
Graylog Documentation

Release 2.0.0

Graylog, Inc.

Apr 04, 2018

1	Architectural considerations	3
1.1	Minimum setup	3
1.2	Bigger production setup	4
1.3	Graylog Architecture Deep Dive	5
2	Getting Started	7
2.1	Install & Configure	7
2.2	Import the VM	8
2.3	Start VM	8
2.4	Connect to the Web Console	9
2.5	Configure Graylog Input	11
2.6	Get Messages In	13
2.7	Check If You Have Messages	14
2.8	If You Don't Have Messages	15
2.9	Create Your Dashboard	15
2.10	Get Alerted	19
3	Installing Graylog	25
3.1	Virtual Machine Appliances	25
3.2	Operating System Packages	29
3.3	Chef, Puppet, Ansible, Vagrant	37
3.4	Docker	37
3.5	Vagrant	41
3.6	OpenStack	42
3.7	Amazon Web Services	43
3.8	Microsoft Windows	44
3.9	Manual Setup	44
3.10	System requirements	49
4	Upgrading Graylog	51
4.1	From 1.x to 2.x	51
5	Configuring Graylog	57
5.1	The graylog-ctl script	57
5.2	Web interface	65
5.3	Load balancer integration	69
5.4	Using HTTPS	70
5.5	Multi-node Setup	76

5.6	Elasticsearch	79
5.7	Backup	88
5.8	Default file locations	88
5.9	Graylog REST API	91
6	Securing Graylog	97
6.1	Logging user activity	97
7	Sending in log data	99
7.1	What are Graylog message inputs?	99
7.2	Content packs	99
7.3	Syslog	99
7.4	GELF / Sending from applications	100
7.5	Using Apache Kafka as transport queue	100
7.6	Using RabbitMQ (AMQP) as transport queue	101
7.7	Microsoft Windows	101
7.8	Heroku	101
7.9	Ruby on Rails	103
7.10	Raw/Plaintext inputs	104
7.11	JSON path from HTTP API input	104
7.12	Reading from files	105
8	Graylog Collector Sidecar	107
8.1	Installation	107
8.2	Configuration	108
8.3	Use the Graylog web interface to configure remote collectors	109
8.4	Outputs, Inputs and Snippets	110
8.5	Debug	111
9	Graylog Collector (deprecated)	113
9.1	Installation	113
9.2	Configuration	117
9.3	Running Graylog Collector	124
9.4	Command Line Options	126
9.5	Troubleshooting	127
10	Searching	129
10.1	Search query language	129
10.2	Time frame selector	131
10.3	Saved searches	132
10.4	Analysis	133
10.5	Export results as CSV	135
10.6	Search result highlighting	136
10.7	Search configuration	136
11	Streams	141
11.1	What are streams?	141
11.2	Alerts	142
11.3	Outputs	147
11.4	Use cases	148
11.5	How are streams processed internally?	148
11.6	Stream Processing Runtime Limits	149
11.7	Programmatic access via the REST API	150
11.8	FAQs	152

12	Dashboards	155
12.1	Why dashboards matter	155
12.2	How to use dashboards	156
12.3	Examples	158
12.4	Widgets from streams	159
12.5	Result	159
12.6	Widget types explained	159
12.7	Modifying dashboards	160
12.8	Dashboard permissions	162
13	Extractors	165
13.1	The problem explained	165
13.2	Graylog extractors explained	165
13.3	Import extractors	166
13.4	Using regular expressions to extract data	167
13.5	Using Grok patterns to extract data	168
13.6	Using the JSON extractor	169
13.7	Automatically extract all key=value pairs	169
13.8	Normalization	171
14	Processing Pipelines	175
14.1	Pipelines	175
14.2	Rules	176
14.3	Stream connections	178
14.4	Functions	179
15	Message rewriting with Drools	189
15.1	Getting Started	189
15.2	Example rules file	189
15.3	Parsing Message and adding fields	190
15.4	Blacklisting messages	191
16	Blacklisting	193
16.1	How to	193
16.2	Based on regular expressions	193
17	Geolocation	195
17.1	Setup	195
17.2	Visualize geolocations in a map	201
17.3	FAQs	204
18	The Graylog index model explained	207
18.1	Overview	207
18.2	Eviction of indices and messages	208
18.3	Keeping the metadata in synchronisation	209
18.4	Manually cycling the deflector	210
19	Indexer failures	211
19.1	Common indexer failure reasons	211
20	Users and Roles	213
20.1	Users	213
20.2	Roles	215
20.3	System users	218
20.4	External authentication	221

21	Plugins	225
21.1	General information	225
21.2	Prerequisites	225
21.3	Creating a plugin skeleton	225
21.4	Example Alarm Callback plugin	226
21.5	Creating a plugin for the web interface	227
21.6	Best practices for web plugin development	229
21.7	Building plugins	229
21.8	Installing and loading plugins	229
22	External dashboards	231
22.1	CLI stream dashboard	231
22.2	Browser stream dashboard	232
23	Graylog Marketplace	233
23.1	GitHub integration	233
23.2	General best practices	234
23.3	4 Types of Add-Ons	234
23.4	Contributing plug-ins	234
23.5	Contributing content packs	235
23.6	Contributing GELF libraries	235
23.7	Contributing other content	235
24	Frequently asked questions	237
24.1	General	237
24.2	Architecture	237
24.3	Installation / Setup	238
24.4	Functionality	239
24.5	Graylog & Integrations	240
24.6	Troubleshooting	241
24.7	Support	242
25	GELF	243
25.1	Structured events from anywhere. Compressed and chunked.	243
25.2	GELF via UDP	243
25.3	GELF via TCP	244
25.4	GELF Payload Specification	244
25.5	Example payload	245
26	The thinking behind the Graylog architecture and why it matters to you	247
26.1	A short history of Graylog	247
26.2	The log management market today	247
26.3	The future	249
27	Changelog	251
27.1	Graylog 2.0.3	251
27.2	Graylog 2.0.2	251
27.3	Graylog 2.0.1	252
27.4	Graylog 2.0.0	253
27.5	Graylog 1.3.4	257
27.6	Graylog 1.3.3	257
27.7	Graylog 1.3.2	258
27.8	Graylog 1.3.1	258
27.9	Graylog 1.3.0	258
27.10	Graylog 1.2.2	259

27.11	Graylog 1.2.1	259
27.12	Graylog 1.2.0	260
27.13	Graylog 1.2.0-rc.4	260
27.14	Graylog 1.2.0-rc.2	261
27.15	Graylog 1.1.6	262
27.16	Graylog 1.1.5	262
27.17	Graylog 1.1.4	263
27.18	Graylog 1.1.3	263
27.19	Graylog 1.1.2	263
27.20	Graylog 1.1.1	264
27.21	Graylog 1.1.0	265
27.22	Graylog 1.1.0-rc.3	265
27.23	Graylog 1.1.0-rc.1	265
27.24	Graylog 1.1.0-beta.3	266
27.25	Graylog 1.1.0-beta.2	267
27.26	Graylog 1.0.2	268
27.27	Graylog 1.0.1	268
27.28	Graylog 1.0.0	269
27.29	Graylog 1.0.0-rc.4	269
27.30	Graylog 1.0.0-rc.3	270
27.31	Graylog 1.0.0-rc.2	270
27.32	Graylog 1.0.0-rc.1	270
27.33	Graylog 1.0.0-beta.2	272
27.34	Graylog 1.0.0-beta.2	273
27.35	Graylog2 0.92.4	273
27.36	Graylog 1.0.0-beta.1	273
27.37	Graylog2 0.92.3	274
27.38	Graylog2 0.92.1	274
27.39	Graylog2 0.92.0	275
27.40	Graylog2 0.92.0-rc.1	275
27.41	Graylog2 0.91.3	276
27.42	Graylog2 0.91.3	276
27.43	Graylog2 0.92.0-beta.1	276
27.44	Graylog2 0.91.1	277
27.45	Graylog2 0.90.1	277
27.46	Graylog2 0.91.0-rc.1	277
27.47	Graylog2 0.90.0	278
27.48	Graylog2 0.20.3	278
27.49	Graylog2 0.20.2	279
28	Introduction	281
29	Setup	283
29.1	Requirements	283
29.2	Installation	283
29.3	Server Restart	284
29.4	License Installation	285
30	Archiving	287
30.1	Setup	287
30.2	Usage	291
31	Changelog	297
31.1	Graylog Enterprise 1.0.1	297
31.2	Graylog Enterprise 1.0.0	297

NOTE: There are multiple options for reading this documentation. See [link](#) to the lower left.

Architectural considerations

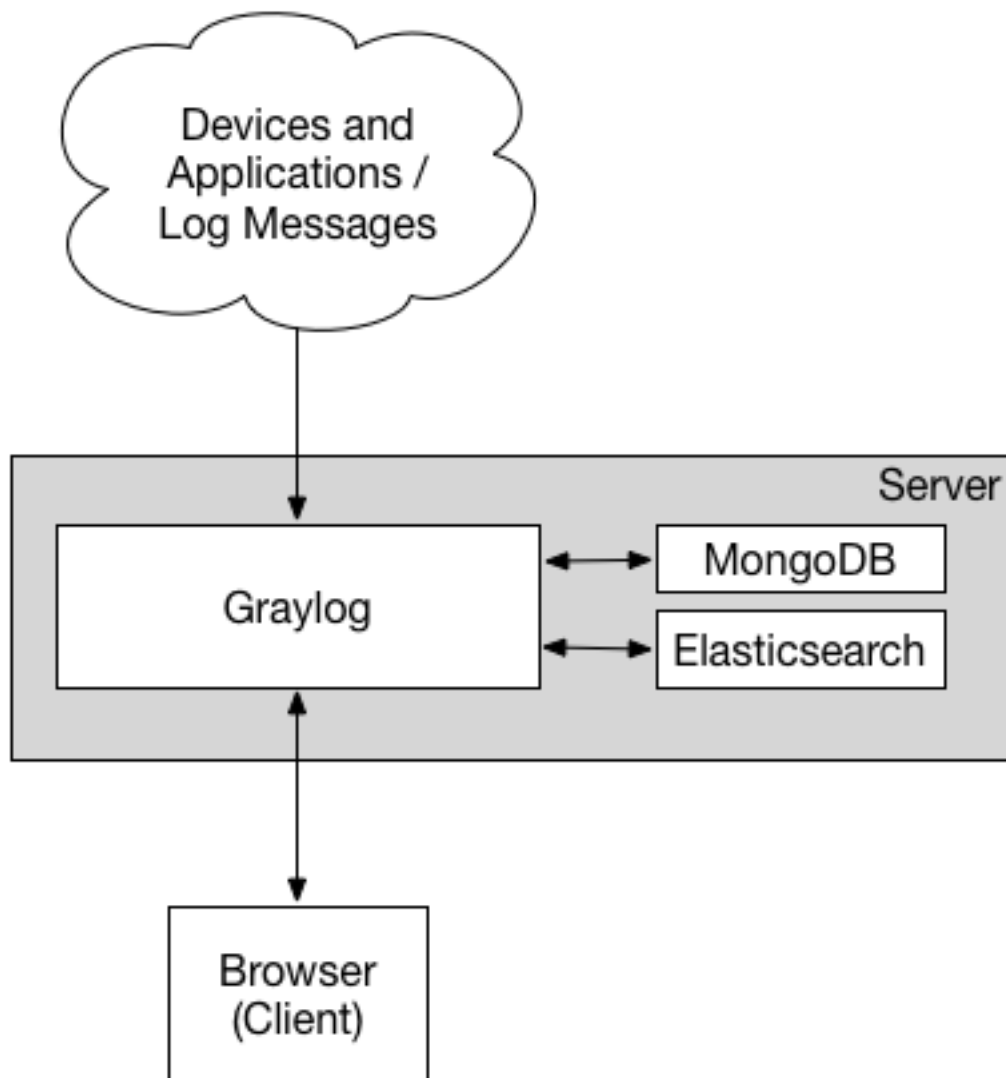
There are a few rules of thumb when scaling resources for Graylog:

- Graylog nodes should have a focus on CPU power. These also serve the user interface to the browser.
- Elasticsearch nodes should have as much RAM as possible and the fastest disks you can get. Everything depends on I/O speed here.
- MongoDB is storing meta information and configuration data and doesn't need many resources.

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

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.

