elasticsearch 5.0.

Messages that match regular expression `ethernet[0-9]+`:

```
/ethernet[0-9]+/
```

Note: Please refer to the Elasticsearch documentation about the Regular expression syntax for details about the supported regular expression dialect.

By default all terms or phrases are OR connected so all messages that have at least one hit are returned. You can use **Boolean operators and groups** for control over this:

```
"ssh login" AND source:example.org
("ssh login" AND (source:example.org OR source:another.example.org)) OR _exists_
↪:always_find_me
```

You can also use the NOT operator:

```
"ssh login" AND NOT source:example.org
NOT example.org
```

Note that AND, OR, and NOT are case sensitive and must be typed in all upper-case.

Wildcards: Use *?* to replace a single character or *** to replace zero or more characters:

```
source:*.org
source:exam?le.org
source:exam?le.*
```

Note that leading wildcards are disabled to avoid excessive memory consumption! You can enable them in your Graylog configuration file:

```
allow_leading_wildcard_searches = true
```

Also note that `message`, `full_message`, and `source` are the only fields that are being analyzed by default. While wildcard searches (using `*` and `?`) work on all indexed fields, analyzed fields will behave a little bit different. See wildcard and regexp queries for details.

Fuzziness: You can search for similar terms:

```
ssh logni~
source:exmaple.org~
```

This example is using the Damerau–Levenshtein distance with a default distance of *2* and will match "ssh login" and "example.org" (intentionally misspelled in the query).

You can change the distance like this:

```
source:exmaple.org~1
```

You can also use the fuzzyness operator to do a **proximity** search where the terms in a phrase can have different/fuzzy distances from each other and don't have to be in the defined order:

```
"foo bar"~5
```

Numeric fields support **range queries**. Ranges in square brackets are inclusive, curly brackets are exclusive and can even be combined:

```
http_response_code:[500 TO 504]
http_response_code:{400 TO 404}
bytes:{0 TO 64]
http_response_code:[0 TO 64}
```

You can also do searches with one side unbounded:

```
http_response_code:>400
http_response_code:<400
http_response_code:>=400
http_response_code:<=400
```

It is also possible to combine unbounded range operators:

```
http_response_code:(>=400 AND <500)
```

It is possible make a **range query** on the date field. It is important that the selected period of time at the timepicker fits the range you want to search in. If you search in the last 5 minutes, but the searched time is a week in the past the query will not return anything. The dates needs to be UTC and the format needs to be like Graylog displays them.:

```
timestamp:["2019-07-23 09:53:08.175" TO "2019-07-23 09:53:08.575"]
```

9.1.2 Escaping

The following characters must be escaped with a backslash:

```
& | : \ / + - ! ( ) { } [ ] ^ " ~ * ?
```

Example:

```
resource:\/posts\/45326
```

9.2 Time frame selector

The time frame selector defines in what time range to search in. It offers three different ways of selecting a time range and is vital for search speed: If you know you are only interested in messages of the last hour, only search in that time frame. This will make Graylog search in *relevant indices* only and greatly reduce system load and required resources.
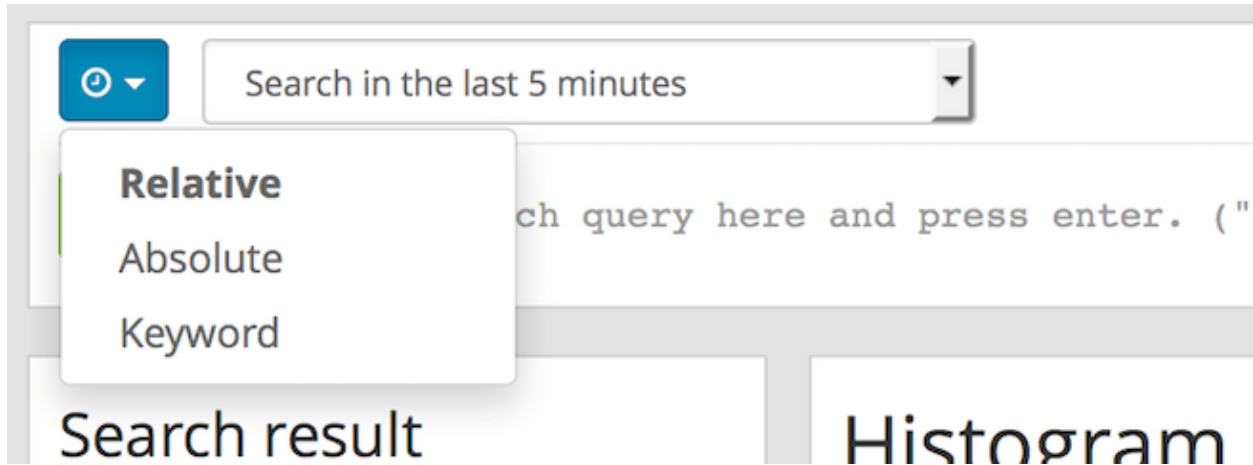
9.2.1 Relative time frame selector

The relative time frame selector lets you look for messages from the selected option to the time you hit the search button. The selector offers a wide set of relative time frames that fit most of your search needs.

9.2.2 Absolute time frame selector

When you know exactly the boundaries of your search, you want to use the absolute time frame selector. Simply introduce the dates and times for the search manually or click in the input field to open up a calendar where you can choose the day with your mouse.

9.2.3 Keyword time frame selector

Graylog offers a keyword time frame selector that allows you to specify the time frame for the search in natural language like *last hour* or *last 90 days*. The web interface shows a preview of the two actual timestamps that will be used for the search.
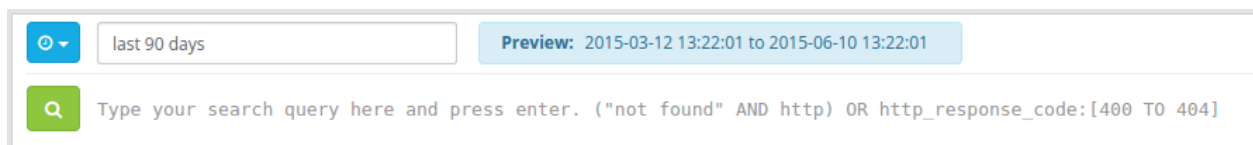
Here are a few examples for possible values.

- "last month" searches between one month ago and now

- "4 hours ago" searches between four hours ago and now

- "1st of april to 2 days ago" searches between 1st of April and 2 days ago

- "yesterday midnight +0200 to today midnight +0200" searches between yesterday midnight and today midnight in timezone +0200 - will be 22:00 in UTC

The time frame is parsed using the natty natural language parser. Please consult its documentation for details.

9.3 Saved searches

Sometimes you may want to search a specific search configuration to be used later. Graylog provides a saved search functionality to accomplish exactly that.

Once you submitted your search, selected the fields you want to show from the search sidebar, and chosen a resolution for the histogram, click on the *Save search criteria* button on the sidebar.
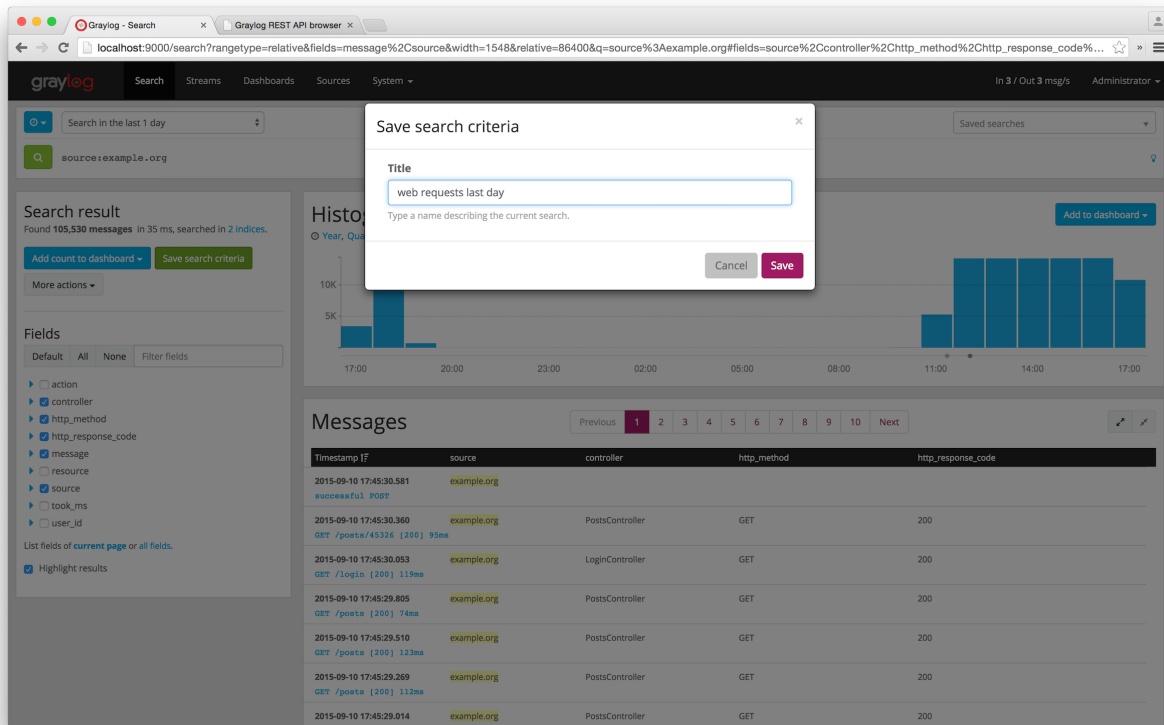
Give a name to the current search and click on save. When you want to use the saved search later on, you only need to select it from the saved search selector.
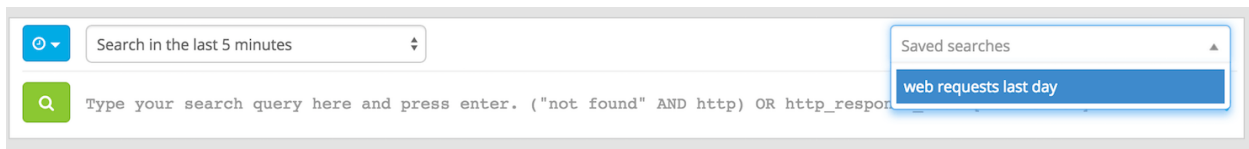
Of course, you can always update the selected fields or name of your saved search. To do so, select the saved search from the saved search selector, update the field selection or histogram resolution, and click on *Saved search -> Update search criteria*. It is also possible to delete the saved search by selecting *Saved search -> Delete saved search*.
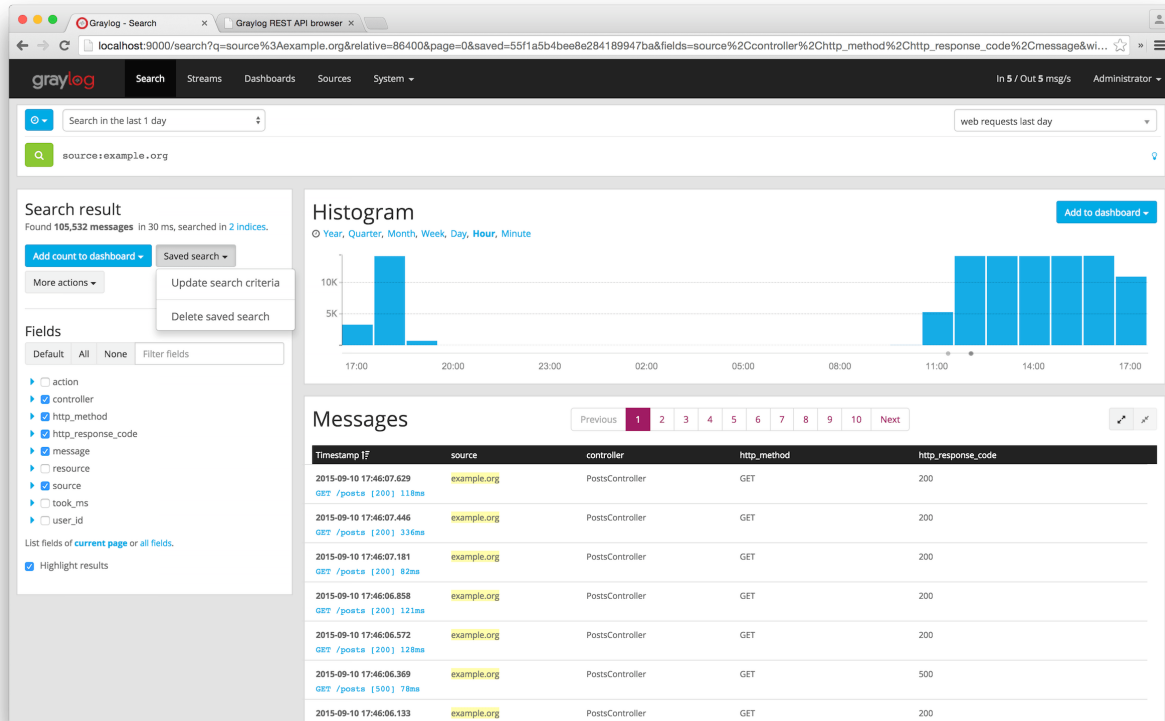
9.4 Histogram

The search page includes a search result histogram, where you can view in a concise way the number of messages received grouped by a certain time period that Graylog will adjust for you.

The histogram also allows you to further narrow down the cause for an issue:

- Delimit the search time range by brushing over the histogram. Just click and drag with your mouse over the chart to select the time range you want to use, and click on the search button to perform that search

- See the time where alerts are triggered in the graph annotations. If you are searching in a stream, you will only see alerts related to that stream
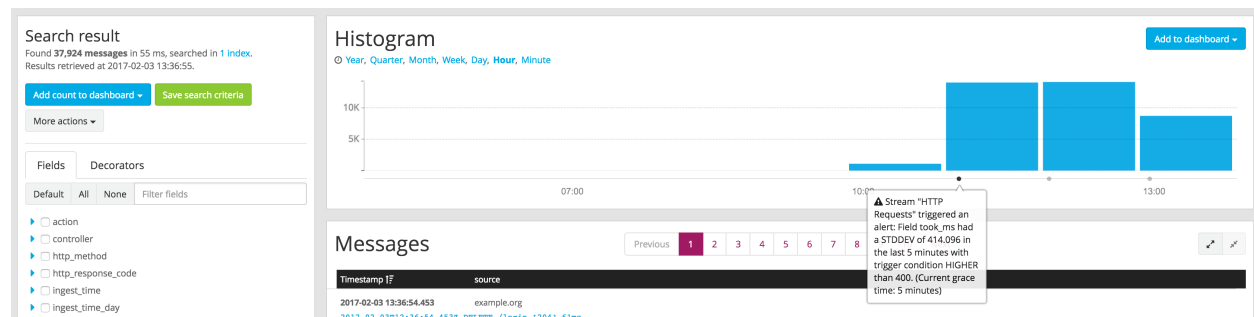
9.5 Analysis

Graylog provides several tools to analyze your search results. It is possible to save these analysis into dashboards, so you can check them over time in a more convenient way. To analyze a field from your search results, expand the field in the search sidebar and click on the button of the analysis you want to perform.
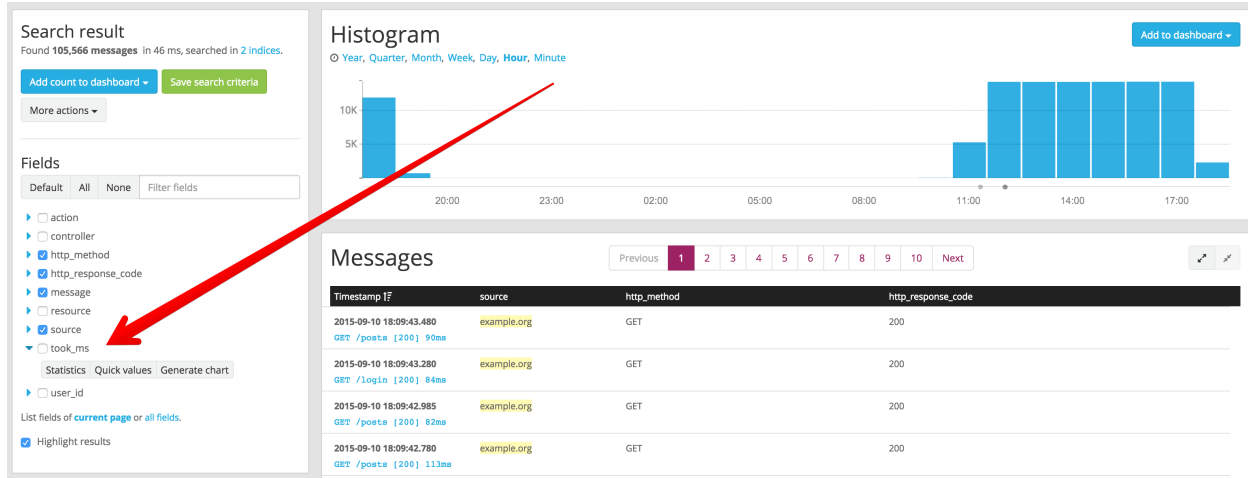
9.5.1 Field statistics

Compute different statistics on your fields, to help you better summarize and understand the data in them.

The statistical information consist of: total, mean, minimum, maximum, standard deviation, variance, sum, and cardinality. On non-numeric fields, you can only see the total amount of messages containing that field, and the cardinality of the field, i.e. the number of unique values it has.

9.5.2 Quick values

Quick values helps you to find out the distribution of values for a field. Alongside a graphic representation of the common values contained in a field, Graylog will display a table with all different values, allowing you to see the number of times they appear. You can include any value in your search query by clicking on the magnifying glass icon located in the value row.
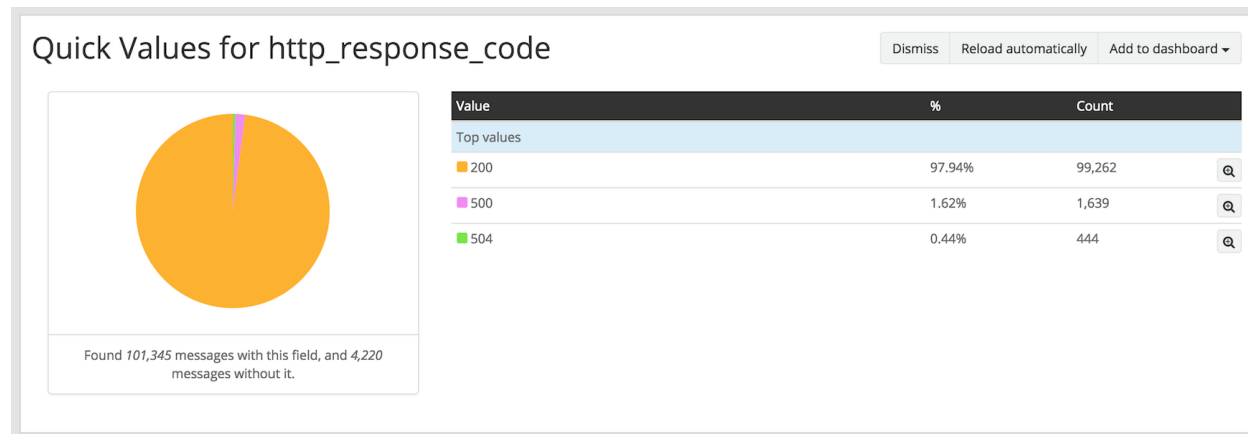
9.5.3 Field graphs

You can create field graphs for any numeric field, by clicking on the *Generate chart* button in the search sidebar. Using the options in the *Customize* menu on top of the field graph, you can change the statistical function used in the graph, the kind of graph to use to represent the values, the graph interpolation, as well as the time resolution.
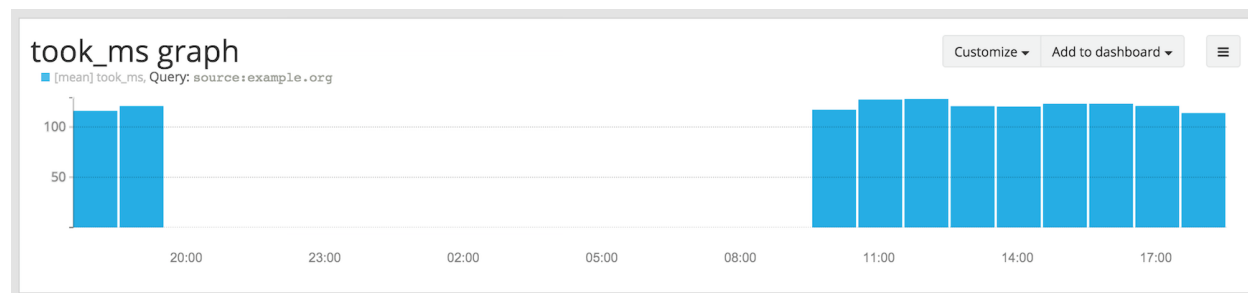
Once you have customized some field graphs, you can also combine them by dragging them from the hamburger icon on the top corner of the graph, and dropping them into another field graph. You can see the location of the hamburger icon and the end result in the following screenshots:
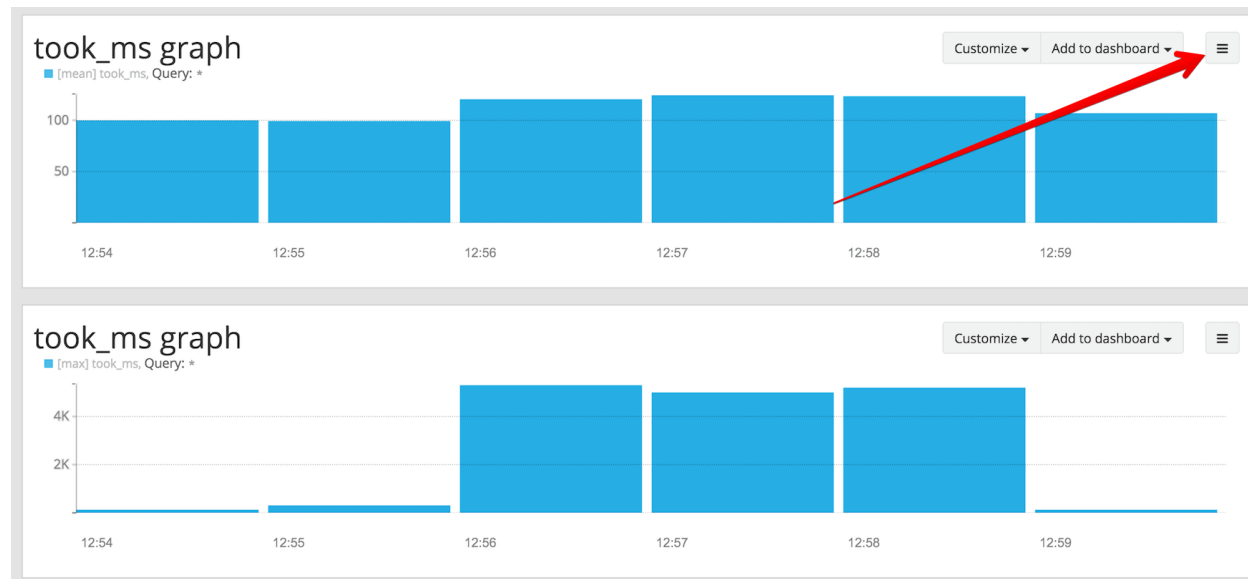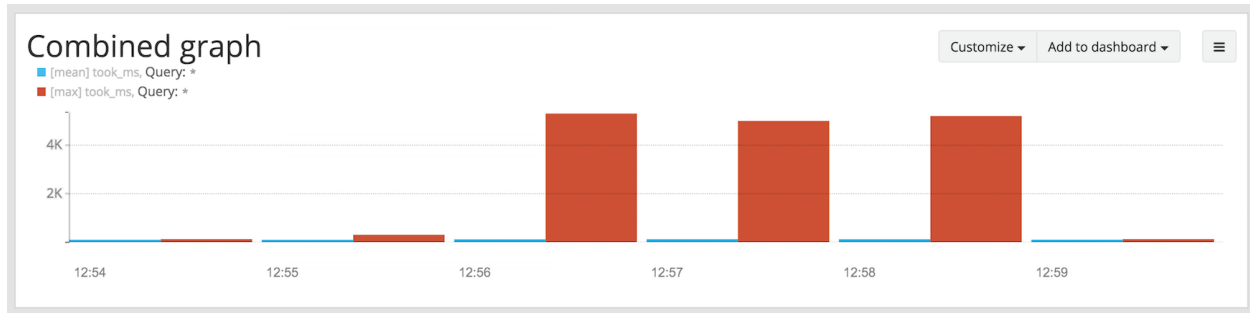
Field graphs appear every time you perform a search, allowing you to compare data, or combine graphs coming from different streams.

9.6 Decorators

Decorators allow you to alter message fields during search time automatically, while *preserving the unmodified message on disk*. Decorators are specially useful to make some data in your fields more readable, combine data in some field, or add new fields with more information about the message. As decorators are configured per stream (including the *default stream*), you are also able to present a single message in different streams differently.

As changes made by decorators are not persisted, you cannot search for decorated values or use field analyzers on them. You can still use those features in the original non-decorated fields.

Decorators are applied on a stream-level, and are shared among all users capable of accessing a stream, so all users can share the same results and benefit from the advantages decorators add.

Graylog includes some message decorators out of the box, but you can add new ones from pipelines or by writing your own as plugins.

In order to apply decorators to your search results, click on the *Decorators* tab in your search sidebar, select the decorator you want to apply from the dropdown, and click on *Apply*. Once you save your changes, the search results will already contain the decorated values.

Search result

Found **1,711 messages** in 79 ms, searched in 2 indices.

Add count to dashboard ▾   Save search criteria

More actions ▾

Fields   Decorators

Syslog Severity Mapper   × ▾   Apply

What are message decorators?

↕ Syslog Severity Mapper   Actions ▾

```
source_field: level
target_field: level
```

↕ Format String   Actions ▾

```
format_string: Request to
${controller}#${action} finished in
${took_ms}ms with code ${http_response_code}
require_all_fields: true
target_field: results
```

When you apply multiple decorators to the same search results, you can change the order in which they are applied at any time by using drag and drop in the decorator list.

9.6.1 Syslog severity mapper

The syslog severity mapper decorator lets you convert the numeric syslog level of syslog messages, to a human readable string. For example, applying the decorator to the `level` field in your logs would convert the syslog level `4` to `Warning (4)`.

To apply a syslog severity mapper decorator, you need to provide the following data:

- **Source field**: Field containing the numeric syslog level
- **Target field**: Field to store the human readable string. It can be the same one as the source field, if you wish to replace the numeric value on your search results

9.6.2 Format string

The format string decorator provides a simple way of combining several fields into one. It can also be used to modify the content of a field in, without altering the stored result in Elasticsearch.

To apply a format string decorator you need to provide the following data:

- **Format string**: Pattern used to format the resulting string. You can provide fields in the message by enclosing them in `${}`. E.g. `${source}` will add the contents of the `source` message field into the resulting string
- **Target field**: Field to store the resulting value
- **Require all fields** (optional): Check this box to only format the string when all other fields are present

For example, using the format string `Request to ${controller}#${action} finished in ${took_ms}ms with code ${http_response_code}`, could produce the text `Request to PostsController#show finished in 57ms with code 200`, and make it visible in one of the message fields in your search results.

9.6.3 Pipeline Decorator

The pipeline decorator provides a way to decorate messages by processing them with an existing *processing pipeline*. In contrast to using a processing pipeline, changes done to the message by the pipeline are not persisted. Instead, the pipeline is used at search time to modify the *presentation* of the message.

The prerequisite of using the pipeline decorator is that an existing pipeline is required.

Note: Please take note, that the pipeline you use for decoration should not be connected to a stream. This would mean that it is run twice (during indexing *and* search time) for each message, effectively rendering the second run useless.

When you are done creating a pipeline, you can now add a decorator using it on any number of streams. In order to create one, you proceed just like for any other decorator type, by clicking on the *Decorator* sidebar, selecting the type ("Pipeline Processor Decorator" in this case) and clicking the *Apply* button next to one.
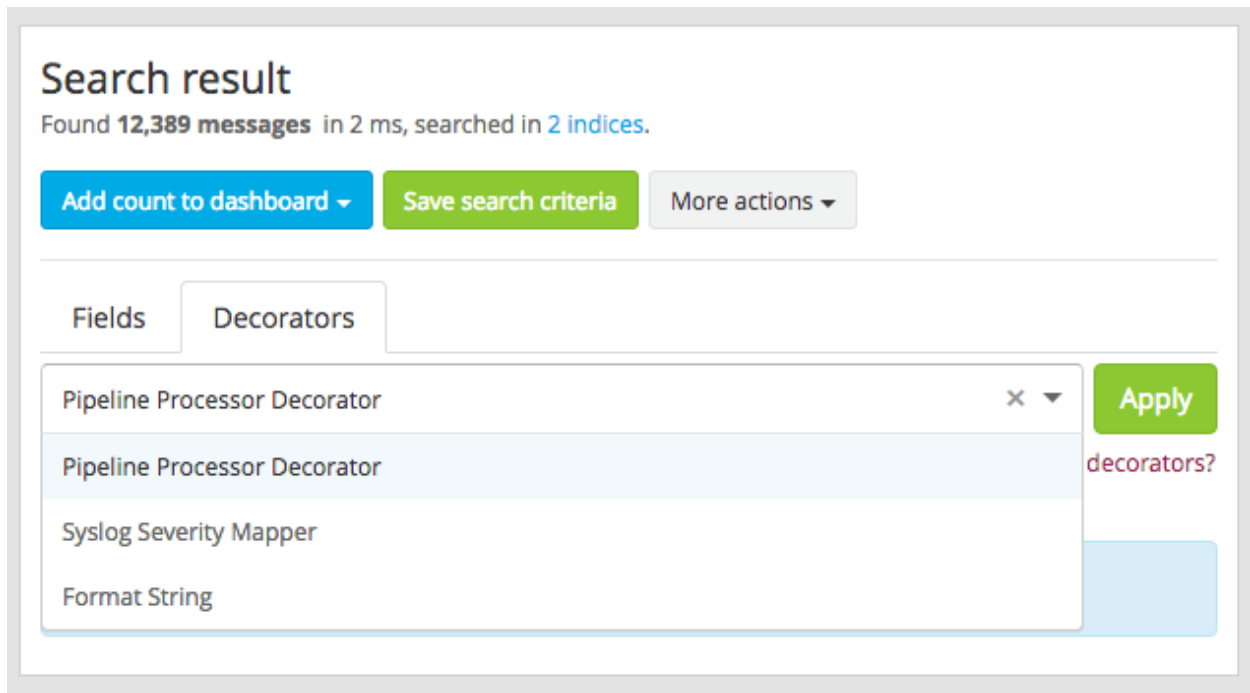
Upon clicking *Apply*, the pipeline to be used for decorating can be selected.
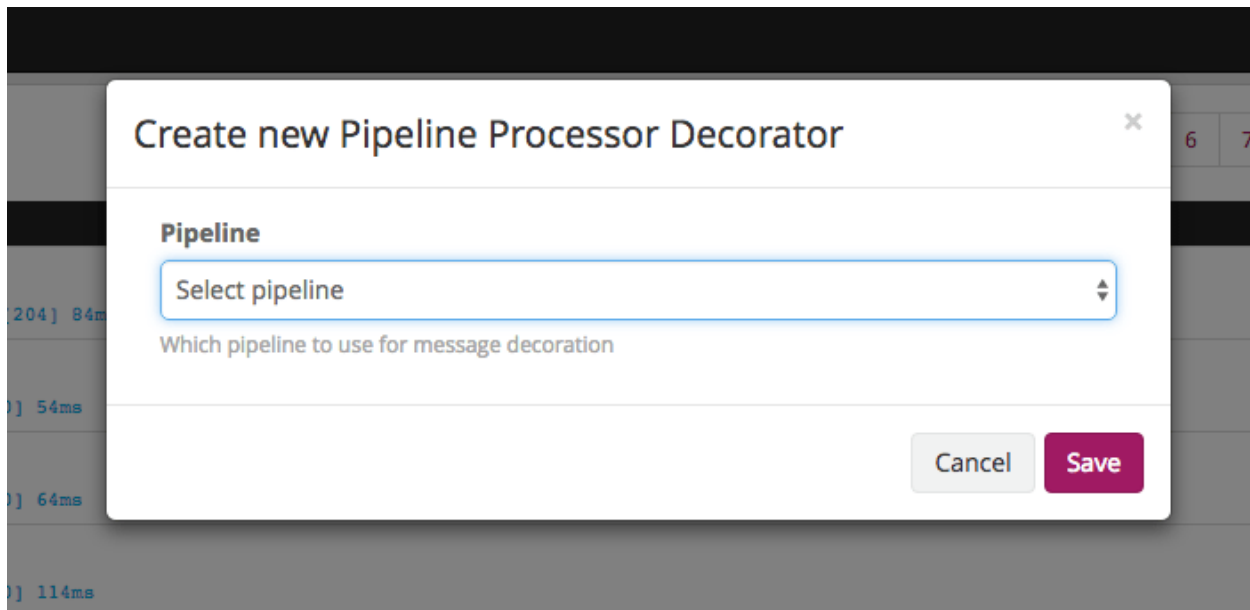
After selecting a pipeline and clicking *Save*, you are already set creating a new pipeline decorator.

9.6.4 Debugging decorators

When a message is not decorated as expected, or you need to know how it looked like originally, you can see all changes that were done during decoration by clicking "Show changes" in the message details.

In this view, deleted content is shown in red, while added content is shown in green. This means that added fields will have a single green entry, removed fields a single red entry and modified fields will have two entries, a red and a green one.

9.6.5 Further functionality

If the existing decorators are not sufficient for your needs, you can either search the Graylog marketplace, or *write your own decorator*.

9.7 Export results as CSV

It is also possible to export the results of your search as a CSV document. To do so, select all fields you want to export in the search sidebar, click on the *More actions* button, and select *Export as CSV*.
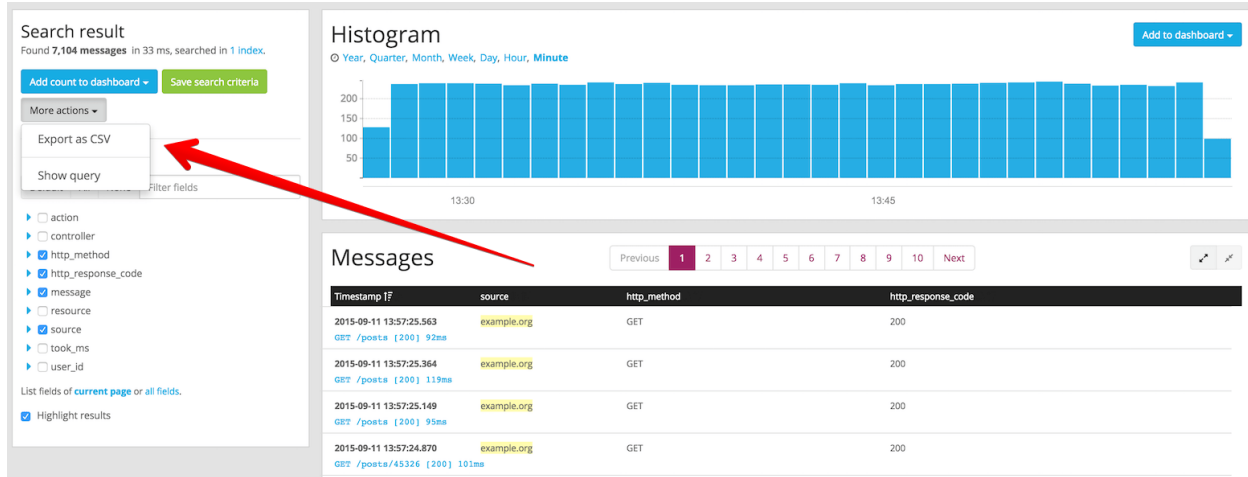
Hint: Some Graylog inputs keep the original message in the `full_message` field. If you need to export the original message, you can do so by clicking on the *List all fields* link at the bottom of the sidebar, and then selecting the `full_message` field.

> **Warning:** Exporting results to a CSV will **not** preserve sorting because Graylog is using the virtual `_doc` field to "sort" documents for performance reasons. If you need to have the exported data ordered you will need to either make a scroll query to ElasticSearch and process it after, or to download the file and post process it via other means.

9.8 Search result highlighting

Graylog supports search result highlighting since v0.20.2:
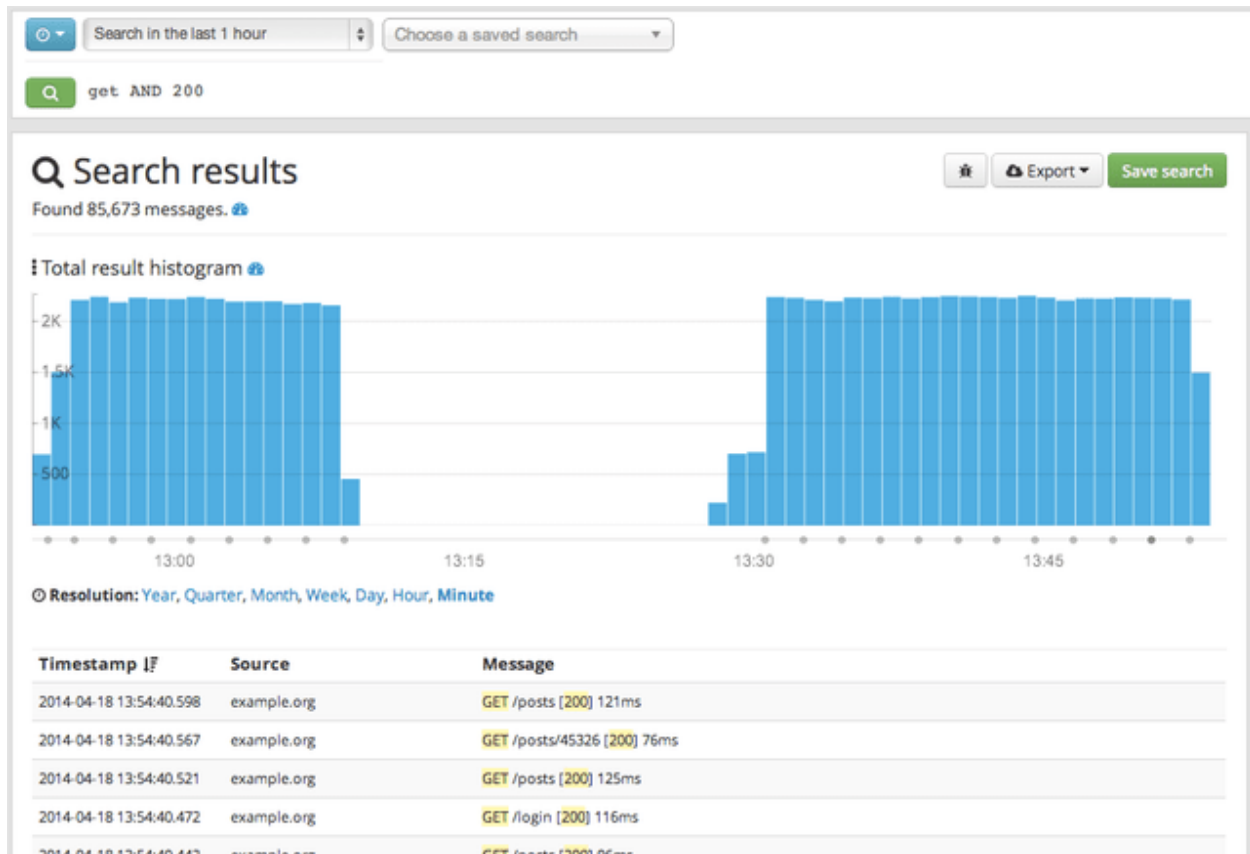
9.8.1 Enabling/Disabling search result highlighting

Using search result highlighting will result in slightly higher resource consumption of searches. You can enable and disable it using a configuration parameter in the `graylog.conf` of your Graylog nodes:

```
allow_highlighting = true
```

9.9 Search configuration

Graylog allows customizing the options allowed to search queries, like limiting the time range users can select or configuring the list of displayed relative time ranges.

All search configuration settings can be customized using the web interface on the *System -> Configurations* page in the *Search configuration* section.

9.9.1 Query time range limit

Sometimes the amount of data stored in Graylog is quite big and spans a wide time range (e. g. multiple years). In order to prevent normal users from accidentally running search queries which could use up lots of resources, it is possible to limit the time range that users are allowed to search in.

Using this feature, the time range of a search query exceeding the configured query time range limit will automatically be adapted to the given limit.

The query time range limit is a *duration* formatted according to ISO 8601 following the basic format `P<date>T<time>` with the following rules:

Designator	Description
P	Duration designator (for period) placed at the start of the duration representation
Y	Year designator that follows the value for the number of years
M	Month designator that follows the value for the number of months
W	Week designator that follows the value for the number of weeks
D	Day designator that follows the value for the number of days
T	Time designator that precedes the time components of the representation
H	Hour designator that follows the value for the number of hours
M	Minute designator that follows the value for the number of minutes
S	Second designator that follows the value for the number of seconds

Examples:

ISO 8601 duration	Description
P30D	30 days
PT1H	1 hour
P1DT12H	1 day and 12 hours

More details about the format of ISO 8601 durations can be found on Wikipedia.

9.9.2 Relative time ranges

The list of time ranges displayed in the *Relative time frame selector* can be configured, too. It consists of a list of *ISO 8601* durations which the users can select on the search page.

Update Search Configuration ✕

☐ Enable query limit

Relative Timerange Options

Configure the available options for the **relative** time range selector as **ISO8601 duration**

PT5M	5	Search in the last 5 minutes	🗑
PT15M	15	Search in the last 15 minutes	🗑
PT30M	30	Search in the last 30 minutes	🗑
PT1H	1h	Search in the last 1 hour	🗑
PT2H	2h	Search in the last 2 hours	🗑
PT8H	8h	Search in the last 8 hours	🗑
P1D	1d	Search in the last 1 day	🗑
P2D	2d	Search in the last 2 days	🗑
P5D	5d	Search in the last 5 days	🗑
P7D	7d	Search in the last 7 days	🗑

EXTENDED SEARCH

10.1 Views

A views contains a set of queries. Each query has a collection of widgets which display messages and charts depending on the search string entered in the search bar and the selected time range. A view can be saved with a name to keep the current progress and continue later on. Saved views can be shared, so other people can use them in their processes. For a better work flow Parameters can be added in the search query. Parameters are part of the Graylog Enterprise plugin.
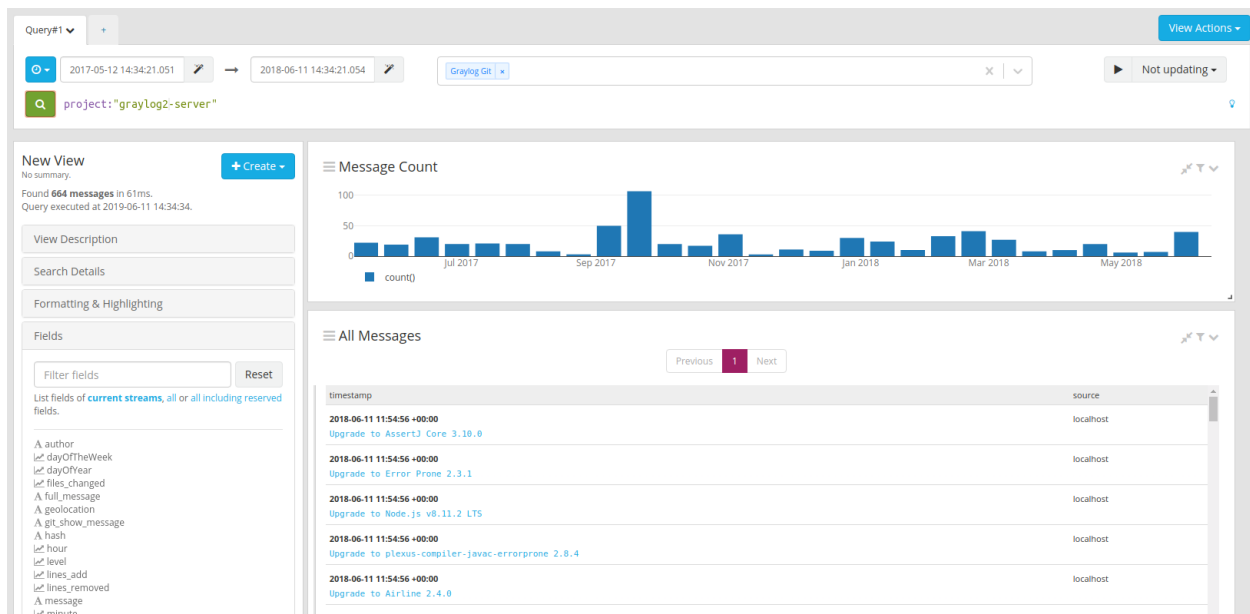
10.2 Widgets

A widget is either a Message Table or an Aggregation. It can be freely placed inside a query. A widget can be edited or duplicated by clicking on the chevron on the right side in the head of the widget. Next to the chevron is a filter symbol, where filters can be added to the top search query.
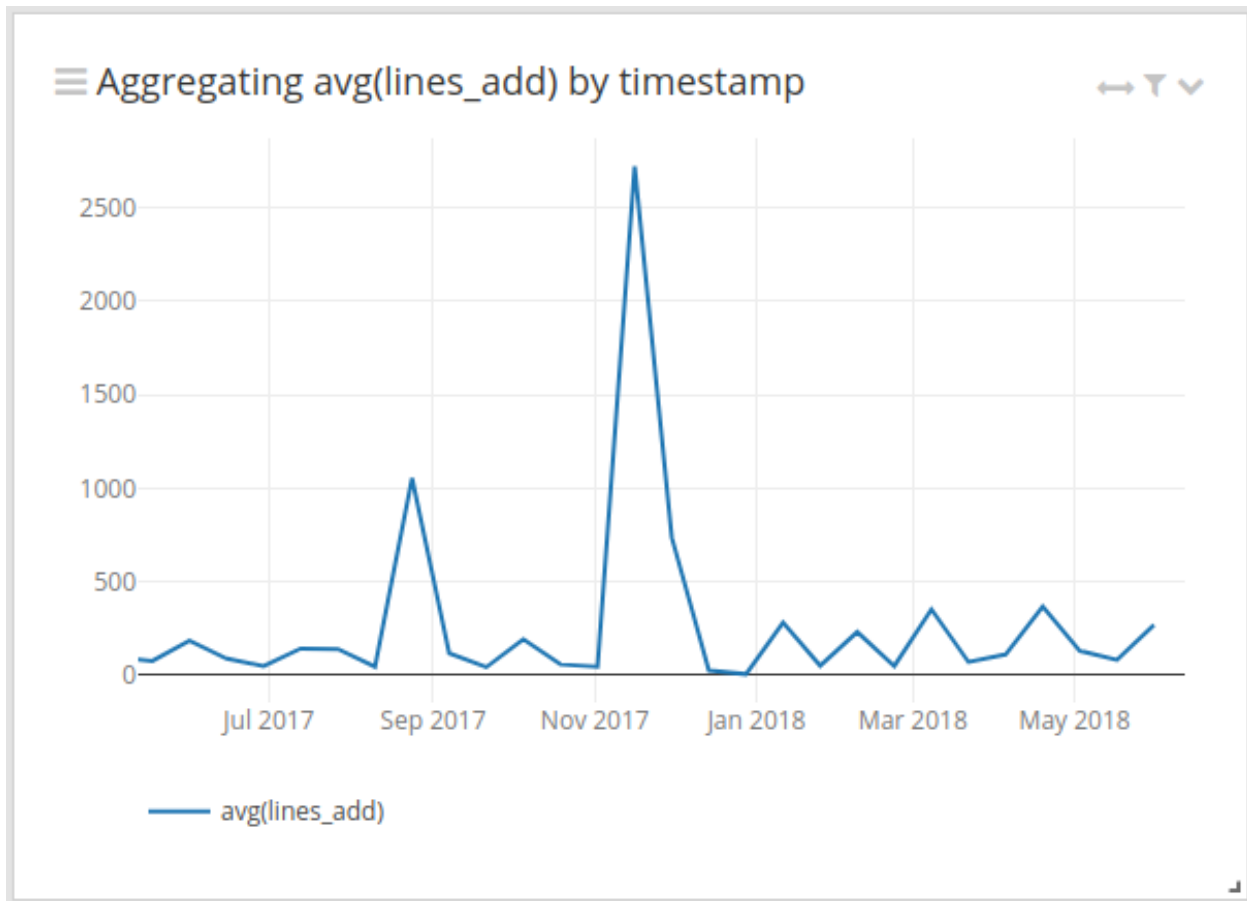
10.3 Aggregation

The goal of an aggregation is to reduce the number of data points in a meaningful way to get an answer from them. Data points can be numeric field types in a message (e.g. a took_ms field which contains how long a page needed to be rendered). Or string values which can be used for grouping the aggregation (e.g an action field which contains the name of the controller action).

10.3.1 Creating an aggregation

By clicking on + *Create -> Custom Aggreatation* a new empty widget will be shown on the very top of the Extended Search page. A click on the chevron icon on the right side of the head will open the widget in a modal in the edit mode.
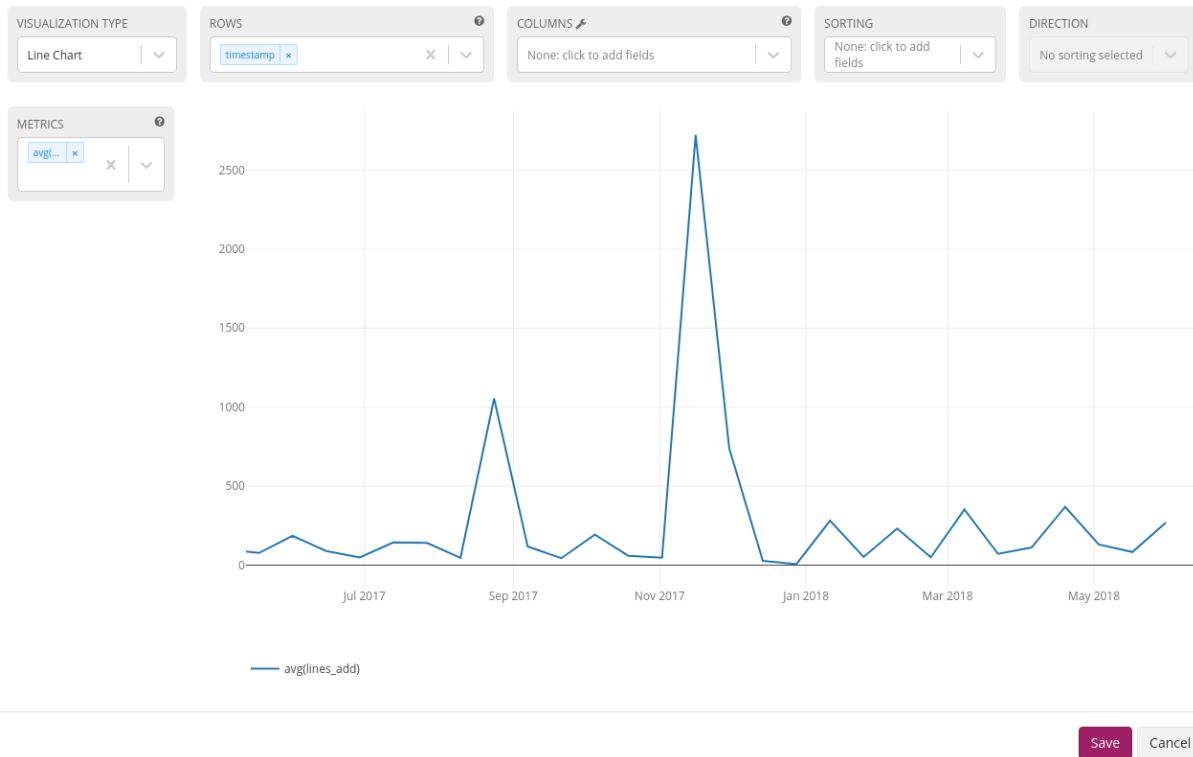
METRICS **METRICS** are a collection of functions to aggregate data points. The result of the aggregation depends on the grouping of **ROWS** and/or **COLUMNS**. The data points of a field will be aggregated to the grouping. *Example* The `avg()` function will find the average of the numeric data points `took_ms` around the configured grouping.

ROWS/COLUMNS Allows selecting fields whose values will be used to group results into new rows/columns. If the field is a `timestamp` for a row it will divide the data points into intervals. Otherwise the aggregation will take by default up to 15 elements of the selected field and apply the selected **METRICS** function to the data points. *Example* The `timestamp` field is aggregated with `avg()` on `took_ms`. The column `action` will give the average loading time for a page per action for every 5 minutes.

VISUALIZATION To display the result of an aggregation it is often easier to compare lots of result values graphically. `Bar Chart`, `Data Table`, `Line Chart`, `Pie Chart`, `Scatter Plot`, `Single Number` or `World Map` can be used as **VISUALIZATION**. The `World Map` needs geographical points in the form of `latitude`, `longitude`.

SORTING/DIRECTION The order of the result values can be configured here. **SORTING** defines by which field the sorting should happen and **DIRECTION** configures if it will be `ascending` or `descending`.

10.4 Message Table

The Message Table displays the messages and their fields. The Message Table can be configured to show the message fields and the actual message. The actual message is rendered in a blue font, below the fields. Clicking on a message row opens the detailed view of a message with all its fields.

10.5 Value and Field Actions

In the Sidebar and on Data Tables and Detail Message Rows are values and fields visible. By clicking on a value or a field a context menu will be shown where different actions can be executed.

10.5.1 Field actions

Based on the type of the field and where the menu is opened different Field actions are shown when a field name (and not its value) is clicked.
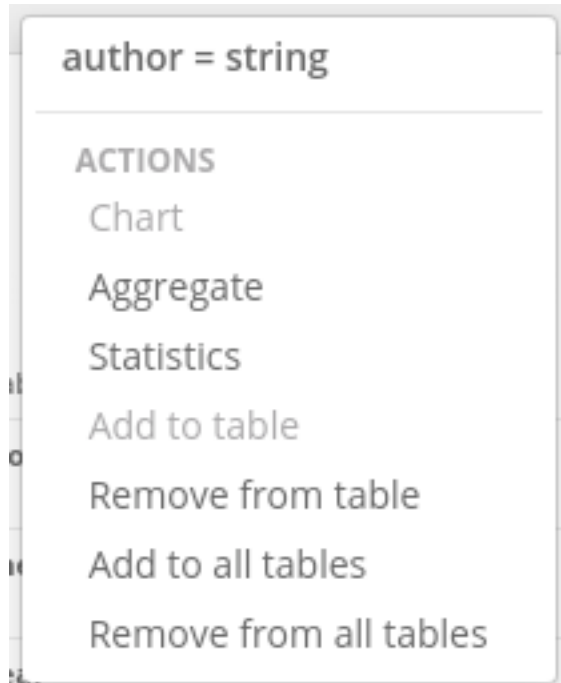
Chart This will generate a new Widget containing a line chart where the fields average value is displayed over time. This chart can be taken as an starting point for a more defined aggregation. This is only possible on fields from a numerical type.

Aggregate This action will generate a new Widget containing a data table where the fields value are listed in the rows and the count of occurrence will be displayed next to it. This was formerly known as the "Quick Values" action.

Statistics Here the field values will be given to various statistics function depending on the type of the field. The result will be displayed in a Data Table Widget.

Add to table Add the field to the displayed fields of the message table where the Field Actions menu is shown.

Add to all tables Add the field to the displayed fields of all tables.

Remove from table Remove the field from the list displayed fields from this table.

Remove from table Remove the field from the list displayed fields from all tables.

10.5.2 Value actions

The value actions produce different results depending on the type of the value and where the menu is opened. The following actions can be executed.
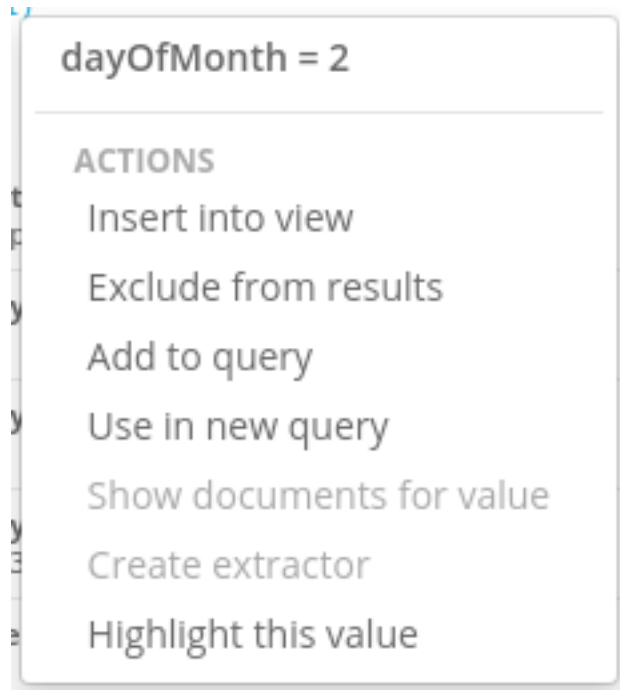
Insert into view This action will open up a modal where a view can be selected. A select-able list of Parameters will be shown from the selected view and after choosing a parameter a new browser tab will be opened containing the view with the value used in the parameter. This action is only available in enterprise Graylog.

Exclude from results Will add *NOT field:value* to the query to exclude all results where the field contains the value of the value action.

Add to query Will add *field:value* to the query to filter the results additionally for where the field has the value of the value action.

Use in new query Will open a new view tab with *field:value* as query string.

Show documents for value Available in Data Tables it will show the documents which where aggregated to display this value.

Create extractor For values of type string in Message Tables a short cut to create an extractor is given with this action.

Highlight this value This action will highlight this value for this field in all Message Tables and Data Tables.

ELEVEN

STREAMS

11.1 What are streams?

The Graylog streams are a mechanism to route messages into categories in realtime while they are processed. You define rules that instruct Graylog which message to route into which streams. Imagine sending these three messages to Graylog:

```
message: INSERT failed (out of disk space)
level: 3 (error)
source: database-host-1

message: Added user 'foo'.
level: 6 (informational)
source: database-host-2

message: smtp ERR: remote closed the connection
level: 3 (error)
source: application-x
```

One of the many things that you could do with streams is creating a stream called *Database errors* that is catching every error message from one of your database hosts.

Create a new stream with these rules, selecting the option to match all rules:

- Field `level` must be greater than `4`
- Field `source` must match regular expression `^database-host-\d+`

This will route every new message with a `level` higher than *WARN* and a `source` that matches the database host regular expression into the stream.

A message will be routed into every stream that has all (or any) of its rules matching. This means that a message can be part of many streams and not just one.

The stream is now appearing in the streams list and a click on its title will show you all database errors.

Streams can be used to be alerted in case certain condition happens. We cover more topics related to alerts in *Alerts*.

11.1.1 What's the difference to saved searches?

The biggest difference is that streams are processed in realtime. This allows realtime alerting and forwarding to other systems. Imagine forwarding your database errors to another system or writing them to a file by regularly reading them from the message storage. Realtime streams do this much better.

Another difference is that searches for complex stream rule sets are always comparably cheap to perform because a message is *tagged* with stream IDs when processed. A search for Graylog internally always looks like this, no matter how many stream rules you have configured:

```
streams:[STREAM_ID]
```

Building a query with all rules would cause significantly higher load on the message storage.

11.2 How do I create a stream?

1. Navigate to the streams section from the top navigation bar.

2. Click "Create stream".

3. Save the stream after entering a name and a description. For example *All error messages* and *Catching all error messages from all sources*. The stream is now saved but **not yet activated**.

4. Click on "Edit rules" for the stream you just created. That will open a page where you can manage and test stream rules.

5. Choose how you want to evaluate the stream rules to decide which messages go into the stream:

   • *A message must match all of the following rules* (logical AND): Messages will only be routed into the stream if all rules in the stream are fulfilled. This is the default behavior

   • *A message must match at least one of the following rules* (logical OR): Messages will be routed into the stream if one or more rules in the stream are fulfilled

6. Add stream rules, by indicating the field that you want to check, and the condition that should satisfy. Try the rules against some messages by loading them from an input or manually giving a message ID. Once you are satisfied with the results, click on "I'm done".

7. The stream is still paused, click on the "Start stream" button to activate the stream.

11.3 Index Sets

For starters, you should read *Index model* for a comprehensive description of the index set functionality in Graylog.

Every stream is assigned to an index set which controls how messages routed into that stream are being stored into Elasticsearch. The stream overview in the web interface shows the assigned index set for each stream.

Index sets can be assigned to a stream when creating the stream and changed later when editing the stream settings.

Important: Graylog will not automatically copy messages into new Elasticsearch indices if another index set is being assigned to a stream.

Graylog routes every message into the **All messages** stream by default, unless the message is removed from this stream with a pipeline rule (see *Processing Pipelines*) or it's routed into a stream marked with **Remove matches from 'All messages' stream**.

The latter is useful if messages should be stored with different settings than the ones in the **Default index set**, for example web server access logs should only be stored for 4 weeks while all other messages should be stored for 1 year.

11.3.1 Storage requirements

Graylog writes messages once for each index set into Elasticsearch. This means that if all streams are using the **Default index set**, each message will be written exactly once into Elasticsearch, no matter into how many streams the message has been sent. This can be thought of a kind of de-duplication.

If some streams use other index sets and the **Remove matches from 'All messages' stream** setting is not enabled, messages will be written into Elasticsearch at least twice, once for the **Default index set** and once for the assigned index set. This means that the same message will be stored in two or more indices in Elasticsearch with different index settings.

Unless you explicitly want to store messages multiple times in different Elasticsearch indices, either assign the **Default index set** to the respective streams or enable the **Remove matches from 'All messages' stream** setting for the respective streams.

11.4 Outputs

The stream output system allows you to forward every message that is routed into a stream to other destinations.

Outputs are managed globally (like message inputs) and not for single streams. You can create new outputs and activate them for as many streams as you like. This way you can configure a forwarding destination once and select multiple streams to use it.

Graylog ships with default outputs and can be extended with *Plugins*.

11.5 Use cases

These are a few example use cases for streams:

- Forward a subset of messages to other data analysis or BI systems to reduce their license costs.
- Monitor exception or error rates in your whole environment and broken down per subsystem.
- Get a list of all failed SSH logins and use *quick values* to analyze which user names where affected.
- Catch all HTTP POST requests to `/login` that were answered with a HTTP 302 and route them into a stream called *Successful user logins*. Now get a chart of when users logged in and use *quick values* to get a list of users that performed the most logins in the search time frame.

11.6 How are streams processed internally?

Every message that comes in is matched against the rules of a stream. For messages satisfying *all* or *at least one* of the stream rules (as configured in the stream), the internal ID of that stream is stored in the `streams` array of the processed message.

