Graylog Documentation

Release 1.1.6

Graylog, Inc.

Aug 04, 2017

Contents

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

Contents:

CHAPTER 1

Architectural considerations

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

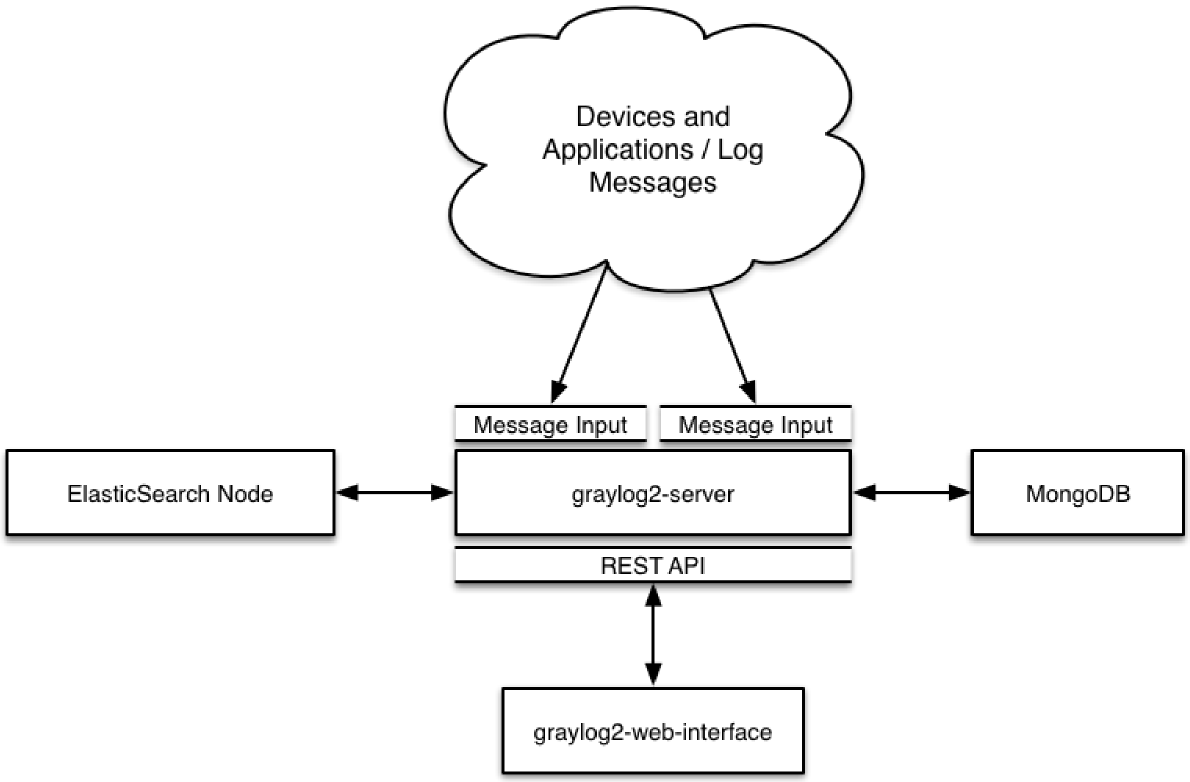
- `graylog-server` nodes should have a focus on CPU power.
- Elasticsearch nodes should have as much RAM as possible and the fastest disks you can get. Everything depends on I/O speed here.
- MongoDB is only being used to store configuration and the dead letter messages, and can be sized fairly small.
- `graylog-web-interface` nodes are mostly waiting for HTTP answers of the rest of the system and can also be rather small.
- `graylog-radio` nodes act as workers. They don't know each other and you can shut them down at any point in time without changing the cluster state at all.

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

MongoDB is only storing meta information and will be abstracted with a general database layer in future versions. This will allow you to use other databases like MySQL instead.

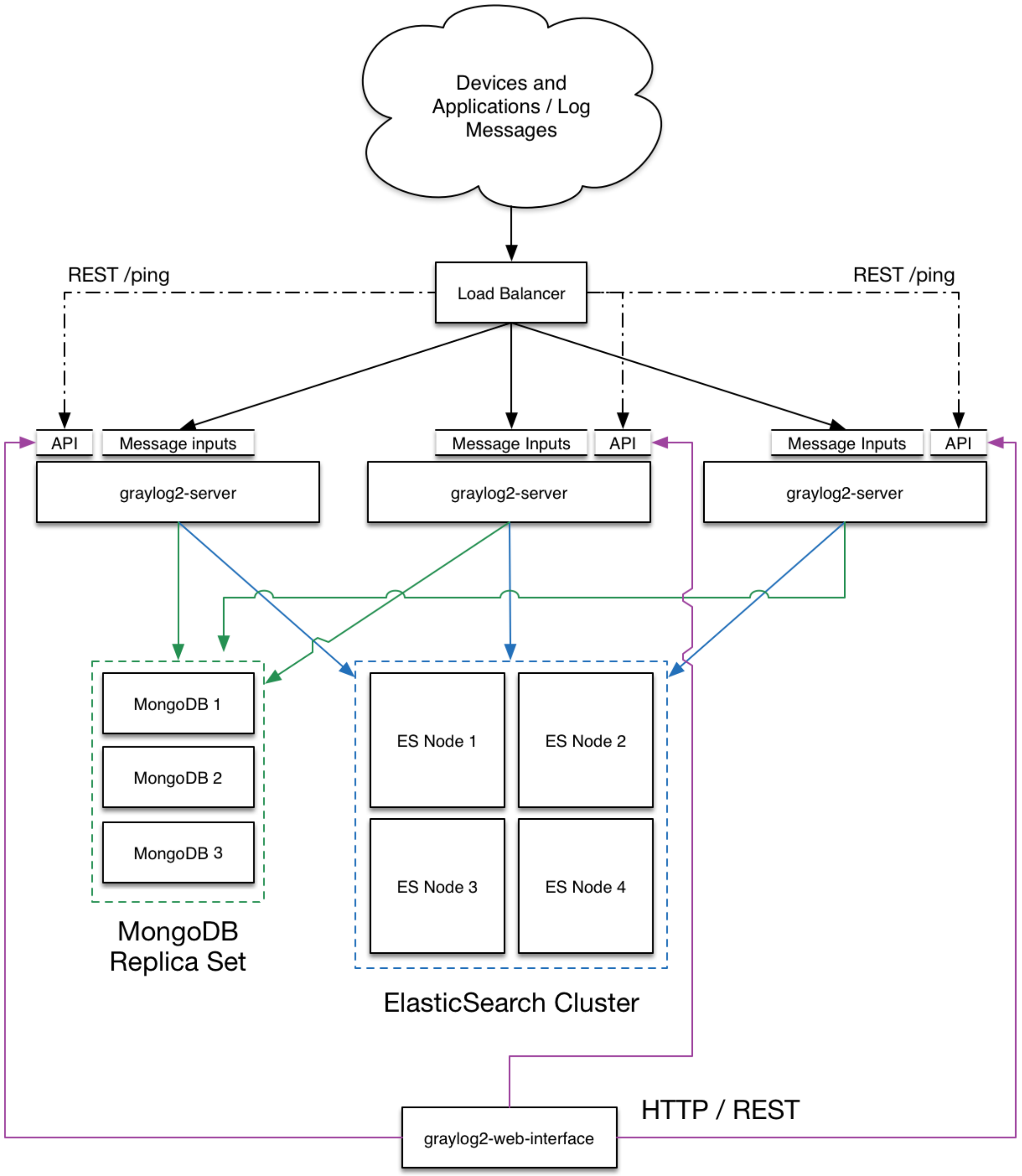
Minimum setup

This is a minimum Graylog setup that can be used for smaller, non-critical, or test setups. None of the components is redundant but it is easy and quick to setup.



Bigger production setup

This is a setup for bigger production environments. It has several `graylog-server` nodes behind a load balancer that share the processing load. The load balancer can ping the `graylog-server` nodes via REST/HTTP to check if they are alive and take dead nodes out of the cluster.



Highly available setup with Graylog Radio

Beginning with Graylog 1.0 on, we no longer recommend running Graylog Radio because we are now using a high-performant message journal (from the Apache Kafka project) in every *graylog-server* instance which is spooling all incoming messages to disk immediately and is able to buffer load spikes just at least as good as Graylog Radio was, but with less dependencies and maintenance overhead.

If you are running a setup with Graylog Radio we recommend to shut down the Graylog Radio architecture including AMQP or Kafka brokers completely and directly send messages to the *graylog-server* nodes. If you have been using Graylog Radio for load balancing, you should now put a classic load balancer in front of your *graylog-server* nodes.

This approach has been proven to work great in large high-throughput setups of several of our large scale customers and immensely reduced complexity of their setups.

The Kafka and AMQP inputs are still supported and can be used to build a custom setup using message brokers, if you want to keep using that. A reason for this might be that Graylog is not the only subscriber to the messages on the bus. However we would recommend to use Graylog forwarders to either write to a message bus after processing or write to other systems directly.

CHAPTER 2

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. (Note: The *virtual machine appliances* are suitable for production usage because they are also prepared to scale out to some level when required.)

System requirements

The Graylog server application has the following prerequisites:

- Some modern Linux distribution (Debian Linux, Ubuntu Linux, or CentOS recommended)
- Elasticsearch 1.6.2 or later (Elasticsearch 2.x is currently not supported)
- MongoDB 2.0 or later (latest stable version is recommended)
- Oracle Java SE 7 or later (Oracle Java SE 8 is supported, OpenJDK 7 and OpenJDK 8 also work; latest stable update is recommended)

The Graylog web interface has the following prerequisites:

- Some modern Linux distribution (Debian Linux, Ubuntu Linux, or CentOS recommended)
- Oracle Java SE 7 or later (Oracle Java SE 8 is supported, OpenJDK 7 and OpenJDK 8 also work; latest point release is recommended)

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

Choose an installation method:

Virtual Machine Appliances

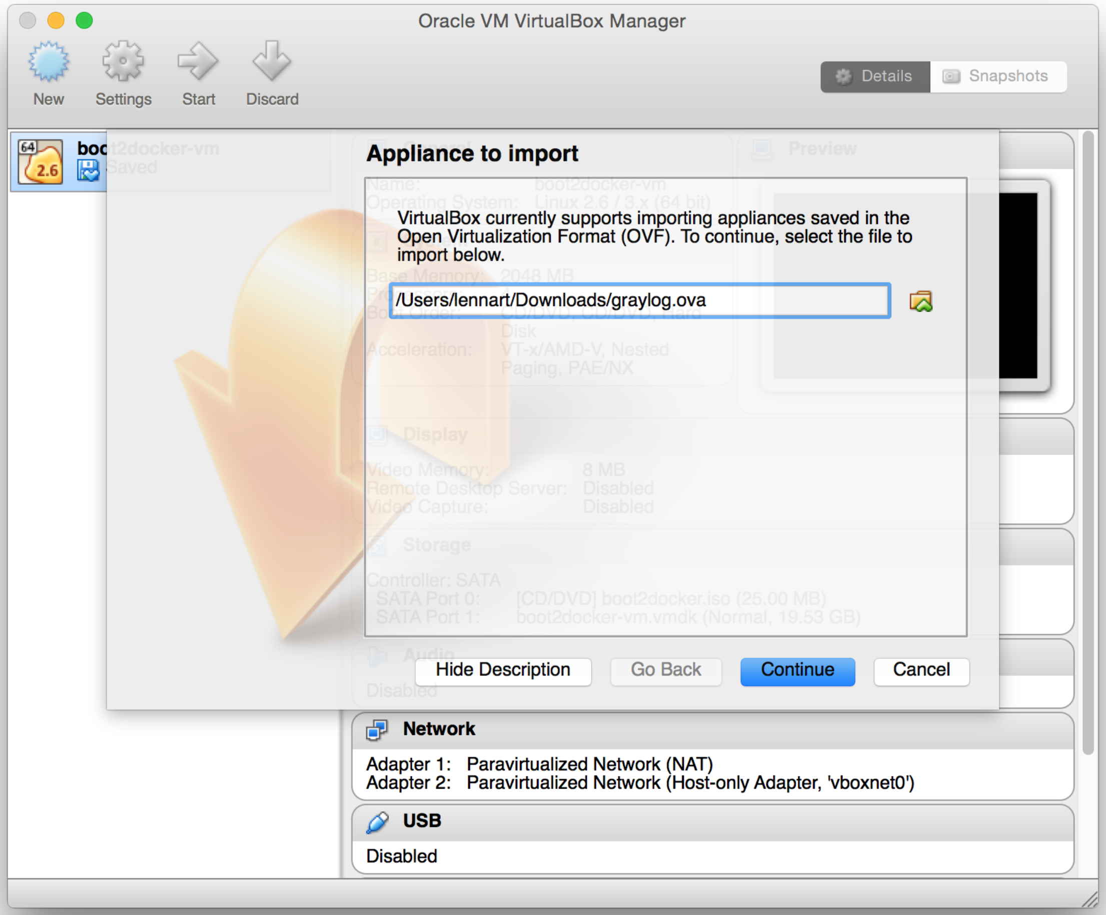
Download

Download the OVA image from here and save it to your disk locally.

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*:



Appliance to import

VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

/Users/lennart/Downloads/graylog.ova

Network

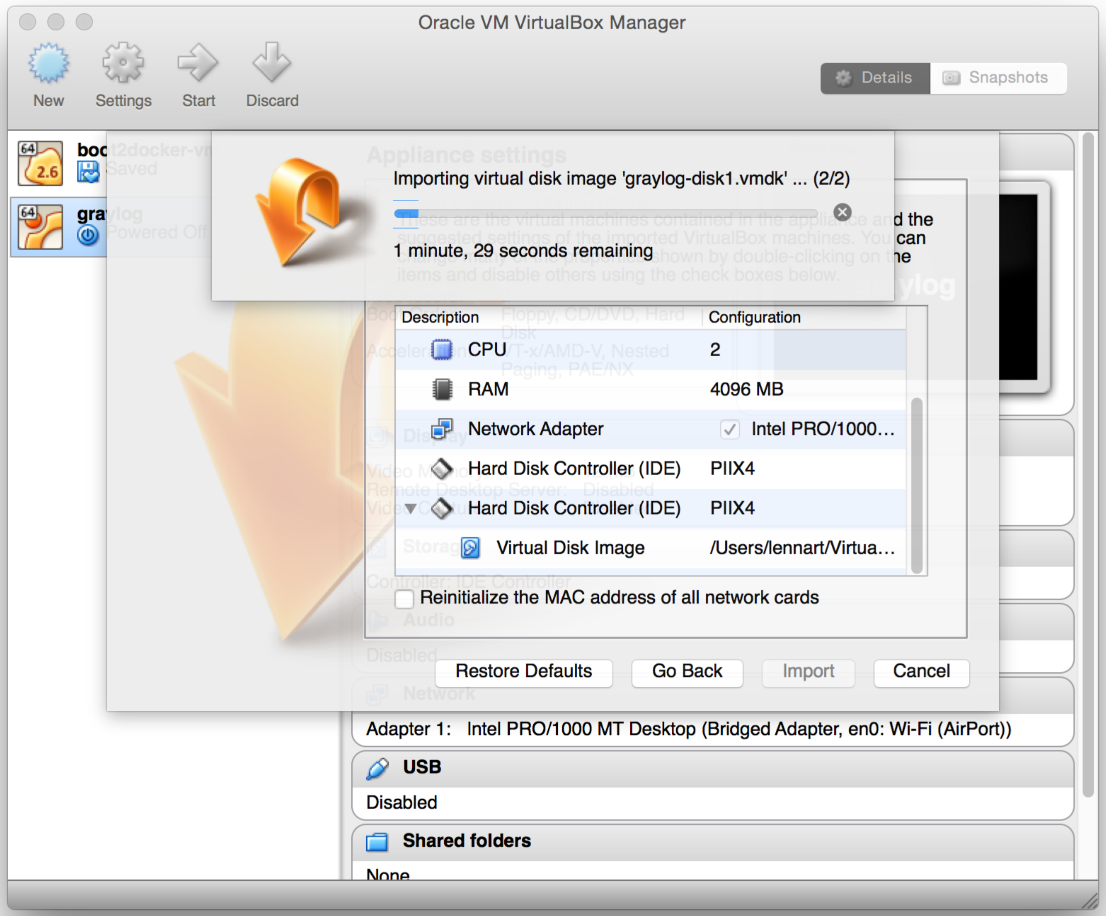
Adapter 1: Paravirtualized Network (NAT)
Adapter 2: Paravirtualized Network (Host-only Adapter, 'vboxnet0')

USB

Disabled

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:



Importing virtual disk image 'graylog-disk1.vmdk' ... (2/2)

1 minute, 29 seconds remaining

Description	Configuration
CPU	2
RAM	4096 MB
Network Adapter	✓ Intel PRO/1000…
Hard Disk Controller (IDE)	PIIX4
Hard Disk Controller (IDE)	PIIX4
Virtual Disk Image	/Users/lennart/Virtua…

Reinitialize the MAC address of all network cards

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, en0: Wi-Fi (AirPort))

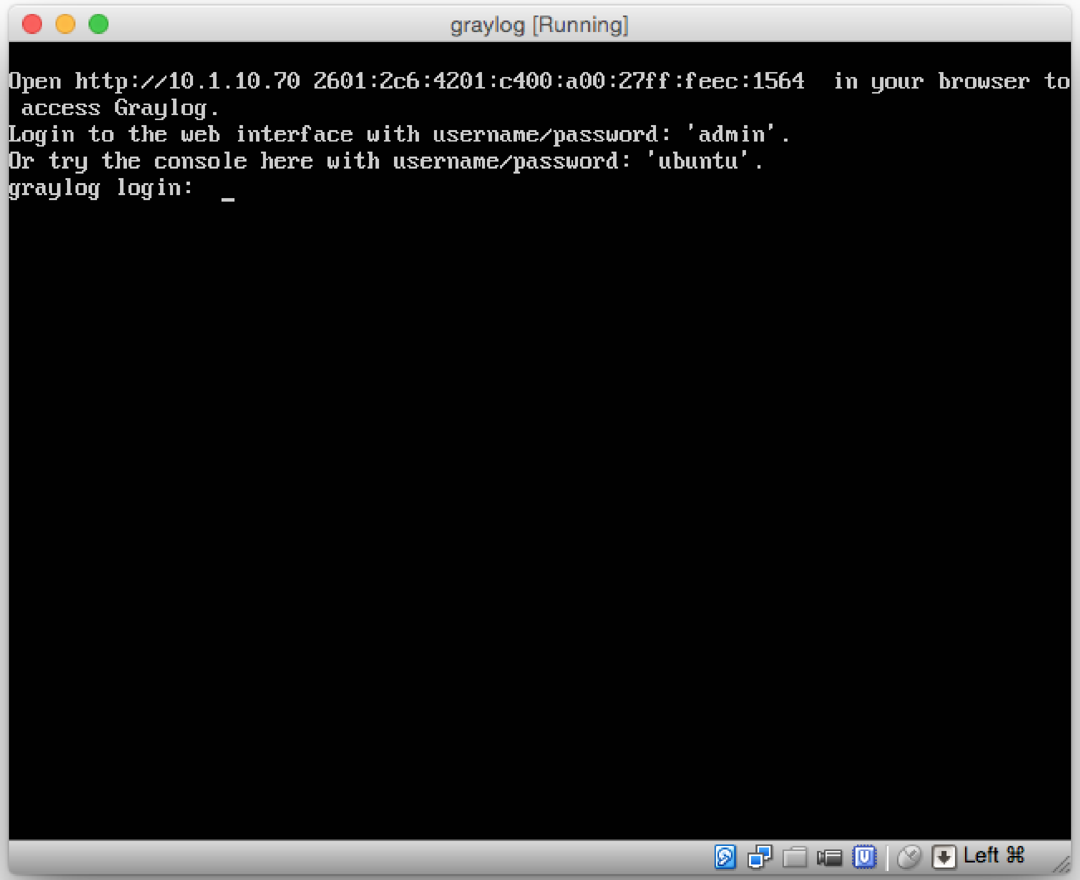
USB

Disabled

Shared folders

None

You can now start the VM and should see a login shell like this when the boot completed:



```
Open http://10.1.10.70 2601:2c6:4201:c400:a00:27ff:feec:1564  in your browser to
 access Graylog.
Login to the web interface with username/password: 'admin'.
Or try the console here with username/password: 'ubuntu'.
graylog login: _
```

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 if you plan to go into production with the appliance.

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* with the password *admin*.

Basic configuration

We are shipping the `graylog-ctl` tool with the virtual machine appliances to get you started with a customised setup as quickly as possible. Run these (optional) commands to configure the most basic settings of Graylog in the appliance:

```
sudo graylog-ctl set-email-config <smtp server> [--port=<smtp port> --user=<username>␣
↪--password=<password>]
sudo graylog-ctl set-admin-password <password>
sudo graylog-ctl set-timezone <zone acronym>
sudo graylog-ctl reconfigure
```

The `graylog-ctl` has much more functionality and is documented *here*. We strongly recommend to learn more about it to ensure smooth operation of your virtual appliance.

The graylog-ctl script

Some packages of Graylog (for example the *virtual machine appliances*) ship with a pre-installed `graylog-ctl` script to allow you easy configuration of certain settings.

Important: The manual setup, operating system packages, configuration management scripts etc are not shipping with this.

Configuration commands

The following commands are changing the configuration of Graylog:

Command	Description
sudo graylog-ctl set-admin-password <password>	Set a new admin password
sudo graylog-ctl set-admin-username <username>	Set a different username for the admin user
sudo graylog-ctl set-email-config <smtp server> [–port=<smtp port> –user=<username> –password=<password> –no-tls –no-ssl]	Configure SMTP settings to send alert mails
sudo graylog-ctl set-timezone <zone acronym>	Set Graylog's timezone. Make sure system time is also set correctly with `sudo dpkg-reconfigure tzdata`
sudo graylog-ctl set-retention –size=<Gb> OR –time=<hours> –indices=<number> [–journal=<Gb>]	Configure message retention
sudo graylog-ctl enforce-ssl	Enforce HTTPs for the web interface

After setting one or more of these options re-run:

```
sudo graylog-ctl reconfigure
```

You can also edit the full configuration files under `/opt/graylog/conf` manually. restart the related service afterwards:

```
sudo graylog-ctl restart graylog-server
```

Or to restart all services:

```
sudo graylog-ctl restart
```

Multi VM setup

At some point it makes sense to not run all services in one VM anymore. For performance reasons you maybe want to add more Elasticsearch nodes or want to run the web interface separately from the server components. You can reach this by changing IP addresses in the Graylog configuration files or you can use our canned configurations which come with the `graylog-ctl` command.

The idea is to have one VM which is a central point for other VMs to fetch all needed configuration settings to join your cluster. Typically the first VM you spin up is used for this task. Automatically an instance of etcd is started and filled with the necessary settings for other hosts.

For example to split the web interface from the rest of the setup, spin up two VMs from the same graylog image. On the first only start `graylog-server`, `elasticsearch` and `mongodb`:

```
vm1> sudo graylog-ctl set-admin-password sEcReT
vm1> sudo graylog-ctl reconfigure-as-backend
```

On the second VM, start only the web interface but before set the IP of the first VM to fetch configuration data from:

```
vm2> sudo graylog-ctl set-cluster-master <ip-of-vm1>
vm2> sudo graylog-ctl reconfigure-as-webinterface
```

This results in a perfectly fine dual VM setup. However if you want to scale this setup out by adding an additional Elasticsearch node, you can proceed in the same way:

```
vm3> sudo graylog-ctl set-cluster-master <ip-of-vm1>
vm3> sudo graylog-ctl reconfigure-as-datanode
```

The following configuration modes do exist:

Command	Services
sudo graylog-ctl reconfigure	Run all services on this box
sudo graylog-ctl reconfigure-as-backend	Run graylog-server, elasticsearch and mongodb
sudo graylog-ctl reconfigure-as-webinterface	Run only the web interface
sudo graylog-ctl reconfigure-as-datanode	Run only elasticsearch
sudo graylog-ctl reconfigure-as-server	Run graylog-server and mongodb (no elasticsearch)

Extend disk space

All data is stored in one directory `/var/opt/graylog/data`. In order to extend the disk space mount a second drive on this path. Make sure to move old data to the new drive before and give the graylog user permissions to read and write here.

Install Graylog plugins

The Graylog plugin directory is located in `/opt/graylog/plugin/`. Just drop a JAR file there and restart the server with `sudo graylog-ctl restart graylog-server` to load the plugin.

Install Elasticsearch plugins

Elasticsearch comes with a helper program to install additional plugins you can call it like this `sudo JAVA_HOME=/opt/graylog/embedded/jre /opt/graylog/elasticsearch/bin/plugin`

Install custom SSL certificates

During the first reconfigure run self signed SSL certificates are generated. You can replace this certificate with your own to prevent security warnings in your browser. Just drop the key and combined certificate file here: `/opt/graylog/conf/nginx/ca/graylog.crt` respectively `/opt/graylog/conf/nginx/ca/graylog.key`. Afterwards restart nginx with `sudo graylog-ctl restart nginx`.

Configure Message Retention

Graylog is keeping a defined amount of messages. It is possible to decide whether you want to have a set storage size or a set time period of messages. Additionally Graylog writes a so called Journal. This is used to buffer messages in case of a unreachable Elasticsearch backend. To configure those settings use the set-retention command.

Retention by disk size:

```
sudo graylog-ctl set-retention --size=3 --indices=10
sudo graylog-ctl reconfigure
```

Indices would be rotated when they reach a size of 3Gb and Graylog would keep up to 10 indices, resulting in 30Gb maximum disk space.

Retention by time:

```
sudo graylog-ctl set-retention --time=24  --indices=30
sudo graylog-ctl reconfigure
```

Indices would be rotated after 24 hours and 30 indices would be kept, resulting in 30 days of stored logs.