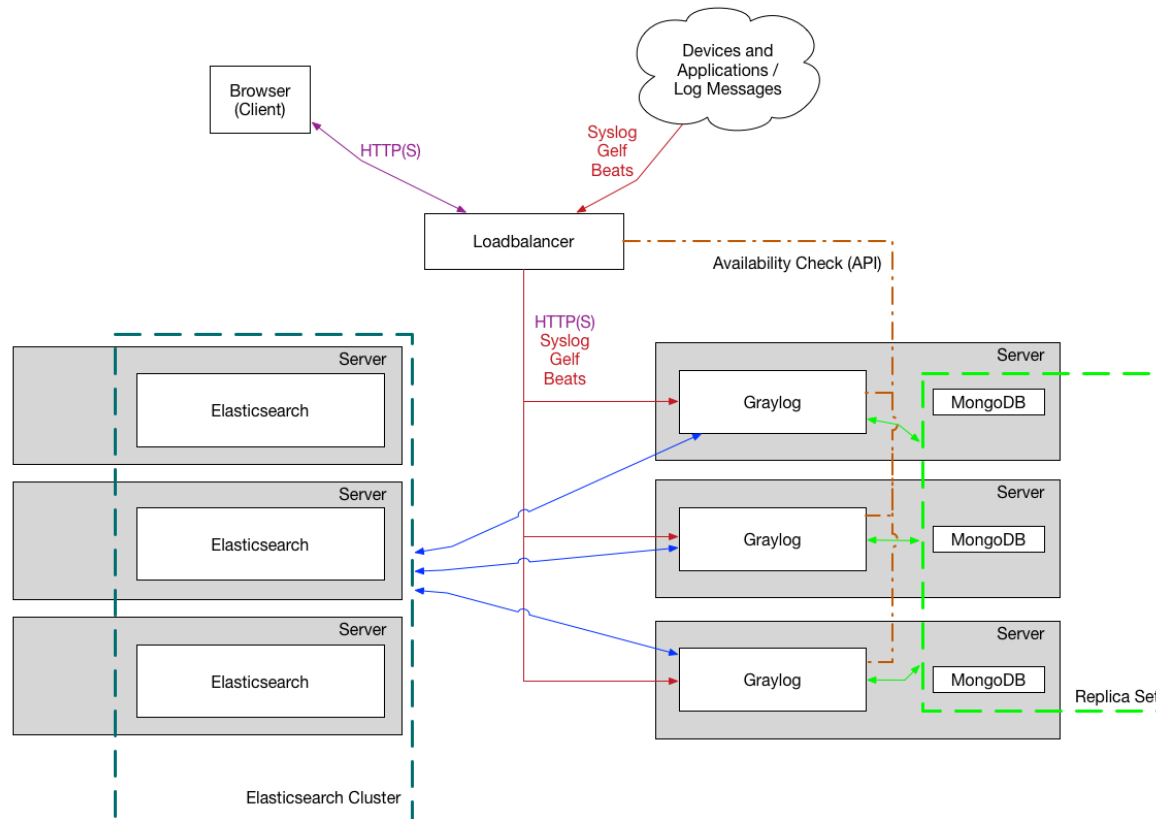


Our *Virtual Machine Appliances* are using this design by default, deploying nginx as *frontend proxy*.

Bigger production setup

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

The load balancer can ping the Graylog nodes via HTTP on the Graylog REST API to check if they are alive and take dead nodes out of the cluster.



How to plan and configure such a setup is covered in our [Multi-node Setup guide](#).

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

Graylog Architecture Deep Dive

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

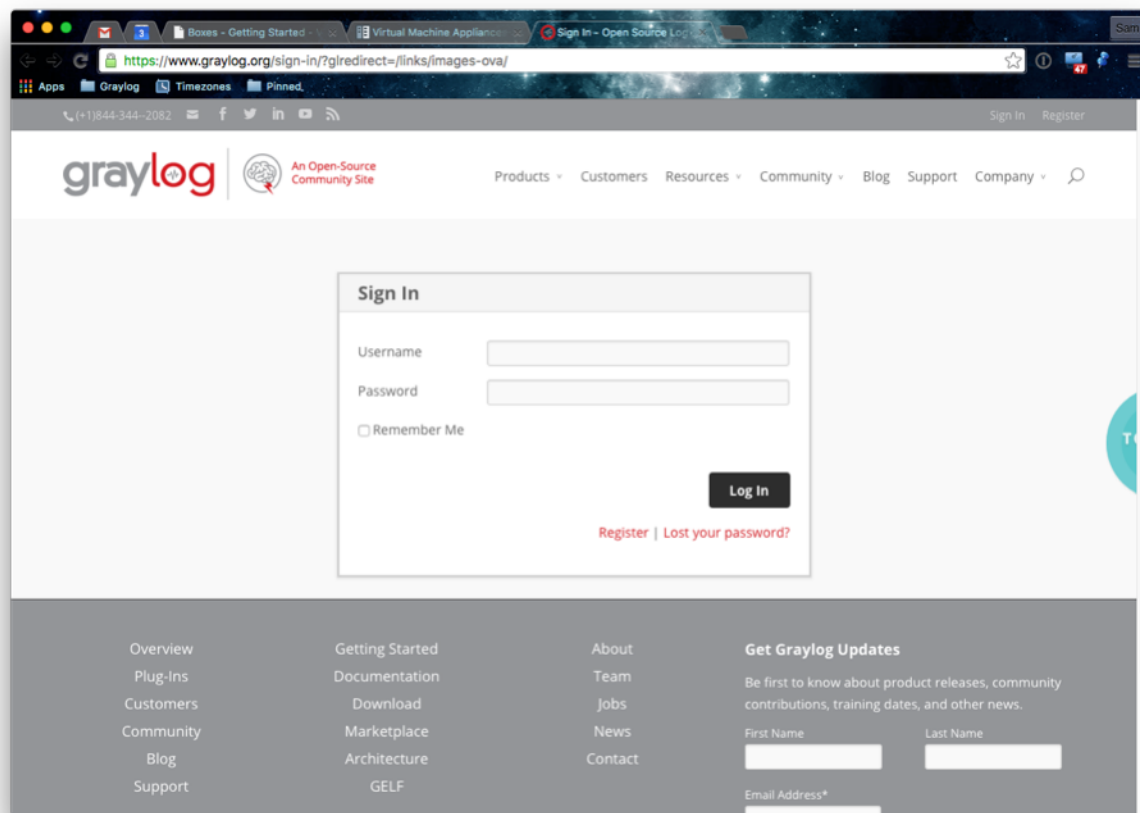
Getting Started

The fastest way to evaluate Graylog is to download the virtual appliance file and run it in VMWare or VirtualBox.

Install & Configure

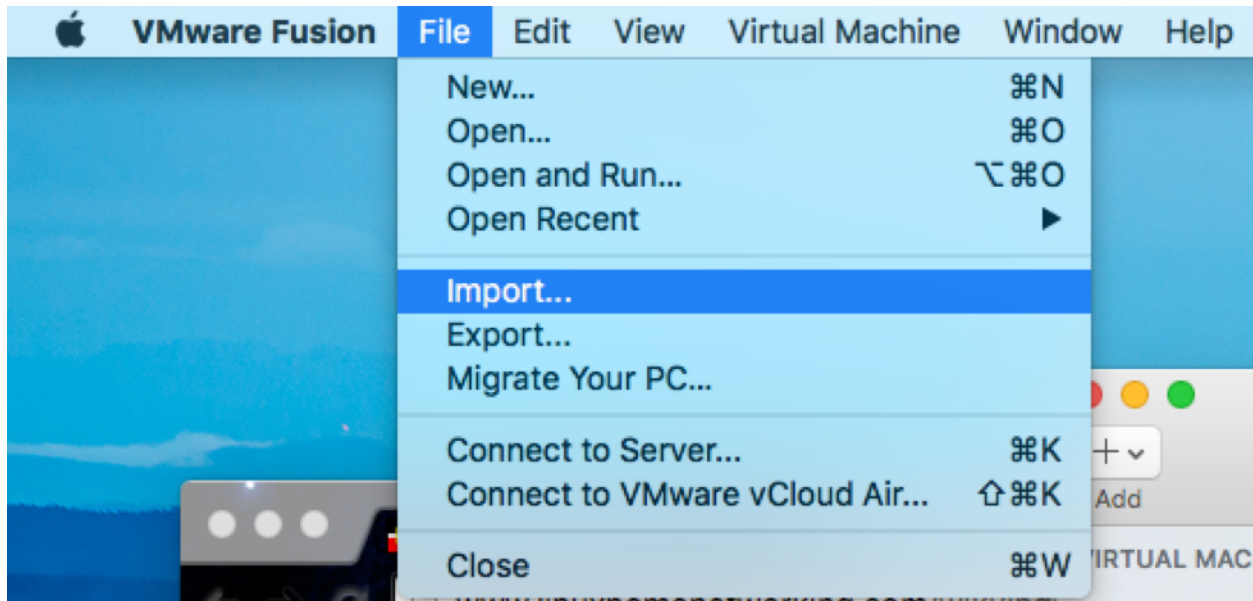
Download Graylog

Go [here](#) and download the virtual image.



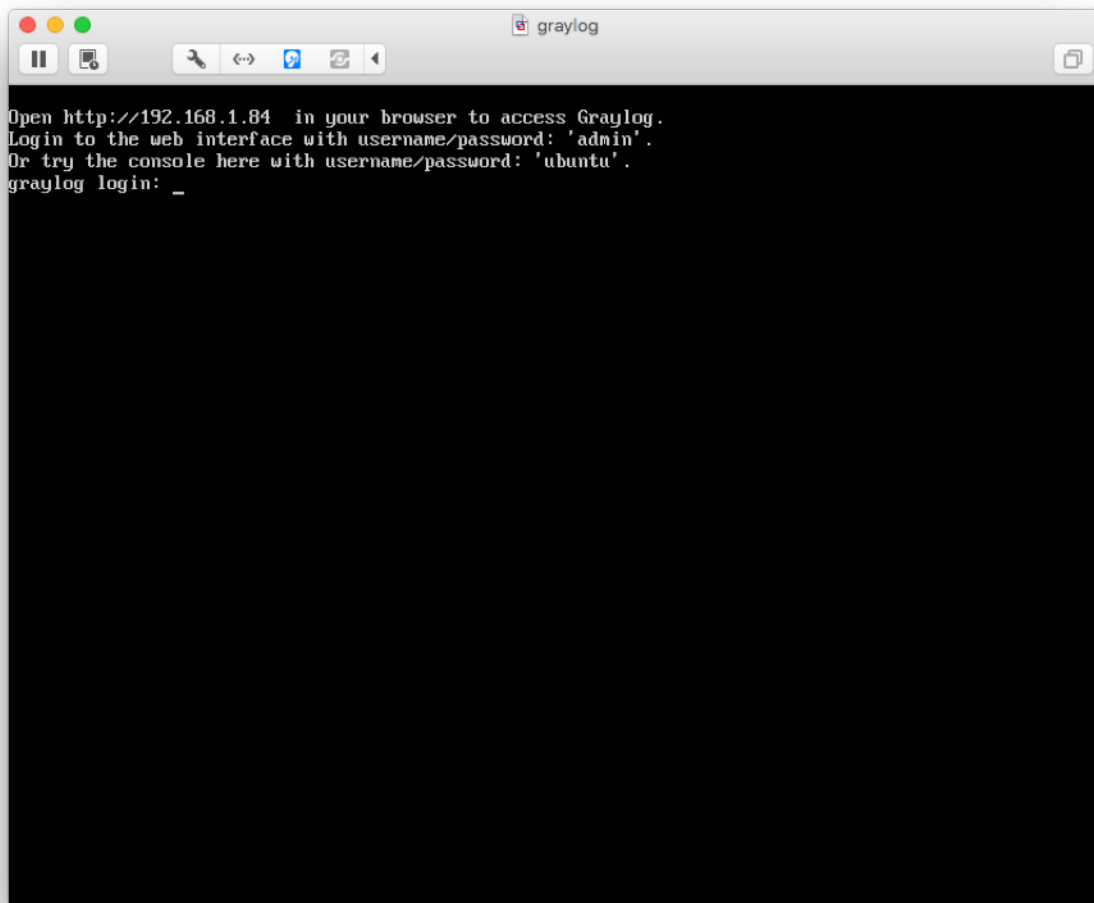
Import the VM

These screenshots are for VMWare (VirtualBox is nearly the same). Select *File -> Import...*, choose the `graylog.ova` file you downloaded, and follow the prompts.



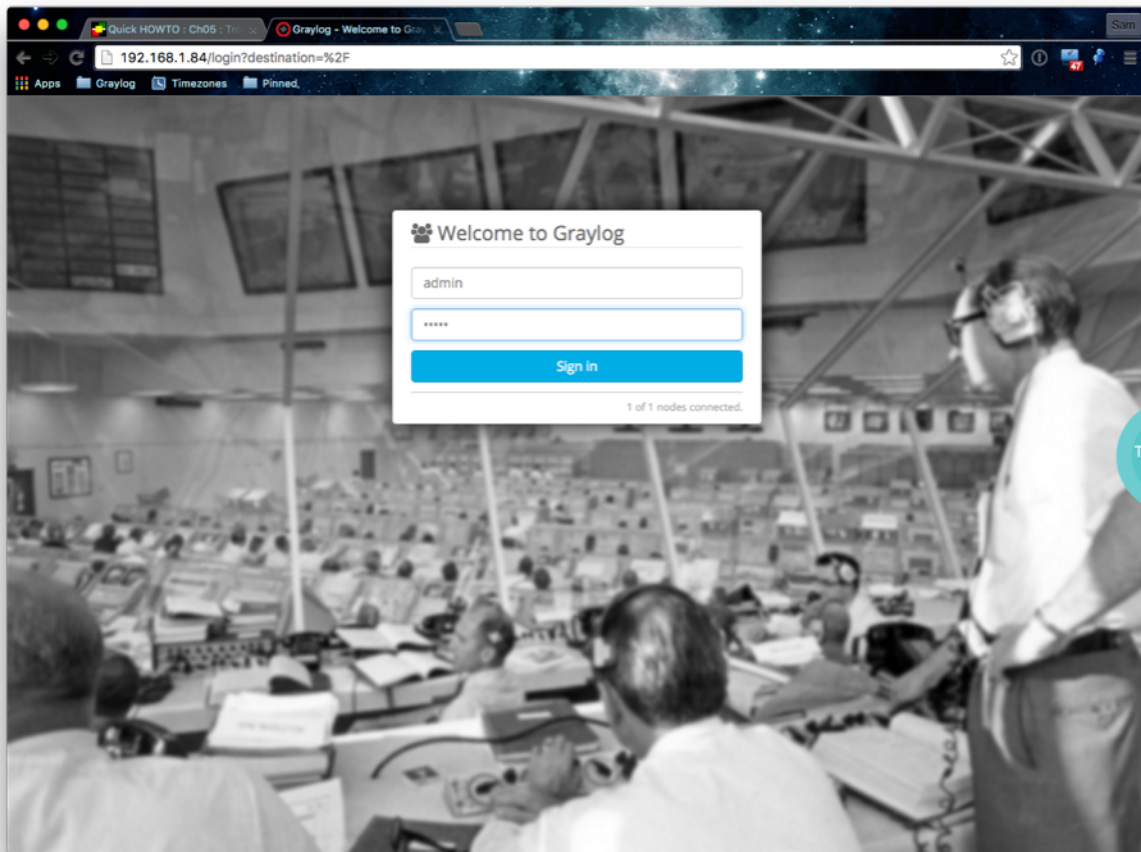
Start VM

This is what you'll see when you run the virtual machine. This is where all the Graylog server processes (more on that later) and the database will run. We can split them apart later for performance, but there's no need to do that right now for a quick overview. Don't close this window just yet, we're going to need the IP for the next step.