All analysis methods and searches that are bound to streams can now easily narrow their operation by searching with a `streams:[STREAM_ID]` limit. This is done automatically by Graylog and does not have to be provided by the user.
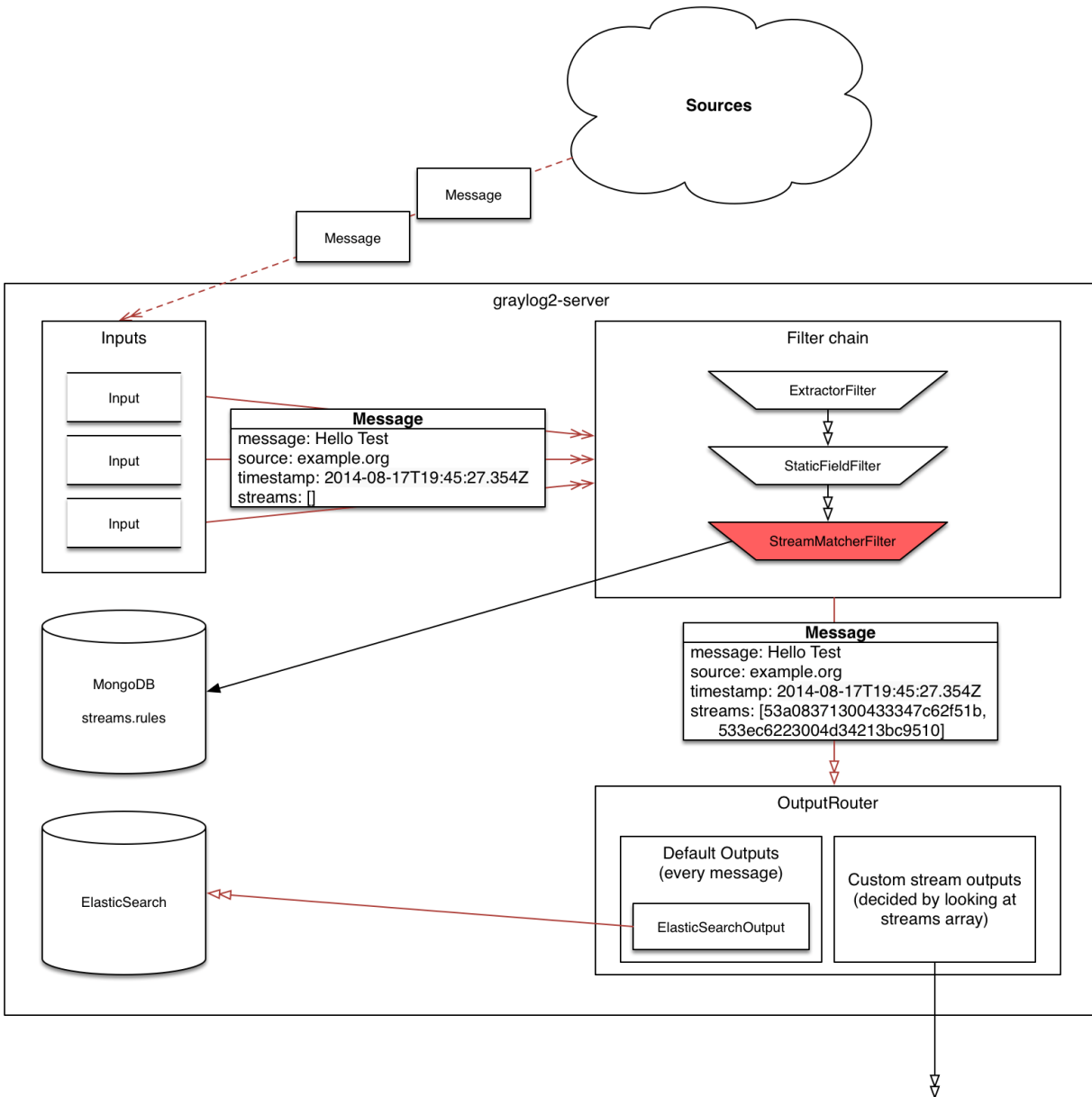
11.7 Stream Processing Runtime Limits

An important step during the processing of a message is the stream classification. Every message is matched against the user-configured stream rules. The message is added to the stream if all or any rules of a stream matches, depending on what the user chose. Applying stream rules is done during the indexing of a message only, so the amount of time spent for the classification of a message is crucial for the overall performance and message throughput the system can handle.

There are certain scenarios when a stream rule takes very long to match. When this happens for a number of messages, message processing can stall, messages waiting for processing accumulate in memory and the whole system could become non-responsive. Messages are lost and manual intervention would be necessary. This is the worst case scenario.

To prevent this, the runtime of stream rule matching is limited. When it is taking longer than the configured runtime limit, the process of matching this exact message against the rules of this specific stream is aborted. Message processing in general and for this specific message continues though. As the runtime limit needs to be configured pretty high (usually a magnitude higher as a regular stream rule match takes), any excess of it is considered a fault and is recorded for this stream. If the number of recorded faults for a single stream is higher than a configured threshold, the stream rule set of this stream is considered faulty and the stream is disabled. This is done to protect the overall stability and performance of message processing. Obviously, this is a tradeoff and based on the assumption, that the total loss of one or more messages is worse than a loss of stream classification for these.

There are scenarios where this might not be applicable or even detrimental. If there is a high fluctuation of the message load including situations where the message load is much higher than the system can handle, overall stream matching can take longer than the configured timeout. If this happens repeatedly, all streams get disabled. This is a clear indicator that your system is overutilized and not able to handle the peak message load.

11.7.1 How to configure the timeout values if the defaults do not match

There are two configuration variables in the configuration file of the server, which influence the behavior of this functionality.

- `stream_processing_timeout` defines the maximum amount of time the rules of a stream are able to spend. When this is exceeded, stream rule matching for this stream is aborted and a fault is recorded. This setting is defined in milliseconds, the default is `2000` (2 seconds).

- `stream_processing_max_faults` is the maximum number of times a single stream can exceed this runtime limit. When it happens more often, the stream is disabled until it is manually reenabled. The default for this setting is `3`.

11.7.2 What could cause it?

If a single stream has been disabled and all others are doing well, the chances are high that one or more stream rules are performing bad under certain circumstances. In most cases, this is related to stream rules which are utilizing regular expressions. For most other stream rules types the general runtime is constant, while it varies very much for regular expressions, influenced by the regular expression itself and the input matched against it. In some special cases, the difference between a match and a non-match of a regular expression can be in the order of 100 or even 1000. This is caused by a phenomenon called *catastrophic backtracking*. There are good write-ups about it on the web which will help you understanding it.

11.7.3 Summary: How do I solve it?

1. Check the rules of the stream that is disabled for rules that could take very long (especially regular expressions).

2. Modify or delete those stream rules.

3. Re-enable the stream.

11.8 Programmatic access via the REST API

Many organisations already run monitoring infrastructure that are able to alert operations staff when incidents are detected. These systems are often capable of either polling for information on a regular schedule or being pushed new alerts - this article describes how to use the Graylog Stream Alert API to poll for currently active alerts in order to further process them in third party products.

11.8.1 Checking for currently active alert/triggered conditions

Graylog stream alerts can currently be configured to send emails when one or more of the associated alert conditions evaluate to true. While sending email solves many immediate problems when it comes to alerting, it can be helpful to gain programmatic access to the currently active alerts.

Each stream which has alerts configured also has a list of active alerts, which can potentially be empty if there were no alerts so far. Using the stream's ID, one can check the current state of the alert conditions associated with the stream using the authenticated API call:

```
GET /streams/<streamid>/alerts/check
```

It returns a description of the configured conditions as well as a count of how many triggered the alert. This data can be used to for example send SNMP traps in other parts of the monitoring system.

Sample JSON return value:

```
{
  "total_triggered": 0,
  "results": [
    {
      "condition": {
        "id": "984d04d5-1791-4500-a17e-cd9621cc2ea7",
        "in_grace": false,
        "created_at": "2014-06-11T12:42:50.312Z",
        "parameters": {
          "field": "one_minute_rate",
          "grace": 1,
          "time": 1,
          "backlog": 0,
          "threshold_type": "lower",
          "type": "mean",
          "threshold": 1
        },
        "creator_user_id": "admin",
        "type": "field_value"
      },
      "triggered": false
    }
  ],
  "calculated_at": "2014-06-12T13:44:20.704Z"
}
```

Note that the result is cached for 30 seconds.

11.8.2 List of already triggered stream alerts

Checking the current state of a stream's alerts can be useful to trigger alarms in other monitoring systems, but if one wants to send more detailed messages to operations, it can be very helpful to get more information about the current state of the stream, for example the list of all triggered alerts since a certain timestamp.

This information is available per stream using the call:

```
GET /streams/<streamid>/alerts?since=1402460923
```

The since parameter is a unix timestamp value. Its return value could be:

```
{
  "total": 1,
  "alerts": [
    {
      "id": "539878473004e72240a5c829",
      "condition_id": "984d04d5-1791-4500-a17e-cd9621cc2ea7",
      "condition_parameters": {
        "field": "one_minute_rate",
        "grace": 1,
        "time": 1,
        "backlog": 0,
        "threshold_type": "lower",
        "type": "mean",
        "threshold": 1
      },
      "description": "Field one_minute_rate had a mean of 0.0 in the last 1 minutes␣
→with trigger condition lower than 1.0. (Current grace time: 1 minutes)",
      "triggered_at": "2014-06-11T15:39:51.780Z",
      "stream_id": "53984d8630042acb39c79f84"
    }
  ]
}
```

Using this information more detailed messages can be produced, since the response contains more detailed information about the nature of the alert, as well as the number of alerts triggered since the timestamp provided.

Note that currently a maximum of 300 alerts will be returned.

11.9 FAQs

11.9.1 Using regular expressions for stream matching

Stream rules support matching field values using regular expressions. Graylog uses the Java Pattern class to execute regular expressions.

For the individual elements of regular expression syntax, please refer to Oracle's documentation, however the syntax largely follows the familiar regular expression languages in widespread use today and will be familiar to most.

However, one key question that is often raised is matching a string in case insensitive manner. Java regular expressions are case sensitive by default. Certain flags, such as the one to ignore case sensitivity can either be set in the code, or as an inline flag in the regular expression.

To for example route every message that matches the browser name in the following user agent string:

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko)␣
→Chrome/32.0.1700.107 Safari/537.36
```

the regular expression `.*applewebkit.*` will not match because it is case sensitive. In order to match the expression using any combination of upper- and lowercase characters use the `(?i)` flag as such:

```
(?i).*applewebkit.*
```

Most of the other flags supported by Java are rarely used in the context of matching stream rules or extractors, but if you need them their use is documented on the same Javadoc page by Oracle.

11.9.2 Can I add messages to a stream after they were processed and stored?

No. Currently there is no way to re-process or re-match messages into streams.

Only new messages are routed into the current set of streams.

11.9.3 Can I write own outputs, alert conditions or notifications?

Yes. Please refer to the *Plugins* documentation page.

ALERTS

Alerts are created using Event Definitions that consist of Conditions. When a given condition is met it will be stored as an Event and can be used to trigger a notification. If your system has an enterprise license, then Events may be combined to create Correlations.

Graylog ships with default *alert conditions* and *alert notifications*, and both can be extended with *Plugins*.

12.1 Alerts & Events

As of Graylog 3.1.0, the Alerts page has changed to reflect a new method of generating Alerts. An Alert is triggered when a defined Event is detected. An Event is a condition that matches a log to a time period or aggregation. The Event may be used to group similar fields, change field content, or create new field content for use with Alerting and Correlation (an enterprise feature.)

If no Events have been defined, the Alerts & Events page will display the "Get Started!" button as shown below.
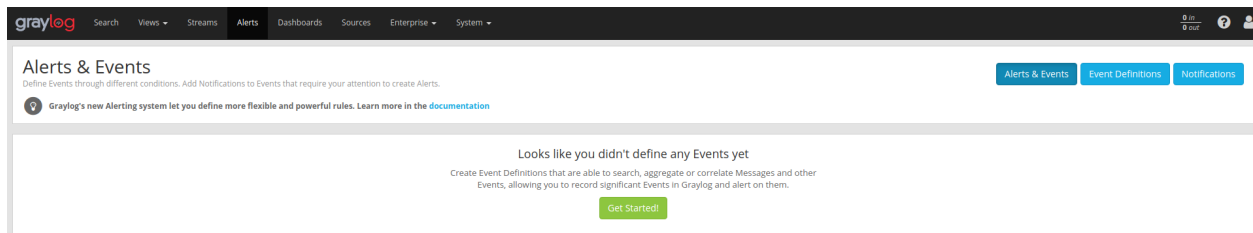
12.2 Defining an Event

12.2.1 Event Details

When you click on the "Get Started!" button you will be presented with a set of dialogues that allow you to set Title, Description, and Priority. You may click back on the selection bar to step backward in the definition process at any time.

12.2.2 Filter & Aggregation

Filter

By combining a Filter and an Aggregation you can specifically describe the criteria of an Event. Define a Filter by using Search Query in the same syntax as the Search page. Select a Stream in which the message can found. Define the window of time that the Filter will search backward to match messages. The search will be executed at the given interval. If the Filter matches an Event can be created. However, it may be useful to augment the filtered data with an aggregation

Filter

Add information to filter the log messages that are relevant for this Event Definition.

Search Query

origin:firewall

Search query that Messages should match. You can use the same syntax as in the Search page.

Streams (Optional)

Cisco_ASA ×

Select streams the search should include. Searches in all streams if empty.

Search within the last

1   minutes ▾

Execute search every

1   minutes ▾

Create Events for Definition if...

○ Filter has results

● Aggregation of results reaches a threshold

If the defined Filter matches messages currently within the Graylog Server, they will be displayed in the Filter Preview panel on the right.

Aggregation

An Aggregation can run a mathematical operation on either a numeric field value or the raw count of messages generated that match the Filter. Also, Aggregation can now group matches by a selected field before making the comparison. For instance, if the field username is defined, then it is possible to alert on five successive failed logins by a particular username. This use case is shown below.

Create Events for Definition

If		Is	Threshold
count()	username	>	5

12.2.3 Fields

Creating Custom Fields allows the Event to populate data from the original log into the Graylog Events index. This prevents the operator from having to run subsequent searches to get vital information. This can also be used to limit the amount of data sent to a Notification target. The Event will be recorded to the "All Events" stream and will contain the Custom Field, as well as the result of the Aggregation that triggered the Event.

Messages

Previous **1** Next

Timestamp ↓F	source
2019-08-20 06:41:27.376	freki
Firewall Event Count by Interface: Outside - count(src_interface)=12.0	
2019-08-20 06:41:27.376	freki
Firewall Event Count by Interface: outside - count(src_interface)=12.0	
2019-08-20 06:41:27.376	freki
Firewall Event Count by Interface: UDMZ2 - count(src_interface)=2.0	
2019-08-20 06:41:27.376	freki
Firewall Event Count by Interface: Outside - count(src_interface)=12.0	
2019-08-20 06:41:27.376	freki
Firewall Event Count by Interface: outside - count(src_interface)=12.0	

12.2.4 Notifications

After defining the Events that are needed to trigger an Alert it is possible to attach a Notifcation. By attaching a Notification to an Event or group of Events we can determine how and when information will flow out from Graylog. Notifications can be created by selecting the Notifications button under the Alerts tab, or by defining them in the Event workflow.

12.2.5 Alert notifications types explained

In this section we explain what the default alert notifications included in Graylog do, and how to configure them. Alert notifications are meant to be extensible through *Plugins*, you can find more types in the Graylog Marketplace or even create your own.

Important: In previous versions of Graylog (before 2.2.0), the email alarm notification was used, when alert conditions existed for a stream, but no alarm notification had been created before. This has been changed, so that if there is no alarm notification existing for a stream, alerts will be shown in the interface but no other action is performed. To help users coming from earlier version, there is a migration job which is being run once, creating the email alarm notification explicitly for qualifying streams, so the old behavior is preserved.

Email alert notification

The email alert notification can be used to send an email to the configured alert receivers when the conditions are triggered.

Make sure to check the *email-related configuration settings* in the Graylog configuration file.

Three configuration options are available for the alert notification to customize the email that will be sent. The *email body* and *email subject* are JMTE templates. JMTE is a minimal template engine that supports variables, loops and conditions. See the JMTE documentation for a language reference.

We expose the following objects to the templates.

stream The stream this alert belongs to.

- `stream.id` ID of the stream

- `stream.title` title of the stream

- `stream.description` stream description

stream_url A string that contains the HTTP URL to the stream.

check_result The check result object for this stream.

- `check_result.triggeredCondition` string representation of the triggered alert condition

- `check_result.triggeredAt` date when this condition was triggered

- `check_result.resultDescription` text that describes the check result

backlog A list of `message` objects. Can be used to iterate over the messages via `foreach`.

message (only available via iteration over the backlog object) The message object has several fields with details about the message. When using the `message` object without accessing any fields, the `toString()` method of the underlying Java object is used to display it.

- `message.id` autogenerated message id

- `message.message` the actual message text

- `message.source` the source of the message

- `message.timestamp` the message timestamp

- `message.fields` map of key value pairs for all the fields defined in the message

The `message.fields` fields can be useful to get access to arbitrary fields that are defined in the message. For example `message.fields.full_message` would return the `full_message` of a GELF message.

Editing alert configuration ✕

Title

Notify ops

E-Mail Subject

Graylog alert for stream: ${stream.title}: ${check_result.resultDescription}

The subject of sent out mail alerts

Sender (optional)

graylog@example.org

The sender of sent out mail alerts

E-Mail Body (optional)

```
###########
Alert Description: ${check_result.resultDescription}
Date: ${check_result.triggeredAt}
Stream ID: ${stream.id}
Stream title: ${stream.title}
Stream description: ${stream.description}
Alert Condition Title: ${alertCondition.title}
${if stream_url}Stream URL: ${stream_url}${end}

Triggered condition: ${check_result.triggeredCondition}
```

The template to generate the body from

User Receivers (optional)

Select User Receivers ▾

Graylog usernames that should receive this alert

E-Mail Receivers (optional)

✕ ops@graylog.example.org   ✕ ▾

E-Mail addresses that should receive this alert

Cancel   Save

HTTP alert notification

The HTTP alert notification lets you configure an endpoint that will be called when the alert is triggered.

Graylog will send a POST request to the notification URL including information about the alert. Here is an example of the payload included in a notification:

```
{
    'event_definition_id': '5d5ae2a165ea93218fcd7382',
    'event_definition_type': 'aggregation-v1',
    'event_definition_title': 'example',
    'event_definition_description': 'Message count alert condition',
    'job_definition_id': '5d5ae769683c3dba791b74bd',
    'job_trigger_id': '5d8cd93365ea937cbc6be163',
    'event': {
        'id': '01DNQ30W0Y20SZAMJZQS7HV4BN',
        'event_definition_type': 'aggregation-v1',
        'event_definition_id': '5d5ae2a165ea93218fcd7382',
        'origin_context': None,
        'timestamp': '2019-09-26T15:27:49.644Z',
        'timestamp_processing': '2019-09-26T15:28:51.230Z',
        'timerange_start': '2019-09-26T15:27:45.679Z',
        'timerange_end': '2019-09-26T15:28:45.678Z',
        'streams': [],
        'source_streams': ['5a5e3147683c3d7cd137d667'],
        'message': 'event: count()=2.0',
        'source': 'graylog.example.com',
        'key_tuple': [],
        'key': '',
        'priority': 2,
        'alert': True,
        'fields': {}
    },
    'backlog': [{
            'index': 'graylog_1097',
            'message': 'ERROR: This is an example error message',
            'timestamp': '2019-09-26T15:27:46.408Z',
            'source': '127.0.0.1',
            'stream_ids': ['579a14fee96e9f287aa9fd79', '5a5e3147683c3d7cd137d667'],
            'fields': {
                'via_input': 'input_name',
                'level': 6,
                'alert': 'example',
                'gl2_remote_ip': '127.0.0.1',
                'gl2_remote_port': 514,
                'gl2_message_id': '01DNQ2YWQFHSJZE9T6JEBSYSVY',
                'gl2_source_node': '7a05ad73-6141-43fa-a324-2ec2598e3645',
                'gl2_source_input': '5799f612e96e9f287aa9dcb0',
                'extreme_timestamp': 'Sep 26 10:27:46',
                'facility': 'local6'
            },
            'id': '30f9c681-e072-11e9-8e57-0050568a570f'
        }, {
            'index': 'graylog_1097',
            'message': 'ERROR: This is an example error message',
            'timestamp': '2019-09-26T15:27:49.644Z',
            'source': '127.0.0.1',
            'stream_ids': ['579a14fee96e9f287aa9fd79', '5a5e3147683c3d7cd137d667'],
```

```
            'fields': {
                'via_input': 'input_name',
                'level': 6,
                'alert': 'example',
                'gl2_remote_ip': '127.0.0.1',
                'gl2_remote_port': 514,
                'gl2_message_id': '01DNQ2YZWK5J8B6BXNGQVJX57D',
                'gl2_source_node': '113a4960-7cf2-43fc-b827-d81592dd1aea',
                'gl2_source_input': '5799f612e96e9f287aa9dcb0',
                'extreme_timestamp': 'Sep 26 10:27:49',
                'facility': 'local6'
            },
            'id': '32e78cc0-e072-11e9-8358-0050568a6438'
        }
    ]
}
```

Legacy Script alert notification

The Script Alert Notification lets you configure a script that will be executed when the alert is triggered.

Important: Script Alert Notification is an Enterprise Integrations plugin feature and thus requires an *Enterprise license*.

Create new Script Alert Callback   ✕

Title

Your Title

Script Path

your-script.py

The path to the script file within the script folder [/Absolute/Path/To/graylog2-server/scripts].
See requirements in documentation.

Script Timeout

30000

The script timeout in milliseconds.

Script Arguments (optional)

```
${stream_id}
${stream_name}
${stream_description}
${stream_url}
${alert_description}
${alert_triggered_at}
${condition_id}
${condition_description}
${condition_title}
${condition_type}
${condition_grace}
${condition_repeat_notifications}
```

The script arguments. Arguments can be space or new-line separated. See documentation.

☑ **Send Alert Data Through STDIN** (optional)

Send a JSON object containing alert data to the script through STDIN. See documentation for
parsing information.

Cancel   Save

These are the supported configuration options.

Script Path The path to where the script is located. Must me within the *permitted script path* (which is customizable).

Script Timeout The maximum time (in milliseconds) the script will be allowed to execute before being forcefully terminated.

Script Arguments String of parameters in which the delimiters are either a space-delimited or a new-line. The following argument variables may be used:

> **Stream** The stream this alert belongs to.
>
> > - `stream_id` ID of the stream
> > - `stream_name` title of the stream
> > - `stream_description` stream description
> > - `stream_url` a string that contains the URL to the view the relevant messages for the alert. Make sure to set the *HTTP URL* configuration parameter, as there is no default.
>
> **Alert** The check result object for this stream.
>
> > - `alert_description` text that describes the check result
> > - `alert_triggered_at` date when this condition was triggered
>
> **Condition** The available conditions to request are
>
> > - `condition_id` ID of the condition
> > - `condition_description` description of the condition
> > - `condition_title` title of the condition
> > - `condition_type` type of condition
> > - `condition_grace` grace period for the condition
> > - `condition_repeat_notification` repeat notification of the script

Send Alert Data Through STDIN Sends JSON alert data through standard in. You can use a JSON parser in your script.

```
{
  "stream_id": "000000000000000000000001",
  "stream_name": "All messages",
  "stream_description": "Stream containing all messages",
  "stream_url": "http://localhost:8080///streams/000000000000000000000001/
↪messages?rangetype=absolute&from=2019-01-25T20:57:50.793Z&to=2019-01-
↪25T21:02:50.793Z&q=*",
  "alert_description": "Stream received messages matching <has_field:\"true\">␣
↪(Current grace time: 0 minutes)",
  "alert_triggered_at": "2019-01-25T21:02:50.793Z",
  "condition_id": "ea9fcdff-2037-44f9-801e-099bf4bb3dbd",
  "condition_description": "field: has_field, value: true, grace: 0, repeat␣
↪notifications: false",
  "condition_title": "has_field",
  "condition_type": "field_content_value",
  "condition_grace": 0,
  "condition_parameters": {
    "backlog": 10,
    "repeat_notifications": false,
    "field": "has_field",
    "query": "*",
```

(continues on next page)

```
    "grace": 0,
    "value": "true"
  },
  "condition_repeat_notifications": false,
  "message_backlog": [
    {
      "has_field": "true",
      "gl2_remote_ip": "127.0.0.1",
      "gl2_remote_port": 56246,
      "streams": [
        "000000000000000000000001"
      ],
      "gl2_source_node": "e065896b-8a9a-4f45-83f2-e740525ed035",
      "_id": "92839500-20e4-11e9-8175-0637e3f7ecfc",
      "source": "example.org",
      "message": "Hello there",
      "gl2_source_input": "5c2e99687a90e30a3512f766",
      "facility": "test",
      "timestamp": "2019-01-25T21:02:49.423Z"
    },
    {
      "has_field": "true",
      "gl2_remote_ip": "127.0.0.1",
      "gl2_remote_port": 56245,
      "streams": [
        "000000000000000000000001"
      ],
      "gl2_source_node": "e065896b-8a9a-4f45-83f2-e740525ed035",
      "_id": "928087c0-20e4-11e9-8175-0637e3f7ecfc",
      "source": "example.org",
      "message": "Hello there",
      "gl2_source_input": "5c2e99687a90e30a3512f766",
      "facility": "test",
      "timestamp": "2019-01-25T21:02:49.403Z"
    }
  ],
  "message_backlog_size": 5
}
```

Script Alert Notification success is determined by its exit value; success equals zero. Any non-zero exit value will cause it to fail. Returning any error text through STDERR will also cause the alarm callback to fail.

Here is a sample Python script that shows all of the supported Script Alert Notification functionality (argument parsing, STDIN JSON parsing, STDOUT, exit values, and returning an exit value).:

```python
#!/usr/bin/env python3
import json
import sys
import time


# Function that prints text to standard error
def print_stderr(*args, **kwargs):
    print(*args, file=sys.stderr, **kwargs)

# Main function
if __name__ == "__main__":
```

```python
    # Print out all input arguments.
    sys.stdout.write("All Arguments Passed In: " + ' '.join(sys.argv[1:]) + "\n")
    sys.stdout.write("Stream Name: " + sys.argv[2] + "\n")
    sys.stdout.write("Stream Description: " + sys.argv[3] + "\n")
    sys.stdout.write("Alert Triggered At: " + sys.argv[6] + "\n")

    # Turn stdin.readlines() array into a string
    std_in_string = ''.join(sys.stdin.readlines())

    # Load JSON
    alert_object = json.loads(std_in_string)

    # Extract some values from the JSON.
    sys.stdout.write("Values from JSON: \n")
    sys.stdout.write("Stream ID: " + alert_object["stream_id"] + "\n")
    sys.stdout.write("Stream Name: " + alert_object["stream_name"] + "\n")
    sys.stdout.write("Alert Triggered At: " + alert_object["alert_triggered_at"]
→+ "\n")

    # Extract Message Backlog field from JSON.
    sys.stdout.write("\n\nFields:\n")
    for message in alert_object["message_backlog"]:
        for field in message.keys():
            print("Field: " + field)
            print("Value: " + str(message[field]))

    # Write to stderr if desired
    # print_stderr("Test return through standard error")

    # Return an exit value. Zero is success, non-zero indicates failure.
    exit(0)
```

12.2.6 Event Summary

When all of the components have been defined the Event Summary will be displayed to the user. At this time, the user may select a previous point in the Workflow to change a parameter. The user may also cancel out of the workflow, select done. The Event may be viewed under Alerts>Event Definitions.

12.3 All events stream

The All events stream can be used to view all previous Events that have been triggered. The Event is recorded with the fields defined in the Custom Fields portion of the Event.

DASHBOARDS

13.1 Why dashboards matter

Using dashboards allows you to build pre-defined views on your data to always have everything important just one click away.

Sometimes it takes domain knowledge to be able to figure out the search queries to get the correct results for your specific applications. People with the required domain knowledge can define the search query once and then display the results on a dashboard to share them with co-workers, managers, or even sales and marketing departments.

This guide will take you through the process of creating dashboards and storing information on them. At the end you will have a dashboard with automatically updating information that you can share with anybody or just a subset of people based on permissions.
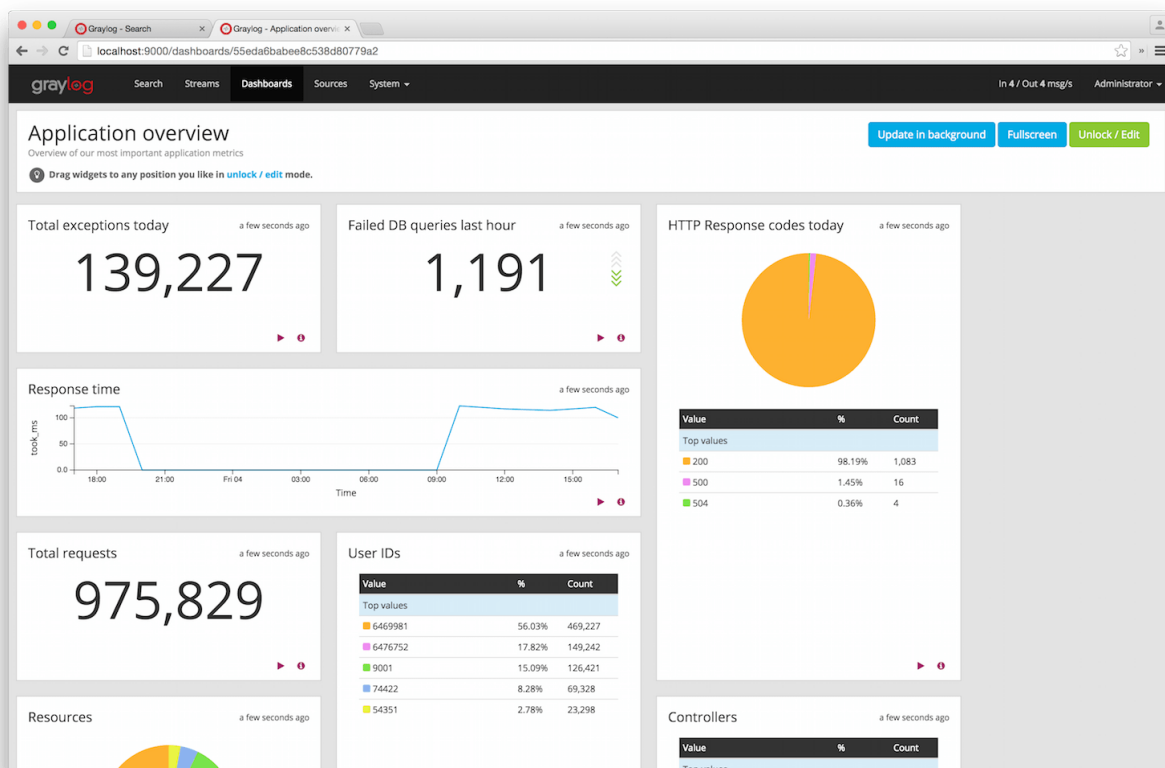
13.2 How to use dashboards

13.2.1 Creating an empty dashboard

Navigate to the *Dashboards* section using the link in the top menu bar of your Graylog web interface. The page is listing all dashboards that you are allowed to view. (More on permissions later.) Hit the *Create dashboard* button to create a new empty dashboard.

The only required information is a *title* and a *description* of the new dashboard. Use a specific but not too long title so people can easily see what to expect on the dashboard. The description can be a bit longer and could contain more detailed information about the displayed data or how it is collected.

Hit the *Create* button to create the dashboard. You should now see your new dashboard on the dashboards overview page. Click on the title of your new dashboard to see it. Next, we will be adding widgets to the dashboard we have just created.
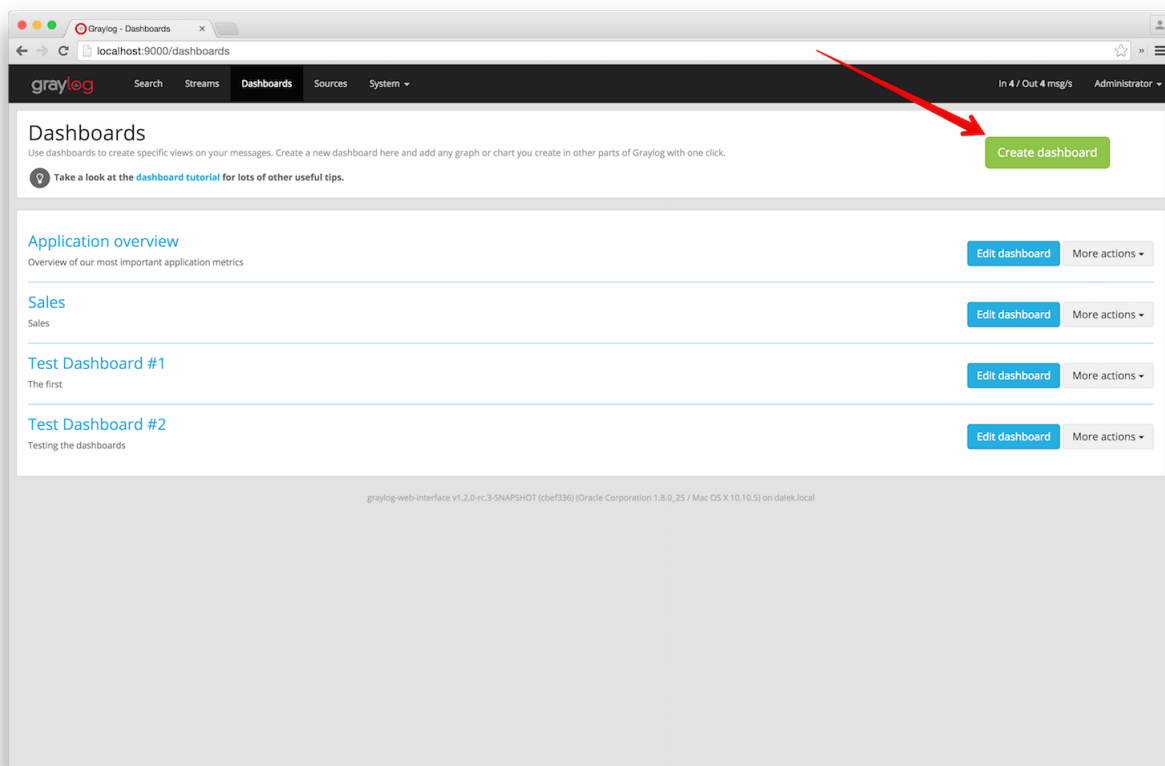
13.2.2 Adding widgets

You should have your empty dashboard in front of you. Let's add some widgets! You can add search result information to dashboards with a couple of clicks. The following search result types can be added to dashboards:

- Search result counts
- Search result histogram charts
- Statistical values
- Field value charts

- Stacked charts

- Quick values results

You can learn more about the different widget types in *Widget types explained*.

Once you can see the results of your search, you will see buttons with the "Add to dashboard" text, that will allow you to select the dashboard where the widget will be displayed, and configure the widget.
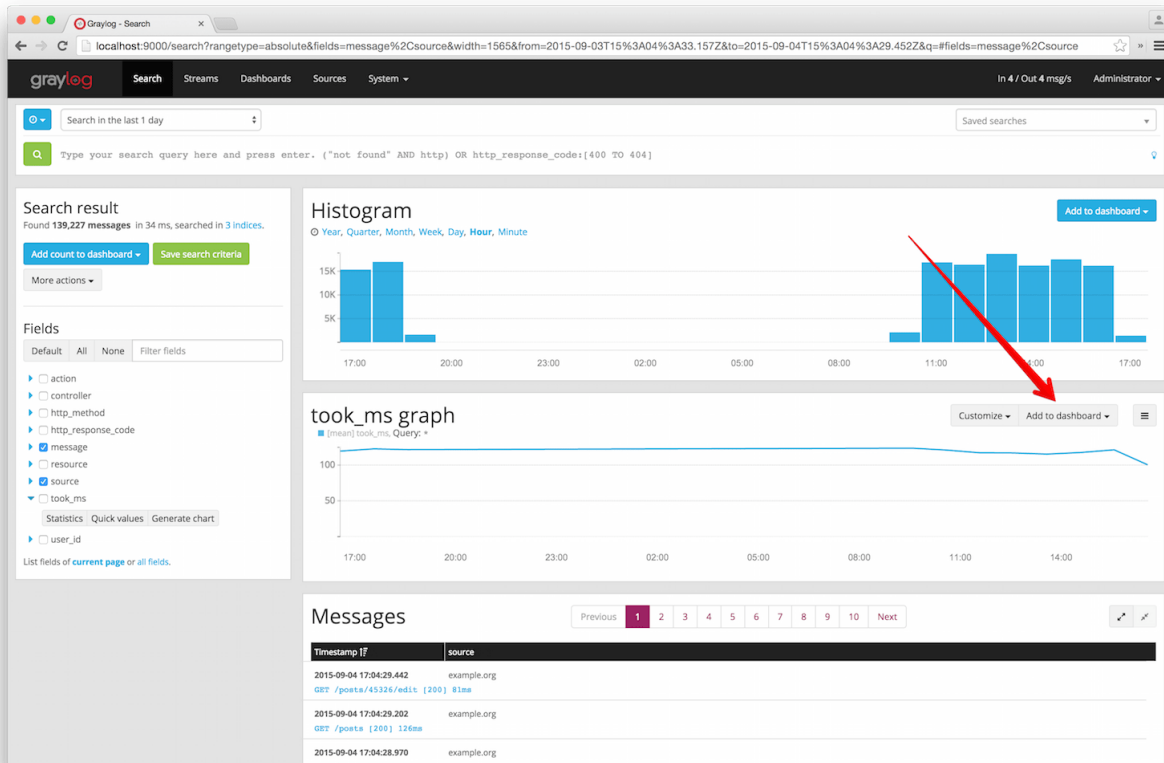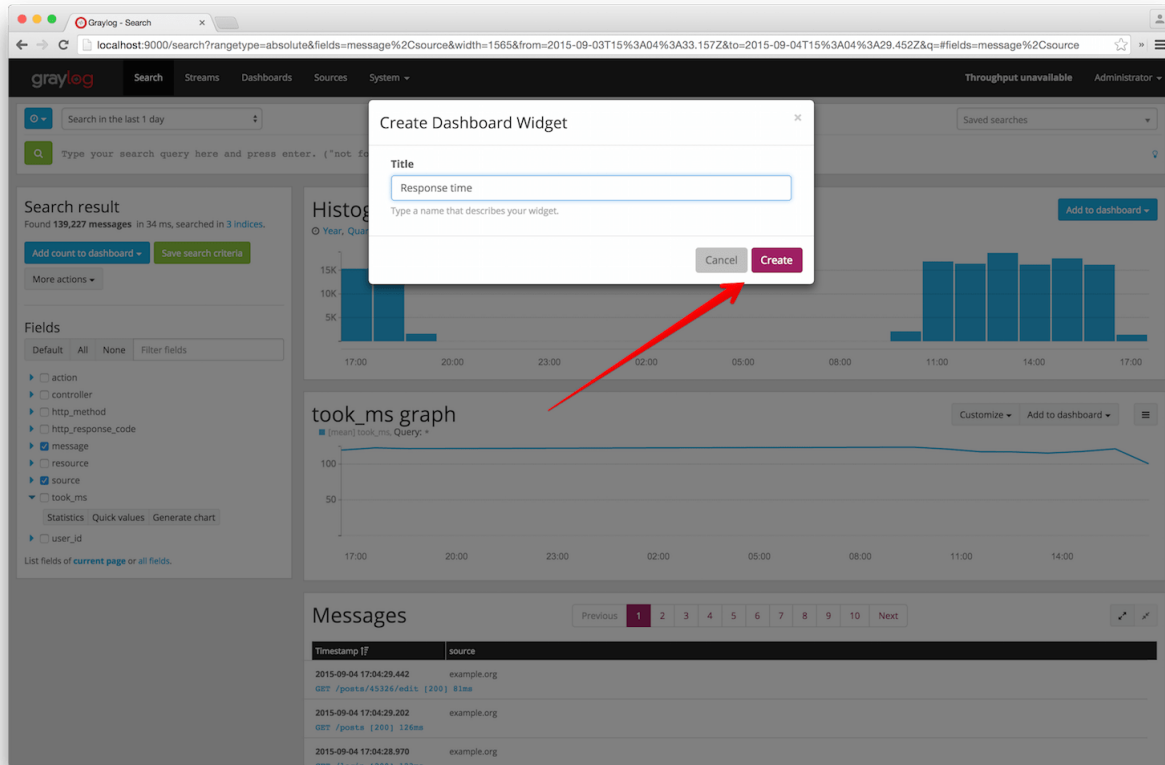
13.3 Examples

It is strongly recommended to read the getting started guide on basic searches and analysis first. This will make the following examples more obvious for you.

- **Top log sources today**
    - Example search: `*`, timeframe: Last 24 hours
    - Expand the `source` field in the sidebar and hit *Quick values*
    - Add quick values to dashboard
- **Number of exceptions in a given app today**
    - Example search: `source:myapp AND Exception`, timeframe: Last 24 hours
    - Add search result count to dashboard
- **Response time chart of a given app**
    - Example search: `source:myapp2`, any timeframe you want
    - Expand a field representing the response time of requests in the sidebar and hit *Generate chart*
    - Add chart to dashboard

13.4 Widgets from streams

You can of course also add widgets from stream search results. Every widget added this way will always be bound to streams. If you have a stream that contains every SSH login you can just search for everything (*) in that stream and store the result count as *SSH logins* on a dashboard.

13.5 Result

You should now see widgets on your dashboard. You will learn how to modify the dashboard, and edit widgets in the next chapter.
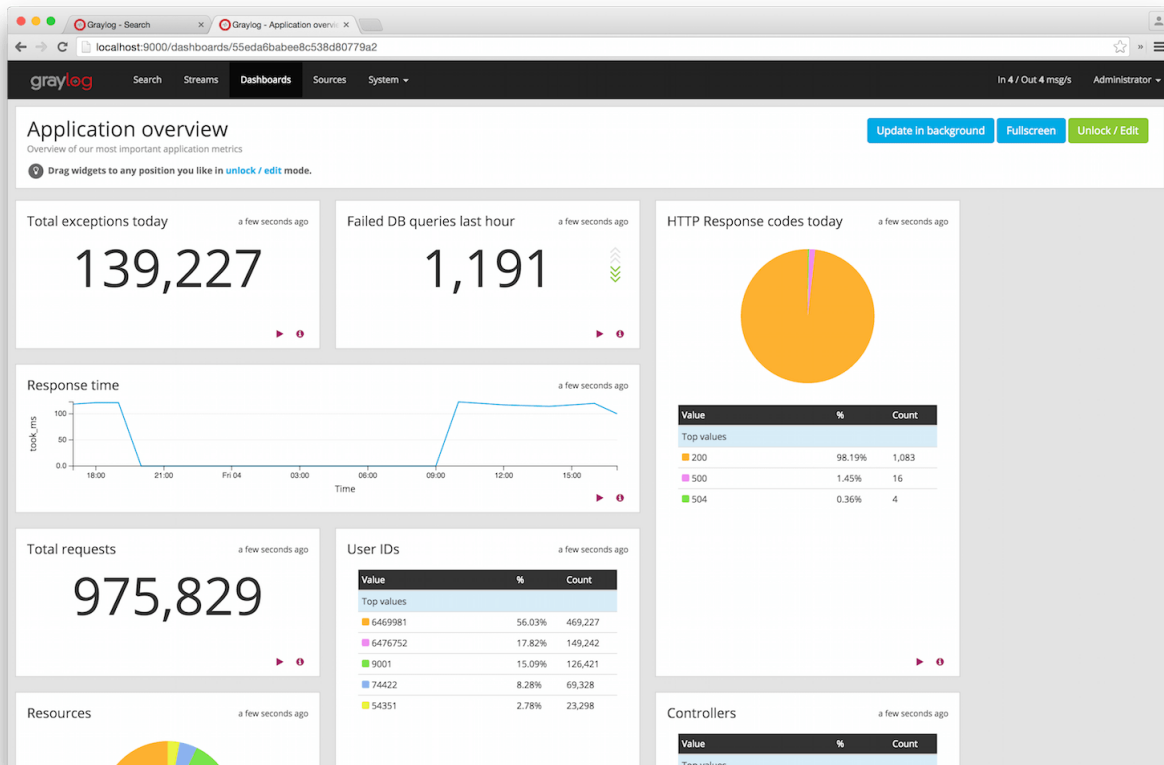
13.6 Widget types explained

Graylog supports a wide variety of widgets that allow you to quickly visualize data from your logs. This section intends to give you some information to better understand each widget type, and how they can help you to see relevant details from the many logs you receive.

13.6.1 Search result counts

This kind of widget includes a count of the number of search results for a given search. It can help you to quickly visualize things like the number of exceptions an application logs, or the number of requests your site receives.

All search result counts created with a relative time frame can additionally display trend information. The trend is calculated by comparing the count for the given time frame, with the one resulting from going further back the same amount of time. For example, to calculate the trend in a search result count with a relative search of *5 minutes ago*, Graylog will count the messages in the last 5 minutes, and compare that with the count of the previous 5 minutes.

13.6.2 Search result histogram charts

The search result histogram displays a chart using the time frame of your search, graphing the number of search result counts over time. It may help you to visualize how the number of request to your site change over time, or to see how many downloads a file has over time.

Changing the graph resolution, you can decide how much time each bar of the graph represents.

13.6.3 Statistical values

You can add to your dashboard any statistical value calculated for a field. This may help you to see the mean time response for your application, or how many unique servers are handling requests to your application, by using the cardinality value of that field. Please refer to *Field statistics* for more information on the available statistical functions and how to display them in your searches.

As with search result counts, you can also add trend information to statistical value widgets created with a relative time frame.

13.6.4 Field value charts

To draw an statistical value over time, you can use a field value chart. They could help you to see the evolution of the number of unique users visiting your site in the last week. In the *Field graphs* section we explain how to create these charts and ways you can customize them.

13.6.5 Stacked charts

Stacked charts group several field value charts under the same axes. They let you compare different values in a compact way, like the number of visits to two different websites. As explained in *Field graphs*, stacked charts are basically field value charts represented in the same axes.

13.6.6 Quick values results

In order to show a list of values a certain field contains and their distribution, you can use a quick value widget. This may help you to see the percentage of failed requests in your application, or which parts of your application experience more problems. Please refer to *Quick values* to see how to request this information in your search result page.

The quick values information can be represented as a pie chart and/or as a table, so you can choose what is the best fit for your needs.

13.7 Modifying dashboards

You need to *unlock* dashboards to make any changes to them. Hit the "Unlock/Edit" button in the top right corner of a dashboard to unlock it. You should now see different icons at the bottom of each widget, that allow you to perform more actions.

13.7.1 Unlocked dashboard widgets explained

Unlocked dashboard widgets have two buttons that should be pretty self-explanatory.

- Delete widget
- Edit widget configuration
- Change widget size (when you hover over the widget)
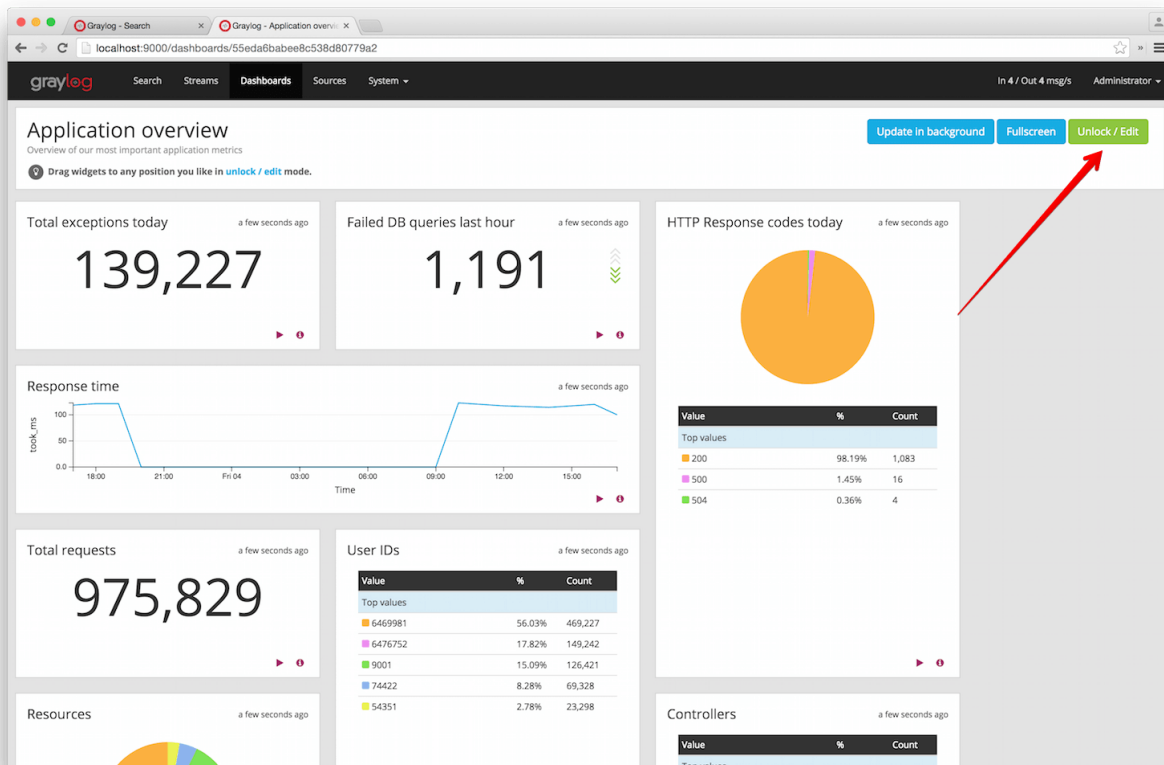
13.7.2 Widget cache times

Widget values are cached in `graylog-server` by default. **This means that the cost of value computation does not grow with every new device or even browser tab displaying a dashboard.** Some widgets might need to show real-time information (set cache time to 1 second) and some widgets might be updated way less often (like *Top SSH users this month*, cache time 10 minutes) to save expensive computation resources.

13.7.3 Repositioning widgets

Just grab a widget with your mouse in unlocked dashboard mode and move it around. Other widgets should adopt and re-position intelligently to make place for the widget you are moving. The positions are automatically saved when dropping a widget.

13.7.4 Resizing widgets

When hovering over a widget, you will see that a gray arrow appears in its bottom-right corner. You can use that icon to resize widgets. Their contents will adapt to the new size automatically!
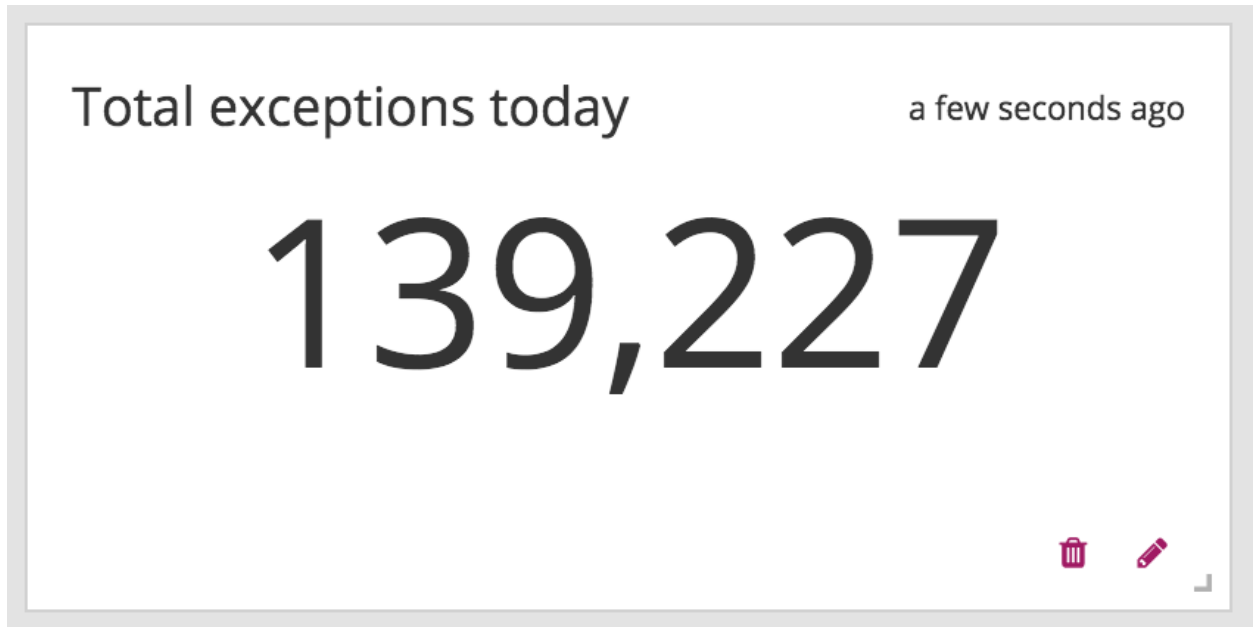
13.8 Dashboard permissions

Graylog users in the *Admin* role are always allowed to view and edit all dashboards. Users in the *Reader* role are by default not allowed to view or edit **any** dashboard.
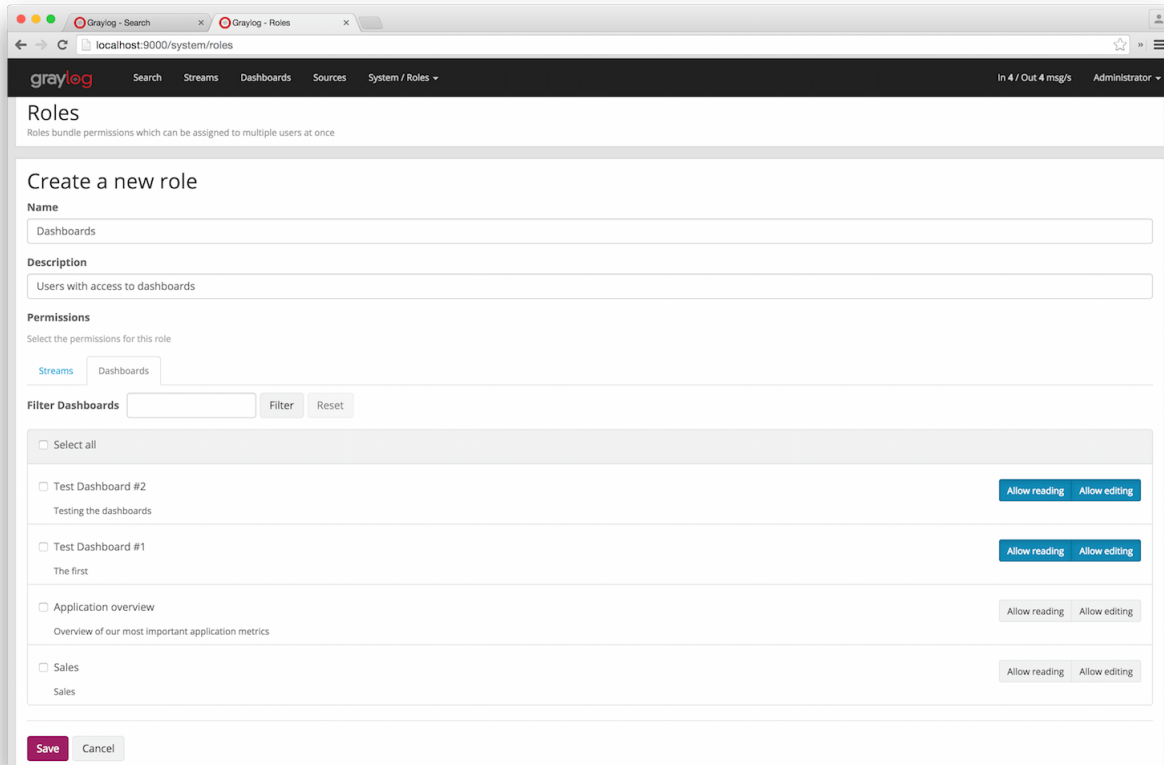
Navigate to *System -> Roles* and create a new role that grant the permissions you wish. You can then assign that new role to any users you wish to give dashboard permissions in the *System -> Users* page.

You can read more about *user permissions and roles*.

13.8.1 That's it!

Congratulations, you have just gone through the basic principles of Graylog dashboards. Now think about which dashboards to create. We suggest:

- Create dashboards for yourself and your team members
- Create dashboards to share with your manager
- Create dashboards to share with the CIO of your company

Think about which information you need access to frequently. What information could your manager or CIO be interested in? Maybe they want to see how the number of exceptions went down or how your team utilized existing hardware better. The sales team could be interested to see signup rates in realtime and the marketing team will love you for providing insights into low level KPIs that is just a click away.

FOURTEEN

EXTRACTORS

14.1 The problem explained

Syslog (RFC3164, RFC5424) is the de facto standard logging protocol since the 1980s and was originally developed as part of the sendmail project. It comes with some annoying shortcomings that we tried to improve in *GELF* for application logging.

Because syslog has a clear specification in its RFCs it should be possible to parse it relatively easy. Unfortunately there are a lot of devices (especially routers and firewalls) out there that send logs looking like syslog but actually breaking several rules stated in the RFCs. We tried to write a parser that reads all of them as good as possible and failed. Such a loosely defined text message usually breaks the compatibility in the first date field already. Some devices leave out hostnames completely, some use localized time zone names (e. g. "MESZ" instead of "CEST"), and some just omit the current year in the timestamp field.

Then there are devices out there that at least do not claim to send syslog when they don't but have another completely separate log format that needs to be parsed specifically.

We decided not to write custom message inputs and parsers for all those thousands of devices, formats, firmwares and configuration parameters out there but came up with the concept of *Extractors* introduced in the *v0.20.0* series of Graylog.

14.2 Graylog extractors explained

The extractors allow you to instruct Graylog nodes about how to extract data from any text in the received message (no matter from which format or if an already extracted field) to message fields. You may already know why structuring data into fields is important if you are using Graylog: There are a lot of analysis possibilities with full text searches but the real power of log analytics unveils when you can run queries like `http_response_code:>=500 AND user_id:9001` to get all internal server errors that were triggered by a specific user.

Wouldn't it be nice to be able to search for all blocked packages of a given source IP or to get a quickterms analysis of recently failed SSH login usernames? Hard to do when all you have is just a single long text message.

> **Attention:** Graylog extractors only work on text fields but won't be executed for numeric fields or anything other than a string.

Creating extractors is possible via either Graylog REST API calls or from the web interface using a wizard. Select a message input on the *System -> Inputs* page and hit *Manage extractors* in the actions menu. The wizard allows you to load a message to test your extractor configuration against. You can extract data using for example regular expressions, Grok patterns, substrings, or even by splitting the message into tokens by separator characters. The wizard looks like this and should be pretty intuitive:

You can also choose to apply so called *converters* on the extracted value to for example convert a string consisting of numbers to an integer or double value (important for range searches later), anonymize IP addresses, lower-/uppercase a string, build a hash value, and much more.

14.3 Import extractors

The recommended way of importing extractors in Graylog is using *Content packs*. The Graylog Marketplace provides access to many content packs that you can easily download and import into your Graylog setup.

You can still import extractors from JSON if you want to. Just copy the JSON extractor export into the import dialog of a message input of the fitting type (every extractor set entry in the directory tells you what type of input to spawn, e. g. syslog, GELF, or Raw/plaintext) and you are good to go. The next messages coming in should already include the extracted fields with possibly converted values.

A message sent by Heroku and received by Graylog with the imported *Heroku* extractor set on a plaintext TCP input looks like this: (look at the extracted fields in the message detail view)

✉ a700ec50-dfbf-11e3-b51a-12313c013090 ↰

Received by ☁ *Heroku test 2* on ⅄ *3c1749a2 / 54.247.*

Timestamp: *2014-05-20 03:39:57.284*

Index: *graylog2_1*

[Actions ▾]

bytes

   31094

client_ip

   164.177.

connect_time_ms

   0

drain_id

   d.158ca493-4116-451a-9e6e-8158d609eb53

dyno

   web.1

facility

   local3

heroku_component

   router

heroku_source_type

   heroku

http_method

   get

http_status

   200

level

   Info [6]

message

   at=info method=GET path=/ host=www.graylog2.org request_id=f3ffb87e-

14.4 Using regular expressions to extract data

Extractors support matching field values using regular expressions. Graylog uses the Java Pattern class to evaluate regular expressions.

For the individual elements of regular expression syntax, please refer to Oracle's documentation, however the syntax largely follows the familiar regular expression languages in widespread use today and will be familiar to most.

However, one key question that is often raised is matching a string in case insensitive manner. Java regular expressions are case sensitive by default. Certain flags, such as the one to ignore case sensitivity can either be set in the code, or as an inline flag in the regular expression.

For example, to create an extractor that matches the browser name in the following user agent string:

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko)␣
↪Chrome/32.0.1700.107 Safari/537.36
```

the regular expression `(applewebkit)` will not match because it is case sensitive. In order to match the expression using any combination of upper- and lowercase characters use the `(?i)` flag as such:

```
(?i)(applewebkit)
```

Most of the other flags supported by Java are rarely used in the context of matching stream rules or extractors, but if you need them their use is documented on the same Javadoc page by Oracle. One common reason to use regular expression flags in your regular expression is to make use of what is called non-capturing groups. Those are parentheses which only group alternatives, but do not make Graylog extract the data they match and are indicated by `(?:)`.

14.5 Using Grok patterns to extract data

Graylog also supports the extracting data using the popular Grok language to allow you to make use of your existing patterns.

Grok is a set of regular expressions that can be combined to more complex patterns, allowing to name different parts of the matched groups.

By using Grok patterns, you can extract multiple fields from a message field in a single extractor, which often simplifies specifying extractors.

Simple regular expressions are often sufficient to extract a single word or number from a log line, but if you know the entire structure of a line beforehand, for example for an access log, or the format of a firewall log, using Grok is advantageous.

For example a firewall log line could contain:

```
len=50824 src=172.17.22.108 sport=829 dst=192.168.70.66 dport=513
```

We can now create the following patterns on the `System/Grok Patterns` page in the web interface:

```
BASE10NUM (?<![0-9.+-])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+)))
NUMBER (?:%{BASE10NUM})
IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-
↪f]{1,4}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}
↪)|:))|(([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d\
↪d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}
↪(((:[0-9A-Fa-f]{1,4}){1,3})|((:[0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\
↪d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-
↪Fa-f]{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,2}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.
↪(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){2}(((:[0-9A-Fa-f]
↪{1,4}){1,5})|((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-
↪5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-Fa-f]{1,4})
↪{1,6})|((:[0-9A-Fa-f]{1,4}){0,4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-
↪4]\d|1\d\d|[1-9]?\d)){3}))|:))|(:(((:[0-9A-Fa-f]{1,4}){1,7})|((:[0-9A-Fa-f]{1,4}){0,
↪5}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}
↪))|:)))(%.+)?
```

```
IPV4 (?<![0-9])(?:(?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-
→9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-
→4][0-9]|[0-1]?[0-9]{1,2}))(?![0-9])
IP (?:%{IPV6}|%{IPV4})
DATA .*?
```

Then, in the extractor configuration, we can use these patterns to extract the relevant fields from the line:

```
len=%{NUMBER:length} src=%{IP:srcip} sport=%{NUMBER:srcport} dst=%{IP:dstip} dport=%
→{NUMBER:dstport}
```

This will add the relevant extracted fields to our log message, allowing Graylog to search on those individual fields, which can lead to more effective search queries by allowing to specifically look for packets that came from a specific source IP instead of also matching destination IPs if one would only search for the IP across all fields.

If the Grok pattern creates many fields, which can happen if you make use of heavily nested patterns, you can tell Graylog to skip certain fields (and the output of their subpatterns) by naming a field with the special keyword UNWANTED.