Both commands can be extended with the –journal switch to set the maximum journal size in Gb:

```
sudo graylog-ctl set-retention --time=24  --indices=30 --journal=5
sudo graylog-ctl reconfigure
```

Assign a static IP

Per default the appliance make use of DHCP to setup the network. If you want to access Graylog under a static IP please follow these instructions:

```
$ sudo ifdown eth0
```

Edit the file `/etc/network/interfaces` like this (just the important lines):

```
auto eth0
  iface eth0 inet static
  address <static IP address>
  netmask <netmask>
  gateway <default gateway>
  pre-up sleep 2
```

Activate the new IP and reconfigure Graylog to make use of it:

```
$ sudo ifup eth0
$ sudo graylog-ctl reconfigure
```

Wait some time until all services are restarted and running again. Afterwards you should be able to access Graylog with the new IP.

Upgrade Graylog

Always perform a full backup or snapshot of the appliance before proceeding. Only upgrade if the release notes say the next version is a drop-in replacement:

```
wget https://packages.graylog2.org/releases/graylog2-omnibus/ubuntu/graylog_latest.deb
sudo graylog-ctl stop
sudo dpkg -G -i graylog_latest.deb
sudo graylog-ctl reconfigure
```

Production readiness

You can use the Graylog appliances (OVA, Docker, AWS, ...) for small production setups but please consider to harden the security of the box before.

- Set another password for the default ubuntu user
- Disable remote password logins in /etc/ssh/sshd_config and deploy proper ssh keys
- Seperate the box network-wise from the outside, otherwise Elasticsearch can be reached by anyone

If you want to create your own customised setup take a look at our *other installation methods*.

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 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 for the following operating systems.

- Ubuntu 12.04, 14.04
- Debian 7, 8
- CentOS 6, 7

The repositories can be setup 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.

Make sure to install and configure Java (>= 7), MongoDB and Elasticsearch before starting the Graylog services.

Ubuntu 14.04

Download and install graylog-1.1-repository-ubuntu14.04_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-1.1-repository-ubuntu14.04_
↪latest.deb
$ sudo dpkg -i graylog-1.1-repository-ubuntu14.04_latest.deb
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install graylog-server graylog-web
```

After the installation successfully completed, Graylog can be started with the following commands:

```
$ sudo start graylog-server
$ sudo start graylog-web
```

Ubuntu 12.04

Download and install graylog-1.1-repository-ubuntu12.04_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-1.1-repository-ubuntu12.04_
↪latest.deb
$ sudo dpkg -i graylog-1.1-repository-ubuntu12.04_latest.deb
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install graylog-server graylog-web
```

After the installation successfully completed, Graylog can be started with the following commands:

```
$ sudo start graylog-server
$ sudo start graylog-web
```

Debian 7

Download and install graylog-1.1-repository-debian7_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-1.1-repository-debian7_
↪latest.deb
$ sudo dpkg -i graylog-1.1-repository-debian7_latest.deb
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install graylog-server graylog-web
```

After the installation successfully completed, Graylog can be started with the following commands:

```
$ sudo service graylog-server start
$ sudo service graylog-web start
```

Debian 8

Download and install graylog-1.1-repository-debian8_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-1.1-repository-debian8_
↪latest.deb
$ sudo dpkg -i graylog-1.1-repository-debian8_latest.deb
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install graylog-server graylog-web
```

After the installation successfully completed, Graylog can be started with the following commands:

```
$ sudo systemctl start graylog-server
$ sudo systemctl start graylog-web
```

CentOS 6

Download and install graylog-1.1-repository-el6_latest.rpm via `rpm(8)`:

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-1.1-repository-
↪el6_latest.rpm
$ sudo yum install graylog-server graylog-web
```

After the installation successfully completed, Graylog can be started with the following commands:

```
$ sudo service graylog-server start
$ sudo service graylog-web start
```

CentOS 7

Download and install graylog-1.1-repository-el7_latest.rpm via `rpm(8)`:

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-1.1-repository-
↪el7_latest.rpm
$ sudo yum install graylog-server graylog-web
```

After the installation successfully completed, Graylog can be started with the following commands:

```
$ sudo systemctl start graylog-server
$ sudo systemctl start graylog-web
```

Feedback

Please open an issue in the Github repository if you run into any packaging related issues. **Thank you!**

Chef, Puppet, Ansible, Vagrant

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.puppetlabs.com/graylog2/graylog2
- https://galaxy.ansible.com/list#/roles/3162

There are also official Vagrant images if you want to spin up a local virtual machine quickly:

- https://github.com/Graylog2/graylog2-images/tree/master/vagrant

Docker

Requirements

You need a recent *docker* version installed, take a look here for instructions.

This will create a container with all Graylog services running:

```
$ docker pull graylog2/allinone
$ docker run -t -p 9000:9000 -p 12201:12201 graylog2/allinone
```

Using the beta container

You can also run a pre-release or beta version of Graylog using Docker. Just replace *graylog2/allinone* with *graylog2/allinone-beta*. Note that you will have to replace this not only in the *docker run* command above but also in subsequent commands of this documentation. We only recommend to run beta versions if you are an experienced Graylog user and know what you are doing.

Usage

After starting the container, your Graylog instance is ready to use. You can reach the web interface by pointing your browser to the IP address of your Docker host: *http://<host IP>:9000*

The default login is Username: *admin*, Password: *admin*.

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 get through. You already exposed the default GELF port 12201, so it is a good idea to start a GELF TCP input there.

To start another input you have to expose the right port e.g. to start a raw TCP input on port 5555; stop your container and recreate it, whilst appending *-p 5555:5555* to your run argument. Similarly, the same can be done for UDP by appending *-p 5555:5555/udp* option. Then you can send raw text to Graylog like *echo 'first log message' | nc localhost 5555*

Additional options

You can configure the most important aspects of your Graylog instance through environment variables. In order to set a variable add a *-e VARIABLE_NAME* option to your *docker run* command. For example to set another admin password start your container like this:

```
$ docker run -t -p 9000:9000 -p 12201:12201 -e GRAYLOG_PASSWORD=SeCuRePwD graylog2/
↪allinone
```

Variable Name	Configuration Option
GRAYLOG_PASSWORD	Set admin password
GRAYLOG_USERNAME	Set username for admin user (default: admin)
GRAYLOG_TIMEZONE	Set [timezone (TZ)](http://en.wikipedia.org/wiki/List_of_tz_database_time_zones) you are in
GRAYLOG_SMTP_SERVER	Hostname/IP address of your SMTP server for sending alert mails
GRAYLOG_RETENTION	Configure how long or how many logs should be stored
GRAYLOG_NODE_ID	Set server node ID (default: random)
GRAYLOG_SERVER_SECRET	Set salt for encryption
GRAYLOG_MASTER	IP address of a remote master container (see multi container setup)
GRAYLOG_SERVER	Run only server components
GRAYLOG_WEB	Run web interface only
ES_MEMORY	Set memory used by Elasticsearch (syntax: 1024m). Defaults to 60% of host memory

Examples

Set an admin password:

```
GRAYLOG_PASSWORD=SeCuRePwD
```

Change admin username:

```
GRAYLOG_USERNAME=root
```

Set your local timezone:

```
GRAYLOG_TIMEZONE=Europe/Berlin
```

Set a SMTP server for alert e-mails:

```
GRAYLOG_SMTP_SERVER="mailserver.com"
```

Disable TLS/SSL for mail delivery:

```
GRAYLOG_SMTP_SERVER="mailserver.com --no-tls --no-ssl"
```

Set SMTP server with port, authentication, backlink URL and changed sender address:

```
GRAYLOG_SMTP_SERVER="example.com --port=465 --user=username@mailserver.com --
↪password=SecretPassword --from-email=graylog@example.com --web-url=http://my.
↪graylog.host"
```

Set a static server node ID:

```
GRAYLOG_NODE_ID=de305d54-75b4-431b-adb2-eb6b9e546014
```

Set a configuration master for linking multiple containers:

```
GRAYLOG_MASTER=192.168.3.15
```

Only start server services:

```
GRAYLOG_SERVER=true
```

Only run web interface:

```
GRAYLOG_WEB=true
```

Keep 30Gb of logs, distributed across 10 Elasticsearch indices:

```
GRAYLOG_RETENTION="--size=3 --indices=10"
```

Keep one month of logs, distributed across 30 indices with 24 hours of logs each:

```
GRAYLOG_RETENTION="--time=24 --indices=30"
```

Limit amount of memory Elasticsearch is using:

```
ES_MEMORY=2g
```

Persist data

In order to persist log data and configuration settings mount the Graylog data directory outside the container:

```
$ docker run -t -p 9000:9000 -p 12201:12201 -e GRAYLOG_NODE_ID=some-rand-omeu-
↪uidasnodeid -e GRAYLOG_SERVER_SECRET=somesecretsaltstring -v /graylog/data:/var/opt/
↪graylog/data -v /graylog/logs:/var/log/graylog graylog2/allinone
```

Please make sure that you always use the same node-ID and server secret. Otherwise your users can't login or inputs will not be started after creating a new container on old data.

Other volumes to persist:

Path	Description
/var/opt/graylog/data	Elasticsearch for raw log data and MongoDB as configuration store
/var/log/graylog	Internal logs for all running services
/opt/graylog/plugin	Graylog server plugins

Multi container setup

The Omnibus package used for creating the container is able to split Graylog into several components. This works in a Docker environment as long as your containers run on the same hardware respectively the containers need to have direct network access between each other. The first started container is the so called *master*, other containers can grab configuration options from here.

To setup two containers, one for the web interface and one for the server component do the following:

Start the *master* with Graylog server parts:

```
$ docker run -t -p 12900:12900 -p 12201:12201 -p 4001:4001 -e GRAYLOG_SERVER=true␣
↪graylog2/allinone
```

The configuration port 4001 is now accessible through the host IP address.

Start the web interface in a second container and give the host address as *master* to fetch configuration options:

```
$ docker run -t -p 9000:9000 -e GRAYLOG_MASTER=<host IP address> -e GRAYLOG_WEB=true␣
↪graylog2/allinone
```

SSL Support

Graylog comes with a pre-configured SSL configuration. On start-up time a self-signed certificate is generated and used on port 443 to provide the web interface via HTTPS. Simply expose the port like this:

```
$ docker run -t -p 443:443 graylog2/allinone
```

It is also possible to swap the certificate with your own files. To achieve this mount the CA directory to the Docker host:

```
$ docker run -t -p 443:443 -v /somepath/ca:/opt/graylog/conf/nginx/ca graylog2/
↪allinone
```

If you put a file called */somepath/ca/graylog.crt* respectively */somepath/ca/graylog.key* in place before starting the container, Graylog will pick up those files and make use of your own certificate.

Build

To build the image from scratch run:

```
$ docker build -t graylog .
```

Vagrant

Requirements

You need a recent *vagrant* version, take a look here.

Installation

These steps will create a Vagrant virtual machine with all Graylog services running:

```
$ wget https://raw.githubusercontent.com/Graylog2/graylog2-images/master/vagrant/
↪Vagrantfile
$ vagrant up
```

Usage

After starting the virtual machine, your Graylog instance is ready to use. You can reach the web interface by pointing your browser to the IP address of your localhost: *http://<host IP>:9000*

The default login is Username: *admin*, Password: *admin*.

Basic configuration

We are shipping the `graylog-ctl` tool with the virtual machine appliances to get you started with a customised setup as quickly as possible. Run these (optional) commands to configure the most basic settings of Graylog in the appliance:

```
sudo graylog-ctl set-email-config <smtp server> [--port=<smtp port> --user=<username>␣
↪--password=<password>]
sudo graylog-ctl set-admin-password <password>
sudo graylog-ctl set-timezone <zone acronym>
sudo graylog-ctl reconfigure
```

The `graylog-ctl` has much more functionality and is documented *here*. We strongly recommend to learn more about it to ensure smooth operation of your virtual appliance.

OpenStack

Installation

These steps will download the Graylog image, uncompress it and import it into the Openstack image store:

```
$ wget https://packages.graylog2.org/releases/graylog2-omnibus/qcow2/graylog.qcow2.gz
$ gunzip graylog.qcow2.gz
$ glance image-create --name='graylog' --is-public=true --container-format=bare --
↪disk-format=qcow2 --file graylog.qcow2
```

You should now see an image called *graylog* in the Openstack web interface under *Images*

Usage

Launch a new instance of the image, make sure to reserve at least 4GB ram for the instance. After spinning up, login with the username *ubuntu* and your selected ssh key. Run the reconfigure program in order to setup Graylog and start all services:

```
$ ssh ubuntu@<vm IP>
$ sudo graylog-ctl reconfigure
```

Open *http://<vm ip>* in your browser to access the Graylog web interface. Default username and password is *admin*.

Basic configuration

We are shipping the `graylog-ctl` tool with the virtual machine appliances to get you started with a customised setup as quickly as possible. Run these (optional) commands to configure the most basic settings of Graylog in the appliance:

```
sudo graylog-ctl set-email-config <smtp server> [--port=<smtp port> --user=<username>␣
↪--password=<password>]
sudo graylog-ctl set-admin-password <password>
sudo graylog-ctl set-timezone <zone acronym>
sudo graylog-ctl reconfigure
```

The `graylog-ctl` has much more functionality and is documented *here*. We strongly recommend to learn more about it to ensure smooth operation of your virtual appliance.

Amazon Web Services

AMIs

Select your AMI and AWS Region.

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.
- Login to the instance as user *ubuntu*
- Run *sudo graylog-ctl reconfigure*
- Open port 80 and ports for receiving log messages in the security group of the appliance

Open *http://<vm ip>* in your browser to access the Graylog web interface. Default username and password is *admin*.

Basic configuration

We are shipping the `graylog-ctl` tool with the virtual machine appliances to get you started with a customised setup as quickly as possible. Run these (optional) commands to configure the most basic settings of Graylog in the appliance:

```
sudo graylog-ctl set-email-config <smtp server> [--port=<smtp port> --user=<username>␣
↪--password=<password>]
sudo graylog-ctl set-admin-password <password>
sudo graylog-ctl set-timezone <zone acronym>
sudo graylog-ctl reconfigure
```

The `graylog-ctl` has much more functionality and is documented *here*. We strongly recommend to learn more about it to ensure smooth operation of your virtual appliance.

Microsoft Windows

Unfortunately there is no officially 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. This will be improved in future versions.

Please note that this impacts Graylog's ability to buffer messages, so we strongly recommend running the Linux-based OVAs on Windows.

Manual Setup

Prerequisites

You will need to have the following services installed on either the host you are running `graylog-server` on or on dedicated machines:

- Elasticsearch 1.6.2 or later (Elasticsearch 2.x is currently not supported)
- MongoDB 2.0 or later (latest stable version is recommended)
- Oracle Java SE 7 or later (Oracle Java SE 8 is supported, OpenJDK 7 and OpenJDK 8 also work; latest stable update is recommended)

Most standard MongoDB packages of Linux distributions are outdated. Use the official MongoDB APT repository (available for many distributions and operating systems)

You also **must** install **Java 7** or higher! Java 6 is not compatible with Graylog and will also not receive any more publicly available bug and security fixes by Oracle.

A more detailed guide for installing the dependencies will follow. **The only important thing for Elasticsearch is that you set the exactly same cluster name (e. g. ‘‘cluster.name: graylog‘‘) that is being used by Graylog in the Elasticsearch configuration (‘‘conf/elasticsearch.yml‘‘).**

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 yourpassword | shasum -a 256` and you will be able to log in to the web interface with username *admin* and password *yourpassword*.

- **elasticsearch_max_docs_per_index = 20000000**
  - How many log messages to keep per index. This setting multiplied with `elasticsearch_max_number_of_indices` results in the maximum number of messages in your Graylog setup. It is always better to have several more smaller indices than just a few larger ones.
- **elasticsearch_max_number_of_indices = 20**
  - How many indices to have in total. If this number is reached, the oldest index will be deleted. **Also take a look at the other retention strategies that allow you to automatically delete messages based on their age.**
- **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`.
- **mongodb_***
  - Enter your MongoDB connection and authentication information here. Make sure that you connect the web interface to the same database. You don't need to configure `mongodb_user` and `mongodb_password` if `mongodb_useauth` is set to `false`.

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-7-jre
```

You need at least Java 7 as Java 6 has reached EOL.

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-server` 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 `graylog-server` logs in the directory `logs/`.

Important: All `graylog-server` instances must have synchronised time. We strongly recommend to use NTP or similar mechanisms on all machines of your Graylog infrastructure.

Supplying external logging configuration

The `graylog-server` uses Log4j for its internal logging and ships with a default log configuration file which is embedded within the shipped JAR.

In case you need to overwrite the configuration `graylog-server` uses, you can supply a Java system property specifying the path to the configuration file in your `graylogctl` script. Append this before the *-jar* paramter:

```
-Dlog4j.configuration=file:///tmp/logj4.xml
```

Substitute the actual path to the file for the `/tmp/log4j.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 its logs to grow without bounds.

One such example `log4j.xml` configuration is shown below. Graylog includes the `log4j-extras` companion classes to support time based and size based log rotation. This is the example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration PUBLIC "-//APACHE//DTD LOG4J 1.2//EN" "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

    <appender name="FILE" class="org.apache.log4j.rolling.RollingFileAppender">
        <rollingPolicy class="org.apache.log4j.rolling.FixedWindowRollingPolicy" >
            <param name="activeFileName" value="/tmp/server.log" /> <!-- ADAPT -->
            <param name="fileNamePattern" value="/tmp/server.%i.log" /> <!-- ADAPT -->
            <param name="minIndex" value="1" /> <!-- ADAPT -->
            <param name="maxIndex" value="10" /> <!-- ADAPT -->
        </rollingPolicy>
        <triggeringPolicy class="org.apache.log4j.rolling.SizeBasedTriggeringPolicy">
            <param name="maxFileSize" value="5767168" /> <!-- ADAPT: For example 5.
↪5MB in bytes -->
        </triggeringPolicy>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d %-5p: %c - %m%n"/>
        </layout>
    </appender>

    <!-- Application Loggers -->
    <logger name="org.graylog2">
        <level value="info"/>
    </logger>
    <!-- this emits a harmless warning for ActiveDirectory every time which we can't␣
↪work around :( -->
    <logger name="org.apache.directory.api.ldap.model.message.BindRequestImpl">
        <level value="error"/>
    </logger>
    <!-- Root Logger -->
    <root>
        <priority value="info"/>
        <appender-ref ref="FILE"/>
    </root>

</log4j: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`
- `-t`, `--configtest`: Validate the Graylog configuration and exit with exit code 0 if the configuration file is syntactically correct, exit code 1 and a description of the error otherwise
- `-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.
- `-r`, `--no-retention`: Do not automatically delete old/outdated indices
- `-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-server* instance refuses to listen on IPv4 addresses and always chooses for example a *rest_listen_address* like *:::12900* 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
```

Manual setup: graylog-web-interface on Linux

Prerequisites

The only thing you need is at least one compatible `graylog-server` node. Please use the same version number to make sure that it is compatible.

You also **must** use **Java 7**! Java 6 is not compatible with Graylog and will also not receive any more publicly available bug and security fixes by Oracle.

Downloading and extracting the web-interface

Download the package from the download pages.

Extract the archive:

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

Configuring the web interface

Open `conf/graylog-web-interface.conf` and set the two following variables:

- `graylog2-server.uris="http://127.0.0.1:12900/"`: This is the list of `graylog-server` nodes the web interface will try to use. You can configure one or multiple, separated by commas. Use the `rest_listen_uri` (configured in `graylog.conf`) of your `graylog-server` instances here.
- `application.secret=""`: A secret for encryption. Use a long, randomly generated string here. (for example generated using `pwgen -N 1 -s 96`)

Starting the web interface

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-7-jre
```

You need at least Java 7 as Java 6 has reached EOL.

Now start the web interface:

```
~$ bin/graylog-web-interface
Play server process ID is 5723
[info] play - Application started (Prod)
[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000
```

The web interface will listen on port 9000. You should see a login screen right away after pointing your browser to it. Log in with username `admin` and the password you configured at `root_password_sha2` in the `graylog.conf` of your `graylog-server`.

Changing the listen port and address works like this:

```
~$ bin/graylog-web-interface -Dhttp.port=1234 -Dhttp.address=127.0.0.1
```

Java generally prefers to bind to an IPv6 address if that is supported by your system, while you might want to prefer IPv4. To change Java's default preference you can pass `-Djava.net.preferIPv4Stack=true` to the startup script:

```
~$ bin/graylog-web-interface -Djava.net.preferIPv4Stack=true
```

All those `-D` settings can also be added to the `JAVA_OPTS` environment variable which is being read by the startup script, too.

You can start the web interface in background for example like this:

```
~$ nohup bin/graylog-web-interface &
```

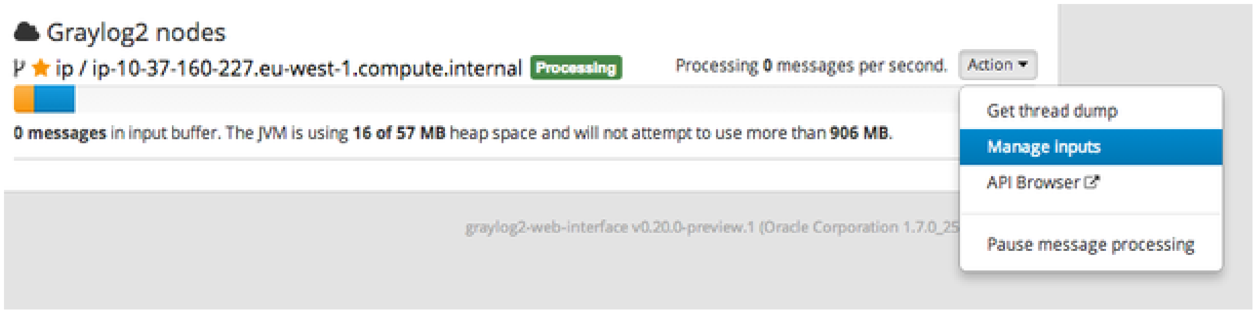
Custom configuration file path

You can put the configuration file into another directory like this:

~$ bin/graylog-web-interface -Dconfig.file=/etc/graylog-web-interface.conf

Create a message input and send a first message

Log in to the web interface and navigate to *System* -> *Nodes*. Select your `graylog-server` node there and click on *Manage inputs*.

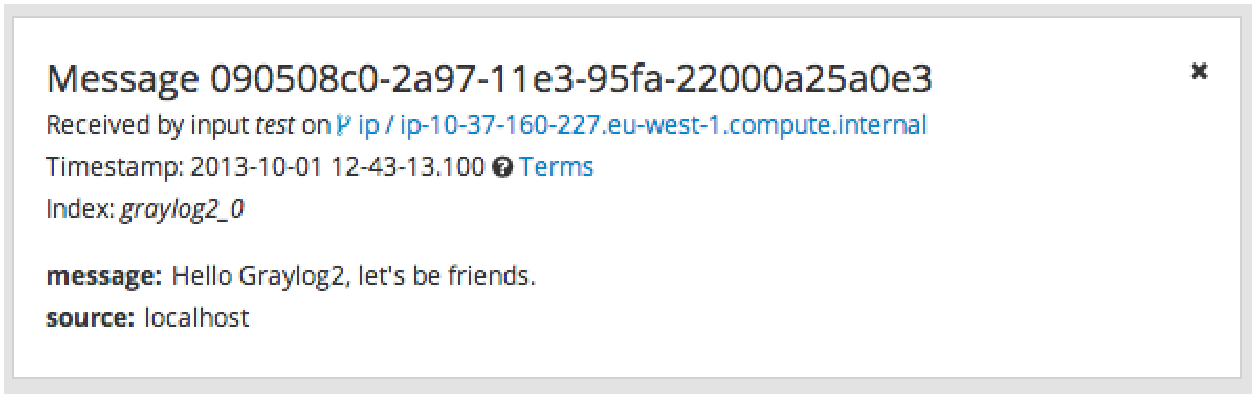

Graylog2 nodes
ip / ip-10-37-160-227.eu-west-1.compute.internal
Processing
Processing 0 messages per second.
Action
Get thread dump
Manage Inputs
API Browser
Pause message processing
graylog2-web-interface v0.20.0-preview.1 (Oracle Corporation 1.7.0_25

0 messages in input buffer. The JVM is using 16 of 57 MB heap space and will not attempt to use more than 906 MB.

Launch a new *Raw/Plaintext UDP* input, listening on port `9099` and listening on `127.0.0.1`. 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 searchbar 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:



Message 090508c0-2a97-11e3-95fa-22000a25a0e3
Received by input *test* on ip / ip-10-37-160-227.eu-west-1.compute.internal
Timestamp: 2013-10-01 12-43-13.100 Terms
Index: *graylog2_0*

message: Hello Graylog2, let's be friends.
source: localhost

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.

HTTPS

Enabling HTTPS is easy. Just start the web interface like this:

```
bin/graylog-web-interface -Dhttps.port=443
```

This will generate self-signed certificate. To use proper certificates you must configure a Java key store. Most signing authorities provide instructions on how to create a Java keystore and the official keystore utility docs can be found here.

- `https.keyStore` The path to the keystore containing the private key and certificate, if not provided generates a keystore for you
- `https.keyStoreType` The key store type, defaults to JKS
- `https.keyStorePassword` The password, defaults to a blank password
- `https.keyStoreAlgorithm` The key store algorithm, defaults to the platforms default algorithm

To disable HTTP without SSL completely and enforce HTTPS, use this parameter:

```
-Dhttp.port=disabled
```

Configuring logging

The default setting of the web interface is to write its own logs to `STDOUT`. You can take control of the logging by specifying an own Logback configuration file to use:

```
bin/graylog-web-interface -Dlogger.file=/etc/graylog-web-interface-log.xml
```

This is an example Logback configuration file that has a disabled `STDOUT` appender and an enabled appender that writes to a file (`/var/log/graylog/web/graylog-web-interface.log`), keeps 30 days of logs in total and creates a new log file if a file should have reached a size of 100MB:

```
<configuration>

    <!--
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%date %-5level [%thread] - [%logger]- %msg%n</pattern>
        </encoder>
    </appender>
    -->