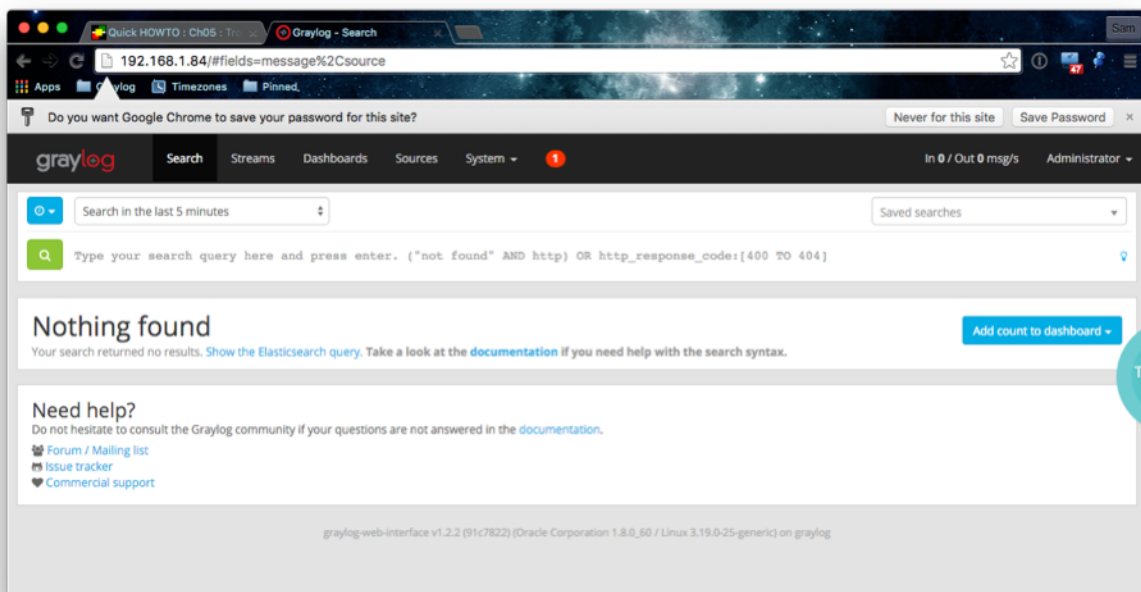


Connect to the Web Console

Go to a web browser on your host machine and type the IP address you saw in the previous screen. You should get a Graylog Web Console login page. Enter `admin/admin` to login.



Logging in will get you to blank screen. Not very fun? No problem, let's get some stuff in there!



Configure Graylog Input

Now you are not sending data to Graylog, so you need to configure an input. This will tell Graylog to accept the log messages.

Go back to the Graylog console open in your browser and click *System -> Inputs*. Then select Syslog UDP and click *Launch new input*. Fill out the circles with the values in the screen shown below.

Launch new input: *Syslog UDP*

Node(s) to spawn input on:
Select the node you want to spawn this input on.

ac472930 / graylog

or:
☐ Global input (started on all nodes)

Title
Select a name of your new input that describes it.

Syslog UDP

Bind address
Address to listen on. For example 0.0.0.0 or 127.0.0.1.

127.0.0.1

Port
Port to listen on.

5140

Receive Buffer Size (optional)

262144

The size in bytes of the recvBufferSize for network connections to this input.

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.

☐ **Force rDNS?** (optional)
Force rDNS resolution of hostname? Use if hostname cannot be parsed.

☒ **Allow overriding date?** (optional)
Allow to override with current date if date could not be parsed?

☐ **Store full message?** (optional)
Store the full original syslog message as full_message?

☐ **Expand structured data?** (optional)
Expand structured data elements by prefixing attributes with their SD-ID?

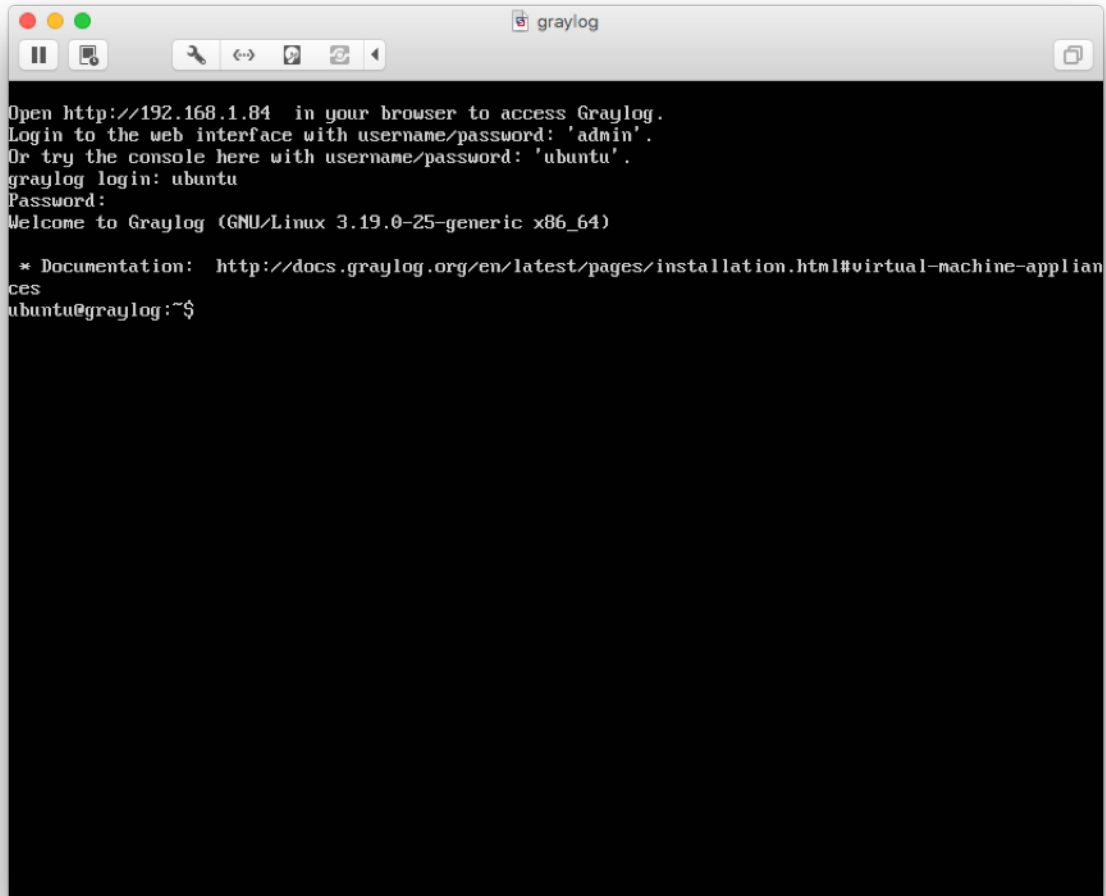
Close

Launch

Get Messages In

Log into the VM

We're going to use `rsyslog` because we already have it from the Graylog server image. So, go to the image and login with `ubuntu/ubuntu`.



Modify `rsyslog.conf`

Go to the `/etc` directory, and use `vi`, `vim` ([vim Cheat Sheet](#)), or the editor of your choice to modify the `/etc/rsyslog.conf` file. There are excellent resources on the web for [rsyslog configuration](#).

At the bottom of the file, add the following so messages will forward:

```
*. * @127.0.0.1:5140;RSYSLOG_SyslogProtocol23Format
```

In case you wanted to know, `@` means UDP, `127.0.0.1` is localhost, and `5140` is the port.

You can find out more about ingesting syslog messages with Graylog in our [Syslog configuration guide](#).

Restart rsyslog

Type:

```
$sudo service rsyslog status
$sudo service rsyslog restart
```

If you have modified the config file and it is somehow invalid, the service command will not bring rsyslog back up - so don't worry, you can always delete the line!

Check If You Have Messages

After that, you should see the Syslog UDP input appear on the screen.

The screenshot shows the 'Local inputs' section with 'Syslog UDP (Syslog UDP)' listed as a running input. Below the input name, there is a code block showing configuration details:

```
override_source:
recv_buffer_size: 262144
allow_override_date: true
bind_address: 127.0.0.1
port: 5140
```

To the right of the configuration, there are buttons for 'Show received messages', 'Manage extractors', 'Stop input', and 'More actions'. Further right, a 'Throughput / Metrics' section shows '1 minute average rate: 0 msg/s' and 'Network IO: ~0B ~0B (total: ~5.7KB ~0B)'.

Click *Show received messages* button on this screen, and you should have messages at the bottom. It may take a few minutes before you have messages coming in.

The screenshot shows the Graylog search results page. The top navigation bar includes 'graylog', 'Search', 'Streams', 'Dashboards', 'Sources', and 'System'. The search bar shows 'Search in the last 8 hours' and 'Saved searches'. Below the search bar, the search criteria is 'g12_source_input:563bec4be4b09abf23ab88b0'. The search result shows 'Found 64 messages in 61 ms, searched in 1 index.' with buttons for 'Add count to dashboard', 'Save search criteria', and 'More actions'. The 'Fields' section shows checkboxes for 'facility', 'level', 'message', and 'source'. The 'Histogram' section shows a bar chart with a peak at 18:00. The 'Messages' section shows a table of messages:

Timestamp [T]	source
2015-11-06 18:35:33.000	graylog
graylog ntpd_intres[891]: host name not found: 3.pool.ntp.org	
2015-11-06 18:35:33.000	graylog
graylog ntpd_intres[891]: host name not found: 1.pool.ntp.org	
2015-11-06 18:35:33.000	graylog
graylog ntpd_intres[891]: host name not found: 2.pool.ntp.org	
2015-11-06 18:35:33.000	graylog
graylog ntpd_intres[891]: host name not found: 0.pool.ntp.org	

BOOM! Now that you have messages coming in, this is where the fun starts.

Skip the next section if you are all good.

If You Don't Have Messages

1. Check to see that you made the proper entries in the rsyslog configuration file.
2. Check the syslog UDP configuration and make sure that is right - remember we changed the default port to 5140.

3. Check to see if rsyslog messages are being forwarded to the port. You can use the `tcpdump` command to do this:

```
$ sudo tcpdump -i lo host 127.0.0.1 and udp port 5140
```

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

```
$ sudo netstat -peanut | grep ":5140"
```

Create Your Dashboard

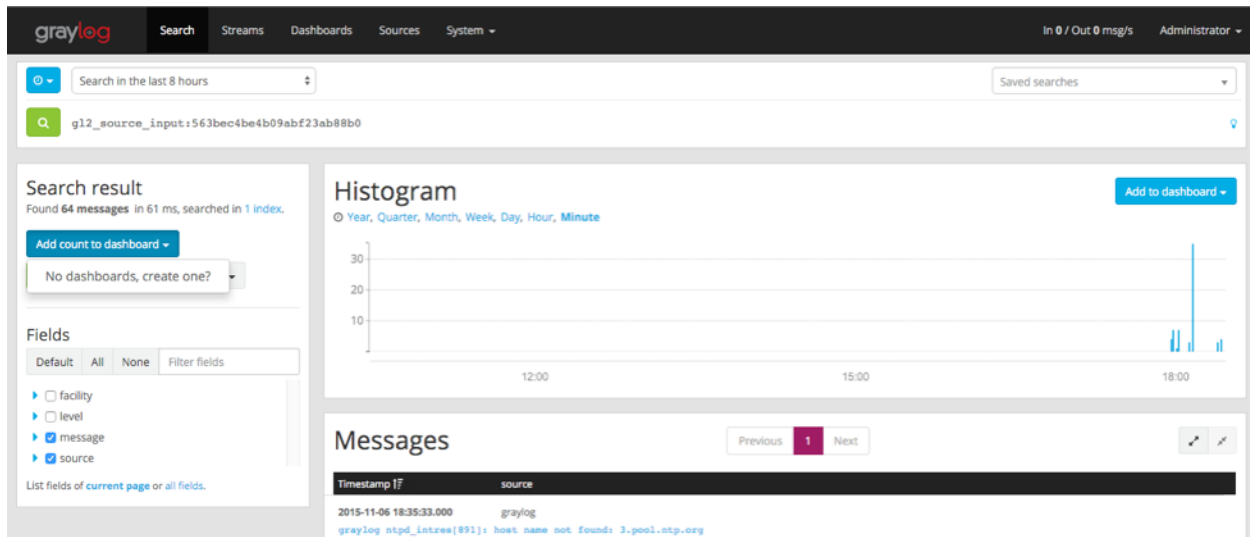
You should be at a screen like the one below. If you dozed off or went to cook some meatballs, go to System -> Inputs, select the Syslog UDP input you created, and hit Show messages.

Now it's go-time.

You've got data coming in, let's add information to a dashboard to better visualize the data we want to see.

Add a Dashboard

We'll start by adding the message count data to a dashboard. Click *Add count to dashboard*, and it will say *No Dashboards, create one? Yes! Click that.*



Give your new dashboard a title and description.

New Dashboard ×

Title:

Description:

Cancel Save

Add a Dashboard Widget

Now it will let you create a widget. In this case, we are creating a widget from our search result of message count in the last 8 hours. I like to put a timeframe in the title, and trends are always a big bowl of sunshine.

Create Dashboard Widget ×

Title

Type a name that describes your widget.