Let's say you want to parse a line like:

```
type:44 bytes:34 errors:122
```

but you are only interested in the second number bytes. You could use a pattern like:

```
type:%{BASE10NUM:type} bytes:%{BASE10NUM:bytes} errors:%{BASE10NUM:errors}
```

However, this would create three fields named type, bytes, and errors. Even not naming the first and last patterns would still create a field names BASE10NUM. In order to ignore fields, but still require matching them use UNWANTED:

```
type:%{BASE10NUM:UNWANTED} bytes:%{BASE10NUM:bytes} errors:%{BASE10NUM:UNWANTED}
```

This now creates only a single field called bytes while making sure the entire pattern must match.

If you already know the data type of the extracted fields, you can make use of the type conversion feature built into the Graylog Grok library. Going back to the earlier example:

```
len=50824 src=172.17.22.108 sport=829 dst=192.168.70.66 dport=513
```

We know that the content of the field len is an integer and would like to make sure it is stored with that data type, so we can later create field graphs with it or access the field's statistical values, like average etc.

Grok directly supports converting field values by adding ;datatype at the end of the pattern, like:

```
len=%{NUMBER:length;int} src=%{IP:srcip} sport=%{NUMBER:srcport} dst=%{IP:dstip}␣
→dport=%{NUMBER:dstport}
```

The currently supported data types, and their corresponding ranges and values, are:

Type	Range	Example
byte	-128 ... 127	`%{NUMBER:fieldname;byte}`
short	-32768 ... 32767	`%{NUMBER:fieldname;short}`
int	-2^31 ... 2^31 -1	`%{NUMBER:fieldname;int}`
long	-2^63 ... 2^63 -1	`%{NUMBER:fieldname;long}`
float	32-bit IEEE 754	`%{NUMBER:fieldname;float}`
double	64-bit IEEE 754	`%{NUMBER:fieldname;double}`
boolean	*true*, *false*	`%{DATA:fieldname;boolean}`
string	Any UTF-8 string	`%{DATA:fieldname;string}`
date	See SimpleDateFormat	`%{DATA:timestamp;date;dd/MMM/yyyy:HH:mm:ss Z}`
datetime	Alias for *date*	

There are many resources are the web with useful patterns, and one very helpful tool is the Grok Debugger, which allows you to test your patterns while you develop them.

Graylog uses Java Grok to parse and run Grok patterns.

14.6 Using the JSON extractor

Since version 1.2, Graylog also supports extracting data from messages sent in JSON format.

Using the JSON extractor is easy: once a Graylog input receives messages in JSON format, you can create an extractor by going to *System -> Inputs* and clicking on the *Manage extractors* button for that input. Next, you need to load a message to extract data from, and select the field containing the JSON document. The following page let you add some extra information to tell Graylog how it should extract the information. Let's illustrate how a message would be extracted with an example message:

```
{"level": "ERROR", "details": {"message": "This is an example error message",
→"controller": "IndexController", "tags": ["one", "two", "three"]}}
```

Using the default settings, that message would be extracted into these fields:

details_tags one, two, three

level ERROR

details_controller IndexController

details_message This is an example error message

In the create extractor page, you can also customize how to separate list of elements, keys, and key/values. It is also possible to flatten JSON structures or expand them into multiple fields, as shown in the example above.

14.7 Automatically extract all key=value pairs

Sometimes you will receive messages like this:

```
This is a test message with some key/value pairs. key1=value1 some_other_key=foo
```

You might want to extract all `key=value` pairs into Graylog message fields without having to specify all possible key names or even their order. This is how you can easily do this:

Create a new extractor of type "Copy Input" and select to read from the field `message`. (Or any other string field that contains `key=value` pairs.) Configure the extractor to store the (copied) field value to the same field. In this case

`message`. The trick is to add the "Key=Value pairs to fields" converter as last step. Because we use the "Copy Input" extractor, the converter will run over the complete field you selected and convert all `key=value` pairs it can find.

This is a screenshot of the complete extractor configuration:
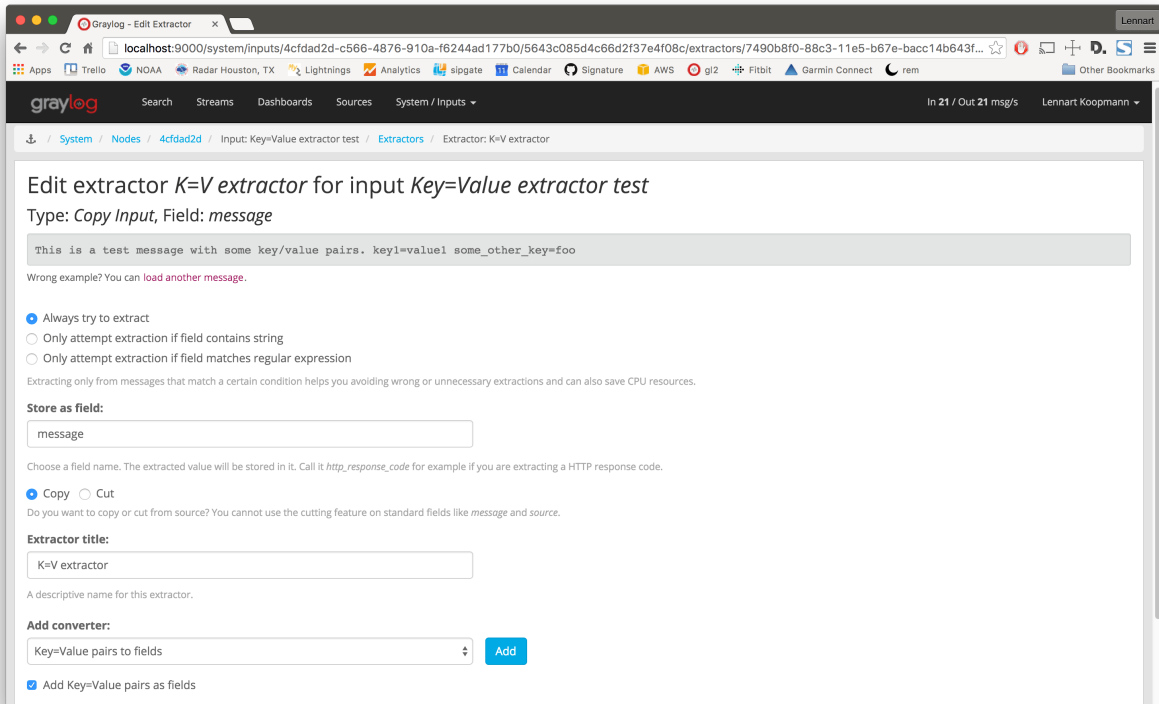
. . . and this is the resulting message:

14.8 Normalization

Many log formats are similar to each other, but not quite the same. In particular they often only differ in the names attached to pieces of information.

For example, consider different hardware firewall vendors, whose models log the destination IP in different fields of the message, some use `dstip`, some `dst` and yet others use `destination-address`:

```
2004-10-13 10:37:17 PDT Packet Length=50824, Source address=172.17.22.108, Source␣
→port=829, Destination address=192.168.70.66, Destination port=513
2004-10-13 10:37:17 PDT len=50824 src=172.17.22.108 sport=829 dst=192.168.70.66␣
→dport=513
2004-10-13 10:37:17 PDT length="50824" srcip="172.17.22.108" srcport="829" dstip="192.
→168.70.66" dstport="513"
```

You can use one or more non-capturing groups to specify the alternatives of the field names, but still be able to extract the a parentheses group in the regular expression. Remember that Graylog will extract data from the first matched group of the regular expression. An example of a regular expression matching the destination IP field of all those log messages from above is:

```
(?:dst|dstip|[dD]estination\saddress)="?(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"?
```

This will only extract the IP address without caring about which of the three naming schemes was used in the original log message. This way you don't have to set up three different extractors.

14.8.1 The standard date converter

Date parser converters for extractors allow you to convert extracted data into timestamps - Usually used to set the timestamp of a message based on some date it contains. Let's assume we have this message from a network device:

```
<131>: foo-bar-dc3-org-de01: Mar 12 00:45:38: %LINK-3-UPDOWN: Interface␣
→GigabitEthernet0/31, changed state to down
```

Extracting most of the data is not a problem and can be done easily. Using the date in the message (*Mar 12 00:45:38*) as Graylog message timestamp however needs to be done with a date parser converter.

Use a copy input extractor rule to select the timestamp and apply the *Date* converter with a format string:

```
MMM dd HH:mm:ss
```

(format string table at the end of this page)

Store as field:

timestamp

Choose a field name. The extracted value will be stored in it. Call it *http_response_code* for example if you are extracting a HTTP response code.

⊙ Copy ○ Cut
Do you want to copy or cut from source?

Extractor title:

Timestamp

A descriptive name of this extractor.

Add converter:

Date ⇕ Add

☑ Convert to date type
Format string: 💡

MMM dd HH:mm:ss

Please note that you cannot use the cutting feature on standard fields like *message* and *source*.

Create extractor

✉ 4765e370-aa42-11e3-a7dd-4c8d79f2b596 ↰

Received by ☁ *Cisco System Messages* on ⑂ fb66b27e / 10.226.163.44

Timestamp: *2014-03-12 00:45:38.000*

Index: *graylog2_356*

Actions ▾

facility

   local0

level

   Error [3]

local_facility

   link

local_level

   3

message

   Interface GigabitEthernet0/31, changed state to down

source

   foo-bar-dc3-org-de01

type

   updown

Standard date converter format string table

Symbol	Meaning	Presentation	Examples
G	era	text	AD
C	century of era (>=0)	number	20
Y	year of era (>=0)	year	1996
x	weekyear	year	1996
w	week of weekyear	number	27
e	day of week	number	2
E	day of week	text	Tuesday; Tue
y	year	year	1996
D	day of year	number	189
M	month of year	month	July; Jul; 07
d	day of month	number	10
a	halfday of day	text	PM
K	hour of halfday (0~11)	number	0
h	clockhour of halfday (1~12)	number	12
H	hour of day (0~23)	number	0
k	clockhour of day (1~24)	number	24
m	minute of hour	number	30
s	second of minute	number	55
S	fraction of second	millis	978
z	time zone	text	Pacific Standard Time; PST
Z	time zone offset/id	zone	-0800; -08:00; America/Los_Angeles
'	escape for text	delimiter	
''	single quote	literal	'

14.8.2 The flexible date converter

Now imagine you had one of those devices that send messages that are not so easy to parse because they do not follow a strict timestamp format. Some network devices for example like to send days of the month without adding a padding 0 for the first 9 days. You'll have dates like `Mar 9` and `Mar 10` and end up having problems defining a parser string for that. Or maybe you have something else that is really exotic like just *last wednesday* as timestamp. The flexible date converter is accepting any text data and tries to build a date from that as good as it can.

Examples:

- **Mar 12**, converted at 12:27:00 UTC in the year 2014: 2014-03-12T12:27:00.000

- **2014-3-12 12:27**: 2014-03-12T12:27:00.000

- **Mar 12 2pm**: 2014-03-12T14:00:00.000

Note that the flexible date converter is using UTC as time zone by default unless you have time zone information in the parsed text or have configured another time zone when adding the flexible date converter to an extractor (see this comprehensive list of time zones available for the flexible date converter).

PROCESSING PIPELINES

Graylog's new processing pipelines plugin allows greater flexibility in routing, blacklisting, modifying, and enriching messages as they flow through Graylog.

Pipelines and rules are not configuration for pre-built code, as extractors and stream rules are, but are instead represented as code, much like Drools rules. This gives them great flexibility and extensibility, and enables live changes to Graylog's message processing behavior.

The language used for pipeline rules is very simple and can be extended by functions, which are fully pluggable.

The following pages introduce the concepts of pipelines, rules, stream connections, and the built-in functions.

15.1 Pipelines

15.1.1 Overview

Pipelines are the central concept tying together the processing steps applied to your messages.

Pipelines contain rules and can be connected to one or more streams, enabling fine-grained control of the processing applied to messages.

Processing rules are simply conditions followed by a list of actions, and do not have control flow by themselves. Therefore, pipelines have one additional concept: stages.

Think of stages as groups of conditions and actions which need to run in order. All stages with the same priority run at the same time across all connected pipelines. Stages provide the necessary control flow to decide whether or not to run the remaining stages in a pipeline.

15.1.2 Pipeline structure

Internally pipelines are represented as code. Let's have a look at a simple example and understand what each part does:

```
pipeline "My new pipeline"
stage 1 match all
  rule "has firewall fields";
  rule "from firewall subnet";
stage 2 match either
  rule "geocode IPs";
  rule "anonymize source IPs";
end
```

This code snippet declares a new pipeline named `My new pipeline`, which has two stages.

Stages are run in the order of their given *priority*, and aren't otherwise named. Stage priorities can be any integer, positive or negative, you prefer. In our example the first stage has a priority of 1 and the second stage a priority of 2, however -99 and 42 could be used instead. Ordering based upon stage priority gives you the ability to run certain rules before or after others, which might exist in other connected pipelines, without modifying those other connected pipelines. This is particularly handy when dealing with changing data formats.

For example, if there was a second pipeline declared with a stage assigned priority 0, that stage's rules would run before either of the ones from the example (priorities 1 and 2, respectively). Note that the order in which stages are declared is irrelevant, since they are sorted according to their priority.

Stages then list the *rule references* they want to be executed, as well as whether *any* or *all* of the rules' conditions need to be satisfied to continue running the pipeline.

In our example, imagine rule *"has firewall fields"* checks for the presence of message fields `src_ip` and `dst_ip`, but does not have any actions to run. For a message without both fields the rule's condition would evaluate to `false` and the pipeline would abort after stage 1, as the stage requires *all* rules be satisfied (`match all`). With the pipeline aborted, stage 2 would not run.

`match either` acts as an `OR` operator, only requiring a single rule's condition evaluate to `true` in order to continue pipeline processing. Note that actions are still ran for all matching rules in the stage, even if it is the final stage in the pipeline.

Rules are referenced by their names, and can therefore be shared among many different pipelines. The intention is to enable creation of reusable building blocks, making it easier to process the data specific to your organization or use case.

Read more about *Rules* in the next section.

15.2 Rules

15.2.1 Overview

Rules are the cornerstone of processing pipelines. They contain the logic about how to change, enrich, route, and drop messages.

To avoid the complexities of a complete programming language, Graylog supports a small rule language to express processing logic. The rule language is intentionally limited to allow for easier understanding, faster learning, and better runtime optimization.

The real work of rules is done in *functions*, which are completely pluggable. Graylog already ships with a great number of built-in functions, providing data conversion, string manipulation, data retrieval using *lookup tables*, JSON parsing, and much more.

We expect that special purpose functions will be written and shared by the community, enabling faster innovation and problem solving than previously possible.

15.2.2 Rule Structure

Building upon the previous example in the *Pipelines* section, let's look at examples of some of the rules we've referenced:

```
rule "has firewall fields"
when
    has_field("src_ip") && has_field("dst_ip")
then
end
```

```
rule "from firewall subnet"
when
    cidr_match("10.10.10.0/24", to_ip($message.gl2_remote_ip))
then
end
```

Firstly, apart from naming the rule structure follows a simple *when, then* pattern. In the *when* clause we specify a boolean expression which is evaluated in the context of the current message in the pipeline. These are the conditions used by the pipeline processor to determine whether to run a rule, and collectively (when evaluating the containing stage's `match all` or `match any` requirement) whether to continue in a pipeline.

Note that the *has firewall fields* rule uses the built-in function `has_field` to check whether the message has the `src_ip` and `dst_ip` fields, as we want to use them in a later stage of the pipeline. This rule has no actions to run in its *then* clause, since we only want to use it to determine whether subsequent stages should run.

The second rule, *from firewall subnet*, uses the built-in function *cidr_match*, which takes a CIDR pattern and an IP address. In this case we reference a field from the currently-processed message using the message reference syntax `$message`.

Graylog always sets the `gl2_remote_ip` field on messages, so we don't need to check whether that field exists. If we wanted to use a field that might not exist on all messages we'd first use the `has_field` function to ensure its presence.

Note the call to `to_ip` around the `gl2_remote_ip` field reference. This is necessary since the field is stored as a *string* internally, and `cidr_match` requires an IP address object for its `ip` parameter.

Requiring an explicit conversion to an IP address object demonstrates an important feature of Graylog's rule language: enforcement of type safety to ensure that you end up with the data in the correct format. All too often everything is treated as a string, which wastes enormous amounts of cycles on data conversion and prevents proper analysis of the data.

We again have no actions to run, since we're just using the rule to manage the pipeline's flow, so the *then* block is empty.

You might be wondering why we didn't just combine the *has firewall fields* and *from firewall subnet* rules, since they seem to be serving the same purpose. While we could absolutely do so, recall that rules are intended to be reusable building blocks. Imagine you have a another pipeline for a different firewall subnet. Rather than duplicating the logic to check for `src_ip` and `dst_ip`, and updating each rule if anything ever changes (e.g. additional fields), you can simply add the *has firewall fields* rule to your new stage. With this approach you only need to update a single rule, with the change immediatedly taking effect for all pipelines referencing it. Nice!

15.2.3 Data Types

As we have seen in the previous section, we need to make sure to use the proper data types when calling functions.

Graylog's rule language parser rejects invalid use of types, making it safe to write rules.

The six built-in types in Graylog are `string` (a UTF-8 string), `double` (corresponds to Java's `Double`), `long` (Java's `Long`), `boolean` (Boolean), `void` (indicating a function has no return value to prevent it being used in a condition), and `ip` (a subset of `InetAddress`), but plugins are free to add additional types as they see fit. The rule processor takes care of ensuring that values and functions agree on the types being used.

By convention, functions that convert types start with the prefix `to_`. Please refer to the *Functions* index for a list.

15.2.4 Conditions

In Graylog's rules the **when** clause is a boolean expression, which is evaluated against the processed message.

Expressions support the common boolean operators `AND` (or `&&`), `OR` (`||`), `NOT` (`!`), and comparison operators (<, <=, >, >=, ==, !=).

Any function that returns a value can be called in the **when** clause, but it must eventually evaluate to a boolean. For example: we were able to use `to_ip` in the *from firewall subnet* since it was being passed to `cidr_match`, which returns a boolean, but could not use `route_to_stream` since it doesn't return a value.

The condition must not be empty, but can simply consist of the boolean literal `true`. This is useful when you always want to execute a rule's actions.

If a condition calls a function which is not present (perhaps due to a typo or missing plugin) the call evaluates to `false`.

Note: Comparing two fields can be done when you use the same data type, e.g. `to_string($message.src_ip) == to_string($message.dst_ip)` will compare the two strings and will become `true` on match. Comparing different data types evaluates to `false`.

15.2.5 Actions

A rule's **then** clause contains a list of actions which are evaluated in the order they appear.

There are two different types of actions:

- Function calls
- Variable assignments

Function calls look exactly like they do in conditions. All functions, including those which do not return a value, may be used in the **then** clause.

Variable assignments have the following form:

```
let name = value;
```

Variables are useful to avoid recomputing expensive parsing of data, holding on to temporary values, or making rules more readable.

Variables need to be defined before they can be used. Their fields (if any) can be accessed using the `name.field` notation in any place where a value of the field's type is required.

The list of actions can be empty, in which case the rule is essentially a pluggable condition to help manage a pipeline's processing flow.

15.3 Stream connections

15.3.1 Overview

Pipelines by themselves do not process any messages. For a pipeline to actually do any work it must first be connected to one or more streams, which enables fine-grained control of the messages processed by that pipeline.

Note that the built-in function `route_to_stream` causes a message to be routed to a particular stream. After the routing occurs, the pipeline engine will look up and start evaluating any pipelines connected to that stream.

Although pipelines can trigger other pipelines via message routing, incoming messages must be processed by an initial set of pipelines connected to one or more streams.

15.3.2 The All messages stream

All messages received by Graylog are initially routed into the **All messages** stream. You can use this stream as the entry point to pipeline processing, allowing incoming messages to be routed to more streams and being processed subsequently.

However, if you prefer to use the original stream matching functionality (i.e. stream rules), you can configure the *Pipeline Processor* to run after the *Message Filter Chain* (in the *Message Processors Configuration* section of the *System -> Configurations* page) and connect pipelines to existing streams. This gives you fine-grained control over the extraction, conversion, and enrichment process.

15.3.3 The importance of message processor ordering

It's important to note that the order of message processors may have a significant impact on how your messages get processed.

For example: *Message Filter Chain* is responsible for setting static fields and running extractors defined on inputs, as well as evaluation of stream rules. If you create a pipeline that expects the presence of a static field, but the *Pipeline Processor* runs before *Message Filter Chain*, that field will not be available for use in your pipeline.

When designing your streams and pipelines be aware of the message processor order, especially if you have dependencies on earlier message processing.

15.4 Functions

15.4.1 Overview

Functions are the means of interacting with the messages Graylog processes.

Functions are written in Java and are pluggable, allowing Graylog's pipeline processing capabilities to be easily extended.

Conceptually a function receives parameters, the current message context, and (potentially) returns a value. The data types of its return value and parameters determine where it can be used in a rule. Graylog ensures the rules are sound from a data type perspective.

A function's parameters can either be passed as named pairs or position, as long as optional parameters are declared as coming last. The functions' documentation below indicates which parameters are optional by wrapping them in square brackets.

Let's look at a small example to illustrate these properties:

```
rule "function howto"
when
    has_field("transaction_date")
then
    // the following date format assumes there's no time zone in the string
    let new_date = parse_date(to_string($message.transaction_date), "yyyy-MM-dd␣
↪HH:mm:ss");
    set_field("transaction_year", new_date.year);
end
```

In this example, we check if the current message contains the field `transaction_date` and then, after converting it to a string, try to parse it according to the format string `yyyy-MM-dd HH:mm:ss`, so for example the string `2016-03-05 14:45:02` would match. The `parse_date` function returns a `DateTime` object from the Java Joda-Time library, allowing easier access to the date's components.

We then add the transaction's year as a new field, `transaction_year` to the message.

You'll note that we didn't specify a time zone for our date, but Graylog still had to pick one. Graylog never relies on the local time of your server, as that makes it nearly impossible to figure out why date handling came up with its result.

The reason Graylog knows which timezone to use is because `parse_date` actually takes four parameters, rather than the two we've given it in this example. The other two parameters are a `String` called `timezone` (default value: `"UTC"`) and a `String` called `locale` (default value: the default locale of the system running Graylog) which both are optional.

Let's assume we have another message field called `transaction_timezone`, which is sent by the application and contains the time zone ID the transaction was done in (hopefully no application in the world sends its data like this, though):

```
rule "function howto"
when
    has_field("transaction_date") && has_field("transaction_timezone")
then
    // the following date format assumes there's no time zone in the string
    let new_date = parse_date(
                        to_string($message.transaction_date),
                        "yyyy-MM-dd HH:mm:ss",
                        to_string($message.transaction_timezone)
                );
    set_field("transaction_year", new_date.year);
end
```

Now we're passing the `parse_date` function its `timezone` parameter the string value of the message's `transaction_timezone` field.

In this case we only have a single optional parameter, which makes it easy to simply omit it from the end of the function call. However, if there are multiple optional parameters, or if there are so many parameters that it gets difficult to keep track of which positions correspond to which parameters, you can also use the named parameter variant of function calls. In this mode the order of the parameters does not matter, but all required ones still need to be there.

In our case the alternative version of calling `parse_date` would look like this:

```
rule "function howto"
when
    has_field("transaction_date") && has_field("transaction_timezone")
then
    // the following date format assumes there's no time zone in the string
    let new_date = parse_date(
                        value: to_string($message.transaction_date),
                        pattern: "yyyy-MM-dd HH:mm:ss",
                        timezone: to_string($message.transaction_timezone)
                );
    set_field("transaction_year", new_date.year);
end
```

All parameters in Graylog's processing functions, listed below, are named.

15.4.2 Function Index

The following list describes the built-in functions that ship with Graylog. Additional third party functions are available via plugins in the marketplace.

Table 1: Built-in Functions

Name	Description
debug	Print the passed value as string in the Graylog log.
to_bool	Converts the single parameter to a boolean value using its string value.
to_double	Converts the first parameter to a double floating point value.
to_long	Converts the first parameter to a long integer value.
to_string	Converts the first parameter to its string representation.
to_url	Converts a value to a valid URL using its string representation.
to_map	Converts a value to a map.
is_null	Checks whether a value is `null`.
is_not_null	Checks whether a value is not `null`.
is_boolean	Checks whether a value is a boolean value (`true` or `false`).
is_number	Checks whether a value is a numeric value (of type `long` or `double`).
is_double	Checks whether a value is a floating point value (of type `double`).
is_long	Checks whether a value is an integer value (of type `long`).
is_string	Checks whether a value is a string.
is_collection	Checks whether a value is an iterable collection.
is_list	Checks whether a value is an iterable list.
is_map	Checks whether a value is a map.
is_date	Checks whether a value is a date (of type `DateTime`).
is_period	Checks whether a value is a time period (of type `Period`).
is_ip	Checks whether a value is an IP address (IPv4 or IPv6).
is_json	Checks whether a value is a parsed JSON tree.
is_url	Checks whether a value is a parsed URL.
abbreviate	Abbreviates a String using ellipses.
capitalize	Capitalizes a String changing the first letter to title case.
uncapitalize	Uncapitalizes a String changing the first letter to lower case.
uppercase	Converts a String to upper case.
lowercase	Converts a String to lower case.
swapcase	Swaps the case of a String.
contains	Checks if a string contains another string.

Continued on next page

Table 1 – continued from previous page

Name	Description
replace	Replaces the first "max" or all occurrences of a string within another string
starts_with	Checks if a string starts with a given prefix.
ends_with	Checks if a string ends with a given suffix.
substring	Returns a substring of `value` with the given start and end offsets.
concat	Concatenates two strings.
split	Split a string around matches of this pattern (Java syntax).
regex	Match a regular expression against a string, with matcher groups.
regex_replace	Match a regular expression against a string and replace with string.
grok	Applies a Grok pattern to a string.
key_value	Extracts key/value pairs from a string.
crc32	Returns the hex encoded CRC32 digest of the given string.
crc32c	Returns the hex encoded CRC32C (RFC 3720, Section 12.1) digest of the given string.
md5	Returns the hex encoded MD5 digest of the given string.
murmur3_32	Returns the hex encoded MurmurHash3 (32-bit) digest of the given string.
murmur3_128	Returns the hex encoded MurmurHash3 (128-bit) digest of the given string.
sha1	Returns the hex encoded SHA1 digest of the given string.
sha256	Returns the hex encoded SHA256 digest of the given string.
sha512	Returns the hex encoded SHA512 digest of the given string.
parse_json	Parse a string into a JSON tree.
select_jsonpath	Selects one or more named JSON Path expressions from a JSON tree.
to_ip	Converts the given string to an IP object.
cidr_match	Checks whether the given IP matches a CIDR pattern.
from_input	Checks whether the current message was received by the given input.
route_to_stream	Assigns the current message to the specified stream.
remove_from_stream	Removes the current message from the specified stream.
create_message	**Currently incomplete** Creates a new message which will be evaluated by the entire processing pipeline.
clone_message	Clones a message.
drop_message	This currently processed message will be removed from the processing pipeline after the rule finishes.
has_field	Checks whether the currently processed message contains the named field.
remove_field	Removes the named field from the currently processed message.
set_field	Sets the name field to the given value in the currently processed message.
set_fields	Sets multiple fields to the given values in the currently processed message.
rename_field	Rename a message field.
syslog_facility	Converts a syslog facility number to its string representation.
syslog_level	Converts a syslog level number to its string representation.
expand_syslog_priority	Converts a syslog priority number to its level and facility.
expand_syslog_priority_as_string	Converts a syslog priority number to its level and facility string representations.
now	Returns the current date and time.
parse_date	Parses a date and time from the given string, according to a strict pattern.
flex_parse_date	Attempts to parse a date and time using the Natty date parser.
parse_unix_milliseconds	Attempts to parse a UNIX millisecond timestamp (milliseconds since 1970-01-01T00:00:00.000Z).
format_date	Formats a date and time according to a given formatter pattern.
to_date	Converts a type to a date.
years	Create a period with a specified number of years.
months	Create a period with a specified number of months.

Continued on next page

Table 1 – continued from previous page

Name	Description
weeks	Create a period with a specified number of weeks.
days	Create a period with a specified number of days.
hours	Create a period with a specified number of hours.
minutes	Create a period with a specified number of minutes.
seconds	Create a period with a specified number of seconds.
millis	Create a period with a specified number of millis.
period	Parses an ISO 8601 period from the specified string.
lookup	Looks up a multi value in the named lookup table.
lookup_value	Looks up a single value in the named lookup table.

debug

```
debug(value:  any)
```

Print any passed value as string in the Graylog log.

Note: The debug message will only appear in the log of the Graylog node that was processing the message you are trying to debug.

Example:

```
// Print: "INFO : org.graylog.plugins.pipelineprocessor.ast.functions.Function -␣
↪PIPELINE DEBUG: Dropped message from <source>"
let debug_message = concat("Dropped message from ", to_string($message.source));
debug(debug_message);
```

to_bool

```
to_bool(value:  any)
```

Converts the single parameter to a boolean value using its string value.

to_double

```
to_double(value:  any, [default:  double])
```

Converts the first parameter to a double floating point value.

to_long

```
to_long(value:  any, [default:  long])
```

Converts the first parameter to a long integer value.

to_string

```
to_string(value:  any, [default:  string])
```

Converts the first parameter to its string representation.

to_url

```
to_url(url:  any, [default:  string])
```

Converts the given `url` to a valid URL.

to_map

```
to_map(value:  any)
```

Converts the given map-like value to a valid map.

The `to_map()` function currently only supports converting a parsed JSON tree into a map so that it can be used together with *set_fields*.

Example:

```
let json = parse_json(to_string($message.json_payload));
let map = to_map(json);
set_fields(map);
```

See also:

- *set_fields*

- *parse_json*

is_null

```
is_null(value:  any)
```

Checks if the given value is `null`.

Example:

```
// Check if the `src_addr` field is null (empty).
// If null, boolean true is returned. If not null, boolean false is returned.
is_null(src_addr)
```

is_not_null

```
is_not_null(value:  any)
```

Checks if the given value is not `null`.

Example:

```
// Check if the `src_addr` field is not null.
// If not null, boolean true is returned. If null, boolean false is returned.
is_not_null(src_addr)
```

is_boolean

```
is_boolean(value:  any)
```

Checks whether the given value is a boolean value (`true` or `false`).

is_number

```
is_number(value:  any)
```

Checks whether the given value is a numeric value (of type `long` or `double`).

See also:

- *is_double*
- *to_double*
- *is_long*
- *to_long*

is_double

```
is_double(value:  any)
```

Checks whether the given value is a floating point value (of type `double`).

See also:

- *to_double*

is_long

```
is_long(value:  any)
```

Checks whether the given value is an integer value (of type `long`).

See also:

- *to_long*

is_string

```
is_string(value:  any)
```

Checks whether the given value is a string.

See also:

- *to_string*

is_collection

`is_collection(value:  any)`

Checks whether the given value is an iterable collection.

is_list

`is_list(value:  any)`

Checks whether the given value is an iterable list.

is_map

`is_map(value:  any)`

Checks whether the given value is a map.

See also:

- *to_map*

is_date

`is_date(value:  any)`

Checks whether the given value is a date (of type `DateTime`).

See also:

- *now*
- *parse_date*
- *flex_parse_date*
- *parse_unix_milliseconds*

is_period

`is_period(value:  any)`

Checks whether the given value is a time period (of type `Period`).

See also:

- *years*
- *months*
- *weeks*
- *days*
- *hours*
- *minutes*
- *seconds*
- *millis*

- *period*

is_ip

```
is_ip(value:   any)
```

Checks whether the given value is an IP address (IPv4 or IPv6).

See also:

- *to_ip*

is_json

```
is_json(value:   any)
```

Checks whether the given value is a parsed JSON tree.

See also:

- *parse_json*

is_url

```
is_url(value:   any)
```

Checks whether the given value is a parsed URL.

See also:

- *to_url*

abbreviate

```
abbreviate(value:   string, width:   long)
```

Abbreviates a String using ellipses, the width defines the maximum length of the resulting string.

capitalize

```
capitalize(value:   string)
```

Capitalizes a String changing the first letter to title case.

uncapitalize

```
uncapitalize(value:   string)
```

Uncapitalizes a String changing the first letter to lower case.

uppercase

```
uppercase(value:  string, [locale:  string])
```

Converts a String to upper case. The locale (IETF BCP 47 language tag) defaults to "en".

lowercase

```
lowercase(value:  string, [locale:  string])
```

Converts a String to lower case. The locale (IETF BCP 47 language tag) defaults to "en".

swapcase

```
swapcase(value:  string)
```

Swaps the case of a String changing upper and title case to lower case, and lower case to upper case.

contains

```
contains(value:  string, search:  string, [ignore_case:  boolean])
```

Checks if `value` contains `search`, optionally ignoring the case of the search pattern.

Example:

```
// Check if the `example.org` is in the `hostname` field. Ignore case.
contains(to_string($message.hostname), "example.org", true)
```

replace

```
replace(value:  string, search:  string, [replacement:  string], [max:  long])
```

Replaces the first `max` or all occurences of a string within another string. `max` is `-1` per defaults which means to replace **all** occurences, `1` only the first one, `2` the first two and so on.

Example:

```
// Correct misspelled message "foo rooft oota"
let new_field = replace(to_string($message.message), "oo", "u");    // "fu ruft uta"
let new_field = replace(to_string($message.message), "oo", "u", 1); // "fu rooft oota"
```

starts_with

```
starts_with(value:  string, prefix:  string, [ignore_case:  boolean])
```

Checks if `value` starts with `prefix`, optionally ignoring the case of the string.

Example:

```
// Returns true
starts_with("Foobar Baz Quux", "foo", true);
// Returns false
starts_with("Foobar Baz Quux", "Quux");
```

ends_with

```
ends_with(value:  string, suffix:  string, [ignore_case:  boolean])
```

Checks if `value` ends with `suffix`, optionally ignoring the case of the string.

Example:

```
// Returns true
starts_with("Foobar Baz Quux", "quux", true);
// Returns false
starts_with("Foobar Baz Quux", "Baz");
```

substring

```
substring(value:  string, start:  long, [end:  long])
```

Returns a substring of `value` starting at the `start` offset (zero based indices), optionally ending at the `end` offset. Both offsets can be negative, indicating positions relative to the end of `value`.

concat

```
concat(first:  string, second:  string)
```

Returns a new string combining the text of `first` and `second`.

Note: The `concat()` function only concatenates two strings. If you want to build a string from more than two sub-strings, you'll have to use `concat()` multiple times, see the example below.

Example:

```
// Build a message like:
// 'TCP connect from 88.99.35.172 to 192.168.1.10 Port 443'
let build_message_0 = concat(to_string($message.protocol), " connect from ");
let build_message_1 = concat(build_message_0, to_string($message.src_ip));
let build_message_2 = concat(build_message_1, " to ");
let build_message_3 = concat(build_message_2, to_string($message.dst_ip));
let build_message_4 = concat(build_message_3, " Port ");
let build_message_5 = concat(build_message_4, to_string($message.dst_port));
set_field("message", build_message_5);
```

split

```
split(pattern:  string, value:  string, [limit:  int])
```

Split a `value` around matches of `pattern`. Use `limit` to indicate the number of times the pattern should be applied.

Note: Patterns have to be valid Java String literals, please ensure you escape any backslashes in your regular expressions!

---

regex

```
regex(pattern:  string, value:  string, [group_names:  array[string])
```

Match the regular expression in `pattern` against `value`. Returns a match object, with the boolean property `matches` to indicate whether the regular expression matched and, if requested, the matching groups as `groups`. The groups can optionally be named using the `group_names` array. If not named, the groups names are strings starting with `"0"`.

Note: Patterns have to be valid Java String literals, please ensure you escape any backslashes in your regular expressions!

---

regex_replace

```
regex_replace(pattern:  string, value:  string, replacement:  string,
[replace_all:  boolean])
```

Match the regular expression in `pattern` against `value` and replace it, if matched, with `replacement`. You can use numbered capturing groups and reuse them in the replacement string. If `replace_all` is set to `true`, then all matches will be replaced, otherwise only the first match will be replaced.

Examples:

```
// message = 'logged in user: mike'
let username = regex_replace(".*user: (.*)", to_string($message.message), "$1");

// message = 'logged in user: mike'
let string = regex_replace("logged (in|out) user: (.*)", to_string($message.message),
→"User $2 is now logged $1");
```

Note: Patterns have to be valid Java String literals, please ensure you escape any backslashes in your regular expressions!

---

grok

```
grok(pattern:  string, value:  string, [only_named_captures:  boolean])
```

Applies the grok pattern `grok` to `value`. Returns a match object, containing a Map of field names and values. You can set `only_named_captures` to `true` to only return matches using named captures.

Tip: The result of executing the `grok` function can be passed as argument for *set_fields* to set the extracted fields into a message.

See also:

- *set_fields*

key_value

```
key_value(
  value: string,
  [delimiters: string],
  [kv_delimiters: string],
  [ignore_empty_values: boolean],
  [allow_dup_keys: boolean],
  [handle_dup_keys: string],
  [trim_key_chars: string],
  [trim_value_chars: string]
)
```

Extracts key-value pairs from the given `value` and returns them as a Map of field names and values. You can optionally specify:

delimiters Characters used to separate pairs. We will use each character in the string, so you do not need to separate them. Default value: `<whitespace>`.

kv_delimiters Characters used to separate keys from values. Again, there is no need to separate each character. Default value: =.

ignore_empty_values Ignores keys containing empty values. Default value: `true`.

allow_dup_keys Indicates if duplicated keys are allowed. Default value: `true`.

handle_dup_keys How to handle duplicated keys (if `allow_dup_keys` is set). It can take the values `take_first`, which will only use the first value for the key; or `take_last`, which will only use the last value for the key. Setting this option to any other value will change the handling to concatenate, which will combine all values given to the key, separating them with the value set in this option. For example, setting `handle_dup_keys: ","`, would combine all values given to a key `a`, separating them with a comma, such as `1,2,foo`. Default value: `take_first`.

trim_key_chars Characters to trim (remove from the beginning and end) from keys. Default value: no trim.

trim_value_chars Characters to trim (remove from the beginning and end) from values. Default value: no trim.

Tip: The result of executing the `key_value` function can be passed as argument for *set_fields* to set the extracted fields into a message.

See also:

- *set_fields*

crc32

```
crc32(value:  string)
```

Creates the hex encoded CRC32 digest of the `value`.

crc32c

crc32c(value: string)

Creates the hex encoded CRC32C (RFC 3720, Section 12.1) digest of the `value`.

md5

md5(value: string)

Creates the hex encoded MD5 digest of the `value`.

murmur3_32

murmur3_32(value: string)

Creates the hex encoded MurmurHash3 (32-bit) digest of the `value`.

murmur3_128

murmur3_128(value: string)

Creates the hex encoded MurmurHash3 (128-bit) digest of the `value`.

sha1

sha1(value: string)

Creates the hex encoded SHA1 digest of the `value`.

sha256

sha256(value: string)

Creates the hex encoded SHA256 digest of the `value`.

sha512

sha512(value: string)

Creates the hex encoded SHA512 digest of the `value`.

parse_json

parse_json(value: string)

Parses the `value` string as JSON, returning the resulting JSON tree.

See also:

- *to_map*

---

select_jsonpath

`select_jsonpath(json: JsonNode, paths: Map<string, string>)`

Evaluates the given `paths` against the `json` tree and returns the map of the resulting values.

See also:

- *is_json*

- *parse_json*

to_ip

`to_ip(ip: string)`

Converts the given `ip` string to an IpAddress object.

See also:

- *cidr_match*

cidr_match

`cidr_match(cidr: string, ip: IpAddress)`

Checks whether the given `ip` address object matches the `cidr` pattern.

See also:

- *to_ip*

from_input

`from_input(id: string | name: string)`

Checks whether the currently processed message was received on the given input. The input can be looked up by either specifying its `name` (the comparison ignores the case) or the `id`.

route_to_stream

`route_to_stream(id: string | name: string, [message: Message], [remove_from_default: boolean])`

Routes the `message` to the given stream. The stream can be looked up by either specifying its `name` or the `id`.

If `message` is omitted, this function uses the currently processed message.

This causes the message to be evaluated on the pipelines connected to that stream, unless the stream has already been processed for this message.

If `remove_from_default` is `true`, the message is also removed from the default stream "All messages".

Example:

```
// Route the current processed message to a stream with ID `512bad1a535b43bd6f3f5e86`␣
→(preferred method)
route_to_stream(id: "512bad1a535b43bd6f3f5e86");

// Route the current processed message to a stream named `Custom Stream`
route_to_stream(name: "Custom Stream");
```

remove_from_stream

```
remove_from_stream(id:  string | name:  string, [message:  Message])
```

Removes the `message` from the given stream. The stream can be looked up by either specifying its `name` or the `id`.

If `message` is omitted, this function uses the currently processed message.

If the message ends up being on no stream anymore, it is implicitly routed back to the default stream "All messages". This ensures that you the message is not accidentally lost due to complex stream routing rules. If you want to discard the message entirely, use the `drop_message` function.

create_message

```
create_message([message:  string], [source:  string], [timestamp:  DateTime])
```

Creates a new message with from the given parameters. If any of them is omitted, its value is taken from the corresponding fields of the currently processed message. If `timestamp` is omitted, the timestamp of the created message will be the timestamp at that moment.

clone_message

```
clone_message([message:  Message])
```

Clones a message. If `message` is omitted, this function uses the currently processed message.

drop_message

```
drop_message(message:  Message)
```

The processing pipeline will remove the given `message` after the rule is finished executing.

If `message` is omitted, this function uses the currently processed message.

This can be used to implement flexible blacklisting based on various conditions.

Example:

```
rule "drop messages over 16383 characters"
when
    has_field("message") AND
    regex(pattern: "^.{16383,}$", value: to_string($message.message)).matches == true
then
    drop_message();
    // added debug message to be notified about the dropped message
    debug( concat("dropped oversized message from ", to_string($message.source)));
end
```

has_field

```
has_field(field:  string, [message:  Message])
```

Checks whether the given `message` contains a field with the name `field`.

If `message` is omitted, this function uses the currently processed message.

remove_field

```
remove_field(field:  string, [message:  Message])
```

Removes the given field with the name `field` from the given `message`, unless the field is reserved.

If `message` is omitted, this function uses the currently processed message.

set_field

```
set_field(field:  string, value:  any, [prefix:  string], [suffix:  string],
[message:  Message])
```

Sets the given field named `field` to the new `value`. The `field` name must be valid, and specifically cannot include a `.` character. It is trimmed of leading and trailing whitespace. String values are trimmed of whitespace as well.

The optional `prefix` and `suffix` parameters specify which prefix or suffix should be added to the inserted field name.

If `message` is omitted, this function uses the currently processed message.

See also:

- *set_fields*

set_fields

```
set_fields(fields:  Map<string, any>, [prefix:  string], [suffix:  string],
[message:  Message])
```

Sets all of the given name-value pairs in `field` in the given message. This is a convenience function acting like *set_field*. It can be helpful for using the result of a function like *select_jsonpath* or *regex* in the currently processed message especially when the key names are the result of a regular expression.

The optional `prefix` and `suffix` parameters specify which prefix or suffix should be added to the inserted field names.

If `message` is omitted, this function uses the currently processed message.

See also:

- *set_field*
- *to_map*
- *grok*
- *key_value*

rename_field

`rename_field(old_field: string, new_field: string, [message: Message])`

Modifies the field name `old_field` to `new_field` in the given message, keeping the field value unchanged.

syslog_facility

`syslog_facility(value: any)`

Converts the syslog facility number in `value` to its string representation.

syslog_level

`syslog_level(value: any)`

Converts the syslog severity number in `value` to its string representation.

expand_syslog_priority

`expand_syslog_priority(value: any)`

Converts the syslog priority number in `value` to its numeric severity and facility values.

expand_syslog_priority_as_string

`expand_syslog_priority_as_string(value: any)`

Converts the syslog priority number in `value` to its severity and facility string representations.

now

`now([timezone: string])`

Returns the current date and time. Uses the default time zone `UTC`.

See also:

- *is_date*

parse_date

`parse_date(value: string, pattern: string, [locale: string], [timezone: string])`

Parses the `value` into a date and time object, using the `pattern`. If no timezone is detected in the pattern, the optional timezone parameter is used as the assumed timezone. If omitted the timezone defaults to `UTC`.

The format used for the `pattern` parameter is identical to the pattern of the Joda-Time DateTimeFormat.

Symbol	Meaning	Presentation	Examples
G	era	text	AD
C	century of era (>=0)	number	20
Y	year of era (>=0)	year	1996
x	weekyear	year	1996
w	week of weekyear	number	27
e	day of week	number	2
E	day of week	text	Tuesday; Tue
y	year	year	1996
D	day of year	number	189
M	month of year	month	July; Jul; 07
d	day of month	number	10
a	halfday of day	text	PM
K	hour of halfday (0~11)	number	0
h	clockhour of halfday (1~12)	number	12
H	hour of day (0~23)	number	0
k	clockhour of day (1~24)	number	24
m	minute of hour	number	30
s	second of minute	number	55
S	fraction of second	millis	978
z	time zone	text	Pacific Standard Time; PST
Z	time zone offset/id	zone	-0800; -08:00; America/Los_Angeles
'	escape for text	delimiter	
''	single quote	literal	'

The format used for the `locale` parameter is a valid language tag according to IETF BCP 47 which can be parsed by the Locale#forLanguageTag(String) method.

Also see IANA Language Subtag Registry.

If no locale was specified, the locale of the system running Graylog (the default locale) is being used.

Examples:

Language Tag	Description
en	English
en-US	English as used in the United States
de-CH	German for Switzerland

See also:

- *is_date*

flex_parse_date

```
flex_parse_date(value:  string, [default:  DateTime], [timezone:  string])
```

Uses the Natty date parser to parse a date and time `value`. If no timezone is detected in the pattern, the optional timezone parameter is used as the assumed timezone. If omitted the timezone defaults to `UTC`.

In case the parser fails to detect a valid date and time the `default` date and time is being returned, otherwise the expression fails to evaluate and will be aborted.

See also:

- *is_date*

parse_unix_milliseconds

```
parse_unix_milliseconds(value:  long)
```

Attempts to parse a UNIX millisecond timestamp (milliseconds since 1970-01-01T00:00:00.000Z) into a proper `DateTime` object.

Example:

```
// 1519902000000 == 2018-03-01T12:00:00.000Z
let timestamp = parse_unix_milliseconds(1519902000000);
set_field("timestamp", timestamp);
```

See also:

- *is_date*

format_date

```
format_date(value:  DateTime, format:  string, [timezone:  string])
```

Returns the given date and time `value` formatted according to the `format` string. If no timezone is given, it defaults to `UTC`.

to_date

```
to_date(value:  any, [timezone:  string])
```

Converts `value` to a date. If no `timezone` is given, it defaults to `UTC`.

See also:

- *is_date*

years

```
years(value:  long)
```

Create a time period with `value` number of years.

See also:

- *is_period*
- *period*

months

```
months(value:   long)
```

Create a time period with `value` number of months.

See also:

- *is_period*

- *period*

weeks

```
weeks(value:   long)
```

Create a time period with `value` number of weeks.

See also:

- *is_period*

- *period*

days

```
days(value:   long)
```

Create a time period with `value` number of days.

See also:

- *is_period*

- *period*

hours

```
hours(value:   long)
```

Create a time period with `value` number of hours.

See also:

- *is_period*

- *period*

minutes

```
minutes(value:   long)
```

Create a time period with `value` number of minutes.

See also:

- *is_period*

- *period*

seconds

```
seconds(value:   long)
```

Create a time period with `value` number of seconds.

See also:

- *is_period*
- *period*

millis

```
millis(value:   long)
```

Create a time period with `value` number of milliseconds.

See also:

- *is_period*
- *period*

period

```
period(value:   string)
```

Parses an ISO 8601 time period from `value`.

See also:

- *is_period*
- *years*
- *months*
- *days*
- *hours*
- *minutes*
- *seconds*
- *millis*

lookup

```
lookup(lookup_table:   string, key:   any, [default:   any])
```

Looks up a multi value in the named lookup table.

Example:

```
rule "dst_ip geoip lookup"
when
  has_field("dst_ip")
then
  let geo = lookup("geoip-lookup", to_string($message.dst_ip));
  set_field("dst_ip_geolocation", geo["coordinates"]);
```

```
   set_field("dst_ip_geo_country_code", geo["country"].iso_code);
   set_field("dst_ip_geo_country_name", geo["country"].names.en);
   set_field("dst_ip_geo_city_name", geo["city"].names.en);
end
```

lookup_value

```
lookup_value(lookup_table:  string, key:  any, [default:  any])
```

Looks up a single value in the named lookup table.

Example:

```
// Lookup a value in lookup table "ip_lookup" where the key is the string␣
↪representation of the src_addr field.
lookup_value("ip_lookup", to_string($message.src_addr));
```

15.5 Usage

15.5.1 Overview

Once you understand the concepts explained in *Pipelines*, *Rules*, and *Stream connections*, you're ready to start creating your own processing pipelines. This page gives you the information you need to get started with the user interface.

15.5.2 Configuration

Configure the message processor

Before start using the processing pipelines you need to ensure the *Pipeline Processor* message processor is enabled and correctly configured. You can do so by going to the *System -> Configurations* page, and checking the configuration in the *Message Processors Configuration* section.

On the Configurations page, you need to **enable the Pipeline Processor** message processor and, if you want your pipelines to have access to static fields set on inputs and/or fields set by extractors, **set the Pipeline Processor after the Message Filter Chain**.

15.5.3 Manage rules

You can create, edit, and delete your pipeline rules in the *Manage rules* page, under *System -> Pipelines*.

Clicking on *Create Rule* or *Edit* in one of the rules will open a page where you can write your own rule. The page lists available functions and their details to make the task a bit more manageable.

15.5.4 Managing pipelines

Once there are some rules in Graylog, you can create pipelines that use them to modify and enrich your messages.

To manage your pipelines, access *Manage pipelines* page under *System -> Pipelines*. This page is where you can create, edit, and delete pipelines.
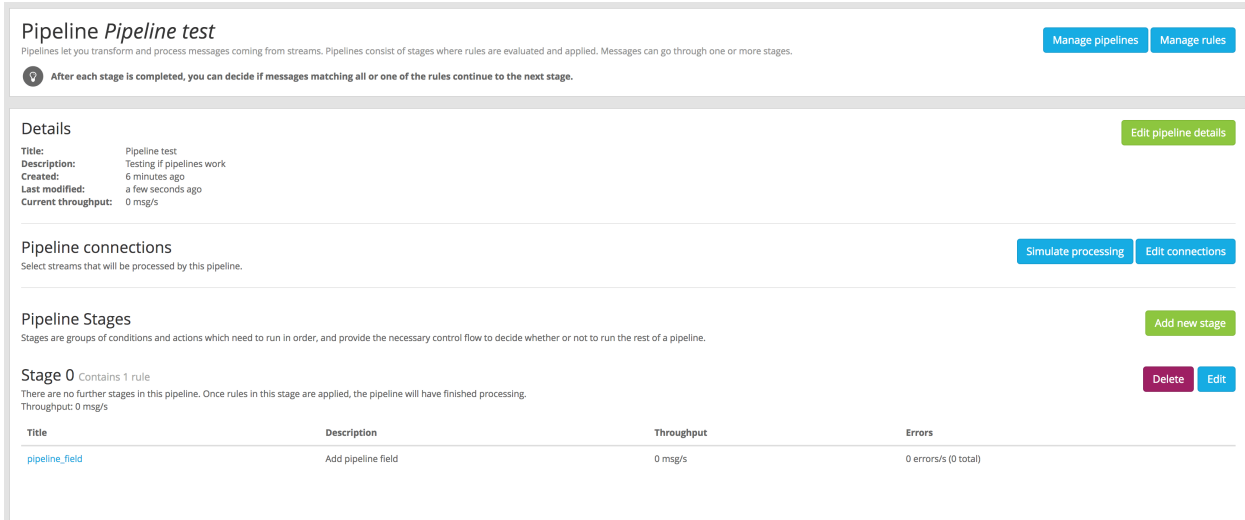
In order to create or edit pipelines, and as explained in *Pipelines*, you need to add your rules to a stage, which has a certain priority. The Web interface will let you add rules to the default stage (priority 0), and to create new stages with potentially different priorities.

A pipeline can have more than one stage, and when you create or edit a stage you need to select how to proceed to the next stage in the pipeline:

All rules on this stage match the message This option will only consider further stages in the pipeline when all conditions in rules evaluated in this stage are `true`. This is equivalent to `match all` in the *Pipelines* section.

At least one of the rules on this stage matches the message Selecting this option will continue to further stages in the pipeline when one or more of the conditions in rules evaluated in this stage are `true`. This is equivalent to `match either` in the *Pipelines* section.

15.5.5 Connect pipelines to streams

You can decide which streams are connected to a pipeline from the pipeline details page. Under *System -> Pipelines*, click on the title of the pipeline you want to connect to a stream, and then click on the *Edit connections* button.

You can assign many pipelines to the same stream, in which case all connected pipelines will process messages routed into that stream based upon the overall order of stage priorities.

Remember, as mentioned in the *Stream connections* documentation, the *All messages* stream is where all messages are initially routed, and is therefore a good place to apply pipelines applicable to all of your messages. Such pipelines might be responsible for stream routing, blacklisting, field manipulation, etc.

15.5.6 Simulate your changes

After performing some changes in a processing pipeline, you most likely want to see how they are applied to incoming messages. This is what the pipeline simulator is for.

Click the *Simulate processing* button under *System -> Pipelines* or in the pipeline details page to access the pipeline simulator.

In order to test the message processing you need to provide a raw message that will be routed into the stream you want to simulate. The raw message should use the same format Graylog will receive. For example: you can type a *GELF* message, in the same format your GELF library would send, in the *Raw message* field. Don't forget to select the correct codec for the message you provide.

After specifying the message and codec, click *Load message* to start the simulation and display the results.

The simulation provides the following results:

Changes summary Provides a summary of modified fields in the original message, as well as a list of added and
dropped messages.

Results preview Shows all fields in the processed message.

Simulation trace Displays a trace of the processing, indicating which rules were evaluated and which were executed.
It also includes a timeline, in microseconds, to allow you to see which rules and pipelines are taking up the most
time during message processing.

LOOKUP TABLES

Graylog 2.3 introduced the lookup tables feature. It allows you to lookup/map/translate message field values into new values and write them into new message fields or overwrite existing fields. A simple example is to use a static CSV file to map IP addresses to host names.

16.1 Components

The lookup table systems consists of four components.

- Data adapters
- Caches
- Lookup tables
- Lookup results

16.1.1 Data Adapters

Data adapters are used to do the actual lookup for a value. They might read from a CSV file, connect to a database or execute HTTP requests to receive the lookup result.

Data adapter implementations are pluggable and new ones can be added through plugins.

16.1.2 Caches

The caches are responsible for caching the lookup results to improve the lookup performance and/or to avoid overloading databases and APIs. They are separate entities to make it possible to reuse a cache implementation for different data adapters. That way, the data adapters do not have to care about caching and do not have to implement it on their own.

Cache implementations are pluggable and new ones can be added through plugins.

Important: The CSV file adapter reads the entire contents of the file into HEAP memory. Ensure that you size the HEAP accordingly.

Note: The CSV file adapter refreshes its contents within each check interval if the file was changed. If the cache was purged but the check interval has not elapsed, lookups might return expired values.

16.1.3 Lookup Tables

The lookup table component ties together a data adapter instance and a cache instance. It is needed to actually enable the usage of the lookup table in extractors, converters, pipeline functions and decorators.

16.1.4 Lookup Results

The lookup result is returned by a lookup table through the data adapter and can contain two types of data. A **single value** and a **multi value**.

The **single value** can be a string, number or boolean and will be used in extractors, converters, decorators and pipeline rules. In our CSV example to lookup host names for IP addresses, this would be the host name string.

A **multi value** is a map/dictionary-like data structure and can contain several different values. This is useful if the data adapter can provide multiple values for a key. A good example for this would be the geo-ip data adapter which does not only provide the latitude and longitude for an IP address, but also information about the city and country of the location. Currently, the multi value can only be used in a pipeline rule when using the `lookup()` pipeline function.

Example 1: Output for a CSV data adapter including a single value and a multi value.

Test lookup
You can manually trigger the data adapter using this form. The data will be not cached.

Key

127.0.0.1

Key to look up a value for.

Look up

Lookup result

```
{
  "single_value": "localhost",
  "multi_value": {
    "value": "localhost"
  },
  "ttl": 9223372036854776000,
  "empty": false
}
```

Example 2: Output for the geo-ip data adapter including a single value and a multi value.

Test lookup
You can manually trigger the data adapter using this form. The data will be not cached.

Key

```
8.8.8.8
```

Key to look up a value for.

Look up

Lookup result

```
{
  "single_value": "37.3845,-122.0881",
  "multi_value": {
    "city": {
      "confidence": null,
      "geoname_id": 5375480,
      "names": {
        "de": "Mountain View",
        "ru": "Маунтин-Вью",
        "ja": "マウンテンビュー",
        "en": "Mountain View",
        "fr": "Mountain View",
        "zh-CN": "芒廷维尤"
      }
    },
    "continent": {
```

16.2 Setup

The lookup tables can be configured on the "System/Lookup Tables" page.

You need to create at least one data adapter and one cache before you can create your first lookup table. The following example setup creates a lookup table with a CSV file data adapter and an in-memory cache.

16.2.1 Create Data Adapter

Navigate to "System/Lookup Tables" and click the "Data Adapters" button in the top right corner. Then you first have to select a data adapter type.

Every data adapter form includes data adapter specific documentation that helps you to configure it correctly.

16.2.2 Create Cache

Navigate to "System/Lookup Tables" and click the "Caches" button in the top right corner. Then you first have to select a cache type.

Every cache form includes cache specific documentation that helps you to configure it correctly.

16.2.3 Create Lookup Table

Now you can create a lookup table with the newly created data adapter and cache by navigating to "System/Lookup Tables" and clicking "Create lookup table".

Make sure to select the data adapter and cache instances in the creation form.

Default Values

Every lookup table can optionally be configured with default values which will be used if a lookup operation does not return any result.

16.3 Usage

Lookup tables can be used with the following Graylog components.

- Extractors
- Converters
- Decorators
- Pipeline rules

16.3.1 Extractors

A lookup table extractor can be used to lookup the value of a message field in a lookup table and write the result into a new field or overwrite an existing field.

16.3.2 Converters

When you use an extractor to get values out of a text message, you can use a lookup table converter to do a lookup on the extracted value.

16.3.3 Decorators

A lookup table decorator can be used to enrich messages by looking up values at search time.

16.3.4 Pipeline Rules

There are two lookup functions that can be used in a pipeline rule, `lookup()` and `lookup_value()`. The first returns the **multi value** data of the lookup result, the second returns the **single value**.

16.4 Built-in Data Adapters

The following Data Adapters are shipped with Graylog by default. Detailed on-screen documentation for each is available on the Add/Edit Data Adapter page in Graylog.

16.4.1 CSV File Adapter

Performs key/value lookups from a CSV file.

16.4.2 DNS Lookup Adapter

Provides the ability to perform the following types of DNS resolutions:

- Resolve hostname to IPv4 address (A records)
- Resolve hostname to IPv6 address (AAAA records)
- Resolve hostname to IPv4 and IPv6 address (A and AAAA records)
- Reverse lookup (PTR record)
- Text lookup (TXT records)

16.4.3 DSV File from HTTP Adapter

Performs key/value from a DSV file. This adapter supports more advanced customization than the CSV File adapter (such a custom delimiter and customizable key/value columns).

16.4.4 HTTP JSONPath Adapter

Executes HTTP GET requests to lookup a key and parses the result based on configured JSONPath expressions.

16.4.5 Geo IP - MaxMind Databases

Provides the ability to extract geolocation information of IP addresses from MaxMind ASN, Country and City databases.

SEVENTEEN

GEOLOCATION

Graylog lets you extract and visualize geolocation information from IP addresses in your logs. Here we will explain how to configure the geolocation resolution, and how to create a map with the extracted geo-information.

17.1 Setup

Graylog ships with geolocation capabilities by default but **some configuration is still required on your side**. This section explains how to configure the functionality in detail.

On Graylog 3.0, the preferred way of configuring geolocation is by using *Lookup Tables*, as it provides more flexibility and is compatible with more database types. If you would rather use the old Message Processor, please check the 2.5 documentation.

Note: Before you get started, we recommend taking a look at some Lookup Table concepts in *the documentation*.

17.1.1 Download the database

In the first place, you need to download a geolocation database. The Lookup Table Geo IP Data Adapter supports both **MaxMind City and Country databases** in the **MaxMind DB format**, as the GeoIP2 Databases or GeoLite2 Databases that MaxMind provides.

The next step is to store the geolocation database on all servers running Graylog. Make sure you grant the right permissions to the file so the user running Graylog can read the database.

Note: As-of December 30, 2019, you will need to create an account to obtain a license key to download the database(s).

More information is available on MaxMind's blog post.

17.1.2 Configure Lookup Table

The next step is to configure a Graylog Lookup Table that is able to use the geolocation database. Follow the *Lookup Tables setup documentation* to see what you need to do. In most common cases you need to:

1. Create a Geo IP Data Adapter and point it to the location where you store the database. You can additionally test the Data Adapter to ensure it all works as expected.

2. Create a Cache (if needed) to make your lookups faster.

3. Create a Lookup Table that uses the Data Adapter and Cache you created in previous steps.

17.1.3 Use the Lookup Table

Now you are almost ready to extract geolocation information from IP addresses. All you need to do is to use the Lookup Table you created in the previous step in a Extractor, Converter, Decorator or Pipeline Rule. Take a look at the *Lookup Tables usage documentation* for more information.

Note: Make sure to read *The importance of message processor ordering*, specially if you will use the Lookup Table with a Pipeline, in order to better understand how Graylog will process messages.

17.2 Visualize geolocations in a map

Graylog can display maps from geolocation stored in any field, as long as the geo-points are using the `latitude, longitude` format. The default return value of the Geo IP Data Adapter returns the coordinates in the right format, so you most likely don't need to do anything special if you are using a Lookup Table for extracting geolocation information.