    <appender name="ROLLING_FILE" class="ch.qos.logback.core.rolling.
↪RollingFileAppender">
        <file>/var/log/graylog/web/graylog-web-interface.log</file>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <FileNamePattern>/var/log/graylog/web/graylog-web-interface.log.%d{yyyy-
↪MM-dd}.%i.log.gz</FileNamePattern>
            <MaxHistory>30</MaxHistory>
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.
↪rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>100MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
        </rollingPolicy>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%date [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="INFO">
        <!--<appender-ref ref="STDOUT" />-->
        <appender-ref ref="ROLLING_FILE" />
    </root>

</configuration>
```

CHAPTER 3

Configuring and tuning 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.

Configuration

Configuration of graylog-server nodes

The most important settings to make a successful connection are the Elasticsearch cluster name and the discovery mode. Graylog is able to discover the Elasticsearch nodes using multicast. This is great for development and proof of concepts but we recommend to use classic unicast discovery in production.

Cluster Name

You need to tell `graylog-server` which Elasticsearch cluster to join. The Elasticsearch cluster default name is *elasticsearch* and configured for every Elasticsearch node in its `elasticsearch.yml` configuration file with the setting `cluster.name`. Configure the same name in every `graylog.conf` as `elasticsearch_cluster_name`. We recommend to call the cluster `graylog-production` and not `elasticsearch`.

The `elasticsearch.yml` file is typically located in `/etc/elasticsearch/`.

Discovery mode

The default discovery mode is multicast. Graylog will try to find other Elasticsearch nodes automatically. This usually works fine when everything is running on the same system but gets problematic quickly when running in a bigger network topology. We recommend to use unicast for production setups. Configure Zen unicast discovery in Graylog with the following lines in your configuration file:

```
# Disable multicast
elasticsearch_discovery_zen_ping_multicast_enabled = false
# List of Elasticsearch nodes to connect to
elasticsearch_discovery_zen_ping_unicast_hosts = es-node-1.example.org:9300,es-node-2.
↪example.org:9300
```

Also make sure to configure Zen unicast discovery in the Elasticsearch configuration file by adding the `discovery.zen.ping.multicast.enabled` and `discovery.zen.ping.unicast.hosts` setting with the list of Elasticsearch nodes to `elasticsearch.yml`:

```
discovery.zen.ping.multicast.enabled: false
discovery.zen.ping.unicast.hosts: ["es-node-1.example.org:9300" , "es-node-2.example.
↪org:9300"]
```

The Elasticsearch default communication port is *9300/tcp* (not to be confused with the HTTP interface running on port *9200/tcp* by default). The communication port can be changed in the Elasticsearch configuration file (`elasticsearch.yml`) with the configuration setting `transport.tcp.port`. Make sure that Elasticsearch binds to a network interface that Graylog can connect to (see `network.host`).

Configuration of Elasticsearch nodes

Disable dynamic scripting

Elasticsearch prior to version 1.2 had an insecure default configuration which could lead to a remote code execution. (see here and here for details)

Make sure to add `script.disable_dynamic: true` to the `elasticsearch.yml` file to disable the dynamic scripting feature and prevent possible remote code executions.

Control access to Elasticsearch ports

Since Elasticsearch has no authentication mechanism at time of this writing, 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 Elasticsearch documentation page.

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.

Merge throttling

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 per index it creates. This is enough tuning for a lot of use cases and setups. A more detailed guide on deeper tuning of Elasticsearch is following.

Cluster Status explained

Elasticsearch provides a classification for the cluster health:

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 are restored.

YELLOW

The yellow status means that all of the primary shards are available but some or all shard replicas are not.

With only one Elasticsearch node, the cluster state cannot become green because shard replicas cannot be assigned. This can be solved by adding another Elasticsearch node to the cluster.

If the cluster is supposed to have only one node it is okay to be in the yellow state.

GREEN

The cluster is fully operational. All primary and replica shards are available.

CHAPTER 4

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.

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.

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 here 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.

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.

Sending syslog from Linux hosts

rsyslog

Forwarding syslog messages with rsyslog is easy. The only important thing to get the most out of your logs is following RFC 5424. The following examples configures your `rsyslog` daemon to send RFC 5424 date to Graylog syslog inputs:

UDP:

```
$template GRAYLOGRFC5424,"<%PRI%>%PROTOCOL-VERSION% %TIMESTAMP:::date-rfc3339%
↪%HOSTNAME% %APP-NAME% %PROCID% %MSGID% %STRUCTURED-DATA% %msg%\n"
*.* @graylog.example.org:514;GRAYLOGRFC5424
```

TCP:

```
$template GRAYLOGRFC5424,"<%PRI%>%PROTOCOL-VERSION% %TIMESTAMP:::date-rfc3339%
↪%HOSTNAME% %APP-NAME% %PROCID% %MSGID% %STRUCTURED-DATA% %msg%\n"
*.* @@graylog.example.org:514;GRAYLOGRFC5424
```

(The difference between UDP and TCP is using `@` instead of `@@` as target descriptor.)

Alternatively, the rsyslog built-in template RSYSLOG_SyslogProtocol23Format sends log messages in the same format as above. This exists in rsyslog versions of at least 5.10 or later.

The UDP examples above becomes:

```
*.* @graylog.example.org:514;RSYSLOG_SyslogProtocol23Format
```

syslog-ng

Configuring syslog-ng to send syslog to Graylog is equally simple. Use the `syslog` function to send RFC 5424 formatted syslog messages via TCP to the remote Graylog host:

```
# Define TCP syslog destination.
destination d_net {
    syslog("graylog.example.org" port(514));
};
# Tell syslog-ng to send data from source s_src to the newly defined syslog␣
↪destination.
log {
    source(s_src); # Defined in the default syslog-ng configuration.
    destination(d_net);
};
```

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 behaviour 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.

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 on graylog.org.

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.

Microsoft Windows

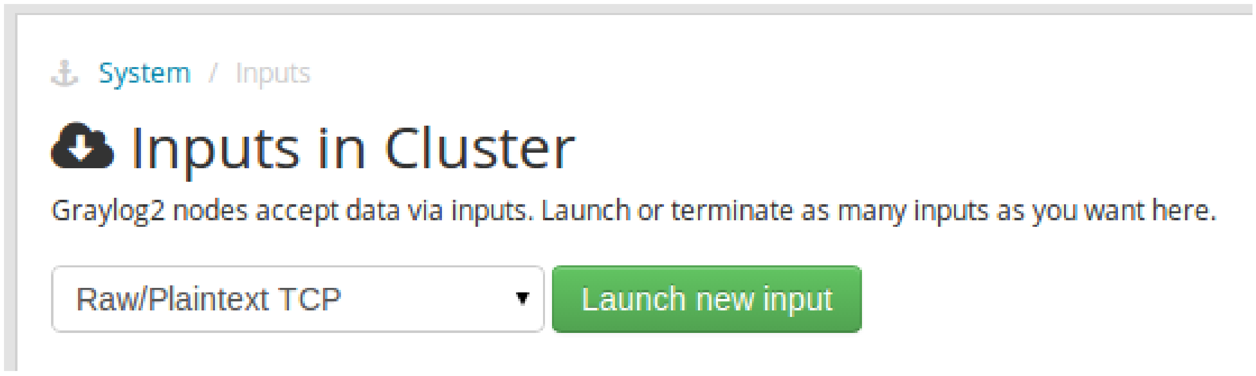
Our recommended way to forward Windows log data (for example EventLog) to Graylog is to use our own *log collector*. It comes with native support for reading Windows event logs.

Heroku

Heroku allows you to forward the logs of your application to a custom syslog server by creating a so called Syslog drain. The drain sends all logs to the configured server(s) via TCP. Following example shows you how to configure Graylog to receive the Heroku logs and extract the different fields into a structured log message.

Creating a Graylog input for Heroku log messages

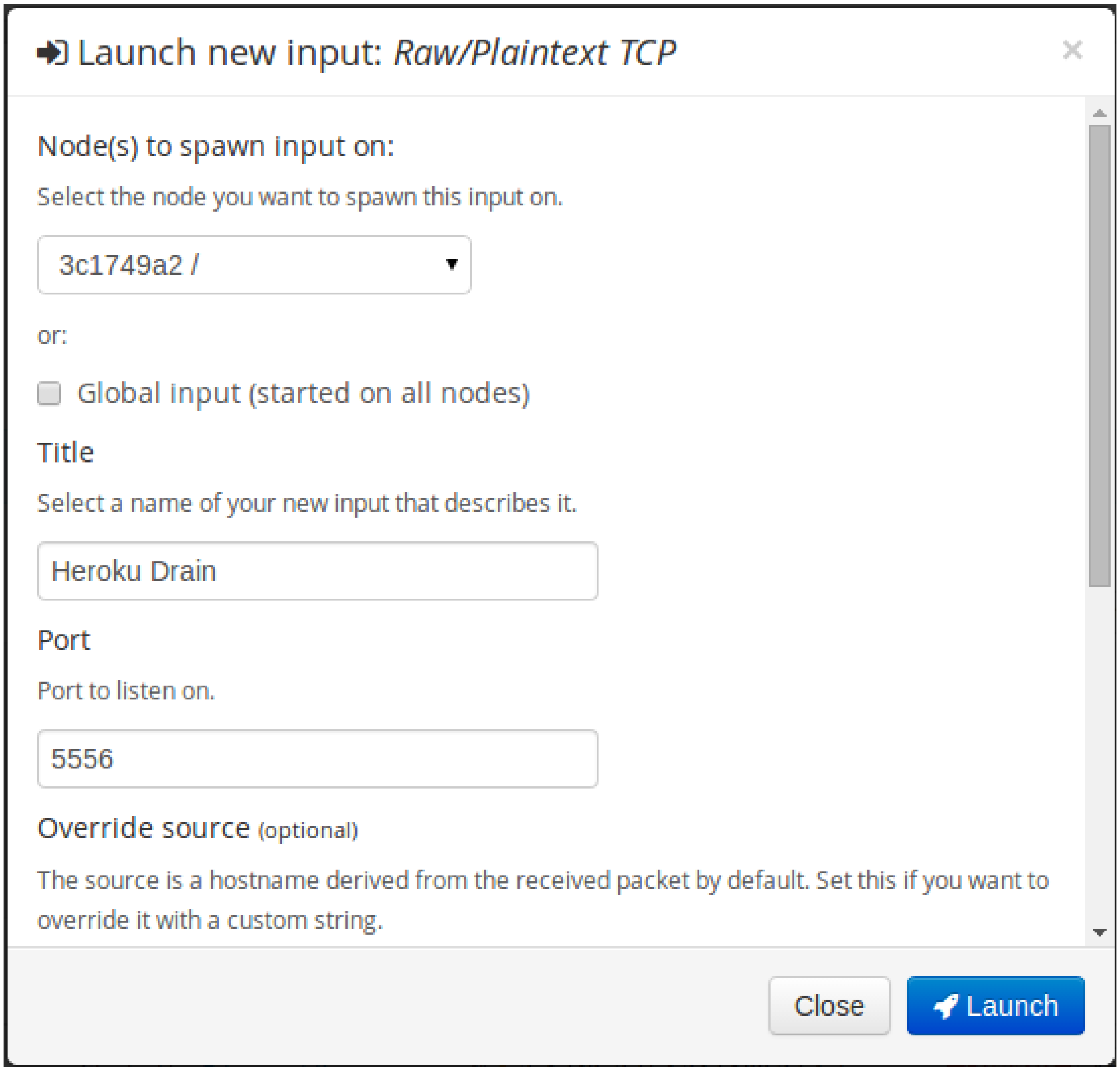
Create a new **RAW/Plaintext TCP** input as shown below.



System / Inputs

Inputs in Cluster

Graylog2 nodes accept data via inputs. Launch or terminate as many inputs as you want here.



Launch new input: *Raw/Plaintext TCP*

Node(s) to spawn input on:

Select the node you want to spawn this input on.

3c1749a2 /

or:

Global input (started on all nodes)

Title

Select a name of your new input that describes it.

Heroku Drain

Port

Port to listen on.

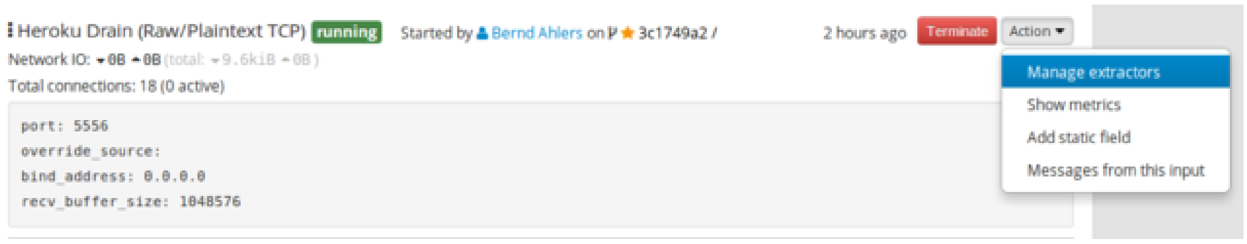
5556

Override source (optional)

The source is a hostname derived from the received packet by default. Set this if you want to override it with a custom string.

The Graylog Extractor library contains a set of extractors to parse the Heroku log format. You can import that set into the newly created input so all parts of the log messages will be extracted into separate fields:

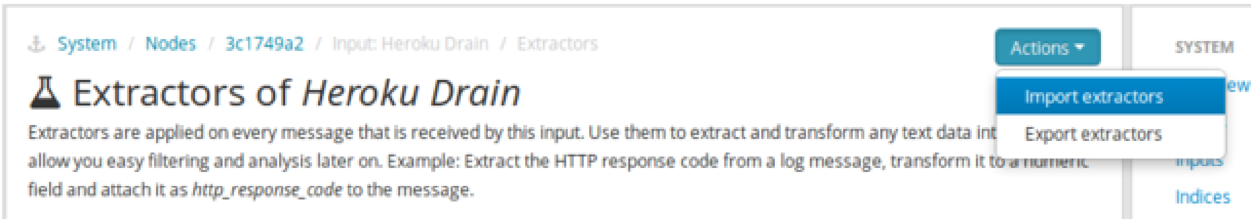
Open the extractor management for the input.



Network IO: 0B 0B (total: 9.6kiB 0B)
Total connections: 18 (0 active)

```
port: 5556
override_source:
bind_address: 0.0.0.0
recv_buffer_size: 1048576
```

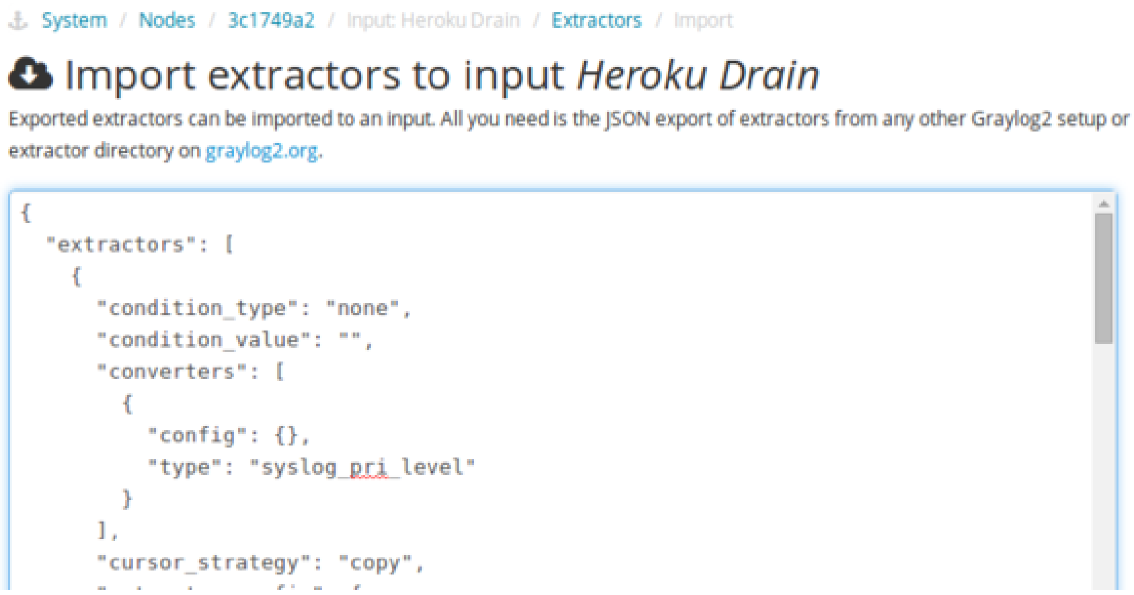
Go to the extractor import.


System / Nodes / 3c1749a2 / Input: Heroku Drain / Extractors
Actions
Import extractors
Export extractors
SYSTEM
Inputs
Indices

Extractors of *Heroku Drain*

Extractors are applied on every message that is received by this input. Use them to extract and transform any text data int allow you easy filtering and analysis later on. Example: Extract the HTTP response code from a log message, transform it to a numeric field and attach it as *http_response_code* to the message.

Paste the extractor JSON string into the form and submit.

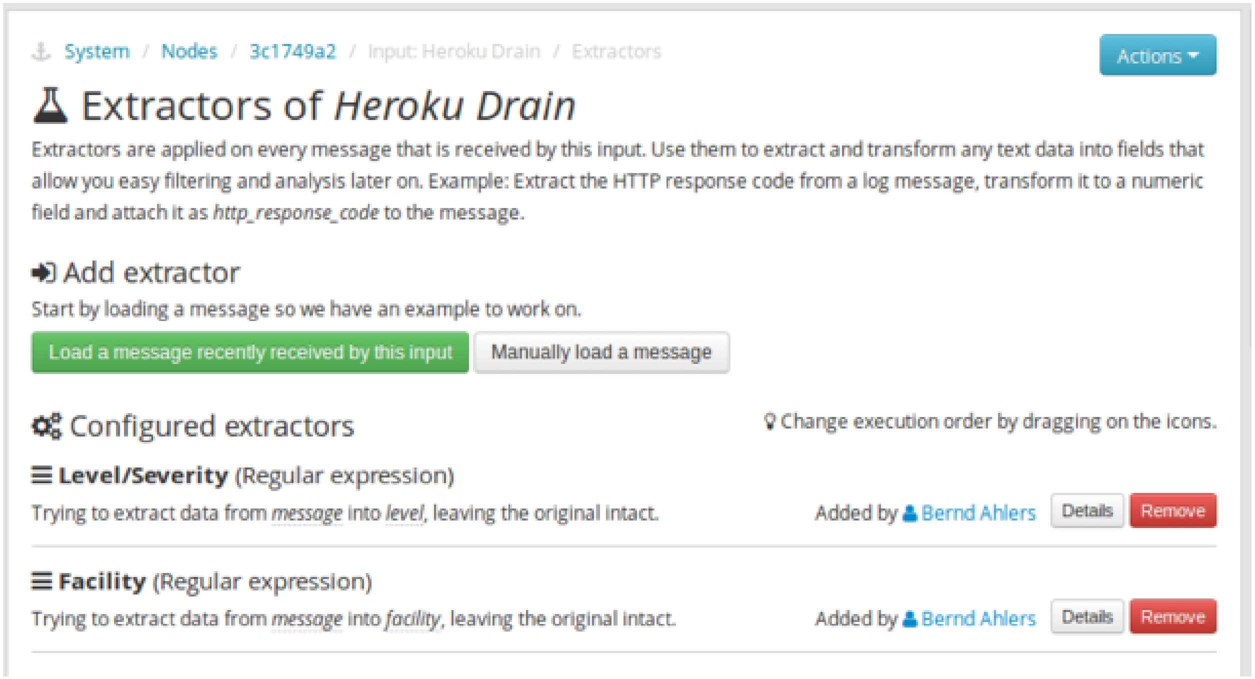

System / Nodes / 3c1749a2 / Input: Heroku Drain / Extractors / Import

Import extractors to input *Heroku Drain*

Exported extractors can be imported to an input. All you need is the JSON export of extractors from any other Graylog2 setup or from the extractor directory on graylog2.org.

```
{
  "extractors": [
    {
      "condition_type": "none",
      "condition_value": "",
      "converters": [
        {
          "config": {},
          "type": "syslog_pri_level"
        }
      ],
      "cursor_strategy": "copy",
```

That is all that is needed on the Graylog side. Make sure your firewall setup allows incoming connections on the inputs port!


System / Nodes / 3c1749a2 / Input: Heroku Drain / Extractors
Actions
Load a message recently received by this input
Manually load a message
Details
Remove

Extractors of *Heroku Drain*

Extractors are applied on every message that is received by this input. Use them to extract and transform any text data into fields that allow you easy filtering and analysis later on. Example: Extract the HTTP response code from a log message, transform it to a numeric field and attach it as *http_response_code* to the message.

Add extractor

Start by loading a message so we have an example to work on.

Configured extractors

Change execution order by dragging on the icons.

Level/Severity (Regular expression)

Trying to extract data from *message* into *level*, leaving the original intact. Added by Bernd Ahlers

Facility (Regular expression)

Trying to extract data from *message* into *facility*, leaving the original intact. Added by Bernd Ahlers

Configuring Heroku to send data to your Graylog setup

Heroku has a detailed documentation regarding the Syslog drains feature. The following example shows everything that is needed to setup the drain for you application:

```
$ cd path/to/your/heroku/app
$ heroku drains
No drains for this app
$ heroku drains:add syslog://graylog.example.com:5556
Successfully added drain syslog://graylog.example.com:5556
$ heroku drains
syslog://graylog.example.com:5556 (d.8cf52d32-7d79-4653-baad-8cb72bb23ee1)
```

The Heroku CLI tool needs to be installed for this to work.

You Heroku application logs should now show up in the search results of your Graylog instance.

Ruby on Rails

This is easy: You just need to combine a few components.

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.

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" })
```

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
```

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.

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.

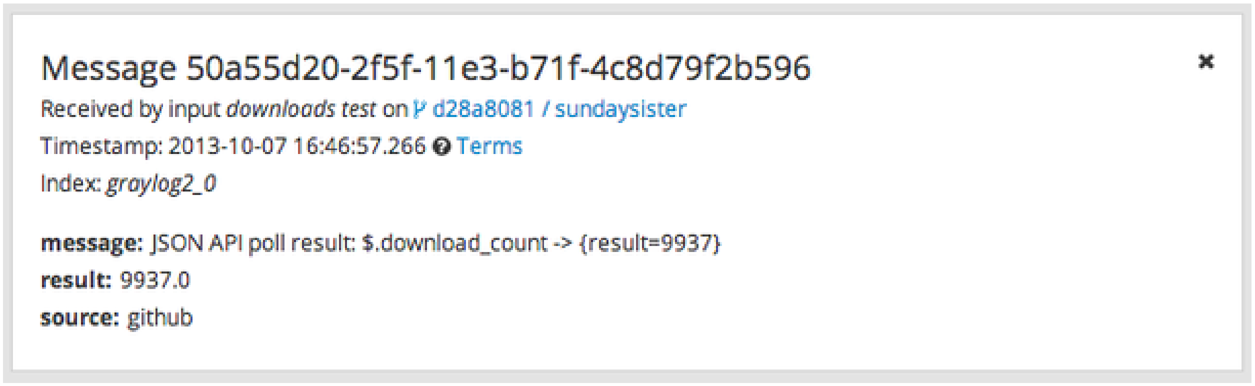
Example

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,
  "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:



Message 50a55d20-2f5f-11e3-b71f-4c8d79f2b596
Received by input *downloads test* on d28a8081 / sundaysister
Timestamp: 2013-10-07 16:46:57.266 Terms
Index: *graylog2_0*

message: JSON API poll result: $.download_count -> {result=9937}
result: 9937.0
source: github

You can use Graylog to analyse your download counts now.

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.

Reading from files

Graylog is currently not providing an out-of-the-box way to read log messages from files. We do however recommend two fantastic tools to do that job for you. Both come with native Graylog (GELF) outputs:

- fluentd
- logstash

CHAPTER 5

Graylog Collector

Graylog Collector is a lightweight Java application that allows you to forward data from log files to a Graylog cluster. The collector can read local log files and also Windows Events natively, it then can forward the log messages over the network using the GELF format.

Installation

Linux/Unix

You need to have Java >= 7 installed to run the collector.

Operating System Packages

We offer official package repositories for the following operating systems.

- Ubuntu 12.04, 14.04
- Debian 8
- CentOS 7

Please open an issue in the Github repository if you run into any packaging related issues. **Thank you!**

Ubuntu 14.04

Download and install graylog-collector-latest-repository-ubuntu14.04_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-collector-latest-
↪repository-ubuntu14.04_latest.deb
$ sudo dpkg -i graylog-collector-latest-repository-ubuntu14.04_latest.deb
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install graylog-collector
```

Ubuntu 12.04

Download and install graylog-collector-latest-repository-ubuntu12.04_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-collector-latest-
↪repository-ubuntu12.04_latest.deb
$ sudo dpkg -i graylog-collector-latest-repository-ubuntu12.04_latest.deb
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install graylog-collector
```

Debian 8

Download and install graylog-collector-latest-repository-debian8_latest.deb via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ wget https://packages.graylog2.org/repo/packages/graylog-collector-latest-
↪repository-debian8_latest.deb
$ sudo dpkg -i graylog-collector-latest-repository-debian8_latest.deb
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install graylog-collector
```

CentOS 7

Download and install graylog-collector-latest-repository-el7_latest.rpm via `rpm(8)`:

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-collector-latest-
↪repository-el7_latest.rpm
$ sudo yum install graylog-collector
```

Manual Setup

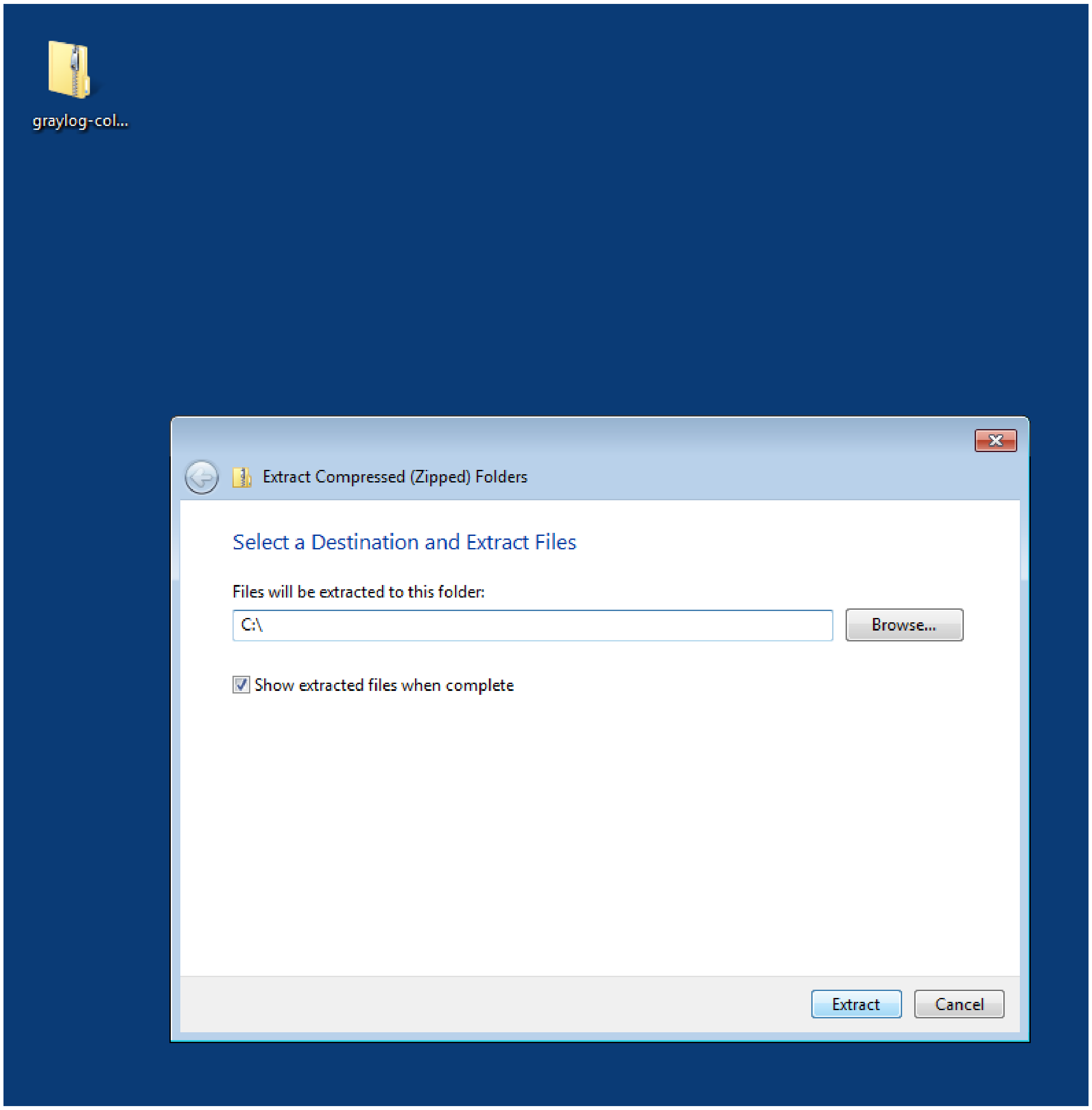
1. Download the latest collector release. (find download links in the collector repository README)
2. Unzip collector tgz file to target location
3. cp `config/collector.conf.example` to `config/collector.conf`
4. Update server-url in collector.conf to correct Graylog server address (required for registration)
5. Update file input configuration with the correct log files
6. Update outputs->gelf-tcp with the correct Graylog server address (required for sending GELF messages)

Note: The collector will not start properly if you do not set the URL or the correct input log files and GELF output configuration

Windows

You need to have Java >= 7 installed to run the collector.

Download a release zip file from the collector repository README. Unzip the collector zip file to target location.



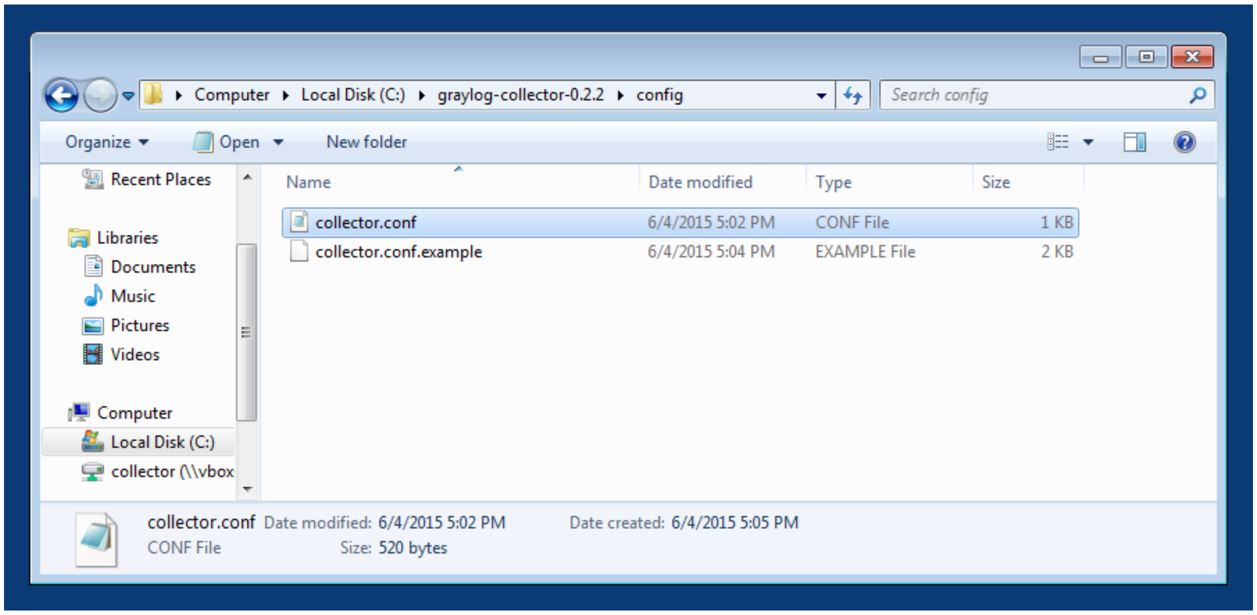
Select a Destination and Extract Files

Files will be extracted to this folder:

C:\

[✓] Show extracted files when complete

Change into the extracted collector directory and create a collector configuration file in `config\collector.conf`.



Name	Date modified	Type	Size
collector.conf	6/4/2015 5:02 PM	CONF File	1 KB
collector.conf.example	6/4/2015 5:04 PM	EXAMPLE File	2 KB

collector.conf Date modified: 6/4/2015 5:02 PM Date created: 6/4/2015 5:05 PM
CONF File Size: 520 bytes

The following configuration file shows a good starting point for Windows systems. It collects the *Application*, *Security*, and *System* event logs. Replace the `<your-graylog-server-ip>` with the IP address of your Graylog server.

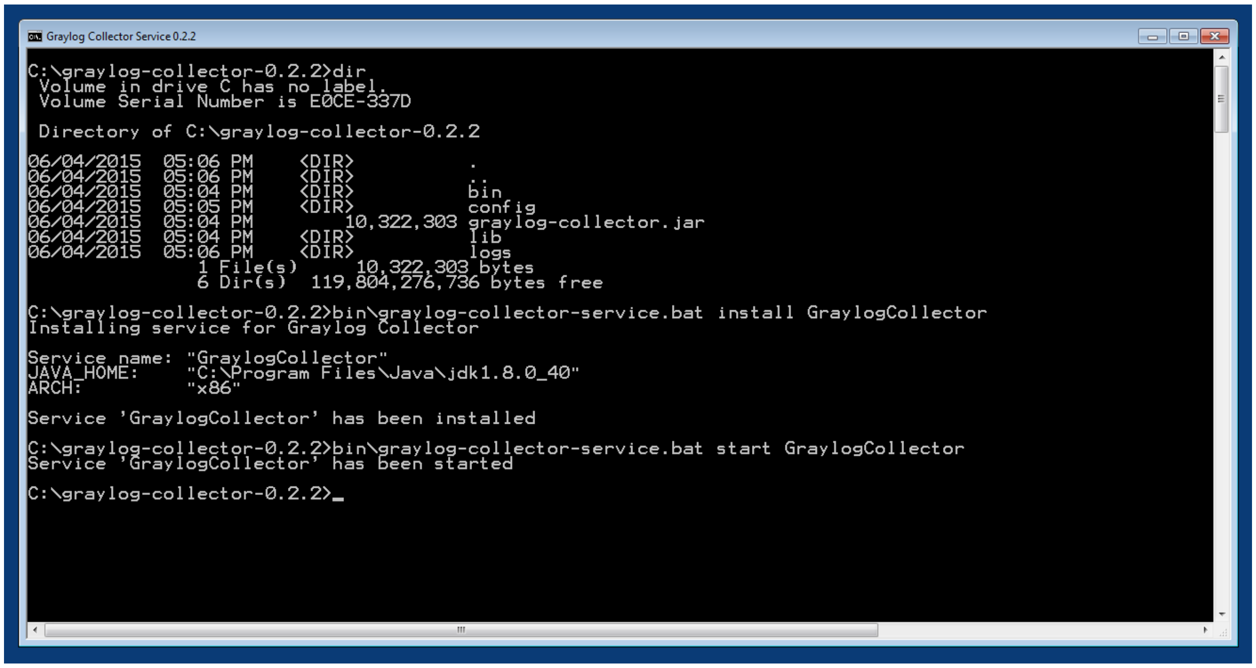
Example:

```
server-url = "http://<your-graylog-server-ip>:12900/"

inputs {
  win-eventlog-application {
    type = "windows-eventlog"
    source-name = "Application"
    poll-interval = "1s"
  }
  win-eventlog-system {
    type = "windows-eventlog"
    source-name = "System"
    poll-interval = "1s"
  }
  win-eventlog-security {
    type = "windows-eventlog"
    source-name = "Security"
    poll-interval = "1s"
  }
}

outputs {
  gelf-tcp {
    type = "gelf"
    host = "<your-graylog-server-ip>"
    port = 12201
  }
}
```

Start a `cmd.exe`, change to the collector installation path and execute the following commands to install the collector as Windows service.

Commands:

```
C:\> cd graylog-collector-0.2.2
C:\graylog-collector-0.2.2> bin\graylog-collector-service.bat install GraylogCollector
C:\graylog-collector-0.2.2> bin\graylog-collector-service.bat start GraylogCollector
```



```
C:\graylog-collector-0.2.2>dir
 Volume in drive C has no label.
 Volume Serial Number is E0CE-337D

 Directory of C:\graylog-collector-0.2.2

06/04/2015  05:06 PM    <DIR>          .
06/04/2015  05:06 PM    <DIR>          ..
06/04/2015  05:04 PM    <DIR>          bin
06/04/2015  05:05 PM    <DIR>          config
06/04/2015  05:04 PM        10,322,303 graylog-collector.jar
06/04/2015  05:04 PM    <DIR>          lib
06/04/2015  05:06 PM    <DIR>          logs
               1 File(s)     10,322,303 bytes
               6 Dir(s)  119,804,276,736 bytes free

C:\graylog-collector-0.2.2>bin\graylog-collector-service.bat install GraylogCollector
Installing service for Graylog Collector

Service name: "GraylogCollector"
JAVA_HOME:    "C:\Program Files\Java\jdk1.8.0_40"
ARCH:         "x86"

Service 'GraylogCollector' has been installed

C:\graylog-collector-0.2.2>bin\graylog-collector-service.bat start GraylogCollector
Service 'GraylogCollector' has been started

C:\graylog-collector-0.2.2>_
```

Configuration

You will need a configuration file before starting the collector. The configuration file is written in the HOCON format which is a human-optimized version of JSON.

If you choose the operating system installation method, the configuration file defaults to `/etc/graylog/collector/collector.conf`. For the manual installation method you have to pass the path to the configuration to the start script. (see *Running Graylog Collector*)

Here is a minimal configuration example that collects logs from the `/var/log/syslog` file and sends them to a Graylog server:

```
server-url = "http://10.0.0.1:12900/"

inputs {
  syslog {
    type = "file"
    path = "/var/log/syslog"
  }
}

outputs {
  graylog-server {
    type = "gelf"
    host = "10.0.0.1"
    port = 12201
  }
}
```

There are a few global settings available as well as several sections which configure different subsystems of the collector.

Global Settings

`server-url` - The API URL of the Graylog server Used to send a heartbeat to the Graylog server.

(default: `"http://localhost:12900"`)

`enable-registration` - Enable heartbeat registration Enables the heartbeat registration with the Graylog server. The collector will not contact the Graylog server API for heartbeat registration if this is set to `false`.

(default: `true`)

`collector-id` - Unique collector ID setting The ID used to identify this collector. Can be either a string which is used as ID, or the location of a file if prefixed with `file:`. If the file does not exist, an ID will be generated and written to that file. If it exists, it is expected to contain a single string without spaces which will be used for the ID.

(default: `"file:config/collector-id"`)

Input Settings

The input settings need to be nested in a `input { }` block. Each input has an ID and a type:

```
inputs {
  syslog {          // => The input ID
    type = "file"   // => The input type
    ...
  }
}
```

An input ID needs to be unique among all configured inputs. If there are two inputs with the same ID, the last one wins.

The following input types are available.

File Input

The file input follows files in the file system and reads log data from them.

`type` This needs to be set to `"file"`.

`path` The path to a file that should be followed.

Please make sure to escape the `\` character in Windows paths: `path = "C:\\Program Files\\Apache2\\logs\\www.example.com.access.log"`

(default: none)

`path-glob-root` The globbing root directory that should be monitored. See below for an explanation on globbing.

Please make sure to escape the `\` character in Windows paths: `path = "C:\\Program Files\\Apache2\\logs\\www.example.com.access.log"`

(default: none)

`path-glob-pattern` The globbing patttern. See below for an explanation on globbing.

(default: none)

content-splitter The content splitter implementation that should be used to detect the end of a log message.

Available content splitters: NEWLINE, PATTERN

See below for an explanation on content splitters.

(default: "NEWLINE")

content-splitter-pattern The pattern that should be used for the PATTERN content splitter.

(default: none)

charset Charset of the content in the configured file(s).

Can be one of the Supported Charsets of the JVM.

(default: "UTF-8")

reader-interval The interval in which the collector tries to read from every configured file. You might set this to a higher value like 1s if you have files which do not change very often to avoid unnecessary work.

(default: "100ms")

Globbing / Wildcards

You might want to configure the collector to read from lots of different files or files which have a different name each time they are rotated. (i.e. time/date in a filename) The file input supports this via the `path-glob-root` and `path-glob-pattern` settings.

A usual glob/wildcard string you know from other tools might be `/var/log/apache2/**/*.{access,error}.log`. This means you are interested in all log files which names end with `.access.log` or `.error.log` and which are in a sub directory of `/var/log/apache2`. Example: `/var/log/apache2/example.com/www.example.com.access.log`

For compatibility reasons you have to split this string into two parts. The root and the pattern.

Examples:

```
// /var/log/apache2/**/*.{access,error}.log
path-glob-root = "/var/log/apache2"
path-glob-pattern = "**/*.{access,error}.log"

// C:\Program Files\Apache2\logs\*.access.log
path-glob-root = "C:\\Program Files\\Apache2\\logs" // Make sure to escape the \
↪character in Windows paths!
path-glob-pattern = "*.access.log"
```

The file input will monitor the `path-glob-root` for new files and checks them against the `path-glob-pattern` to decide if they should be followed or not.

All available special characters for the glob pattern are documented in the Java docs for the getPathMatcher() method.

Content Splitter

One common problem when reading from plain text log files is to decide when a log message is complete. By default, the file input considers each line in a file to be a separate log message:

```
Jul 15 10:27:08 tumbler anacron[32426]: Job `cron.daily' terminated  # <-- Log
↪message 1
Jul 15 10:27:08 tumbler anacron[32426]: Normal exit (1 job run)      # <-- Log
↪message 2
```

But there are several cases where this is not correct. Java stack traces are a good example:

```
2015-07-10T11:16:34.486+01:00 WARN  [InputBufferImpl] Unable to process event␣
↪RawMessageEvent{raw=null, uuid=bde580a0-26ec-11e5-9a46-005056b26ca9,␣
↪encodedLength=350}, sequence 19847516
java.lang.NullPointerException
        at org.graylog2.shared.buffers.JournallingMessageHandler$Converter.
↪apply(JournallingMessageHandler.java:89)
        at org.graylog2.shared.buffers.JournallingMessageHandler$Converter.
↪apply(JournallingMessageHandler.java:72)
        at com.google.common.collect.Lists$TransformingRandomAccessList$1.
↪transform(Lists.java:617)
        at com.google.common.collect.TransformedIterator.next(TransformedIterator.
↪java:48)
        at java.util.AbstractCollection.toArray(AbstractCollection.java:141)
        at java.util.ArrayList.<init>(ArrayList.java:177)
        at com.google.common.collect.Lists.newArrayList(Lists.java:144)
        at org.graylog2.shared.buffers.JournallingMessageHandler.
↪onEvent(JournallingMessageHandler.java:61)
        at org.graylog2.shared.buffers.JournallingMessageHandler.
↪onEvent(JournallingMessageHandler.java:36)
        at com.lmax.disruptor.BatchEventProcessor.run(BatchEventProcessor.java:128)
        at com.codahale.metrics.InstrumentedExecutorService$InstrumentedRunnable.
↪run(InstrumentedExecutorService.java:176)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.
↪java:1142)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
↪java:617)
        at java.lang.Thread.run(Thread.java:745)
2015-07-10T11:18:18.000+01:00 WARN  [InputBufferImpl] Unable to process event␣
↪RawMessageEvent{raw=null, uuid=bde580a0-26ec-11e5-9a46-005056b26ca9,␣
↪encodedLength=350}, sequence 19847516
java.lang.NullPointerException
        ...
        ...
```

This should be one message but using a newline separator here will not work because it would generate one log message for each line.

To solve this problem, the file input can be configured to use a `PATTERN` content splitter. It creates separate log messages based on a regular expression instead of newline characters. A configuration for the stack trace example above could look like this:

```
inputs {
  graylog-server-logs {
    type = "file"
    path = "/var/log/graylog-server/server.log"
    content-splitter = "PATTERN"
    content-splitter-pattern = "^\\d{4}-\\d{2}-\\d{2}T" // Make sure to escape the \␣
↪character!
  }
}
```

This instructs the file input to split messages on a timestamp at the beginning of a line. So the first stack trace in the message above will be considered complete once a new timestamp is detected.

Windows Eventlog Input

The Windows eventlog input can read event logs from Windows systems.

`type` This needs to be set to `"windows-eventlog"`.

`source-name` The Windows event log system has several different sources from which events can be read.

Common source names: `Application`, `System`, `Security`

(default: `"Application"`)

`poll-interval` This controls how often the Windows event log should be polled for new events.

(default: `"1s"`)

Example:

```
inputs {
  win-eventlog-application {
    type = "windows-eventlog"
    source-name = "Application"
    poll-interval = "1s"
  }
}
```

Output Settings

The output settings need to be nested in a `output { }` block. Each output has an ID and a type:

```
outputs {
  graylog-server { // => The output ID
    type = "gelf"  // => The output type
    ...
  }
}
```

An output ID needs to be unique among all configured outputs. If there are two outputs with the same ID, the last one wins.

The following output types are available.

GELF Output

The GELF output sends log messages to a GELF TCP input on a Graylog server.

`type` This needs to be set to `"gelf"`.

`host` Hostname or IP address of the Graylog server.

(default: none)

`port` Port of the GELF TCP input on the Graylog server host.

(default: none)

`client-tls` Enables TLS for the connection to the GELF TCP input. Requires a TLS-enabled GELF TCP input on the Graylog server. (default: false)

`client-tls-cert-chain-file` Path to a TLS certificate chain file. If not set, the default certificate chain of the JVM will be used.

(default: none)

client-tls-verify-cert Verify the TLS certificate of the GELF TCP input on the Graylog server.

You might have to disable this if you are using a self-signed certificate for the GELF input and do not have any certificate chain file.

(default: `true`)

client-queue-size The GELF client library that is used for this output has an internal queue of messages. This option configures the size of this queue.

(default: `512`)

client-connect-timeout TCP connection timeout to the GELF input on the Graylog server.

(default: `5000`)

client-reconnect-delay The delay before the output tries to reconnect to the GELF input on the Graylog server.

(default: `1000`)

client-tcp-no-delay Sets the `TCP_NODELAY` option on the TCP socket that connects to the GELF input.

(default: `true`)

client-send-buffer-size Sets the TCP send buffer size for the connection to the GELF input.

It uses the JVM default for the operating system if set to `-1`.

(default: `-1`)

STDOUT Output

The STDOUT output prints the string representation of each message to STDOUT. This can be useful for debugging purposes but should be disabled in production.

type This needs to be set to `"stdout"`.

Static Message Fields

Sometimes it is useful to be able to add some static field to a message. This can help selecting extractors to run on the server, simplify stream routing and can make searching/filtering for those messages easier.

Every collector input can be configured with a `message-fields` option which takes key-value pairs. The key needs to be a string, the value can be a string or a number.

Example:

```
inputs {
  apache-logs {
    type = "file"
    path = "/var/log/apache2/access.log"
    message-fields = {
      "program" = "apache2"
      "priority" = 3
    }
  }
}
```

Each static message field will end up in the GELF message and shows up in the web interface as a separate field.

An input might overwrite a message field defined in the input configuration. For example the file input always sets a `source_file` field with the path to the file where the message has been read from. If you configure a `source_file` message field, it will be overwritten by the input.

Input/Output Routing

Every message that gets read by the configured inputs will be routed to every configured output. If you have two file inputs and two GELF outputs, every message will be received by both outputs. You might want to send some logs to only one output or have one output only accept logs from a certain input, though.

The collector provides two options for inputs and outputs which can be used to influence the message routing.

Inputs have a `outputs` option and outputs have a `inputs` option. Both take a comma separated list of input/output IDs.

Example:

```
inputs {
  apache-logs {
    type = "file"
    path-glob-root = "/var/log/apache2"
    path-glob-pattern = "*.{access,error}.log"
    outputs = "gelf-1,gelf-2"
  }
  auth-log {
    type = "file"
    path = "/var/log/auth.log"
  }
  syslog {
    type = "file"
    path = "/var/log/syslog"
  }
}

outputs {
  gelf-1 {
    type = "gelf"
    host = "10.0.0.1"
    port = 12201
  }
  gelf-2 {
    type = "gelf"
    host = "10.0.0.1"
    port = 12202
  }
  console {
    type = "stdout"
    inputs = "syslog"
  }
}
```

Routing for this config:

- `apache-logs` messages will only go to `gelf-1` and `gelf-2` outputs.
- `auth-log` messages will go to `gelf-1` and `gelf-2` outputs.
- `syslog` messages will go to all outputs.

- `console` output will only receive messages from `syslog` input.

inputs | outputs	gelf-1	gelf-2	console
apache-logs			
auth-log			
syslog			

This is pretty powerful but might get confusing when inputs and outputs have the routing fields. This is how it is implemented in pseudo-code:

```
var message = Object(message)
var output = Object(gelf-output)

if empty(output.inputs) AND empty(message.outputs)

  // No output routing configured, write the message to the output.
  output.write(message)

else if output.inputs.contains(message.inputId) OR message.outputs.contains(output.id)

  // Either the input that generated the message has the output ID in its "outputs"␣
↪field
  // or the output has the ID of the input that generated the message in its "inputs"␣
↪field.
  output.write(message)

end
```

Running Graylog Collector

You will need a configuration file before starting the collector. See the configuration documentation above for detailed instructions on how to configure it.

Linux/Unix

The start method for the collector depends on the installation method your choose.

Operating System Package

We ship startup scripts in our OS packages that use the startup method of the particular operating system.

OS	Init System	Example
Ubuntu	upstart	`sudo start graylog-collector`
Debian	systemd	`sudo systemctl start graylog-collector`
CentOS	systemd	`sudo systemctl start graylog-collector`

Manual Setup

If you use the manual setup, the location of the start script depends on where you extracted the collector.

Example:

```
$ bin/graylog-collector run -f config/collector.conf
```

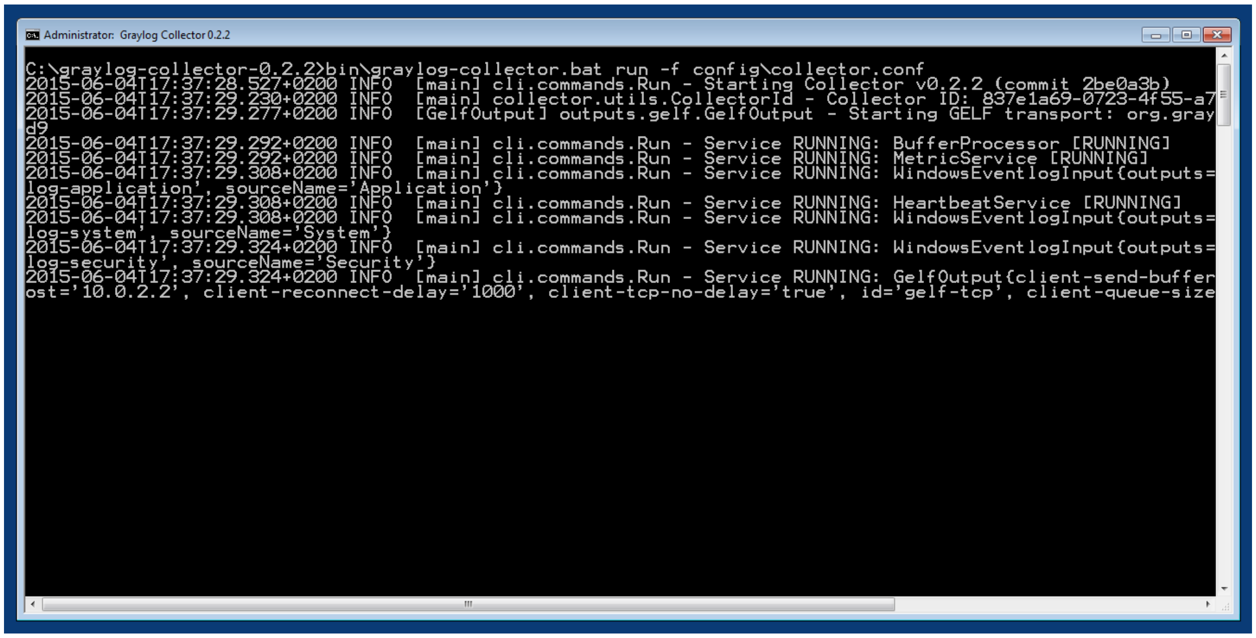
Windows

You probably want to run the collector as Windows service as described in the Windows installation section above. If you want to run it from the command line, run the following commands.

Make sure you have a valid configuration file in `config\collector.conf`.

Commands:

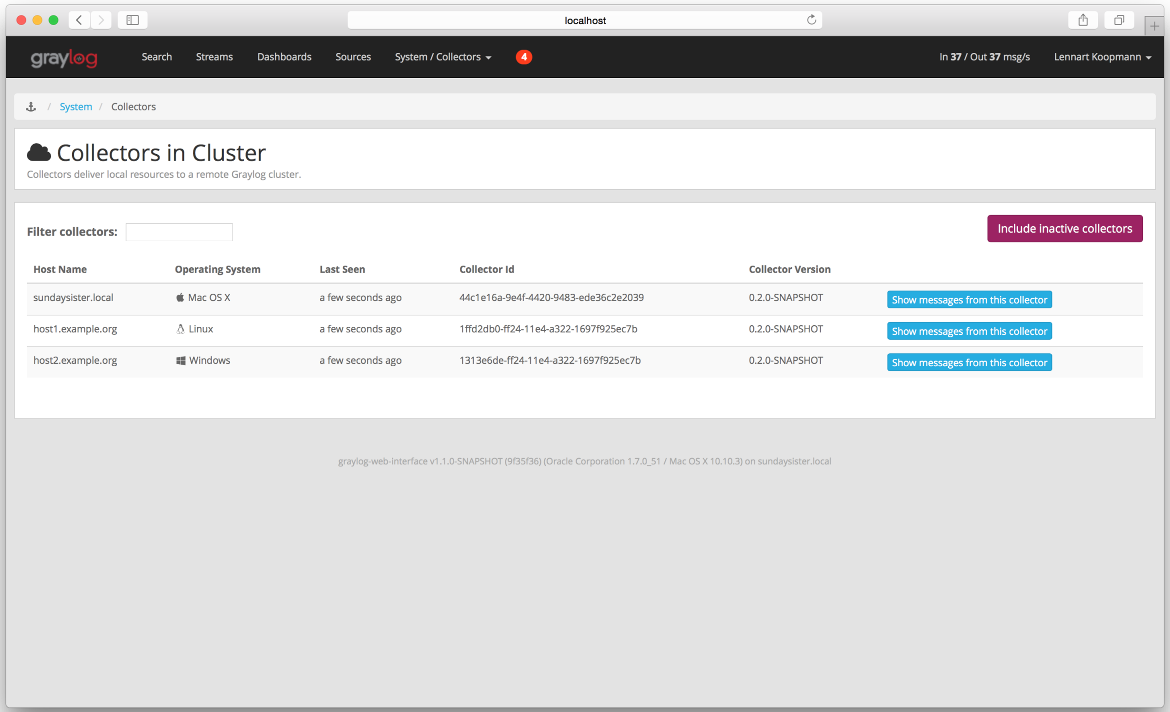
```
C:\> cd graylog-collector-0.2.2
C:\graylog-collector-0.2.2> bin\graylog-collector.bat run -f config\collector.conf
```



```
C:\graylog-collector-0.2.2>bin\graylog-collector.bat run -f config\collector.conf
2015-06-04T17:37:28.527+0200 INFO  [main] cli.commands.Run - Starting Collector v0.2.2 (commit 2be0a3b)
2015-06-04T17:37:29.230+0200 INFO  [main] collector.utils.CollectorId - Collector ID: 837e1a69-0723-4f55-a7
2015-06-04T17:37:29.277+0200 INFO  [GelfOutput] outputs.gelf.GelfOutput - Starting GELF transport: org.gray
d9
2015-06-04T17:37:29.292+0200 INFO  [main] cli.commands.Run - Service RUNNING: BufferProcessor [RUNNING]
2015-06-04T17:37:29.292+0200 INFO  [main] cli.commands.Run - Service RUNNING: MetricService [RUNNING]
2015-06-04T17:37:29.308+0200 INFO  [main] cli.commands.Run - Service RUNNING: WindowsEventlogInput{outputs=
log-application', sourceName='Application'}
2015-06-04T17:37:29.308+0200 INFO  [main] cli.commands.Run - Service RUNNING: HeartbeatService [RUNNING]
2015-06-04T17:37:29.308+0200 INFO  [main] cli.commands.Run - Service RUNNING: WindowsEventlogInput{outputs=
log-system', sourceName='System'}
2015-06-04T17:37:29.324+0200 INFO  [main] cli.commands.Run - Service RUNNING: WindowsEventlogInput{outputs=
log-security', sourceName='Security'}
2015-06-04T17:37:29.324+0200 INFO  [main] cli.commands.Run - Service RUNNING: GelfOutput{client-send-buffer
ost='10.0.2.2', client-reconnect-delay='1000', client-tcp-no-delay='true', id='gelf-tcp', client-queue-size
```

Collector Status

Once the collector has been deployed successfully, you can check on the status from the Graylog UI.



Collectors in Cluster

Collectors deliver local resources to a remote Graylog cluster.

Filter collectors:

Host Name	Operating System	Last Seen	Collector Id	Collector Version	
sundaysister.local	Mac OS X	a few seconds ago	44c1e16a-9e4f-4420-9483-ede36c2e2039	0.2.0-SNAPSHOT	Show messages from this collector
host1.example.org	Linux	a few seconds ago	1ffd2db0-ff24-11e4-a322-1697f925ec7b	0.2.0-SNAPSHOT	Show messages from this collector
host2.example.org	Windows	a few seconds ago	1313e6de-ff24-11e4-a322-1697f925ec7b	0.2.0-SNAPSHOT	Show messages from this collector

graylog-web-interface v1.1.0-SNAPSHOT (9f35f36) (Oracle Corporation 1.7.0_51 / Mac OS X 10.10.3) on sundaysister.local

You can reach the collector status overview page this way:

1. Log into Graylog Web Interface
2. Navigate to System / Collectors
3. Click Collectors

Troubleshooting

Check the standard output of the collector process for any error messages or warnings. Messages not arriving in your Graylog cluster? Check possible firewalls and the network connection.

Command Line Options

Linux/Unix

The collector offers the following command line options:

```
usage: graylog-collector <command> [<args>]

The most commonly used graylog-collector commands are:

    help       Display help information

    run        Start the collector

    version    Show version information on STDOUT
```

```
See 'graylog-collector help <command>' for more information on a specific command.

NAME
       graylog-collector run - Start the collector

SYNOPSIS
       graylog-collector run -f <configFile>

OPTIONS
       -f <configFile>
           Path to configuration file.
```

Correctly Configured Collector Log Sample

This is the *STDOUT* output of a healthy collector starting:

```
2015-05-12T16:00:10.841+0200 INFO  [main] o.graylog.collector.cli.commands.Run -␣
↪Starting Collector v0.2.0-SNAPSHOT (commit a2ad8c8)
2015-05-12T16:00:11.489+0200 INFO  [main] o.g.collector.utils.CollectorId - Collector␣
↪ID: cf4734f7-01d6-4974-a957-cb71bbd826b7
2015-05-12T16:00:11.505+0200 INFO  [GelfOutput] o.g.c.outputs.gelf.GelfOutput -␣
↪Starting GELF transport: org.graylog2.gelfclient.GelfConfiguration@3952e37e
2015-05-12T16:00:11.512+0200 INFO  [main] o.graylog.collector.cli.commands.Run -␣
↪Service RUNNING: BufferProcessor [RUNNING]
2015-05-12T16:00:11.513+0200 INFO  [main] o.graylog.collector.cli.commands.Run -␣
↪Service RUNNING: MetricService [RUNNING]
2015-05-12T16:00:11.515+0200 INFO  [main] o.graylog.collector.cli.commands.Run -␣
↪Service RUNNING: FileInput{id='local-syslog', path='/var/log/syslog', charset='UTF-8
↪', outputs='', content-splitter='NEWLINE'}
2015-05-12T16:00:11.516+0200 INFO  [main] o.graylog.collector.cli.commands.Run -␣
↪Service RUNNING: GelfOutput{port='12201', id='gelf-tcp', client-send-buffer-size=
↪'32768', host='127.0.0.1', inputs='', client-reconnect-delay='1000', client-connect-
↪timeout='5000', client-tcp-no-delay='true', client-queue-size='512'}
2015-05-12T16:00:11.516+0200 INFO  [main] o.graylog.collector.cli.commands.Run -␣
↪Service RUNNING: HeartbeatService [RUNNING]
2015-05-12T16:00:11.516+0200 INFO  [main] o.graylog.collector.cli.commands.Run -␣
↪Service RUNNING: StdoutOutput{id='console', inputs=''}
```

Troubleshooting

Unable to send heartbeat

The collector registers with your Graylog server on a regular basis to make sure it shows up on the Collectors page in the Graylog web interface. This registration can fail if the collector cannot connect to the server via HTTP on port `12900`:

```
2015-06-06T10:45:14.964+0200 WARN  [HeartbeatService RUNNING] collector.heartbeat.
↪HeartbeatService - Unable to send heartbeat to Graylog server: ConnectException:␣
↪Connection refused
```

Possible solutions

- Make sure the server REST API is configured to listen on a reachable IP address. Change the "rest_listen_uri" setting in the Graylog server config to this: `rest_listen_uri = http://0.0.0.0:12900/`
- Correctly configure any firewalls between the collector and the server to allow HTTP traffic to port `12900`.

CHAPTER 6

Search query language

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 login)
```

Messages where the field *type* includes the exact phrase *ssh login*:

```
type:"ssh login"
```

Messages that do not have the field *type*:

```
_missing_:type
```

Messages that have the field *type*:

```
_exists_:type
```

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-server.conf`: `allow_leading_wildcard_searches = true`

Also note that *message*, *full_message*, and *source* are the only fields that can be searched via wildcard by default.

Fuzziness: You can search for similar but not equal terms:

```
ssh logni~
source:exmaple.org~
```

This is using the Damerau–Levenshtein distance with a default distance of *2*. 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)
```

Escaping

The following characters must be escaped with a backslash:

```
&& || : \ / + - ! ( ) { } [ ] ^ " ~ * ?
```

Example:

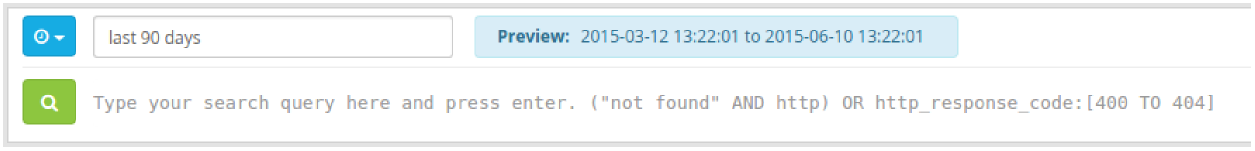
```
resource:\/posts\/45326
```

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. You can read more about this here: *The Graylog index model explained*

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.


last 90 days
Preview: 2015-03-12 13:22:01 to 2015-06-10 13:22:01
Type your search query here and press enter. ("not found" AND http) OR http_response_code:[400 TO 404]

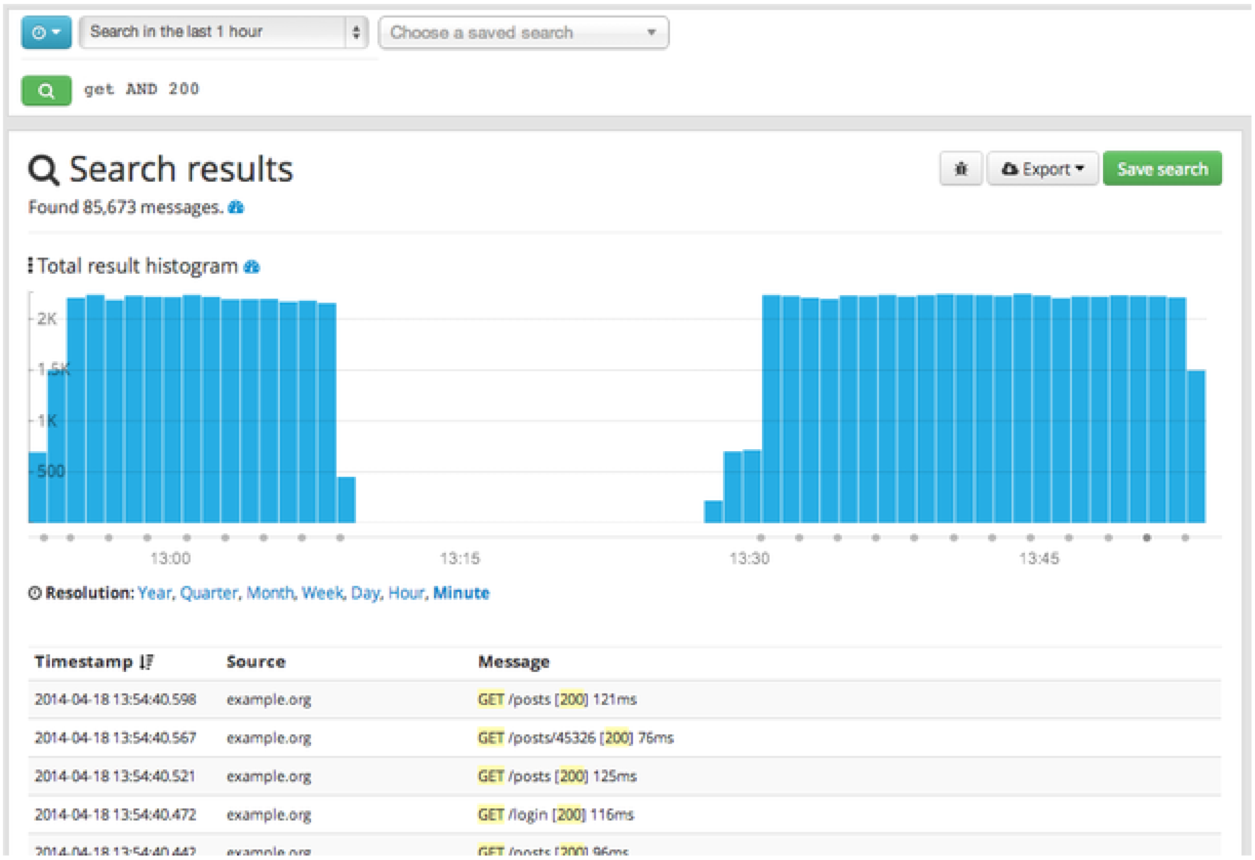
Here are a few examples for possible values.

- "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.

Search result highlighting

Graylog supports search result highlighting since v0.20.2:



Timestamp	Source	Message
2014-04-18 13:54:40.598	example.org	GET /posts [200] 121ms
2014-04-18 13:54:40.567	example.org	GET /posts/45326 [200] 76ms
2014-04-18 13:54:40.521	example.org	GET /posts [200] 125ms
2014-04-18 13:54:40.472	example.org	GET /login [200] 116ms
2014-04-18 13:54:40.442	example.org	GET /posts [200] 96ms

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-server` nodes:

```
allow_highlighting = true
```

CHAPTER 7

Streams

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: (stream rules are AND connected)

- 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 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.

The next parts of this document cover how to be alerted in case of too many errors, some specific error types that should never happen or how to forward the errors to another system or endpoint.

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.

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*
4. The stream is now saved but **not yet activated**. Add stream rules in the next dialogue. Try it against some messages by entering a message ID on the same page. Save the rules when the right messages are matched or not matched.
5. The stream is marked as *paused* in the list of streams. Activate the stream by hitting *Resume this stream* in the *Action* dropdown.

Alerts

You can define conditions that trigger alerts. For example whenever the stream *All production exceptions* has more than 50 messages per minute or when the field *milliseconds* had a too high standard deviation in the last five minutes.

Hit *Manage alerts* in the stream *Action* dropdown to see already configured alerts, alerts that were fired in the past or to configure new alert conditions.

You can configure the interval for alert checks in your *graylog.conf* using the *alert_check_interval* variable. The default is to check for alerts every 60 seconds.

Graylog ships with default *alert callbacks* and can be extended with plugins

Alert condition types explained

Message count condition

This condition triggers whenever the stream received more than X messages in the last Y minutes. Perfect for example to be alerted when there are many exceptions on your platform. Create a stream that catches every error message and be alerted when that stream exceeds normal throughput levels.

Field value condition

Triggers whenever the result of a statistical computation of a numerical message field in the stream is higher or lower than a given threshold. Perfect to monitor for performance problems: Be alerted whenever the standard deviation of the response time of your application was higher than X in the last Y minutes.

Field string value condition

This condition triggers whenever the stream received at least one message since the last alert run that has a field set to a given value. Get an alert when a message with the field *type* set to *security* arrives in the stream.

Important: We do not recommend to run this on analyzed fields like *message* or *full_message* because it is broken down to terms and you might get unexpected alerts. For example a check for *security* would also alert if a message with the field set to *no security* is received because it was broken down to *no* and *security*. This only happens on the analyzed *message* and *full_message* in Graylog. Please also take note that only a single alert is raised for this condition during the alerting interval, although multiple messages containing the given value may have been received since the last alert.

What is the difference between alert callbacks and alert receivers?

There are two groups of entities configuring what happens when an alert is fired: Alarm callbacks and alert receivers.

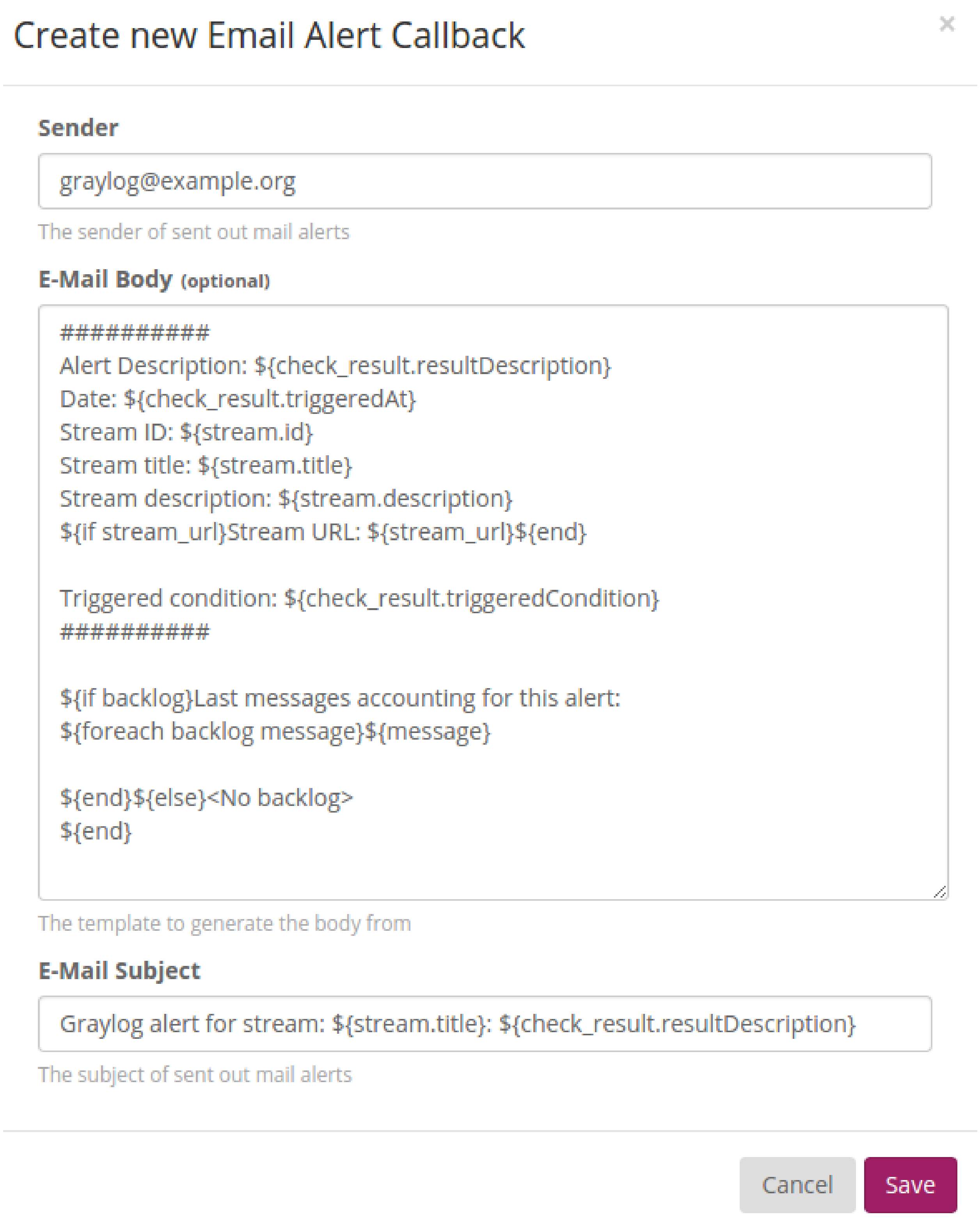
Alarm callbacks are a list of events that are being processed when an alert is triggered. There could be an arbitrary number of alarm callbacks configured here. If there is no alarm callback configured at all, a default email transport will be used to notify about the alert. If one or more alarm callbacks are configured (which might include the email alarm callback or not) then all of them are executed for every alert.

If the email alarm callback is used because it appears once or multiple times in the alarm callback list, or the alarm callback list is empty so the email transport is used per default, then the list of alert receivers is used to determine which recipients should receive the alert nofications. Every Graylog user (which has an email address configured in their account) or email address in that list gets a copy of the alerts sent out.

Email Alert Callback

The email alert callback can be used to send an email to the configured alert receivers when the conditions are triggered.

Three configuration options are available for the alert callback to customize the email that will be sent.


Create new Email Alert Callback

Sender

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}
${if stream_url}Stream URL: ${stream_url}${end}

Triggered condition: ${check_result.triggeredCondition}
##########

${if backlog}Last messages accounting for this alert:
${foreach backlog message}${message}

${end}${else}<No backlog>
${end}
```

The template to generate the body from

E-Mail Subject

Graylog alert for stream: ${stream.title}: ${check_result.resultDescription}

The subject of sent out mail alerts

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.

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.

Use cases

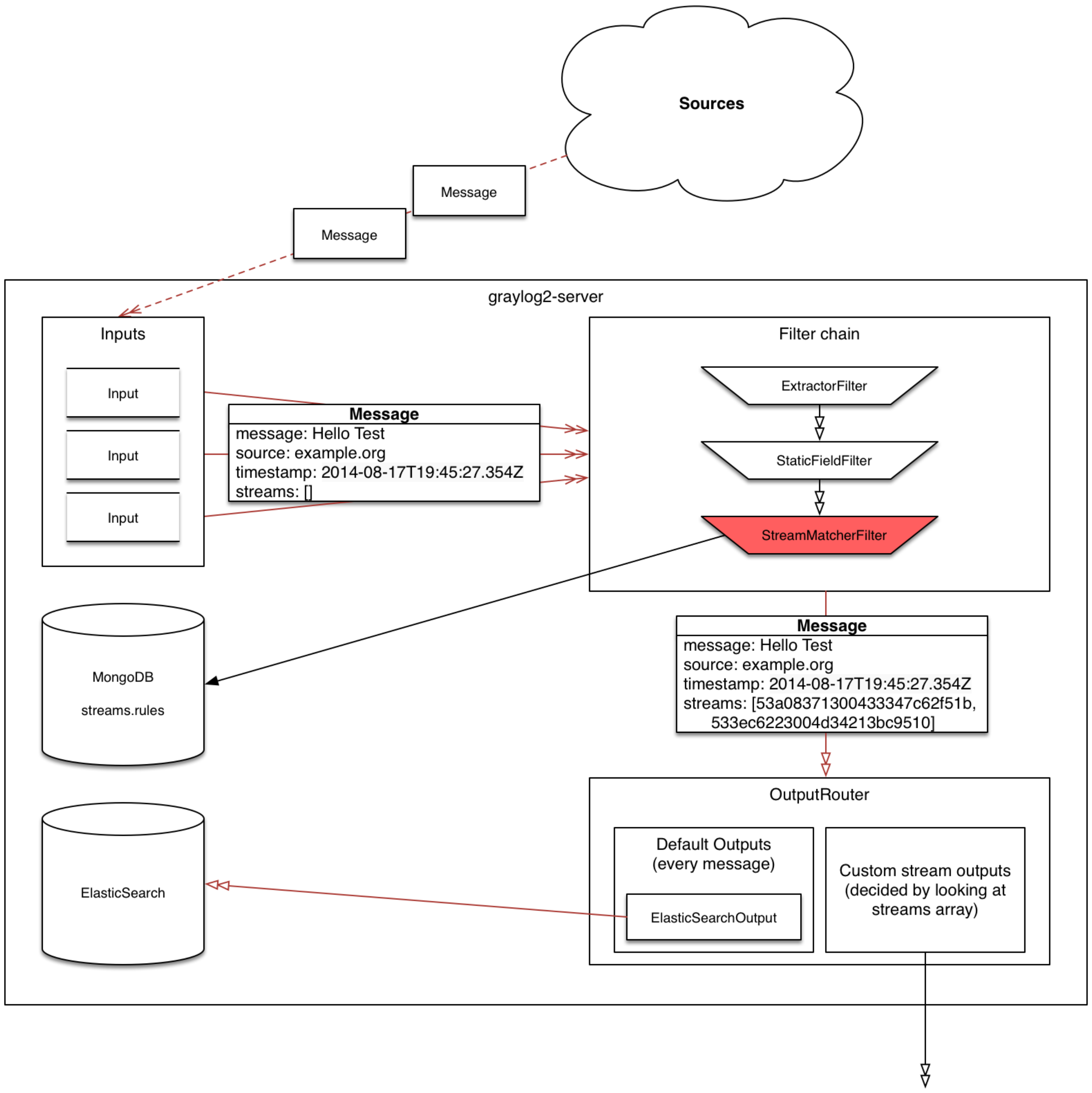
These are a few example use cases for streams:

- 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 the *quickvalues* 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 the *quickvalues* to get a list of users that performed the most logins in the search time frame.