☒ Display trend

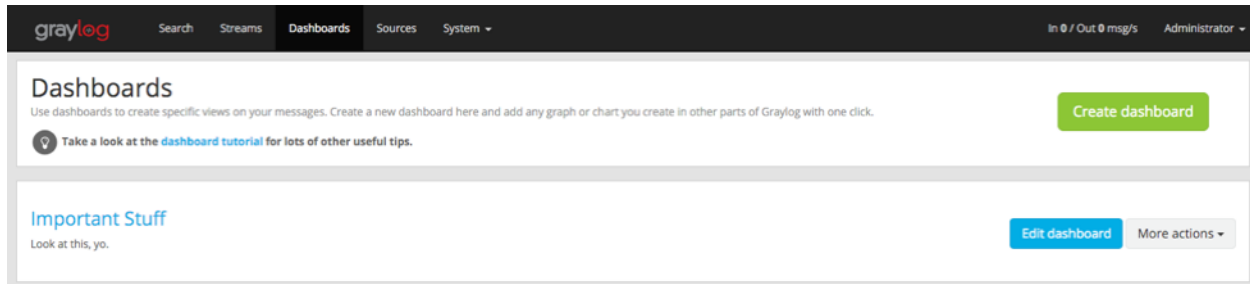
Show trend information for this number.

☒ Lower is better

Use green colour when trend goes down.

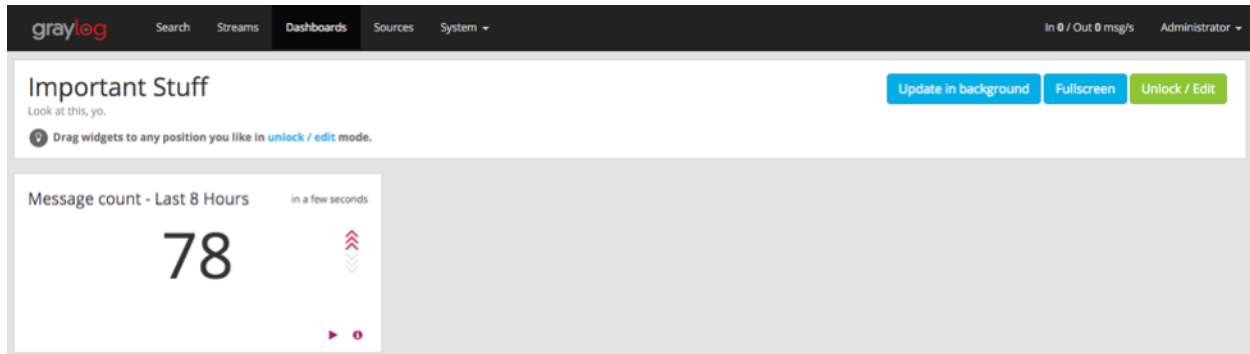
Cancel Create

When you hit create - *wa la!* Nothing happens. All you UX types, relax, we know. For now, click Dashboards and then the name of your dashboard.



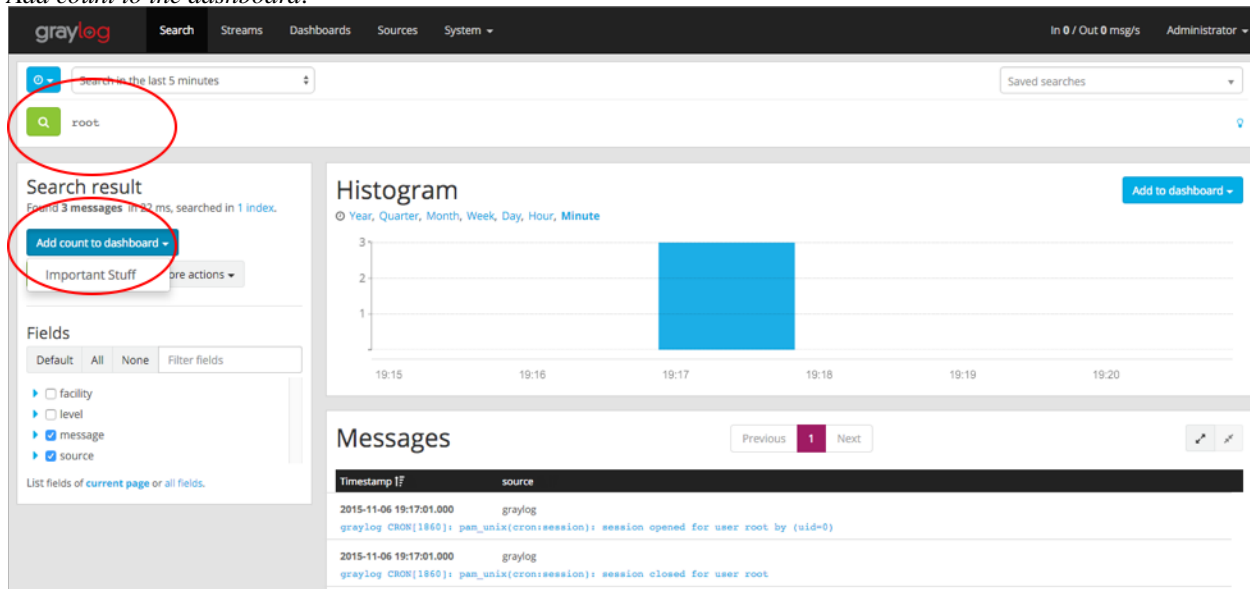
Smile

And you'll end up with the widget you created!



Extra Credit - One more

Let's add a widget for root activity, because that sounds like it may actually be useful. We need to start with a search query for root. Click *Search*. Type root in the search and select your timeframe. Once the search results come in, click *Add count to the dashboard*.



Give your chart a title and hit *Create*.

Create Dashboard Widget

Title

Root Activity - Last 5 Minutes

Type a name that describes your widget.

☐ Display trend

Show trend information for this number.

☐ Lower is better

Use green colour when trend goes down.

Cancel

Create

The new widget is now on the screen. Good job - you've got this!

graylog

Search Streams Dashboards Sources System

In 0 / Out 0 msg/s Administrator

Important Stuff

Look at this, yo.

Drag widgets to any position you like in [unlock / edit mode](#).

Message count - Last 8 Hours

in a few seconds

93

Root Activity - Last 5 Minutes

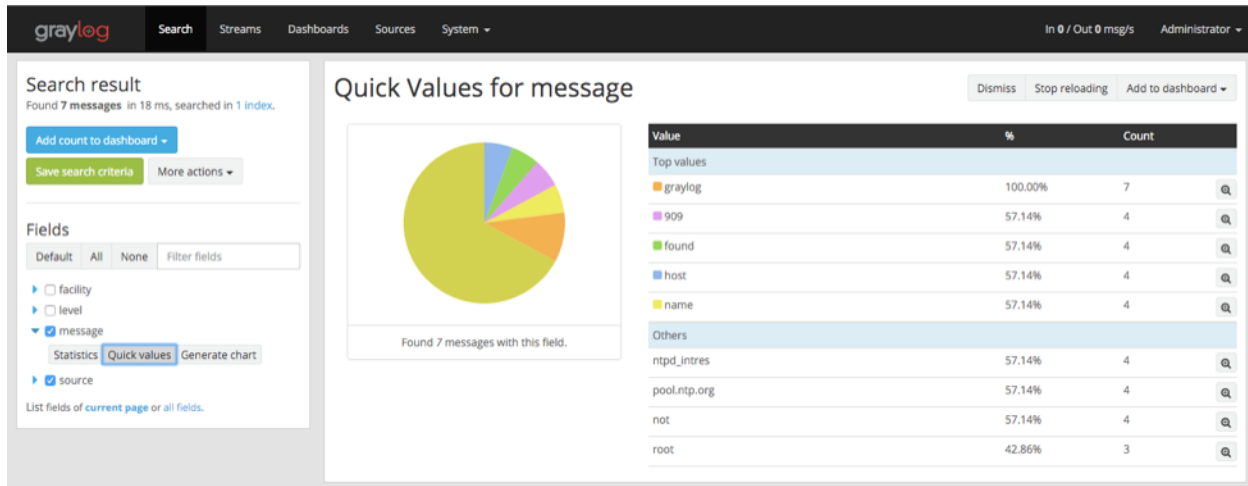
in a few seconds

0

Go play around. If you want to know how to create more exciting charts and graphs, check out the section below.

Extra Credit - Graphs

Let's start by searching for all messages within the last 1 hour. To do this, click *Search*, select Search in the last 1 hour, and leave the search bar blank. Once the search results populate, expand the messages field in the Search results sidebar and select *Quick Values*. Click *Add to dashboard* to add this entire pie chart and data table to your dashboard.



I like to track password changes, privilege assignments, root activity, system events, user logins, etc. Go knock yourself out and show your co-workers.

Once you have a few widgets in your dashboard, go into unlock / edit mode to quickly edit any widget, rearrange them on your dashboard, or delete. Make sure to click Lock to save!

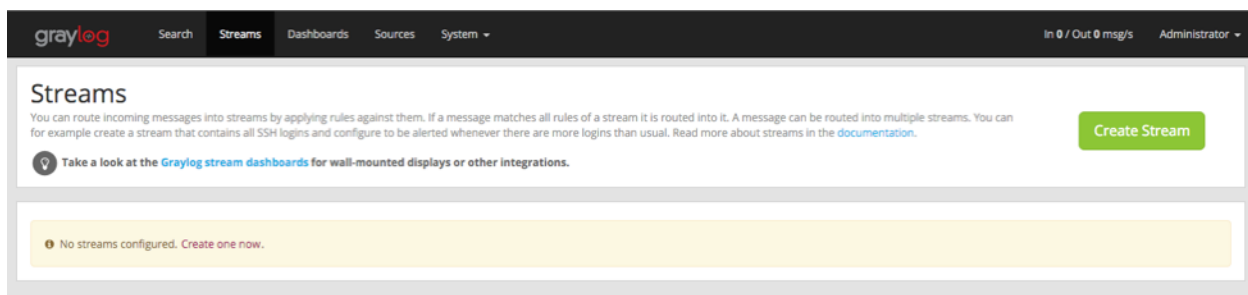
Get Alerted

I know, we're all lazy, and busy. Nobody wants to just stare at a dashboard all day like it's the World Cup. That's for management.

Let's configure some proactive alerts to let us know when something needs our attention.

Create a Stream

In order to set up an alert, we need to first create a stream. Streams process incoming messages in real time based on conditions that you set. Click *Streams*.



Let's create a stream for all incoming security/authentication error messages. Click Create Stream.

Type in a Title and Description.

Creating Stream ×

Title

Security/Auth Errors from Syslogs

Description

Security

Cancel

Save

Create a Stream Rule

Next, we are going to configure the stream to process our Syslog UDP input messages for any security alerts.

Hit the *Edit rules* button.

Security/Auth Errors from Syslogs stopped

Security

0 messages/second, No configured rules. [Show stream rules](#)

Edit rules

Manage outputs

Manage alerts

Start stream

More actions ▾

Pick the Syslog UDP Input, and click Add stream rule.

Rules of stream »Security/Auth Errors from Syslogs«

This screen is dedicated to an easy and comfortable creation and manipulation of stream rules. You can see the effect configured stream rules have on message matching here.

1. Load a message to test rules

Recent

Manual

Select an Input from the list below and click "Load Message" to load the most recent message from this input.

Syslog UDP (org.graylog2.inputs.syslog.udp.SyslogUDPin ▾)

Load Message

2. Manage stream rules

Please load a message to check if it would match against these rules and therefore be routed into this stream.

☒ A message must match all of the following rules

☐ A message must match at least one of the following rules

No rules defined.

I'm done!

Add stream rule

Then, type in the values shown below and hit save.

New Stream Rule

Field

Type

Value

☐ Inverted

Result: Field *facility* must match exactly *security/authorization*

The server will try to convert to strings or numbers based on the matcher type as good as it can.

[Take a look at the matcher code on GitHub](#)

Regular expressions use Java syntax. 💡

Cancel Save

Then click I'm done!

We have just configured this stream to process in real time all the messages that come in from the security/authorization facility.

Now let's create the alert.

Create the Alert

You can now either output your new stream to a 3rd party application or database, or trigger an alert to ping you in real time when a message that matches our stream rule comes in. Let's create an alert that will email us when there are more than 2 messages in the last 2 minutes . Click *Manage Alerts*.

Security/Auth Errors from Syslogs stopped

Security
 0 messages/second, Must match all of the 1 configured stream rule(s). [Show stream rules](#)

Edit rules Manage outputs Manage alerts Start stream More actions ▾

In the Add new alert condition section, let's configure and add a new alert. Select message count condition, and configure the rest based on my screenshot (input 2's in every field). Then click Add alert condition.

The screenshot shows the Graylog web interface for configuring alerts on a specific stream. The top navigation bar includes 'graylog', 'Search', 'Streams', 'Dashboards', 'Sources', and 'System'. The user is logged in as 'Administrator' with 'In 0 / Out 0 msg/s'.

Alerts configuration for stream »Security/Auth Errors from Syslogs«

You can define thresholds on any message field or message count of a stream and be alerted based on this definition.

[Learn more about alerts in the documentation.](#)

Add new alert condition

Message count condition [Configure new alert condition](#)

Trigger alert when there are ☒ more ☐ less than messages in the last minutes and then wait at least minutes until triggering a new alert. (grace period)

When sending an alert, include the last messages of the stream evaluated for this alert condition.

[Add alert condition](#)

Configured alert conditions

No configured alarm conditions.

Callbacks

The following callbacks will be performed when this stream triggers an alert.

Select Callback Type [Add callback](#) [Find more callbacks](#)

No configured alarm callbacks.

Receivers

The following Graylog users will be notified about alerts via email if they have configured an email address in their profile. You can also add any other email address to the alert receivers if it has no Graylog user associated.

[Send test alert](#)

No configured alert receivers.