17.2.1 Display a map in the search results page

On any search result page, you can expand the field you want to use to draw a map in the search sidebar, and click on the *World Map* link. That will show a map with all different points stored in that field.
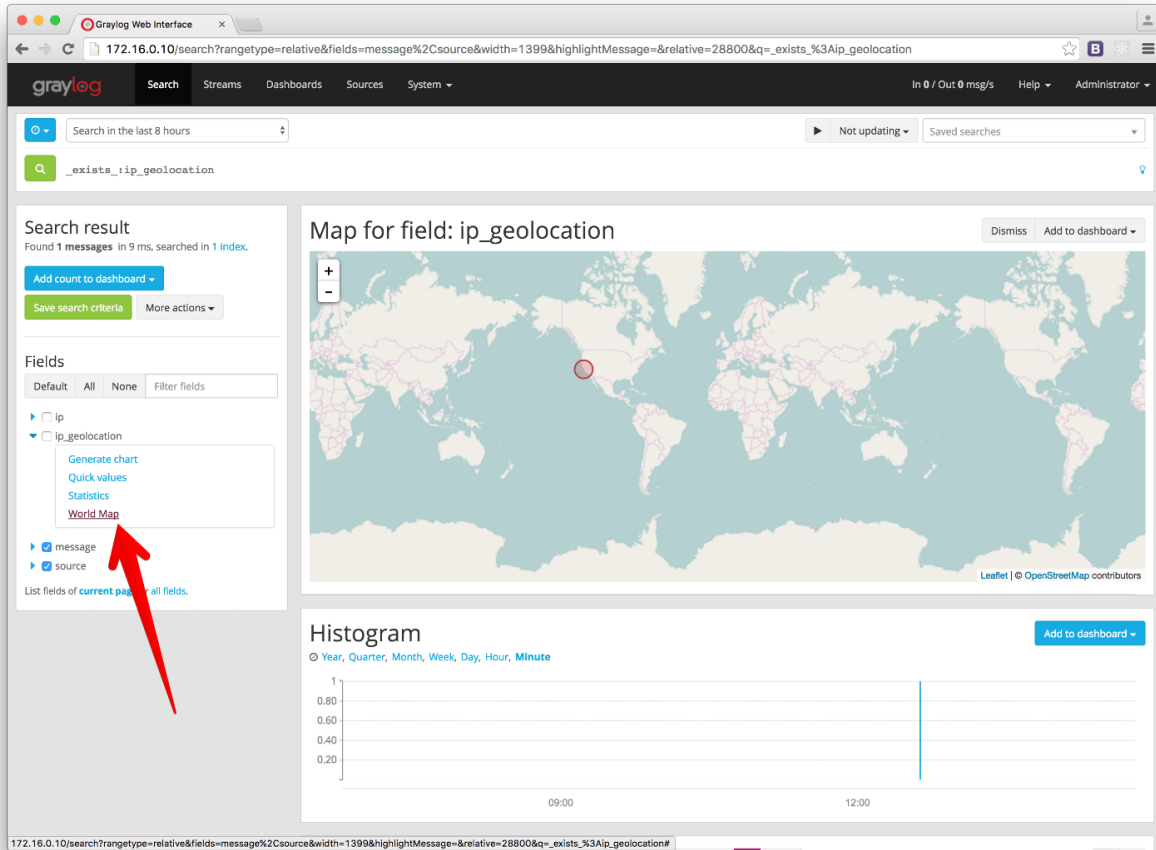
17.2.2 Add map to a dashboard

You can add the map visualization into any dashboards as you do with other widgets. Once you displayed a map in the search result page, click on *Add to dashboard*, and select the dashboard where you want to add the map.
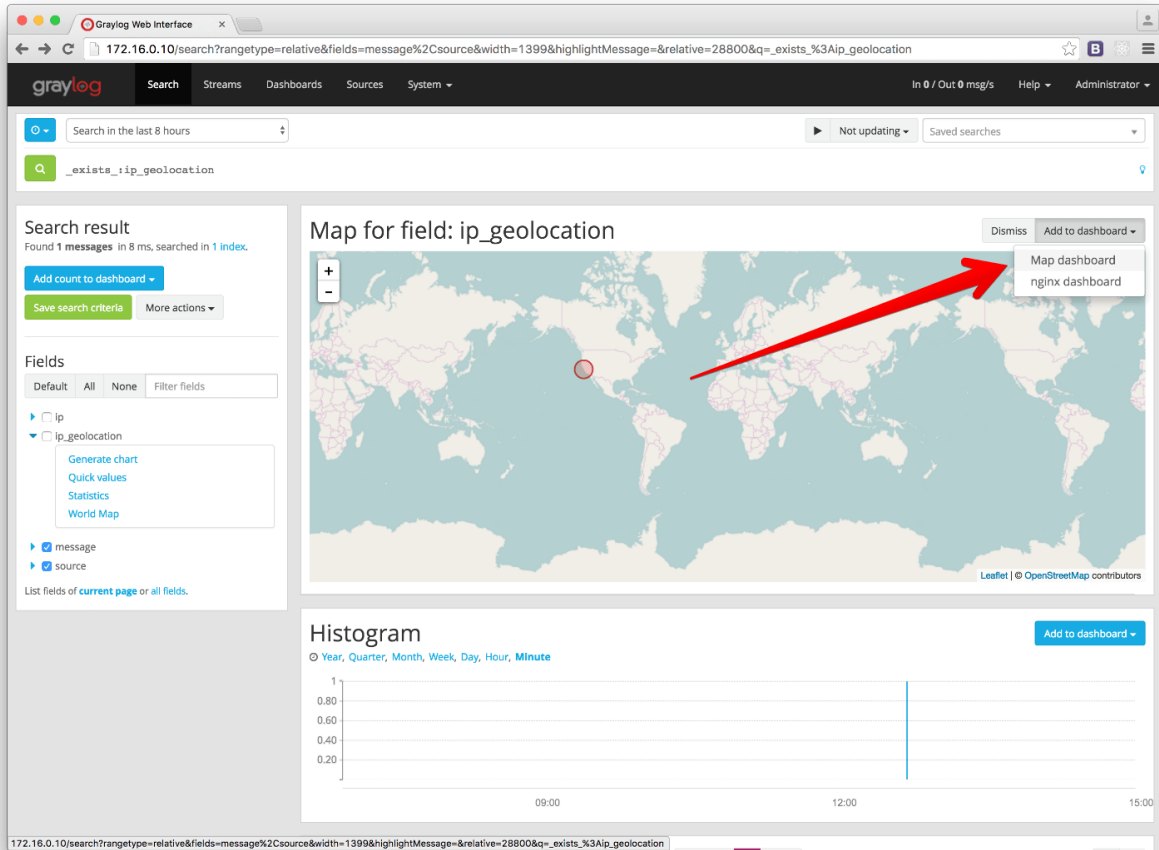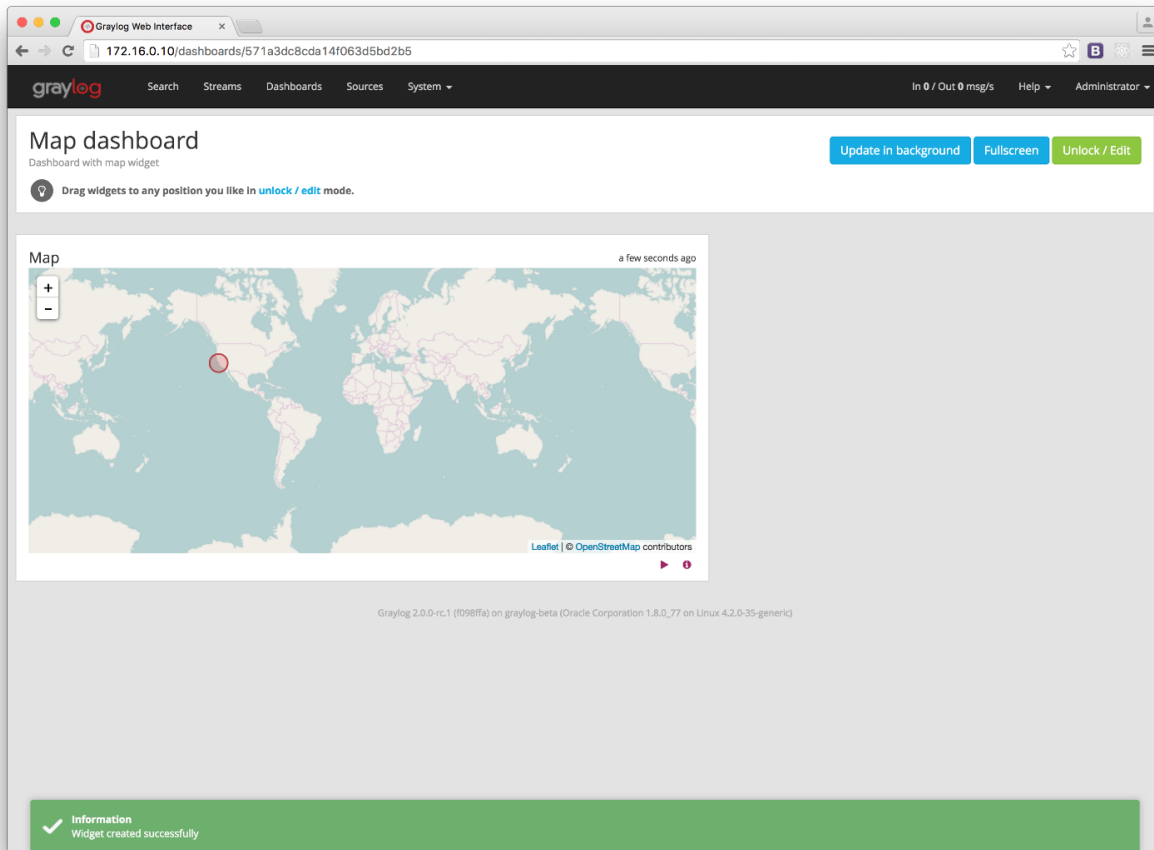
17.3 FAQs

17.3.1 Will Graylog extract IPs from all fields?

No, you can configure which fields you want to extract data from in the Pipeline Rule or Extractor using the Lookup Table configured in the *setup section*.

17.3.2 What geo-information is extracted from IPs?

Depending on the database you use, the extracted information will be different. By using a Pipeline Rule alongside a Lookup Table, you can extract any information returned by the MaxMind Database for the IP in your logs.

17.3.3 Where is the extracted geo-information stored?

Extracted geo-information is stored in message fields, which you can name as you wish.

17.3.4 Which geo-points format does Graylog use to store coordinates?

Graylog returns the geolocation information in the `latitude,longitude` format. The Map visualization also requires that format to be able to draw the coordinates on a map.

17.3.5 I have a field in my messages with coordinates information already, can I use it in Graylog?

Yes, you can display a map for coordinates as long as they are in the `latitude,longitude` format.

17.3.6 Not all fields containing IP addresses are resolved. Why does this happen?

Most likely it is a misconfiguration issue. It is easier to extract information if **IP addresses are in their own field**. You should also make sure your **Message Processors are in the right order** in the *Message Processors Configuration*, as explained in *The importance of message processor ordering*.

INDEXER FAILURES

Every Graylog node is constantly keeping track about every indexing operation it performs. This is important for making sure that you are not silently losing any messages. The web interface can show you a number of write operations that failed and also a list of failed operations. Like any other information in the web interface this is also available via the REST APIs so you can hook it into your own monitoring systems.

Information about the indexing failure is stored in a capped MongoDB collection that is limited in size. A lot (many tens of thousands) of failure messages should fit in there but it should not be considered a complete collection of all errors ever thrown.

18.1 Common indexer failure reasons

There are some common failures that can occur under certain circumstances. Those are explained here:

18.1.1 MapperParsingException

An error message would look like this:

```
MapperParsingException[failed to parse [failure]]; nested: NumberFormatException[For
→input string: "some string value"];
```

You tried to write a `string` into a numeric field of the index. The indexer tried to convert it to a number, but failed because the `string` did contain characters that could not be converted.

This can be triggered by for example sending GELF messages with different field types or extractors trying to write `strings` without converting them to numeric values first. **The recommended solution is to actively decide on field types**. If you sent in a field like `http_response_code` with a numeric value then you should never change that type in the future.

The same can happen with all other field types like for example booleans.

Note that index cycling is something to keep in mind here. The first type written to a field per index wins. If the Graylog index cycles then the field types are starting from scratch for that index. If the first message written to that index has the `http_response_code` set as `string` then it will be a `string` until the index cycles the next time. Take a look at *Index model* for more information.

USERS AND ROLES

Graylog has a granular permission system which secures the access to its features. Each interaction which can look at data or change configuration in Graylog must be performed as an authenticated user.

Each user can have varying levels of access to Graylog's features, which can be controlled with assigning roles to users.

The following sections describe the capabilities of users and roles **and also how to use LDAP for authentication**.

19.1 Users

It is recommended to create an account for each individual user accessing Graylog.

User accounts have the usual properties such as a login name, email address, full name, password etc. In addition to these fields, you can also configure the session timeout, roles and timezone.

19.1.1 Sessions

Each login for a user creates a session, which is bound to the browser the user is currently using. Whenever the user interacts with Graylog this session is extended.

For security reasons you will want to have Graylog expire sessions after a certain period of inactivity. Once the interval specified by `timeout` expires the user will be logged out of the system. Requests like displaying throughput statistics do not extend the session, which means that if the user keeps Graylog open in a browser tab, but does not interact with it, their session will expire as if the browser was closed.

Logging out explicitly terminates the session.

19.1.2 Timezone

Since Graylog internally processes and stores messages in the UTC timezone, it is important to set the correct timezone for each user.

Even though the system defaults are often enough to display correct times, in case your team is spread across different timezones, each user can be assigned and change their respective timezone setting. You can find the current timezone settings for the various components on the **System / Overview** page of your Graylog web interface.

19.1.3 Initial Roles

Each user needs to be assigned at least one role, which governs the basic set of permissions this user has in Graylog.

Normal users, which do not need to create inputs, outputs or perform administrative tasks like managing access control etc, should be assigned the built in `Reader` role in addition to the custom roles which grant access to streams and dashboards.

19.2 Roles

In Graylog, roles are named collections of individual permissions which can be assigned to users. Previous Graylog versions could only assign individual permissions to each user in the system, making updating stream or dashboard permissions for a large group of users difficult to deal with.

Starting with Graylog 1.2 you can create roles which bundle permissions to streams and dashboards. These roles can then be assigned to any number of users and later be updated to include new streams and dashboards.

The two roles `Admin` and `Reader` are built in and cannot be changed. The `Admin` role grants all permissions and should only be assigned to users operating Graylog. The `Reader` role grants the basic permissions every user needs to be able to use Graylog. The interface will ensure that every user at least has the `Reader` role in addition to more business specific roles you create.

Roles cannot be deleted as long as users are still assigned to them to prevent accidentally locking users out.

19.2.1 Creating a role

In order to create a new role, choose the green **Add new role** button on the **System / Authentication / Roles** page.

This will display a dialog allowing you to describe the new role and select the permissions it grants.

After naming the role, select the permissions you want to grant using the buttons to the right of the respective stream or dashboard names. For each stream or dashboard you can select whether to grant `edit` or `read` permissions, but note that edit permissions always imply read permissions as well.

In case you have many streams or dashboards you can use the filter to narrow the list down, and use the checkboxes on the left hand side of the table to select multiple items. You can then use the bulk action buttons on the right hand side to toggle the permissions for all of the selected items at once.

Once you are done, be sure to save your changes. The save button is disabled until you select at least one permission.

19.2.2 Editing a role

Administrators can edit roles to add or remove access to new streams and dashboards in the system. The two built in `Admin` and `Reader` roles cannot be edited or deleted because they are vital for Graylog's permission system.

Simply choose the **Edit** button on the **System / Authentication / Roles** page and change the settings of the role in the following page:

You can safely rename the role as well as updating its description, the existing role assignment for users will be kept.

19.2.3 Deleting a role

Deleting roles checks whether a role still has users assigned to it, to avoid accidentally locking users out. If you want to remove a role, please remove it from all users first.

19.3 Permission system

The Graylog permission system is extremely flexible and allows you to create users that are only allowed to perform certain REST calls. The *Roles* UI allows you to create roles based on stream or dashboard access but does not expose permissions on a REST call level yet. This guide describes how to create those roles using the Graylog REST API.

Imagine we want to create a role that is only allowed to start or stop message processing on `graylog-server` nodes.

19.3.1 REST call permissions

Almost every REST call in Graylog has to be authenticated or it will return an HTTP 403 (Forbidden). In addition to that, the requesting user also has to have the permissions to execute the REST call. A Graylog admin user can always execute all calls and roles based on the standard stream or dashboard permissions can execute calls related to those entities.

If you want to create a user that can only execute calls to start or stop message processing you have to find the name of the required permission first.

You can learn about available permissions by querying the `/system/permissions` endpoint:

```
curl -XGET -u ADMIN:PASSWORD 'http://graylog.example.org:9000/api/system/permissions?
→pretty=true'
```

The server responds with a list such as this:

```
{
  "permissions" : {
    "outputs" : [ "create", "edit", "terminate", "read" ],
    "users" : [ "tokencreate", "rolesedit", "edit", "permissionsedit", "list",
→"tokenlist", "create", "passwordchange", "tokenremove" ],
    "processing" : [ "changestate" ],
    ...
  }
}
```

Starting and stopping message processing corresponds to the `changestate` permission in the `processing` category. We combine both pieces to the permission key `processing:changestate`.

19.3.2 Creating the role

You can create a new role using the REST API like this:

```
curl -v -XPOST -u ADMIN:PASSWORD -H 'Content-Type: application/json' -H 'X-Requested-
→By: cli' 'http://graylog.example.org:9000/api/roles' -d '{"read_only": false,
→"permissions": ["processing:changestate"],"name": "Change processing state",
→"description": "Permission to start or stop processing on Graylog nodes"}'
```

Notice the `processing:changestate` permission that we assigned. Every user with this role will be able to start and stop processing on `graylog-server` nodes. Graylog's standard `reader` permissions do not provide any access to data or maintenance functionalities.

This is the POST body in an easier to read formatting:

```
{
  "name": "Change processing state",
  "description": "Permission to start or stop processing on graylog-server nodes",
  "permissions": [
    "processing:changestate"
  ],
  "read_only": false
}
```

19.3.3 Assigning the role to a user

Create a new user in the Graylog web interface and assign the new role to it:

Every user needs to at least have the standard "Reader" permissions but those do not provide any access to data or maintenance functionalities.

Now request the user information to see what permissions have been assigned:

```
$ curl -XGET -u ADMIN:PASSWORD 'http://graylog.example.org:9000/api/users/
→maintenanceuser?pretty=true'
{
  "id" : "563d1024d4c63709999c4ac2",
  "username" : "maintenanceuser",
  "email" : "it-ops@example.org",
  "full_name" : "Rock Solid",
  "permissions" : [
    "indexercluster:read",
    "messagecount:read",
    "journal:read",
    "inputs:read",
    "metrics:read",
    "processing:changestate",
    "savedsearches:edit",
    "fieldnames:read",
    "buffers:read",
    "system:read",
    "users:edit:maintenanceuser",
    "users:passwordchange:maintenanceuser",
    "savedsearches:create",
    "jvmstats:read",
    "throughput:read",
    "savedsearches:read",
    "messages:read"
  ],
  "preferences" : {
    "updateUnfocussed" : false,
    "enableSmartSearch" : true
  },
  "timezone" : "America/Chicago",
  "session_timeout_ms" : 300000,
  "read_only" : false,
  "external" : false,
  "startpage" : { },
  "roles" : [
    "Change processing state",
    "Reader"
  ]
}
```

Now you can use this user in your maintenance scripts or automated tasks.

19.4 External authentication

19.4.1 LDAP / Active Directory

It is possible to use an external LDAP or Active Directory server to perform user authentication in Graylog.

Since Graylog 1.2.0, you can also use LDAP groups to perform authorization by mapping them to Graylog roles.

Configuration

To set up your LDAP or Active Directory server, go to **System / Authentication / LDAP/Active Directory**.

Once LDAP is enabled, you need to provide some details about the directory server.

Please test the server connection before continuing to the next steps.

User mapping

In order to be able to look for users in the LDAP server you configured, Graylog needs to know some more details about it: the base tree to limit user search queries, the pattern used to look for users, and the field containing the full name of the user. You can test the configuration any time by using the login test form that you can find at the bottom of that page.

The login test information will indicate if Graylog was able to load the given user (and perform authentication, if a password was provided), and it will display all LDAP attributes belonging to the user, as you can see in the screenshot.

That's it for the basic LDAP configuration. Don't forget to save your settings at this point!

Group mapping

You can additionally control the default permissions for users logging in with LDAP or Active Directory by mapping LDAP groups into Graylog roles. That is extremely helpful if you already use LDAP groups to authorize users in your organization, as you can control the default permissions members of LDAP groups will have.
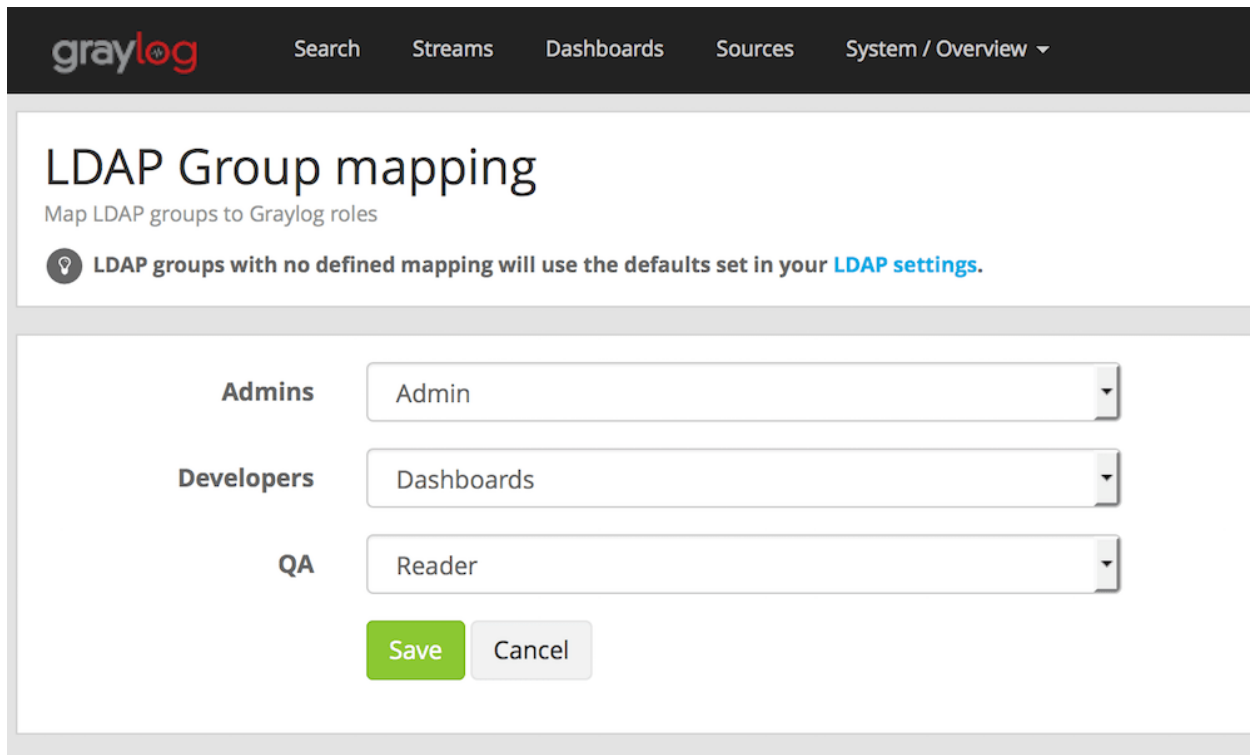
Once you configure group mapping, Graylog will rely on your LDAP groups to assign roles into users. That means that each time an LDAP user logs into Graylog, their roles will be assigned based on the LDAP groups their belong to.

In first place, you need to fill in the details in the *Group Mapping* section under **System / Authentication / LDAP/Active Directory**, by giving the base where to limit group searches, a pattern used to look for groups, and the group name attribute.

Then you need to select which default user role will be assigned to any users authenticated with the LDAP server should have. It is also possible to assign additional roles to any users logging in with LDAP. Please refer to *Roles* for more details about user roles.

Note: Graylog only synchronizes with LDAP when users log in. After changing the default and additional roles for LDAP users, you may need to modify existing users manually or delete them in order to force them to log in again.

You can test the group mapping information by using the login test form, as it will display LDAP groups that the test user belongs to. Save the LDAP settings once you are satisfied with the results.

Finally, in order to map LDAP groups into roles, you need to go to **System / Authentication / LDAP/Active Directory -> LDAP group mapping**. This page will load all available LDAP groups using the configuration you previously provided, and will allow you to select a Graylog role which defines the permissions that group will have inside Graylog.

Note: Loading LDAP groups may take some time in certain configurations, specially if you have many groups. In those cases, creating a better filter for groups may help with the loading times.

Note: Remember that Graylog only synchronizes with LDAP when users log in, so you may need to modify existing users manually after changing the LDAP group mapping.

Troubleshooting

LDAP referrals for groups can be a problem during group mapping. Referral issues are most likely to come up with larger AD setups. The Active Directory servers literally refer to other servers in search results, and it is the client's responsibility to follow all referrals. Support for that is currently not implemented in Graylog.

Referral issues can be detected by warnings in the server logs about group mapping failing, for example:

```
2016-04-11T15:52:06.045Z WARN  [LdapConnector] Unable to iterate over user's groups,
unable to perform group mapping. Graylog does not support LDAP referrals at the␣
↪moment.
Please see http://docs.graylog.org/en/2.2/pages/users_and_roles/external_auth.html
↪#troubleshooting
```

These issues may be resolved by either managing the groups manually, or configuring the LDAP connection to work against the global catalog. The first solution means simply that the LDAP group settings must not be set, and the groups are managed locally. The global catalog solution requires using the 3268/TCP, or 3269/TCP (TLS) port of eligible Active Directory server. The downside is that using the global catalog service consumes slightly more server resources.

19.4.2 Single Sign-On

The SSO Authentication Plugin for Graylog allows to use arbitrary HTTP request headers for authenticating Graylog users.

Once the plugin has been downloaded and installed on all Graylog nodes, it can be configured on the **System / Authentication / Single Sign-On (SSO)** page.

The HTTP request header containing the Graylog username can be configured in the **Username Header** field and should contain exactly one HTTP header name. Most HTTP request header based single sign-on solutions are using the `Remote-User` or `X-Forwarded-User` HTTP request header.

In order to only allow trusted proxy servers to provide the Graylog username, the **Request must come from a trusted proxy** checkbox must be checked. The list of trusted proxy servers can be edited on each Graylog node in the configuration file using the trusted_proxies configuration setting.

If user accounts not existing in the Graylog user database should automatically be created on the first login, the

Automatically create user checkbox must be checked. The automatically created users can also be customized to retrieve their full name or email address from another HTTP request header, otherwise the defaults are being used.
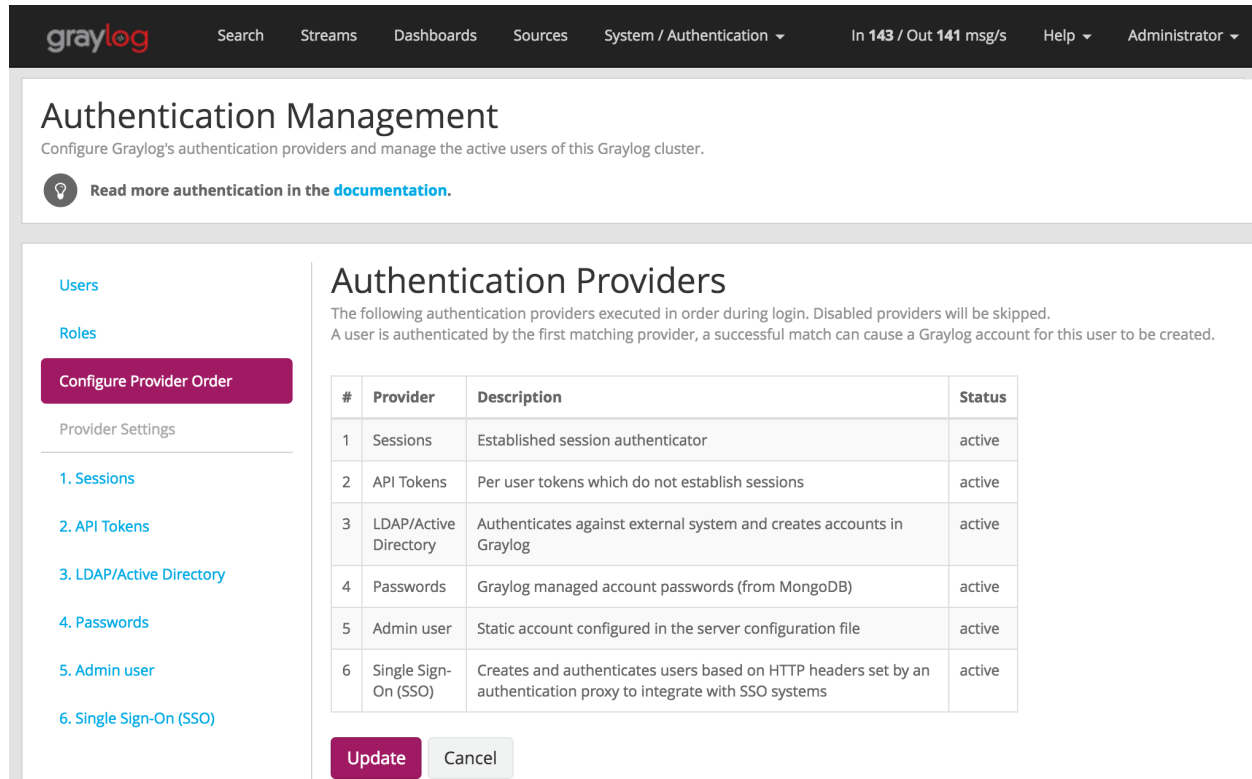
If **Synchronize the roles of the user from the specified HTTP header** is checked, Graylog expects the roles of the user in HTTP-Headers, with a prefix configured in **Role Header** (e.g. Roles would match Roles_0, Roles_1). The header value can contain a comma-separated list of role-names. Graylog adds only already existing roles to the user. Only the roles from the header will be synchronized to the roles of the user, if the user has roles, that are not supplied in the header, they will be removed from the user.

19.5 Authentication providers

Graylog 2.1.0 and later supports pluggable authentication providers. This means, that Graylog cannot only use the builtin authentication mechanisms like its internal user database, LDAP/Active Directory, or access tokens, but can also be extended by plugins to support other authentication mechanisms, for example Single Sign-On or Two Factor Authentication.

19.5.1 Configuration

The order in which the authentication providers will be queried can be configured in the Graylog web interface on the **System / Authentication / Configure Provider Order** page.

If a user tries to log into Graylog, the authentication providers will be queried in the configured order until a successful authentication attempt has been made (in which case the user will be logged in) or all authentication providers have denied authentication (in which case the user will not be logged in and get an error message).

By clicking on the **Update** button on the **System / Authentication / Configure Provider Order** page, the order of authentication providers can be customized.

Update Authentication Provider Configuration

Order
Use drag and drop to change the execution order of the authentication providers.

- ↕ Sessions
- ↕ API Tokens
- ↕ Single Sign-On (SSO)
- ↕ LDAP/Active Directory
- ↕ Passwords
- ↕ Admin user

Status
Change the checkboxes to change the status of an authentication provider.

Provider	Enabled
Sessions	☑
API Tokens	☑
Single Sign-On (SSO)	☑
LDAP/Active Directory	☑
Passwords	☑
Admin user	☑

Cancel   Save

INTEGRATIONS

20.1 Integrations Setup

Integrations are tools that help Graylog work with external systems and will typically be content packs, inputs, or lookup tables.

Integrations are distributed in two plugins:

- `graylog-integrations-plugins`

- `graylog-enterprise-integrations-plugins`

Important: The plugins needs to be installed on all your Graylog nodes!

20.1.1 Installation

To install the plugins, you can use one of the following options.

Operating System Packages

If you have installed Graylog using linux system packages (as described in the *Operating System Packages* installation guides), then you can use the following DEB or RPM operating system packages.

DEB

The installation on distributions like Debian or Ubuntu can be done with *apt-get* as installation tool from the previous installed online repository.

```
$ sudo apt-get install graylog-integrations-plugins graylog-enterprise-integrations-
→plugins
```

RPM

The installation on distributions like CentOS or RedHat can be done with *yum* as installation tool from the previous installed online repository.

```
$ sudo yum install graylog-integrations-plugins graylog-enterprise-integrations-
↪plugins
```

Tarballs

If you have done a manual installation, you download the tarballs from the following links.

- https://downloads.graylog.org/releases/graylog-integrations/graylog-integrations-plugins-3.0.0.tgz

- https://downloads.graylog.org/releases/graylog-enterprise-integrations/graylog-enterprise-integrations-plugins-3.0.0.tgz

20.1.2 Server Restart

Make sure to restart your Graylog servers once the plugins are installed.

20.1.3 Installation Success

The following server log message will indicate that each plugin was installed properly.

```
INFO: [CmdLineTool] Loaded plugin: Integrations Plugin 3.0.0 [org.graylog.
↪integrations.IntegrationsPlugin]
INFO: [CmdLineTool] Loaded plugin: Enterprise Integrations Plugin 3.0.0 [org.graylog.
↪integrations.EnterpriseIntegrationsPlugin]
```

Integrations are tools that help Graylog work with external systems. Integrations will typically be content packs, inputs, or lookup tables and can either be open source or Enterprise.

Reference the *Integrations Setup* document for installation instructions.

Below are the available features:

20.2 Open Source

- *AWS Kinesis/CloudWatch Input*
- *Palo Alto Network Input*

20.2.1 AWS Kinesis/CloudWatch Input

Note: This input is available since Graylog version 3.1.1. Installation of an additional `graylog-integrations-plugins` package is required. See the *Integrations Setup* page for more info.

Attention: An understanding of how AWS CloudWatch and Kinesis is required.

Attention: Versions of Graylog and `graylog-integrations-plugins` must be the same.

This input allows Graylog to read log messages from CloudWatch via Kinesis. When reading logs from CloudWatch, Kinesis is required in order to stream messages to Graylog.

The following message types are supported:

CloudWatch Logs Raw text strings within in Cloudwatch.

CloudWatch Flow Logs Flow Logs within a Cloud Watch log group.

Kinesis Raw Logs Raw text strings written to Kinesis.

Manual Setup Flow

For this setup to function as expected, the Least Privilege Policy shown below must be allowed for the authorized user. (See *Permission Policies* below)

1) **AWS Kinesis Authorize** Type in input name, AWS Access Key, AWS Secret Key and select AWS Region in order to authorize Graylog and click the **Authorize & Choose Stream** button to continue. (See image below.)

2) **AWS Kinesis Setup** Select the Kinesis stream to pull logs from and click the **Verify Stream & Format** button to continue.

3) **AWS CloudWatch Health Check** Graylog will read a message from the Kinesis stream and check it's format. We'll automatically parse the message if it's a Flow Log.

4) **AWS Kinesis Review** Final step to review and finalize the details for the input.

Automatic Setup Flow

When adding the AWS Kinesis/CloudWatch input to Graylog, you will be guided throughout the setup process. For this setup to function as expected, the Recommended Policy shown below must be allowed for the authorized user. (See *Permission Policies* below)

1) **AWS Kinesis Authorize** Type in input name, AWS Access Key, AWS Secret Key and select AWS Region in order to authorize Graylog and click the **Authorize & Choose Stream** button to continue. (See image above)

2) **AWS Kinesis Setup** In the blue dialog box (seen in the image above), click the **Setup Kinesis Automatically** button. Type in a name for the Kinesis stream name, and select a Cloudwatch log Group from the dropdown list and click the **Begin Automated Setup** button.

    You will be prompted with *Kinesis Auto Setup Agreement* and will need to acknowledge that you are aware of the resources that will be created and click the **I Agree! Create these AWS resources now.** button. (See images below).

    Once agreed and acknowledge, the auto-setup will detail and reference the resources that were created and you can click the **Continue Setup** button. (see *Executing Auto-Setup* image below)

3) **AWS CloudWatch Health Check** Graylog will read a message from the Kinesis stream and check its format. Graylog will attempt to automatically parse the message if it is of a known type.

4) **AWS Kinesis Review** Final step to review and finalize the details for the input.

Executing Auto-Setup

✔ Create Kinesis Stream

Success. The new stream [Kinesis_Stream/arn:aws:kinesis:⬛⬛⬛⬛⬛⬛⬛:stream /Kinesis_Stream] was created with [1] shard.

✔ Create Subscription Policy

Success! The role [graylog-cloudwatch-role-2019-08-30-12-17-08/arn:aws:iam::⬛⬛⬛⬛⬛:role/graylog-cloudwatch-role-2019-08-30-12-17-08] has been created.

✔ Create Subscription

Success. The subscription filter [filter-name] was added for the CloudWatch log group [CloudWatch_Log_Group].

It may take up to ten minutes for the first messages to arrive in the Kinesis stream. The Kinesis Health Check in the following step will not complete successfully until messages are present in the stream. Please see the official CloudWatch Subscriptions documentation for more information.

Continue Setup

Permission Policies

Manual Setup Flow Permissions

AWS Policy Permissions

Please verify that you have granted your AWS IAM user sufficient permissions. You can use the following policies for reference.

Show Recommended Policy   ❯
To be able to use all available functionality for Kinesis setup.

Hide Least Privilege Policy   ❯
Doesn't include Kinesis auto-subscription controls.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GraylogKinesisAutoSetup",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "logs:DescribeSubscriptionFilters",
        "logs:PutLogEvents",
        "kinesis:CreateStream",
        "kinesis:DescribeStreamConsumer",
        "kinesis:PutRecord",
        "kinesis:RegisterStreamConsumer"
      ],
      "Resource": "*"
    }
  ]
}
```

Automatic Setup Flow Permissions

AWS Policy Permissions

Please verify that you have granted your AWS IAM user sufficient permissions. You can use the following policies for reference.

Hide Recommended Policy
To be able to use all available functionality for Kinesis setup.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GraylogKinesisSetup",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData",
        "dynamodb:CreateTable",
        "dynamodb:DescribeTable",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaceAttribute",
        "ec2:DescribeNetworkInterfaces",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DescribeLoadBalancers",
        "iam:CreateRole",
        "iam:GetRole",
        "iam:PutRolePolicy",
        "kinesis:CreateStream",
        "kinesis:DescribeStream",
        "kinesis:GetRecords",
        "kinesis:GetShardIterator",
        "kinesis:ListShards",
        "kinesis:ListStreams",
        "logs:DescribeLogGroups",
        "logs:PutSubscriptionFilter"
      ],
      "Resource": "*"
    }
  ]
}
```

20.2.2 Palo Alto Networks Input

Note: This input is available since Graylog version 2.5.0. Installation of an additional `graylog-integrations-plugins` package is required. See the *Integrations Setup* page for more info.

This input allows Graylog to receive `SYSTEM`, `THREAT` and `TRAFFIC` logs directly from a Palo Alto device and the Palo Alto Panorama system. Logs are sent with a typical Syslog header followed by a comma-separated list of fields. The fields order might change between versions of PAN OS.

Example `SYSTEM` message:

```
<14>1 2018-09-19T11:50:35-05:00 Panorama-1 - - - - 1,2018/09/19 11:50:35,000710000506,
↪SYSTEM,general,0,2018/09/19 11:50:35,,general,,0,0,general,informational,"Deviating␣
↪device: Prod--2, Serial: 007255000045717, Object: N/A, Metric: mp-cpu, Value: 34",
↪1163103,0x0,0,0,0,0,,Panorama-1
```

To get started, add a new Palo Alto Networks Input (TCP) in the `System > Inputs` area in Graylog. Specify the Graylog Node, Bind address, Port, and adjust the field mappings as needed.

This input ships with a field configuration that is compatible with PAN OS 8.1. Other versions can easily be supported by customizing the `SYSTEM`, `THREAT` and `TRAFFIC` mappings on the Add/Edit input page in Graylog.

The configuration for each message type is a CSV block that must include the `position`, `field`, and `type` headers.

For example:

```
position,field,type
1,receive_time,STRING
2,serial_number,STRING
3,type,STRING
4,content_threat_type,STRING
5,future_use1,STRING
...
```

Accepted values for each column:

Field	Accepted Values
`position`	A positive integer value.
`field`	A contiguous string value to use for the field name. Must not include the reserved field names: `_id`, `message`, `full_message`, `source`, `timestamp`, `level`, `streams`
`type`	One of the following supported types: `BOOLEAN`, `LONG`, `STRING`

The validity of each CSV configuration is checked when the Palo Alto input is started. If the CSV is malformed (or contains invalid properties), the input will fail to start. An error describing the specific issue will be logged in the `graylog-server` log file and also displayed at the top of the `http://<grayloghost>/system/overview` page for the affected node.

For example:

The mappings for each type look like this on the add/edit input page:

System Message Mappings (optional)

```
position,field,type
1,receive_time,STING
2,serial_number,STRING
3,type,STRING
4,content_threat_type,STRING
5,future_use1,STRING
6,generated_time,STRING
7,virtual_system,STRING
8,event_id,STRING
9,object,STRING
```

CSV string representing the fields/positions/data types to parse. (See documentation)

Threat Message Mappings (optional)

```
position,field,type
1,receive_time,STRING
2,serial_number,STRING
3,type,STRING
4,threat_content_type,STRING
5,future_use1,STRING
6,generated_time,STRING
7,src_ip,STRING
8,dest_ip,STRING
9,nat_src_ip,STRING
```

CSV string representing the fields/positions/data types to parse. (See documentation)

Traffic Message Mappings (optional)

```
position,field,type
1,receive_time,STRING
2,serial_number,STRING
3,type,STRING
4,threat_content_type,STRING
5,future_use1,STRING
6,generated_time,STRING
7,src_ip,STRING
8,dest_ip,STRING
9,nat_src_ip,STRING
```

CSV representing the fields/positions/data types to parse. (See documentation)

Cancel   Save

The mappings built into the plugin by default are based on the following PAN OS 8.1 specifications. If you are running PAN OS 8.1, then there is no need to edit the mappings. However, if you are running a different version of PAN OS, please reference the official Palo Alto Networks log fields documentation that that version and customize the mappings on the Add/Edit Input page accordingly.

We have included a links to a few recent versions here for reference.

Version 8.1

- 8.1 - Traffic Log Fields

- 8.1 - Threat Log Fields

- 8.1 - System Log Fields

Version 8.0

- 8.0 - Traffic Log Fields

- 8.0 - Threat Log Fields

- 8.0 - System Log Fields

Version 7.1

- 7.1 - Traffic Log Fields

- 7.1 - Threat Log Fields

- 7.1 - System Log Fields

Also see Documentation for older PAN OS versions.

20.3 Enterprise

Enterprise Integrations plugin feature require an Graylog Enterprise license . For a comprehensive list of available features included, see our *Enterprise List page*

- *Forwarder*

- *Script Alert Notification*

20.3.1 Forwarder

The Forwarder provides the ability to forward messages from one Graylog cluster to another over HTTP/2. This centralizes log messages from multiple distributed Graylog source clusters into one destination cluster, which allows centralized alerting, reporting, and oversight.

Two Graylog clusters are required to use the Forwarder: A Graylog source cluster (Forwarder Output) and a Graylog destination cluster (Forwarder Input). The Graylog source cluster will forward messages, and the Graylog destination cluster will receive messages being forwarded.

Note: This is an Enterprise Integrations feature and is only available since Graylog version 3.0.1 and thus requires an Enterprise license. See the *Integrations Setup* page for more info.
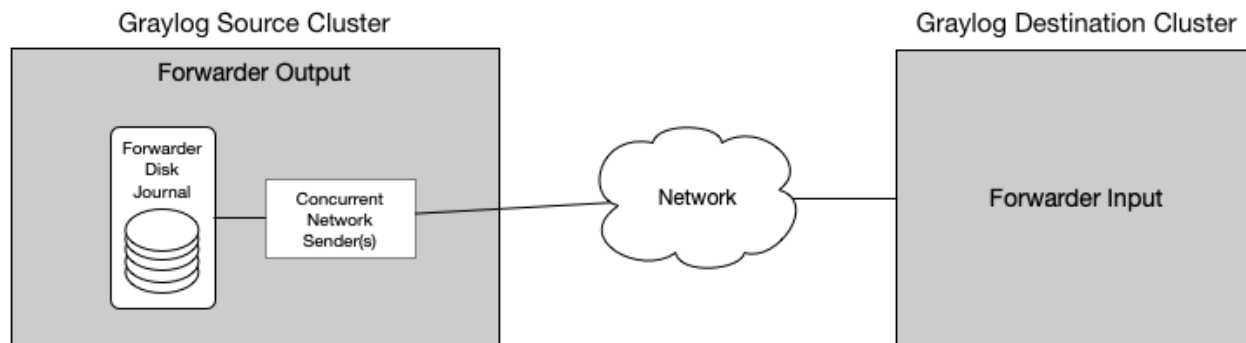
Forwarder Output

The Forwarder Output (Graylog source cluster) is responsible for forwarding messages to the Graylog destination cluster. It first writes the messages to an on-disk journal in the Graylog source cluster (Forwarder Output). Messages stay in the on-disk journal until the Graylog destination cluster is available to receive messages.

Messages are only forwarded until after they are done being processed through the pipeline of the Graylog source cluster, but simultaneously as they are written to Elasticsearch.

Forwarder Journal

The Forwarder is equipped with a disk journal. This journal immediately persists messages received from the Graylog Output system to disk before attempting to send them to the remote Graylog destination cluster. This allows the Forwarder to keep receiving and reliably queuing messages, even if the remote Graylog destination cluster is temporarily unavailable due to network issues. The Journal has many configuration options (such as Maximum Journal Size) available and described on below.

Forwarder Output Options

The Graylog Forwarder is capable of forwarding messages at very high throughput rates. Many hardware factors will affect throughput (such as CPU clock speed, number of CPU cores, available memory, and network bandwidth). Several Forwarder Output configuration options are also available to help you tune performance for your throughput requirements and environment.

Create new Output ✕

Title

Select a name of your new output that describes it.

Hostname

The destination host name or IP address where the Graylog Forwarder input is running.

Port

13301

The destination port that the Graylog Forwarder input is listening on.

Journal Segment Size

100mb

The soft maximum for the size of a segment file in the log.

Journal Segment Age

1h

The disk journal segment age.

Maximum Journal Size

5gb

The maximum size for the disk journal.

Maximum Journal Message Age

12h

The maximum time that a message will be stored in the disk journal.

Journal Message Flush Interval

1000000

The number of messages that can be written to the log before a flush is forced.

Maximum Journal Flush Age

1m

The amount of time the log can have dirty data before a flush is forced.

Journal Buffer Size

1024

The size of the pre-journal buffer. This number must be a power of two.

Number of Handlers Journal Buffer Encoders

3

The number of concurrent journal encoders. This is a fast operation. A low number (3) is probably acceptable.

Send Buffer Size

1024

The size of the post-journal send buffer. This number must be a power of two.

Sender Encoders

3

The number of concurrent send encoders. This is a fast operation. A low number (3) is probably acceptable.

Concurrent Network Senders

1

The number of concurrent network senders. Use multiple senders to increase throughput. This number generally should not exceed the number of cores on a machine.

GRPC Request Timeout

10000

Request timeout for GRPC in milliseconds

Maximum Journal Read Batch Size

500

The maximum number of messages read from the journal at once.

☐ Enable Compression

The option to compress messages when they are transported

TLS Trusted Certificate Chain File (optional)

Path to the trusted certificate chain file for verifying the remote endpoint's certificate. The file should contain an X.509 certificate collection in PEM format.

☐ Enable TLS

Option to enable TLS.

Cancel   Save

Forwarder Input

The Forwarder Input (Graylog destination cluster) is responsible for receiving messages that have been forwarded from the Graylog cluster source.

When the Graylog destination cluster (Forwarder Input) receives the forwarded messages, the following relevant fields are added to help track which Graylog cluster and node the messages originated from.

- **`gl2_source_cluster_id`**

    – The id of the source Graylog cluster.

- **`gl2_source_node_id`**

    – The id of the source Graylog node.

Forwarder Input Options

Launch new *Forwarder* input ✕

☐ Global

Should this input start on all nodes

Node

Select Node ▾

On which node should this input start

Title

Select a name of your new input that describes it.

Bind Address

0.0.0.0

Address to listen on. For example 0.0.0.0 or 127.0.0.1.

Port

13301 ⬍

Port number to listen on

☐ Enable TLS

Option to enable TLS for connection

TLS Trusted Certificate Chain File (optional)

Path to the trusted certificate chain file. The file should contain an X.509 certificate collection in PEM format.

TLS Private Key File (optional)

Path to the TLS private key file. The file should be in PEM format

Cancel   Save

SSL/TLS

TLS encryption is supported to ensure secure transport of forwarded messages. You can enable it by checking the Enable TLS check box on both the Forwarder input and output. The Forwarder Input requires that both the certificate and key locations must be specified. The Forwarder Output requires only the certification location be specified.

Note: Only X.509 certificates and keys in PEM format are supported. TLS Authentication is not currently supported.

Load Balancing

The Forwarder uses HTTP/2 (gRPC) for transport. When only one Concurrent Network Sender is used, then load balancing is not supported. However, if more than one Concurrent Network Senders are used, then load balancing is supported, which allows each of these sender connections to be distributed to the destination host. For more information see Load Balancing gRPC.

TWENTYONE

PLUGINS

21.1 About Plugins

Graylog offers various extension points to customize and extend its functionality through writing Java code.

The first step for writing a plugin is creating a skeleton that is the same for each type of plugin. The next chapter is explaining how to do this and will then go over to chapters explaining plugin types in detail.

21.2 Plugin Types

Graylog comes with a stable plugin API for the following plugin types:

- **Inputs:** Accept/write any messages into Graylog
- **Outputs:** Forward ingested messages to other systems as they are processed
- **Services:** Run at startup and able to implement any functionality
- *Event Notifications*: Called when an event alert has been triggered
- **Processors:** Transform/drop incoming messages (can create multiple new messages)
- **Filters:** (Deprecated) Transform/drop incoming messages during processing
- **REST API Resources:** An HTTP resource exposed as part of the Graylog REST API
- **Periodical:** Called at periodical intervals during server runtime
- *Decorators*: Used during search time to modify the presentation of messages
- **Authentication Realms**: Allowing to implement different authentication mechanisms (like single sign-on or 2FA)

21.2.1 API concepts

Graylog uses certain patterns in its code bases to make it easier to write extensions. It is important to know about these to be successful in writing custom for it.

You can browse the Graylog Javadoc documentation for details on each class and method mentioned here.

Factory Class

Many newer Graylog extension points split the common aspects of custom code into three different classes:

- instance creation - an, usually inner, interface commonly called `Factory`
- configuration - the factory returns a `ConfigurationRequest` instance (or a wrapped instance of it), commonly called `Config`
- descriptor - the factory returns a display descriptor instance, commonly called `Descriptor`

Say Graylog exposes an extension point interface called `ExtensionPoint`, which contains inner interfaces calles `Factory`, `Config` and `Descriptor`. An implementation of `ExtensionPoint` then looks as following:

```java
public AwesomeExtension implements ExtensionPoint {

        public interface Factory extends ExtensionPoint.Factory {
                @Override
                AwesomeExtension create(Decorator decorator);

                @Override
                AwesomeExtension.Config getConfig();

                @Override
                AwesomeExtension.Descriptor getDescriptor();
        }

        public static class Config implements ExtensionPoint.Config {
                @Override
                public ConfigurationRequest getRequestedConfiguration() {
                        return new ConfigurationRequest();
                }
        }

        public static class Descriptor extends ExtensionPoint.Descriptor {
                public Descriptor() {
                        super("awesome", "http://docs.graylog.org/", "Awesome
→Extension");
                }
        }
}
```

This pattern is used to prevent instantiation of extensions just to get their descriptor or configuration information, because some extensions might be expensive to set up or require some external service and configuration to work.

The factory itself is built using Guice's assisted injection for auto-wired factories. This allows plugin authors (and Graylog's internals as well) to cleanly describe their extension as well as taking advantage of dependency injection.

To register such an extension, Graylog typically offers a convenience method via its Guice modules (`GraylogModule` or `PluginModule`). For example alert conditions follow the same pattern and are registered as such:

```java
public class SampleModule extends PluginModule {
        // other methods omitted for clarity
        @Override
        protected void configure() {
                addAlertCondition(SampleAlertCondition.class.getCanonicalName(),
                                SampleAlertCondition.class,
                                SampleAlertCondition.Factory.class);
```

```
        }
}
```

21.2.2 Event Notifications

Event Notifications are responsible for sending information about events to external systems, such as sending an email, push notifications, opening tickets, writing to chat systems, etc.

They receive the event they were triggered for as well as a context object with additional metadata. Each notification consists of two classes. A configuration class, keeping parameters for the notification, and a second class with the actual notification code.

Class Overview

The interfaces to implement are `org.graylog.events.notifications.EventNotificationConfig` and `org.graylog.events.notifications.EventNotification`. Both classes are used to register a new notification type via the `PluginModule` class:

```
addNotificationType(TYPE_NAME,
                    EventNotificationConfig.class,
                    EventNotification.class,
                    EventNotification.Factory.class);
```

Example Event Notification

You can use the built-in notifications as full examples one for a configuration class, and one for a notification class.

To create an event notification plugin start by implementing the `EventNotificationConfig` interface:

```
public abstract class HTTPEventNotificationConfig implements EventNotificationConfig
```

Every notification has a unique type name. Usually a string apended by a version number, e.g.:

```
public static final String TYPE_NAME = "http-notification-v1"
```

Continue with the parameters which are needed for the notification. For example for a HTTP notification an URL is needed:

```
private static final String FIELD_URL = "url";

@JsonProperty(FIELD_URL)
public abstract String url();
```

A Builder for the configuration class is needed:

```
public static Builder builder() {
    return Builder.create();
}

@AutoValue.Builder
public static abstract class Builder implements EventNotificationConfig.Builder
↪<Builder> {
```

```
  @JsonCreator
  public static Builder create() {
    return new AutoValue_HTTPEventNotificationConfig.Builder()
                                                 .type(TYPE_NAME);
    }


    @JsonProperty(FIELD_URL)
    public abstract Builder url(String url);


    public abstract HTTPEventNotificationConfig build();
}
```

Additionally interface methods are to implement:

public JobTriggerData toJobTriggerData(EventDto dto)

This is called when the scheduler executes the notification and is gathering data for the job.

public ValidationResult validate()

On the API level, inputs should be validated, put the checks here.

public EventNotificationConfigEntity toContentPackEntity(EntityDescriptorIds entityDescriptorIds)

If content-pack support is desired this method is called to transfer a notification to a content pack entity.

Take a look at the examples for more details on this.

After creating the configuration class the actual notification code can be written. This is done by implementing the `EventNotification` interface in a new class:

```
public class HTTPEventNotification implements EventNotification
```

There is just a single method to override. The notification logic goes into `execute(EventNotificationContext ctx)`.

The context object is keeping data about the event that lead to the notification, see this class for more informations.

For getting a backlog of messages which were the source of the event itself a service is provided:

```
ImmutableList<MessageSummary> backlog = notificationCallbackService.
→getBacklogForEvent(ctx)
```

User Interface

Event Notifications need to provide some UI components that let the user enter Notification details in a form, and also display a summary of the Notification in the Event Definition summary.

First of all, Event Notifications need to register a plugin for `eventNotificationTypes`. As an example, we will show the definition of the HTTP Notification type:

```
eventNotificationTypes: [
  {
    type: 'http-notification-v1',
    displayName: 'HTTP Notification',
    formComponent: HttpNotificationForm,
    summaryComponent: HttpNotificationSummary,
    defaultConfig: {
        url: '',
```

```
      },
   }
]
```

Here is a description of each field the Notification needs to provide:

- `type` Unique type identifying the Notification. You should use the same type as used in the server class
- `displayName` Human readable short name that describes the Notification. It will be used in select inputs
- `formComponent` React component providing the form elements the user should fill in order to create the Event Notification
- `defaultConfig` Object including the default configuration that is used once the Notification type is selected. We recommend defining the default configuration in your `formComponent` and then simply add a reference to it here
- `summaryComponent` React component displaying a summary of the Event Notification

In order to help you write the required React components, we now describe what props they will receive and what is expected from the given components.

Form component

This component should present inputs that need to be filled out in order to configure the Event Notification. The component will receive the following props:

```
config: PropTypes.object
onChange: PropTypes.func
validation: PropTypes.object
```

- `config` Contains the current configuration the user gave for the Notification. This will be set by default to the object given as `defaultConfig` in the plugin definition
- `onChange` Function to call when an input changes. The function expects to receive the complete configuration object as first argument. Please remember you should not directly modify the `config` prop, but instead clone the object first and apply modifications there
- `validation` Contains an object with validation information. The object has the following structure:

```
validation: {
  errors: {
    url: [
      "HTTP Notification URL cannot be empty."
    ],
  },
  failed: true
}
```

With that, once the user is done configuring the Event Notification, Graylog will submit the defined configuration into the server and create the Event Notification.

Summary component

This component should render a summary of all options configured in the Event Notification. It will be displayed in the summary step of the Event Definition form. The component receives the following props:

```
type: PropTypes.string,
notification: PropTypes.object,
definitionNotification: PropTypes.object,
```

- `type` Contains the `displayName` property defined in the Notification plugin

- `notification` Contains the Notification object, including its configuration

- `definitionNotification` Contains the `notification_id` used by the Event Definition. This is only required in case the Notification plugin is not installed or was deleted

In order to follow the same style as other Notifications in the summary component, we highly recommend using the `CommonNotificationSummary` component to render all common properties all Notifications have: title, type, description, and the children you pass. The component receives the same props as this one, plus the children you want to render with the custom plugin configuration. As an example, this is how the HttpNotificationSummary renders its summary:

```
<CommonNotificationSummary {...this.props}>
  <React.Fragment>
    <tr>
      <td>URL</td>
      <td><code>{notification.config.url}</code></td>
    </tr>
  </React.Fragment>
</CommonNotificationSummary>
```

21.2.3 Decorators

Decorators can be used to transform a message field at display time. Multiple decorators can be applied at the same time, but you cannot make any assumptions about their order, as that is user defined. Stacked decorators receive the value of the previous decorator results.

They are typically used to map between the stored value and a human readable form of that value, for example like the *Syslog severity mapper* (compare its code) maps between numeric values and their textual representation.

Other uses include looking up user names based on a user's ID in a remote database, triggering a `whois` request on a domain name etc.

Class Overview

You need to implement the `org.graylog2.plugin.decorators.SearchResponseDecorator` interface. This class must declare a *Factory Class*.

Beyond the factory, configuration and descriptor classes, the only thing that a decorator needs to implement is the `apply` function:

```
SearchResponse apply(SearchResponse searchResponse);
```

The `org.graylog2.rest.resources.search.responses.SearchResponse` class represents the result that is being returned to the web interface (or other callers of the REST API).

You are free to modify any field, create new fields or remove fields. However, the web interface makes certain assumptions regarding fields that start with gl2_ and requires at least the timestamp, source and message fields to be present.

Thrown exceptions are being logged as errors and lead to returning the original search response, without any modifications.

Example

Please refer to the sample plugin implementation for the full code.

Bindings

Compare with the code in the sample plugin.

```java
public class SampleModule extends PluginModule {

  @Override
  public Set<? extends PluginConfigBean> getConfigBeans() {
      return Collections.emptySet();
  }

  @Override
  protected void configure() {
    installSearchResponseDecorator(searchResponseDecoratorBinder(),
                    PipelineProcessorMessageDecorator.class,
                    PipelineProcessorMessageDecorator.Factory.class);
  }
}
```

User Interface

Decorators have no custom user interface elements.

21.3 Writing Plugins

What you need in your development environment before starting is:

- git

- maven

- rpm

If you plan to write a web plugin, you'll also need:

- node

- yarn

There are lots of different ways to get those on your local machine, unfortunately we cannot list all of them, so please refer to your operating system-specific documentation,

Graylog uses a couple of conventions and techniques in its code, so be sure to read about the *API concepts* for an overview.

21.3.1 Sample Plugin

In the following sections we will create a plugin skeleton based on a maven archetype. The skeleton is similar to the sample plugin on Github. This documentation will link to specific parts for your reference. It is fully functional, even though it does not implement any useful functionality. Its purpose is to provide a reference for helping to implement your own plugins.

21.3.2 Creating a plugin skeleton

The easiest way to get started is to use our Graylog meta project, which will create a complete plugin project infrastructure with all required classes, build definitions, and configurations. Using the meta project allows you to have the Graylog server project and your own plugins (or 3rd party plugins) in the same project, which means that you can run and debug everything in your favorite IDE or navigate seamlessly in the code base.

Note: We are working on a replacement tool for the `graylog-project` meta project, but for the time being it still works.

Maven is a widely used build tool for Java, that comes pre-installed on many operating systems or can be installed using most package managers. Make sure that you have at least version 3 before you go on.

Use it like this:

```
$ git clone git@github.com:Graylog2/graylog-project.git
```

This will create a checkout of the meta project in your current work dir. Now change to the `graylog-project` directory and follow the `Install CLI Tool` and `Bootstrap` from its README. Bootstrapping the `garylog-project` will create the directory `graylog-project-repos`. It contains all required repositories and will also be the home of your new plugin.

Now you can bootstrap the plugin you want to write, by running the following command, inside the `graylog-project-repos` directory:

```
$ mvn archetype:generate -DarchetypeGroupId=org.graylog -DarchetypeArtifactId=graylog-
↪plugin-archetype
```

It will ask you a few questions about the plugin you are planning to build. Let's say you work for a company called ACMECorp and want to build an alarm callback plugin that creates a JIRA ticket for each alarm that is triggered:

```
groupId: com.acmecorp
artifactId: graylog-plugin-jira-alarmcallback
version: 1.0.0
package: com.acmecorp
githubRepo: exampleGithubRepo
pluginClassName: JiraAlarmCallback
```

Note that you do not have to tell the archetype wizard what kind of plugin you want to build because it is creating the generic plugin skeleton for you, and nothing that is related to the actual implementation. More on this in the example plugin chapters later. It is important that your `artifactId` has the prefix `graylog-plugin-`. The `githubRepo` must be the desired plugins repo name, not the full github URL. The repository is not required for the development, but a common part of the plugins meta information.

You now have a new folder called `graylog-plugin-jira-alarmcallback`, which includes a complete plugin skeleton including Maven build files. To be able to make a complete build of the project, you need to add the newly created plugin to the graylog-project POM as a module. Open `pom.xml` (residing in your `graylog-project`

directory) and find a couple of `<module>` statements in the file. Add the following line (after adapting it to your naming):

> <module>../graylog-project-repos/graylog-plugin-jira-alarmcallback</module>

Make sure to update the `graylog-plugin-web-parent` version inside the plugins `pom.xml`. You can find the current version inside the related `relativePath` property. The last necessary step, to get started with the development, is to execute `mvn compile` inside the `graylog-project` dir.

You should be finished now, and every Java IDE out there can now import the project automatically without any required further configuration.

In IntelliJ IDEA for example you can just use the *File -> Open* dialog to open the `graylog-project` directory as a fully configured Java project, which should include the Graylog server and your plugin as submodules.

Please pay close attention to the README file of the Graylog meta project and follow any further instructions listed there to set up your IDE properly.

If you want to continue working on the command line, you can do the following to compile the server and your plugin:

```
$ mvn package
```

21.3.3 The anatomy of a plugin

Each plugin contains information to describe itself and register the extensions it contains.

Note: A single plugin can contain multiple extensions to Graylog.

For example a hypothetical plugin might contribute an input, an output and alert notifications to communicate with systems. For convenience this would be bundled in a single plugin registering multiple extensions.

Required classes

At the very minimum you need to implement two interfaces:

- `org.graylog2.plugin.Plugin` - which is the entry to your plugin code
- `org.graylog2.plugin.PluginMetaData` - which describes your plugin

The `bootstrap-plugin` script generates these implementations for you, and you simply need to fill out the details.

Graylog uses Java's ServiceLoader mechanism to find your plugin's main class, so if you rename your `Plugin` implementation, you need to also adjust the service file. Please also see Google Guava's AutoService which Graylog uses in conjunction with the plain ServiceLoader.

In addition to the service, Graylog needs an additional resource file called `graylog-plugin.properties` in a special location. This file contains information about the plugin, specifically which classloader the plugin needs to be in, so it needs to be read before the plugin is actually loaded. Typically you can simply take the default that has been generated for you.