How are streams processed internally?

The most important thing to know about Graylog stream matching is that there is no duplication of stored messages. Every message that comes in is matched against all rules of a stream. The internal ID of every stream that has *all* rules matching is appended to 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.



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. If every rule of a stream matches, the message is added to this stream. 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.

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`.

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.

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.

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.

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.

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.

FAQs

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.

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.

Can I write own outputs or alert callbacks methods?

Yes. Please refer to the plugins documentation page.

CHAPTER 8

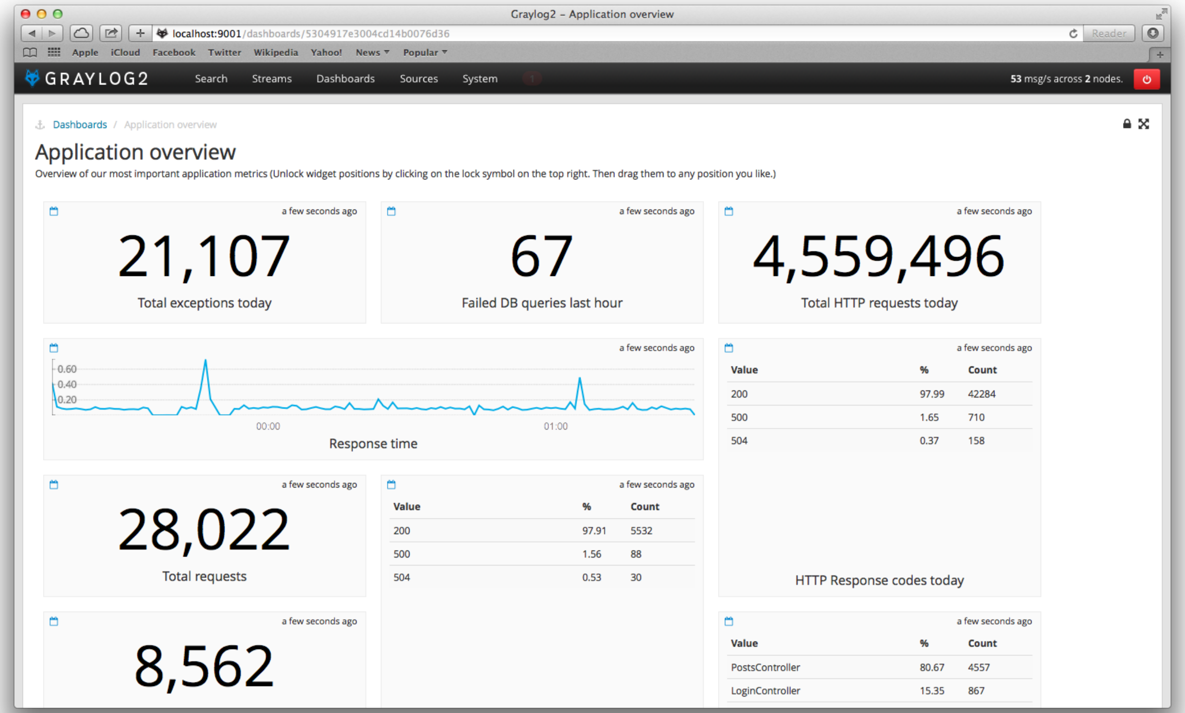
Dashboards

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.


Graylog2 – Application overview
localhost:9001/dashboards/5304917e3004cd14b0076d36
Apple iCloud Facebook Twitter Wikipedia Yahoo! News Popular
GRAYLOG2
Search Streams Dashboards Sources System
53 msg/s across 2 nodes.
Dashboards / Application overview
a few seconds ago
0.60
0.40
0.20
00:00
01:00

Application overview

Overview of our most important application metrics (Unlock widget positions by clicking on the lock symbol on the top right. Then drag them to any position you like.)

21,107
Total exceptions today

67
Failed DB queries last hour

4,559,496
Total HTTP requests today

Response time

Value	%	Count
200	97.99	42284
500	1.65	710
504	0.37	158

HTTP Response codes today

28,022
Total requests

Value	%	Count
200	97.91	5532
500	1.56	88
504	0.53	30

8,562

Value	%	Count
PostsController	80.67	4557
LoginController	15.35	867

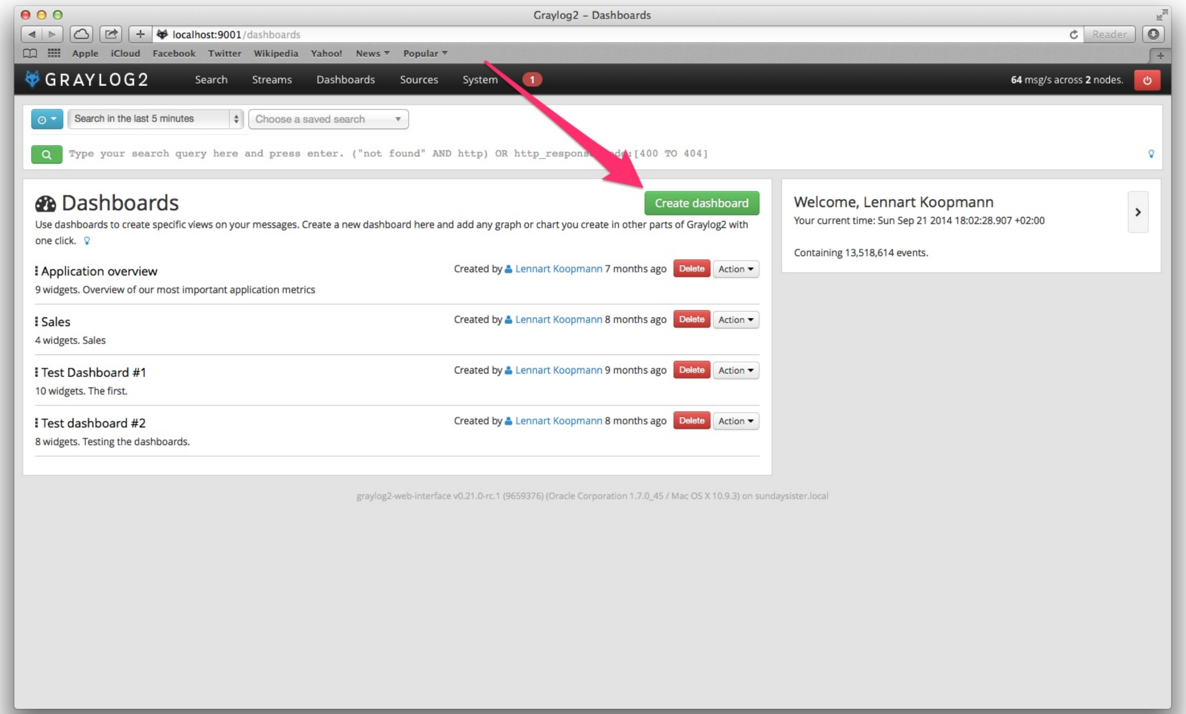
How to use dashboards

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.



64 msg/s across 2 nodes.

Type your search query here and press enter. ("not found" AND http) OR http_response_code:[400 TO 404]

Dashboards

Use dashboards to create specific views on your messages. Create a new dashboard here and add any graph or chart you create in other parts of Graylog2 with one click.

Application overview — Created by Lennart Koopmann 7 months ago
9 widgets. Overview of our most important application metrics

Sales — Created by Lennart Koopmann 8 months ago
4 widgets. Sales

Test Dashboard #1 — Created by Lennart Koopmann 9 months ago
10 widgets. The first.

Test dashboard #2 — Created by Lennart Koopmann 8 months ago
8 widgets. Testing the dashboards.

Welcome, Lennart Koopmann
Your current time: Sun Sep 21 2014 18:02:28.907 +02:00

Containing 13,518,614 events.

graylog2-web-interface v0.21.0-rc.1 (9659376) (Oracle Corporation 1.7.0_45 / Mac OS X 10.9.3) on sundaysister.local

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 just one click. The following search result types can be added to dashboards:

- Search result counts
- Search result histogram charts
- Field value charts
- Quickvalue results

Once you can see the results of your search, you will see a small blue icon next to the right of the result count and histogram title. Hovering over this will show "Add to dashboard" and clicking the icon will prompt you with a list of dashboards you've created. Select a dashboard to add the widget to it.

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 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

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.

Result

You should now see widgets on your dashboard. You will learn how to modify the dashboard, change cache times and widget positioning in the next chapter.

Modifying dashboards

You need to *unlock* dashboards to make any changes to them. Hit the lock icon in the top right corner of a dashboard to unlock it. You should now see new icons in the widget appearing.

Unlocked dashboard widgets explained

Unlocked dashboard widgets have three buttons that should be pretty self-explanatory.

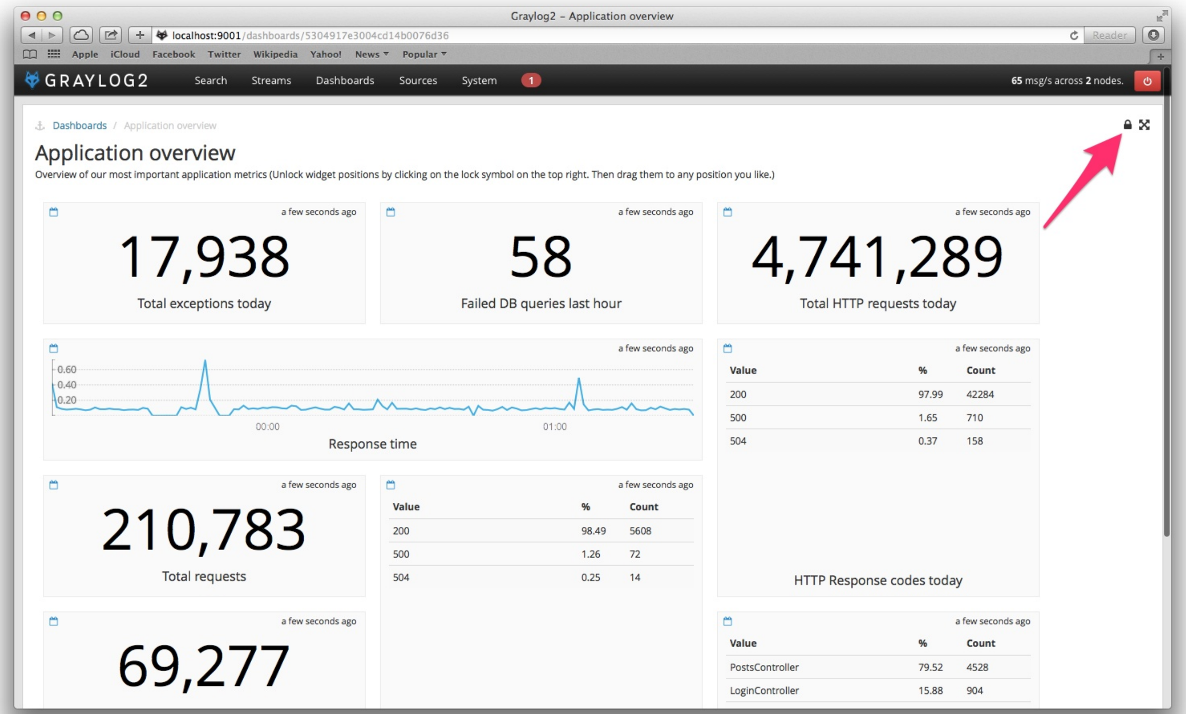
- Delete widget
- Change cache time of widget
- Change title of widget

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.

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.


Graylog2 – Application overview
localhost:9001/dashboards/5304917e3004cd14b0076d36
Reader
Apple iCloud Facebook Twitter Wikipedia Yahoo! News Popular
GRAYLOG2
Search
Streams
Dashboards
Sources
System
1
65 msg/s across 2 nodes.
Dashboards / Application overview

Application overview

Overview of our most important application metrics (Unlock widget positions by clicking on the lock symbol on the top right. Then drag them to any position you like.)

a few seconds ago
17,938
Total exceptions today

a few seconds ago
58
Failed DB queries last hour

a few seconds ago
4,741,289
Total HTTP requests today

a few seconds ago
Response time

a few seconds ago

Value	%	Count
200	97.99	42284
500	1.65	710
504	0.37	158

HTTP Response codes today

a few seconds ago
210,783
Total requests

a few seconds ago

Value	%	Count
200	98.49	5608
500	1.26	72
504	0.25	14

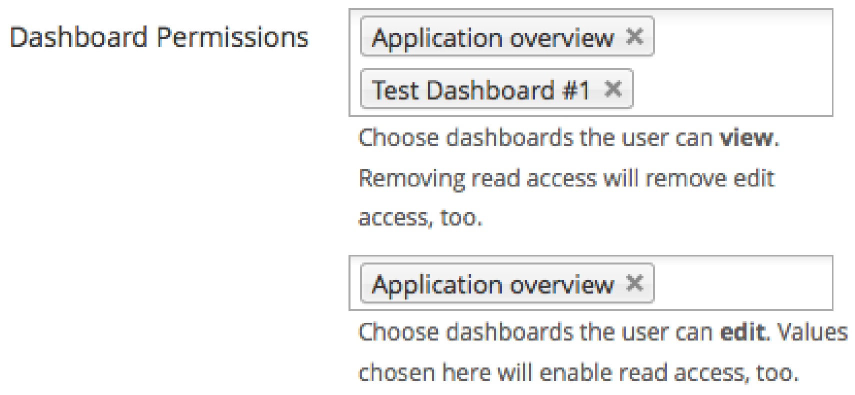
a few seconds ago
69,277

a few seconds ago

Value	%	Count
PostsController	79.52	4528
LoginController	15.88	904

Dashboard permissions

Graylog users with administrator permissions are always allowed to view and edit all dashboards. Users with *reader* permissions are by default not allowed to view or edit **any** dashboard.



Dashboard Permissions

Application overview ×
Test Dashboard #1 ×

Choose dashboards the user can **view**. Removing read access will remove edit access, too.

Application overview ×

Choose dashboards the user can **edit**. Values chosen here will enable read access, too.

Navigate to *System* -> *Users* and select a *reader* user you wish to give dashboard permissions. Hit the *edit* button and assign dashboard *view* and *edit* permissions in the edit user dialogue. Don't forget to save the user!

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.

CHAPTER 9

Extractors

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.

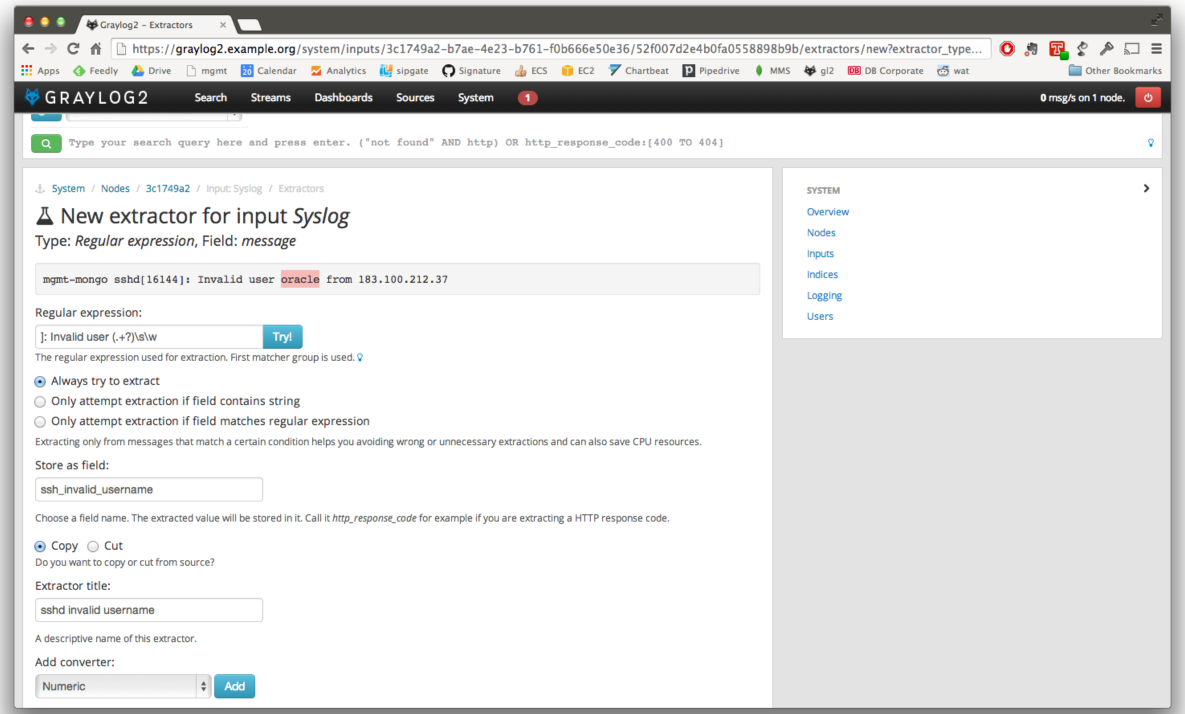
We decided not to write custom message inputs and parsers for all those thousands of devices, formats, firmwares and configuration parameters out there but came up with the concept of *Extractors* introduced the *v0.20.0* series of Graylog.

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.

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:


Graylog2 – Extractors
https://graylog2.example.org/system/inputs/3c1749a2-b7ae-4e23-b761-f0b666e50e36/52f007d2e4b0fa0558898b9b/extractors/new?extractor_type...
Apps Feedly Drive mgmt Calendar Analytics sipgate Signature ECS EC2 Chartbeat Pipedrive MMS gl2 DB Corporate wat Other Bookmarks
GRAYLOG2 Search Streams Dashboards Sources System 1
0 msg/s on 1 node.
Type your search query here and press enter. ("not found" AND http) OR http_response_code:[400 TO 404]
System / Nodes / 3c1749a2 / Input: Syslog / Extractors
SYSTEM
Overview
Nodes
Inputs
Indices
Logging
Users

New extractor for input *Syslog*

Type: *Regular expression*, Field: *message*

```
mgmt-mongo sshd[16144]: Invalid user oracle from 183.100.212.37
```

Regular expression:]: Invalid user (.+?)\s\w Try!

The regular expression used for extraction. First matcher group is used.

[✓ Always try to extract]
Only attempt extraction if field contains string
Only attempt extraction if field matches regular expression

Extracting only from messages that match a certain condition helps you avoiding wrong or unnecessary extractions and can also save CPU resources.

Store as field: ssh_invalid_username

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: sshd invalid username

A descriptive name of this extractor.

Add converter: Numeric Add

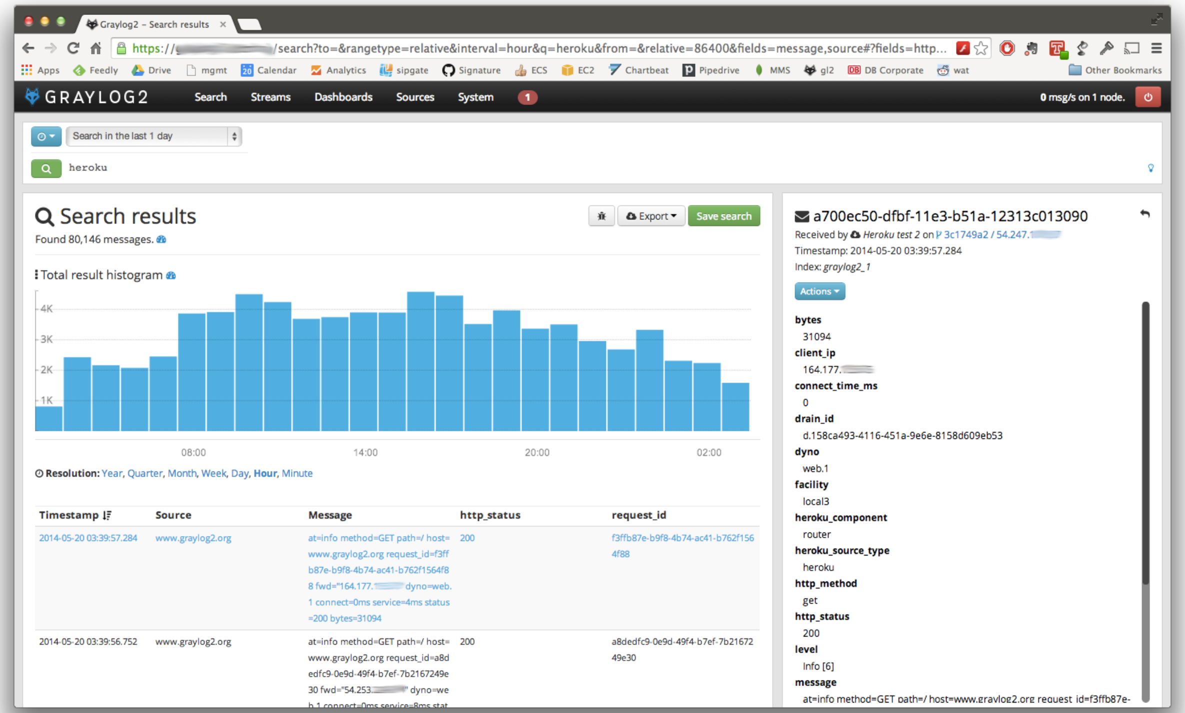
You can also choose to apply so called *converters* on the extracted value to for example convert a string consisting of numbers to an integer or double value (important for range searches later), anonymize IP addresses, lower-/uppercase a string, build a hash value, and much more.

The extractor directory

The data source library provides access to a lot of extractors that you can easily import into your Graylog setup.

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)



Search results

Found 80,146 messages.

Resolution: Year, Quarter, Month, Week, Day, Hour, Minute

Timestamp	Source	Message	http_status	request_id
2014-05-20 03:39:57.284	www.graylog2.org	at=info method=GET path=/ host=www.graylog2.org request_id=f3ffb87e-b9f8-4b74-ac41-b762f1564f88 fwd="164.177.[redacted]" dyno=web.1 connect=0ms service=4ms status=200 bytes=31094	200	f3ffb87e-b9f8-4b74-ac41-b762f1564f88
2014-05-20 03:39:56.752	www.graylog2.org	at=info method=GET path=/ host=www.graylog2.org request_id=a8dedfc9-0e9d-49f4-b7ef-7b2167249e30 fwd="54.253.[redacted]" dyno=web.1 connect=0ms service=8ms stat	200	a8dedfc9-0e9d-49f4-b7ef-7b216724 9e30

a700ec50-dfbf-11e3-b51a-12313c013090

Received by *Heroku test 2* on 3c1749a2 / 54.247.[redacted]
Timestamp: 2014-05-20 03:39:57.284
Index: *graylog2_1*

bytes	31094
client_ip	164.177.[redacted]
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-

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.

To for example 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 `(?:)`.

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.

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.