Username: [Subscribe](#) Email address: [Subscribe](#)

Send a Test Email

In the Callbacks section, select email alert callback, and input your email address in the Receivers section. After you've added a callback type and receiver, hit the blue 'Send test alert' button.

Callbacks

The following callbacks will be performed when this stream triggers an alert.

Select Callback Type
Add callback
Find more callbacks

Email Alert Callback

Executed once per triggered alert condition.

Edit callback
Delete callback

body:

```

#####
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}

sender:
  graylog@example.org
subject:
  Graylog alert for stream: ${stream.title}: ${check_result.resultDescription}

```

Receivers

The following Graylog users will be notified about alerts via email if they have configured an email address in their profile. You can also add any other email address to the alert receivers if it has no Graylog user associated.

Send test alert

sam@graylog.com

Username:
Subscribe
Email address:
Subscribe

Going Further

If you want to configure an SMTP server, you can refer to the [this documentation](#).

If you want to make this stream active, just go back to Streams and where you see the stream name, click the green *Start stream* button.

Security/Auth Errors from Syslogs

stopped

Edit rules
Manage outputs
Manage alerts
Start stream
More actions

Security

0 messages/second, Must match all of the 1 configured stream rule(s). [Show stream rules](#)

You are done - go grab a Creamsicle, take a deep breath, and chillax. Tomorrow you can configure all your own logs and alerts. To help, go and get some deep knowledge in the official [documentation](#).

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.)

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

Virtual Machine Appliances

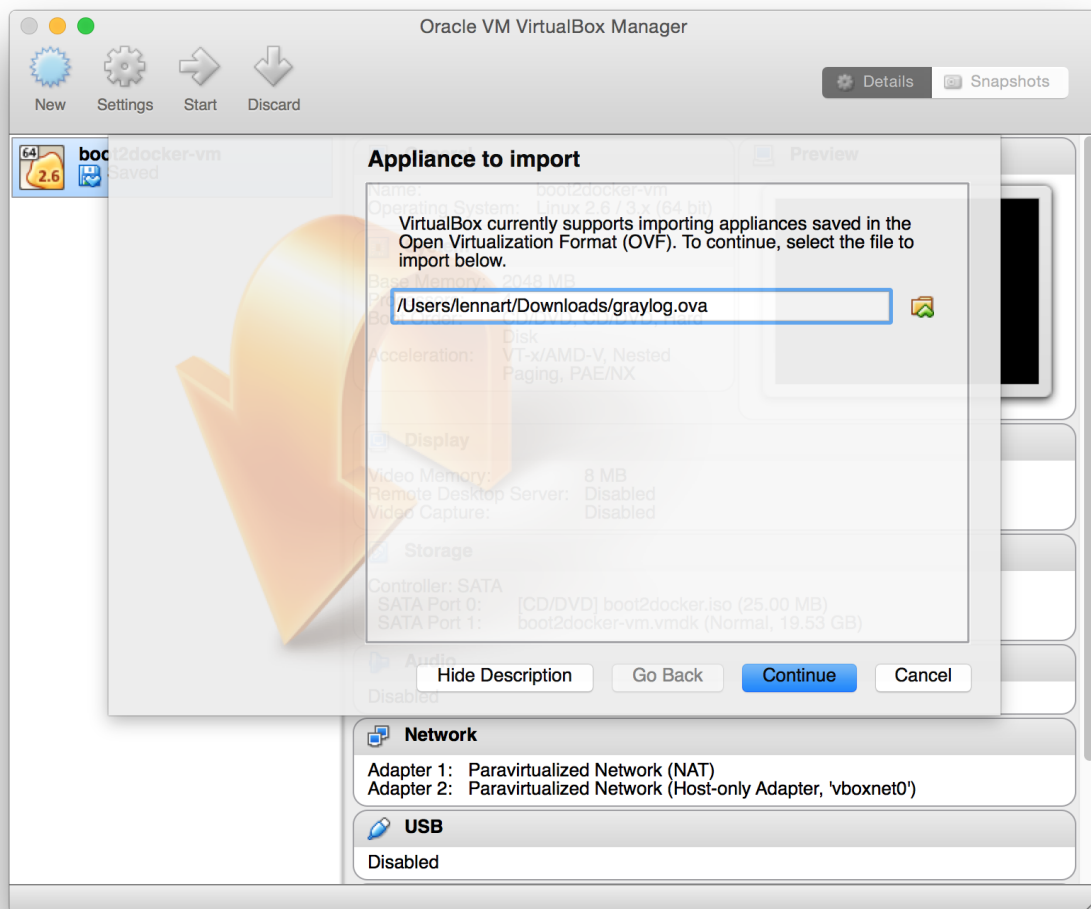
Download

Download the OVA image from [here](#) and save it to your disk locally.

Run the image

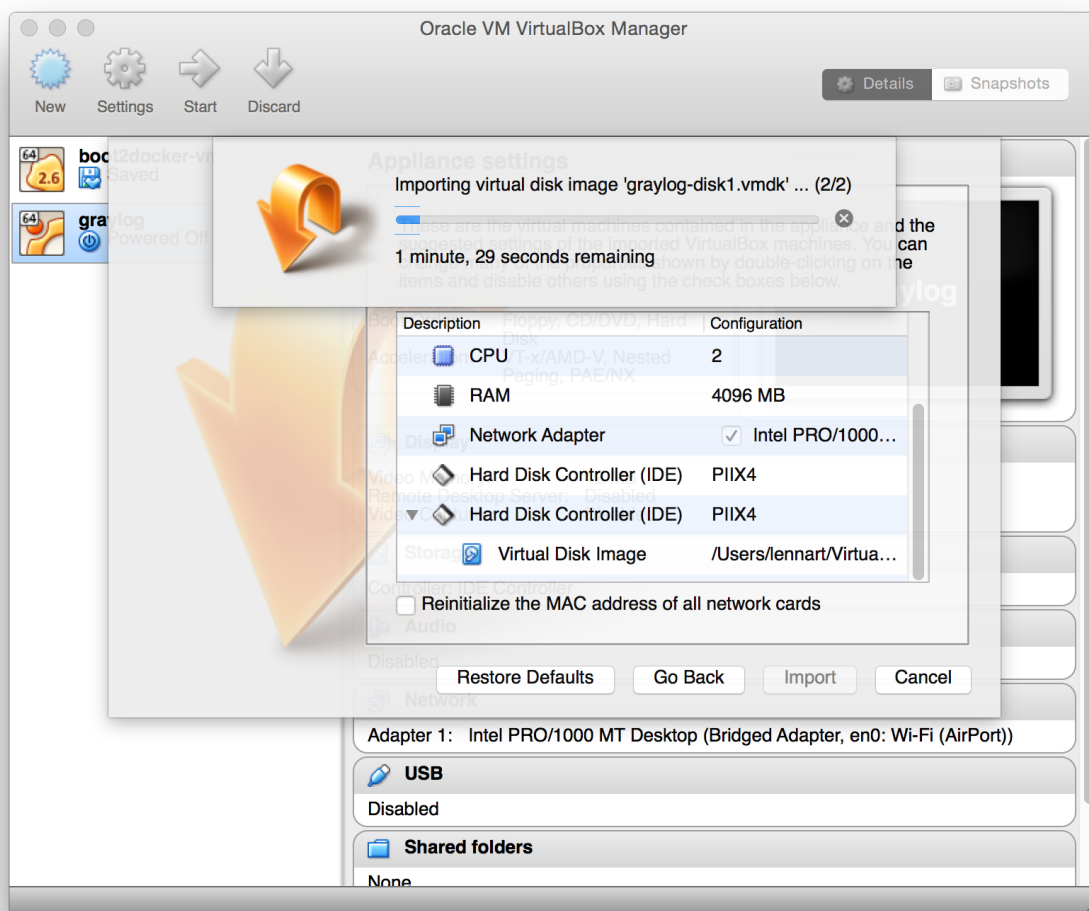
You can run the OVA in many systems like **VMware** or **Virtualbox**. In this example we will guide you through running the OVA in the free Virtualbox on OSX.

In Virtualbox select *File -> Import appliance*:

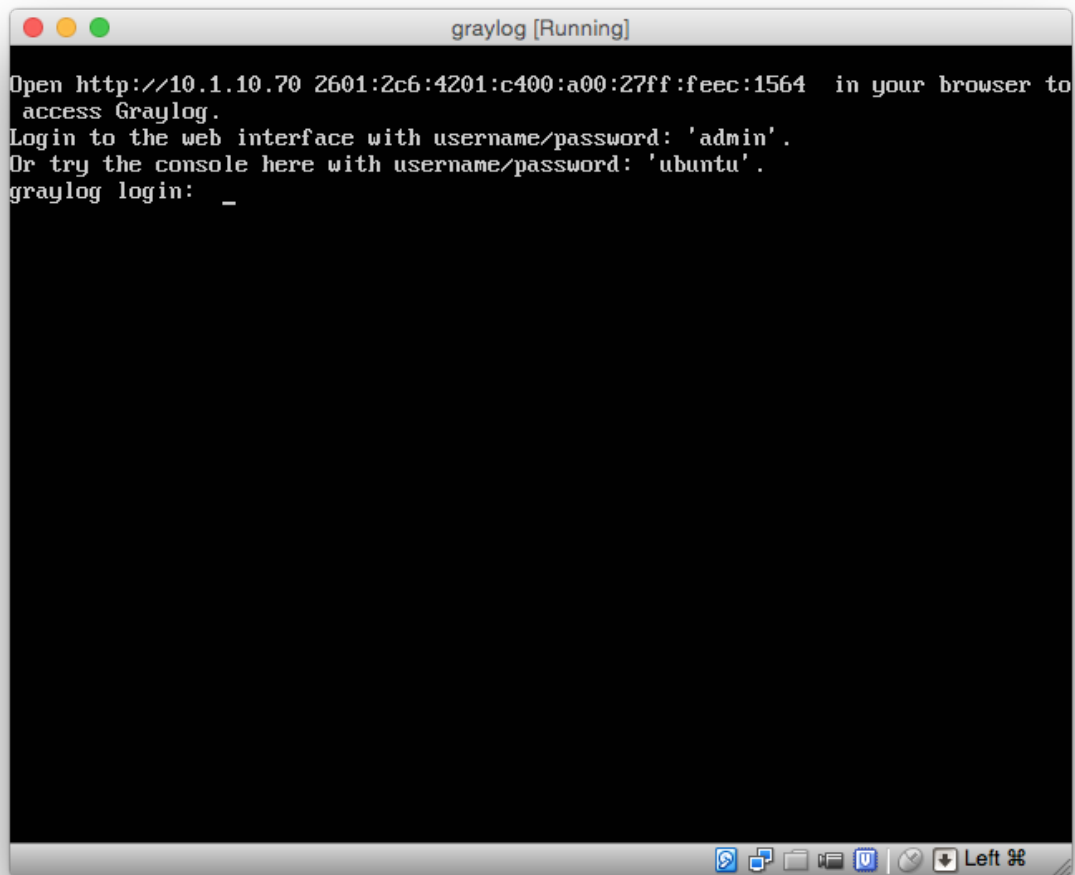


Hit *Continue* and keep the suggested settings on the next page as they are. Make sure that you have enough RAM and CPUs on your local machine. You can lower the resources the virtual machine will get assigned but we recommend to not lower it to ensure a good Graylog experience. In fact you might have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

Press *Import* to finish loading the OVA into Virtualbox:



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



Note: If you don't have a working DHCP server for your virtual machine, you will get the error message:

“Your appliance came up without a configured IP address. Graylog is probable not running correctly!”

In this case, you have to login and edit `/etc/network/interfaces` in order to setup a fixed IP address. Then manually reconfigure Graylog as shown in the following paragraphs.

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). DHCP should be enabled in your network otherwise take a look into the `graylog-ctl` command to apply a static IP address to the appliance.

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.

VMWare tools

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

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

Update OVA to latest Version

If you want to update your Appliance to the newest release without deploying a new template the best solution is covered [here](#).

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
- Separate the box network-wise from the outside, otherwise Elasticsearch can be reached by anyone
- add additional RAM to the appliance and raise the *java heap*!
- add additional HDD to the appliance and *extend disk space*.
- add the appliance to your monitoring and metric systems.

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. The packages have been tested on the following operating systems:

- Ubuntu 12.04, 14.04, 16.04
- Debian 7, 8
- RHEL/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.

Prerequisites

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

- Java (≥ 8)
- MongoDB (≥ 2.4)
- Elasticsearch ($\geq 2.x$, $<2.4.x$)

Caution: Graylog 2.x **does not** work with Elasticsearch 5.x!

DEB / APT

Download and install `graylog-2.0-repository_latest.deb` via `dpkg(1)` and also make sure that the `apt-transport-https` package is installed:

```
$ sudo apt-get install apt-transport-https
$ wget https://packages.graylog2.org/repo/packages/graylog-2.0-repository_latest.deb
$ sudo dpkg -i graylog-2.0-repository_latest.deb
$ sudo apt-get update
$ sudo apt-get install graylog-server
```

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

OS	Init System	Command
Ubuntu 14.04, 12.04	upstart	<code>sudo start graylog-server</code>
Debian 7	SysV	<code>sudo service graylog-server start</code>
Debian 8, Ubuntu 16.04	systemd	<code>sudo systemctl start graylog-server</code>

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

OS	Init System	Command
Ubuntu 14.04, 12.04	upstart	<code>sudo rm -f /etc/init/graylog-server.override</code>
Debian 7	SysV	<code>sudo update-rc.d graylog-server defaults 95 10</code>
Debian 8, Ubuntu 16.06	systemd	<code>sudo systemctl enable graylog-server</code>

Manual Repository Installation

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

First, add the [Graylog GPG keyring](#) which is being used to sign the packages to your system.