Registering your extension

So far the plugin itself does not do anything, because it neither implements any of the available extensions, nor could Graylog know which ones are available from your code.

Graylog uses dependency injection to wire up its internal components as well as the plugins. Thus the extensions a plugin provides need to be exposed as a PluginModule which provides you with a lot of helper methods to register the various available extensions to cut down the boiler plate code you have to write.

An empty module is created for you.

> **Caution:** The `PluginModule` exposes a lot of extension points, but not all of them are considered stable API for external use.
>
> If in doubt, please reach out to us on our community support channels.

Please refer to the available *Plugin Types* for detailed information what you can implement. The *Sample Plugin* contains stub implementations for each of the supported extensions.

Web Plugin creation

Sometimes your plugin is not only supposed to work under the hoods inside a Graylog server as an input, output, alarm callback, etc. but you also want to contribute previously nonexisting functionality to Graylog's web interface. Since version 2.0 this is now possible. When using the most recent Graylog meta project to bootstrap the plugin skeleton, you are already good to go for this. Otherwise please see our chapter about *Creating a plugin skeleton*.

The Graylog web interface is written in JavaScript, based on React. It is built using webpack, which is bundling all JavaScript code (and other files you use, like stylesheets, fonts, images, even audio or video files if you need them) into chunks digestible by your browser and npm, which is managing our external (and own) dependencies. During the build process all of this will be bundled and included in the jar file of your plugin.

This might be overwhelming at first if you are not accustomed to JS-development, but fortunately we have set up a lot to make writing plugins easier for you!

If you use our proposed way for *Creating a plugin skeleton*, and followed the part about the *Writing Plugins*, you are already good to go for building a plugin with a web part. **All you need is a running Graylog server on your machine.** Everything else is fetched at build time!

Getting up and running with a web development environment is as easy as this (assuming you have node & yarn installed):

```
$ cd graylog2-server/graylog2-web-interface
$ yarn install
[...]
$ yarn start
[...]
$ open http://localhost:8080
```

This starts the development web server. It even tries to open a browser window going to it (probably working on Mac OS X only).

If your Graylog server is not running on `http://localhost:9000/api/`, then you need to edit `graylog2-server/graylog2-web-interface/config.js` (in your `graylog-project` directory) and adapt the `gl2ServerUrl` parameter.

Web Plugin structure

These are the relevant files and directories in your plugin directory for the web part of it:

webpack.config.js This is the configuration file for the webpack module bundler. Most of it is already preconfigured by our `PluginWebpackConfig` class, so the file is very small. You can override/extend every configuration option by passing a webpack snippet though.

build.config.js.sample In this file you can customize some of the parameters of the build. There is one mandatory parameter named `web_src_path` which defines the absolute or relative location to a checkout of the Graylog source repository.

package.json This is a standard npm JSON file describing the web part of your plugin, especially its dependencies. You can read more about its format.

src/web This is where the actual code for thw web part of your plugin goes to. For the start there is a simple `index.jsx` file, which shows you how to register your plugin and the parts it provides with the Graylog web interface. We will get to this in detail later.

21.3.4 Required conventions for web plugins

Plugin Entrypoint

There is a single file which is the entry point of your plugin, which means that the execution of your plugin starts there. By convention this is *src/web/index.jsx*. You can rename/move this file, you just have to adapt your webpack configuration to reflect this change, but it is not recommended.

In any case, this file needs to contain the following code at the very top:

```
// eslint-disable-next-line no-unused-vars
import webpackEntry from 'webpack-entry';
```

This part is responsible to include and execute the webpack-entry file, which is responsible to set up webpack to use the correct URL format when loading assets for this plugin. If you leave this out, erratic behavior will be the result.

Linking to other pages from your plugin

If you want to generate links from the web frontend to other pages of your plugin or the main web interface, you need to use the `Routes.pluginRoute()` helper method to generate the URLs properly.

See this file for more information.

21.3.5 Best practices for web plugin development

Using ESLint

ESLint is an awesome tool for linting JavaScript code. It makes sure that any written code is in line with general best practises and the project-specific coding style/guideline. We at Graylog are striving to make the best use of this tools as possible, to help our developers and you to generate top quality code with little bugs. Therefore we highly recommend to enable it for a Graylog plugin you are writing.

Code Splitting

Both the web interface and plugins for it depend on a number of libraries like React, RefluxJS and others. To prevent those getting bundled into *both* the web interface *and* plugin assets, therefore wasting space or causing problems (especially React does not like to be present more than once), we extract those into a commons chunk which is reused by the web interface and plugins.

This has no consequences for you as a plugin author, because the configuration to make use of this is already generated for you when using the meta project or the maven archetype. But here are some details about it:

Common libraries are built into a separate `vendor` bundle using an own configuration file named webpack.vendor.js. Using the DLLPlugin a manifest is extracted which allow us to reuse the generated bundle. This is then imported in our main web interface webpack configuration file and the corresponding generated webpack config file for plugins.

21.3.6 Building plugins

Building the plugin is easy because the meta project has created all necessary files and settings for you. Just run `mvn package` either from the meta project's directory `graylog-project` (to build the server *and* the plugin) or from the plugin directory (to build the plugin only):

```
$ mvn package
```

This will generate a `.jar` file in `target/` that is the complete plugin file:

```
$ ls target/jira-alarmcallback-1.0.0-SNAPSHOT.jar
target/jira-alarmcallback-1.0.0-SNAPSHOT.jar
```

21.4 Installing and loading plugins

The only thing you need to do to run the plugin in Graylog is to copy the `.jar` file to your plugins folder that is configured in your `graylog.conf`. The default is just `plugin/` relative from your `graylog-server` directory.

This is a list of default plugin locations for the different installation methods.

Table 1: Plugin Installation Locations

Installation Method	Directory
Operating System Packages	`/usr/share/graylog-server/plugin/`
Manual Setup	`/<extracted-graylog-tarball-path>/plugin/`

Restart `graylog-server` and the plugin should be available to use from the web interface immediately.

CONTENT PACKS

22.1 What are content packs?

Content packs are a convenient way to share configuration. A content pack is a JSON file which contains a set of configurations of Graylog components. This JSON file can be uploaded to Graylog instances and then installed. A user who took the time to create a input, pipelines and dashboard for a certain type of log format, can so easily share his efforts with the community.

Content packs can be found on the Graylog Marketplace.

> **Warning:** Content packs in 3.0 have changed fundamentally from previous versions. Graylog will try to support older versions in the future, but at this point there is no guarantee that older content packs still work.

22.1.1 Parameter

Content packs can have parameters. Those parameters help to adjust the configuration to the needs of the user. A good example usage is the port of an input. The creator of the content pack may have his input running on port 55055, but the user of the content pack may already have an input running on that specific port. The creator can specify a parameter and assign it to the port. The user of the content pack will be asked for a value of the parameter on installation. The provided value will then be used as the port of the input on the new system.

22.2 How do I create a Content Pack?

1. Navigate to **System / Content Packs**.

2. Click on **Create a Content Pack** on the upper right side of the page. A new content pack is created in a wizard. On the left side of that page is the navigation of the wizard. There are 3 steps to content pack creation: **Content Selection**, **Parameters** and **Preview**. In the middle part of the page is the form of each wizard step. And on the right side is the summary of the content pack in creation.

3. Fill out the general information of the content pack.

4. The **Content Selection** offers configurations which can be included to the content pack. Necessary dependencies will be included automatically.

> **Warning:** The one exception to this rule is dependencies for pipeline rules. Currently, grok patterns and lookup tables for pipeline rules must be added manually to the content pack. Support for automatic inclusion of pipeline rule dependencies will be added in a future release.

5. Click on **Next** or **Parameter** to go the Parameter Page. Parameter are placeholders which will be filled out during installation of a content pack. That way, parts of the configuration may be adjusted according to the needs of the user. To create a parameter click on **Create Parameter**. In the opening modal can the name, type and default value of the parameter be specified. Graylog supports four types of configuration values: String, Integer, Double and Boolean.

6. The created parameter can be assigned to a configuration key by pressing **Edit** on one of the previously selected configurations under **Entity List**.

7. The final step of creation can be reached by clicking on **Next** or **Preview**. On the preview page displays a summary of the new content pack. This page is meant for a final close inspection of the content pack before creation.

8. To finish the creation click on **Create** or **Create and Download**.

22.3 Upload a content pack

Content packs may be downloaded at the Graylog Marketplace. To upload these content packs navigate to **System / Content Packs** and click on **Upload**. The now showing modal has a file finder to select the downloaded content pack. Click on **Upload** to finish the process. The uploaded content pack may now be installed on the new Graylog system.

22.4 Installing a content pack

To install the newest version of a content pack, navigate to **System / Content Packs**. This page shows the list of uploaded and created content packs. By clicking **Install** on the desired content pack, a modal will open which will ask for a **Install Comment** and the values of the parameters. It also shows the list of configurations that will be installed on the system. Click on **Install** to complete the installation.

Note: Some entities need a unique title or name (e.g Lookup Table). When installing such an entity and the title is already present on the system, then Graylog will use the installed entity instead of installing a new one. Even when the new configuration differs from the already installed one.

22.5 Uninstalling a content pack

Navigate to **System / Content Packs** and click on the name of the content pack that should be uninstalled. The displayed page shows the details of a uploaded or created content pack.

On the left, select the version of the content pack. Below that is a list of installations of that content pack. Click **Uninstall** next to the desired installation. A list of entities about to be removed will be displayed.

TWENTYTHREE

EXTERNAL DASHBOARDS

There are other frontends that are connecting to the Graylog REST API and display data or information in a special way.

23.1 CLI stream dashboard

This official Graylog dashboard which is developed by us is showing live information of a specific stream in your terminal. For example it is the perfect companion during a deployment of your platform: Run it next to the deployment output and show information of a stream that is catching all errors or exceptions on your systems.
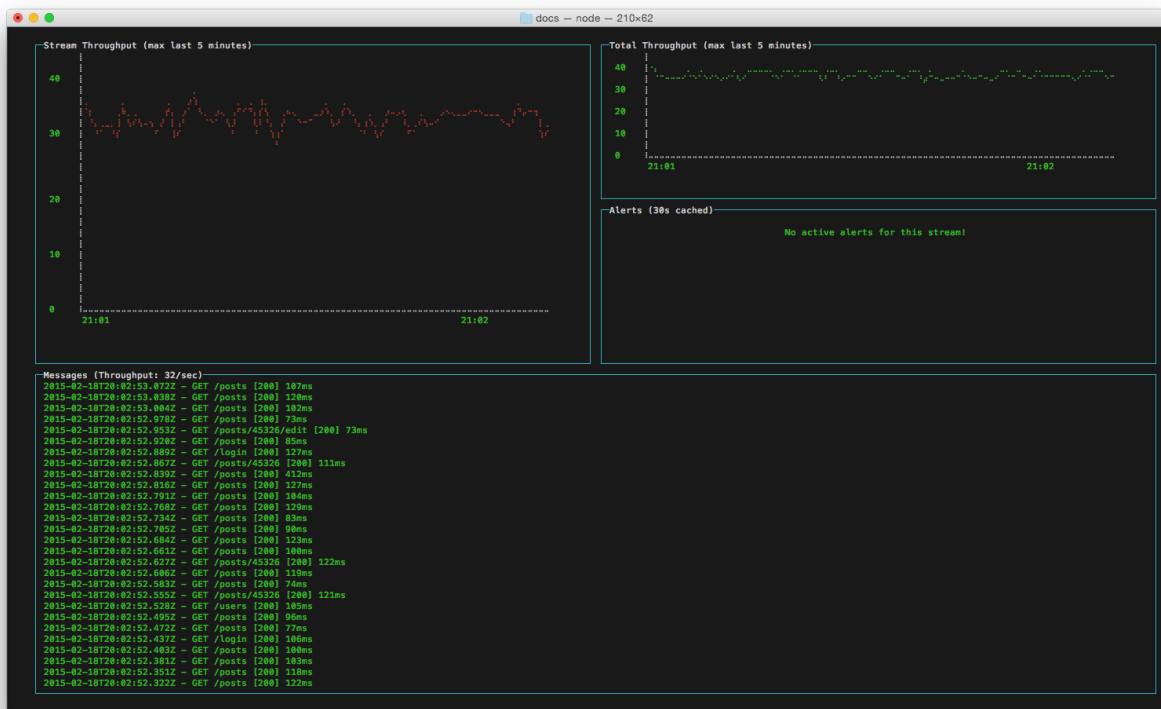
The CLI stream dashboard documentation is available on GitHub.

TWENTYFOUR

GRAYLOG MARKETPLACE

The Graylog Marketplace is the central directory of add-ons for Graylog. It contains plugins, content packs, GELF libraries and more content built by Graylog developers and community members.
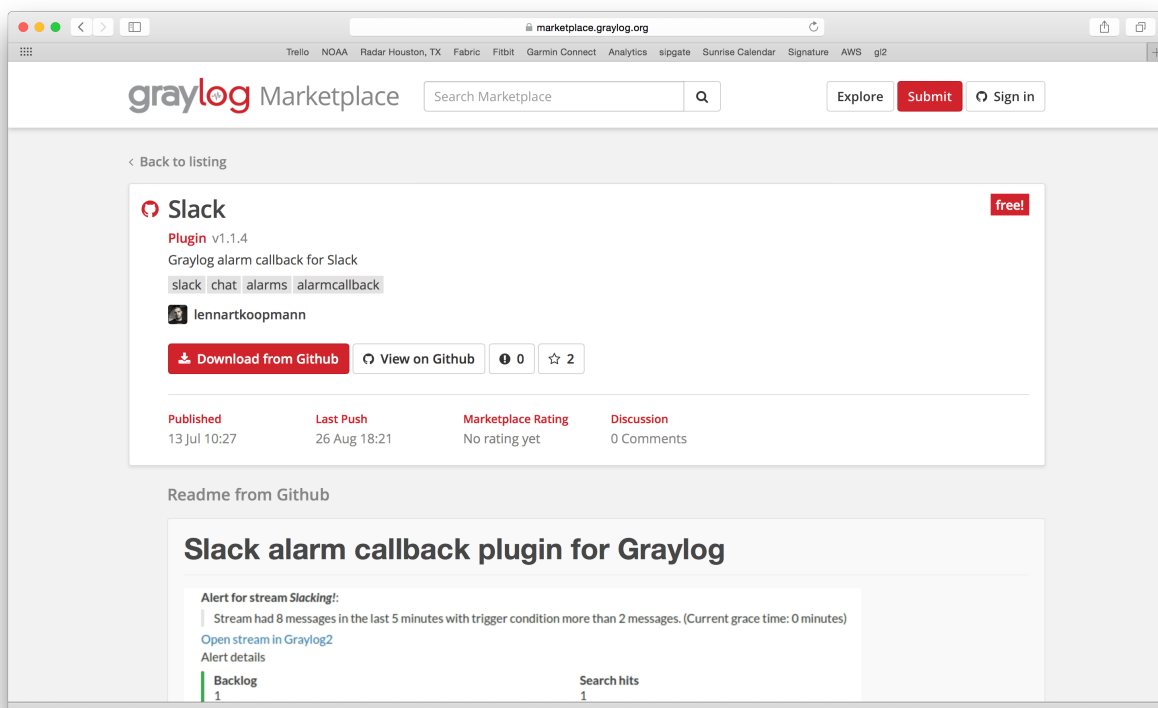
24.1 GitHub integration

The Marketplace is deeply integrated with GitHub. You sign-in with your GitHub account if you want to submit content and only have to select an existing repository to list on the Marketplace.

From there on you manage your releases and code changes in GitHub. The Marketplace will automatically update your content.

There is no need to sign-in if you only want to browse or download content.

24.2 General best practices

24.2.1 README content

We kindly ask you to provide an as descriptive as possible `README` file with your submission. This file will be displayed on the Marketplace detail page and should provide the following information:

- What is it.

- Why would you want to use it? (Use cases)

- Do you have to register somewhere to get for example an API token?

- How to install and configure it.

- How to use it in a Graylog context.

Take a look at the Splunk plug-in as an example.

The README supports Markdown for formatting. You cannot submit content that does not contain a `README` file.

24.2.2 License

You cannot submit content that does not contain a `LICENSE` or `COPYING` file. We recommend to consult ChooseAL-icense.com if you are unsure which license to use.

24.3 4 Types of Add-Ons

Plug-Ins: Code that extends Graylog to support a specific use case that it doesn't support out of the box.

Content Pack: A file that can be uploaded into your Graylog system that sets up streams, inputs, extractors, dashboards, etc. to support a given log source or use case.

GELF Library: A library for a programming language or logging framework that supports sending log messages in GELF format for easy integration and pre-structured messages.

Other Solutions: Any other content or guide that helps you integrate Graylog with an external system or device. For example, how to configure a specific device to support a format Graylog understands out of the box.

24.4 Contributing plug-ins

You *created a Graylog plugin* and want to list it in the Marketplace? This is great. Here are the simple steps to follow:

1. Create a GitHub repository for your plugin

2. Include a *README* and a *LICENSE* file in the repository.

3. Push all your code to the repository.

4. Create a GitHub release and give it the name of the plugin version. For example `0.1`. The Marketplace will always show and link the latest version. You can upload as many release artifacts as you want here. For example the `.jar` file together with `DEB` and `RPM` files. The Marketplace will link to the detail page of a release for downloads.

5. Submit the repository to the Marketplace

24.5 Contributing content packs

Graylog content packs can be shared on the Marketplace by following these steps:

1. Download a Graylog content pack from the Graylog Web Interface and save the generated JSON in a file called `content_pack.json`.

2. Create a GitHub repository for your content pack

3. Include a *README* and a *LICENSE* file in the repository.

4. Include the `content_pack.json` file in the root of your GitHub repository.

5. Submit the repository to the Marketplace

24.6 Contributing GELF libraries

A GELF library can be added like this:

1. Create a GitHub repository for your GELF library.

2. Include a *README* and a *LICENSE* file in the repository.

3. Describe where to download and how to use the GELF library in the `README`.

24.7 Contributing other content

You want to contribute content that does not really fit into the other categories but describes how to integrate a certain system or make it send messages to Graylog?

This is how you can do it:

1. Create a GitHub repository for your content

2. Include a *README* and a *LICENSE* file in the repository.

3. All content goes into the `README`.

FREQUENTLY ASKED QUESTIONS

25.1 General

25.1.1 Do I need to buy a license to use Graylog?

We believe software should be open and accessible to all. You should not have to pay to analyze your own data, no matter how much you have.

Graylog is licensed under the GNU General Public License. We do not require license fees for production or non-production use.

25.1.2 How long do you support older versions of the Graylog product?

For our commercial support customers, we support older versions of Graylog up to 12 months after the next major release is available. So if you're using 1.X, you will continue to receive 1.X support up to a full year after 2.0 has been released.

25.2 Architecture

25.2.1 What is MongoDB used for?

Graylog uses MongoDB to store your configuration data, not your log data. Only metadata is stored, such as user information or stream configurations. None of your log messages are ever stored in MongoDB. This is why MongoDB does not have a big system impact, and you won't have to worry too much about scaling it. With our recommended setup architecture, MongoDB will simply run alongside your graylog-server processes and use almost no resources.

25.2.2 Can you guide me on how to replicate MongoDB for High Availability?

MongoDB actually supplies this information as part of their documentation. Check out :

- About MongoDB Replica Sets.
- How to convert a standalone MongoDB node to a replica set.

After you've done this, add all MongoDB nodes into the replica_set configuration in all graylog-server.conf files.

25.2.3 I have datacenters across the world and do not want logs forwarding from everywhere to a central location due to bandwidth, etc. How do I handle this?

You can have multiple graylog-server instances in a federated structure, and forward select messages to a centralized GL server.

25.2.4 Which load balancers do you recommend we use with Graylog?

You can use any. We have clients running AWS ELB, HAProxy, F5 BIG-IP, and KEMP.

25.2.5 Isn't Java slow? Does it need a lot of memory?

This is a concern that we hear from time to time. We understand Java has a bad reputation from slow and laggy desktop/GUI applications that eat a lot of memory. However, we are usually able to prove this assumption wrong. Well written Java code for server systems is very efficient and does not need a lot of memory resources.

Give it a try, you might be surprised!

25.2.6 Does Graylog encrypt log data?

All log data is stored in Elasticsearch. Elastic recommends you use *dm-crypt* at the file system level.

25.2.7 Where are the log files Graylog produces?

You can find the log data for Graylog under the below directory with timestamps and levels and exception messages. This is useful for debugging or when the server won't start.

> /var/log/graylog-server/server.log

If you use the pre-build appliances, take a look into

> /var/log/graylog/<servicename>/current

25.3 Installation / Setup

25.3.1 Should I download the OVA appliances or the separate packages?

If you are downloading Graylog for the first time to evaluate it, go for the appliance. It is really easy, and can be quickly setup so you can understand if Graylog is right for you. If you are wanting to use Graylog at some scale in production, and do things like high availability (Mongo replication) we recommend you go for the separate packages.

25.3.2 How do I find out if a specific log source is supported?

We support many log sources – and more are coming everyday. For a complete list, check out Graylog Marketplace, the central repository of Graylog extensions. There are 4 types of content on the Marketplace:

- Plug-Ins: Code that extends Graylog to support a specific use case that it doesn't support out of the box.

- Content Pack: A file that can be uploaded into your Graylog system that sets up streams, inputs, extractors, dashboards, etc. to support a given log source or use case.

- GELF Library: A library for a programming language or logging framework that supports sending log messages in GELF format for easy integration and pre-structured messages.

- Other Solutions: Any other content or guide that helps you integrate Graylog with an external system or device. For example, how to configure a specific device to support a format Graylog understands out of the box.

25.3.3 Can I install the Graylog Server on Windows?

Even though our engineers say it is "technically possible", don't do it. The Graylog server is built using Java, so technically it can run anywhere. But we currently have it optimized to run better on other operating systems. If you don't feel comfortable running your own Linux system, we recommend you use our Linux virtual appliance which will run under VMWare.

25.3.4 Can I run Graylog on Azure?

You can create a Linux VM and use our *step-by-step* to install your customized Graylog. As a second option you can use this guide to convert our *Appliance* into some Azure compatible virtual machine.

25.4 Functionality

25.4.1 Can Graylog automatically clean old data?

Absolutely we have *data retention features*.

25.4.2 Does Graylog support LDAP / AD and its groups?

Yup, we're all over this too with read/write roles and group permissions. To start, see *this*. If you want to get very granular, you can go through the Graylog REST API.

25.4.3 Do we have a user audit log for compliance?

Graylog Enterprise includes audit log functionality. You can explore the *documentation* for more details.

25.4.4 Does Graylog have reporting functionality?

Since Graylog 3.0, Graylog Enterprise includes reporting capabilities. Take a look at the *documentation* for more details.

25.4.5 Can I filter inbound messages before they are processed by the Graylog server?

Yes, check out our page on how to use blacklisting.

25.4.6 Dedicated Partition for the Journal

If you create a dedicated Partition for your Kafka Journal, you need to watch that this is a clean directory. Even *lost+found* can break it, for your reference.

25.4.7 Raise the Java Heap

On Systems that run as virtual appliances or are installed with *DEB / APT* this setting can be made in `/etc/default/graylog-server`.

Systems that are installed with *RPM / YUM / DNF* the file is found in `/etc/sysconfig/graylog-server`.

25.4.8 How can I start an input on a port below 1024?

If you try to start an input on one of the privileged ports , it will only work for the "root" user. To be able to use a privileged port, you can use authbind on Debian-based systems, or you redirect the traffic with an `iptables` rule like this:

```
iptables -t nat -A PREROUTING -p tcp --dport 514 -j REDIRECT --to 1514
iptables -t nat -A PREROUTING -p udp --dport 514 -j REDIRECT --to 1514
```

The input needs to be started on port 1514 in this case and will be made available on port 514 to the outside. The clients can then send data to port 514.

25.5 Graylog & Integrations

25.5.1 What is the best way to integrate my applications to Graylog?

We recommend that you use *GELF*. It's easy for your application developers and eliminates the need to store the messages locally. Also, GELF can just send what app person wants so you don't have to build extractors or do any extra processing in Graylog.

25.5.2 I have a log source that creates dynamic syslog messages based on events and subtypes and grok patterns are difficult to use - what is the best way to handle this?

Not a problem! Use our *key=value extractor*.

25.5.3 I want to archive my log data. Can I write to another database, for example HDFS / Hadoop, from Graylog?

Yes, you can output data from Graylog to a different database. We currently have an HDFS output plug-in in the Marketplace - thank you sivasamyk!

It's also easy and fun to *write your own*, which you can then add to Graylog Marketplace for others to use.

25.5.4 I don't want to use Elasticsearch as my backend storage system – can I use another database, like MySQL, Oracle, etc?

Currently you can't. We only recommend another database if you want it for secondary storage.

25.5.5 How can I create a restricted user to check internal Graylog metrics in my monitoring system?

You can create a restricted user which only has access to the `/system/metrics` resource on the Graylog REST API. This way it will be possible to integrate the internal metrics of Graylog into your monitoring system. Giving the user only restricted access will minimize the impact of these credentials getting compromised.

Send a POST request via the Graylog API Browser or curl to the `/roles` resource of the Graylog REST API:

```
{
  "name": "Metrics Access",
  "description": "Provides read access to all system metrics",
  "permissions": ["metrics:*"],
  "read_only": false
}
```

The following curl command will create the required role (modify the URL of the Graylog REST API, here `http://127.0.0.1:9000/api/`, and the user credentials, here `admin`/`admin`, according to your setup):

```
$ curl -u admin:admin -H "Content-Type: application/json" -H 'X-Requested-By: cli' -X
↪POST -d '{"name": "Metrics Access", "description": "Provides read access to all
↪system metrics", "permissions": ["metrics:*"], "read_only": false}' 'http://127.0.0.
↪1:9000/api/roles'
```

25.6 Troubleshooting

25.6.1 I'm sending in messages, and I can see they are being accepted by Graylog, but I can't see them in the search. What is going wrong?

A common reason for this issue is that the timestamp in the message is wrong. First, confirm that the message was received by selecting 'all messages' as the time range for your search. Then identify and fix the source that is sending the wrong timestamp.

25.6.2 I have configured an SMTP server or an output with TLS connection and receive handshake errors. What should I do?

Outbound TLS connections have CA (*certification authority*) certificate verification enabled by default. In case the target server's certificate is not signed by a CA found from trust store, the connection will fail. A typical symptom for this is the following error message in the server logs:

```
Caused by: javax.mail.MessagingException: Could not convert socket to TLS; nested
→exception is: javax.net.ssl.SSLHandshakeException: sun.security.validator.
→ValidatorException: PKIX path building failed: sun.security.provider.certpath.
→SunCertPathBuilderException: unable to find valid certification path to requested
→target
```

This should be corrected by either adding the missing CA certificates to the Java default trust store (typically found at `$JAVA_HOME/jre/lib/security/cacerts`), or a custom store that is configured (by using `-Djavax.net.ssl.trustStore`) for the Graylog server process. The same procedure applies for both missing valid CAs and self-signed certificates.

For Debian/Ubuntu-based systems using OpenJDK JRE, CA certificates may be added to the systemwide trust store. After installing the JRE (including `ca-certificates-java`, ergo `ca-certificates` packages), place `name-of-certificate-dot-crt` (in PEM format) into `/usr/local/share/ca-certificates/` and run `/usr/sbin/update-ca-certificates`. The hook script in `/etc/ca-certificates/update.d/` should automatically generate `/etc/ssl/certs/java/cacerts`.

Fedora/RHEL-based systems may refer to Shared System Certificates in the Fedora Project Wiki.

25.6.3 Suddenly parts of Graylog did not work as expected

If you notice multiple different non working parts in Graylog and found something like `java.lang.OutOfMemoryError: unable to create new native thread` in your Graylog Server logfile, you need to raise the process/thread limit of the graylog user. The limit can be checked with `ulimit -u` and you need to check how you can raise `nproc` in your OS.

25.6.4 I cannot go past page 66 in search results

Elasticsearch limits the number of messages per search result to 10000 by default. Graylog displays 150 messages per page, which means that the last full page with default settings will be page 66.

You can increase the maximum result window by adjusting the parameter `index.max_result_window` as described in the Elasticsearch index modules dynamic settings, but be careful as this requires more memory in your Elasticsearch nodes for deep pagination.

This setting can be dynamically updated in Elasticsearch, so that it does not require a cluster restart to be effective.

25.6.5 My field names contain dots and stream alerts do not match anymore

Due to restrictions in certain Elasticsearch versions, Graylog needs to convert field names that contain `.` characters with another character, by default the replacement character is `_`.

This replacement is done just prior to writing messages to Elasticsearch, which causes a mismatch between what stream rules and alert conditions see as field names when they are evaluated.

Stream rules, the conditions that determine whether or not a message is routed to a stream, are being run as data is being processed by Graylog. These see the field names as containing the dots.

However, alert conditions, which are also attached to streams, are converted to searches and run in the background. They operate on stored data in Elasticsearch and thus see the replacement character for the dots. Thus alert conditions need to use the `_` instead of `.` when referring to fields. There is currently no way to maintain backwards compatibility and transparently fixing this issue, so you need to take action.

The best option, apart from not sending fields with dots, is to remember to write alert conditions using the replacement character, and never use `.` in the field names. In general Graylog will use the version with `_` in searches etc.

For example, if an incoming message contains the field `docker.container` stream rules use that name, whereas alert conditions need to use `docker_container`. You will notice that the search results also use the latter name.

25.6.6 What does "Uncommited messages deleted from journal" mean?

Some messages were deleted from the Graylog journal before they could be written to Elasticsearch. Please verify that your Elasticsearch cluster is healthy and fast enough. You may also want to review your Graylog journal settings and set a higher limit.

This can happen when Graylog is not able to connect to Elasticsearch or the Elasticsearch Cluster is not able to process the ingested messages in time. Add more resources to Elasticsearch or adjust *the output settings* from Graylog to Elasticsearch.

25.6.7 What does "Journal utilization is too high" mean?

Journal utilization is too high and may go over the limit soon. Please verify that your Elasticsearch cluster is healthy and fast enough. You may also want to review your Graylog journal settings and set a higher limit.

This can happen when Graylog is not able to connect to Elasticsearch or the Elasticsearch Cluster is not able to process the ingested messages in time. Add more resources to Elasticsearch or adjust *the output settings* from Graylog to Elasticsearch.

25.6.8 How do I fix the "Deflector exists as an index and is not an alias" error message?

Graylog is using an Elasticsearch index alias per index set pointing to the active write index, the so-called "deflector", to write messages into Elasticsearch such as `graylog_deflector` in the default index set.

Please refer to *Index model* for a more in-depth explanation of the Elasticsearch index model used by Graylog.

In some rare situations, there might be an Elasticsearch index with a name which has been reserved for the deflector of an index set managed by Graylog, so that Graylog is unable to create the proper Elasticsearch index alias.

This error situation leads to the following system notification in Graylog:

```
> Deflector exists as an index and is not an alias.
> The deflector is meant to be an alias but exists as an index. Multiple failures of␣
→infrastructure can lead to this. Your messages are still indexed but searches and
→all maintenance tasks will fail or produce incorrect results. It is strongly␣
→recommend that you act as soon as possible.
```

The logs of the Graylog *master* node will contain a warning message similar to the following:

```
WARN  [IndexRotationThread] There is an index called [graylog_deflector]. Cannot fix␣
→this automatically and published a notification.
```

1. Stop all Graylog nodes

2. (*OPTIONAL*) If you want to keep the already ingested messages, reindex them into the Elasticsearch index with the greatest number, e. g. `graylog_23` if you want to fix the deflector `graylog_deflector`, via the Elasticsearch Reindex API.

3. Delete the `graylog_deflector` index via the Elasticsearch Delete Index API.

4. Add `action.auto_create_index:  false` to the configuration files of all Elasticsearch nodes in your cluster and restart these Elasticsearch nodes, see Elasticsearch Index API - Automatic Index Creation and Creating an Index for details.

5. Start the Graylog *master* node.

6. Manually rotate the active write index of the index set on the *System* / *Indices* / *Index Set* page in the *Maintenance* dropdown menu.

7. (*OPTIONAL*) Start all remaining Graylog *slave* nodes.

25.6.9 How do I enable debug logging for a specific plugin or area of Graylog?

When troubleshooting an issue in Graylog, it might be useful to enable `debug` or `trace` logging for the entire Graylog subsystem in (*System* > *Logging*). However, you may find that this generates too much log output (possibly making it difficult to review log messages for a particular area of concern).

Graylog supports the ability to enable `debug` or `trace` logging for specific application areas or plugins. To do this, execute the following terminal command against a particular Graylog node.:

```
curl -I -X PUT http://<graylog-username>:<graylog-password>@<graylog-node-ip>:9000/
→api/system/loggers/<application-package>/level/debug \
-H 'X-Requested-By: graylog-api-user'
```

Note: The `application-package` is the Java package for the area of concern (eg. `org.graylog.aws` for the AWS plugin or `org.graylog2.lookup` for Lookup Tables). You might need to look at the Graylog source code to identify the desired application-package.

25.6.10 Have another troubleshooting question?

See below for some additional support options where you can ask your question.

25.7 Support

25.7.1 I think I've found a bug, how do I report it?

Think you spotted a bug? Oh no! Please report it in our issue trackers so we can take a look at it. All issue trackers are hosted on GitHub, tightly coupled to our code and milestones. Don't hesitate to open issues – we'll just close them if there is nothing to do. Most issues will be in the Graylog server repository, but you should choose others if you have found a bug in one of the plugins.

25.7.2 I'm having issues installing or configuring Graylog, where can I go for support?

Check out the Graylog Community Forums – you can search for your problem which may already have an answer, or post a new question.

Another source is the Graylog channel on Matrix.org or the #graylog IRC chat channel on freenode (both are bridged, so you'll see messages from either channels). Our developers and a lot of community members hang out here. Just join the channel and add any questions, suggestions or general topics you have.

If you're looking for professional commercial support from the Graylog team, we do that too. Please get in touch here for more details.

GELF

26.1 Structured events from anywhere. Compressed and chunked.

The Graylog Extended Log Format (GELF) is a log format that avoids the shortcomings of classic plain syslog:

- Limited to length of 1024 bytes – Not much space for payloads like backtraces

- No data types in structured syslog. You don't know what is a number and what is a string.

- The RFCs are strict enough but there are so many syslog dialects out there that you cannot possibly parse all of them.

- No compression

Syslog is okay for logging system messages of your machines or network gear. GELF is a great choice for logging from within applications. There are libraries and appenders for many programming languages and logging frameworks so it is easy to implement. You could use GELF to send every exception as a log message to your Graylog cluster. You don't have to care about timeouts, connection problems or anything that might break your application from within your logging class because GELF can be sent via UDP.

26.2 GELF via UDP

26.2.1 Chunking

UDP datagrams are usually limited to a size of 8192 bytes. A lot of compressed information fits in there but you sometimes might just have more information to send. This is why Graylog supports chunked GELF.

You can define chunks of messages by prepending a byte header to a GELF message including a message ID and sequence number to reassemble the message later.

Most GELF libraries support chunking transparently and will detect if a message is too big to be sent in one datagram.

Of course TCP would solve this problem on a transport layer but it brings other problems that are even harder to tackle: You would have to care about slow connections, timeouts and other nasty network problems.

With UDP you may just lose a message while with TCP it could bring your whole application down when not designed with care.

Of course TCP makes sense in some (especially high volume environments) so it is your decision. Many GELF libraries support both TCP and UDP as transport. Some do even support HTTP.

Prepend the following structure to your GELF message to make it chunked:

- **Chunked GELF magic bytes - 2 bytes:** `0x1e 0x0f`

- **Message ID - 8 bytes:** Must be the same for every chunk of this message. Identifying the whole message and is used to reassemble the chunks later. Generate from millisecond timestamp + hostname for example.

- **Sequence number - 1 byte:** The sequence number of this chunk. Starting at 0 and always less than the sequence count.

- **Sequence count - 1 byte:** Total number of chunks this message has.

All chunks **MUST** arrive within 5 seconds or the server will discard all already arrived and still arriving chunks. A message **MUST NOT** consist of more than 128 chunks.

Attention: Please note, that the UDP-Inputs of Graylog use the `SO_REUSEPORT` socket option, which was introduced in Linux kernel version 3.9. So be aware, that UDP inputs will **NOT** work on Linux kernel versions prior to 3.9.

26.2.2 Compression

When using UDP as transport layer, GELF messages can be sent uncompressed or compressed with either GZIP or ZLIB.

Graylog nodes detect the compression type in the GELF magic byte header automatically.

Decide if you want to trade a bit more CPU load for saving a lot of network bandwidth. GZIP is the protocol default.

26.3 GELF via TCP

At the current time, GELF TCP only supports uncompressed and non-chunked payloads. Each message needs to be delimited with a null byte (\0) when sent in the same TCP connection.

Attention: GELF TCP **does not support** compression due to the use of the null byte (\0) as frame delimiter.

26.4 GELF Payload Specification

Version 1.1 (11/2013)

A GELF message is a JSON string with the following fields:

- **version `string (UTF-8)`**

    - GELF spec version – "1.1"; **MUST** be set by client library.

- **host `string (UTF-8)`**

    - the name of the host, source or application that sent this message; **MUST** be set by client library.

- **short_message `string (UTF-8)`**

    - a short descriptive message; **MUST** be set by client library.

- **full_message `string (UTF-8)`**

    - a long message that can i.e. contain a backtrace; optional.

- **timestamp `number`**

> – Seconds since UNIX epoch with optional decimal places for milliseconds; *SHOULD* be set by client library. Will be set to the current timestamp (now) by the server if absent.

- **level `number`**

  > – the level equal to the standard syslog levels; optional, default is 1 (ALERT).

- **facility `string (UTF-8)`**

  > – optional, deprecated. Send as additional field instead.

- **line `number`**

  > – the line in a file that caused the error (decimal); optional, deprecated. Send as additional field instead.

- **file `string (UTF-8)`**

  > – the file (with path if you want) that caused the error (string); optional, deprecated. Send as additional field instead.

- **_[additional field] `string (UTF-8) or number`**

  > – every field you send and prefix with an underscore (_) will be treated as an additional field. Allowed characters in field names are any word character (letter, number, underscore), dashes and dots. The verifying regular expression is: `^[\w\.\-]*$`. Libraries SHOULD not allow to send id as additional field (`_id`). Graylog server nodes omit this field automatically.

26.5 Example payload

This is an example GELF message payload. Any graylog-server node accepts and stores this as a message when GZIP/ZLIB compressed or even when sent uncompressed over a plain socket (without newlines).

Note: Newlines must be denoted with the `\n` escape sequence to ensure the payload is valid JSON as per RFC 7159.

```
{
  "version": "1.1",
  "host": "example.org",
  "short_message": "A short message that helps you identify what is going on",
  "full_message": "Backtrace here\n\nmore stuff",
  "timestamp": 1385053862.3072,
  "level": 1,
  "_user_id": 9001,
  "_some_info": "foo",
  "_some_env_var": "bar"
}
```

26.5.1 Sending GELF messages via UDP using netcat

Sending an example message to a GELF UDP input (running on host `graylog.example.com` on port 12201):

```
echo -n '{ "version": "1.1", "host": "example.org", "short_message": "A short message
→", "level": 5, "_some_info": "foo" }' | nc -w0 -u graylog.example.com 12201
```

26.5.2 Sending GELF messages via TCP using netcat

Sending an example message to a GELF TCP input (running on host `graylog.example.com` on port 12201):

```
echo -n -e '{ "version": "1.1", "host": "example.org", "short_message": "A short␣
→message", "level": 5, "_some_info": "foo" }'"\0" | nc -w0 graylog.example.com 12201
```

26.5.3 Sending GELF messages via HTTP using curl

Sending an example message to a GELF HTTP input (running on `http://graylog.example.com:12201/gelf`):

```
curl -X POST -H 'Content-Type: application/json' -d '{ "version": "1.1", "host":
→"example.org", "short_message": "A short message", "level": 5, "_some_info": "foo" }
→' 'http://graylog.example.com:12201/gelf'
```

THE THINKING BEHIND THE GRAYLOG ARCHITECTURE AND WHY IT MATTERS TO YOU

27.1 A short history of Graylog

The Graylog project was started by Lennart Koopmann some time around 2009. Back then the most prominent log management software vendor issued a quote for a one year license of their product that was so expensive that he decided to write a log management system himself. Now you might call this a bit over optimistic (*I'll build this in two weeks*, end of quote) but the situation was hopeless: there was basically no other product on the market and especially no open source alternatives.

27.2 The log management market today

Things have changed a bit since 2009. Now there are viable open source projects with serious products and a growing list of SaaS offerings for log management.

27.2.1 Architectural considerations

Graylog has been successful in providing log management software **because it was built for log management from the beginning**. Software that stores and analyzes log data must have a very specific architecture to do it efficiently. It is more than just a database or a full text search engine because it has to deal with both text data and metrics data on a time axis. Searches are always bound to a time frame (relative or absolute) and only going back into the past because future log data has not been written yet. **A general purpose database or full text search engine that could also store and index the private messages of your online platform for search will never be able to effectively manage your log data.** Adding a specialized frontend on top of it makes it look like it could do the job in a good way but is basically just putting lipstick on the wrong stack.

A log management system has to be constructed of several services that take care of processing, indexing, and data access. The most important reason is that you need to scale parts of it horizontally with your changing use cases and usually the different parts of the system have different hardware requirements. All services must be tightly integrated to allow efficient management and configuration of the system as a whole. A data ingestion or forwarder tool is hard to tedious to manage if the configuration **has** to be stored on the client machines and is not possible via for example REST APIs controlled by a simple interface. A system administrator needs to be able to log into the web interface of a log management product and select log files of a remote host (that has a forwarder running) for ingestion into the tool.

You also want to be able to see the health and configuration of all forwarders, data processors and indexers in a central place because the whole log management stack can easily involve thousands of machines if you include the log emitting clients into this calculation. You need to be able to see which clients are forwarding log data and which are not to make sure that you are not missing any important data.

Graylog is coming the closest to the Splunk architecture:

- **Graylog was solely built as a log management system from the first line of code.** This makes it very efficient and easy to use.

- The `graylog-server` component sits in the middle and works around shortcomings of Elasticsearch (a full text search engine, not a log management system) for log management. It also builds an abstraction layer on top of it to make data access as easy as possible without having to select indices and write tedious time range selection filters, etc. - Just submit the search query and Graylog will take care of the rest for you.

- All parts of the system are tightly integrated and many parts speak to each other to make your job easier.

- Like WordPress makes MySQL a good solution for blogging, Graylog makes Elasticsearch a good solution for logging. You should never have a system or frontend query Elasticsearch directly for log management so we are putting `graylog-server` in front of it.
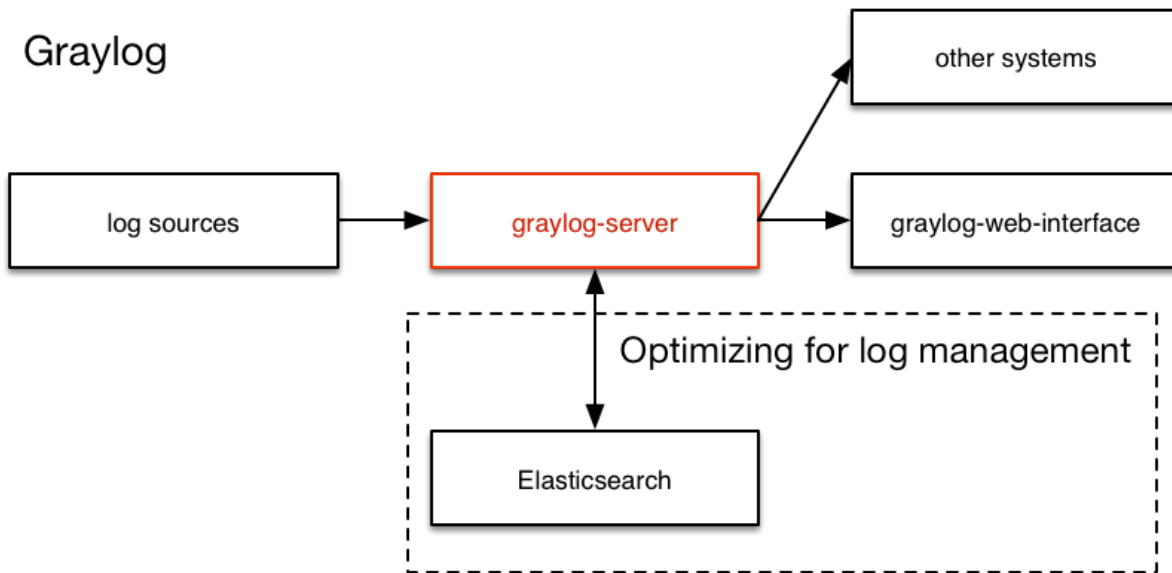
27.2.2 Blackboxes

Closed source systems tend to become black boxes that you cannot extend or adapt to fit the needs of your use case. This is an important thing to consider especially for log management software. The use cases can range from simple syslog centralization to ultra flexible data bus requirements. A closed source system will always make you depending on the vendor because there is no way to adapt. As your setup reaches a certain point of flexibility you might hit a wall earlier than expected.

Consider spending a part of the money you would spend for the wrong license model for developing your own plugins or integrations.

27.3 The future

Graylog is the only open source log management system that will be able to deliver functionality and scaling in a way that Splunk does. It will be possible to replace Elasticsearch with something that is really suited for log data analysis without even changing the public facing APIs.

CHANGELOG

28.1 Graylog 3.1.4

Released: 2020-01-14

28.1.1 Core

Added

- Add URL whitelist service to make sure that lookup data adapters and event notifications cannot use arbitrary URLs. **Attention:** The URL whitelist is enabled by default! On the first server startup after the upgrade, the URL whitelist service will automatically generate whitelist entries for existing lookup data adapters and event notifications. See *URL whitelist documentation* for details.

Changed

- Reduce amount of log messages when job scheduler is disabled on a node. Graylog2/graylog2-server#6451 Graylog2/graylog2-server#6816

Fixed

- Fix global API browser link. Graylog2/graylog2-server#6823 Graylog2/graylog2-server#6826

- Ensure that aggregation events always have a `source_streams` field for permission checks. Graylog2/graylog2-server#6876 Graylog2/graylog2-server#6877

- Fix caching issue with `route_to_stream` pipeline function. Graylog2/graylog2-server#4954 Graylog2/graylog2-server#6788

- Fix issue with calculating index ranges. Graylog2/graylog2-server#6828 Graylog2/graylog2-server#6957 **Attention:** If you are affected by Graylog2/graylog2-server#6828, you have to manually recalculated index ranges for all your index sets.

- Improve API browser for event resources. Graylog2/graylog2-server#6742 Graylog2/graylog2-server#6955

- Fix message indexing issue that could potentially stop message indexing completely. Graylog2/graylog2-server#5091 Graylog2/graylog2-server#6965 Graylog2/graylog2-server#7071

28.1.2 Integrations Plugin

Added

- Add support for automatic authentication scheme for the AWS Kinesis integration. (environment variables, Java properties, instance roles) Graylog2/graylog-plugin-integrations#264 Graylog2/graylog-plugin-integrations#279

- Add "assume role" support for AWS Kinesis integration. Graylog2/graylog-plugin-integrations#264 Graylog2/graylog-plugin-integrations#279

Fixed

- Fix rate limit issue with AWS Kinesis streams. Graylog2/graylog-plugin-integrations#341 Graylog2/graylog-plugin-integrations#342

28.2 Graylog 3.1.3

Released: 2019-11-06

28.2.1 Core

Added

- Add test button to allow testing of event notifications. Graylog2/graylog2-server#6138 Graylog2/graylog2-server#6549

- Add support for MaxMind ASN database type in lookup tables. Graylog2/graylog2-server#6631

- Add `metric_counter_inc` pipeline function. Graylog2/graylog2-server#6644

- Add `length` pipeline function. Graylog2/graylog2-server#6645

- Add `traffic_accounting_size` pipeline function. Graylog2/graylog2-server#6646

- Implement pipeline rule timers to help debugging slow rules. Graylog2/graylog2-server#5494 Graylog2/graylog2-server#6680

Changed

- Hide event streams from pipelines because events are currently not supported in pipeline rules. Graylog2/graylog2-server#6454 Graylog2/graylog2-server#6545

- Increase default connection limits for Elasticsearch connections. Graylog2/graylog2-server#4637 Graylog2/graylog2-server#6679

Fixed

- Fix problem with message highlighting. Graylog2/graylog2-server#6511 Graylog2/graylog2-server#6526

- Improve event and alert processing on nodes with low ingest rates. Graylog2/graylog2-server#6453 Graylog2/graylog2-server#6548

- Remove event streams from global search to fix custom sorting issue. Graylog2/graylog2-server#6384 Graylog2/graylog2-server#6544

- Fix event processing on nodes with a disabled journal. Graylog2/graylog2-server#6449 Graylog2/graylog2-server#6547

- Fix UI problem with select components. Graylog2/graylog2-server#6554

- Fix problem adding multiple percentile aggregations in views. Graylog2/graylog2-server#6597 Graylog2/graylog2-server#6638

- Fix API browser links in setups with multiple nodes. Graylog2/graylog2-server#2360 Graylog2/graylog2-server#5920 Graylog2/graylog2-server#6636

- Fix display of backlog size for legacy event notification plugins. Graylog2/graylog2-server#6647

- Fix issue with MongoDB index creation. Graylog2/graylog2-server#6383 Graylog2/graylog2-server#6643

- Fix stream permission checks for view searches. Graylog2/graylog2-server#6651

- Fix stream sorting in streams filter option in views. Graylog2/graylog2-server#6514 Graylog2/graylog2-server#6657

- Fix problem with event index set migration. Graylog2/graylog2-server#6507 Graylog2/graylog2-server#6662 Graylog2/graylog2-server#6674

- Ensure Grok pattern reloading on bulk pattern import. Graylog2/graylog2-server#6673

- Fix error handling in pipeline parser. Graylog2/graylog2-server#6696 Graylog2/graylog2-server#6731

28.2.2 Integrations Plugin

Fixed

- Fix checkpointing AWS Kinesis streams. Graylog2/graylog-plugin-integrations#286 Graylog2/graylog-plugin-integrations#294

28.3 Graylog 3.1.2

Released: 2019-09-12

28.3.1 Core

Fixed

- Cannot read property 'query' of undefined when replaying search Graylog2/graylog2-server#6355

- Cannot focus input fields in modal form (Dashboard) Graylog2/graylog2-server#6356

- Server start fails with *Index with name: compound_0 already exists with a different name* Graylog2/graylog2-server#6383

- Alerting not working if cluster contains nodes with no active inputs Graylog2/graylog2-server#6415

28.4 Graylog 3.1.1

Released: 2019-09-04

28.4.1 Core

Fixed

- Error when loading a view. Graylog2/graylog2-server#6346

- Fix server startup issue resulting from long index name. Graylog2/graylog2-server#6322

28.4.2 Integrations Plugin

Added

- Add a new *AWS Kinesis/CloudWatch Input* to Graylog, which guides the user through the setup process and performs checks along the way. It also supports an automated CloudWatch Logs to Kinesis Streams setup which helps to automate much of the complicated manual setup.

28.5 Graylog 3.1.0

Released: 2019-08-16

28.5.1 Views & Extended Search

- This feature was partially (everything besides support for parameters in queries) open-sourced in this version. Formerly it was accessible only through the commercial enterprise plugin.

- The API prefix for the views/extended search endpoints has changed from `/api/plugins/org.graylog.plugins.enterprise/(views|search)` to `/api/views` & `/api/views/search`.

- The configuration file directive specifying the maximum age of an unreferenced search object before it is purged has changed from `enterprise_search_maximum_search_age` to `views_maximum_search_age`.

28.5.2 Core

Added

- Add `GracefulShutdownService` Java API to allow plugins to participate in a graceful server shutdown. Graylog2/graylog2-server#5738

- Add `MessageOutput.Factory2` Java API to allow writing output plugins that receive the output instance ID. Graylog2/graylog2-server#5670

- Add metrics for Sidecar HTTP API resources. Graylog2/graylog2-server#5632

- Add new pipeline rule function `grok_exists` to check for the existence of Grok patterns. Graylog2/graylog2-server#5699 Graylog2/graylog2-server#5689

- Allow the `group.id` setting in a Kafka input to be configured by users. Graylog2/graylog2-server#5867 Graylog2/graylog2-server#3976 (Thank you @pdepaepe)

- Add "show documents for value" field value action to views.

- Add automatic refresh feature to views search.

- Add support for extractor creation links to views.

- Add support for highlighting terms in message search results in views.

- Allow user defined granularity for auto interval time-bucketing in views.

- Add views action to duplicate query tabs. (including widgets)

- Make message fields in views message list sortable.

- Add support for stacked bar charts in views.

- Make colors in view charts configurable.

- Add support for percentile metrics in views.

- Add syntax highlighting for pipeline rule editor. Graylog2/graylog2-server#5957

- Add new alerts and events system.

- Add default events and system events streams. Graylog2/graylog2-server#6089 Graylog2/graylog2-server#6085

- Add default `gl2_message_id` field for each new message. Graylog2/graylog2-server#6074 Graylog2/graylog2-server#5994

- Add *experimental* JDK 11 support. Graylog2/graylog2-server#5909 (please don't use in production, yet)

- Add default filebeat collector config for Windows. Graylog2/graylog2-server#6287 Graylog2/collector-sidecar#350

Changed

- Add a high-resolution logo and favicon and optimize all image files. Graylog2/graylog2-server#5748

- Improve autogenerated UI elements for boolean fields. Graylog2/graylog2-server#5678

- Make modal dialogs only close with ESC key, Cancel and "X" buttons. They don't close anymore when users click outside of the modal content. Graylog2/graylog2-server#5823

- Improve error messages when updating Grok patterns. Graylog2/graylog2-server#5629

- Improve error messages for search errors. Graylog2/graylog2-server#5681

- Improve navigation bar layout. Graylog2/graylog2-server#6001

- Improve views query autocompletion suggestions.

- Move Save/Cancel buttons in views widget edit mode.

- Use dynamic pager in views message lists. Graylog2/graylog2-server#6090

- Set default value for email sender to empty value. Graylog2/graylog2-server#5981 (Thank you @bjoernhaeuser)

- Improve stream router engine performance. Graylog2/graylog2-server#6170

- Avoid logging excessive exceptions when other Graylog nodes are not reachable via HTTP. Graylog2/graylog2-server#6229

- Darken disabled inputs on the inputs page, making it easier to see when an input is disabled. Graylog2/graylog2-server#6257

Removed

- Remove "show documents" action for message widgets.

- Remove old stream alerts system. (replaced by new alerts and events system, including automatic alerts migration)

Fixed

- Fix slow loading HTTP API and UI after server startup. Graylog2/graylog2-server#5669

- Fix API path prefixing to allow running Graylog with a sub-path again. Graylog2/graylog2-server#5703 Graylog2/graylog2-server#5702

- Fix issue with dashboard data migration. Graylog2/graylog2-server#5751 Graylog2/graylog2-server#5737

- Add wildcard support to search page query parser. Graylog2/graylog2-server#5730 Graylog2/graylog2-server#5719

- Fix "Expect: 100-Continue" handling in HTTP inputs. Graylog2/graylog2-server#5725 Graylog2/graylog2-server#5690

- Fix HTTP input keep-alive handling. Graylog2/graylog2-server#5728 Graylog2/graylog2-server#5720

- Fix Grok pattern handling with OR patterns. Graylog2/graylog2-server#5749 Graylog2/graylog2-server#4773

- Fix issues with Cisco ASA NetFlow handling. Graylog2/graylog2-server#5729 Graylog2/graylog2-server#5715

- Fix support for underscores (___) in Grok pattern match group names. Graylog2/graylog2-server#5800 Graylog2/graylog2-server#5704 Graylog2/graylog2-server#5563

- Fix auto-completion in several drop-down fields across the UI. Graylog2/graylog2-server#5665 Graylog2/graylog2-server#5659

- Fix problem with loading the extractors page in the UI. Graylog2/graylog2-server#5954 Graylog2/graylog2-server#5948

- Fix links on stream outputs page. Graylog2/graylog2-server#5953 Graylog2/graylog2-server#5952

- Fix edit mode of message widget in views.

- Don't refresh view query when changing query tabs.

- Copy widget filter when creating statistics widget for a field in views.

- Support nested maps in lists for content packs. Graylog2/graylog2-server#6003 Graylog2/graylog2-server#5742

- Reload changed Grok patterns in Grok extractor. Graylog2/graylog2-server#5982 Graylog2/graylog2-server#5833

- Fix support for TLS trusted certificate directories in inputs. Graylog2/graylog2-server#5958 Graylog2/graylog2-server#5939

- Fix plugin UI routes when running the web interface with a sub-path. Graylog2/graylog2-server#6150 Graylog2/graylog2-server#6149

- Fix rendering of events in single message view. Graylog2/graylog2-server#6156 Graylog2/graylog2-server#6142

- Fix rendering of LDAP page. Graylog2/graylog2-server#6249 Graylog2/graylog2-server#6247

- Fix pagination issue with newer JDKs. Graylog2/graylog2-server#6256 Graylog2/graylog2-server#6254

- Fix issue with UDP based inputs becoming unresponsive. Graylog2/graylog2-server#6263 Graylog2/graylog2-server#5701

28.6 Graylog 3.0.2

Released: 2019-05-03

Integrations Plugin

- Fix issue handling quoted values in PaloAlto input Graylog2/graylog-plugin-integrations#15 Graylog2/graylog-plugin-integrations#16

28.7 Graylog 3.0.1

Released: 2019-04-01

Core

- Fix dashboard position migration. Graylog2/graylog2-server#5737 Graylog2/graylog2-server#5763

- Fix HTTP 100 handling in http inputs. Graylog2/graylog2-server#5690 Graylog2/graylog2-server#5725

- Fix http input keep-alive handling. Graylog2/graylog2-server#5720 Graylog2/graylog2-server#5728

- Fix running Graylog web interface under a path prefix. Graylog2/graylog2-server#5702 Graylog2/graylog2-server#5703

- Fix issue with wildcards in the search query parser when running with newer Elasticsearch versions. Graylog2/graylog2-server#5719 Graylog2/graylog2-server#5766

- Fix Grok patterns that use "OR" to not return "null" values. Graylog2/graylog2-server#4773 Graylog2/graylog2-server#5749

- Fix NetFlow parsing for Cisco ASA devices. Graylog2/graylog2-server#5715 Graylog2/graylog2-server#5729

- Fix Grok patterns to support underscores in match group names again. Graylog2/graylog2-server#5704 Graylog2/graylog2-server#5563 Graylog2/graylog2-server#5800

- Document password escaping issue for the MongoDB connection URL. Graylog2/graylog2-server#5680 Graylog2/graylog2-server#5764

Integrations Plugin

- Fix input parsing problem in PaloAlto input. Graylog2/graylog-plugin-integrations#10 Graylog2/graylog-plugin-integrations#11

Threatintel Plugin

- Fix problem with content pack migration. Graylog2/graylog-plugin-threatintel#123

28.8 Graylog 3.0.0

Released: 2019-02-14

- Announcement blog post: https://www.graylog.org/post/announcing-graylog-v3-0-ga

- Upgrade notes: *Upgrading to Graylog 3.0.x*

A detailed changelog is following soon!