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:

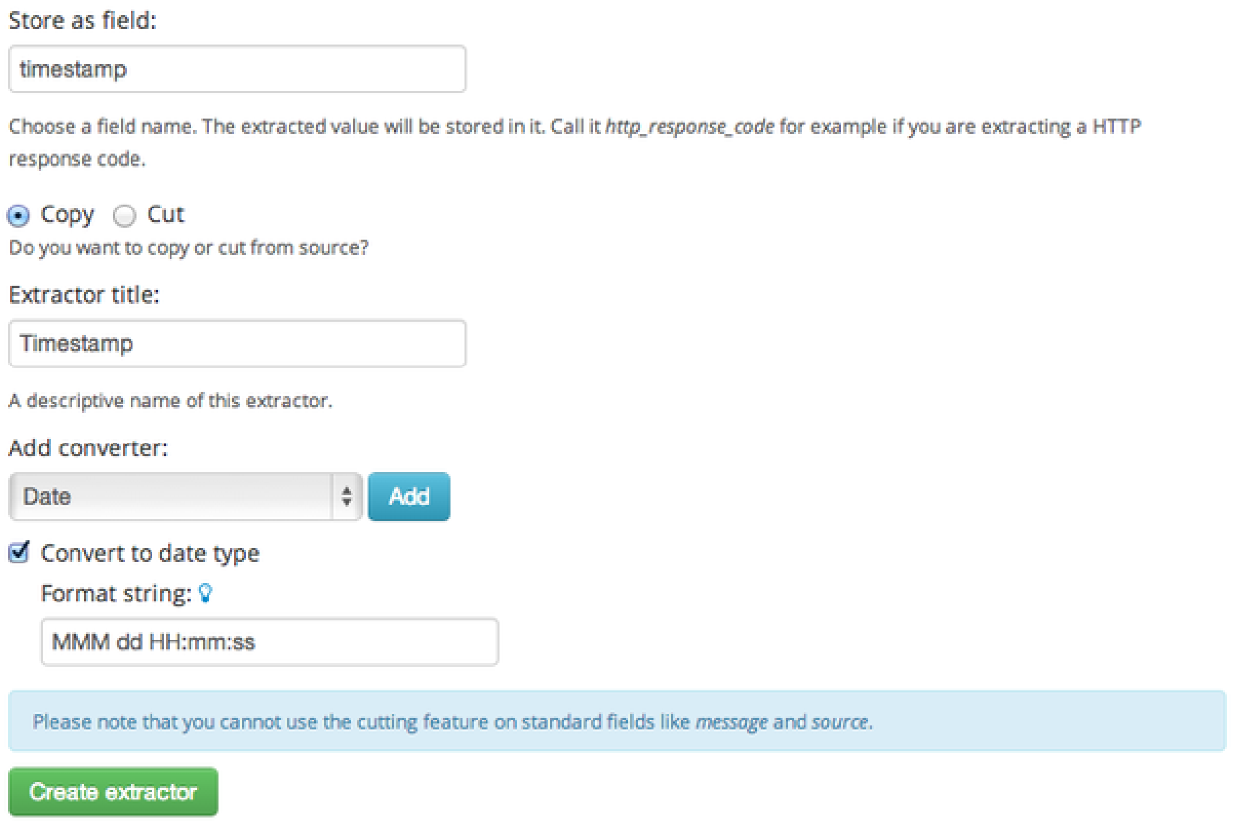
```
<131>: foo-bar-dc3-org-de01: Mar 12 00:45:38: %LINK-3-UPDOWN: Interface␣
↪GigabitEthernet0/31, changed state to down
```

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 standard 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:

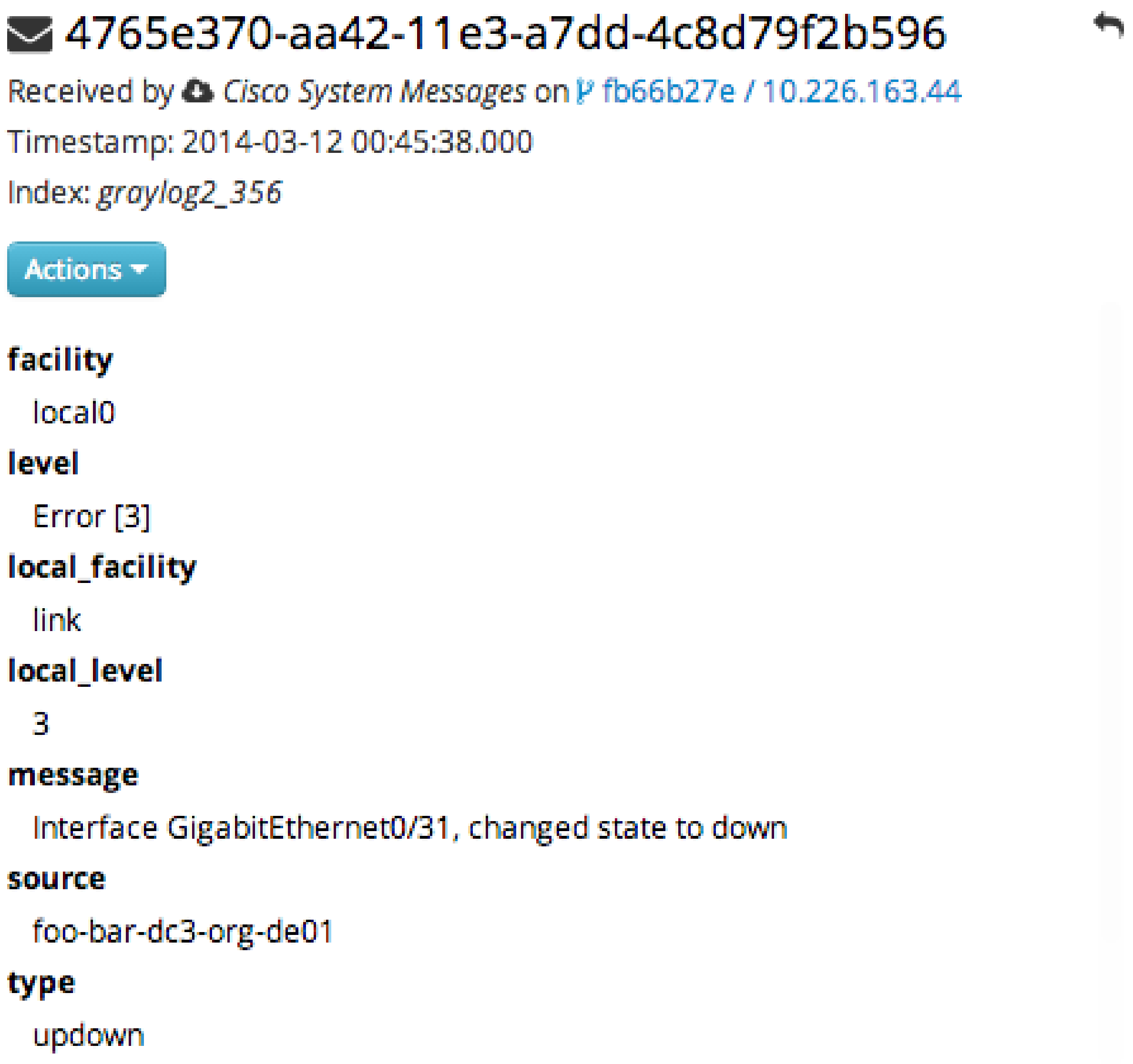
Date

[✓] 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*.


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*

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	‘

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).

CHAPTER 10

Message rewriting with Drools

Graylog can optionally use Drools Expert to evaluate all incoming messages against a user defined rules file. Each message will be evaluated prior to being written to the outputs.

The rule file location is defined in the Graylog configuration file:

```
# Drools Rule File (Use to rewrite incoming log messages)
rules_file = /etc/graylog.d/rules/graylog.drl
```

The rules file is located on the file system with a `.drl` file extension. The rules file can contain multiple rules, queries and functions, as well as some resource declarations like imports, globals, and attributes that are assigned and used by your rules and queries.

For more information on the DRL rules syntax please read the Drools User Guide.

Getting Started

1. Uncomment the `rules_file` line in the Graylog configuration file.
2. Copy the sample rules file to the location specified in your Graylog configuration file.
3. Modify the rules file to parse/rewrite/filter messages as needed.

Example rules file

This is an example rules file:

```
import org.graylog2.plugin.Message

rule "Rewrite localhost host"
    when
        m : Message( source == "localhost" )
    then
```

```
        m.addField("source", "localhost.example.com" );
        System.out.println( "[Overwrite localhost rule fired] : " + m.toString() );
end

rule "Drop UDP and ICMP Traffic from firewall"
    when
        m : Message( getField("full_message") matches "(?i).*(ICMP|UDP) Packet(.
↪|\n|\r)*" && source == "firewall" )
    then
        m.setFilterOut(true);
        System.out.println("[Drop all syslog ICMP and UDP traffic] : " + m.toString()␣
↪);
end
```

Parsing Message and adding fields

In the following script we turn the PID and the src IP into additional fields:

```
import org.graylog2.plugin.Message
import java.util.regex.Matcher
import java.util.regex.Pattern

// Raw Syslog Apr 18 15:34:58 server01 smtp-glass[3371]: NEW (1/0) on=1.1.1.1:9100,␣
↪src=2.2.2.2:38776, ident=, dst=3.3.3.3:25, id=1303151698.3371
rule "SMTP Glass Logging to GELF"
  when
      m : Message( message matches "^smtp-glass.*" )
  then
      Matcher matcher = Pattern.compile("smtp-glass\\\[(\\\d+)].* src (\\\d+.\\\d+.
↪\\\d+.\\\d+)").matcher(m.getMessage());
      if (matcher.find()) {
         m.addField("_pid", Long.valueOf(matcher.group(1)));
         m.addField("_src", matcher.group(2));
      }
end
```

Another example: Adding additional fields and changing the message itself

We send Squid access logs to Graylog using Syslog. The problem is that the *host* field of the message was set to the IP addrress of the Squid proxy, which not very useful. This rule overwrites the source and adds other fields:

```
import org.graylog2.plugin.Message
import java.util.regex.Matcher
import java.util.regex.Pattern
import java.net.InetAddress;

/*
Raw Syslog: squid[2099]: 1339551529.881  55647 1.2.3.4 TCP_MISS/200 22 GET http://www.
↪google.com/

squid\[\d+\]: (\d+\.\d+) *(\d+) *(\d+.\d+.\d+.\d+) *(\w+\/\w+) (\d+) (\w+) (.*)
matched: 13:1339551529.881
matched: 29:55647
matched: 35:1.2.3.4
```

```
matched: 47:TCP_MISS/200
matched: 60:22
matched: 64:GET
matched: 68:http://www.google.com/
*/

rule "Squid Logging to GELF"
    when
        m : Message( getField("facility") == "local5" )
    then
        Matcher matcher = Pattern.compile("squid\\[\\d+\\]: (\\d+.\\d+) *(\\d+)␣
↪*(\\d+.\\d+.\\d+.\\d+) *(\\w+\\/\\w+) (\\d+) (\\w+) (.*)").matcher(m.getMessage());

        if (matcher.find()) {
            m.addField("facility", "squid");
            InetAddress addr = InetAddress.getByName(matcher.group(3));
            String host = addr.getHostName();
            m.addField("source",host);
            m.addField("message",matcher.group(6) + " " + matcher.group(7));
            m.addField("_status",matcher.group(4));
            m.addField("_size",matcher.group(5));
        }
end
```

CHAPTER 11

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.

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

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 /system/lbstatus
```

The status knows two different states, `ALIVE` 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 `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 /system/lbstatus/override/alive
PUT /system/lbstatus/override/dead
```

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.

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.

Web Interface

It is possible to use the Graylog web interface behind a load balancer for high availability purposes.

However, in order to make the various metrics work in Graylog's web interface, you need to enable sticky sessions in your load balancer, or configure the second instance to be a failover instance only, which only gets requests in case the first instance is no longer reachable.

There are various terms used for sticky sessions. Session persistence or session management are also in use.

Please refer to your vendor's documentation to learn about how to enable sticky sessions.

Information for some popular load balancers and their settings can be found through the following links:

- Amazon Web Services ELB
- HAProxy
- F5 BIG-IP
- KEMP

CHAPTER 12

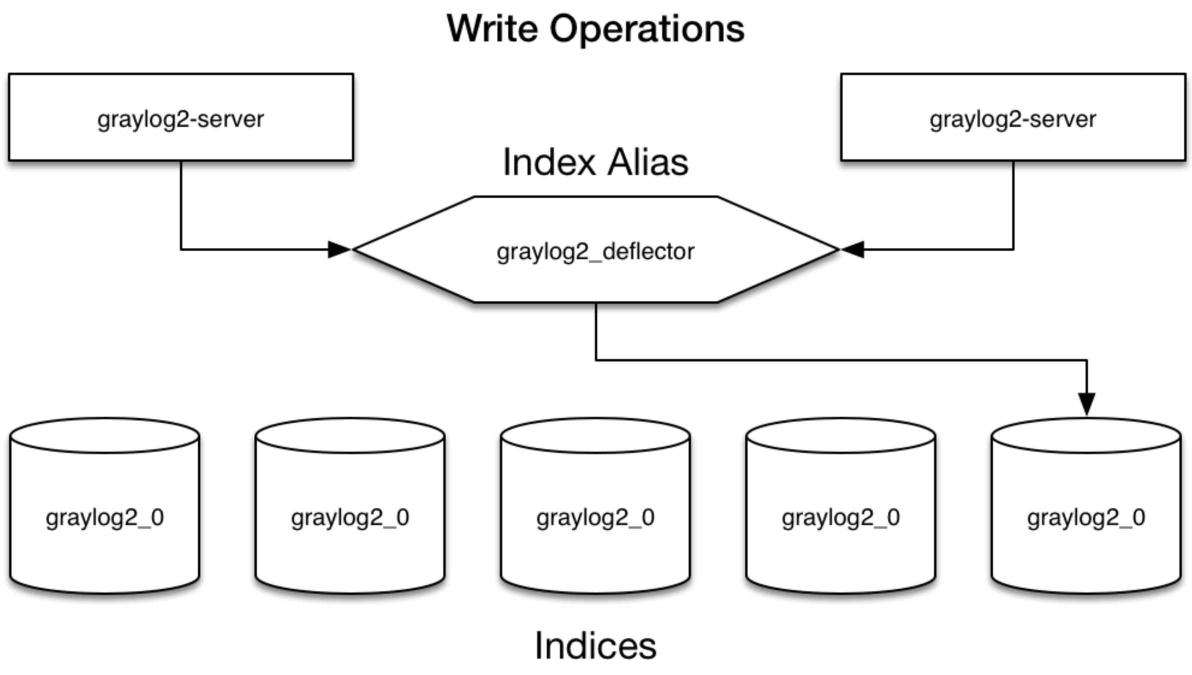
The Graylog index model explained

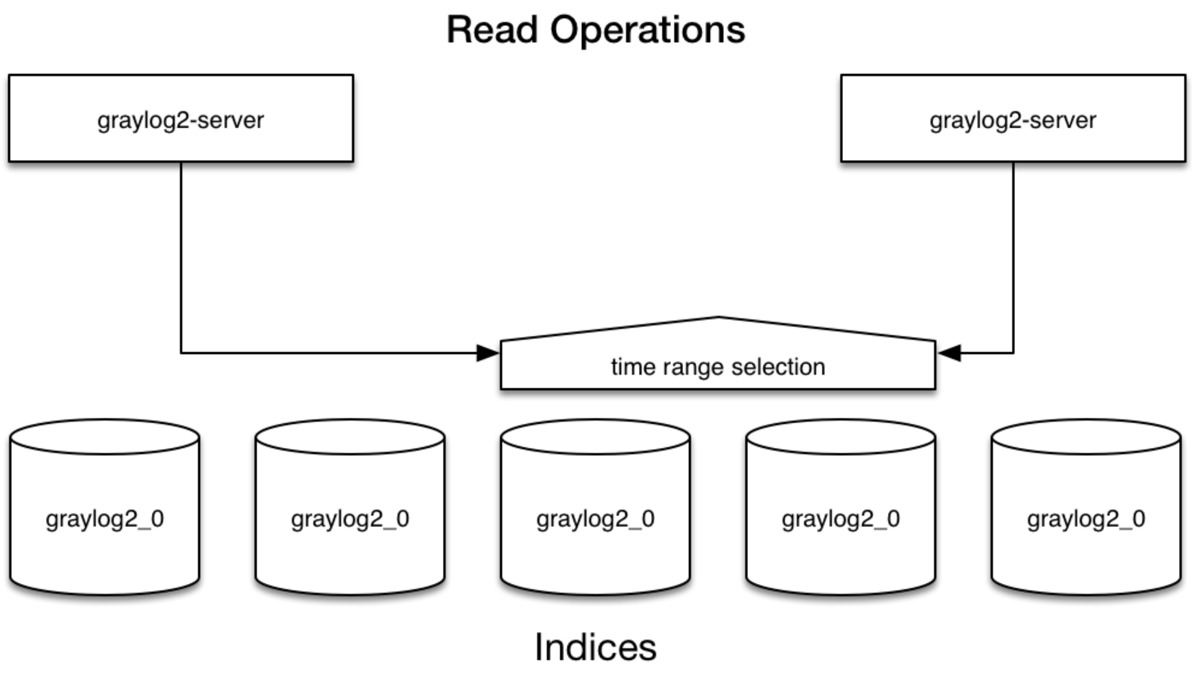
Overview

Graylog is transparently managing a set of indices to optimise search and analysis operations for speed and low resource utilisation. The system is maintaining an index alias called *graylog_deflector* that is always pointing to the current write-active index. We always have exactly one index to which new messages are appended until the configured maximum size (`elasticsearch_max_docs_per_index` in your `graylog.conf`) is reached.

A background task is running every minute and checks if the maximum size is reached. A new index is created and prepared when that happens. Once the index is considered to be ready to be written to, the `graylog_deflector` is atomically switched to the it. That means that all writing nodes can always write to the deflector alias without even knowing what the currently active write-active index is.

Note that there are also time based retention settings since v1.0 of Graylog. This allows you to instruct Graylog to keep messages based on their age and not the total amount. You can find the corresponding configuration settings in your `graylog.conf`.





Almost every read operation is performed with a given time range. Because Graylog is only writing sequentially it can keep a cached collection of information about which index starts at what point in time. It selects a lists of indices to query when having a time range provided. If no time range is provided it will search in all indices it knows.

Eviction of indices and messages

You have configured the maximum number of indices in your `graylog.conf` (`elasticsearch_max_number_of_indices`). When that number is reached the oldest indices will automatically be deleted. The deleting is performed by the *graylog-server* master node in a background process that is continuously comparing the actual number of indices with the configured maximum:

```
elasticsearch_max_docs_per_index * elasticsearch_max_number_of_indices
  = maximum number of messages stored
```

Keeping the metadata in synchronisation

Graylog will on notify you when the stored metadata about index time ranges has run out of sync. This can for example happen when you delete indices by hand. 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 synchronisation problems:

```
$ curl -XPOST http://127.0.0.1:12900/system/indices/ranges/rebuild
```

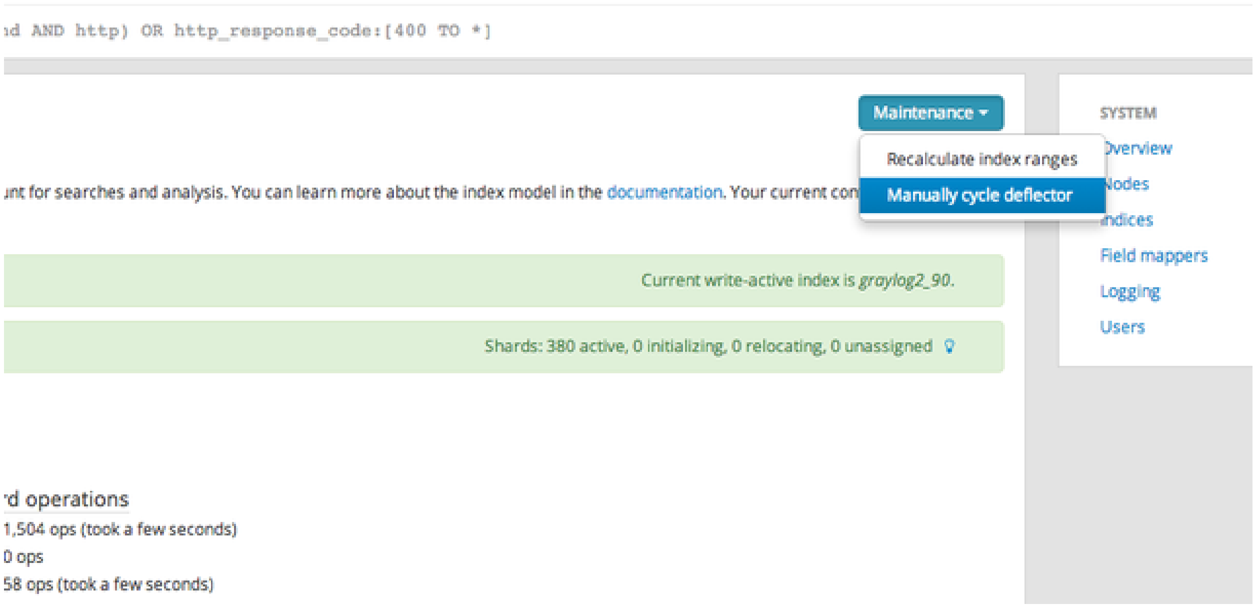
This will trigger a systemjob:

```
INFO : org.graylog2.system.jobs.SystemJobManager - Submitted SystemJob <ef7057c0-5ae3-
↪11e3-b935-4c8d79f2b596> [org.graylog2.indexer.ranges.RebuildIndexRangesJob]
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Re-calculating index␣
↪ranges.
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Calculated range of␣
↪[graylog2_56] in [640ms].
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Calculated range of␣
↪[graylog2_18] in [66ms].
...
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Done calculating index␣
↪ranges for 88 indices. Took 4744ms.
INFO : org.graylog2.system.jobs.SystemJobManager - SystemJob <ef7057c0-5ae3-11e3-b935-
↪4c8d79f2b596> [org.graylog2.indexer.ranges.RebuildIndexRangesJob] finished in␣
↪4758ms.
```

Manually cycling the deflector

Sometimes you might want to cycle the deflector manually and not wait until the configured maximum number of messages in the newest index is reached. You can do this either via a REST call against the *graylog-server* master node or via the web interface:

```
$ curl -XPOST http://127.0.0.1:12900/system/deflector/cycle
```



ınt for searches and analysis. You can learn more about the index model in the documentation. Your current con

Current write-active index is *graylog2_90*.

Shards: 380 active, 0 initializing, 0 relocating, 0 unassigned

'd operations
1,504 ops (took a few seconds)
0 ops
58 ops (took a few seconds)

This triggers the following log output:

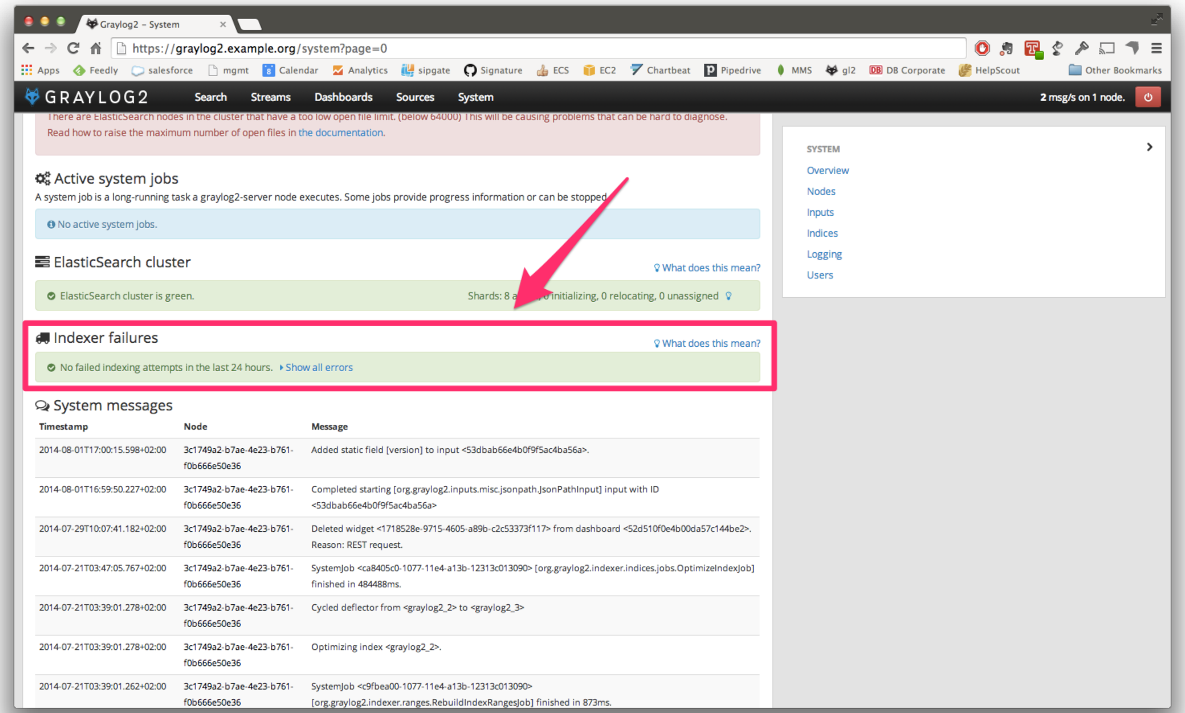
```
INFO : org.graylog2.rest.resources.system.DeflectorResource - Cycling deflector.␣
↪Reason: REST request.
INFO : org.graylog2.indexer.Deflector - Cycling deflector to next index now.
INFO : org.graylog2.indexer.Deflector - Cycling from <graylog2_90> to <graylog2_91>
INFO : org.graylog2.indexer.Deflector - Creating index target <graylog2_91>...
INFO : org.graylog2.indexer.Deflector - Done!
INFO : org.graylog2.indexer.Deflector - Pointing deflector to new target index....
INFO : org.graylog2.indexer.Deflector - Flushing old index <graylog2_90>.
INFO : org.graylog2.indexer.Deflector - Setting old index <graylog2_90> to read-only.
INFO : org.graylog2.system.jobs.SystemJobManager - Submitted SystemJob <a05e0d60-5c34-
↪11e3-8df7-4c8d79f2b596> [org.graylog2.indexer.indices.jobs.OptimizeIndexJob]
INFO : org.graylog2.indexer.Deflector - Done!
INFO : org.graylog2.indexer.indices.jobs.OptimizeIndexJob - Optimizing index
↪<graylog2_90>.
INFO : org.graylog2.system.jobs.SystemJobManager - SystemJob <a05e0d60-5c34-11e3-8df7-
↪4c8d79f2b596> [org.graylog2.indexer.indices.jobs.OptimizeIndexJob] finished in␣
↪334ms.
```

CHAPTER 13

Indexer failures and dead letters

Indexer failures

Every `graylog-server` instance 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.



There are ElasticSearch nodes in the cluster that have a too low open file limit. (below 64000) This will be causing problems that can be hard to diagnose. Read how to raise the maximum number of open files in the documentation.

Active system jobs

A system job is a long-running task a graylog2-server node executes. Some jobs provide progress information or can be stopped.

No active system jobs.

ElasticSearch cluster

ElasticSearch cluster is green. Shards: 8 a… 0 initializing, 0 relocating, 0 unassigned

Indexer failures

No failed indexing attempts in the last 24 hours.

System messages

Timestamp	Node	Message
2014-08-01T17:00:15.598+02:00	3c1749a2-b7ae-4e23-b761-f0b666e50e36	Added static field [version] to input <53dbab66e4b0f9f5ac4ba56a>.
2014-08-01T16:59:50.227+02:00	3c1749a2-b7ae-4e23-b761-f0b666e50e36	Completed starting [org.graylog2.inputs.misc.jsonpath.JsonPathInput] input with ID <53dbab66e4b0f9f5ac4ba56a>
2014-07-29T10:07:41.182+02:00	3c1749a2-b7ae-4e23-b761-f0b666e50e36	Deleted widget <1718528e-9715-4605-a89b-c2c53373f117> from dashboard <52d510f0e4b00da57c144be2>. Reason: REST request.
2014-07-21T03:47:05.767+02:00	3c1749a2-b7ae-4e23-b761-f0b666e50e36	SystemJob <ca8405c0-1077-11e4-a13b-12313c013090> [org.graylog2.indexer.indices.jobs.OptimizeIndexJob] finished in 484488ms.
2014-07-21T03:39:01.278+02:00	3c1749a2-b7ae-4e23-b761-f0b666e50e36	Cycled deflector from <graylog2_2> to <graylog2_3>
2014-07-21T03:39:01.278+02:00	3c1749a2-b7ae-4e23-b761-f0b666e50e36	Optimizing index <graylog2_2>.
2014-07-21T03:39:01.262+02:00	3c1749a2-b7ae-4e23-b761-f0b666e50e36	SystemJob <c9fbea00-1077-11e4-a13b-12313c013090> [org.graylog2.indexer.ranges.RebuildIndexRangesJob] finished in 873ms.