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

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

```
deb https://packages.graylog2.org/repo/debian/ stable 2.0
```

RPM / YUM / DNF

Download and install `graylog-2.0-repository_latest.rpm` via `rpm(8)`:

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-2.0-repository_latest.rpm
$ sudo yum install graylog-server
```

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

OS	Init System	Command
CentOS 6	SysV	<code>sudo service graylog-server start</code>
CentOS 7	systemd	<code>sudo systemctl start graylog-server</code>

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

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

Manual Repository Installation

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

First, add the [Graylog GPG key](#) which is being used to sign the packages to your system.

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

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

```
[graylog]
name=graylog
baseurl=https://packages.graylog2.org/repo/el/stable/2.0/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-graylog
```

Step-by-step guides

Ubuntu installation

This guide describes the fastest way to install Graylog on Ubuntu 16.04 LTS. All links and packages are present at the time of writing but might need to be updated later on.

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

Prerequisites

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

```
$ apt-get install apt-transport-https openjdk-8-jre-headless uuid-runtime pwgen
```

MongoDB

The Version included in Ubuntu 16.04 LTS can be used together with Graylog 2.0.0 and higher:

```
$ apt-get install mongodb-server
```

Elasticsearch

Graylog 2.0.0 and higher requires Elasticsearch 2.x, so we took the installation instructions from [the Elasticsearch installation guide](#):

```
$ wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -  
$ echo "deb https://packages.elastic.co/elasticsearch/2.x/debian stable main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch.list  
$ sudo apt-get update && sudo apt-get install elasticsearch
```

Make sure to modify the [Elasticsearch configuration file](#) (/etc/elasticsearch/elasticsearch.yml) and set the [cluster name](#) to graylog:

```
cluster.name: graylog
```

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

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

Graylog

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

```
$ wget https://packages.graylog2.org/repo/packages/graylog-2.0-repository_latest.deb  
$ sudo dpkg -i graylog-2.0-repository_latest.deb  
$ sudo apt-get update && sudo apt-get install graylog-server
```

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

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

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

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

Feedback

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

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

Debian installation

This guide describes the fastest way to install Graylog on Debian Linux 8 (Jessie). All links and packages are present at the time of writing but might need to be updated later on.

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

Prerequisites

Not all required dependencies are available in the standard repository, so we need to add [Debian Backports](#) to the list of package sources:

```
$ sudo echo "deb http://ftp.debian.org/debian jessie-backports main" > /etc/apt/sources.list.d/backports.list
$ sudo apt-get update
```

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

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

MongoDB

The version of MongoDB included in Debian Jessie is recent enough to be used with Graylog 2.0.0 and higher:

```
$ apt-get install mongodb-server
```

Elasticsearch

Graylog 2.0.0 and higher requires Elasticsearch 2.x, so we took the installation instructions from [the Elasticsearch installation guide](#):

```
$ wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
$ echo "deb https://packages.elastic.co/elasticsearch/2.x/debian stable main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch.list
$ sudo apt-get update && sudo apt-get install elasticsearch
```

Make sure to modify the [Elasticsearch configuration file](#) (/etc/elasticsearch/elasticsearch.yml) and set the [cluster name](#) to graylog:

```
cluster.name: graylog
```

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

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

Graylog

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

```
$ wget https://packages.graylog2.org/repo/packages/graylog-2.0-repository_latest.deb
$ sudo dpkg -i graylog-2.0-repository_latest.deb
$ sudo apt-get update && sudo apt-get install graylog-server
```

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

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

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

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

Feedback

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

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

CentOS installation

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

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

Prerequisites

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

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

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

MongoDB

Installing MongoDB on CentOS should follow [the tutorial for RHEL and CentOS](#) from the MongoDB documentation. First add the repository file `/etc/yum.repos.d/mongodb-org-3.2.repo` with the following contents:

```
[mongodb-org-3.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.2.asc
```

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

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

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

Elasticsearch

Graylog 2.0.0 and higher requires Elasticsearch 2.x, so we took the installation instructions from [the Elasticsearch installation guide](#).

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

```
[elasticsearch-2.x]
name=Elasticsearch repository for 2.x packages
baseurl=https://packages.elastic.co/elasticsearch/2.x/centos
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

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

Make sure to modify the [Elasticsearch configuration file](#) (`/etc/elasticsearch/elasticsearch.yml`) and set the [cluster name](#) to graylog:

```
cluster.name: graylog
```

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

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

Graylog

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

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-2.0-repository_latest.rpm
$ sudo yum install graylog-server
```

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

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

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

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

SELinux information

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

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

- Allow the web server to access the network: `sudo setsebool -P httpd_can_network_connect 1`
- If the policy above does not comply with your security policy, you can also allow access to each port individually:
 - Graylog REST API: `sudo semanage port -a -t http_port_t -p tcp 12900`
 - Graylog web interface: `sudo semanage port -a -t http_port_t -p tcp 9000`
 - Elasticsearch (only if the HTTP API is being used): `sudo semanage port -a -t http_port_t -p tcp 9200`
- Allow using MongoDB's default port (27017/tcp): `sudo semanage port -a -t mongod_port_t -p tcp 27017`

If you run a single server environment with *NGINX or Apache proxy*, enabling the Graylog REST API is enough. All other rules are only required in a multi-node setup.

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

Further reading

- <https://www.nginx.com/blog/nginx-se-linux-changes-upgrading-rhel-6-6/>
- <https://wiki.centos.org/HowTos/SELinux>
- <https://wiki.centos.org/TipsAndTricks/SelinuxBooleans>
- <http://www.serverlab.ca/tutorials/linux/administration-linux/troubleshooting-selinux-centos-red-hat/>

- https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/SELinux_Users_and_Administrators_Guide/
- <https://www.digitalocean.com/community/tutorials/an-introduction-to-selinux-on-centos-7-part-1-basic-concepts>

Feedback

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

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

Feedback

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

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

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.puppet.com/graylog/graylog>
- <https://galaxy.ansible.com/Graylog2/graylog-ansible-role>

There are also official [Vagrant](#) images if you want to spin up a local virtual machine quickly:

- <https://github.com/Graylog2/graylog2-images/tree/2.0/vagrant>

Docker

Requirements

You need a recent *docker* version installed, take a look [here](#) for instructions.

This will create three containers with all Graylog services running:

```
$ docker run --name some-mongo -d mongo:3
$ docker run --name some-elasticsearch -d elasticsearch:2.3 elasticsearch -Des.cluster.name="graylog2"
$ docker run --link some-mongo:mongo --link some-elasticsearch:elasticsearch -d graylog2/server
```

Testing a beta version

You can also run a pre-release or beta version of Graylog using Docker. The pre-releases are included in the *graylog2/server* image. Follow this [guide](#) and pick an alpha/beta/rc tag like:

```
$ docker run --link some-mongo:mongo --link some-elasticsearch:elasticsearch -d graylog2/server:2.1.0
```

We only recommend to run beta versions if you are an experienced Graylog user and know what you are doing.

Settings

Graylog comes with a default configuration that works out of the box but you have to set a password for the admin user. Also the web interface needs to know how to connect from your browser to the Graylog API. Both can be done via environment variables:

```
-e GRAYLOG_PASSWORD_SECRET=somepasswordpepper
-e GRAYLOG_ROOT_PASSWORD_SHA2=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
-e GRAYLOG_REST_TRANSPORT_URI="http://127.0.0.1:12900"
```

In this case you can login to Graylog with the username and password *admin*. Generate your own password with this command:

```
$ echo -n yourpassword | shasum -a 256
```

This all can be put in a *docker-compose.yml* file, like:

```
version: '2'
services:
  mongo:
    image: "mongo:3"
  elasticsearch:
    image: "elasticsearch:2.3"
    command: "elasticsearch -Des.cluster.name='graylog'"
  graylog:
    image: graylog2/server:2.0.3-2
    environment:
      GRAYLOG_PASSWORD_SECRET: somepasswordpepper
      GRAYLOG_ROOT_PASSWORD_SHA2: 8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
      GRAYLOG_REST_TRANSPORT_URI: http://127.0.0.1:12900
    depends_on:
      - mongo
      - elasticsearch
    ports:
      - "9000:9000"
      - "12900:12900"
```

After starting the three containers with *docker-compose up* open your browser with the URL *http://127.0.0.1:9000* and login with *admin: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 go through.

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*

Persist log data

In order to make the log data and configuration of Graylog persistent, you can use external volumes to store all data. In case of a container restart simply re-use the existing data from the former instances. Create the configuration directory and copy the default files:

```
mkdir /graylog/config
cd /graylog/config
wget https://raw.githubusercontent.com/Graylog2/graylog2-images/2.0/docker/config/graylog.conf
wget https://raw.githubusercontent.com/Graylog2/graylog2-images/2.0/docker/config/log4j2.xml
```

The *docker-compose.yml* file looks like this:

```
version: '2'
services:
  mongo:
    image: "mongo:3"
    volumes:
      - /graylog/data/mongo:/data/db
  elasticsearch:
    image: "elasticsearch:2.3"
    command: "elasticsearch -Des.cluster.name='graylog'"
    volumes:
      - /graylog/data/elasticsearch:/usr/share/elasticsearch/data
  graylog:
    image: graylog2/server
    volumes:
      - /graylog/data/journal:/usr/share/graylog/data/journal
      - /graylog/config:/usr/share/graylog/data/config
    environment:
      GRAYLOG_PASSWORD_SECRET: somepasswordpepper
      GRAYLOG_ROOT_PASSWORD_SHA2: 8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
      GRAYLOG_REST_TRANSPORT_URI: http://127.0.0.1:12900
    depends_on:
      - mongo
      - elasticsearch
    ports:
      - "9000:9000"
      - "12900:12900"
      - "12201/udp:12201/udp"
      - "1514/udp:1514/udp"
```

Start all services with exposed data directories:

```
$ docker-compose up
```

Configuration

Every configuration option can be set via environment variables, take a look [here](#) for an overview. Simply prefix the parameter name with *GRAYLOG_* and put it all in upper case. Another option would be to store the configuration file outside of the container and edit it directly.

Plugins

In order to add plugins you can build a new image based on the existings *graylog2/server* image with the needed plugin included. Simply create a new Dockerfile in an empty directory:

```
FROM graylog2/server:2.0.3-2
RUN wget -O /usr/share/graylog/plugin/graylog-plugin-beats-1.0.3.jar https://github.com/Graylog2/graylog-plugin-beats/releases/download/1.0.3/graylog-plugin-beats-1.0.3.jar
```

Build a new image from that:

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

In this example we created a new image with the Beats plugin installed. From now on reference to that image instead of the `graylog2/server` e.g. in a `docker-compose.yml` file:

```
version: '2'
services:
  mongo:
    image: "mongo:3"
    volumes:
      - /graylog/data/mongo:/data/db
  elasticsearch:
    image: "elasticsearch:2.3"
    command: "elasticsearch -Des.cluster.name='graylog'"
    volumes:
      - /graylog/data/elasticsearch:/usr/share/elasticsearch/data
  graylog:
    image: graylog-with-beats-plugin
...
```

Problems

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

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

- The *devicemapper* storage driver can produce problems with Graylogs disk journal on some systems. In this case please pick another driver like *aufs* or *overlay*. Have a look [here](#)

Build

To build the image from scratch run:

```
$ docker build --build-arg GRAYLOG_VERSION=${GRAYLOG_VERSION} -t graylog2/server .
```

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
- Separate the box network-wise from the outside, otherwise Elasticsearch can be reached by anyone
- add additional RAM to the appliance and raise the *java heap*!
- add additional HDD to the appliance and *extend disk space*.
- add the appliance to your monitoring and metric systems.

If you want to create your own customised setup take a look at our *[other installation methods](#)*.

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/2.0/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 : *http://localhost:8080*

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.

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
- Separate the box network-wise from the outside, otherwise Elasticsearch can be reached by anyone
- add additional RAM to the appliance and raise the *java heap*!
- add additional HDD to the appliance and *extend disk space*.
- add the appliance to your monitoring and metric systems.

If you want to create your own customised setup take a look at our [other installation methods](#).

OpenStack

Installation

Download the Graylog image from the [package](#) site, uncompress it and import it into the Openstack image store:

```
$ wget https://packages.graylog2.org/releases/graylog-omnibus/qcow2/graylog-2.0.0-1.qcow2.gz
$ gunzip graylog.qcow2.gz
$ glance image-create --name='graylog' --is-public=true --container-format=bare --disk-format=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.

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
- Separate the box network-wise from the outside, otherwise Elasticsearch can be reached by anyone
- add additional RAM to the appliance and raise the *java heap*!
- add additional HDD to the appliance and *extend disk space*.
- add the appliance to your monitoring and metric systems.

If you want to create your own customised setup take a look at our [other installation methods](#).

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 12900 in the applied security group to access the web interface.
- additionally open more ports for ingesting log data, like 514 for syslog or 12201 for the GELF protocol.

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

Networking

Your browser needs access to port 80 or 443 for reaching the web interface. The interface itself creates a connection back to the REST API of the Graylog server on port 12900. As long as you are in a private network like Amazon VPC for instance, this works out of the box. If you want to use the *public* IP address of your VM, this mechanism doesn't work automatically anymore. You have to tell Graylog how to reach the API from the users browser perspective:

```
sudo graylog-ctl set-external-ip http://<public ip>:12900
sudo graylog-ctl reconfigure
```

Also make sure that this port is open, even on the public IP.

HTTPS

In order to enable HTTPS for the web interface both ports need to be encrypted. Otherwise the web browser would show an error message. For this reason we created a proxy configuration on the appliance that can be enabled by running:

```
sudo graylog-ctl enforce-ssl
sudo graylog-ctl reconfigure
```

This command combines the Graylog web interface and the API on port 443. The API is accessible via the path `/api`. For this reason you have to set the external IP to an HTTPS address with the appended path `/api`:

```
sudo graylog-ctl set-external-ip https://<public ip>:443/api
sudo graylog-ctl reconfigure
```

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.