28.9 Graylog 2.5.2

Released: 2019-03-15

Core

- Mask password fields of inputs returned by the REST API. Graylog2/graylog2-server#5432 Graylog2/graylog2-server#5733

Integrations Plugin

- Fix input parsing problem in PaloAlto input. Graylog2/graylog-plugin-integrations#10 Graylog2/graylog-plugin-integrations#11

28.10 Graylog 2.5.1

Released: 2018-12-19

Core

- Improve description of `web_endpoint_uri` in `graylog.conf`. Graylog2/graylog2-server#5359

- Add CSRF backward compatibility for older Sidecars. Graylog2/graylog2-server#5388 Graylog2/graylog2-server#4987

AWS Plugin

- Remove low limit for GZIP decompression of AWS events. Graylog2/graylog-plugin-aws#98

Pipeline Processor Plugin

- Fix IPv6 handling in IPv4 `cidr_match` pipeline function. Graylog2/graylog-plugin-pipeline-processor#254 Graylog2/graylog2-server#5405

28.11 Graylog 2.5.0

Released: 2018-11-30

Core

- Improve CSRF protection by requiring a custom HTTP header for non-GET requests sent to the API. (requires sidecar 0.1.7) Graylog2/graylog2-server#4998 Graylog2/graylog2-server#5012 Graylog2/graylog2-server#5182

- Improve alert conditions by making the query string configurable. Graylog2/graylog2-server#5277 Graylog2/graylog2-server#3966

- Improve alert overview for streams. Graylog2/graylog2-server#5311

- Add test button for alert conditions. Graylog2/graylog2-server#5322

- Add DNS lookup adapter that supports foward and reverse lookups. Graylog2/graylog2-server#5274 Graylog2/graylog2-server#4200 Graylog2/graylog2-server#5124 Graylog2/graylog-plugin-threatintel#64

- Add support for Elasticsearch 6.x. Graylog2/graylog2-server#5020 Graylog2/graylog2-server#5064

- Update time zone database for the web interface. Graylog2/graylog2-server#5260 Graylog2/graylog2-server#5245

- Fix description and default values for the DSV HTTP lookup table adapter. Graylog2/graylog2-server#4973 @zionio

- Fix slow and unreliable CSV export by using a bigger default batch size. Graylog2/graylog2-server#5172 Graylog2/graylog2-server#5304

- Fix index stats in index set overview. Graylog2/graylog2-server#5306

- Fix security issue with the users API where regular users could retrieve details of other users. Graylog2/graylog2-server#5308 Graylog2/graylog2-server#5068 @radykal-com

- Fix backslash escaping for phrase searches. Graylog2/graylog2-server#5314 Graylog2/graylog2-server#4111 Graylog2/graylog2-server#5266

Integrations Plugin

- Add Palo Alto input

AWS Plugin

- Add throttling support to the AWS Flow Logs input. Graylog2/graylog-plugin-aws#94 Graylog2/graylog-plugin-aws#85

- Improve logging for the CloudTrail input. Graylog2/graylog-plugin-aws#95 Graylog2/graylog-plugin-aws#80

Pipeline Processor Plugin

- Fix key-value function to be more robust with splitting values. Graylog2/graylog-plugin-pipeline-processor#249 Graylog2/graylog2-server#4920 @radykal-com

- Add support for decimal IPv4 representation in the `to_ip` function. Graylog2/graylog-plugin-pipeline-processor#253 Graylog2/graylog2-server#5268

28.12 Graylog 2.4.7

Released: 2019-03-01

Core

- Mask password fields of inputs returned by the REST API. Graylog2/graylog2-server#5408 Graylog2/graylog2-server#5734

28.13 Graylog 2.4.6

Released: 2018-07-16

Core

- Unbreak input throttling by publishing throttle state again. Graylog2/graylog2-server#4850 Graylog2/graylog2-server#4321

- Check for permission before terminating input. Graylog2/graylog2-server#4867 Graylog2/graylog2-server#4858

- Require user to be authenticated to retrieve plugin list. Graylog2/graylog2-server#4868 Graylog2/graylog2-server#4863

- Add missing permission checks. Graylog2/graylog2-server#4873 Graylog2/graylog2-server#4861 Graylog2/graylog2-server#4859

- UI fixes for index rotation strategy dropdown. Graylog2/graylog2-server#4826 Graylog2/graylog2-server#4769

- XSS fix for typeahead components. Graylog2/graylog2-server#4904

- Fix potential NullPointerException in csv and maxmind data adapters. Graylog2/graylog2-server#4912 Graylog2/graylog2-server#4748

- Add "http_non_proxy_hosts" configuration option. Graylog2/graylog2-server#4915 Graylog2/graylog2-server#4905 Graylog2/graylog2-server#4392

28.14 Graylog 2.4.5

Released: 2018-05-28

Core

- Improve HTTP proxy authentication to support more proxy server software. Graylog2/graylog2-server#4788 Graylog2/graylog2-server#4790

- Fix an issue where the Elasticsearch URL credentials have been written to the Graylog log file. Graylog2/graylog2-server#4804 Graylog2/graylog2-server#4806

- Fix issue with deleting dashboard widgets. Graylog2/graylog2-server#4525 Graylog2/graylog2-server#4808

28.15 Graylog 2.4.4

Released: 2018-05-02

Core

- Fix issues with loading example message for extractor. Graylog2/graylog2-server#4553 Graylog2/graylog2-server#4559

- Allow : character in password. Graylog2/graylog2-server/#4557

- Fix lookup table cache entry count metric. Graylog2/graylog2-server#4558

- Simplify plugin development for lookup table plugins. Graylog2/graylog2-server#4586 Graylog2/graylog2-server#4587

- Fix security issue with unescaped text in notification and dashboard names. Graylog2/graylog2-server#4727 Graylog2/graylog2-server#4739

- Improve high-availability behavior with missing Elasticsearch nodes. Graylog2/graylog2-server#4738

- Fix issue with updating saved searches. Graylog2/graylog2-server#2285 Graylog2/graylog2-server#4743

- Fix issue assigning stream outputs to the default stream. Graylog2/graylog2-server#4747 Graylog2/graylog2-server#4754

- Improve rendering of input configuration forms. Graylog2/graylog2-server#4755 Graylog2/graylog2-server#4745 by @Al2Klimov

- Add code to allow HTTP proxy authentication in `http_proxy_uri` config option. Graylog2/graylog2-server#4594 Graylog2/graylog2-server#4758

ThreatIntel Plugin

- Fix issue with missing threat names in lookup results. Graylog2/graylog-plugin-threatintel#99 by @pbr0ck3r

AWS Plugin

- Support new AWS regions. Graylog2/graylog-plugin-aws#71 Graylog2/graylog-plugin-aws#73 Graylog2/graylog-plugin-aws#75

28.16 Graylog 2.4.3

Released: 2018-01-24

https://www.graylog.org/blog/108-announcing-graylog-v2-4-3

Core

- Fix timezone issue when using the timestamp field in quick values. Graylog2/graylog2-server#4519 Graylog2/graylog2-server#4509

28.17 Graylog 2.4.2

Released: 2018-01-24

Core

- Fix issue with creating dashboards as non-admin user. Graylog2/graylog2-server#4505 Graylog2/graylog2-server#4511

- Fix edge case in recent message loader for extractors. Graylog2/graylog2-server#4513 Graylog2/graylog2-server#4510

- Fix formatting issue when using quick values on the timestamp field. Graylog2/graylog2-server#4423 Graylog2/graylog2-server#4288

- Fix issue with handling the timestamp field in stacked quick value widgets. Graylog2/graylog2-server#4516 Graylog2/graylog2-server#4509

Threatintel Plugin

- Fix README.md about `tor_lookup()` function. Graylog2/graylog2-server#86

28.18 Graylog 2.4.1

Released: 2018-01-19

https://www.graylog.org/blog/107-announcing-graylog-v2-4-1

Core

- Fix SyslogCodecTest.testCiscoSyslogMessages. Graylog2/graylog2-server#4446

- Fix privileges for input start/stop operations. Graylog2/graylog2-server#4454 Graylog2/graylog2-server#4455 Graylog2/graylog2-server#4439

- Fix problem with Elasticsearch indexing over HTTPS. Graylog2/graylog2-server#4485 Graylog2/graylog2-server#4232

- Fix web plugin compatibility problem. Graylog2/graylog2-server#4496

- Fix problem that didn't allow a reader user to edit their own profile. Graylog2/graylog2-server#4488 Graylog2/graylog2-server#4494 Graylog2/graylog2-server#4442 Graylog2/graylog2-server#4420

Pipeline Processor Plugin

- Fix pipeline interpreter log message by handing change event. Graylog2/graylog-plugin-pipeline-processor#235 Graylog2/graylog-plugin-pipeline-processor#241

AWS Plugin

- Fix problem updating the AWS configuration in the UI. Graylog2/graylog-plugin-aws#58 Graylog2/graylog-plugin-aws#60

- Add missing proxy configuration option to UI. Graylog2/graylog-plugin-aws#59 Graylog2/graylog-plugin-aws#61

Threatintel Plugin

- Fix `otx_lookup_domain` pipeline function. Graylog2/graylog-plugin-threatintel#83 Graylog2/graylog2-server#4489

28.19 Graylog 2.4.0

Released: 2017-12-22

https://www.graylog.org/blog/106-announcing-graylog-v2-4-0

No changes since 2.4.0-rc.2.

28.20 Graylog 2.4.0-rc.2

Released: 2017-12-20

Core

- Fixed node-id validator problem by removing the validator for now. Graylog2/graylog2-server#4433

28.21 Graylog 2.4.0-rc.1

Released: 2017-12-19

https://www.graylog.org/blog/105-announcing-graylog-v2-4-0-rc-1

Core

- Fix problem with new node-id file check that got introduced in 2.4.0-beta.4. Graylog2/graylog2-server#4428 # 4427

Threatintel Plugin

- Improve Whois data adapter to always use the ARIN registry for now. Graylog2/graylog2-server#78 76

- Fix object comparison in Whois data adapter. Graylog2/graylog2-server#69

28.22 Graylog 2.4.0-beta.4

Released: 2017-12-15

Core

- Improve HTTPJSONPath lookup table data adapter UI. Graylog2/graylog2-server#4406
- Add and use ExternalLink and ExternalLinkButton components. Graylog2/graylog2-server#4414
- Improve UI components when user lacks permissions. Graylog2/graylog2-server#4416 Graylog2/graylog2-server#4407
- Fix output traffic graph on "System/Overview". Graylog2/graylog2-server#4418 Graylog2/graylog2-server#4395
- Improve query suggestions for field existence. Graylog2/graylog2-server#4422 Graylog2/graylog2-server#4362
- Check node-id file permissions and improve error messages. Graylog2/graylog2-server#4417 Graylog2/graylog2-server#4410

Pipeline Processor Plugin

- Fix problem with `null` values in the `select_jsonpath` function. Graylog2/graylog-plugin-pipeline-processor#233 Graylog2/graylog-plugin-pipeline-processor#232

Threatintel Plugin

- Fix several issues with OTX and Whois data adapters. Graylog2/graylog-plugin-threatintel#75

Anonymous Usage-Stats Plugin

- The plugin got removed.

28.23 Graylog 2.4.0-beta.3

Released: 2017-12-04

Core

- Improve documentation for outputbuffer settings. Graylog2/graylog2-server#4331
- Improve QuickValues stacking. Graylog2/graylog2-server#4343
- Improve auth providers UI. Graylog2/graylog2-server#4347
- Add pluggable global notification area to page. Graylog2/graylog2-server#4353 Graylog2/graylog2-server#4389 Graylog2/graylog2-server#4393
- Fix changing the default index set from the UI. Graylog2/graylog2-server#4377
- Add global traffic counter to system overview page. Graylog2/graylog2-server#4357
- Remove anonymous usage-stats plugin. Graylog2/graylog2-server#4349

AWS Plugin

- Add "logGroup" and "logStream" attributes to flow log and raw log codecs. Graylog2/graylog-plugin-aws#55 Graylog2/graylog-plugin-aws#54

CEF Plugin

- Upgrade to CEF parser 0.0.1.10. Graylog2/graylog-plugin-cef#23 Graylog2/graylog-plugin-cef#24

Threatintel Plugin

- Fix lookup table used in `tor_lookup()` function. Graylog2/graylog-plugin-threatintel#71

- Fix lookup table used in `spamhaus_lookup_ip()` function. Graylog2/graylog-plugin-threatintel#73

28.24 Graylog 2.4.0-beta.2

Released: 2017-11-07

https://www.graylog.org/blog/104-announcing-graylog-v2-4-0-beta-2

Core

- Improve UI elements for field analyzers. Graylog2/graylog2-server#4280 Graylog2/graylog2-server#4230

- Add upgrade notes for new plugins. Graylog2/graylog2-server#4187

- Fix query button in QuickValues widget. Graylog2/graylog2-server#4216 Graylog2/graylog2-server#4278

- Improve QuickValues histogram data. Graylog2/graylog2-server#4312 Graylog2/graylog2-server#4309

- Add loading indicator when reloading field chart data. Graylog2/graylog2-server#4319

- Add feedback on create widget modal. Graylog2/graylog2-server#4320 Graylog2/graylog2-server#4318

- Improve robustness of QuickValues widget with stacked fields. Graylog2/graylog2-server#4322 Graylog2/graylog2-server#4289 Graylog2/graylog2-server#4287 Graylog2/graylog2-server#4082

28.25 Graylog 2.4.0-beta.1

Released: 2017-10-20

https://www.graylog.org/blog/103-announcing-graylog-v2-4-0-beta-1

Core

- Ship NetFlow plugin by default. Graylog2/graylog2-server#4086

- Ship AWS plugin by default. Graylog2/graylog2-server#4085

- Ship Threat Intelligence plugin by default. Graylog2/graylog2-server#4084

- Ship CEF plugin by default. Graylog2/graylog2-server#4161

- Fix race condition in user session removal. Graylog2/graylog2-server#4041 Graylog2/graylog2-server#3634 Graylog2/graylog2-server#3948 Graylog2/graylog2-server#3973

- Update web interface dependencies and fix deprecations. Graylog2/graylog2-server#4057 Graylog2/graylog2-server#4059 Graylog2/graylog2-server#4037 Graylog2/graylog2-server#4192 Graylog2/graylog2-server#4189

- Improve Elasticsearch query performance. Graylog2/graylog2-server#4056 Graylog2/graylog2-server#4051

- Improve web UI performance by using React production mode. Graylog2/graylog2-server#4048

- Add possibility for server plugins to add database migrations. Graylog2/graylog2-server#4095

- Add support for custom HTTP headers in HTTPJSONPath lookup table adapter. Graylog2/graylog2-server#4094

- Fix HTTP header size settings. Graylog2/graylog2-server#4128 Graylog2/graylog2-server#4118

- Add `/system/indices/index_sets/{id}/stats` REST API endpoint to fetch stats for a single index set. Graylog2/graylog2-server#4129 Graylog2/graylog2-server#4088

- Add DSV over HTTP data adapter for lookup tables. Graylog2/graylog2-server#4096

- Improve and update Elasticsearch testing infrastructure. Graylog2/graylog2-server#4125 Graylog2/graylog2-server#4165

- Improve dashboard widget layout to show long widget titles. Graylog2/graylog2-server#4072 @billmurrin

- Fix problem in GELF output by removing special handling of the `facility`. Graylog2/graylog2-server#4141 Graylog2/graylog2-server#4140

- Expose `LdapUserAuthenticator#syncLdapUser()` method to allow usage from plugins. Graylog2/graylog2-server#4159 @gaspardpetit

- Fix problem with getting Elasticsearch stats. Graylog2/graylog2-server#4127 Graylog2/graylog2-server#4119

- Fix Elasticsearch document counting with lots of indices. Graylog2/graylog2-server#4147 Graylog2/graylog2-server#4136

- Fix link placement in multi select UI elements. Graylog2/graylog2-server#3911

- Fix HTTP problems when searching in lots of indices. Graylog2/graylog2-server#4149 Graylog2/graylog2-server#4054 Graylog2/graylog2-server#4168

- Fix config issues with stacked charts. Graylog2/graylog2-server#4151 Graylog2/graylog2-server#4150

- Improve eslint rules for UI development. Graylog2/graylog2-server#4173

- Update serveral server dependencies. Graylog2/graylog2-server#4134

- Add config option to disable analysis features (such a QuickValues) for certain message fields. Graylog2/graylog2-server#4175 Graylog2/graylog2-server#3957

- Improve message separator handling for TCP based inputs. Graylog2/graylog2-server#4106 Graylog2/graylog2-server#4105

- Improve CSV lookup table adapter to use the same field for key and value. Graylog2/graylog2-server#4181 Graylog2/graylog2-server#4177

- Add possibility to create charts for non-numeric fields to show cardinality and total counts. Graylog2/graylog2-server#4182 Graylog2/graylog2-server#4083

- Improve widget and grid positioning and styling. Graylog2/graylog2-server#4160 Graylog2/graylog2-server#4209

- Improve UI testing environment. Graylog2/graylog2-server#4162

- Improve error logging on indexing failures. Graylog2/graylog2-server#4195 Graylog2/graylog2-server#4166

- Improve styling for highlighting checkbox in the search sidebar. Graylog2/graylog2-server#4201

- Fix problem with lookup table content pack import. Graylog2/graylog2-server#4197 Graylog2/graylog-plugin-threatintel#57

- Add configuration options to QuickValue widget. Graylog2/graylog2-server#4205 Graylog2/graylog2-server#4082 Graylog2/graylog2-server#4259 Graylog2/graylog2-server#4258

- Improve styling and positioning for search page widget buttons. Graylog2/graylog2-server#4219

- Improve permission handling to allow regular users to create dashboards. Graylog2/graylog2-server#4155

- Add stats summary for all index sets to the "System/Indices" page. Graylog2/graylog2-server#4211 Graylog2/graylog2-server#4204

- Improve table layout in lookup table UI for entries with long descriptions. Graylog2/graylog2-server#4239 Graylog2/graylog2-server#4172

- Fix permission check which was hiding a menu item in the UI. Graylog2/graylog2-server#4237 Graylog2/graylog2-server#4234

- Fix error with message histogram selection. Graylog2/graylog2-server#4243 Graylog2/graylog2-server#4214

- Add histogram option to QuickValue widget to show values over time. Graylog2/graylog2-server#4244 Graylog2/graylog2-server#4082

- Fix permission handling for editing/deleting roles. Graylog2/graylog2-server#4265

- Fix some smaller lookup table issues. Graylog2/graylog2-server#4266

Map Widget plugin

- Improve rendering and styling for map widget. Graylog2/graylog-plugin-map-widget#53 Graylog2/graylog-plugin-map-widget#54

- Improve styling and positioning for search page widget buttons. Graylog2/graylog-plugin-map-widget#56

Pipeline Processor plugin

- Add various Base encoding functions. (e.g. Base16, Base32, Base64) Graylog2/graylog-plugin-pipeline-processor#190

- Fix sorting of pipeline rules. Graylog2/graylog-plugin-pipeline-processor#208

- Fix `parse_json()` function on invalid input. Graylog2/graylog-plugin-pipeline-processor#210 Graylog2/graylog-plugin-pipeline-processor#209

- Fix NullPointerException when parsing invalid rules. Graylog2/graylog-plugin-pipeline-processor#212 Graylog2/graylog-plugin-pipeline-processor#211

- Improve documentation for lookup table function. Graylog2/graylog-plugin-pipeline-processor#217 @supah-greg

- Fix numeric conversions with `to_double()` and `to_long()`. Graylog2/graylog-plugin-pipeline-processor#219

- Improve rule function documentation in editor by sorting functions alphabetically by name. Graylog2/graylog-plugin-pipeline-processor#222

- Add `remove_from_default` option to `route_to_stream()` function. Graylog2/graylog-plugin-pipeline-processor#220

- Add `remove_from_stream()` function. Graylog2/graylog-plugin-pipeline-processor#220

Collector plugin

- Add `exclude_files` configuration option for filebeat collectors. Graylog2/graylog-plugin-collector#63 @silenceper

AWS plugin

- Fix problem with parsing SNS notification messages. Graylog2/graylog-plugin-aws#47 Graylog2/graylog-plugin-aws#44

- Add support for overriding the `source` field for all input types. Graylog2/graylog-plugin-aws#46

- Add support for cross account role based authentication. Graylog2/graylog-plugin-aws#49 Graylog2/graylog-plugin-aws#48 @radykal-com

CEF plugin

- Improve CEF parser and add proper testing infrastructure.

- Fix problems with Kafka and AMQP inputs.

NetFlow plugin

- Improved NetFlow v9 support. Graylog2/graylog-plugin-netflow#21

Threat Intelligence plugin

- Convert plugin to use the lookup table system. Graylog2/graylog-plugin-threatintel#48

- Fix problem with absent OTX API key in plugin configuration. Graylog2/graylog-plugin-threatintel#54 Graylog2/graylog-plugin-threatintel#53

- Improve pipeline function parameter consistency. Graylog2/graylog-plugin-threatintel#58

- Improve lookup table based data adapters. Graylog2/graylog-plugin-threatintel#63 Graylog2/graylog-plugin-threatintel#61 Graylog2/graylog-plugin-threatintel#59 Graylog2/graylog-plugin-threatintel#67

28.26 Graylog 2.3.2

Released: 2017-10-19

https://www.graylog.org/blog/102-announcing-graylog-v2-3-2

Core

- Fix permission handling for editing/deleting roles. Graylog2/graylog2-server#4270 Graylog2/graylog2-server#4254

- Fix CSV export when using lots of Elasticsearch index shards. Graylog2/graylog2-server#4269 Graylog2/graylog2-server#4190

- Fix infinite redirect loop when accessing non-permitted resources/entities. Graylog2/graylog2-server#4139 Graylog2/graylog2-server#4117

28.27 Graylog 2.3.1

Released: 2017-08-25

https://www.graylog.org/blog/100-announcing-graylog-v2-3-1

Core

- Fix NullPointerException for field stats. Graylog2/graylog2-server#4026 Graylog2/graylog2-server#4045 Graylog2/graylog2-server#4046

- Make GELF parser less strict. Graylog2/graylog2-server#4055

- Fix search requests with selected fields by using source filtering. Graylog2/graylog2-server#4069 Graylog2/graylog2-server#4077 Graylog2/graylog2-server#4068

- Add missing index for *session_id* in "sessions" MongoDB collection. Graylog2/graylog2-server#4070 Graylog2/graylog2-server#4076

- Fix search errors when lots of indices will be used. Graylog2/graylog2-server#4062 Graylog2/graylog2-server#4078 Graylog2/graylog2-server#4054

- Upgrade to Jest 2.4.7+jackson. Graylog2/graylog2-server#4107

- Fix search term highlighting. Graylog2/graylog2-server#4108 Graylog2/graylog2-server#4101

Pipeline Processor Plugin

- Make `locale` parameter of `parse_date()` optional. Graylog2/graylog-plugin-pipeline-processor#202

28.28 Graylog 2.3.0

Released: 2017-07-26

https://www.graylog.org/blog/98-announcing-graylog-v2-3-0

Core

- Fix remote address field for GELF UDP inputs. Graylog2/graylog2-server#3982 Graylog2/graylog2-server#3988 Graylog2/graylog2-server#3980

- Improve error messages for rotation strategies. Graylog2/graylog2-server#3995 Graylog2/graylog2-server#3990

- Fix legend for stacked charts. Graylog2/graylog2-server#4010 Graylog2/graylog2-server#3996

- Fix size based index rotation strategy. Graylog2/graylog2-server#4011 Graylog2/graylog2-server#4008 Graylog2/graylog2-server#3997

- Implement retry handling for failed Elasticsearch requests. Graylog2/graylog2-server#4012 Graylog2/graylog2-server#3993

- Fix NullPointerException in ExceptionUtils. Graylog2/graylog2-server#4014 Graylog2/graylog2-server#4003

- Avoid noisy stack traces when Elasticsearch is not available. Graylog2/graylog2-server#3934 Graylog2/graylog2-server#3861

- Do not run SetIndexReadOnlyAndCalculateRangeJob if index is closed. Graylog2/graylog2-server#3931 Graylog2/graylog2-server#3909

- Fix issues when updating users and user roles. Graylog2/graylog2-server#3942 Graylog2/graylog2-server#3918

- Improved IPv6 support. Graylog2/graylog2-server#3926 Graylog2/graylog2-server#3870

- Fix login code to unbreak SSO plugin. Graylog2/graylog2-server#3973 Graylog2/graylog2-server#3948

- Allow case-insensitive lookups for CSV lookup data adapter. Graylog2/graylog2-server#3971 Graylog2/graylog2-server#3961

- Allow manual lookup table cache purge via UI and API. Graylog2/graylog2-server#3967 Graylog2/graylog2-server#3962

- Mark Message class as not thread-safe. Graylog2/graylog2-server#3978 Graylog2/graylog2-server#3876

- Fail fast and loud for invalid GELF messages. Graylog2/graylog2-server#3972 Graylog2/graylog2-server#3970

- Add support for automatic Elasticsearch node discovery. Graylog2/graylog2-server#3805

- Fix DateHistogram-related functionality in Searches class. Graylog2/graylog2-server#3806

- Hide update spinner with auto-update interval <=5s. Graylog2/graylog2-server#3738 Graylog2/graylog2-server#3723 @billmurrin

- Small spelling/documentation fixes. Graylog2/graylog2-server#3817

- Fix pagination and offset/total hits in Searches. Graylog2/graylog2-server#3821 Graylog2/graylog2-server#3813

- Add sort order to terms API call. Graylog2/graylog2-server#3829

- Don't start stopped inputs after updating them. Graylog2/graylog2-server#3824 Graylog2/graylog2-server#3479

- Allow specifying locale for Date converter. Graylog2/graylog2-server#3820

- Hide "Delete from stream" button if stream is undefined. Graylog2/graylog2-server#3822

- Don't reload errorstates on pages that don't need them. Graylog2/graylog2-server#3839 Graylog2/graylog2-server#3834

- Emit StreamsChangedEvent and StreamDeletedEvent in BundleImporter. Graylog2/graylog2-server#3848 Graylog2/graylog2-server#3842

- Add Lookup Table search result decorator. Graylog2/graylog2-server#3852 Graylog2/graylog2-server#3844

- Check Elasticsearch version when creating index template. Graylog2/graylog2-server#3862

- Add admin user to list of receivers in EmailAlarmCallback. Graylog2/graylog2-server#3864 Graylog2/graylog2-server#3859

- Fix parameters for count query in `Searches#count()`. Graylog2/graylog2-server#3865 Graylog2/graylog2-server#3841

- Add search system for objects in MongoDB Graylog2/graylog2-server#3877

- Make Kafka config setting `auto.offset.reset` configurable for input. Graylog2/graylog2-server#3743 Graylog2/graylog2-server#3894 @r4um

- Use preemptive authentication for Elasticsearch if credentials are given. Graylog2/graylog2-server#3895 Graylog2/graylog2-server#3907

- Add lookup adapter and cache config validation. Graylog2/graylog2-server#3836

- Unbreak elasticsearch duration config settings. Graylog2/graylog2-server#3899

- Fix lookup table UI state problem. Graylog2/graylog2-server#3898

- Enable search for lookup tables, data adapters and caches. Graylog2/graylog2-server#3878

- Make Elasticsearch version injectable. Graylog2/graylog2-server#3896

- Refactor lifecycle for lookup adapters and caches. Graylog2/graylog2-server#3873

- Introduce setting for enabling ES request compression. Graylog2/graylog2-server#3901

- Add content pack import/export for lookup tables, caches and adapters. Graylog2/graylog2-server#3892

- Upgrade to Jackson 2.8.9. Graylog2/graylog2-server#3908

- Fix and centralize lookup adapter/cache error handling. Graylog2/graylog2-server#3905

- Switch RoleResponse to java.util.Optional to fix serialization. Graylog2/graylog2-server#3915

- Add lookup table/cache/adapter permissions. Graylog2/graylog2-server#3914

- Collect and show metrics for lookup caches and adapters. Graylog2/graylog2-server#3917

- Remove obsolete "disableExpensiveUpdates" user preference. Graylog2/graylog2-server#3922

- Migrate to Jackson-based release of Jest 2.4.5. Graylog2/graylog2-server#3925

- Use aliases for reopened indices. Graylog2/graylog2-server#3897

- Add default values for lookup tables. Graylog2/graylog2-server#3921

- Add support for updating extractors in InputService. Graylog2/graylog2-server#3910

- Fix index set overview with closed indices. Graylog2/graylog2-server#3930

- Don't check ES cluster health when flushing messages. Graylog2/graylog2-server#3927

- Retrying bulk indexing in case of all IOExceptions. Graylog2/graylog2-server#3929 Graylog2/graylog2-server#3941

- Add support for automatic Elasticsearch node discovery. Graylog2/graylog2-server#3805

- Fix DateHistogram-related functionality in Searches class. Graylog2/graylog2-server#3806

- Hide update spinner with auto-update interval <=5s. Graylog2/graylog2-server#3738 Graylog2/graylog2-server#3723 @billmurrin

- Small spelling/documentation fixes. Graylog2/graylog2-server#3817

- Fix pagination and offset/total hits in Searches. Graylog2/graylog2-server#3821 Graylog2/graylog2-server#3813

- Add timing metrics to GelfOutput. Graylog2/graylog2-server#3810 Graylog2/graylog2-server#3716

- Add sort order to terms API call. Graylog2/graylog2-server#3829

- Don't start stopped inputs after updating them. Graylog2/graylog2-server#3824 Graylog2/graylog2-server#3479

- Allow specifying locale for Date converter. Graylog2/graylog2-server#3820

- Hide "Delete from stream" button if stream is undefined. Graylog2/graylog2-server#3822

- Don't reload errorstates on pages that don't need them. Graylog2/graylog2-server#3839 Graylog2/graylog2-server#3834

- Emit StreamsChangedEvent and StreamDeletedEvent in BundleImporter. Graylog2/graylog2-server#3848 Graylog2/graylog2-server#3842

- Add Lookup Table search result decorator. Graylog2/graylog2-server#3852 Graylog2/graylog2-server#3844

- Check Elasticsearch version when creating index template. Graylog2/graylog2-server#3862

- Add admin user to list of receivers in EmailAlarmCallback. Graylog2/graylog2-server#3864 Graylog2/graylog2-server#3859

- Fix parameters for count query in `Searches#count()`. Graylog2/graylog2-server#3865 Graylog2/graylog2-server#3841

- Allow version '0' for structured syslog messages. Graylog2/graylog2-server#3503

- Ignore Content-Type in `HttpTransport`. Graylog2/graylog2-server#3508 Graylog2/graylog2-server#3477

- Ensure that `index_prefix` is lower case. Graylog2/graylog2-server#3509 Graylog2/graylog2-server#3476

- Make map in `MessageInput#asMap()` mutable. Graylog2/graylog2-server#3521 Graylog2/graylog2-server#3515

- Fix pagination for alert conditions. Graylog2/graylog2-server#3529 Graylog2/graylog2-server#3528

- Wait until alert notification types are loaded. Graylog2/graylog2-server#3537 Graylog2/graylog2-server#3534

- Upgrade development environment to Webpack v2. Graylog2/graylog2-server#3460

- Add an option to repeat alert notifications again. Graylog2/graylog2-server#3536 Graylog2/graylog2-server#3511

- Fix accidentally changed exports of `UsersStore` Graylog2/graylog2-server#3560 Graylog2/graylog2-server#3556

- Properly escape username/roles in web interface. Graylog2/graylog2-server#3570 Graylog2/graylog2-server#3569

- Improved dashboard grid system. Graylog2/graylog2-server#3575

- Add support for sorting by count to `Search#terms()`. Graylog2/graylog2-server#3540 (@billmurrin)

- Fix for copy query button. Graylog2/graylog2-server#3548 (@billmurrin)

- Fix issue with cloning streams. Graylog2/graylog2-server#3615 Graylog2/graylog2-server#3608

- Prevent session extension when polling system messages. Graylog2/graylog2-server#3632 Graylog2/graylog2-server#3628

- Prevent session extension when polling system jobs. Graylog2/graylog2-server#3625 Graylog2/graylog2-server#3587

- Prevent NPE due to race between rotation and retention threads. Graylog2/graylog2-server#3637 Graylog2/graylog2-server#3494

- Fix problem with message decorators and field selection. Graylog2/graylog2-server#3585 Graylog2/graylog2-server#3584

- Fix issue with loading indicator on an empty search result page. Graylog2/graylog2-server#3652 Graylog2/graylog2-server#3650

- Fix navigation in LDAP users UI. Graylog2/graylog2-server#3651 Graylog2/graylog2-server#3485

- Ensure that plugin RPMs will be built for Linux. Graylog2/graylog2-server#3658 Graylog2/graylog2-server#3657

- Fix reloading problem with content packs and GROK patterns. Graylog2/graylog2-server#3621 Graylog2/graylog2-server#3610

- Add support for Cisco and FortiGate Syslog messages. Graylog2/graylog2-server#3599

- Fix permission problem for inputs API. Graylog2/graylog2-server#3681

- Restore removal of role permissions upon roles update. Graylog2/graylog2-server#3683

- Comply with grace condition when repeat alert notifications is enabled. Graylog2/graylog2-server#3676 Graylog2/graylog2-server#3579

- Invalidate dashboards data after logout. Graylog2/graylog2-server#3700 Graylog2/graylog2-server#3693

- Fix OptionalStringValidator and validations for extractors. Graylog2/graylog2-server#3633 Graylog2/graylog2-server#3630

- Better time range for "Show Received Messages" button on inputs page. Graylog2/graylog2-server#3725

- Remove deprecated rotation/retention configuration resources. Graylog2/graylog2-server#3724

- Introduce lookup tables feature. Graylog2/graylog2-server#3748

- Creating dashboard from search page does now select the right ID Graylog2/graylog2-server#3786 Graylog2/graylog2-server#3785

- Fix importing of dashboards from content packs Graylog2/graylog2-server#3766 Graylog2/graylog2-server#3765

Beats Plugin

- Spelling fixes. Graylog2/graylog-plugin-beats#22 @jsoref

Collector Plugin

- Increase "Show messages" time range.

- Allow collectors list to be filtered by tags. Graylog2/graylog-plugin-collector#52

- UI and UX improvements. Graylog2/graylog-plugin-collector#55

- Fix configuration setting usage for collector heartbeat interval. Graylog2/graylog-plugin-collector#58

- Prevent unwanted session extension. Graylog2/graylog-plugin-collector#49

Map Widget Plugin

- Adjust to lookup cache and adapter changes in server. Graylog2/graylog-plugin-map-widget#48 Graylog2/graylog-plugin-map-widget#50

- Fix refresh handling in error conditions. Graylog2/graylog-plugin-map-widget#49

- Fix HMR as in Graylog/graylog2-server#3591 Graylog2/graylog-plugin-map-widget#51

- Update to a new GeoIP2 release.

- Add lookup tables data adapter for the GeoIP2 database. Graylog2/graylog-plugin-map-widget#40

Pipeline Processor Plugin

- Improve robustness of `clone_message()` pipeline function. Graylog2/graylog-plugin-pipeline-processor#192 Graylog2/graylog2-server#3880

- Fix pipline condition handling of "match all"/"match either". Graylog2/graylog-plugin-pipeline-processor#193 Graylog2/graylog2-server#3924

- Fix serialization/deserialization of pipeline StageSource. Graylog2/graylog-plugin-pipeline-processor#195 Graylog2/graylog-plugin-pipeline-processor#194

- Improve error handling with invalid expressions. Graylog2/graylog-plugin-pipeline-processor#196 Graylog2/graylog-plugin-pipeline-processor#185

- Spelling fixes. Graylog2/graylog-plugin-pipeline-processor#181 @jsoref

- Add support for custom locale in `parse_date()` function. Graylog2/graylog-plugin-pipeline-processor#184 Graylog2/graylog-plugin-pipeline-processor#183

- Smaller UI and UX changes. Graylog2/graylog-plugin-pipeline-processor#186

- New function: debug() Graylog2/graylog-plugin-pipeline-processor#188

- Allow snake-case access to bean objects Graylog2/graylog-plugin-pipeline-processor#189

- Improve lookup functions. Graylog2/graylog-plugin-pipeline-processor#191

- Spelling fixes. Graylog2/graylog-plugin-pipeline-processor#181 @jsoref

- Use uppercase timezone in `TimezoneAwareFunction` and fix default value. Graylog2/graylog2-server#169 Graylog2/graylog2-server#168

- Add `lookup` and `lookup_value` pipeline functions for lookup table support. Graylog2/graylog2-server#177

28.29 Graylog 2.2.3

Released: 2017-04-04

https://www.graylog.org/blog/92-announcing-graylog-v2-2-3

Core

- Prevent unwanted session extension. Graylog2/graylog2-server#3583

- Properly escape username/roles in web interface. Graylog2/graylog2-server#3588

- Allow "-" in the path pattern for the index range rebuild endpoint. Graylog2/graylog2-server#3600

- Copy Query Button Fix. Graylog2/graylog2-server#3491

- Fixing slicing of alert notifications in pagination. Graylog2/graylog2-server#3619

- Fix cloning alert conditions with identical id when cloning stream. Graylog2/graylog2-server#3616

- Use UTC for embedded Date objects as well. Graylog2/graylog2-server#3626

- Prevent session extension for polling system messages. Graylog2/graylog2-server#3638
- Support replacing whitespace in nested keys for JSON extractor. Graylog2/graylog2-server#3623
- Prevent NPE due to race between rotation and retention threads. Graylog2/graylog2-server#3640
- Prevent session extension in SystemJobsStore. Graylog2/graylog2-server#3625
- Render loading indicator on no results page. Graylog2/graylog2-server#3667
- Using consistent collection of non displayable fields to filter against. Graylog2/graylog2-server#3668
- Ensure that plugin RPMs will be built for Linux. Graylog2/graylog2-server#3659
- Fix navigation in LDAP components. Graylog2/graylog2-server#3670
- Publish GrokPatternsChangedEvent when using content packs. Graylog2/graylog2-server#3621
- Add support for Cisco and FortiGate syslog messages. Graylog2/graylog2-server#3599
- Effectively change user permissions when listing inputs. Graylog2/graylog2-server#3682
- Restore removal of role permissions upon roles update. Graylog2/graylog2-server#3684
- Comply with grace condition when repeat alert notifications is enabled Graylog2/graylog2-server#3676

Pipeline Processor

- Use uppercase timezone in TimezoneAwareFunction and fix default value. Graylog2/graylog-plugin-pipeline-processor#169

28.30 Graylog 2.2.2

Released: 2017-03-03

https://www.graylog.org/blog/90-announcing-graylog-v2-2-2

Core

- Give an option to repeat alert notifications. Graylog2/graylog2-server#3511
- Wait until notification types are loaded. Graylog2/graylog2-server#3534
- Fixing slicing of alert conditions in pagination. Graylog2/graylog2-server#3528
- Fix command line help of the server jar. Graylog2/graylog2-server#3527

28.31 Graylog 2.2.1

Released: 2017-02-20

https://www.graylog.org/blog/89-announcing-graylog-v2-2-1

Core

- Allow version '0' for structured syslog messages. Graylog2/graylog2-server#3502
- Ignore `Content-Type` in `HttpTransport`. Graylog2/graylog2-server#3477
- Ensure that `index_prefix` is lower case. Graylog2/graylog2-server#3476
- Add missing whitespace in SystemOutputsPage. Graylog2/graylog2-server#3505

28.32 Graylog 2.2.0

Released: 2017-02-14

https://www.graylog.org/blog/88-announcing-graylog-v2-2-0

Core

- Warn about pipeline stream routing Graylog2/graylog2-server#3472
- Add npm shrinkwrap for 2.2.0 Graylog2/graylog2-server#3468
- Use consistent case in old message count conditions Graylog2/graylog2-server#3454
- Fix stream filter view. Graylog2/graylog2-server#3390
- Use the default index set by default in stream create form. Graylog2/graylog2-server#3397
- Fix broken decorator with duplicate messages. Graylog2/graylog2-server#3400
- Updating index sets store when stream form is opened. Graylog2/graylog2-server#3410
- Only show extractor actions on string fields. Graylog2/graylog2-server#3404
- Use correct format when adding timestamp to search. Graylog2/graylog2-server#3412
- Properly escape strings sent to /messages/{index}/analyze. Graylog2/graylog2-server#3418
- Retain input and stream IDs in content packs. Graylog2/graylog2-server#3416
- Use "order": -1 in default index template to allow override. Graylog2/graylog2-server#3422
- Improve base-line memory usage. Graylog2/graylog2-server#3425
- Use condition and notification placeholders. Graylog2/graylog2-server#3432
- Get field stats for indices only, which contain field. Graylog2/graylog2-server#3424
- Redirect to overview after editing index set configuration. Graylog2/graylog2-server#3438
- Send "stream" property when updating a decorator. Graylog2/graylog2-server#3439
- Adding simple cache for index sets in MongoIndexSetRegistry. Graylog2/graylog2-server#3440
- Restrict search in RecentMessageLoader to 1 hour. Graylog2/graylog2-server#3367
- Remove stray whitespace in MongoIndexSet. Graylog2/graylog2-server#3371
- Add more detail to index range system notification. Graylog2/graylog2-server#3372
- Suppress error notification when loading a potentially missing input. Graylog2/graylog2-server#3373
- Ensure resolved at is non-null on resolvedSecondsAgo. Graylog2/graylog2-server#3376
- Do not allow testing a message against the default stream. Graylog2/graylog2-server#3377
- Propagate shard failure in multi-index search to global search failure. Graylog2/graylog2-server#3379
- Add support for arrays to "contains" stream rule. Graylog2/graylog2-server#3380
- Automatically unsubscribe from DecoratorsStore in SearchPage. Graylog2/graylog2-server#3363
- Keep modified search bar params when opening modal. Graylog2/graylog2-server#3384
- Reset keyword content after changing range type. Graylog2/graylog2-server#3386
- Disable the "set as default" action for the default index set. Graylog2/graylog2-server#3385
- Unregistering component callbacks from Stream(Rules)Store. Graylog2/graylog2-server#3378

- Check for stream existence when displaying replay link. Graylog2/graylog2-server#3387
- Reuse Generator instance in DocumentationResource. Graylog2/graylog2-server#3293
- Fix: Refreshing saved searches store after deleting one. Graylog2/graylog2-server#3309
- Escape curly braces in Swagger resource URIs for challenged browsers. Graylog2/graylog2-server#3290
- Unbreak Firefox by requesting JSON when pinging the API Graylog2/graylog2-server#3312
- Escape search query when using surrounding search. Graylog2/graylog2-server#3314
- Close idle GELF HTTP connections after a timeout. Graylog2/graylog2-server#3315
- Ensure that index ranges are deleted when an index set is removed. Graylog2/graylog2-server#3310
- Ignore reopened indices for count-based retention strategies. Graylog2/graylog2-server#3321
- Ignore deprecated "default" field in IndexSetConfig. Graylog2/graylog2-server#3329
- Use last executed search data on auto-refresh. Graylog2/graylog2-server#3330
- Show stats for each index set on the index sets overview page. Graylog2/graylog2-server#3322
- Not fetching/checking unnecessary streams in AlertScannerThread. Graylog2/graylog2-server#3336
- Add more metrics for extractors. Graylog2/graylog2-server#3332
- Saved search improvements. Graylog2/graylog2-server#3338
- Warn when neither reader nor admin roles are selected for a user. Graylog2/graylog2-server#3337
- Prevent setting default index set readonly. Graylog2/graylog2-server#3339
- Add migration to fix parameter value types for alert conditions. Graylog2/graylog2-server#3340
- Fix unlock dashboard from link. Graylog2/graylog2-server#3344
- Allow re-configuration of shards and replicas in the UI. Graylog2/graylog2-server#3349
- Adapt grace period to latest changes in alerts. Graylog2/graylog2-server#3346
- Go back in history instead to users page when canceling user form. Graylog2/graylog2-server#3350
- Improve wrapping of entity title and description. Graylog2/graylog2-server#3351
- Keep stream filter after editing stream. Graylog2/graylog2-server#3352
- Guard against duplication key error from MongoDB. Graylog2/graylog2-server#3358
- Keep calling convention of SearchPage._refreshData consistent. Graylog2/graylog2-server#3357
- Creating MongoDB indices in services running conditional queries. Graylog2/graylog2-server#3345
- Fix NPE in MongoDbSessionDAO#doDelete(). Graylog2/graylog2-server#3227
- Support syslog messages with ISO-8601 timestamps. Graylog2/graylog2-server#3228
- Use local copies of Droid Sans font in Swagger UI. Graylog2/graylog2-server#3229
- Remove empty row if input description is empty. Graylog2/graylog2-server#3237
- Make "sender" optional in EmailAlarmCallback. Graylog2/graylog2-server#3224
- Fix URL for throbber image. Graylog2/graylog2-server#3242
- Remove special handling in SplitAndCountConverter. Graylog2/graylog2-server#3230
- Add missing AuditBindings to journal commands. Graylog2/graylog2-server#3226

- Don't check private key/certificate if REST API and web interface on same port. Graylog2/graylog2-server#3231

- Add configuration settings for timeout and concurrency of OptimizeIndexJob. Graylog2/graylog2-server#3225

- Change way of exporting CSV search results. Graylog2/graylog2-server#3238

- Add space in processing limit notification. Graylog2/graylog2-server#3256

- Only recalculate index set ranges in index set maintenance menu. Graylog2/graylog2-server#3252

- Fix alert condition validations. Graylog2/graylog2-server#3257

- Validate alarm callbacks before saving them. Graylog2/graylog2-server#3262

- Only update index ranges for managed indices. Graylog2/graylog2-server#3259

- Resolve alerts from deleted alert conditions. Graylog2/graylog2-server#3265

- Alert UI changes. Graylog2/graylog2-server#3266

- Properly track stream throughput for the default stream. Graylog2/graylog2-server#3278

- Add support for OPTIONS request to HttpTransport. Graylog2/graylog2-server#3234

- Add list of stream IDs to Message#toElasticSearchObject(). Graylog2/graylog2-server#3277

- Fix document counts with an empty index set. Graylog2/graylog2-server#3291

- Handle ElasticsearchException in Counts class. Graylog2/graylog2-server#3288

- Move client-side split/import of Grok pattern files to server. Graylog2/graylog2-server#3284

- Showing index set in stream listing only if user is permitted to. Graylog2/graylog2-server#3300

- Fix reloading after notification changes. Graylog2/graylog2-server#3264

- Add "messages:analyze" permission to reader permission set. Graylog2/graylog2-server#3305

- Disable alarm notification controls if user is not permitted to edit. Graylog2/graylog2-server#3303

- Changing conditional to check for presence of index set definition. Graylog2/graylog2-server#3304

- Allow to get a thread dump as plain text. Graylog2/graylog2-server#3289

- Add missing authentication to ClusterResource and ClusterStatsResource. Graylog2/graylog2-server#3279

- Save anchor and last rotation in TimeBasedRotationStrategy depending on IndexSet. Graylog2/graylog2-server#3306

- Fix loading of plugins in development mode. Graylog2/graylog2-server#3185

- Add contains string tester. Graylog2/graylog2-server#3186

- Index set fixes. Graylog2/graylog2-server#3193

- Add synthetic JavaBean getters to AutoValue classes. Graylog2/graylog2-server#3188

- Improve IndexSetValidator. Graylog2/graylog2-server#3197

- Add custom Jackson (de-) serializer for ZonedDateTime and DateTime. Graylog2/graylog2-server#3198

- Improved alarm callback testing. Graylog2/graylog2-server#3196

- Fix DateTime serialization. Graylog2/graylog2-server#3202

- Fix labels on field graphs. Graylog2/graylog2-server#3204

- Alerts cleanup. Graylog2/graylog2-server#3205

- Index set UI fixes. Graylog2/graylog2-server#3203

- Fix quickvalues and field statistics refresh. Graylog2/graylog2-server#3206

- Allow fetching streams by index set ID. Graylog2/graylog2-server#3207

- UI improvements. Graylog2/graylog2-server#3213

- IndexSet default setting. Graylog2/graylog2-server#3209

- Alerts UI improvements. Graylog2/graylog2-server#3214

- Create DefaultIndexSetConfig when creating the default index set. Graylog2/graylog2-server#3215

- ClusterEventPeriodical must use MongoJackObjectMapper. Graylog2/graylog2-server#3217

- Fix V20161130141500_DefaultStreamRecalcIndexRanges. Graylog2/graylog2-server#3222

- Migration improvements. Graylog2/graylog2-server#3211

Beats plugin

- Add support for Metricbeat

- Extract "fields" for every type of beat

Pipeline processor plugin

- Allow duplicate stream titles in route_to_stream. Graylog2/graylog-plugin-pipeline-processor#154

- Do not use lambdas with gauge metrics. Graylog2/graylog-plugin-pipeline-processor#152

- Add clone_message() function. Graylog2/graylog-plugin-pipeline-processor#153

- Track total pipeline interpreter executionTime as a single metric. Graylog2/graylog-plugin-pipeline-processor#155

Collector sidecar plugin

- Fix: Reload on Beats configuration change.

- Update Beats to version 5.1.1

- Fix race conditions in start/stop/restart code for the exec runner Graylog2/collector-sidecar#123

- Add debug switch Graylog2/collector-sidecar#124

- Using Modern UI in a standard way (thanks to @nicozanf) Graylog2/collector-sidecar#125

- Extract etag cache into its own service. Graylog2/graylog-plugin-collector#43

28.33 Graylog 2.1.3

Released: 2017-01-26

https://www.graylog.org/blog/84-announcing-graylog-2-1-3

Core

- Use "order": -1 in default index template to allow override. Graylog2/graylog2-server#3426

- Add missing authentication to ClusterResource and ClusterStatsResource. Graylog2/graylog2-server#3427

- Unbreak Firefox by requesting JSON when pinging the API. Graylog2/graylog2-server#3430

- Use custom Grizzly error page to prevent XSS. Graylog2/graylog2-server#3428

Beats plugin

- Add support for Metricbeat. Graylog2/graylog-plugin-beats#19

• Extract "fields" for every type of beat. Graylog2/graylog-plugin-beats#18

28.34 Graylog 2.1.2

Released: 2016-11-04

https://www.graylog.org/blog/75-announcing-graylog-v2-1-2

Core

• Improve logging in DecodingProcessor. Graylog2/graylog2-server#3025, Graylog2/graylog2-server#3034

• Support all ZLIB compression levels for GELF messages. Graylog2/graylog2-server#3022, Graylog2/graylog2-server#3036

• Implement "contains" stream rule. Graylog2/graylog2-server#3020, Graylog2/graylog2-server#3037

• Make ValidatorProvider a Singleton. Graylog2/graylog2-server#3019, Graylog2/graylog2-server#3038

• Fix NPE in MongoProbe if MongoDB doesn't run with MMAPv1. Graylog2/graylog2-server#3018, Graylog2/graylog2-server#3039

• Fix NPE in Indices#numberOfMessages(String). Graylog2/graylog2-server#3016, Graylog2/graylog2-server#3041

• Only create new LdapConnectionConfig if LDAP is enabled. Graylog2/graylog2-server#3017, Graylog2/graylog2-server#3040

• Properly track replace-all flag and pass through to API. Graylog2/graylog2-server#3023, Graylog2/graylog2-server#3043

• Replace Jersey GZipEncoder with Grizzly's GZipFilter. Graylog2/graylog2-server#3021, Graylog2/graylog2-server#3044

• Prevent n+1 query loading for Stream Rules. Graylog2/graylog2-server#3024, Graylog2/graylog2-server#3035. Thank you @bjoernhaeuser!

• Handle search execution errors. Graylog2/graylog2-server#3027, Graylog2/graylog2-server#3045

• Calculate cardinality on field graphs. Graylog2/graylog2-server#3028, Graylog2/graylog2-server#3046

• Update stats function in field graph description. Graylog2/graylog2-server#3029, Graylog2/graylog2-server#3047

• Use response status 500 if search failed but wasn't syntax error. Graylog2/graylog2-server#3026, Graylog2/graylog2-server#3042

• Improved search indicators. Graylog2/graylog2-server#3031, Graylog2/graylog2-server#3050

• Fix field analyzers loading when search changes. Graylog2/graylog2-server#3030, Graylog2/graylog2-server#3049

• Close search query autocompletion on enter. Graylog2/graylog2-server#3032, Graylog2/graylog2-server#3051

• Refresh stream search when stream changes. Graylog2/graylog2-server#3033, Graylog2/graylog2-server#3052

• Update Joda-Time and moment-timezone. Graylog2/graylog2-server#3059, Graylog2/graylog2-server#3060

• Search button does not always trigger a new search. Graylog2/graylog2-server#3062, Graylog2/graylog2-server#3063

Beats plugin

- Fix frame decoding in case of lost connection. Graylog2/graylog-plugin-beats#14, Graylog2/graylog-plugin-beats#15, Graylog2/graylog-plugin-beats#17. Thank you @hc4!

- Support messages >1024 bytes in BeatsFrameDecoder. Graylog2/graylog-plugin-beats#10, Graylog2/graylog-plugin-beats#12

Pipeline processor plugin

- Don't doubly negate the value of the expression. Graylog2/graylog-plugin-pipeline-processor#126, Graylog2/graylog-plugin-pipeline-processor#127

28.35 Graylog 2.1.1

Released: 2016-09-14

https://www.graylog.org/blog/69-announcing-graylog-v2-1-1

Core

- Proxied requests query other nodes in parallel. Graylog2/graylog2-server#2764, Graylog2/graylog2-server#2779

- Fix 404s on IE 11 using compatibility view. Graylog2/graylog2-server#2768, Graylog2/graylog2-server#2782

- Modify actions in search page triggering a page reload. Graylog2/graylog2-server#2488, Graylog2/graylog2-server#2798

- Do not display login form while loading. Graylog2/graylog2-server#2770, Graylog2/graylog2-server#2802

- Check in SearchPage if search is in progress, reuse promise then. Graylog2/graylog2-server#2799, Graylog2/graylog2-server#2803

- Use index and message_id as message identifier. Graylog2/graylog2-server#2801, Graylog2/graylog2-server#2804

- Fix: file handle leak in KeyUtil (SSL). Graylog2/graylog2-server#2808. Thank you @gbu-censhare!

- Use current search time configuration for CSV export. Graylog2/graylog2-server#2795, Graylog2/graylog2-server#2809

- Explicitly close okhttp response body, avoiding leak connection warning. Graylog2/graylog2-server#2811. Thank you @chainkite!

- Properly close OkHttp Response objects to avoid resource leaks. Graylog2/graylog2-server#2812

- Remove ldap settings check from authenticators. Graylog2/graylog2-server#2817, Graylog2/graylog2-server#2820

Map plugin

- Ignore internal message fields (starting with "gl2_"). Graylog2/graylog-plugin-map-widget#17

Pipeline processor plugin

- Display boolean values in pipeline simulator. Graylog2/graylog-plugin-pipeline-processor#54, Graylog2/graylog-plugin-pipeline-processor#99

- Use case insensitive lookup for timezone IDs. Graylog2/graylog-plugin-pipeline-processor#100, Graylog2/graylog-plugin-pipeline-processor#102

28.36 Graylog 2.1.0

Released: 2016-09-01

https://www.graylog.org/blog/68-announcing-graylog-v-2-1-0-ga

Core

- Refactoring of audit events. Graylog2/graylog2-server#2687

- Add a prop to display/hide the page selector. Graylog2/graylog2-server#2711

- Ensure that `rest_transport_uri` can override the URI scheme of `rest_listen_uri`. Graylog2/graylog2-server#2680, Graylog2/graylog2-server#2704

- Handle indexer cluster down in web interface. Graylog2/graylog2-server#2623, Graylog2/graylog2-server#2713

- Prevent NPE and verbose logging if converter returns null. Graylog2/graylog2-server#2717, Graylog2/graylog2-server#2729

- Let widget replay search button open in a new tab or window. Graylog2/graylog2-server#2725, Graylog2/graylog2-server#2726

- Return `"id"` instead of `"_id"` for message decorators. Graylog2/graylog2-server#2734, Graylog2/graylog2-server#2735

- Make id field consistent for alarm callback histories. Graylog2/graylog2-server#2737

- Audit event changes. Graylog2/graylog2-server#2718

- Let specific stores reuse promises if request is in progress. Graylog2/graylog2-server#2625, Graylog2/graylog2-server#2712

- Disable editing controls for decorator if user lacks permissions. Graylog2/graylog2-server#2730, Graylog2/graylog2-server#2736

- Styling of decorator list. Graylog2/graylog2-server#2743, Graylog2/graylog2-server#2744

- Do not load plugins for journal commands. Graylog2/graylog2-server#2667

- Use proper other count for pie chart slices. Graylog2/graylog2-server#2639, Graylog2/graylog2-server#2671

- Removing unused prop type in StreamRuleList component. Graylog2/graylog2-server#2673

- Add a generic search form component. Graylog2/graylog2-server#2678

- Decorator improvements. Graylog2/graylog2-server#2519, Graylog2/graylog2-server#2666, Graylog2/graylog2-server#2674

- Only show notification link when there are notifications. Graylog2/graylog2-server#2677

- Enable gzip per default for REST API listener. Graylog2/graylog2-server#2670, Graylog2/graylog2-server#2672

- Improvements in raw message loader. Graylog2/graylog2-server#2684

- Allow users of MessageFields to disable field actions. Graylog2/graylog2-server#2685

- Generating a relative redirect URL for web interface in root resource. Graylog2/graylog2-server#2593, Graylog2/graylog2-server#2675

- Add help text for session's client address. Graylog2/graylog2-server#2656, Graylog2/graylog2-server#2692

- Fix content pack extractor validation. Graylog2/graylog2-server#2663, Graylog2/graylog2-server#2697

- Reset users' startpages if referenced stream/dashboard is deleted. Graylog2/graylog2-server#2400, Graylog2/graylog2-server#2695, Graylog2/graylog2-server#2702

- Fix token creation via API browser. Graylog2/graylog2-server#2668, Graylog2/graylog2-server#2698

- Allow surrounding search to be opened in new tab. Graylog2/graylog2-server#2531, Graylog2/graylog2-server#2699

- Reformatting component, adding error handler for fetching dashboard. Graylog2/graylog2-server#2576, Graylog2/graylog2-server#2703

- Add format string message decorator. Graylog2/graylog2-server#2660

- Reloading CurrentUserStore when updated user is the current user. Graylog2/graylog2-server#2705, Graylog2/graylog2-server#2706

- General UI improvements Graylog2/graylog2-server#2700

- Add Syslog severity mapper decorator. Graylog2/graylog2-server#2590

- Improvements in message decorators. Graylog2/graylog2-server#2592, Graylog2/graylog2-server#2591, Graylog2/graylog2-server#2598, Graylog2/graylog2-server#2654

- Revert "Move link to API Browser into System menu". Graylog2/graylog2-server#2586, Graylog2/graylog2-server#2587

- Print - instead of null when client did not provide user agent header. Graylog2/graylog2-server#2601. Thank you @mikkolehtisalo!

- Change logging in normalizeDn() to debug to avoid noisy warnings. Graylog2/graylog2-server#2599

- Ensure that `{rest,web}_{listen,transport,endpoint}_uri` settings are absolute URIs. Graylog2/graylog2-server#2589, Graylog2/graylog2-server#2596, Graylog2/graylog2-server#2600

- Use HTTP and HTTPS default ports for network settings. Graylog2/graylog2-server#2595, Graylog2/graylog2-server#2605

- Dashboard improvements. Graylog2/graylog2-server#2084, Graylog2/graylog2-server#2281, Graylog2/graylog2-server#2626

- Ensure that `password_secret` is at least 16 characters long. Graylog2/graylog2-server#2619, Graylog2/graylog2-server#2622

- Reduce production .js files sizes by 51%. Graylog2/graylog2-server#2617

- Allow `web_endpoint_uri` to be a relative URI. Graylog2/graylog2-server#2600, Graylog2/graylog2-server#2614

- Use default session attribute for principal. Graylog2/graylog2-server#2620, Graylog2/graylog2-server#2621

- Compile regex pattern for MetricFilter only once. Graylog2/graylog2-server#2637. Thank you again @mikkolehtisalo!

- Fix NPE in Indices#checkForReopened(IndexMetaData). Graylog2/graylog2-server#2628, Graylog2/graylog2-server#2635

- Mark message offset as committed in case of a decoding error. Graylog2/graylog2-server#2627, Graylog2/graylog2-server#2643

- Fix cloning streams and editing legacy stream rules. Graylog2/graylog2-server#2244, Graylog2/graylog2-server#2346, Graylog2/graylog2-server#2646

- Add back storing of index failures in MongoDB. Graylog2/graylog2-server#2633, Graylog2/graylog2-server#2644

- Enable running Graylog REST API on different context path. Graylog2/graylog2-server#2603, Graylog2/graylog2-server#2397, Graylog2/graylog2-server#2634

- Add support for ECDSA private keys to KeyUtil. Graylog2/graylog2-server#2454, Graylog2/graylog2-server#2641

- Check for conflict of `rest_listen_uri` and `web_listen_uri`. Graylog2/graylog2-server#2634, Graylog2/graylog2-server#2652

- Remove uppercase example decorator before 2.1 final. Graylog2/graylog2-server#2588, Graylog2/graylog-plugin-pipeline-processor#73

- Make sure to include charset in getBytes and other relevant code sections. Graylog2/graylog2-server#2567, Graylog2/graylog2-server#2574

- Landing page greets with 2.0 in 2.1. Graylog2/graylog2-server#2579

- Run WebAppNotFoundResponseFilter later and for GET requests only. Graylog2/graylog2-server#2657, Graylog2/graylog2-server#2664

- Update dependencies. Graylog2/graylog2-server#2543, Graylog2/graylog2-server#2565

- Allowing to run REST API and web interface on same port. Graylog2/graylog2-server#2515

- Changing default to make REST API and web if to listen on same host/port. Graylog2/graylog2-server#2446, Graylog2/graylog2-server#2525

- Change plugin REST resource injection to use Class instances. Graylog2/graylog2-server#2492

- Validate that Elasticsearch home/data paths are readable. Graylog2/graylog2-server#2536, Graylog2/graylog2-server#2538

- Fix Version#fromClasspathProperties() when loading from JAR plugin. Graylog2/graylog2-server#2535

- Decorator UI Fixes. Graylog2/graylog2-server#2539

- Fix timing issue in MessageCountAlertCondition. Graylog2/graylog2-server#1704, Graylog2/graylog2-server#2382, Graylog2/graylog2-server#2546

- For HttpHeadersToken pass actual remote address. Graylog2/graylog2-server#2556

- Do not blindly override permission set of ldap users. Graylog2/graylog2-server#2516, Graylog2/graylog2-server#2529

- Display original date time of index ranges on hover. Graylog2/graylog2-server#2549, Graylog2/graylog2-server#2552

- Make it possible for plugins to request a shared class loader. Graylog2/graylog2-server#2436, Graylog2/graylog2-server#2508

- Fix REST API browser after changes to the PluginRestResource injection. Graylog2/graylog2-server#2550

- Make version comparison more lenient with pre-release versions. Graylog2/graylog2-server#2462, Graylog2/graylog2-server#2548

- Always trim message field values on Message class. Graylog2/graylog2-server#1936, Graylog2/graylog2-server#2510

- Fix search results console warnings. Graylog2/graylog2-server#2527

- Fix bulk import of Grok patterns. Graylog2/graylog2-server#2229, Graylog2/graylog2-server#2561

- Add helper method to add AuditLogAppenders. Graylog2/graylog2-server#2562

- Add explanation about the configuration file format. Graylog2/graylog2-server#2563

- Display session information, fix usability in user list, editing users. Graylog2/graylog2-server#2526, Graylog2/graylog2-server#2528, Graylog2/graylog2-server#2540, Graylog2/graylog2-server#2541

- Fix issues with app prefix. Graylog2/graylog2-server#2564, Graylog2/graylog2-server#2583

- Fix extractor and static fields creation in multi-node setups. Graylog2/graylog2-server#2580, Graylog2/graylog2-server#2584

- Authentication improvements. Graylog2/graylog2-server#2572, Graylog2/graylog2-server#2573

- Move Error Prone into default build profile. Graylog2/graylog2-server#2575

- Journal info command does not work. Graylog2/graylog2-server#2493 and Graylog2/graylog2-server#2495

- Search result highlighting color similar to white. Graylog2/graylog2-server#2480

- Cannot POST on Regex Tester (error 500). Graylog2/graylog2-server#2471 and Graylog2/graylog2-server#2472

- Middle-clicking to open new tab not working for some System menu items. Graylog2/graylog2-server#2468

- Json extractor should check for valid lucene keys. Graylog2/graylog2-server#2434 and Graylog2/graylog2-server#2481

- Elasticsearch Red cluster state triggered by index rotation under some conditions. Graylog2/graylog2-server#2371, Graylog2/graylog2-server#2429 and Graylog2/graylog2-server#2477

- Report syntax error when search query contains unescaped slash. Graylog2/graylog2-server#2372 and Graylog2/graylog2-server#2450

- Allowing path prefixes in `web_listen_uri` so web interface is accessible via path != "/". Graylog2/graylog2-server#2271 and Graylog2/graylog2-server#2440

- LDAP group mapping: stringwise comparison fails due to different DN formats. Graylog2/graylog2-server#1790 and Graylog2/graylog2-server#2484

- Json extractor prefix. Graylog2/graylog2-server#1646 and Graylog2/graylog2-server#2481

- LDAP users are shown a change password form. Graylog2/graylog2-server#2124, Graylog2/graylog2-server#2327 and Graylog2/graylog2-server#2485

- Switch message filters from polling to subscribing to change events. Graylog2/graylog2-server#2391 and Graylog2/graylog2-server#2496

- Make auth providers fully pluggable. Graylog2/graylog2-server#2232, Graylog2/graylog2-server#2367 and Graylog2/graylog2-server#2522

- Grok extractor: Allow returning only named captures. Graylog2/graylog2-server#1486 and Graylog2/graylog2-server#2500

- Attempt reading DSA key if RSA failed. Graylog2/graylog2-server#2503. Special thanks to @mikkolehtisalo!

- Fix session validation propagation. Graylog2/graylog2-server#2498

- A wrapper to protect from decompression bombs. Graylog2/graylog2-server#2339. Thank you again, @mikkolehtisalo!

- Make exceptions more useful by providing messages and context. Graylog2/graylog2-server#2478

- Decorate search results. Graylog2/graylog2-server#2408, Graylog2/graylog2-server#2482, Graylog2/graylog2-server#2499, Graylog2/graylog-plugin-pipeline-processor#41, Graylog2/graylog-plugin-pipeline-processor#43 and Graylog2/graylog-plugin-pipeline-processor#52

- Introduce CombinedProvider to sync actions and stores initialization. Graylog2/graylog2-server#2523

- Actually use the bluebird promise in FetchProvider. Graylog2/graylog2-server#2762

- Audit event cleanup. Graylog2/graylog2-server#2746

- Update documentation links. Graylog2/graylog2-server#2759

- Allow child elements in the search form. Graylog2/graylog2-server#2756

- Make key_prefix configuration optional. Graylog2/graylog2-server#2755, Graylog2/graylog2-server#2757

- Invalidating widget result cache cluster wide when a widget changes. Graylog2/graylog2-server#2732, Graylog2/graylog2-server#2745

- Correct documentation links in 'misc/graylog.conf'. Graylog2/graylog2-server#2747. Thank you @supahgreg!

- Throttle LB status if journal utilization is too high. Graylog2/graylog2-server#1100, Graylog2/graylog2-server#1952 and Graylog2/graylog2-server#2312. Thank you @mikkolehtisalo!