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.

Dead letters

This is an experimental feature. You can enable the dead letters feature in your `graylog-server.conf` like this:

```
dead_letters_enabled = true
```

Graylog will write every message that could not be written to Elasticsearch into the MongoDB `dead_letters` collection. The messages will be waiting there for you to be processed in some other way. You could write a script that reads every message from there and transforms it in a way that will allow Graylog to accept it.

A dead letter in MongoDB has exactly the structure (in the `message` field) like the message that would have been written to the indices:

```
$ mongo
MongoDB shell version: 2.4.1
connecting to: test
> use graylog2
switched to db graylog2
> db.dead_letters.find().limit(1).pretty()
{
    "_id" : ObjectId("530a951b3004ada55961ee22"),
    "message" : {
        "timestamp" : "2014-02-24 00:40:59.121",
        "message" : "failing",
        "failure" : "haha",
        "level" : NumberLong(6),
        "_id" : "544575a0-9cec-11e3-b502-4c8d79f2b596",
        "facility" : "gelf-rb",
        "source" : "sundaysister",
        "gl2_source_input" : "52ef64d03004faafd4bb0fc2",
        "gl2_source_node" : "fb66b27e-993c-4595-940f-dd521dcdaa93",
        "file" : "(irb)",
        "line" : NumberLong(37),
        "streams" : [ ],
        "version" : "1.0"
    },
    "timestamp" : ISODate("2014-02-24T00:40:59.137Z"),
    "letter_id" : "54466000-9cec-11e3-b502-4c8d79f2b596"
}
```

The `timestamp` is the moment in time when the message could not be written to the indices and the `letter_id` references to the failed indexing attempt and its error message.

Every failed indexing attempt comes with a field called `written` that indicates if a dead letter was created or not:

```
> db.index_failures.find().limit(1).pretty()
{
  "_id" : ObjectId("530a951b3004ada55961ee23"),
  "timestamp" : ISODate("2014-02-24T00:40:59.136Z"),
  "message" : "MapperParsingException[failed to parse [failure]]; nested:␣
↪NumberFormatException[For input string: \"haha\"]; ",
  "index" : "graylog2_324",
  "written" : true,
  "letter_id" : "54466000-9cec-11e3-b502-4c8d79f2b596",
  "type" : "message"
}
```

Common indexer failure reasons

There are some common failures that can occur under certain circumstances. Those are explained here:

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 *The Graylog index model explained* for more information.

CHAPTER 14

Plugins

General information

Graylog comes with a stable plugin API for the following plugin types since Graylog 1.0:

- **Inputs:** Accept/write any messages into Graylog
- **Outputs:** Forward messages to other endpoints in real-time
- **Services:** Run at startup and able to implement any functionality
- **Alarm Callbacks:** Called when a stream alert condition has been triggered
- **Filters:** Transform/drop incoming messages during processing
- **REST API Resources:** A REST resource to expose as part of the `graylog-server` REST API
- **Periodical:** Called at periodical intervals during server runtime

The first 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.

Creating a plugin skeleton

The easiest way to get started is to use our maven archetype that will create a complete plugin project infrastructure will all required classes, build definitions, and configurations using an interactive wizard.

Maven is a Java widely used build tool that comes pre-installed on many operating systems or can be installed using most package managers. Make sure that it is installed with at least version 3 before you go on.

Use it like this:

```
$ mvn archetype:generate -DarchetypeGroupId=org.graylog -DarchetypeArtifactId=graylog-
↪plugin-archetype
```

It wil 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: jira-alarmcallback
version: 1.0.0-SNAPSHOT
package: com.acmecorp
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 but nothing that is related to the actual implementation. More on this in the example plugin chapters later.

You now have a new folder called `jira-alarmcallback` that includes a complete plugin skeleton including Maven build files. 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 skeleton as a fully configured Java project.

Change some default values

Open the `JiraAlarmCallbackMetaData.java` file and customize the default values like the plugin description, the website URI, and so on. Especially the author name etc. should be changed.

Now go on with implementing the actual login in one of the example plugin chapters below.

Example Alarm Callback plugin

Let's assume you still want to build the mentioned JIRA AlarmCallback plugin. First open the `JiraAlarmCallback.java` file and let it implement the `AlarmCallback` interface:

```
public class JiraAlarmCallback implements AlarmCallback
```

Your IDE should offer you to create the methods you need to implement:

public void initialize(Configuration configuration) throws AlarmCallbackConfigurationException

This is called once at the very beginning of the lifecycle of this plugin. It is common practive to store the `Configuration` as a private member for later access.

public void call(Stream stream, AlertCondition.CheckResult checkResult) throws AlarmCallbackException

This is the actual alarm callback being triggered. Implement your login that creates a JIRA ticket here.

public ConfigurationRequest getRequestedConfiguration()

Plugins can request configurations. The UI in the Graylog web interface is generated from this information and the filled out configuration values are passed back to the plugin in `initialize(Configuration configuration)`.

This is an example configuration request:

```
final ConfigurationRequest configurationRequest = new ConfigurationRequest();
configurationRequest.addField(new TextField(
        "service_key", "Service key", "", "JIRA API token. You can find this token in␣
↪your account settings.",
        ConfigurationField.Optional.NOT_OPTIONAL)); // required, must be filled out
```

```
configurationRequest.addField(new BooleanField(
        "use_https", "HTTPs", true,
        "Use HTTP for API communication?"));
```

public String getName()

Return a human readable name of this plugin.

public Map<String, Object> getAttributes()

Return attributes that might be interesting to be shown under the alarm callback in the Graylog web interface. It is common practice to at least return the used configuration here.

public void checkConfiguration() throws ConfigurationException

Throw a `ConfigurationException` if the user should have entered missing or invalid configuration parameters.

Registering the plugin

You now have to register your plugin in the `JiraAlarmCallbackModule.java` file to make `graylog-server` load the alarm callback when launching. The reason for the manual registering is that a plugin could consist of multiple plugin types. Think of the generated plugin file as a bundle of multiple plugins.

Register your new plugin using the `configure()` method:

```
@Override
protected void configure() {
    addAlarmCallback(JiraAlarmCallback.class);
}
```

Building plugins

Building the plugin is easy because the archetype has created all necessary files and settings for you. Just run `mvn package` from the plugin directory:

```
$ 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
```

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 `plugins/` relative from your `graylog-server` directory.

Restart `graylog-server` and the plugin should be available to use from the web interface immediately.

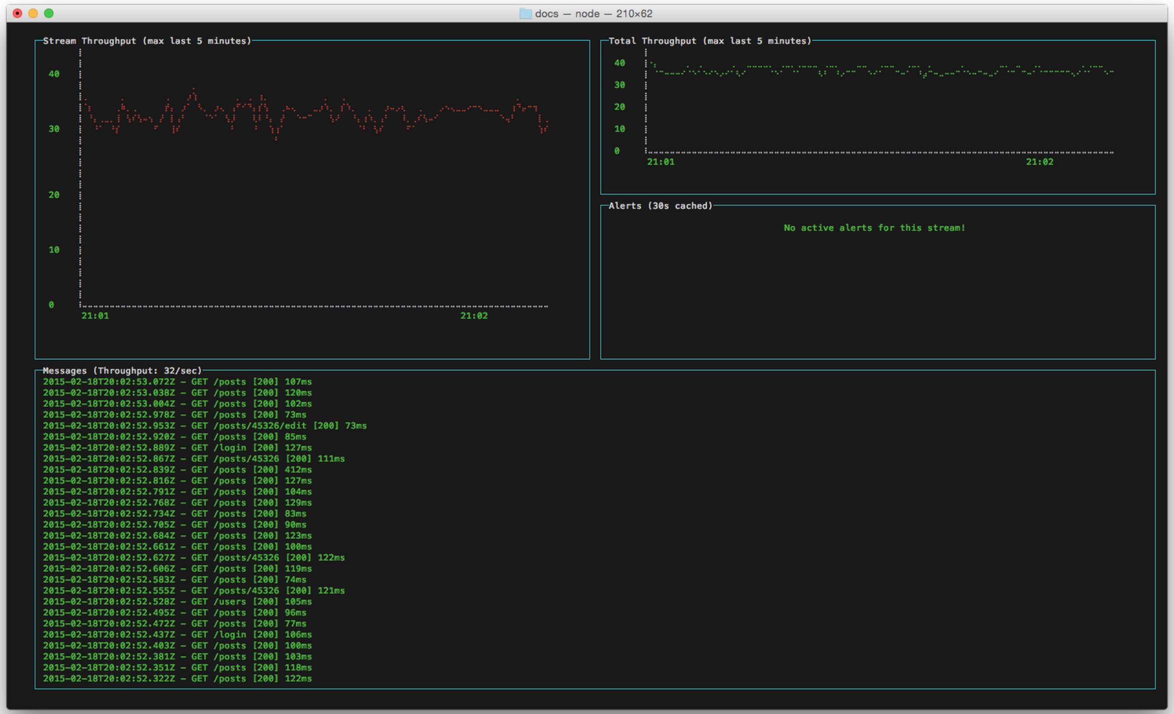
CHAPTER 15

External dashboards

There are other frontends that are connecting to the Graylog REST API and display data or information in a special way.

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.

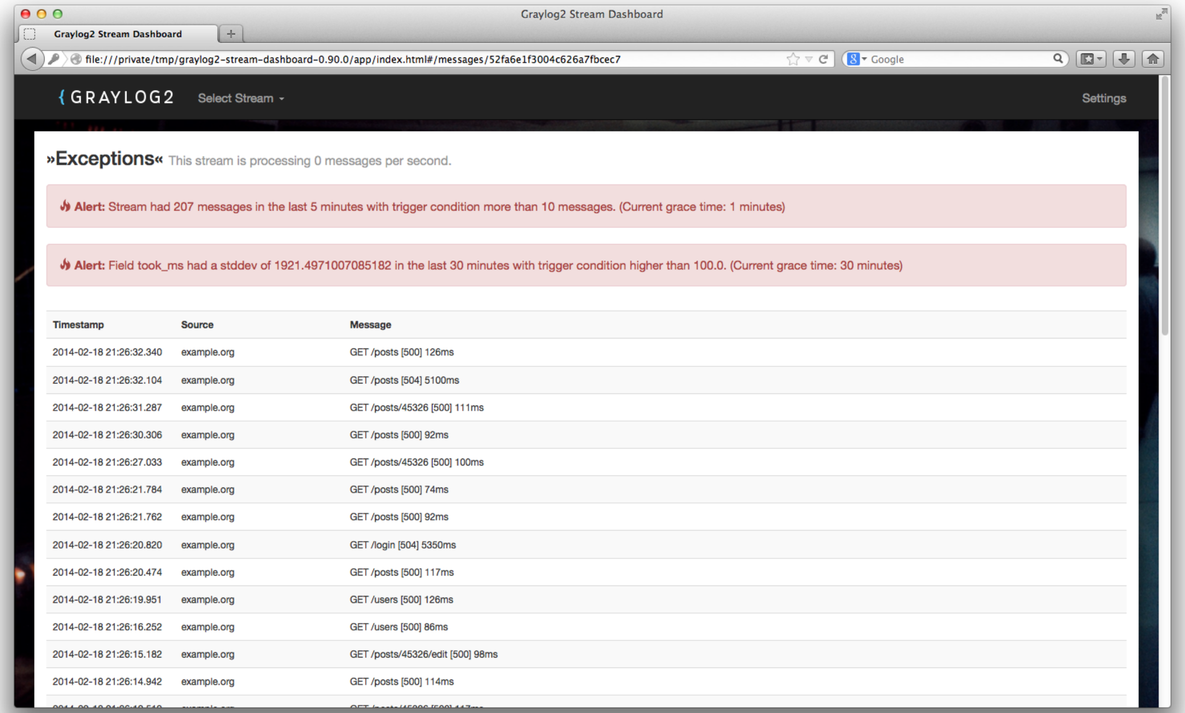

docs — node — 210×62
Stream Throughput (max last 5 minutes)
40
30
20
10
0
21:01
21:02
Total Throughput (max last 5 minutes)
40
30
20
10
0
21:01
21:02
Alerts (30s cached)
No active alerts for this stream!

```
Messages (Throughput: 32/sec)
2015-02-18T20:02:53.072Z - GET /posts [200] 107ms
2015-02-18T20:02:53.038Z - GET /posts [200] 120ms
2015-02-18T20:02:53.004Z - GET /posts [200] 102ms
2015-02-18T20:02:52.978Z - GET /posts [200] 73ms
2015-02-18T20:02:52.953Z - GET /posts/45326/edit [200] 73ms
2015-02-18T20:02:52.920Z - GET /posts [200] 85ms
2015-02-18T20:02:52.889Z - GET /login [200] 127ms
2015-02-18T20:02:52.867Z - GET /posts/45326 [200] 111ms
2015-02-18T20:02:52.839Z - GET /posts [200] 412ms
2015-02-18T20:02:52.816Z - GET /posts [200] 127ms
2015-02-18T20:02:52.791Z - GET /posts [200] 104ms
2015-02-18T20:02:52.768Z - GET /posts [200] 129ms
2015-02-18T20:02:52.734Z - GET /posts [200] 83ms
2015-02-18T20:02:52.705Z - GET /posts [200] 90ms
2015-02-18T20:02:52.684Z - GET /posts [200] 123ms
2015-02-18T20:02:52.661Z - GET /posts [200] 100ms
2015-02-18T20:02:52.627Z - GET /posts/45326 [200] 122ms
2015-02-18T20:02:52.606Z - GET /posts [200] 119ms
2015-02-18T20:02:52.583Z - GET /posts [200] 74ms
2015-02-18T20:02:52.555Z - GET /posts/45326 [200] 121ms
2015-02-18T20:02:52.528Z - GET /users [200] 105ms
2015-02-18T20:02:52.495Z - GET /posts [200] 96ms
2015-02-18T20:02:52.472Z - GET /posts [200] 77ms
2015-02-18T20:02:52.437Z - GET /login [200] 106ms
2015-02-18T20:02:52.403Z - GET /posts [200] 100ms
2015-02-18T20:02:52.381Z - GET /posts [200] 103ms
2015-02-18T20:02:52.351Z - GET /posts [200] 118ms
2015-02-18T20:02:52.322Z - GET /posts [200] 122ms
```

The CLI stream dashboard documentation is available on GitHub.

Browser stream dashboard

This official Graylog dashboard is showing live information of a specific stream in a web browser. It will display and automatically reload the most recent messages and alerts of a stream and is perfect to display on large screens in your office.


Graylog2 Stream Dashboard
Graylog2 Stream Dashboard
file:///private/tmp/graylog2-stream-dashboard-0.90.0/app/index.html#/messages/52fa6e1f3004c626a7fbcec7
Google
{GRAYLOG2
Select Stream
Settings

»Exceptions« This stream is processing 0 messages per second.

Alert: Stream had 207 messages in the last 5 minutes with trigger condition more than 10 messages. (Current grace time: 1 minutes)

Alert: Field took_ms had a stddev of 1921.4971007085182 in the last 30 minutes with trigger condition higher than 100.0. (Current grace time: 30 minutes)

Timestamp	Source	Message
2014-02-18 21:26:32.340	example.org	GET /posts [500] 126ms
2014-02-18 21:26:32.104	example.org	GET /posts [504] 5100ms
2014-02-18 21:26:31.287	example.org	GET /posts/45326 [500] 111ms
2014-02-18 21:26:30.306	example.org	GET /posts [500] 92ms
2014-02-18 21:26:27.033	example.org	GET /posts/45326 [500] 100ms
2014-02-18 21:26:21.784	example.org	GET /posts [500] 74ms
2014-02-18 21:26:21.762	example.org	GET /posts [500] 92ms
2014-02-18 21:26:20.820	example.org	GET /login [504] 5350ms
2014-02-18 21:26:20.474	example.org	GET /posts [500] 117ms
2014-02-18 21:26:19.951	example.org	GET /users [500] 126ms
2014-02-18 21:26:16.252	example.org	GET /users [500] 86ms
2014-02-18 21:26:15.182	example.org	GET /posts/45326/edit [500] 98ms
2014-02-18 21:26:14.942	example.org	GET /posts [500] 114ms

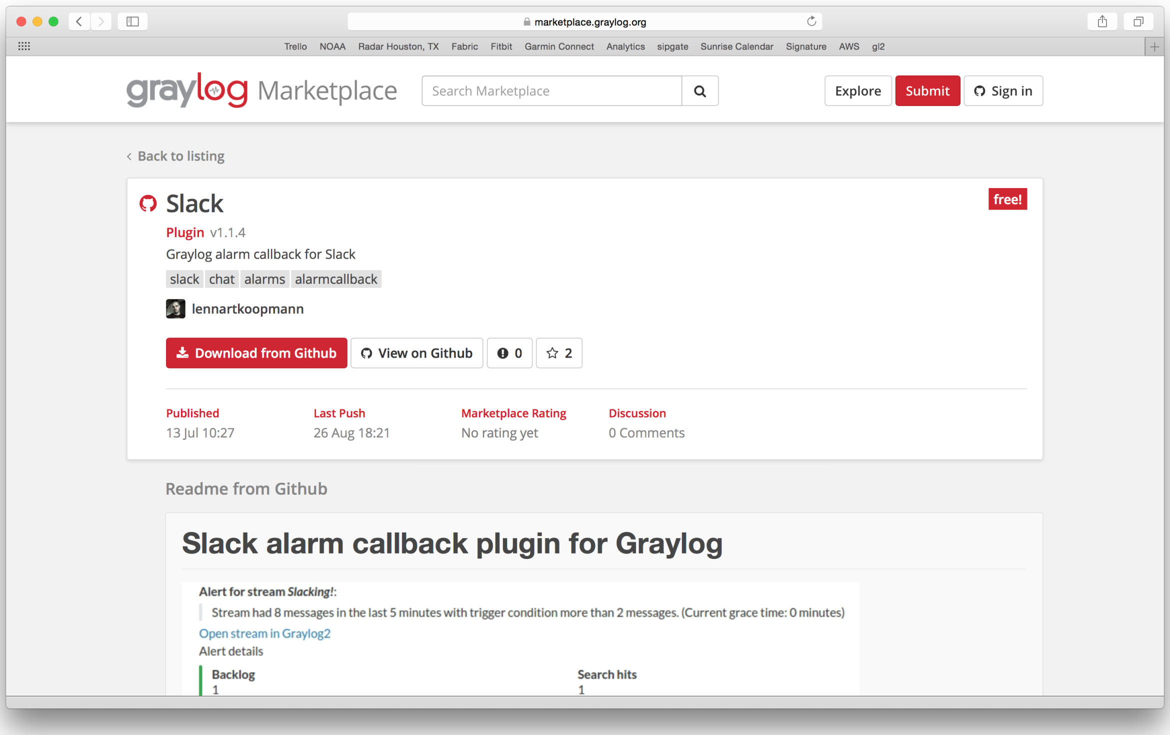
The browser stream dashboard documentation is available on GitHub.

CHAPTER 16

Graylog Marketplace

The Graylog Marketplace is currently in BETA.

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.



Slack

Plugin v1.1.4

Graylog alarm callback for Slack

slack chat alarms alarmcallback

lennartkoopmann

Published	Last Push	Marketplace Rating	Discussion
13 Jul 10:27	26 Aug 18:21	No rating yet	0 Comments

Readme from Github

Slack alarm callback plugin for Graylog

Alert for stream *Slacking!*:

Stream had 8 messages in the last 5 minutes with trigger condition more than 2 messages. (Current grace time: 0 minutes)

Open stream in Graylog2

Alert details

Backlog	Search hits
1	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.

General best practices

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.

License

You cannot submit content that does not contain a `LICENSE` or `COPYING` file. We recommend to consult ChooseALicense.com if you are unsure which license to use.

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

Contributing content packs

Graylog content packs can be shared on the Marketplace by following these steps:

1. Export 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

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`.

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`.

CHAPTER 17

Frequently asked questions

General

Isn't Java slow and needs a lot of memory?

This is a concern that we hear from time to time. We are however usually able to prove this assumption wrong. Java has a bad reputation from slow and laggy desktop/GUI applications that eat a lot of memory. Well written Java code for server systems is very efficient and does not need a lot of resources.

Give it a try, you might be surprised!

I already tried to use Elasticsearch for log management and it did not work well

Sorry to hear that. The good news: Graylog is working around a lot of the log management specific shortcomings of Elasticsearch. Don't get us wrong: Elasticsearch is a great product! It just needs some special handling by our *graylog-server* process that you do not get if you are directly writing your logs and reading information from Elasticsearch. Especially the journalling of Graylog shields Elasticsearch against overloading and failing in weird ways.

What is MongoDB used for?

The MongoDB dependency of Graylog is there to store metadata that is not log data. None of your messages is ever stored in Graylog but for example user information or stream rules are. This is why you should not expect much load on MongoDB and thus don't have to worry too much about scaling it. It will just run aside your *graylog-server* processes and take almost no resources in our recommended setup architectures.

There are plans to introduce a database abstraction layer in the future. This will give you the choice to run MongoDB, MySQL or other databases for storing metadata.

It seems like Graylog has no reporting functionality?

That is correct. For now there is no built-in reporting functionality that sends automated reports. You can however use our REST API to generate and send you own reports. A cron job and the scripting language of your choice should do the trick.

Message parsing

Does Graylog parse syslog?

Yes, 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.

CHAPTER 18

The thinking behind the Graylog architecture and why it matters to you

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.

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.

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.

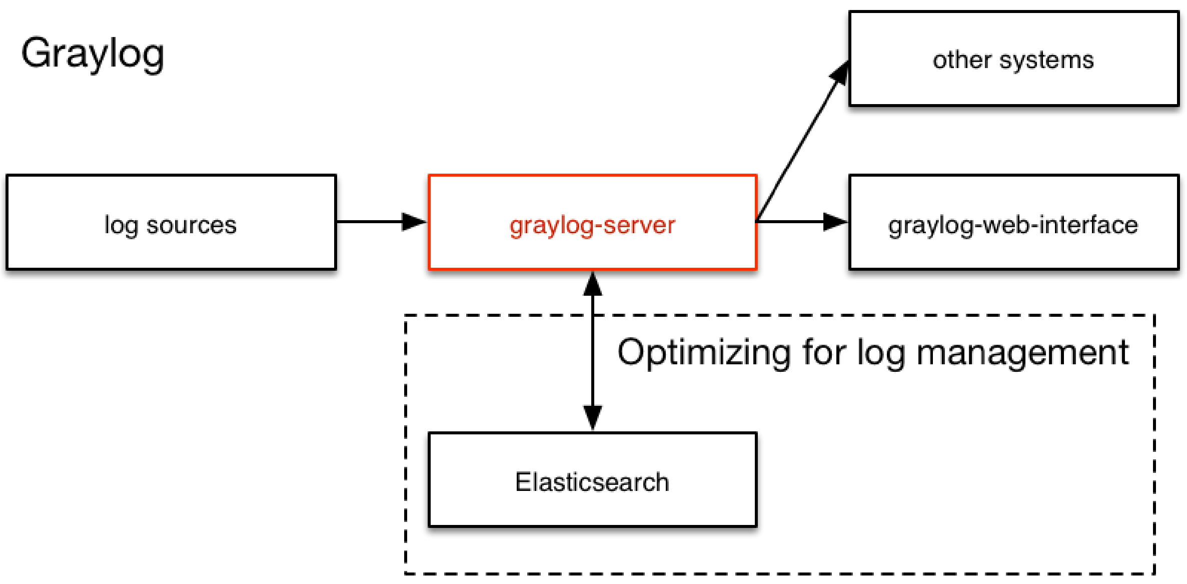
A log management system has to be constructed of several services that take care of processing, indexing, and data access. The most important reason is that you need to scale parts of it horizontally with your changing use cases and usually the different parts of the system have different hardware requirements. 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

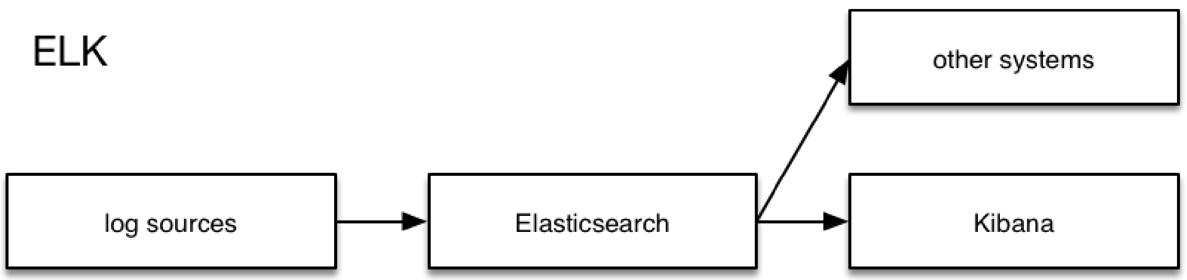
REST APIs controlled by a simple interface. A system adminstrator needs to be able to log into the web interface of a log management product and select log files of a remote host (that has a forwarder running) for ingestion into the tool.

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.





Unlimited data collection

Volume based license models are making your job unnecessary hard. Price is a big factor here but it is even worse that volume based license models make you (or your manager makes you) try to save volume. This means that you will be finding yourself thinking about which data really needs to be ingested. The big problem is that you do not know what you might need the data for in the moment you are sending (or not sending) it. We have seen operations teams during a downtime wishing that they had collected the data of a certain log file that was now not searchable. **This is counter-productive and dangerous. You can be limited by disk space or other resources but never by the license that somebody bought.**

It is also a law of the market that you have to build your volume pricing model on the amount of data that is usually collected **today**. The amount of generated data has increased dramatically and vendors are nailed to their pricing model from 2008. This is why you get quotes that fill you with sadness in today's world.

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.

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.

CHAPTER 19

Changelog

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

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

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

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

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, dispite 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
- "&lt;&gt;" 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 &amp; 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, dispite 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

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

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

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

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

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

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

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)

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)

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

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.

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

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

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

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

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

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)

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.

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)

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).

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

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

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

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

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

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

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

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

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

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

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.