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
- Separate the box network-wise from the outside, otherwise Elasticsearch can be reached by anyone
- add additional RAM to the appliance and raise the *java heap*!
- add additional HDD to the appliance and *extend disk space*.
- add the appliance to your monitoring and metric systems.

If you want to create your own customised setup take a look at our *[other installation methods](#)*.

Microsoft Windows

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

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

```
message_journal_enabled = false
```

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

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

Manual Setup

Graylog server on Linux

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 2.1.x or later](#)
- [MongoDB 2.0 or later](#) (latest stable version is recommended)
- Oracle Java SE 8 or later (Oracle Java SE 7 is end of life and is no longer supported. OpenJDK 8 also works; 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 8** or higher! Java 7 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_uri`
 - Enter your MongoDB connection and authentication information here.

Starting the server

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

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

You need at least Java 8 as Java 7 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
```

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

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

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

Append this before the `-jar` paramter:

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

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

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

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

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

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

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

Command line (CLI) parameters

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

- `-h, --help`: Show help message
- `-f CONFIGFILE, --configfile CONFIGFILE`: Use configuration file *CONFIGFILE* for Graylog; default: `/etc/graylog/server/server.conf`
- `-d, --debug`: Run in debug mode
- `-l, --local`: Run in local mode. Automatically invoked if in debug mode. Will not send system statistics, even if enabled and allowed. Only interesting for development and testing purposes.
- `-p PIDFILE, --pidfile PIDFILE`: Set the file containing the PID of graylog to *PIDFILE*; default: `/tmp/graylog.pid`
- `-np, --no-pid-file`: Do not write PID file (overrides `-p/-pidfile`)
- `--version`: Show version of Graylog and exit

Problems with IPv6 vs. IPv4?

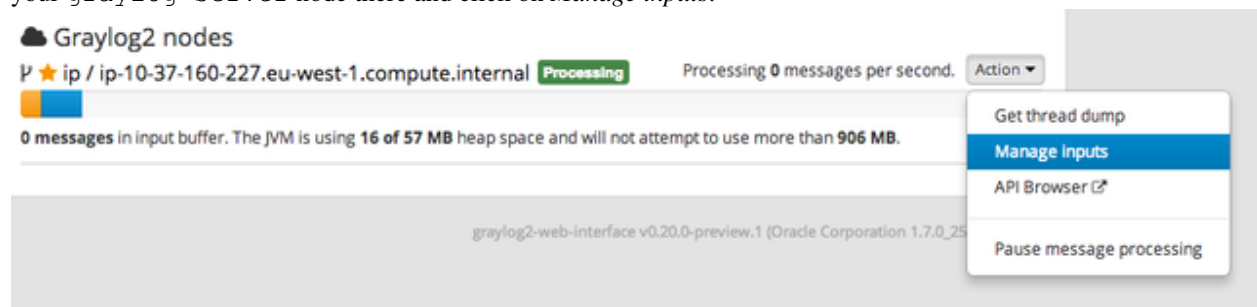
If your *graylog-server* instance refuses to listen on IPv4 addresses and always chooses for example a *rest_listen_address* like `:::12900` you can tell the JVM to prefer the IPv4 stack.

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
```

Create a message input and send a first message

Log in to the web interface on port 9000 (e.g. `http://127.0.0.1:9000`) and navigate to *System -> Nodes*. Select your *graylog-server* node there and click on *Manage inputs*.



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 TLS support in the `/etc/graylog/server/server.conf` like this:

```
web_enable_tls=true
```

This will generate self-signed certificate. To use proper certificates you must provide a PEM-encoded private key in PKCS#8 format and a X.509 certificate. Most certificate authorities provide instructions on how to create such keys and certificates.

The OpenSSL documentation also provides [examples on how to handle PKCS#8 files](#).

- `web_tls_cert_file` The X.509 certificate file (PEM-encoded) to use for securing the web interface port.
- `web_tls_key_file` The private key in PKCS#8 format (PEM-encoded) to use for securing the web interface port.
- `web_tls_key_password` The password, defaults to a blank password

System requirements

The Graylog server application has the following prerequisites:

- Some modern Linux distribution (Debian Linux, Ubuntu Linux, or CentOS recommended)
- [Elasticsearch 2.x](#) (2.1.0 or later, but not 2.4.0 or later)
- [MongoDB 2.4 or later](#) (latest stable version is recommended)
- Oracle Java SE 8 (OpenJDK 8 also works; latest stable update is recommended)

Warning: Graylog 2.0.x currently **does not** work with Elasticsearch 2.4.x or 5.x. The latest supported version is [Elasticsearch 2.3.5](#).

Upgrading Graylog

From 1.x to 2.x

Elasticsearch 2.x

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

Please see [Breaking changes in Elasticsearch 2.x](#) for details.

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

Multicast Discovery

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

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

Default network host

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

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

Index range types

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

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

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

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

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

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

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

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

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

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

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

Graylog Index Template

Graylog applies a custom [index template](#) to ensure that the indexed messages adhere to a specific schema.

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

In order to [delete the index template](#) the following `curl` command has to be issued against one of the Elasticsearch nodes:

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

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

Dots in field names

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

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

MongoDB

Graylog 2.x requires MongoDB 2.4 or newer. We recommend using MongoDB 3.x and the [WiredTiger storage engine](#). When upgrading from MongoDB 2.0 or 2.2 to a supported version, make sure to read the [Release Notes](#) for the particular version.

Log4j 2 migration

Graylog switched its logging backend from [Log4j 1.2](#) to [Log4j 2](#).

Please refer to the [Log4j Migration Guide](#) for information on how to update your existing logging configuration.

Dead Letters feature removed

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

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

Removed configuration settings

Index Retention and Rotation Settings

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

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

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

Overview

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

Table 4.1: Removed configuration settings

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

Changed configuration defaults

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

Table 4.2: Configuration defaults

Setting name	Old default	New default
elasticsearch_cluster_name	graylog2	graylog
elasticsearch_node_name	graylog2-server	graylog-server
elasticsearch_index_prefix	graylog2	graylog
elasticsearch_discovery_zen_ping_unicast_enabled	empty hosts	127.0.0.1:9300
elasticsearch_discovery_zen_ping_multicast_enabled	true	false
mongodb_uri	mongodb://127.0.0.1:27020/graylog2	mongodb://localhost/graylog

Changed prefixes for configuration override

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

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

Table 4.3: Configuration override prefixes

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

REST API Changes

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

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

Web Interface Config Changes

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

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

Configuring Graylog

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: `graylog-ctl` is only available in the virtual machine appliances, but not in the tar-ball (for manual setup), operating system packages, or configuration management scripts (Puppet, Chef, Ansible).

Configuration commands

The following commands are changing the configuration of Graylog:

Command	Description
<code>sudo graylog-ctl set-admin-password <password></code>	Set a new admin password
<code>sudo graylog-ctl set-admin-username <username></code>	Set a different username for the admin user
<code>sudo graylog-ctl set-email-config <smtp server> [--port=<smtp port> --user=<username> --password=<password> --from-email=<sender-address> --web-url=<graylog web-interface url> --no-tls --no-ssl]</code>	Configure SMTP settings to send alert mails
<code>sudo graylog-ctl set-timezone <zone acronym></code>	Set Graylog's time zone from a list of valid time zones . Make sure system time is also set correctly with <code>sudo dpkg-reconfigure tzdata</code> .
<code>sudo graylog-ctl enforce-ssl</code>	Enforce HTTPS for the web interface
<code>sudo graylog-ctl set-node-id <id></code>	Override random server node id
<code>sudo graylog-ctl set-server-secret <secret></code>	Override server secret used for encryption
<code>sudo graylog-ctl disable-internal-logging</code>	Disable sending internal logs (e. g. nginx) from the VM to Graylog
<code>sudo graylog-ctl set-external-ip http[s]://<public IP>:port/</code>	Configure an external IP in the Nginx proxy. This is needed to connect the web interface to the REST API e.g. in NAT'd networks or on AWS.
<code>sudo graylog-ctl set-listen-address --service <web rest transport> --address http://<host>:port</code>	Set the listen address for the web interface, REST API, and the transport URI. Can be used to deal with additional network interfaces.

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 make sense to not run all services on a single VM anymore. For performance reasons you might want to add more Elasticsearch nodes to the cluster or even add a second Graylog server. This can be achieved by changing IP addresses in the Graylog configuration files by hand or 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 the 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 create a small cluster with a dedicated Graylog server node and another for Elasticsearch, spin up two VMs from the same Graylog image. On the first one start only Graylog and MongoDB:

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

On the second VM start only Elasticsearch. Before doing so set the IP of the first VM to fetch the configuration data from there:

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

vm1> sudo graylog-ctl reconfigure-as-server
```

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

vm1> sudo graylog-ctl reconfigure-as-server
vm2> sudo graylog-ctl reconfigure-as-datanode
```

Verify that all nodes are working as a cluster by going to the Kopf plugin on one of the Elasticsearch nodes open `http://vm2:9200/_plugin/kopf/#!/nodes`.

Important: In case you want to add a second Graylog server you have to set the same server secret on all machines. The secret is stored in the file `/etc/graylog/graylog-secrets` and can be applied to other hosts with the `set-server-secret` sub-command.

The following configuration modes do exist:

Command	Services
<code>sudo graylog-ctl reconfigure</code>	Run all services on this box
<code>sudo graylog-ctl reconfigure-as-server</code>	Run Graylog, web and MongoDB (no Elasticsearch)
<code>sudo graylog-ctl reconfigure-as-backend</code>	Run Graylog, Elasticsearch and MongoDB (no nginx for web interface access)
<code>sudo graylog-ctl reconfigure-as-datanode</code>	Run only Elasticsearch

A server with only the web interface running is not supported anymore since Graylog 2.0. The web interface is now included in the server process. But you can create your own service combinations by editing the file `/etc/graylog/graylog-services.json` by hand and enable or disable single services. Just run `graylog-ctl reconfigure` afterwards.

Extend disk space

All data of the appliance setup is stored in `/var/opt/graylog/data`. In order to extend the disk space mount a second (virtual) hard drive into this directory.

Important: Make sure to move old data to the new drive before and give the `graylog` user permissions to read and write here.

Example procedure for the Graylog virtual appliance

Note: These steps require basic knowledge in using Linux and the common shell programs.

- Shutdown the virtual machine as preparation for creating a consistent snapshot.
- Take a snapshot of the virtual machine in case something goes wrong.
 - [Understanding VM snapshots in ESXi / ESX](#)
 - [VMware vSphere: Managing Snapshots](#)
 - [VirtualBox: Snapshots](#)
 - [Parallels: Save Snapshots of a Virtual Machine](#)
 - [Parallels: Working with snapshots](#)
- Attach an additional hard drive to the virtual machine.
 - [VMware Workstation: Adding a New Virtual Disk to a Virtual Machine](#)
 - [VMware vSphere: Virtual Disk Configuration](#)
 - [VirtualBox: Virtual storage](#)
 - [Parallels: Hard Disk](#)
- Start the virtual machine again.
- Stop all services to prevent disk access:

```
$ sudo graylog-ctl stop
```

- Check for the *logical name* of the new hard drive. Usually this is `/dev/sdb`:


```
$ sudo lshw -class disk
```

- Partition and format new disk:

```
$ sudo parted -a optimal /dev/sdb mklabel gpt
# A reboot may be necessary at this point so that the updated GPT is being recognized by the operating system
$ sudo parted -a optimal -- /dev/sdb unit compact mkpart primary ext3 "1" "-1"
$ sudo mkfs.ext4 /dev/sdb1
```

- Mount disk into temporary directory /mnt/tmp:

```
$ sudo mkdir /mnt/tmp
$ sudo mount /dev/sdb1 /mnt/tmp
```

- Copy current data to new disk:

```
$ sudo cp -ax /var/opt/graylog/data/* /mnt/tmp/
```

- Compare both folders:

```
# Output should be: Only in /mnt/tmp: lost+found
$ sudo diff -qr --suppress-common-lines /var/opt/graylog/data /mnt/tmp
```

- Delete old data:

```
$ sudo rm -rf /var/opt/graylog/data/*
```

- Mount new disk into /var/opt/graylog/data directory:

```
$ sudo umount /mnt/tmp
$ sudo mount /dev/sdb1 /var/opt/graylog/data
```

- Make change permanent by adding an entry to /etc/fstab:

```
$ echo '/dev/sdb1 /var/opt/graylog/data ext4 defaults 0 0' | sudo tee -a /etc/fstab
```

- Reboot virtual machine:

```
$ sudo shutdown -r now
```

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`.

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

Warning: The Graylog omnibus package does *not* support unattended upgrading from Graylog 1.x to Graylog 2.0.x!

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. Choose the Graylog version you want to install from the [list of Omnibus packages](#). `graylog_latest.deb` always links to the newest version:

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

Migrate manually from 1.x to 2.0.x

To update a 1.x appliance to 2.0.x the administrator has to purge the Graylog installation, migrate the stored log data and install the new version as Omnibus package. Before upgrading read the [upgrade notes](#). This procedure can potentially delete log data or configuration settings. So it's absolutely necessary to perform a backup or a snapshot before!