- TLS ciphers for inputs should probably be configurable. Graylog2/graylog2-server#2051.

- SelfSignedCertificate should migrate from sun.security.*. Graylog2/graylog2-server#2132 and Graylog2/graylog2-server#2316. Thank you @mikkolehtisalo!

- Fix formatting metric names including more than one namespace prefix. Graylog2/graylog2-server#2254 and Graylog2/graylog2-server#2425.

- Waiting for index range calculation before switching deflector alias. Graylog2/graylog2-server#2264 and Graylog2/graylog2-server#2278.

- Specify application.context. Graylog2/graylog2-server#2271 and Graylog2/graylog2-server#2440.

- Add handler for / in the Graylog REST API. Graylog2/graylog2-server#2376 and Graylog2/graylog2-server#2377.

- User preferred timezone not saved. Graylog2/graylog2-server#2393 and Graylog2/graylog2-server#2395.

- Unable to delete closed index. Graylog2/graylog2-server#2419 and Graylog2/graylog2-server#2437.

- Absolute search results in widget using wrong time. Graylog2/graylog2-server#2428 and Graylog2/graylog2-server#2452.

- Upgrade to Kafka 0.9.0.1. Graylog2/graylog2-server#1912.

- RestAccessLogFilter to use X-Forwarded-For set by trusted proxies. Graylog2/graylog2-server#1981. Thank you @mikkolehtisalo!

- Upgrade to Drools 6.4.0.Final. Graylog2/graylog2-server#2106.

- Stream Rule Titles. Graylog2/graylog2-server#2244.

- Improve search with no results page. Graylog2/graylog2-server#2253.

- Refactor Version class to use com.github.zafarkhaja.semver.Version. Graylog2/graylog2-server#2275.

- Alert condition titles. Graylog2/graylog2-server#2282.

- Upgrade to Jackson 2.7.4. Graylog2/graylog2-server#2304.

- Support changes for pipeline processor simulator. Graylog2/graylog2-server#2320.

- Add dependency on jna to fix chatty Elasticseach log message. Graylog2/graylog2-server#2342.

- Interfaces and simple implementations of an audit log. Graylog2/graylog2-server#2344.

- Do not init available alarm callback types, fetch them explicitly. Graylog2/graylog2-server#2353.

- Move custom analyzer into index template. Graylog2/graylog2-server#2354.

- Remove automatic private key/certificate generation. Graylog2/graylog2-server#2355.

- Improved feedback. Graylog2/graylog2-server#2357.

- Longer retention interval for journal tests. Graylog2/graylog2-server#2388.

- Remove `elasticsearch_discovery_zen_ping_multicast_enabled` setting. Graylog2/graylog2-server#2394.

- Fix unrequested refresh of configuration forms/Reset configuration forms on cancel. Graylog2/graylog2-server#2399.

- Web If: Updating a few dependencies which are safe to update. Graylog2/graylog2-server#2407.

- Added Information for journal partitions. Graylog2/graylog2-server#2412.

- Fix memory problems with webpack-dev-server in development mode. Graylog2/graylog2-server#2433.

- Remove `_ttl` in index mapping. Graylog2/graylog2-server#2435.

- Add raw message loader. Graylog2/graylog2-server#2438.

- Extracting our customized ESLint config into separate module. Graylog2/graylog2-server#2441.

- Remove deprecated MongoDB metrics reporter. Graylog2/graylog2-server#2443.

- Allow access to MongoDatabase in MongoConnection. Graylog2/graylog2-server#2444.

- Add some useful FindBugs plugins. Graylog2/graylog2-server#2447.

- Proxies deflector cycle call to make it available on every node. Graylog2/graylog2-server#2448.

Collector sidecar plugin

- Return updated configuration after changing configuration name

- Prevent crashes when failed to propagate state to the server

- Improve compatibility with old API

- Display collector IP address. Graylog2/graylog-plugin-collector#9

- Ability to clone collector configuration. Graylog2/graylog-plugin-collector#10

- NXLog GELF/TLS input should work without cert files. Graylog2/graylog-plugin-collector#13

- Add `tail_files` option

- Expand verbatim text area if value is present

- Validation improvements

- Add buffer option to NXLog outputs

- Make defaults compatible with Windows hosts

- Add support for Beats. Filebeat, Winlogbeat.

- Beats binaries are bundled with the Collector-Sidecar package

- Improve server side validation. Graylog2/graylog2-server#2247 and Graylog2/graylog-plugin-collector#7.

- Add NXlog GELF TCP and TCP/TLS output

- Add support to clone input, outputs and snippets

- Optionally display collector status information in web interface

- Optionally display log directory listing on status page

- If no node-id is given use the hostname as identification

- Linux distribution is detected and can be used in Snippet template

- Silent install on Windows works now

- Collector log files are now auto-rotated

- Collector processes are supervised and restarted on crashes

- NXlog Inputs and Outputs support free text configuration

- Fix web plugin loading on IE 11

Pipeline processor plugin

- Add parse error handler for precompute args failures. Graylog2/graylog-plugin-pipeline-processor#84, Graylog2/graylog-plugin-pipeline-processor#93

- Add support for DateTime comparison. Graylog2/graylog-plugin-pipeline-processor#86, Graylog2/graylog-plugin-pipeline-processor#92

- Make some small UI changes around RuleHelper. Graylog2/graylog-plugin-pipeline-processor#90

- Use shared classloader so other plugins can contribute functions. Graylog2/graylog-plugin-pipeline-processor#81, Graylog2/graylog-plugin-pipeline-processor#94

- UI improvements. Graylog2/graylog2-server#2683, Graylog2/graylog-plugin-pipeline-processor#83

- Unregister PipelineInterpreter from event bus . Graylog2/graylog-plugin-pipeline-processor#79

- Use find in the regex function. Graylog2/graylog-plugin-pipeline-processor#35, Graylog2/graylog-plugin-pipeline-processor#88

- Dynamic function list. Graylog2/graylog-plugin-pipeline-processor#89

- Unresolved functions not properly handled. Graylog2/graylog-plugin-pipeline-processor#24, Graylog2/graylog-plugin-pipeline-processor#25

- Unwrap JsonNode values. Graylog2/graylog-plugin-pipeline-processor#68, Graylog2/graylog-plugin-pipeline-processor#72

- Add optional prefix/suffix to `set_fields` functions. Graylog2/graylog-plugin-pipeline-processor#74, Graylog2/graylog-plugin-pipeline-processor#75

- Add key-value parsing function. Graylog2/graylog-plugin-pipeline-processor#38, Graylog2/graylog-plugin-pipeline-processor#77

- Allow selection of an input ID for the simulation message. Graylog2/graylog2-server#2610, Graylog2/graylog2-server#2650, Graylog2/graylog-plugin-pipeline-processor#78

- Support "only named captures" for pipeline grok function. Graylog2/graylog-plugin-pipeline-processor#59, Graylog2/graylog-plugin-pipeline-processor#65, Graylog2/graylog2-server#2566, Graylog2/graylog2-server#2577

- Make conversion functions more consistent. Graylog2/graylog2-server#63, Graylog2/graylog2-server#64

- Unescape string literals before using them. Graylog2/graylog-plugin-pipeline-processor#47

- Add `rename_field` function. Graylog2/graylog-plugin-pipeline-processor#50

- Allow null matcher group values in regex function. Graylog2/graylog-plugin-pipeline-processor#49

- Fix 500 error during simulation. Graylog2/graylog-plugin-pipeline-processor#51

- IpAddressConversion caught wrong exception. Graylog2/graylog-plugin-pipeline-processor#32

- Add syslog-related functions. Graylog2/graylog-plugin-pipeline-processor#19.

- Add `concat()` function. Graylog2/graylog-plugin-pipeline-processor#20.

- NPE during preProcessArgs using Grok pattern. Graylog2/graylog-plugin-pipeline-processor#24 and Graylog2/graylog-plugin-pipeline-processor#26.

- Streams without connections stay visible. Graylog2/graylog2-server#2322.
- Add pipeline simulator. Graylog2/graylog-plugin-pipeline-processor#34, Graylog2/graylog-plugin-pipeline-processor#36 and Graylog2/graylog-plugin-pipeline-processor#42.
- Fix page size in function list. Graylog2/graylog-plugin-pipeline-processor#97

28.37 Graylog 2.0.3

Released: 2016-06-20

https://www.graylog.org/blog/58-graylog-v2-0-3-released

Improvements

- Make Message#getStreamIds() more reliable. Graylog2/graylog2-server#2378
- Disabling a configured proxy for requests to localhost/127.0.0.1/::1. Graylog2/graylog2-server#2305

Bug fixes

- Update search query on auto refresh Graylog2/graylog2-server#2385 Graylog2/graylog2-server#2379
- Fix permission checks for non admin users Graylog2/graylog2-server#2366 Graylog2/graylog2-server#2358
- Fix display of total count of indices. Graylog2/graylog2-server#2365 Graylog2/graylog2-server#2359
- Fix base URI for API documentation Graylog2/graylog2-server#2362 Graylog2/graylog2-server#2360
- Fix link to API Browser on Node pages Graylog2/graylog2-server#2361 Graylog2/graylog2-server#2360
- Calculate keyword from and to values on the fly Graylog2/graylog2-server#2335 Graylog2/graylog2-server#2301
- Make MemoryAppender thread-safe Graylog2/graylog2-server#2307 Graylog2/graylog2-server#2302
- Use right metrics to display buffer usage Graylog2/graylog2-server#2300 Graylog2/graylog2-server#2299
- Check if props actually contain configuration fields before copying them Graylog2/graylog2-server#2298 Graylog2/graylog2-server#2297

28.38 Graylog 2.0.2

Released: 2016-05-27

https://www.graylog.org/blog/57-graylog-v2-0-2-released

Improvements

- Improved user form. Graylog2/graylog2-server#2261
- Improved logging of plugin list on server startup. Graylog2/graylog2-server#2290
- Forbid empty passwords when using LDAP. Graylog2/graylog2-server#2214 Graylog2/graylog2-server#2283
- Improved metrics page. Graylog2/graylog2-server#2250 Graylog2/graylog2-server#2255
- Improved search histogram resolution auto selection. Graylog2/graylog2-server#2148 Graylog2/graylog2-server#2289
- Improved cluster overview page. Graylog2/graylog2-server#2291

Bug Fixes

- Fixed concurrency issue with Drools. Graylog2/graylog2-server#2119 Graylog2/graylog2-server#2188 Graylog2/graylog2-server#2231

- Fixed problems with Internet Explorer. Graylog2/graylog2-server#2246

- Fixed issues with old dashboards. Graylog2/graylog2-server#2262 Graylog2/graylog2-server#2163

- Fixed changing log levels via REST API. Graylog2/graylog2-server#1904 Graylog2/graylog2-server#2277

- Fixed plugin inter-dependencies by using one class loader for all plugins. Graylog2/graylog2-server#2280

Plugin: Pipeline Processor

- Add syslog related rule functions. Graylog2/graylog-plugin-pipeline-processor#19

- Add concat rule functions. Graylog2/graylog-plugin-pipeline-processor#20

- Fixed problem with IP address function. Graylog2/graylog-plugin-pipeline-processor#28 Graylog2/graylog-plugin-pipeline-processor#32

- Properly unescape strings in raw literals. Graylog2/graylog-plugin-pipeline-processor#30 Graylog2/graylog-plugin-pipeline-processor#31

28.39 Graylog 2.0.1

Released: 2016-05-11

https://www.graylog.org/blog/56-graylog-v2-0-1-released

Improvements

- Improved session handling. Graylog2/graylog2-server#2157

- Included UPGRADING file in the build artifact. Graylog2/graylog2-server#2170

- Added rotation/retention settings back to the config file. Graylog2/graylog2-server#2181

- Improved proxy setup configuration settings. Graylog2/graylog2-server#2156

- Forbid wildcard host in *rest_transport_uri*. Graylog2/graylog2-server#2205

- Improved robustness for unreachable nodes. Graylog2/graylog2-server#2206

- Use a more lightweight API to get all index names and aliases. Graylog2/graylog2-server#2194 Graylog2/graylog2-server#2210

Bug Fixes

- Fixed some documentation links.

- Fixed inverted stream rules. Graylog2/graylog2-server#2160 Graylog2/graylog2-server#2172

- Fixed swallowed LDAP authentication exception. Graylog2/graylog2-server#2176 Graylog2/graylog2-server#2178

- Fixed insecure handling of PID files. Thanks @juergenhoetzel! Graylog2/graylog2-server#2174

- Fixed alert conditions that have been created in Graylog 1.x. Graylog2/graylog2-server#2169 Graylog2/graylog2-server#2182

- Fixed setting of application context. Graylog2/graylog2-server#2191 Graylog2/graylog2-server#2208

- Fixed setting of custom Elasticsearch analyzer. Graylog2/graylog2-server#2209

- Fixed masking of password config values in the web interface. Graylog2/graylog2-server#2198 Graylog2/graylog2-server#2203

- Fixed URL handling. Graylog2/graylog2-server#2200 Graylog2/graylog2-server#2213

Plugin: Collector

- Rotate nxlog logfiles once a day by default.

- Add GELF TCP output for nxlog.

28.40 Graylog 2.0.0

Released: 2016-04-27

https://www.graylog.org/blog/55-announcing-graylog-v2-0-ga

Note: Please make sure to read the *Upgrade Guide* before upgrading to Graylog 2.0. There are breaking changes!

Feature Highlights

See the release announcement for details on the new features.

- Web interface no longer a separate process

- Support for Elasticsearch 2.x

- Live tail support

- Message Processing Pipeline

- Map Widget Plugin

- Collector Sidecar

- Streams filter UI

- Search for surrounding messages

- Query range limit

- Configurable query time ranges

- Archiving (commercial feature)

Bug Fixes

There have been lots of bug fixes since the 1.3 releases. We only list the ones that we worked on since the 2.0 alpha phase.

- Fixed issues with search page pagination and number of returned results: Graylog2/graylog2-server#1759, Graylog2/graylog2-server#1775, and Graylog2/graylog2-server#1802

- Avoid creating MongoDB collection multiple times: Graylog2/graylog2-server#1747

- Removed number of connected nodes in login page: Graylog2/graylog2-server#1732

- Fix dynamic search result histogram resolution: Graylog2/graylog2-server#1764

- Show overlay in Graylog web interface when Graylog server is not available: Graylog2/graylog2-server#1762

- Fix metric types: Graylog2/graylog2-server#1784

- Only load all metrics on demand: Graylog2/graylog2-server#1782

- Activate search refresh after selecting a refresh interval: Graylog2/graylog2-server#1796

- Fix circular dependencies: Graylog2/graylog2-server#1789

- Only render input forms when input type is available: Graylog2/graylog2-server#1798

- Document web interface configuration settings in graylog.conf. Graylog2/graylog2-server#1777

- Fix roles link to documentation. Graylog2/graylog2-server#1805

- Fix issue with field graphs. Graylog2/graylog2-server#1811

- Fix search result pagination. Graylog2/graylog2-server#1812

- Fix add to query button on quick values. Graylog2/graylog2-server#1797

- Fix URL to Graylog marketplace on content pack export page. Graylog2/graylog2-server#1817

- Fix elasticsearch node name for the Graylog client node. Graylog2/graylog2-server#1814 and Graylog2/graylog2-server#1820

- Fix widget sorting for dashboards.

- Use _ as default key separator in JSON Extractor. Graylog2/graylog2-server#1841

- Clarify that Graylog Collector needs access to rest_listen_uri. Graylog2/graylog2-server#1847

- Fix potential memory leak in GELF UDP handler. Graylog2/graylog2-server#1857 Graylog2/graylog2-server#1862

- Fix user with correct permissions not allowed to view stream: Graylog2/graylog2-server#1887, Graylog2/graylog2-server#1902

- Make pattern to check Graylog-managed indices stricter: Graylog2/graylog2-server#1882, Graylog2/graylog2-server#1888

- Fix throughput counter: Graylog2/graylog2-server#1876

- Fix replay search link in dashboards: Graylog2/graylog2-server#1835

- Render server unavailable page more reliably: Graylog2/graylog2-server#1867

- Fix build issue with maven. Graylog2/graylog2-server#1907 (Thanks @gitfrederic)

- Fix username in REST API access logs. Graylog2/graylog2-server#1815 Graylog2/graylog2-server#1918 (Thanks @mikkolehtisalo)

- Fix alert annotations in message histogram. Graylog2/graylog2-server#1921

- Fix problem with automatic input form reload. Graylog2/graylog2-server#1870 Graylog2/graylog2-server#1929

- Fix asset caching. Graylog2/graylog2-server#1924 Graylog2/graylog2-server#1930

- Fix issue with cursor jumps in the search bar. Graylog2/graylog2-server#1911

- Fix import of Graylog 1.x extractors. Graylog2/graylog2-server#1831 Graylog2/graylog2-server#1937

- Field charts will now use the stream and time range of the current search. Graylog2/graylog2-server#1785 Graylog2/graylog2-web-interface#1620 Graylog2/graylog2-web-interface#1618 Graylog2/graylog2-web-interface#1485 Graylog2/graylog2-server#1938

- Improve browser validations. Graylog2/graylog2-server#1885

- Fix Internet Explorer support. Graylog2/graylog2-server#1935

- Fix issue where a user was logged out when accessing an unauthorized resource. Graylog2/graylog2-server#1944

- Fix issue with surrounding search. Graylog2/graylog2-server#1946

- Fix problem deleting dashboard widget where the plugin got removed. Graylog2/graylog2-server#1943

- Fix permission issue on user edit page. Graylog2/graylog2-server#1964

- Fix histogram time range selection via mouse. Graylog2/graylog2-server#1895

- Fix problems with duplicate Reflux store instances. Graylog2/graylog2-server#1967

- Create PID file earlier in the startup process. Graylog2/graylog2-server#1969 Graylog2/graylog2-server#1978

- Fix content type detection for static assets. Graylog2/graylog2-server#1982 Graylog2/graylog2-server#1983

- Fix caching of static assets. Graylog2/graylog2-server#1982 Graylog2/graylog2-server#1983

- Show error message on malformed search query. Graylog2/graylog2-server#1896

- Fix parsing of GELF chunks. Graylog2/graylog2-server#1986

- Fix problems editing reader users profile. Graylog2/graylog2-server#1984 Graylog2/graylog2-server#1987

- Fix problem with lost extractors and static fields on input update. Graylog2/graylog2-server#1988 Graylog2/graylog2-server#1923

- Improve fetching cluster metrics to avoid multiple HTTP calls. Graylog2/graylog2-server#1974 Graylog2/graylog2-server#1990

- Properly handle empty messages. Graylog2/graylog2-server#1584 Graylog2/graylog2-server#1995

- Add 100-Continue support to HTTP inputs. Graylog2/graylog2-server#1939 Graylog2/graylog2-server#1998

- Fix setting dashboard as start page for reader users. Graylog2/graylog2-server#2005

- Allow dots (".") in LDAP group name mappings. Graylog2/graylog2-server#1458 Graylog2/graylog2-server#2009

- Update user edit form when username changes. Graylog2/graylog2-server#2000

- Fix issue with permissions in user form. Graylog2/graylog2-server#1989

- Update extractor example when message is loaded. Graylog2/graylog2-server#1957 Graylog2/graylog2-server#2013

- Disable log4j2 shutdown hooks to avoid exception on shutdown. Graylog2/graylog2-server#1795 Graylog2/graylog2-server#2015

- Fix styling issue with map widget. Graylog2/graylog2-server#2003

- Fix openstreetmap URL in map widget. Graylog2/graylog2-server#1994

- Fix problem with collector heartbeat validation. Graylog2/graylog2-server#2002 Graylog2/graylog2-web-interface#1726 Graylog2/graylog-plugin-collector#3

- Remove unused command line parameters. Graylog2/graylog2-server#1977

- Fixed timezone issues for date time processing in JSON parser. Graylog2/graylog2-server#2007

- Fixed JavaScript error with field truncation. Graylog2/graylog2-server#2025

- Fixed redirection if user is not authorized. Graylog2/graylog2-server#1985 Graylog2/graylog2-server#2024

- Made changing the sort order in search result table work again. Graylog2/graylog2-server#2028 Graylog2/graylog2-server#2031

- Performance improvements on "System/Indices" page. Graylog2/graylog2-server#2017

- Fixed content-type settings for static assets. Graylog2/graylog2-server#2052

- Fixed return code for invalid input IDs. Graylog2/graylog2-server#1718 Graylog2/graylog2-server#1767

- Improved field analyzer UI. Graylog2/graylog2-server#2022 Graylog2/graylog2-server#2023

- Fixed login with LDAP user. Graylog2/graylog2-server#2045 Graylog2/graylog2-server#2046 Graylog2/graylog2-server#2069

- Fixed issue with bad message timestamps to avoid data loss. Graylog2/graylog2-server#2064 Graylog2/graylog2-server#2065

- Improved handling of Elasticsearch indices. Graylog2/graylog2-server#2058 Graylog2/graylog2-server#2062

- Extractor form improvements for JSON and Grok extractors. Graylog2/graylog2-server#1883 Graylog2/graylog2-server#2020

- Used search refresh to refresh field statistics. Graylog2/graylog2-server#1961 Graylog2/graylog2-server#2068

- Fixed clicking zoom button in quick values. Graylog2/graylog2-server#2040 Graylog2/graylog2-server#2067

- Web interface styling improvements.

- Replaced . in message field keys with a _ for ES 2.x compatibility. Graylog2/graylog2-server#2078

- Fixed unprocessed journal messages reload in node list. Graylog2/graylog2-server#2083

- Fixed problems with stale sessions on the login page. Graylog2/graylog2-server#2073 Graylog2/graylog2-server#2059 Graylog2/graylog2-server#1891

- Fixed issue with index retention strategies. Graylog2/graylog2-server#2100

- Fixed password change form. Graylog2/graylog2-server#2103 Graylog2/graylog2-server#2105

- Do not show search refresh controls on the sources page. Graylog2/graylog2-server#1821 Graylog2/graylog2-server#2104

- Wait for index being available before calculating index range. Graylog2/graylog2-server#2061 Graylog2/graylog2-server#2098

- Fixed issue with sorting extractors. Graylog2/graylog2-server#2086 Graylog2/graylog2-server#2088

- Improve DataTable UI component. Graylog2/graylog-plugin-pipeline-processor#11

- Move TCP keepalive setting into AbstractTcpTransport to simplify input development. Graylog2/graylog2-server#2112

- Fixed issue with Elasticsearch index template update. Graylog2/graylog2-server#2089 Graylog2/graylog2-server#2097

- Ensure that tmpDir is writable when generating self-signed certs in TCP transports. Graylog2/graylog2-server#2054 Graylog2/graylog2-server#2096

- Fixed default values for plugin configuration forms. Graylog2/graylog2-server#2108 Graylog2/graylog2-server#2114

- Dashboard usability improvements. Graylog2/graylog2-server#2093

- Include default values in pluggable entities forms. Graylog2/graylog2-server#2122

- Ignore empty authentication tokens in LdapUserAuthenticator. Graylog2/graylog2-server#2123

- Add REST API authentication and permissions. Graylog2/graylog-plugin-pipeline-processor#15

- Require authenticated user in REST resources. Graylog2/graylog-plugin-pipeline-processor#14

- Lots of UI improvements in the web interface. Graylog2/graylog2-server#2136

- Fixed link to REST API browser. Graylog2/graylog2-server#2133

- Fixed CSV export skipping first chunk. Graylog2/graylog2-server#2128

- Fixed updating content packs. Graylog2/graylog2-server#2138 Graylog2/graylog2-server#2141

- Added missing 404 page. Graylog2/graylog2-server#2139

28.41 Graylog 1.3.4

Released: 2016-03-16

https://www.graylog.org/blog/49-graylog-1-3-4-is-now-available

- Fix security issue which allowed redirecting users to arbitrary sites on login Graylog2/graylog2-web-interface#1729
- Fix issue with time-based index rotation strategy Graylog2/graylog2-server#725 Graylog2/graylog2-server#1693
- Fix issue with `IndexFailureServiceImpl` Graylog2/graylog2-server#1747
- Add default Content-Type to `GettingStartedResource` Graylog2/graylog2-server#1700
- Improve OS platform detection Graylog2/graylog2-server#1737
- Add prefixes `GRAYLOG_` (environment variables) and `graylog.` (system properties) for overriding configuration settings Graylog2/graylog2-server@48ed88d
- Fix URL to Graylog Marketplace on Extractor/Content Pack pages Graylog2/graylog2-server#1817
- Use monospace font on message values Graylog2/graylog2-web-interface@3cce368

28.42 Graylog 1.3.3

Released: 2016-01-14

https://www.graylog.org/graylog-1-3-3-is-now-available/

- Absolute and relative time spans give different results Graylog2/graylog2-server#1572 Graylog2/graylog2-server#1463 Graylog2/graylog2-server#1672 Graylog2/graylog2-server#1679
- Search result count widget not caching Graylog2/graylog2-server#1640 Graylog2/graylog2-server#1681
- Field Value Condition Alert, does not permit decimal values Graylog2/graylog2-server#1657
- Correctly handle null values in nested structures in JsonExtractor Graylog2/graylog2-server#1676 Graylog2/graylog2-server#1677
- Add `Content-Type` and `X-Graylog2-No-Session-Extension` to CORS headers Graylog2/graylog2-server#1682 Graylog2/graylog2-server#1685
- Discard Message Output Graylog2/graylog2-server#1688

28.43 Graylog 1.3.2

Released: 2015-12-18

https://www.graylog.org/graylog-1-3-2-is-now-available/

- Deserializing a blacklist filter (`FilterDescription`) leads to `StackOverflowError` Graylog2/graylog2-server#1641

28.44 Graylog 1.3.1

Released: 2015-12-17

https://www.graylog.org/graylog-1-3-1-is-now-available/

- Add option to AMQP transports to bind the queue to the exchange Graylog2/graylog2-server#1599 Graylog2/graylog2-server#1633
- Install a Graylog index template instead of set mappings on index creation Graylog2/graylog2-server#1624 Graylog2/graylog2-server#1628

28.45 Graylog 1.3.0

Released: 2015-12-09

https://www.graylog.org/graylog-1-3-ga-is-ready/

- Allow index range calculation for a single index. Graylog2/graylog2-server#1451 Graylog2/graylog2-server#1455
- Performance improvements for index ranges.
- Make internal server logs accessible via REST API. Graylog2/graylog2-server#1452
- Make specific configuration values accessible via REST API. Graylog2/graylog2-server#1484
- Added Replace Extractor. Graylog2/graylog2-server#1485
- Added a default set of Grok patterns. Graylog2/graylog2-server#1495
- Log operating system details on server startup. Graylog2/graylog2-server#1244 Graylog2/graylog2-server#1553
- Allow reader users to set a dashboard as start page. Graylog2/graylog2-web-interface#1681
- Auto content pack loader – download and install content packs automatically
- Appliance pre-configured for log ingestion and analysis
- Show a getting started guide on first install. Graylog2/graylog2-web-interface#1662
- Include role permissions in "/roles/{rolename}/members" REST API endpoint. Graylog2/graylog2-server#1549
- Fixed NullPointerException in GELF output. Graylog2/graylog2-server#1538
- Fixed NullPointerException in GELF input handling. Graylog2/graylog2-server#1544
- Use the root user's timezone for LDAP users by default. Graylog2/graylog2-server#1000 Graylog2/graylog2-server#1554
- Fix display of JSON messages. Graylog2/graylog2-web-interface#1686

- Improve search robustness with missing Elasticsearch indices. Graylog2/graylog2-server#1547 Graylog2/graylog2-server#1533

- Fixed race condition between index creation and index mapping configuration. Graylog2/graylog2-server#1502 Graylog2/graylog2-server#1563

- Fixed concurrency problem in GELF input handling. Graylog2/graylog2-server#1561

- Fixed issue with widget value calculation. Graylog2/graylog2-server#1588

- Do not extend user sessions when updating widgets. Graylog2/graylog2-web-interface#1655

- Fixed compatibility mode for Internet Explorer. Graylog2/graylog2-web-interface#1661 Graylog2/graylog2-web-interface#1668

- Fixed whitespace issue in extractor example. Graylog2/graylog2-web-interface#1650

- Fixed several issues on the indices page. Graylog2/graylog2-web-interface#1691 Graylog2/graylog2-web-interface#1692

- Fixed permission issue for stream alert management. Graylog2/graylog2-web-interface#1659

- Fixed deletion of LDAP group mappings when updating LDAP settings. Graylog2/graylog2-server#1513

- Fixed dangling role references after deleting a role Graylog2/graylog2-server#1608

- Support LDAP Group Mapping for Sun Directory Server (new since beta.2) Graylog2/graylog2-server#1583

28.46 Graylog 1.2.2

Released: 2015-10-27

https://www.graylog.org/graylog-1-2-2-is-now-available/

- Fixed a whitespace issue in the extractor UI. Graylog2/graylog2-web-interface#1650

- Fixed the index description on the indices page. Graylog2/graylog2-web-interface#1653

- Fixed a memory leak in the GELF UDP handler code. (Analysis and fix contributed by @lightpriest and @onyx-master on GitHub. Thank you!) Graylog2/graylog2-server#1462, Graylog2/graylog2-server#1488

- Improved the LDAP group handling code to handle more LDAP setups. Graylog2/graylog2-server#1433, Graylog2/graylog2-server#1453, Graylog2/graylog2-server#1491, Graylog2/graylog2-server#1494

- Fixed email alerts for users with multiple email addresses. (LDAP setups) Graylog2/graylog2-server#1439, Graylog2/graylog2-server#1492

- Improve index range handling performance. Graylog2/graylog2-server#1465, Graylog2/graylog2-server#1493

- Fixed JSON extractor with null values. Graylog2/graylog2-server#1475, Graylog2/graylog2-server#1505

- Fixed role assignment when updating user via REST API. Graylog2/graylog2-server#1456, Graylog2/graylog2-server#1507

28.47 Graylog 1.2.1

Released: 2015-09-22

https://www.graylog.org/graylog-1-2-1-is-now-available/

- Fixed various issues around importing and applying content packs Graylog2/graylog2-server#1423, Graylog2/graylog2-server#1434, Graylog2/graylog2-web-interface#1605, Graylog2/graylog2-web-interface#1614

- Fixed loading existing alarm callbacks that had been created with Graylog 1.0.x or earlier Graylog2/graylog2-server#1428

- Fixed compatibility problem with Elasticsearch 1.5.x and earlier Graylog2/graylog2-server#1426

- Fixed handling of statistical functions in field graphs Graylog2/graylog2-web-interface#1604

- Use correct title when adding quick values to a dashboard Graylog2/graylog2-web-interface#1603

28.48 Graylog 1.2.0

Released: 2015-09-14

https://www.graylog.org/announcing-graylog-1-2-ga-release-includes-30-new-features/

- Make sure existing role assignments survive on LDAP account sync. Graylog2/graylog2-server#1405 | Graylog2/graylog2-server#1406

- Use memberOf query for ActiveDirectory to speed up LDAP queries. Graylog2/graylog2-server#1407

- Removed disable_index_range_calculation configuration option. Graylog2/graylog2-server#1411

- Avoid potentially long-running Elasticsearch cluster-level operations by only saving an index range if it actually changed. Graylog2/graylog2-server#1412

- Allow editing the roles of LDAP users. Graylog2/graylog2-web-interface#1598

- Improved quick values widget. Graylog2/graylog2-web-interface#1487

28.49 Graylog 1.2.0-rc.4

Released: 2015-09-08

https://www.graylog.org/announcing-graylog-1-2-rc-4/

- Deprecated MongoDB storage of internal metrics feature.

- Added customizable LDAP filter for user groups lookup. Graylog2/graylog2-server#951

- Allow usage of count and cardinality statistical functions in dashboard widgets. Graylog2/graylog2-server#1376

- Disabled index range recalculation on every index rotation. Graylog2/graylog2-server#1388

- Added automatic migration of user permissions to admin or reader roles. Graylog2/graylog2-server#1389

- Fixed widget problem with invalid timestamps. Graylog2/graylog2-web-interface#1390

- Added config option to enable TLS certificate validation in REST client. Graylog2/graylog2-server#1393

- Fixed rule matching issue in stream routing engine. Graylog2/graylog2-server#1397

- Changed default titles for stream widgets. Graylog2/graylog2-web-interface#1476

- Changed data filters to be case insensitive. Graylog2/graylog2-web-interface#1585

- Improved padding for stack charts. Graylog2/graylog2-web-interface#1568

- Improved resiliency when Elasticsearch is not available. Graylog2/graylog2-web-interface#1518

- Redirect to user edit form after updating a user. Graylog2/graylog2-web-interface#1588

- Improved dashboard widgets error handling. Graylog2/graylog2-web-interface#1590

- Fixed timing issue in streams UI. Graylog2/graylog2-web-interface#1490

- Improved indices overview page. Graylog2/graylog2-web-interface#1593

- Fixed browser back button behavior. Graylog2/graylog2-web-interface#1594

- Fixed accidental type conversion for number configuration fields in alarmcallback plugins. Graylog2/graylog2-web-interface#1596

- Fixed data type problem for extracted timestamps via grok. Graylog2/graylog2-server#1403

28.50 Graylog 1.2.0-rc.2

Released: 2015-08-31

https://www.graylog.org/announcing-graylog-1-2-rc/

- Implement global Elasticsearch timeout and add `elasticsearch_request_timeout` configuration setting. Graylog2/graylog2-server#1220

- Fixed lots of documentation links. Graylog2/graylog2-server#1238

- Groovy shell server removed. Graylog2/graylog2-server#1266

- Lots of index range calculation fixes. Graylog2/graylog2-server#1274

- New Raw AMQP input. Graylog2/graylog2-server#1280

- New Syslog AMQP input. Graylog2/graylog2-server#1280

- Updated bundled Elasticsearch to 1.7.1.

- The fields in configuration dialogs for inputs and outputs are now ordered. Graylog2/graylog2-server#1282

- Allow server startup without working Elasticsearch cluster. Graylog2/graylog2-server#1136, Graylog2/graylog2-server#1289

- Added OR operator to stream matching. Graylog2/graylog2-server#1292, Graylog2/graylog2-web#1552

- New stream router engine with better stream matching performance. Graylog2/graylog2-server#1305, Graylog2/graylog2-server#1309

- Grok pattern import/export support for content packs. Graylog2/graylog2-server#1300, Graylog2/graylog2-web#1527

- Added MessageListCodec interface for codec implementations that can decode multiple messages from one raw message. Graylog2/graylog2-server#1307

- Added keepalive configuration option for all TCP transports. Graylog2/graylog2-server#1287, Graylog2/graylog2-server#1318

- Support for roles and LDAP groups. Graylog2/graylog2-server#1321, Graylog2/graylog2-server#951

- Added timezone configuration option to date converter. Graylog2/graylog2-server#1320, Graylog2/graylog2-server#1324

- Added alarmcallback history feature. Graylog2/graylog2-server#1313, Graylog2/graylog2-web#1537

- Added more configuration options to GELF output. (TCP settings, TLS support) Graylog2/graylog2-server#1337, Graylog2/graylog2-server#979

- Store timestamp and some other internal fields in Elasticsearch as doc values. Removed "elastic-search_store_timestamps_as_doc_values" option from configuration file. Graylog2/graylog2-server#1335, Graylog2/graylog2-server#1342

- Added TLS support for GELF HTTP input. Graylog2/graylog2-server#1348

- Added JSON extractor. Graylog2/graylog2-server#632, Graylog2/graylog2-server#1355, Graylog2/graylog2-web#1555

- Added support for TLS client certificate authentication to all TCP based inputs. Graylog2/graylog2-server#1357, Graylog2/graylog2-server#1363

- Added stacked chart widget. Graylog2/graylog2-server#1284, Graylog2/graylog2-web#1513

- Added cardinality option to field histograms. Graylog2/graylog2-web#1529, Graylog2/graylog2-server#1303

- Lots of dashboard improvements. Graylog2/graylog2-web#1550

- Replaced Gulp with Webpack. Graylog2/graylog2-web#1548

- Updated to Play 2.3.10.

28.51 Graylog 1.1.6

Released: 2015-08-06

https://www.graylog.org/graylog-1-1-6-released/

- Fix edge case in `SyslogOctetCountFrameDecoder` which caused the Syslog TCP input to reset connections (Graylog2/graylog2-server#1105, Graylog2/graylog2-server#1339)

- Properly log errors in the Netty channel pipeline (Graylog2/graylog2-server#1340)

- Prevent creation of invalid alert conditions (Graylog2/graylog2-server#1332)

- Upgrade to Elasticsearch 1.6.2

28.52 Graylog 1.1.5

Released: 2015-07-27

https://www.graylog.org/graylog-1-1-5-released/

- Improve handling of exceptions in the JournallingMessageHandler (Graylog2/graylog2-server#1286)

- Upgrade to Elasticsearch 1.6.1 (Graylog2/graylog2-server#1312)

- Remove hard-coded limit for UDP receive buffer size (Graylog2/graylog2-server#1290)

- Ensure that `elasticsearch_index_prefix` is lowercase (commit 2173225)

- Add configuration option for time zone to `Date` converter (Graylog2/graylog2-server#1320)

- Fix NPE if the disk journal is disabled on a node (Graylog2/graylog2-web-interface#1520)

- Statistic and Chart error: Adding time zone offset caused overflow (Graylog2/graylog2-server#1257)

- Ignore stream alerts and throughput on serialize (Graylog2/graylog2-server#1309)

- Fix dynamic keyword time-ranges for dashboard widgets created from content packs (Graylog2/graylog2-server#1308)

- Upgraded Anonymous Usage Statistics plugin to version 1.1.1

28.53 Graylog 1.1.4

Released: 2015-06-30

https://www.graylog.org/graylog-v1-1-4-is-now-available/

- Make heartbeat timeout option for AmqpTransport optional. Graylog2/graylog2-server#1010

- Export as CSV on stream fails with "Invalid range type provided." Graylog2/graylog2-web-interface#1504

28.54 Graylog 1.1.3

Released: 2015-06-19

https://www.graylog.org/graylog-v1-1-3-is-now-available/

- Log error message early if there is a MongoDB connection error. Graylog2/graylog2-server#1249

- Fixed field content value alert condition. Graylog2/graylog2-server#1245

- Extend warning about SO_RCVBUF size to UDP inputs. Graylog2/graylog2-server#1243

- Scroll on button dropdowns. Graylog2/graylog2-web-interface#1477

- Normalize graph widget numbers before drawing them. Graylog2/graylog2-web-interface#1479

- Fix highlight result checkbox position on old Firefox. Graylog2/graylog2-web-interface#1440

- Unescape terms added to search bar. Graylog2/graylog2-web-interface#1484

- Load another message in edit extractor page not working. Graylog2/graylog2-web-interface#1488

- Reader users aren't able to export search results as CSV. Graylog2/graylog2-web-interface#1492

- List of streams not loaded on message details page. Graylog2/graylog2-web-interface#1496

28.55 Graylog 1.1.2

Released: 2015-06-10

https://www.graylog.org/graylog-v1-1-2-is-now-available/

- Get rid of NoSuchElementException if index alias doesn't exist. Graylog2/graylog2-server#1218

- Make Alarm Callbacks API compatible to Graylog 1.0.x again.   Graylog2/graylog2-server#1221, Graylog2/graylog2-server#1222, Graylog2/graylog2-server#1224

- Fixed issues with natural language parser for keyword time range. Graylog2/graylog2-server#1226

- Unable to write Graylog metrics to MongoDB Graylog2/graylog2-server#1228

- Unable to delete user. Graylog2/graylog2-server#1209

- Unable to unpause streams, despite editing permissions. Graylog2/graylog2-web-interface#1456

- Choose quick values widget size dynamically. Graylog2/graylog2-web-interface#1422

- Default field sort order is not guaranteed after reload. Graylog2/graylog2-web-interface#1436

- Toggling all fields in search list throws error and breaks pagination. Graylog2/graylog2-web-interface#1434

- Improve multi-line log messages support. Graylog2/graylog2-web-interface#612

- NPE when clicking a message from a deleted input on a stopped node. Graylog2/graylog2-web-interface#1444

- Auto created search syntax must use quotes for values with whitespaces in them. Graylog2/graylog2-web-interface#1448

- Quick Values doesn't update for new field. Graylog2/graylog2-web-interface#1438

- New Quick Values list too large. Graylog2/graylog2-web-interface#1442

- Unloading referenced alarm callback plugin breaks alarm callback listing. Graylog2/graylog2-web-interface#1450

- Add to search button doesn't work as expected for "level" field. Graylog2/graylog2-web-interface#1453

- Treat "*" query as empty query. Graylog2/graylog2-web-interface#1420

- Improve title overflow on widgets. Graylog2/graylog2-web-interface#1430

- Convert NaN to 0 on histograms. Graylog2/graylog2-web-interface#1417

- "<>" values in fields are unescaped and don't display in Quick Values. Graylog2/graylog2-web-interface#1455

- New quickvalues are not showing number of terms. Graylog2/graylog2-web-interface#1411

- Default index for split & index extractor results in an error. Graylog2/graylog2-web-interface#1464

- Improve behaviour when field graph fails to load. Graylog2/graylog2-web-interface#1276

- Unable to unpause streams, despite editing permissions. Graylog2/graylog2-web-interface#1456

- Wrong initial size of quick values pie chart. Graylog2/graylog2-web-interface#1469

- Problems refreshing data on quick values pie chart. Graylog2/graylog2-web-interface#1470

- Ignore streams with no permissions on message details. Graylog2/graylog2-web-interface#1472

28.56 Graylog 1.1.1

Released: 2015-06-05

https://www.graylog.org/graylog-v1-1-1-is-now-available/

- Fix problem with missing alarmcallbacks. Graylog2/graylog2-server#1214

- Add additional newline between messages to alert email. Graylog2/graylog2-server#1216

- Fix incorrect index range calculation. Graylog2/graylog2-server#1217, Graylog2/graylog2-web-interface#1266

- Fix sidebar auto-height on old Firefox versions. Graylog2/graylog2-web-interface#1410

- Fix "create one now" link on stream list page. Graylog2/graylog2-web-interface#1424

- Do not update StreamThroughput when unmounted. Graylog2/graylog2-web-interface#1428

- Fix position of alert annotations in search result histogram. Graylog2/graylog2-web-interface#1421

- Fix NPE when searching. Graylog2/graylog2-web-interface#1212

- Hide unlock dashboard link for reader users. Graylog2/graylog2-web-interface#1429

- Open radio documentation link on a new window. Graylog2/graylog2-web-interface#1427

- Use radio node page on message details. Graylog2/graylog2-web-interface#1423

28.57 Graylog 1.1.0

Released: 2015-06-04

https://www.graylog.org/graylog-1-1-is-now-generally-available/

- Properly set `node_id` on message input Graylog2/graylog2-server#1210

- Fixed handling of booleans in configuration forms in the web interface

- Various design fixes in the web interface

28.58 Graylog 1.1.0-rc.3

Released: 2015-06-02

https://www.graylog.org/graylog-v1-1-rc3-is-now-available/

- Unbreak server startup with collector thresholds set. Graylog2/graylog2-server#1194

- Adding verbal alert description to alert email templates and subject line defaults. Graylog2/graylog2-server#1158

- Fix message backlog in default body template in FormattedEmailAlertSender. Graylog2/graylog2-server#1163

- Make RawMessageEvent's fields volatile to guard against cross-cpu visibility issues. Graylog2/graylog2-server#1207

- Set default for "disable_index_range_calculation" to "true".

- Passing in value to text area fields in configuration forms. Graylog2/graylog2-web-interface#1340

- Stream list has no loading spinner. Graylog2/graylog2-web-interface#1309

- Showing a helpful notification when there are no active/inactive collectors. Graylog2/graylog2-web-interface#1302

- Improve behavior when field graphs are stacked. Graylog2/graylog2-web-interface#1348

- Keep new lines added by users on alert callbacks. Graylog2/graylog2-web-interface#1270

- Fix duplicate metrics reporting if two components subscribed to the same metric on the same page. Graylog2/graylog2-server#1199

- Make sidebar visible on small screens. Graylog2/graylog2-web-interface#1390

- Showing warning and disabling edit button for output if plugin is missing. Graylog2/graylog2-web-interface#1185

- Using formatted fields in old message loader. Graylog2/graylog2-web-interface#1393

- Several styling and UX improvements

28.59 Graylog 1.1.0-rc.1

Released: 2015-05-27

https://www.graylog.org/graylog-v1-1-rc1-is-now-available/

- Unable to send email alerts. Graylog2/graylog2-web-interface#1346

- "Show messages from this collector view" displays no messages. Graylog2/graylog2-web-interface#1334

- Exception error in search page when using escaped characters. Graylog2/graylog2-web-interface#1356

- Wrong timestamp on stream rule editor. Graylog2/graylog2-web-interface#1328

- Quickvalue values are not linked to update search query. Graylog2/graylog2-web-interface#1296

- Stream list has no loading spinner. Graylog2/graylog2-web-interface#1309

- Collector list with only inactive collectors is confusing. Graylog2/graylog2-web-interface#1302

- Update sockjs-client to 1.0.0. Graylog2/graylog2-web-interface#1344

- Scroll to search bar when new query term is added. Graylog2/graylog2-web-interface#1284

- Scroll to quick values if not visible. Graylog2/graylog2-web-interface#1284

- Scroll to newly created field graphs. Graylog2/graylog2-web-interface#1284

- Problems with websockets and even xhr streaming. Graylog2/graylog2-web-interface#1344, Graylog2/graylog2-web-interface#1353, Graylog2/graylog2-web-interface#1338, Graylog2/graylog2-web-interface#1322

- Add to search bar not working on sources tab. Graylog2/graylog2-web-interface#1350

- Make field graphs work with streams. Graylog2/graylog2-web-interface#1352

- Improved page design on outputs page. Graylog2/graylog2-web-interface#1236

- Set startpage button missing for dashboards. Graylog2/graylog2-web-interface#1345

- Generating chart for http response code is broken. Graylog2/graylog2-web-interface#1358

28.60 Graylog 1.1.0-beta.3

Released: 2015-05-27

https://www.graylog.org/graylog-1-1-beta-3-is-now-available/

- Kafka inputs now support syslog, GELF and raw messages Graylog2/graylog2-server#322

- Configurable timezone for the flexdate converter in extractors. Graylog2/graylog2-server#1166

- Allow decimal values for greater/smaller stream rules. Graylog2/graylog2-server#1101

- New configuration file option to control the default widget cache time. Graylog2/graylog2-server#1170

- Expose heartbeat configuration for AMQP inputs. Graylog2/graylog2-server#1010

- New alert condition to alert on field content. Graylog2/graylog2-server#537

- Add <code>-Dwebsockets.enabled=false</code> option for the web interface to disable websockets. Graylog2/graylog2-web-interface#1322

- Clicking the Graylog logo redirects to the custom startpage now. Graylog2/graylog2-web-interface#1315

- Improved reset and filter feature in sources tab. Graylog2/graylog2-web-interface#1337

- Fixed issue with stopping Kafka based inputs. Graylog2/graylog2-server#1171
- System throughput resource was always returning 0. Graylog2/graylog2-web-interface#1313
- MongoDB configuration problem with replica sets. Graylog2/graylog2-server#1173
- Syslog parser did not strip empty structured data fields. Graylog2/graylog2-server#1161
- Input metrics did not update after input has been stopped and started again. Graylog2/graylog2-server#1187
- NullPointerException with existing inputs in database fixed. Graylog2/graylog2-web-interface#1312
- Improved browser input validation for several browsers. Graylog2/graylog2-web-interface#1318
- Grok pattern upload did not work correctly. Graylog2/graylog2-web-interface#1321
- Internet Explorer 9 fixes. Graylog2/graylog2-web-interface#1319, Graylog2/graylog2-web-interface#1320
- Quick values feature did not work with reader users. Graylog2/graylog2-server#1169
- Replay link for keyword widgets was broken. Graylog2/graylog2-web-interface#1323
- Provide visual feedback when expanding message details. Graylog2/graylog2-web-interface#1283
- Allow filtering of saved searches again. Graylog2/graylog2-web-interface#1277
- Add back "Show details" link for global input metrics. Graylog2/graylog2-server#1168
- Provide visual feedback when dashboard widgets are loading. Graylog2/graylog2-web-interface#1324
- Restore preview for keyword time range selector. Graylog2/graylog2-web-interface#1280
- Fixed issue where widgets loading data looked empty. Graylog2/graylog2-web-interface#1324

28.61 Graylog 1.1.0-beta.2

Released: 2015-05-20

https://www.graylog.org/graylog-1-1-beta-is-now-available/

- CSV output streaming support including full text message
- Simplified MongoDB configuration with URI support
- Improved tokenizer for extractors
- Configurable UDP buffer size for incoming messages
- Enhanced Grok support with type conversions (integers, doubles and dates)
- Elasticsearch 1.5.2 support
- Added support for integrated Log Collector
- Search auto-complete
- Manual widget resize
- Auto resize of widgets based on screen size
- Faster search results
- Moved search filter for usability
- Updated several icons to text boxes for usability
- Search highlight toggle

- Pie charts (Stacked charts are coming too!)

- Improved stream management

- Output plugin and Alarm callback edit support

- Dashboard widget search edit

- Dashboard widget direct search button

- Dashboard background update support for better performance

- Log collector status UI

28.62 Graylog 1.0.2

Released: 2015-04-28

https://www.graylog.org/graylog-v1-0-2-has-been-released/

- Regular expression and Grok test failed when example message is a JSON document or contains special characters (Graylog2/graylog2-web-interface#1190, Graylog2/graylog2-web-interface#1195)

- "Show message terms" was broken (Graylog2/graylog2-web-interface#1168)

- Showing message indices was broken (Graylog2/graylog2-web-interface#1211)

- Fixed typo in SetIndexReadOnlyJob (Graylog2/graylog2-web-interface#1206)

- Consistent error messages when trying to create graphs from non-numeric values (Graylog2/graylog2-web-interface#1210)

- Fix message about too few file descriptors for Elasticsearch when number of file descriptors is unlimited (Graylog2/graylog2-web-interface#1220)

- Deleting output globally which was assigned to multiple streams left stale references (Graylog2/graylog2-server#1113)

- Fixed problem with sending alert emails (Graylog2/graylog2-server#1086)

- TokenizerConverter can now handle mixed quoted and un-quoted k/v pairs (Graylog2/graylog2-server#1083)

28.63 Graylog 1.0.1

Released: 2015-03-16

https://www.graylog.org/graylog-v1-0-1-has-been-released/

- Properly log stack traces (Graylog2/graylog2-server#970)

- Update REST API browser to new Graylog logo

- Avoid spamming the logs if the original input of a message in the disk journal can't be loaded (Graylog2/graylog2-server#1005)

- Allows reader users to see the journal status (Graylog2/graylog2-server#1009)

- Compatibility with MongoDB 3.0 and Wired Tiger storage engine (Graylog2/graylog2-server#1024)

- Respect `rest_transport_uri` when generating entity URLs in REST API (Graylog2/graylog2-server#1020)

- Properly map `NodeNotFoundException` (Graylog2/graylog2-web-interface#1137)

- Allow replacing all existing Grok patterns on bulk import (Graylog2/graylog2-web-interface#1150)
- Configuration option for discarding messages on error in AMQP inputs (Graylog2/graylog2-server#1018)
- Configuration option of maximum HTTP chunk size for HTTP-based inputs (Graylog2/graylog2-server#1011)
- Clone alarm callbacks when cloning a stream (Graylog2/graylog2-server#990)
- Add `hasField()` and `getField()` methods to `MessageSummary` class (Graylog2/graylog2-server#923)
- Add per input parse time metrics (Graylog2/graylog2-web-interface#1106)
- Allow the use of https://logging.apache.org/log4j/extras/ log4j-extras classes in log4j configuration (Graylog2/graylog2-server#1042)
- Fix updating of input statistics for Radio nodes (Graylog2/graylog2-web-interface#1022)
- Emit proper error message when a regular expression in an Extractor doesn't match example message (Graylog2/graylog2-web-interface#1157)
- Add additional information to system jobs (Graylog2/graylog2-server#920)
- Fix false positive message on LDAP login test (Graylog2/graylog2-web-interface#1138)
- Calculate saved search resolution dynamically (Graylog2/graylog2-web-interface#943)
- Only enable LDAP test buttons when data is present (Graylog2/graylog2-web-interface#1097)
- Load more than 1 message on Extractor form (Graylog2/graylog2-web-interface#1105)
- Fix NPE when listing alarm callback using non-existent plugin (Graylog2/graylog2-web-interface#1152)
- Redirect to nodes overview when node is not found (Graylog2/graylog2-web-interface#1137)
- Fix documentation links to integrations and data sources (Graylog2/graylog2-web-interface#1136)
- Prevent accidental indexing of web interface by web crawlers (Graylog2/graylog2-web-interface#1151)
- Validate grok pattern name on the client to avoid duplicate names (Graylog2/graylog2-server#937)
- Add message journal usage to nodes overview page (Graylog2/graylog2-web-interface#1083)
- Properly format numbers according to locale (Graylog2/graylog2-web-interface#1128, Graylog2/graylog2-web-interface#1129)

28.64 Graylog 1.0.0

Released: 2015-02-19

https://www.graylog.org/announcing-graylog-v1-0-ga/

- No changes since Graylog 1.0.0-rc.4

28.65 Graylog 1.0.0-rc.4

Released: 2015-02-13

https://www.graylog.org/graylog-v1-0-rc-4-has-been-released/

- Default configuration file locations have changed. Graylog2/graylog2-server#950
- Improved error handling on search errors. Graylog2/graylog2-server#954
- Dynamically update dashboard widgets with keyword range. Graylog2/graylog2-server#956, Graylog2/graylog2-web-interface#958
- Prevent duplicate loading of plugins. Graylog2/graylog2-server#948
- Fixed password handling when editing inputs. Graylog2/graylog2-web-interface#1103
- Fixed issues getting Elasticsearch cluster health. Graylog2/graylog2-server#953
- Better error handling for extractor imports. Graylog2/graylog2-server#942
- Fixed structured syslog parsing of keys containing special characters. Graylog2/graylog2-server#845
- Improved layout on Grok patterns page. Graylog2/graylog2-web-interface#1109
- Improved formatting large numbers. Graylog2/graylog2-web-interface#1111
- New Graylog logo.

28.66 Graylog 1.0.0-rc.3

Released: 2015-02-05

https://www.graylog.org/graylog-v1-0-rc-3-has-been-released/

- Fixed compatibility with MongoDB version 2.2. Graylog2/graylog2-server#941
- Fixed performance regression in process buffer handling. Graylog2/graylog2-server#944
- Fixed data type for the `max_size_per_index` config option value. Graylog2/graylog2-web-interface#1100
- Fixed problem with indexer error page. Graylog2/graylog2-web-interface#1102

28.67 Graylog 1.0.0-rc.2

Released: 2015-02-04

https://www.graylog.org/graylog-v1-0-rc-2-has-been-released/

- Better Windows compatibility. Graylog2/graylog2-server#930
- Added helper methods for the plugin API to simplify plugin development.
- Fixed problem with input removal on radio nodes. Graylog2/graylog2-server#932
- Improved buffer information for input, process and output buffers. Graylog2/graylog2-web-interface#1096
- Fixed API return value incompatibility regarding node objects. Graylog2/graylog2-server#933
- Fixed reloading of LDAP settings. Graylog2/graylog2-server#934
- Fixed ordering of message input state labels. Graylog2/graylog2-web-interface#1094

- Improved error messages for journal related errors. Graylog2/graylog2-server#931

- Fixed browser compatibility for stream rules form. Graylog2/graylog2-web-interface#1095

- Improved grok pattern management. Graylog2/graylog2-web-interface#1099, Graylog2/graylog2-web-interface#1098

28.68 Graylog 1.0.0-rc.1

Released: 2015-01-28

https://www.graylog.org/graylog-v1-0-rc-1-has-been-released/

- Cleaned up internal metrics when input is terminating. Graylog2/graylog2-server#915

- Added Telemetry plugin options to example graylog.conf. Graylog2/graylog2-server#914

- Fixed problems with user permissions on streams. Graylog2/graylog2-web-interface#1058

- Added information about different rotation strategies to REST API. Graylog2/graylog2-server#913

- Added better error messages for failing inputs. Graylog2/graylog2-web-interface#1056

- Fixed problem with JVM options in `bin/radioctl` script. Graylog2/graylog2-server#918

- Fixed issue with updating input configuration. Graylog2/graylog2-server#919

- Fixed password updating for reader users by the admin. Graylog2/graylog2-web-interface#1075

- Enabled the `message_journal_enabled` config option by default. Graylog2/graylog2-server#924

- Add REST API endpoint to list reopened indices. Graylog2/graylog2-web-interface#1072

- Fixed problem with GELF stream output. Graylog2/graylog2-server#921

- Show an error message on the indices page if the Elasticsearch cluster is not available. Graylog2/graylog2-web-interface#1070

- Fixed a problem with stopping inputs. Graylog2/graylog2-server#926

- Changed output configuration display to mask passwords. Graylog2/graylog2-web-interface#1066

- Disabled message journal on radio nodes. Graylog2/graylog2-server#927

- Create new message representation format for search results in alarm callback messages. Graylog2/graylog2-server#923

- Fixed stream router to update the stream engine if a stream has been changed. Graylog2/graylog2-server#922

- Fixed focus problem in stream rule modal windows. Graylog2/graylog2-web-interface#1063

- Do not show new dashboard link for reader users. Graylog2/graylog2-web-interface#1057

- Do not show stream output menu for reader users. Graylog2/graylog2-web-interface#1059

- Do not show user forms of other users for reader users. Graylog2/graylog2-web-interface#1064

- Do not show permission settings in the user profile for reader users. Graylog2/graylog2-web-interface#1055

- Fixed extractor edit form with no messages available. Graylog2/graylog2-web-interface#1061

- Fixed problem with node details page and JVM locale settings. Graylog2/graylog2-web-interface#1062

- Improved page layout for Grok patterns.

- Improved layout for the message journal information. Graylog2/graylog2-web-interface#1084, Graylog2/graylog2-web-interface#1085

- Fixed wording on radio inputs page. Graylog2/graylog2-web-interface#1077

- Fixed formatting on indices page. Graylog2/graylog2-web-interface#1086

- Improved error handling in stream rule form. Graylog2/graylog2-web-interface#1076

- Fixed time range selection problem for the sources page. Graylog2/graylog2-web-interface#1080

- Several improvements regarding permission checks for user creation. Graylog2/graylog2-web-interface#1088

- Do not show stream alert test button for reader users. Graylog2/graylog2-web-interface#1089

- Fixed node processing status not updating on the nodes page. Graylog2/graylog2-web-interface#1090

- Fixed filename handling on Windows. Graylog2/graylog2-server#928, Graylog2/graylog2-server#732

28.69 Graylog 1.0.0-beta.2

Released: 2015-01-21

https://www.graylog.org/graylog-v1-0-beta-3-has-been-released/

- Fixed stream alert creation. Graylog2/graylog2-server#891

- Suppress warning message when PID file doesn't exist. Graylog2/graylog2-server#889

- Fixed an error on outputs page with missing output plugin. Graylog2/graylog2-server#894

- Change default heap and garbage collector settings in scripts.

- Add extractor information to log message about failing extractor.

- Fixed problem in SplitAndIndexExtractor. Graylog2/graylog2-server#896

- Improved rendering time for indices page. Graylog2/graylog2-web-interface#1060

- Allow user to edit its own preferences. Graylog2/graylog2-web-interface#1049

- Fixed updating stream attributes. Graylog2/graylog2-server#902

- Stream throughput now shows combined value over all nodes. Graylog2/graylog2-web-interface#1047

- Fixed resource leak in JVM PermGen memory. Graylog2/graylog2-server#907

- Update to gelfclient-1.1.0 to fix DNS resolving issue. Graylog2/graylog2-server#882

- Allow arbitrary characters in user names (in fact in any resource url). Graylog2/graylog2-web-interface#1005, Graylog2/graylog2-web-interface#1006

- Fixed search result CSV export. Graylog2/graylog2-server#901

- Skip GC collection notifications for parallel collector. Graylog2/graylog2-server#899

- Shorter reconnect timeout for Radio AMQP connections. Graylog2/graylog2-server#900

- Fixed random startup error in Radio. Graylog2/graylog2-server#911

- Fixed updating an alert condition. Graylog2/graylog2-server#912

- Add system notifications for journal related warnings. Graylog2/graylog2-server#897

- Add system notifications for failing outputs. Graylog2/graylog2-server#741

- Improve search result pagination. Graylog2/graylog2-web-interface#834

- Improved regex error handling in extractor testing. Graylog2/graylog2-web-interface#1044

- Wrap long names for node metrics. Graylog2/graylog2-web-interface#1028

- Fixed node information progress bars. Graylog2/graylog2-web-interface#1046

- Improve node buffer utilization readability. Graylog2/graylog2-web-interface#1046

- Fixed username alert receiver form field. Graylog2/graylog2-web-interface#1050

- Wrap long messages without break characters. Graylog2/graylog2-web-interface#1052

- Hide list of node plugins if there aren't any plugins installed.

- Warn user before leaving page with unpinned graphs. Graylog2/graylog2-web-interface#808

28.70 Graylog 1.0.0-beta.2

Released: 2015-01-16

https://www.graylog.org/graylog-v1-0-0-beta2/

- SIGAR native libraries are now found correctly (for getting system information)

- plugins can now state if they want to run in server or radio

- Fixed LDAP settings testing. Graylog2/graylog2-web-interface#1026

- Improved RFC5425 syslog message parsing. Graylog2/graylog2-server#845

- JVM arguments are now being logged on start. Graylog2/graylog2-server#875

- Improvements to log messages when Elasticsearch connection fails during start.

- Fixed an issue with AMQP transport shutdown. Graylog2/graylog2-server#874

- After index cycling the System overview page could be broken. Graylog2/graylog2-server#880

- Extractors can now be edited. Graylog2/graylog2-web-interface#549

- Fixed saving user preferences. Graylog2/graylog2-web-interface#1027

- Scripts now return proper exit codes. Graylog2/graylog2-server#886

- Grok patterns can now be uploaded in bulk. Graylog2/graylog2-server#377

- During extractor creation the test display could be offset. Graylog2/graylog2-server#804

- Performance fix for the System/Indices page. Graylog2/graylog2-web-interface#1035

- A create dashboard link was shown to reader users, leading to an error when followed. Graylog2/graylog2-web-interface#1032

- Content pack section was shown to reader users, leading to an error when followed. Graylog2/graylog2-web-interface#1033

- Failing stream outputs were being restarted constantly. Graylog2/graylog2-server#741

28.71 Graylog2 0.92.4

Released: 2015-01-14

https://www.graylog.org/graylog2-v0-92-4/

- [SERVER] Ensure that Radio inputs can only be started on server nodes (Graylog2/graylog2-server#843)
- [SERVER] Avoid division by zero when finding rotation anchor in the time-based rotation strategy (Graylog2/graylog2-server#836)
- [SERVER] Use username as fallback if display name in LDAP is empty (Graylog2/graylog2-server#837)

28.72 Graylog 1.0.0-beta.1

Released: 2015-01-12

https://www.graylog.org/graylog-v1-0-0-beta1/

- Message Journaling
- New Widgets
- Grok Extractor Support
- Overall stability and resource efficiency improvements
- Single binary for `graylog2-server` and `graylog2-radio`
- Inputs are now editable
- Order of field charts rendered inside the search results page is now maintained.
- Improvements in focus and keyboard behaviour on modal windows and forms.
- You can now define whether to disable expensive, frequent real-time updates of the UI in the settings of each user. (For example the updating of total messages in the system)
- Experimental search query auto-completion that can be enabled in the user preferences.
- The API browser now documents server response payloads in a better way so you know what to expect as an answer to your call.
- Now using the standard Java ServiceLoader for plugins.

28.73 Graylog2 0.92.3

Released: 2014-12-23

https://www.graylog.org/graylog2-v0-92-3/

- [SERVER] Removed unnecessary instrumentation in certain places to reduce GC pressure caused by many short living objects (Graylog2/graylog2-server#800)
- [SERVER] Limit Netty worker thread pool to 16 threads by default (see `rest_worker_threads_max_pool_size` in graylog2.conf
- [WEB] Fixed upload of content packs when a URI path prefix (`application.context` in graylog2-web-interface.conf) is being used (Graylog2/graylog2-web-interface#1009)
- [WEB] Fixed display of metrics of type Counter (Graylog2/graylog2-server#795)

28.74 Graylog2 0.92.1

Released: 2014-12-11

https://www.graylog.org/graylog2-v0-92-1/

- [SERVER] Fixed name resolution and overriding sources for network inputs.

- [SERVER] Fixed wrong delimiter in GELF TCP input.

- [SERVER] Disabled the output cache by default. The output cache is the source of all sorts of interesting problems. If you want to keep using it, please read the upgrade notes.

- [SERVER] Fixed message timestamps in GELF output.

- [SERVER] Fixed connection counter for network inputs.

- [SERVER] Added warning message if the receive buffer size (SO_RECV) couldn't be set for network inputs.

- [WEB] Improved keyboard shortcuts with most modal dialogs (e. g. hitting Enter submits the form instead of just closing the dialogs).

- [WEB] Upgraded to play2-graylog2 1.2.1 (compatible with Play 2.3.x and Java 7).

28.75 Graylog2 0.92.0

Released: 2014-12-01

https://www.graylog.org/graylog2-v0-92/

- [SERVER] IMPORTANT SECURITY FIX: It was possible to perform LDAP logins with crafted wildcards. (A big thank you to Jose Tozo who discovered this issue and disclosed it very responsibly.)

- [SERVER] Generate a system notification if garbage collection takes longer than a configurable threshold.

- [SERVER] Added several JVM-related metrics.

- [SERVER] Added support for Elasticsearch 1.4.x which brings a lot of stability and resilience features to Elasticsearch clusters.

- [SERVER] Made version check of Elasticsearch version optional. Disabling this check is not recommended.

- [SERVER] Added an option to disable optimizing Elasticsearch indices on index cycling.

- [SERVER] Added an option to disable time-range calculation for indices on index cycling.

- [SERVER] Lots of other performance enhancements for large setups (i.e. involving several Radio nodes and multiple Graylog2 Servers).

- [SERVER] Support for Syslog Octet Counting, as used by syslog-ng for syslog via TCP (#743)

- [SERVER] Improved support for structured syslog messages (#744)

- [SERVER] Bug fixes regarding IPv6 literals in mongodb_replica_set and elasticsearch_discovery_zen_ping_unicast_hosts

- [WEB] Added additional details to system notification about Elasticsearch max. open file descriptors.

- [WEB] Fixed several bugs and inconsistencies regarding time zones.

- [WEB] Improved graphs and diagrams

- [WEB] Allow to update dashboards when browser window is not on focus (#738)

- [WEB] Bug fixes regarding timezone handling

- Numerous internal bug fixes

28.76 Graylog2 0.92.0-rc.1

Released: 2014-11-21

https://www.graylog.org/graylog2-v0-92-rc-1/

- [SERVER] Generate a system notification if garbage collection takes longer than a configurable threshold.
- [SERVER] Added several JVM-related metrics.
- [SERVER] Added support for Elasticsearch 1.4.x which brings a lot of stability and resilience features to Elasticsearch clusters.
- [SERVER] Made version check of Elasticsearch version optional. Disabling this check is not recommended.
- [SERVER] Added an option to disable optimizing Elasticsearch indices on index cycling.
- [SERVER] Added an option to disable time-range calculation for indices on index cycling.
- [SERVER] Lots of other performance enhancements for large setups (i. e. involving several Radio nodes and multiple Graylog2 Servers).
- [WEB] Upgraded to Play 2.3.6.
- [WEB] Added additional details to system notification about Elasticsearch max. open file descriptors.
- [WEB] Fixed several bugs and inconsistencies regarding time zones.
- Numerous internal bug fixes

28.77 Graylog2 0.91.3

Released: 2014-11-05

https://www.graylog.org/graylog2-v0-90-3-and-v0-91-3-has-been-released/

- Fixed date and time issues related to DST changes
- Requires Elasticsearch 1.3.4; Elasticsearch 1.3.2 had a bug that can cause index corruptions.
- The `mongodb_replica_set` configuration variable now supports IPv6
- Messages read from the on-disk caches could be stored with missing fields

28.78 Graylog2 0.91.3

Released: 2014-11-05

https://www.graylog.org/graylog2-v0-90-3-and-v0-91-3-has-been-released/

- Fixed date and time issues related to DST changes
- The `mongodb_replica_set` configuration variable now supports IPv6
- Messages read from the on-disk caches could be stored with missing fields

28.79 Graylog2 0.92.0-beta.1

Released: 2014-11-05

https://www.graylog.org/graylog2-v0-92-beta-1/

- Content packs
- [SERVER] SSL/TLS support for Graylog2 REST API
- [SERVER] Support for time based retention cleaning of your messages. The old message count based approach is still the default.
- [SERVER] Support for Syslog Octet Counting, as used by syslog-ng for syslog via TCP (Graylog2/graylog2-server#743)
- [SERVER] Improved support for structured syslog messages (Graylog2/graylog2-server#744)
- [SERVER] Bug fixes regarding IPv6 literals in `mongodb_replica_set` and `elasticsearch_discovery_zen_ping_unicast_hosts`
- [WEB] Revamped "Sources" page in the web interface
- [WEB] Improved graphs and diagrams
- [WEB] Allow to update dashboards when browser window is not on focus (Graylog2/graylog2-web-interface#738)
- [WEB] Bug fixes regarding timezone handling
- Numerous internal bug fixes

28.80 Graylog2 0.91.1

Released: 2014-10-17

https://www.graylog.org/two-new-graylog2-releases/

- Messages written to the persisted master caches were written to the system with unreadable timestamps, leading to
- errors when trying to open the message.
- Extractors were only being deleted from running inputs but not from all inputs
- Output plugins were not always properly loaded
- You can now configure the `alert_check_interval` in your `graylog2.conf`
- Parsing of configured Elasticsearch unicast discovery addresses could break when including spaces

28.81 Graylog2 0.90.1

Released: 2014-10-17

https://www.graylog.org/two-new-graylog2-releases/

- Messages written to the persisted master caches were written to the system with unreadable timestamps, leading to errors when trying to open the message.
- Extractors were only being deleted from running inputs but not from all inputs
- Output plugins were not always properly loaded
- You can now configure the `alert_check_interval` in your `graylog2.conf`
- Parsing of configured Elasticsearch unicast discovery addresses could break when including spaces

28.82 Graylog2 0.91.0-rc.1

Released: 2014-09-23

https://www.graylog.org/graylog2-v0-90-has-been-released/

- Optional ElasticSearch v1.3.2 support

28.83 Graylog2 0.90.0

Released: 2014-09-23

https://www.graylog.org/graylog2-v0-90-has-been-released/

- Real-time data forwarding to Splunk or other systems
- Alert callbacks for greater flexibility
- New disk-based architecture for buffering in load spike situations
- Improved graphing
- Plugin API
- Huge performance and stability improvements across the whole stack
- Small possibility of losing messages in certain scenarios has been fixed
- Improvements to internal logging from threads to avoid swallowing Graylog2 error messages
- Paused streams are no longer checked for alerts
- Several improvements to timezone handling
- JavaScript performance fixes in the web interface and especially a fixed memory leak of charts on dashboards
- The GELF HTTP input now supports CORS
- Stream matching now has a configurable timeout to avoid stalling message processing in case of too complex rules or erroneous regular expressions
- Stability improvements for Kafka and AMQP inputs
- Inputs can now be paused and resumed

• Dozens of bug fixes and other improvements

28.84 Graylog2 0.20.3

Released: 2014-08-09

https://www.graylog.org/graylog2-v0-20-3-has-been-released/

• Bugfix: Storing saved searches was not accounting custom application contexts

• Bugfix: Editing stream rules could have a wrong a pre-filled value

• Bugfix: The create dashboard link was shown even if the user has no permission to so. This caused an ugly error page because of the missing permissions.

• Bugfix: graylog2-radio could lose numeric fields when writing to the message broker

• Better default batch size values for the Elasticsearch output

• Improved `rest_transport_uri` default settings to avoid confusion with loopback interfaces

• The deflector index is now also using the configured index prefix

28.85 Graylog2 0.20.2

Released: 2014-05-24

https://www.graylog.org/graylog2-v0-20-2-has-been-released/

• Search result highlighting

• Reintroduces AMQP support

• Extractor improvements and sharing

• Graceful shutdowns, Lifecycles, Load Balancer integration

• Improved stream alert emails

• Alert annotations

• CSV exports via the REST API now support chunked transfers and avoid heap size problems with huge result sets

• Login now redirects to page you visited before if there was one

• More live updating information in node detail pages

• Empty dashboards no longer show lock/unlock buttons

• Global inputs now also show IO metrics

• You can now easily copy message IDs into native clipboard with one click

• Improved message field selection in the sidebar

• Fixed display of floating point numbers in several places

• Now supporting application contexts in the web interface like `http://example.org/graylog2`

• Several fixes for LDAP configuration form

• Message fields in the search result sidebar now survive pagination

- Only admin users are allowed to change the session timeout for reader users

- New extractor: Copy whole input

- New converters: uppercase/lowercase, flexdate (tries to parse any string as date)

- New stream rule to check for presence or absence of fields

- Message processing now supports trace logging

- Better error message for ES discovery problems

- Fixes to GELF HTTP input and it holding open connections

- Some timezone fixes

- CSV exports now only contain selected fields

- Improvements for bin/graylog* control scripts

- UDP inputs now allow for custom receive buffer sizes

- Numeric extractor converter now supports floating point values

- Bugfix: Several small fixes to system notifications and closing them

- Bugfix: Carriage returns were not escaped properly in CSV exports

- Bugfix: Some AJAX calls redirected to the startpage when they failed

- Bugfix: Wrong sorting in sources table

- Bugfix: Quickvalues widget was broken with very long values

- Bugfix: Quickvalues modal was positioned wrong in some cases

- Bugfix: Indexer failures list could break when you had a lot of failures

- Custom application prefix was not working for field chart analytics

- Bugfix: Memory leaks in the dashboards

- Bugfix: NullPointerException when Elasticsearch discovery failed and unicast discovery was disabled

- Message backlog in alert emails did not always include the correct number of messages

- Improvements for message outputs: No longer only waiting for filled buffers but also flushing them regularly. This avoids problems that make Graylog2 look like it misses messages in cheap benchmark scenarios combined with only little throughput.

TWENTYNINE

INTRODUCTION

Graylog Enterprise, built on top of the Graylog open source platform, offers additional features that enable users to deploy Graylog at enterprise scale and apply Graylog to processes and workflows across the whole organization.

Please see the Graylog Enterprise Page for details.

SETUP

Graylog Enterprise comes as a Graylog server plugin which need to be installed in addition to the Graylog open source setup.

30.1 Requirements

The following list shows the minimum required Graylog versions for the Graylog Enterprise plugins.

Table 1: Enterprise Version Requirements

Enterprise Version	Required Graylog Version
1.0.0	2.0.0, 2.0.1
1.0.1	2.0.2, 2.0.3
1.2.0	2.1.0, 2.1.1, 2.1.2
1.2.1	2.1.3
2.2.0	2.2.0
2.2.1	2.2.1
2.3.0	2.3.0
2.3.1	2.3.1
2.3.2	2.3.2
2.4.0	2.4.0
2.4.1	2.4.1
2.4.2	2.4.2
2.4.3	2.4.3
2.4.4	2.4.4
2.4.5	2.4.5
2.4.6	2.4.6
2.5.0	2.5.0
2.5.1	2.5.1
3.0.0	3.0.0
3.0.1	3.0.1
3.0.2	3.0.2
3.1.0	3.1.0
3.1.1	3.1.1
3.1.2	3.1.2
3.1.3	3.1.3

30.2 Installation

Since Graylog 2.4 the Graylog Enterprise plugin can be installed the same way Graylog is installed. In most setups this will be done with the package tool provided by the distribution you are using and the online repository.

Note: For previous versions of Graylog Enterprise please contact your Graylog account manager.

Once you installed the Graylog Enterprise plugin you need to obtain a license from the Graylog Enterprise web page.

Should a simple `apt-get install graylog-enterprise-plugins` or `yum install graylog-enterprise-plugins` not work for you, the following information might help you.

Hint: You might want the *Integrations Plugins* in addition to the Enterprise Plugins. See following the names of all official provides package: `graylog-server graylog-enterprise-plugins graylog-integrations-plugins graylog-enterprise-integrations-plugins`

Important: The Graylog Enterprise plugin need to be installed on all your Graylog nodes!

30.2.1 DEB / RPM Package

The default installation should be done with the system package tools. It includes the repository installation that is described in the *Operating System Packages* installation guides.

When the usage of online repositories is not possible in your environment, you can download the Graylog Enterprise plugins at https://packages.graylog2.org.

Note: These packages can **only** be used when you installed Graylog via the *Operating System Packages*!

DEB

The installation on distributions like Debian or Ubuntu can be done with *apt-get* as installation tool from the previous installed online repository.

```
$ sudo apt-get install graylog-enterprise-plugins
```

RPM

The installation on distributions like CentOS or RedHat can be done with *yum* as installation tool from the previous installed online repository.

```
$ sudo yum install graylog-enterprise-plugins
```

30.2.2 Tarball

If you have done a manual installation you can get the tarball from the download locations listed in the following table.

Table 2: Enterprise Plugins download

Enterprise Version	Download URL
3.0.0	https://downloads.graylog.org/releases/graylog-enterprise/graylog-enterprise-plugins-3.0.0.tgz
3.0.1	https://downloads.graylog.org/releases/graylog-enterprise/graylog-enterprise-plugins-3.0.1.tgz
3.0.2	https://downloads.graylog.org/releases/graylog-enterprise/graylog-enterprise-plugins-3.0.2.tgz
3.1.0	https://downloads.graylog.org/releases/graylog-enterprise/graylog-enterprise-plugins-3.1.0.tgz
3.1.1	https://downloads.graylog.org/releases/graylog-enterprise/graylog-enterprise-plugins-3.1.1.tgz
3.1.2	https://downloads.graylog.org/releases/graylog-enterprise/graylog-enterprise-plugins-3.1.2.tgz
3.1.3	https://downloads.graylog.org/releases/graylog-enterprise/graylog-enterprise-plugins-3.1.3.tgz

The tarball includes the enterprise plugin JAR file and required binaries that need to be installed.

```
$ tar -tzf graylog-enterprise-plugins-3.1.3.tgz
  graylog-enterprise-plugins-3.1.3/LICENSE
  graylog-enterprise-plugins-3.1.3/plugin/graylog-plugin-enterprise-3.1.3.jar
  graylog-enterprise-plugins-3.1.3/bin/headless_shell
  graylog-enterprise-plugins-3.1.3/bin/chromedriver
  graylog-enterprise-plugins-3.1.3/bin/chromedriver_start.sh
```

JAR file

Depending on the Graylog setup method you have used, you have to install the plugin into different locations.

Table 3: Plugin Installation Locations

Installation Method	Directory
Operating System Packages	`/usr/share/graylog-server/plugin/`
Manual Setup	`/<extracted-graylog-tarball-path>/plugin/`

Also check the `plugin_dir` config option in your Graylog server configuration file. The default might have been changed.

Make sure to install the enterprise plugin JAR files alongside the other Graylog plugins. Your plugin directory should look similar to this after installing the enterprise plugins.

```
plugin/
├── graylog-plugin-aws-3.1.3.jar
├── graylog-plugin-collector-3.1.3.jar
├── graylog-plugin-enterprise-3.1.3.jar
└── graylog-plugin-threatintel-3.1.3.jar
```

Binary files

Depending on the Graylog setup method you have used, you have to copy the binaries into different locations.

Table 4: Binaries Installation Locations

Installation Method	Directory
Operating System Packages	`/usr/share/graylog-server/bin/`
Manual Setup	`/<extracted-graylog-tarball-path>/bin/`

Make sure to check the `bin_dir` configuration option set in your Graylog server configuration file, as the default may have changed.

30.3 Server Restart

After you installed the Graylog Enterprise plugins you have to restart each of your Graylog servers to load the plugins.

Note: We recommend restarting one server at a time!

You should see something like the following in your Graylog server logs. It indicates that the plugins have been successfully loaded.

```
2017-12-18T17:39:10.797+01:00 INFO  [CmdLineTool] Loaded plugin: AWS plugins 3.1.3
↪[org.graylog.aws.plugin.AWSPlugin]
2017-12-18T17:39:10.809+01:00 INFO  [CmdLineTool] Loaded plugin: Collector 3.1.3 [org.
↪graylog.plugins.collector.CollectorPlugin]
2017-12-18T17:39:10.811+01:00 INFO  [CmdLineTool] Loaded plugin: Enterprise
↪Integration Plugin 3.1.3 [org.graylog.plugins.enterprise_integration.
↪EnterpriseIntegrationPlugin]
2017-12-18T17:39:10.805+01:00 INFO  [CmdLineTool] Loaded plugin: Graylog Enterprise 3.
↪1.3 [org.graylog.plugins.enterprise.EnterprisePlugin]
2017-12-18T17:39:10.827+01:00 INFO  [CmdLineTool] Loaded plugin: Threat Intelligence
↪Plugin 3.1.3 [org.graylog.plugins.threatintel.ThreatIntelPlugin]
```

30.4 Cluster Setup

If you run a Graylog cluster you need to add the enterprise plugins to every Graylog node. Additionally your load-balancer must route `/api/plugins/org.graylog.plugins.archive/` only to the Graylog master node. Future versions of Graylog will forward these requests automatically to the correct node.

30.5 License Installation

The Graylog Enterprise plugins require a valid license to use the additional features.

Once you have obtained a license you can import it into your Graylog setup by going through the following steps.

1. As an admin user, open the "Enterprise/License" page from the menu in the web interface.

2. Click the Import new license button in the top right hand corner.

3. Copy the license text from the confirmation email and paste it into the text field.

---

4. The license should be valid and a preview of your license details should appear below the text field.

5. Click Import to activate the license.

The license automatically applies to all nodes in your cluster without the need to restart your server nodes.

Note: If there are errors, please check that you copied the entire license from the email without line breaks. The same license is also attached as a text file in case it is wrongly formatted in the email.
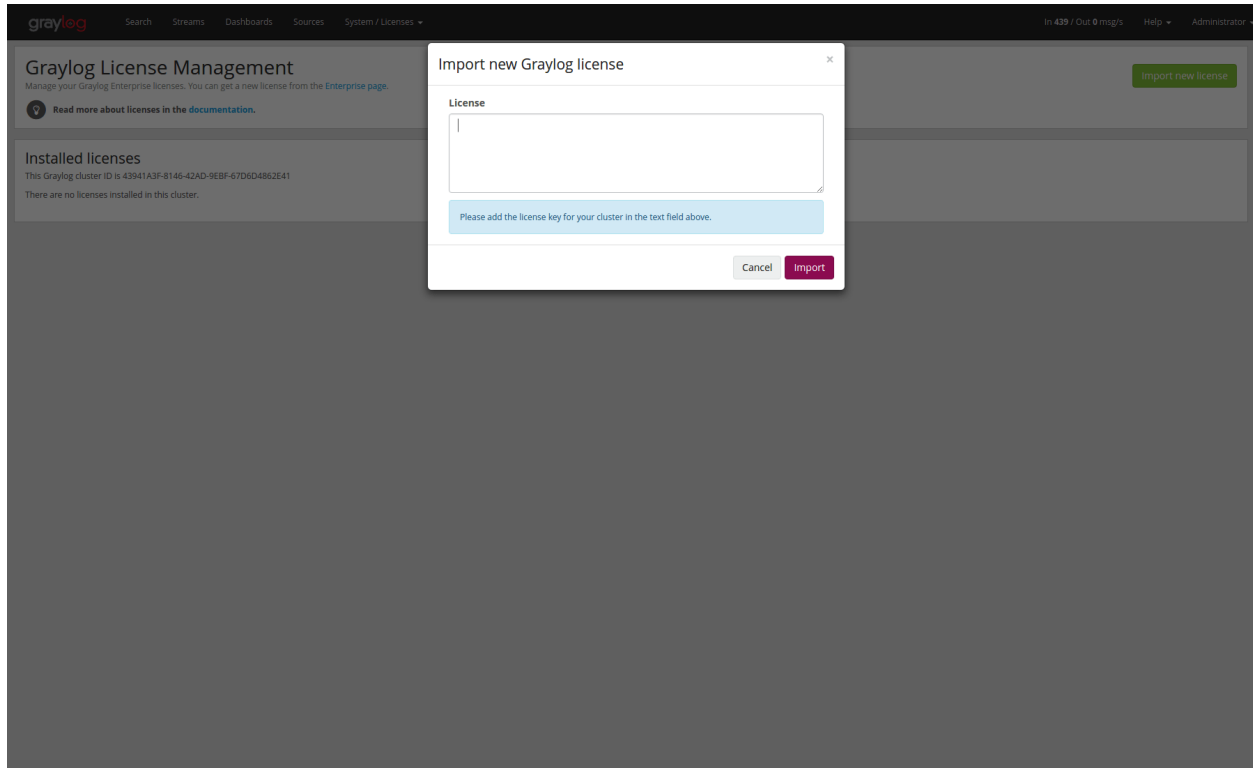
30.6 License Verification

Some Graylog licenses require to check their validity on a regular basis. This includes the free Graylog Enterprise license with a specific amount of traffic included.

If your network environment requires Graylog to use a proxy server in order to communicate with the external services via HTTPS, you'll have to configure the proxy server in the *Graylog configuration file*.

The Graylog web interface shows all details about the license, but if you are still unclear about the requirements, please contact our sales team with your questions.

30.6.1 Details on License Verification

Graylog Enterprise periodically sends the following information to 'api.graylog.com' via HTTPS on TCP port 443 for each installed license:

- A nonce to avoid modified reports

- The ID of the license

- The ID of the Graylog cluster

- A flag indicating if the license is violated

- A flag indicating if the license has expired

- A flag indicating if Graylog detected that the traffic measuring mechanisms have been modified

- A list of how much traffic was received and written by Graylog in the recent days, in bytes

30.6.2 Details on licensed traffic

Graylog has four counters, only the last is counted for the licensed traffic.

- **`org.graylog2.traffic.input`** the incoming message without any decoding, what is written to the journal before any processing.

- **`org.graylog2.traffic.decoded`** the message after the codec of the input has parsed the message (for example syslog parser)

- **`org.graylog2.traffic.system-output-traffic`** currently, this is stored in memory only and includes the traffic from archive restores.

- **`org.graylog2.traffic.output`** what is written to Elasticsearch after all processing is done.

Only the Elasticsearch output is measured, all other outgoing traffic does not count. The measurement happens when the message is serialized to elasticsearch. If a message is written to multiple indices the message will count for each index. It does not matter how many copies (replicas) the index has configured as this is done in elasticsearch.

Each of the counters follows these rules:

- count the length of the field name.

- If the content of the field is a string, the length of the string is counted not the bytes of that string

- **for non-string content in the field, the byte length of that content is counted**

    - byte = 1 byte

    - char/short = 2 bytes

    - bool/int/floar = 4 bytes

    - long/double = 8 bytes

    - dates = 8 bytes

- all internal fields are not countent (those meta information that are created by Graylog)

ARCHIVING

Graylog enables you to configure a retention period to automatically delete older messages - this is to help you control the costs of storage in Elasticsearch. But we know it's not ideal deciding between keeping less messages in Graylog or paying more for hardware. Additionally, many of you are required to store data for long periods of time due to compliance requirements like PCI or HIPAA.

The Archiving functionality allows you to archive log messages until you need to re-import them into Graylog for analysis. You can instruct Graylog to automatically archive log messages to compressed flat files on the local filesystem before retention cleaning kicks in and messages are deleted from Elasticsearch. Archiving also works through a REST call or the web interface if you don't want to wait until retention cleaning to happen. We chose flat files for this because they are vendor agnostic so you will always be able to access your data.

You can then do whatever you want with the archived files: move them to cheap storage, write them on tape, or even print them out if you need to! If you need to search through archived data in the future, you can move any selection of archived messages back into the Graylog archive folder, and the web interface will enable you to temporarily import the archive so you can analyze the messages again in Graylog.

Note: Archiving is a commercial feature and part of Graylog Enterprise.

31.1 Setup

Graylog Archive is a commercial feature that can be installed in addition to the Graylog open source server.

31.1.1 Installation

Archiving is part of the Graylog Enterprise plugin, please check the *Graylog Enterprise setup page* for details on how to install it.

31.1.2 Configuration

Graylog Archive can be configured via the Graylog web interface and does not need any changes in the Graylog server configuration file.

In the web interface menu navigate to "Enterprise/Archives" and click "Configuration" to adjust the configuration.

Archive Options

There are several configuration options to configure archiving.

Table 1: Configuration Options

Name	Description
Backend	Backend on the **master** node where the archive files will be stored.
Max Segment Size	Maximum size (in *bytes*) of archive segment files.
Compression Type	Compression type that will be used to compress the archives.
Checksum Type	Checksum algorithm that is used to calculate the checksum for archives.
Restore index batch size	Elasticsearch batch size when restoring archive files.
Streams to archive	Streams that should be included in the archive.

Backend

The archived indices will be stored in a backend. A backend that stores the data in `/tmp/graylog-archive` is created when the server starts for the first time but you can create a new backend if you want to store the data in a different path.

Max Segment Size

When archiving an index, the archive job writes the data into segments. The *Max Segment Size* setting sets the size limit for each of these data segments.

This allows control over the file size of the segment files to make it possible to process them with tools which have a size limit for files.

Once the size limit is reached, a new segment file will be started.

Example:

```
/path/to/archive/
  graylog_201/
    archive-metadata.json
    archive-segment-0.gz
    archive-segment-1.gz
    archive-segment-2.gz
```

Compression Type

Archives will be compressed with gzip by default. This option can be changed to use a different compression type.

The selected compression type has a big impact on the time it takes to archive an index. Gzip for example is pretty slow but has a great compression rate. Snappy and LZ4 are way faster but the archives will be bigger.

Here is a comparison between the available compression algorithms with test data.

Table 2: Compression Type Comparison

Type	Index Size	Archive Size	Duration
gzip	1 GB	134 MB	15 minutes, 23 seconds
Snappy	1 GB	291 MB	2 minutes, 31 seconds
LZ4	1 GB	266 MB	2 minutes, 25 seconds

Note: Results with your data may vary! Make sure to test the different compression types to find the one that is best for your data.

Warning: The current implementation of LZ4 is not compatible with the LZ4 CLI tools, thus decompressing the LZ4 archives outside of Graylog is currently not possible.

Checksum Type

When writing archives Graylog computes a CRC32 checksum over the files. This option can be changed to use a different checksum algorithm.

The type of checksum depends on the use case. CRC32 and MD5 are quick to compute and a reasonable choice to be able to detect damaged files, but neither is suitable as protection against malicious changes in the files. Graylog also supports using SHA-1 or SHA-256 checksums which can be used to make sure the files were not modified, as they are cryptographic hashes.

The best choice of checksum types depends on whether the necessary system tools are installed to compute them later (not all systems come with a SHA-256 utility for example), speed of checksum calculation for larger files as well as the security considerations.

Restore Index Batch Size

This setting controls the batch size for re-indexing archive data into Elasticsearch. When set to `1000`, the restore job will re-index the archived data in document batches of 1000.

You can use this setting to control the speed of the restore process and also how much load it will generate on the Elasticsearch cluster. The **higher** the batch size, the **faster** the restore will progress and the **more** load will be put on your Elasticsearch cluster in addition to the normal message processing.

Make sure to tune this **carefully** to avoid any negative impact on your message indexing throughput and search speed!

Streams To Archive

This option can be used to select which streams should be included in the archive. With this you are able to archive only your important data instead of archiving everything that is arriving in Graylog.

Note: New streams will be archived automatically. If you create a new stream and don't want it to be archived, you have to disable it in this configuration dialog.

Backends

A backend can be used to store the archived data. For now, we only support a single file system backend type.

File System

The archived indices will be stored in the *Output base path* directory. This directory **needs to exist and be writable for the Graylog server process** so the files can be stored.

Note: Only the **master** node needs access to the *Output base path* directory because the archiving process runs on the master node.

We recommend to put the *Output base path* directory onto a **separate disk or partition** to avoid any negative impact on the message processing should the archiving fill up the disk.

Table 3: Configuration Options

Name	Description
Title	A simple title to identify the backend.
Description	Longer description for the backend.
Output base path	Directory path where the archive files should be stored.

Output base path

The output base path can either be a simple directory path string or a template string to build dynamic paths.

You could use a template string to store the archive data in a directory tree that is based on the archival date.

Example:

```
# Template
/data/graylog-archive/${year}/${month}/${day}

# Result
/data/graylog-archive/2017/04/01/graylog_0
```

Table 4: Available Template Variables

Name	Description
${year}	Archival date year. (e.g. "2017")
${month}	Archival date month. (e.g "04")
${day}	Archival date day. (e.g. "01")
${hour}	Archival date hour. (e.g. "23")
${minute}	Archival date minute. (e.g. "24")
${second}	Archival date second. (e.g. "59")
${index-name}	Name of the archived index. (e.g. "graylog_0")

Index Retention

Graylog is using configurable index retention strategies to delete old indices. By default indices can be *closed* or *deleted* if you have more than the configured limit.

Graylog Archive offers a new index retention strategy that you can configure to automatically archive an index before closing or deleting it.

Index retention strategies can be configured in the system menu under "System/Indices". Select an index set and click "Edit" to change the index rotation and retention strategies.

As with the regular index retention strategies, you can configure a max number of Elasticsearch indices. Once there are more indices than the configured limit, the oldest ones will be archived into the backend and then closed or deleted. You can also decide to not do anything (*NONE*) after archiving an index. In that case **no cleanup of old indices will happen** and you have to take care of that yourself!

31.2 Usage

31.2.1 Creating Archives

There are three ways to create archives from the Graylog Elasticsearch indices.

- *Web Interface*
- *Index Retention*
- *REST API*

Web Interface

You can manually create an archive on the "Enterprise/Archives" page in the web interface.

On the "Create Archive for Index" section of the page is a form where you can select an index and archive it by pressing "Archive Index".

Using this will just archive the index to disk and does not close it or delete it. This is a great way to test the archiving feature without changing your *index retention configuration*.

Index Retention

Graylog Archive ships with an index retention strategy that can be used to automatically create archives before closing or deleting Elasticsearch indices.

This is the easiest way to automatically create archives without custom scripting.

Please see the *Index Retention Configuration* on how to configure it.

REST API

Graylog Archive also offers a REST API that you can use to automate archive creation if you have some special requirements and need a more flexible way to do this.

An index can be archived with a simple curl command:

```
$ curl -s -u admin -H 'X-Requested-By: cli' -X POST http://127.0.0.1:9000/api/plugins/
→org.graylog.plugins.archive/archives/graylog_386
Enter host password for user 'admin': ***************
{
  "archive_job_config" : {
    "archive_path" : "/tmp/graylog-archive",
    "max_segment_size" : 524288000,
    "segment_filename_prefix" : "archive-segment",
    "metadata_filename" : "archive-metadata.json",
    "source_histogram_bucket_size" : 86400000,
    "restore_index_batch_size" : 1001,
    "segment_compression_type": "SNAPPY"
  },
  "system_job" : {
    "id" : "cd7ebfa0-079b-11e6-9e1b-fa163e6e9b8a",
    "description" : "Archives indices and deletes them",
    "name" : "org.graylog.plugins.archive.job.ArchiveCreateSystemJob",
    "info" : "Archiving documents in index: graylog_386",
    "node_id" : "c5df7bff-cafd-4546-ac0a-5ccd2ba4c847",
    "started_at" : "2016-04-21T08:34:03.034Z",
    "percent_complete" : 0,
    "provides_progress" : true,
    "is_cancelable" : true
  }
}
```

That command started a system job in the Graylog server to create an archive for index `graylog_386`. The `system_job.id` can be used to check the progress of the job.

The REST API can be used to automate other archive related tasks as well, like restoring and deleting archives or updating the archive config. See the REST API browser on your Graylog server for details.

31.2.2 Restoring Archives

Note: The restore process adds load to your Elasticsearch cluster because all messages are basically **re-indexed**. Please make sure to keep this in mind and test with smaller archives to see how your cluster behaves. Also use the *Restore Index Batch Size* setting to control the Elasticsearch batch size on re-index.

Graylog Archive offers two ways to restore archived indices.

- *Web Interface*
- *REST API*

Graylog Archive restores all indices into the "Restored Archives" index set to avoid conflicts with the original indices. (should those still exist)

Restored indices are also marked as `reopened` so they are **ignored** by index retention jobs and are not closed or deleted. That means you have to manually delete any restored indices **manually** once you do not need them anymore.

Web Interface

In the web interface you can restore an archive on the "Enterprise/Archives" page by selecting an archive from the list, open the archive details and clicking the "Restore Index" button.

REST API

As with archive creation you can also use the REST API to restore an archived index into the Elasticsearch cluster:

```
$ curl -s -u admin -H 'X-Requested-By: cli' -X POST http://127.0.0.1:9000/api/plugins/
→org.graylog.plugins.archive/archives/graylog_386/restore
Enter host password for user 'admin': ****************
{
    "archive_metadata": {
        "archive_id": "graylog_307",
        "index_name": "graylog_307",
        "document_count": 491906,
        "created_at": "2016-04-14T14:31:50.787Z",
        "creation_duration": 142663,
        "timestamp_min": "2016-04-14T14:00:01.008Z",
        "timestamp_max": "2016-04-14T14:29:27.639Z",
        "id_mappings": {
            "streams": {
                "56fbafe0fb121a5309cef297": "nginx requests"
            },
            "inputs": {
                "56fbafe0fb121a5309cef290": "nginx error_log",
                "56fbafe0fb121a5309cef28d": "nginx access_log"
            },
            "nodes": {
                "c5df7bff-cafd-4546-ac0a-5ccd2ba4c847": "graylog.example.org"
            }
        },
        "histogram_bucket_size": 86400000,
        "source_histogram": {
            "2016-04-14T00:00:00.000Z": {
```

(continues on next page)

```
        "example.org": 227567
      }
    },
    "segments": [
      {
        "path": "archive-segment-0.gz",
        "size": 21653755,
        "raw_size": 2359745839,
        "compression_type": "SNAPPY"
        "checksum": "751e6e76",
        "checksum_type": "CRC32"
      }
    ],
    "index_size": 12509063,
    "index_shard_count": 4
  },
  "system_job": {
    "id": "e680dcc0-07a2-11e6-9e1b-fa163e6e9b8a",
    "description": "Restores an index from the archive",
    "name": "org.graylog.plugins.archive.job.ArchiveRestoreSystemJob",
    "info": "Restoring documents from archived index: graylog_307",
    "node_id": "c5df7bff-cafd-4546-ac0a-5ccd2ba4c847",
    "started_at": "2016-04-21T09:24:51.468Z",
    "percent_complete": 0,
    "provides_progress": true,
    "is_cancelable": true
  }
}
```

The returned JSON payload contains the archive metadata and the system job description that runs the index restore process.

Restore into a separate cluster

As said earlier, restoring archived indices slow down your indexing speed because of added load. If you want to completely avoid adding more load to your Elasticsearch cluster, you can restore the archived indices on a different cluster.

To do that, you only have to transfer the archived indices to a different machine and put them into a configured *Backend*.

Each index archive is in a separate directory, so if you only want to transfer one index to a different machine, you only have to copy the corresponding directory into the backend.

Example:

```
$ tree /tmp/graylog-archive
  /tmp/graylog-archive
  ├── graylog_171
  │   ├── archive-metadata.json
  │   └── archive-segment-0.gz
  ├── graylog_201
  │   ├── archive-metadata.json
  │   └── archive-segment-0.gz
  ├── graylog_268
  │   ├── archive-metadata.json
```

```
            └── archive-segment-0.gz
    ├── graylog_293
    │   ├── archive-metadata.json
    │   └── archive-segment-0.gz
    ├── graylog_307
    │   ├── archive-metadata.json
    │   └── archive-segment-0.gz
    ├── graylog_386
    │   ├── archive-metadata.json
    │   └── archive-segment-0.gz
    └── graylog_81
        ├── archive-metadata.json
        └── archive-segment-0.gz

7 directories, 14 files
```

31.2.3 Searching in Restored Indices

Once an index has been restored from an archive it will be used by search queries automatically.

Every message that gets restored into an Elasticsearch index gets a special `gl2_archive_restored` field with value `true`. This allows you to only search in restored messages by using a query like:

```
_exists_:gl2_archive_restored AND <your search query>
```

Example:

If you want to exclude all restored messages from you query you can use:

```
_missing_:gl2_archive_restored AND <your search query>
```

AUDIT LOG

Audit Log keeps track of changes made by users to a Graylog system.

It records all state changes into the database and makes it possible to search, filter and export all audit log entries.

Note: Audit Log is a commercial feature and part of Graylog Enterprise.

32.1 Setup

Graylog Audit Log is a commercial feature that can be installed in addition to the Graylog open source server.

32.1.1 Installation

Audit Log functionality is part of the Graylog Enterprise plugin, please check the *Graylog Enterprise setup page* for details on how to install it.

32.1.2 Configuration

Graylog Audit Log provides two ways of writing audit log entries:

1. Database
2. Log file via log4j2 appender

Logging to the database is always enabled and cannot be disabled.

Note: All configuration needs to be done in the Graylog server configuration file **and** in the logging configuration. (only if the log4j2 appender is enabled) Check the *default file locations page* for details.

The web interface can show the current configuration.
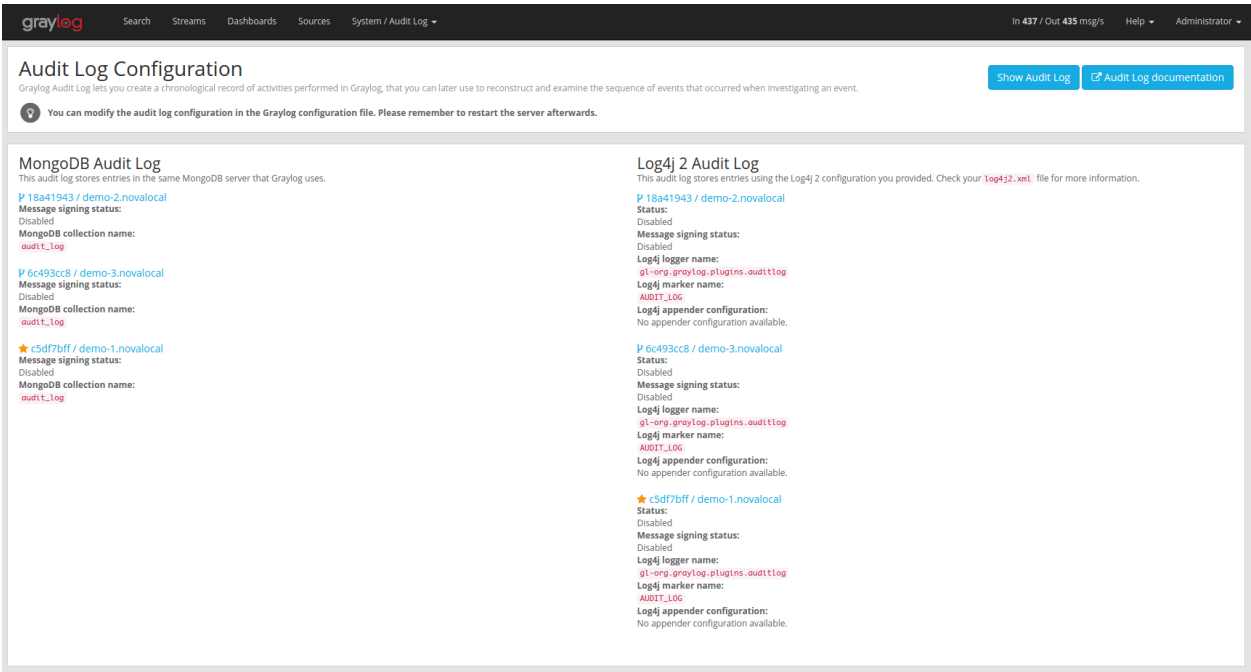
Database Configuration Options

The default MongoDB audit log has a few configuration options available.

Table 1: Configuration Options

Name	Description
`auditlog_mongodb_keep_entries`	delete audit log entries older that configured interval
`auditlog_mongodb_cleanup_interval`	interval of the audit log entry cleanup job
`auditlog_mongodb_collection`	the MongoDB collection to store the audit log entries in

auditlog_mongodb_keep_entries

This configures the interval after which old audit log entries in the MongoDB database will be deleted. You have to use values like `90d` (90 days) to configure the interval.

> **Warning:** Make sure to configure this to fit your needs. Deleted audit log entries are gone forever!

The default value for this is `365d`.

Example:

```
auditlog_mongodb_keep_entries = 365d
```

auditlog_mongodb_cleanup_interval

This configures the interval of the background job that periodically deletes old audit log entries from the MongoDB database. You have to use values like `1h` (1 hour) to configure the interval.

The default value for this is `1h`.

Example:

```
auditlog_mongodb_cleanup_interval = 1h
```

auditlog_mongodb_collection

This configures the name of the MongoDB collection where audit log entries will be stored.

The default value for this is `audit_log`.

Example:

```
auditlog_mongodb_collection = audit_log
```

Log4j2 Configuration Options

The optional log4j2 audit log appender has a few configuration options available.

Note: To configure the log4j2 appender you have to edit the Graylog server configuration file **and** the log4j2.xml file for your setup!

Table 2: Configuration Options

Name	Description
`auditlog_log4j_enabled`	whether the log4j2 appender is enabled or not
`auditlog_log4j_logger_name`	log4j2 logger name
`auditlog_log4j_marker_name`	log4j2 marker name

auditlog_log4j_enabled

The log4j2 audit log appender is disabled by default and can be enabled by setting this option to `true`.

The default value for this is `false`.

Example:

```
auditlog_log4j_enabled = true
```

auditlog_log4j_logger_name

This configures the log4j2 logger name of the audit log.

The default value for this is `gl-org.graylog.plugins.auditlog`.

Example:

```
auditlog_log4j_logger_name = graylog-auditlog
```

auditlog_log4j_marker_name

This configures the log4j2 marker name for the audit log.

The default value for this is `AUDIT_LOG`.

Example:

```
auditlog_log4j_marker_name = AUDIT_LOG
```

Log4j2 Appender Configuration

To write audit log entries into a file you have to enable the log4j2 appender in your Graylog configuration file **and** add some configuration to the `log4j2.xml` file that is used by your server process.

The `log4j2.xml` file location is dependent on your deployment method. so please check the *default file locations page*.

An existing `log4j2.xml` config file needs another `<Logger/>` statement in the `<Loggers/>` section and an additional appender in the `<Appenders/>` section of the file.

> **Warning:** The file on your system might look different than the following example. Make sure to only add the audit log related snippets to your config and do not remove anything else!

Example `log4j2.xml` file with audit log enabled:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration packages="org.graylog2.log4j" shutdownHook="disable">
    <Appenders>
        <!-- Graylog server log file appender -->
        <RollingFile name="rolling-file" fileName="/var/log/graylog-server/server.log
↪" filePattern="/var/log/graylog-server/server.log.%i.gz">
            <PatternLayout pattern="%d{yyyy-MM-dd'T'HH:mm:ss.SSSXXX} %-5p [%c{1}] %m%n
↪"/>
            <Policies>
                <SizeBasedTriggeringPolicy size="50MB"/>
            </Policies>
            <DefaultRolloverStrategy max="10" fileIndex="min"/>
        </RollingFile>

        <!-- #################################################### -->
        <!-- Rotate audit logs daily -->
        <RollingFile name="AUDITLOG" fileName="/var/log/graylog-server/audit.log"␣
↪filePattern="/var/log/graylog-server/audit-%d{yyyy-MM-dd}.log.gz">
```

(continues on next page)

```
                <PatternLayout>
                    <Pattern>%d - %m - %X%n</Pattern>
                </PatternLayout>
                <Policies>
                    <TimeBasedTriggeringPolicy />
                </Policies>
        </RollingFile>
        <!-- ################################################### -->
    </Appenders>
    <Loggers>
        <Logger name="org.graylog2" level="info"/>

        <!-- ################################################### -->
        <!-- Graylog Audit Log.  The logger name has to match the "auditlog_log4j_
→logger_name" setting in the Graylog configuration file -->
        <Logger name="graylog-auditlog" level="info" additivity="false">
            <AppenderRef ref="AUDITLOG"/>
        </Logger>
        <!-- ################################################### -->

        <Root level="warn">
            <AppenderRef ref="rolling-file"/>
        </Root>
    </Loggers>
</Configuration>
```

The config snippets between the `<!-- ######### -->` tags have been added to the existing `log4j2.xml` file.

Make sure that the `name` in the `<Logger />` tag matches the configured `auditlog_log4j_logger_name` in your Graylog server configuration. Otherwise you will not see any log entries in the log file.

Caveats

You have to make sure that the log4j2 related settings in the Graylog server config file and the `log4j2.xml` file are the same on **every node in your cluster**!

Since every Graylog server writes its own audit log entries when the Graylog Enterprise plugin is installed, the log files configured in the `log4j2.xml` file are written on every node. But **only** the entries from the local node will show up in that file.

If you have more than one node, you have to search in all configured files on all nodes to get a complete view of the audit trail.

32.2 Usage

Once you installed the Graylog Enterprise plugin, Graylog will automatically write audit log entries into the database.

32.2.1 View Audit Log Entries

Graylog Audit Log adds a new page to the web interface which can be reached via "Enterprise/Audit Log". You can view and export existing audit log entries in the database.

It also provides a simple search form to search and filter for audit events you are interested in.

32.2.2 Expand Event Details

Every row in the audit event entry table is clickable. Once clicked it will reveal the details of the audit event.

All audit events have static fields like *actor*, *object* and others. In addition to that, every event has some event specific fields.

The fields on the left side in the details are the static fields every event has and the fields on the right side are the event specific fields.

32.2.3 Search & Filter

To make it easier to get to the audit log entries you need, the audit log UI provides a simple query language to search and filter the audit log entries.

You can either enter one or more words into the search field or choose to look for some specific fields in the audit log entries.

Table 3: Available Fields

Name	Description
`actor`	the user that triggered the audit event
`namespace`	the namespace of the audit event; might be different in plugins
`object`	the object of the audit event; what has been changed
`action`	name of the action that has been executed on the object
`success_status`	if the action failed or succeeded
`message`	the actual audit event message

Search for text in the message

If you just want to find some text in the audit event message, you can enter the word you are looking for into the search bar

Search for specific fields

You can also filter the entries for specific fields like the `actor`.

If you want to filter for all events triggered by the user *jane* you can enter `actor:jane` into the search bar.

Maybe you want to filter for events for more than one actor. That can be done by using either `actor:jane,john` or `actor:jane actor:john`.

Or you want to find all audit events which have **not** been triggered by a user. Add a `-` in front of the field name to negate the condition. To show all events **except** those created by user *jane* you can add `-actor:jane` to the search field.

You can mix and match several field queries to find the entries you need. Here are some more examples.

- `actor:jane,john -namespace:server` get all events by users *jane* and *john* which are not in the *server* namespace

- `index action:create` get all events which have the word *index* in the event message and where the action is *create*

- `message:index action:create` same as above, just with an explicit field selector for the message field

32.2.4 Export Entries

If the simple entry viewer is not enough, you can also export the result of your query as JSON or CSV to further process it.

The "Export Results" button next to the search bar can be used to do that.

Note: The export from the UI is currently limited to the newest 10,000 entries. Use the REST API if you need a bigger export.

Export via REST API

If you want to backup the audit log entries or make them available to another system, you can use the REST API to export them.

Example:

```
# Export 20,000 audit log entries in JSON format
curl -u admin:<admin-password> http://127.0.0.1:9000/api/plugins/org.graylog.plugins.
↪auditlog/entries/export/json?limit=20000

# Export 5,000 audit log entries with actor "jane" in CSV format
curl -u admin:<admin-password> http://127.0.0.1:9000/api/plugins/org.graylog.plugins.
↪auditlog/entries/export/csv?limit=5000&query=actor:jane
```

Note: Make sure the `query` parameter is properly escaped if it contains whitespace.

REPORTING

The Reporting feature enables you to create and customize your own reports by using dashboard widgets, schedule reports to be automatically delivered to the people who require them, and manually send or download reports as PDF files at any time.

Additionally we also offer you historic information of report delivery, so you can verify that the scheduled reporting deliveries are working as expected.

Note: Reporting is a commercial feature and part of Graylog Enterprise.

33.1 Setup

Reporting is a commercial Graylog feature that can be installed in addition to the Graylog open source server.

33.1.1 Installation

Reporting is part of the Graylog Enterprise plugin, please check the *Graylog Enterprise setup page* for details on how to install it.

The PDF generation needs the `fontconfig` package installed on the server it is running on.

On a Debian based system use `apt` to install it, e.g.:

```
$ sudo apt-get install fontconfig
```

Respectively on a RedHat based systems use:

```
$ sudo yum install fontconfig
```

33.1.2 Configuration

In most cases you can use the Reporting functionality without making any changes to your Graylog configuration file (check the *default file locations page* to see where you can find it). Below, you will find all available configuration options in case you need to do some advanced configuration.

Table 1: Configuration Options

Name	Description
bin_dir	Directory with binaries needed for PDF generation.
data_dir	Cache directory for PDF generation.
report_disable_sandbox	Disables report generation sandbox.
report_generation_timeout_in_seconds	Timeout in seconds to wait for a report generation.
report_user	Internal user to generate reports.
report_render_uri	URI to connect to Graylog Web Interface.
report_render_engine_port	Port to communicate with background process.

bin_dir

Default value: `bin` - relative to Graylog working directory

The default distribution comes with two binaries needed for PDF generation 'headless_shell' and 'chromedriver'. These binaries are usually located in `/usr/share/graylog-server/bin`.

data_dir

Default value: `data` - relative to Graylog working directory

The PDF generation happens on disk in the first place so Graylog needs a place to write out temporary files. The system packages create `/var/lib/graylog-server` for this purpose. Make sure this directory is correctly configured and read-, and writable for the Graylog Server user.

report_disable_sandbox

Default value: `false`.

To ensure the maximum security in your system, the reporting generation process runs inside a sandbox, which provides a restricted environment for the application. That sandbox can only be used when the process is executed as a normal user, as the `root` user has special administrative privileges that could grant a potential attacker full access to your system.

We recommend leaving this configuration option set to `false`.

Unfortunately, there are two scenarios where the security features provided by the sandbox cannot be used:

- Environments where you want or must use the `root` user to run reporting generation.

- Environments that provide limited kernel capabilities. On the one hand Docker containers limit the kernel capabilities in a way that sandboxing doesn't work. On the other hand some RedHat/CentOS based systems come with older kernel versions which also lack the necessary capabilities. Systems with a kernel version >= 4.x should be fine for the default settings.

In case your Graylog server runs in one of those scenarios, you may consider disabling the sandbox.

Please note that this option only affects the reporting generation process, not the Graylog server.

report_generation_timeout_seconds

Default value: `180`.

Time in seconds to wait for a report to load in the background.

To ensure all widgets in your report have time to fetch their data and load, Graylog will wait up to the value set to this configuration option. When a report takes longer than that to load, the report generation will fail and Graylog will log the error in its logs.

In case reports in your Graylog setup are not being generated and the server displays a timeout error, you may need to increase this value.

report_user

Default value: `graylog-report`.

Graylog user that will be used internally to generate reports in the background. To ensure the user has access to all required information, this user must have the *Report System (Internal)* role assigned.

report_render_uri

Default value: `$http_publish_uri`.

Customize the URI the background process uses to connect to the web interface. By default it uses the value of the *http_publish_uri* option in your Graylog configuration file.

report_render_engine_port

Default value: `9515`.

Customize the port used to communicate with the background process.

33.2 Usage

Reports let you combine multiple Dashboard widgets to create a document that can display certain information contained within a Graylog system in an organized format for specific purposes.

Note: Reports are based on Dashboard widgets, so please ensure you understand *Dashboards* before you get started.

33.2.1 Creating Reports

You can create a report on the "Enterprise/Reports" page in the web interface.
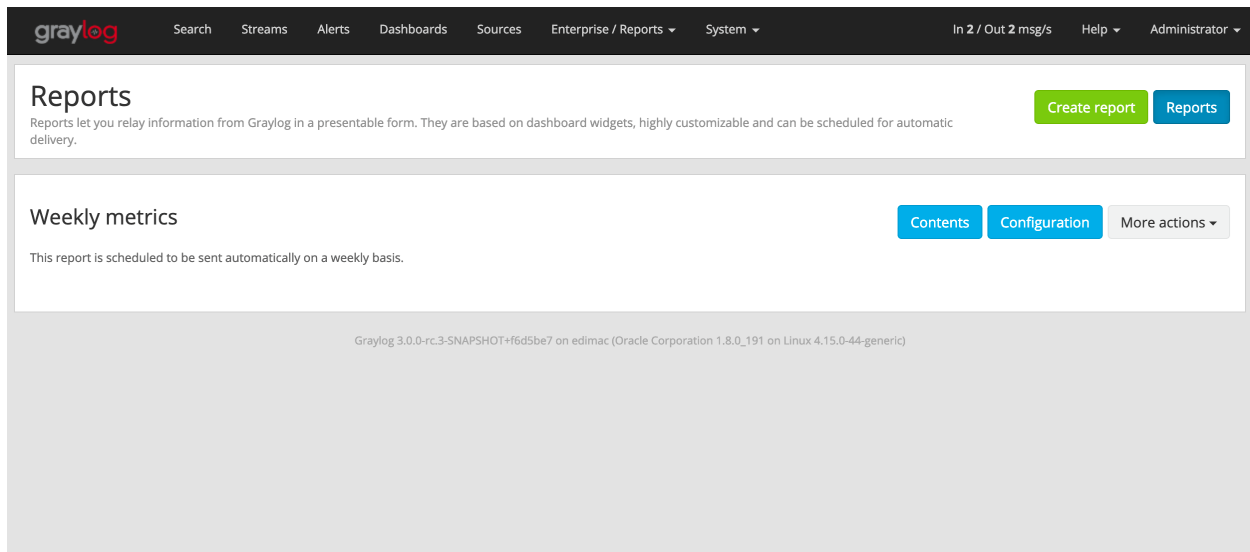
Click on the "Create Report" button to get started. That page contains two different sections:

- **Contents** You can use the form to configure the report's cover page, a short description, and select the widgets that will be part of the report.

- **Summary** This information will usually appear on your right and will follow you as you scroll through the page. It displays a summary of the data included in the report.

Once you are satisfied with the content that will make part of your report, click on "Create report" to store that configuration in the database.

You can change the selected contents of a report any time by going to the "Enterprise/Reports" page and clicking on the "Contents" button for the report you wish to modify.

33.2.2 Configure Reports

The Report Configuration page provides options to schedule the report for delivery and also to configure the layout of the report.

Go to the "Enterprise/Reports" page and click on "Configuration" to open the Report Configuration page.

Scheduling

> **Warning:** Please ensure the *email configuration* in your Graylog configuration file is working before you enable report scheduling.

> **Note:** Scheduling Reports will use resources in the background on both your Graylog and ES cluster. To avoid performance issues, make sure to allocate enough resources for your setup and also disable scheduling of Reports you don't need to be sent automatically.

In the Scheduling section you can configure how often the report will be sent. It is possible to send reports on a daily, weekly or monthly basis.

Here you can also add a subject and body to the email that will contain the report and select Graylog users or external email addresses that should receive the report as email.

Once you update the information, make sure to click on "Update scheduling" to save your changes.

Layout

Much like in a Dashboard, you can drag and drop widgets on the virtual sheet of paper to select the orders the widgets should go in the report. Rearranging widgets will save the change in the layout automatically.

Please note that the cover page will always be the first page of the report, and the next page will start with the report description followed by all widgets in the configured order.

33.2.3 History

As the background generation of reports may fail, the Report History page can help you discover if there were any errors while generating and sending a report in the background.

To open the Report history page for a report, click on the "More actions" button for that report, and select "Report history".

33.2.4 Generating Report On Demand

Download manually

You can generate and download a report manually from the web interface. To do so, go to the "Enterprise/Reports" page, click on the "More actions" button for the report you want to download, and select "Download report now".

Please take into account that the report generation may take a while.

Send report as email manually

In addition to downloading a report on demand, you may also generate and send the report at any time by clicking on "More actions" and "Send report now" on the "Enterprise/Reports" page.

CHANGELOG

34.1 Graylog Enterprise 3.1.4

Released: 2020-01-14

Fixed

- Only write archive metadata if the archiving process succeeded.
- Improve resiliency of widgets in reports.

34.2 Graylog Enterprise 3.1.3

Released: 2019-11-06

Fixed

- Fix problem with correlating events created by aggregation event definitions.
- Remove incomplete archive directory when archive job fails or is stopped.

34.3 Graylog Enterprise 3.1.2

Released: 2019-09-12

No changes since 3.1.1.

34.4 Graylog Enterprise 3.1.1

Released: 2019-09-04

No changes since 3.1.0.

34.5 Graylog Enterprise 3.1.0

Released: 2019-08-16

Added

- Add correlation engine and UI for new alerts and events system.
- Add Enterprise job scheduler implementation.

Removed

- Moved views feature to open-source. (except parameter support)

Fixed

- Fix report service memory leak.
- Fix auto-completion in drop-down fields.
- Fix rendering of archive configuration page

34.6 Graylog Enterprise 3.0.2

Released: 2019-05-03

Integrations Plugin

- Improve Graylog Forwarder configuration defaults.
- Improve Graylog Forwarder error handling.
- Update Graylog Forwarder dependencies.

34.7 Graylog Enterprise 3.0.1

Released: 2019-04-01

- Fix missing authorization checks in the license management.
- Fix view sharing issue for regular users.
- Fix memory leak in the reporting system.

Integrations Plugin

- Add Graylog Forwarder feature.

34.8 Graylog Enterprise 3.0.0

Released: 2019-02-14

- Announcement blog post: https://www.graylog.org/post/announcing-graylog-v3-0-ga
- Upgrade notes: *Upgrading to Graylog 3.0.x*

A detailed changelog is following soon!

Integrations Plugin

- Add Script Alert Notification

34.9 Graylog Enterprise 2.5.2

Released: 2019-03-15

34.9.1 Plugin: License

- Add missing permissions to license HTTP API resources.
- Only show upcoming license expiration warning to admin users.

34.10 Graylog Enterprise 2.5.1

Released: 2018-12-19

No changes since 2.5.0.

34.11 Graylog Enterprise 2.5.0

Released: 2018-11-30

No changes since 2.4.6.

34.12 Graylog Enterprise 2.4.7

Released: 2019-03-01

34.12.1 Plugin: License

- Add missing authorization checks to license resources.

34.13 Graylog Enterprise 2.4.6

Released: 2018-07-16

No changes since 2.4.5.

34.14 Graylog Enterprise 2.4.5

Released: 2018-05-28

No changes since 2.4.4.

34.15 Graylog Enterprise 2.4.4

Released: 2018-05-02

No changes since 2.4.3.

34.16 Graylog Enterprise 2.4.3

Released: 2018-01-24

No changes since 2.4.2.

34.17 Graylog Enterprise 2.4.2

Released: 2018-01-24

No changes since 2.4.1.

34.18 Graylog Enterprise 2.4.1

Released: 2018-01-19

No changes since 2.4.0.

34.19 Graylog Enterprise 2.4.0

Released: 2017-12-22

No changes since 2.4.0-rc.2.

34.20 Graylog Enterprise 2.4.0-rc.2

Released: 2017-12-20

No changes since 2.4.0-rc.1.

34.21 Graylog Enterprise 2.4.0-rc.1

Released: 2017-12-19

No changes since 2.4.0-beta.4.

34.22 Graylog Enterprise 2.4.0-beta.4

Released: 2017-12-15

34.22.1 Plugin: License

- The license page now shows more details about the installed licenses.

34.23 Graylog Enterprise 2.4.0-beta.3

Released: 2017-12-04

No changes since 2.4.0-beta.2.

34.24 Graylog Enterprise 2.4.0-beta.2

Released: 2017-11-07

No changes since 2.4.0-beta.1.

34.25 Graylog Enterprise 2.4.0-beta.1

Released: 2017-10-20

34.25.1 Plugin: Archive

- Add support for Zstandard compression codec.

34.26 Graylog Enterprise 2.3.2

Released: 2017-10-19

34.26.1 Plugin: Archive

- Fix archive creation for indices with lots of shards.

34.27 Graylog Enterprise 2.3.1

Released: 2017-08-25

34.27.1 Plugin: Archive

- Lots of performance improvements (up to 7 times faster)
- Do not delete an index if not all of its documents have been archived

34.28 Graylog Enterprise 2.3.0

Released: 2017-07-26

34.28.1 Plugin: Archive

- Record checksums for archive segment files
- Add two archive permission roles "admin" and "viewer"
- Allow export of filenames from catalog search

34.29 Graylog Enterprise 2.2.3

Released: 2017-04-04

34.29.1 Plugin: Archive

- Metadata is now stored in MongoDB
- Preparation for storage backend support

34.30 Graylog Enterprise 2.2.2

Released: 2017-03-02

34.30.1 Plugin: Audit Log

- Extend integration with the Archive plugin

34.31 Graylog Enterprise 2.2.1

Released: 2017-02-20

34.31.1 Plugin: Archive

- Improve stability and smaller UI fixes

34.32 Graylog Enterprise 2.2.0

Released: 2017-02-09

34.32.1 Plugin: Archive

- Improve index set support

34.33 Graylog Enterprise 1.2.1

Released: 2017-01-26

34.33.1 Plugin: Archive

- Prepare the plugin to be compatible with the new default stream.

34.33.2 Plugin: Audit Log

- Add support for index sets and fix potential NPEs.
- Smaller UI improvements.

34.34 Graylog Enterprise 1.2.0

Released: 2016-09-14

https://www.graylog.org/blog/70-announcing-graylog-enterprise-v1-2

34.34.1 Plugin: Archive

• Add support for selecting which streams should be included in your archives.

34.34.2 Plugin: Audit Log

New plugin to keep track of changes made by users to a Graylog system by automatically saving them in MongoDB.

34.35 Graylog Enterprise 1.1

Released: 2016-09-01

• Added support for Graylog 2.1.0.

34.36 Graylog Enterprise 1.0.1

Released: 2016-06-08

Bugfix release for the archive plugin.

34.36.1 Plugin: Archive

Fixed problem when writing multiple archive segments

There was a problem when exceeding the max segment size so that multiple archive segments are written. The problem has been fixed and wrongly written segments can be read again.

34.37 Graylog Enterprise 1.0.0

Released: 2016-05-27

Initial Release including the Archive plugin.

34.37.1 Plugin: Archive

New features since the last beta plugin:

• Support for multiple compression strategies. (Snappy, LZ4, Gzip, None)