Stop all services but Elasticsearch:

```
$ sudo -s
$ graylog-ctl stop graylog-web
$ graylog-ctl stop graylog-server
$ graylog-ctl stop mongoddb
```

```
$ graylog-ctl stop nginx
$ graylog-ctl stop etcd
```

Check for index range types. The output of this command should be `{}`, if not [read](#) how to fix this:

```
$ curl -XGET <appliance_IP>:9200/_all/_mapping/index_range; echo
{}
```

Delete the Graylog index template:

```
$ curl -X DELETE <appliance_IP>:9200/_template/graylog-internal
```

Migrate appliance configuration:

```
$ cd /etc
$ mv graylog graylog2.0
$ vi graylog2.0/graylog-secrets.json

# Remove the graylog_web section
}, << don't forget the comma!
"graylog_web": {
  "secret_token": "3552c87f3e3..."
}

$ vi graylog2.0/graylog-services.json

# Remove the graylog_web section
}, << don't forget the comma!
"graylog_web": {
  "enabled": true
}

$ vi graylog2.0/graylog-settings.json

# Remove "rotation_size", "rotation_time", "indices"
"enforce_ssl": false,
"rotation_size": 1073741824,
"rotation_time": 0,
"indices": 10,
"journal_size": 1,
```

Migrate appliance data:

```
$ cd /var/opt
$ mv graylog graylog2.0
$ mv graylog2.0/data/elasticsearch/graylog2 graylog2.0/data/elasticsearch/graylog
```

Delete old Graylog version and install new Omnibus package:

```
$ wget http://packages.graylog2.org/releases/graylog-omnibus/ubuntu/graylog_2.0.0-2_amd64.deb
$ apt-get purge graylog
$ dpkg -i graylog_2.0.0-2_amd64.deb
```

Move directories back:

```
$ cd /etc
$ mv graylog2.0 graylog
$ cd /var/opt/
$ mv graylog2.0 graylog
```

Reconfigure and Reboot:

```
$ graylog-ctl reconfigure
$ reboot
```

Graylog should now be updated and old data still available.

Important: The index retention configuration moved from the Graylog configuration file to the web interface. After the first start go to ‘System -> Indices -> Update configuration’ to re-enable your settings.

Advanced Settings

To change certain parameters used by `graylog-ctl` during a reconfigure run you can override all default parameters found in the `attributes` file. If you want to change the username used by Graylog for example, edit the file `/etc/graylog/graylog-settings.json` like this:

```
"custom_attributes": {
  "user": {
    "username": "log-user"
  }
}
```

Afterwards run `sudo graylog-ctl reconfigure` and `sudo graylog-ctl restart`. The first command renders all changed configuration files and the later makes sure that all services restart to activate the change.

There are a couple of other use cases of this, e.g. change the default data directories used by Graylog to `/data` (make sure this is writeable by the graylog user):

```
"custom_attributes": {
  "elasticsearch": {
    "data_directory": "/data/elasticsearch"
  },
  "mongodb": {
    "data_directory": "/data/mongodb"
  },
  "etcd": {
    "data_directory": "/data/etcd"
  },
  "graylog-server": {
    "journal_directory": "/data/journal"
  }
}
```

Or change the default memory settings used by Graylog or Elasticsearch:

```
"custom_attributes": {
  "graylog-server": {
    "memory": "1700m"
  },
  "elasticsearch": {
    "memory": "2200m"
  }
}
```

Again, run `reconfigure` and `restart` afterwards to activate the changes.

Web interface

When your Graylog instance/cluster is up and running, the next thing you usually want to do is check out our web interface, which offers you great capabilities for searching and analyzing your indexed data and configuring your Graylog environment. Per default you can access it using your browser on `http://<graylog-server>:9000`.

Overview

The Graylog web interface was rewritten in JavaScript for 2.0 to be a client-side single-page browser application. This means its code is running solely in your browser, fetching all data via HTTP(S) from the REST API of your Graylog server. Therefore there is a second HTTP listener which is serving the assets for the web interface (all JavaScript, fonts, images, CSS files) to the clients.

Note: Both the web interface port (`http://127.0.0.1:9000/` by default, see `web_listen_uri`) and the REST API port (`http://127.0.0.1:12900` by default, see `rest_listen_uri` and `rest_transport_uri`) must be accessible by everyone using the web interface. This means that both components *must* listen on a public network interface *or* be exposed to one using a proxy or NAT!

Configuration Options

If our default settings do not work for you, there is a number of options in the Graylog server configuration file which you can change to influence its behavior:

Setting	Default	Explanation
<code>web_enable</code>	<code>true</code>	Determines if the web interface endpoint is started or not.
<code>web_listen_uri</code>	<code>http://127.0.0.1:9000/</code>	Default address the web interface listener binds to.
<code>web_endpoint_uri</code>	If not set, <code>rest_transport_uri</code> will be used.	This is the external address of the REST API of the Graylog server. Web interface clients need to be able to connect to this for the web interface to work.
<code>web_enable_cors</code>	<code>false</code>	Support Cross-Origin Resource Sharing for the web interface assets. Not required, because no REST calls are made to this listener.
<code>web_enable_gzip</code>	<code>true</code>	Serve web interface assets using compression.
<code>web_enable_encryption</code>	<code>false</code>	Should the web interface serve assets using encryption or not.
<code>web_tls_certificate</code>	(no default)	Path to TLS certificate file, if TLS is enabled.
<code>web_tls_key</code>	(no default)	Path to private key for certificate, used if TLS is enabled.
<code>web_tls_key_password</code>	(no default)	Password for TLS key (if it is encrypted).
<code>web_thread_pool_size</code>	<code>16</code>	Number of threads used for web interface listener.

How does the web interface connect to the Graylog server?

The web interface is fetching all information it is showing from the REST API of the Graylog server. Therefore it needs to connect to it using HTTP(S). There are several ways how you can define which way the web interface connects to the Graylog server:

- If the HTTP(S) client going to the web interface port sends a `X-Graylog-Server-URL` header, which contains a valid URL, then this is overriding everything else.
- If `web_endpoint_uri` is defined in the Graylog configuration file, this is used if the aforementioned header is not set.
- If both are not defined, `rest_transport_uri` is used.

Browser Compatibility

Writing the web interface as a single-page application is a challenging task. We want to provide the best possible experience to everyone, which often means using modern web technology only available in recent browsers, while keeping a reasonable compatibility with old and less-capable browsers. These browsers are officially supported in Graylog 2.0:

Browser	OS	Minimum Version
Chrome	Windows, OS X, Linux	50
Firefox	Windows, OS X, Linux	45 / 38 ESR
Internet Explorer	Windows	11
Microsoft Edge	Windows	25
Safari	OS X	9

Please take into account that you need to enable JavaScript in order to use Graylog web interface.

Making the web interface work with load balancers/proxies

If you want to run a load balancer/reverse proxy in front of Graylog, you need to make sure that:

- The REST API port is still accessible for clients
- The address for the Graylog server's REST API is properly set (as explained in [How does the web interface connect to the Graylog server?](#)), so it is resolvable and accessible for any client of the web interface.
- You are either using only HTTP or only HTTPS (no mixed content) for both the web interface endpoint and the REST API endpoint.
- If you use SSL, your certificates must be valid and trusted by your clients.

Note: To help you with your specific environment, we have some example configurations. We take the following assumption in all examples. Your Graylog `server.conf` has the following settings set `rest_listen_uri = http://127.0.0.1:12900/` and `web_listen_uri = http://127.0.0.1:9000/`. Your URL will be `graylog.example.org` with the IP `192.168.0.10`.

Using a Layer 3 load balancer (forwarding TCP Ports)

1. Configure your load balancer to forward connections going to `192.168.0.10:80` to `127.0.0.1:9000` (`web_listen_uri`) and `192.168.0.10:12900` to `127.0.0.1:12900` (`rest_listen_uri`).
2. Set `web_endpoint_uri` in your Graylog server config to `http://graylog.example.org:12900`.
3. Start the Graylog server as usual.
4. Access the web interface on `http://graylog.example.org`.
5. Read up on [Using HTTPS](#).

NGINX

REST API and Web Interface on one port (using HTTP):

```
server
{
    listen      80 default_server;
    listen      [::]:80 default_server ipv6only=on;
```

```

server_name graylog.example.org;

location /api/
{
    proxy_set_header    Host $http_host;
    proxy_set_header    X-Forwarded-Host $host;
    proxy_set_header    X-Forwarded-Server $host;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass           http://127.0.0.1:12900/;
}

location /
{
    proxy_set_header    Host $http_host;
    proxy_set_header    X-Forwarded-Host $host;
    proxy_set_header    X-Forwarded-Server $host;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Graylog-Server-URL http://graylog.example.org/api;
    proxy_pass           http://127.0.0.1:9000;
}
}

```

REST API and web interface on separate ports (using HTTP):

```

server
{
    listen      80 default_server;
    listen      [::]:80 default_server ipv6only=on;
    server_name graylog.example.org;

    location /
    {
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    X-Graylog-Server-URL http://graylog.example.org:12900;
        proxy_set_header    Host $http_host;
        proxy_pass           http://127.0.0.1:9000;
    }
}

server
{
    listen      12900;
    server_name graylog.example.org;

    location /
    {
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    Host $http_host;
        proxy_pass           http://127.0.0.1:12900/;
    }
}

```

NGINX can be used for SSL Termination, you would only need to modify the `server listen` directive and add all Information about your certificate.

If you are running multiple Graylog Server you might want to use HTTPS/SSL to connect to the Graylog Servers (on how to Setup read [Using HTTPS](#)) and use HTTPS/SSL on NGINX. The configuration for TLS certificates, keys and ciphers is omitted from the sample config for brevity's sake.

REST API and Web Interface on one port (using HTTPS/SSL):

```
server
{
    listen      443 ssl spdy;
    server_name graylog.example.org;
    # <- your SSL Settings here!

    location /
    {
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    Host $http_host;
        proxy_set_header    X-Graylog-Server-URL https://graylog.example.org/api;
        proxy_pass            http://127.0.0.1:9000;
    }
    location /api/
    {
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    Host $http_host;
        proxy_pass            http://127.0.0.1:12900/;
    }
}
```

Apache httpd 2.x

REST API and Web Interface on one port (using HTTP):

```
<VirtualHost *:80>
    ServerName graylog.example.org
    ProxyRequests Off
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    <Location /api/>
        ProxyPass http://127.0.0.1:12900/
        ProxyPassReverse http://127.0.0.1:12900/
    </Location>
    <Location />
        RequestHeader set X-Graylog-Server-URL "http://graylog.example.org/api/"
        ProxyPass http://127.0.0.1:9000/
        ProxyPassReverse http://127.0.0.1:9000/
    </Location>
</VirtualHost>
```

Caution: Using Apache 2.2 needs the configuration above, if you have Apache 2.4 you need to switch the Locations. This means /api/ must go after /

HAProxy 1.6

REST API and Web Interface on one port (using HTTP):

```
frontend http
    bind 0.0.0.0:80

    option forwardfor
    http-request add-header X-Forwarded-Host %[req.hdr(host)]
```



```

http-request add-header X-Forwarded-Server %[req.hdr(host)]
http-request add-header X-Forwarded-Port %[dst_port]

acl is_graylog hdr_dom(host) -i -m str graylog.example.org
use_backend      graylog if is_graylog

backend graylog
  description      The Graylog Web backend.
  acl is_api var(req.api) -m bool true
  http-request set-var(req.api) bool(true) if { path_beg /api/ }
  http-request set-path %[path,regsub(/^api/,/)]
  http-request set-header X-Graylog-Server-URL http://graylog.example.org/api unless is_api
  use-server graylog_1_rest if is_api
  use-server graylog_1 unless is_api
  server graylog_1_rest 127.0.0.1:12900 maxconn 20 check
  server graylog_1 127.0.0.1:9000 maxconn 20 check

```

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.

The screenshot shows the Graylog web interface. At the top, there's a navigation bar with 'graylog' logo and links for Search, Streams, Dashboards, Sources, and System / Nodes. Below this, the 'Nodes' section is active, showing a real-time overview of nodes. A message states: 'You can pause message processing at any time. The process buffers will not accept any new messages until you resume it. If the message journal is enabled for a node, which it is by default, incoming messages will be persisted to disk, even when processing is disabled.' Below this, it says 'There is 1 active node' and shows a star icon next to 'a50e3d14 / graylog'. It indicates 'In 0 / Out 0 msg/s.' and 'The journal contains 0 unprocessed messages in 1 segment. 0 messages appended, 0 messages read in the last second.' A table shows the 'Current lifecycle state' as 'Running', 'Message processing' as 'Enabled', and 'Load balancer indication' as 'ALIVE'. A progress bar shows 'The JVM is using 773.1 MB of 1.1 GB heap space and will not attempt to use more than 1.4 GB'. A 'More actions' dropdown menu is open, listing: 'Pause message processing', 'Override LB status', 'Graceful shutdown', 'Local message inputs', and 'Get thread dump'.

Web Interface

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

Note: Take care of the configuration you need *with a proxy setup*, it will *not* work out of the box.

In the current Version 2.0 you did not need the sticky Session that was needed in previous Version. Only the API URL `/system/deflector/cycle` need to point to the configured Graylog Master Node. In future Version that might change.

Please refer to your vendor's documentation to learn how to route this URL to a specific host.

Using HTTPS

We highly recommend securing your Graylog installation using SSL/TLS to make sure that no sensitive data is sent over the wire in plain text. To make this work, you need to do two things:

- Enable TLS for the Graylog REST API (`rest_enable_tls`)
- Enable TLS for the web interface endpoint (`web_enable_tls`)

You also need to make sure that you have proper certificates in place, which are valid and trusted by the clients. Not enabling TLS for either one of them will result in a browser error about mixed content and the web interface will cease to work.

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

Certificate/Key file format

When you are configuring TLS, you need to make sure that your certificate/key files are in the right format, which is X.509 for certificates and PKCS#8 for the private keys. Both must be stored in PEM format.

Creating a self-signed private key/certificate

Create a file named `openssl-graylog.cnf` with the following content (customized to your needs):

```
[req]
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no

# Details about the issuer of the certificate
[req_distinguished_name]
C = US
ST = Some-State
L = Some-City
O = My Company
OU = My Division
CN = graylog.example.com

[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

# IP addresses and DNS names the certificate should include
# Use IP.### for IP addresses and DNS.### for DNS names,
# with "###" being a consecutive number.
[alt_names]
IP.1 = 203.0.113.42
DNS.1 = graylog.example.com
```

Create PKCS#5 private key and X.509 certificate:

```
$ openssl version
OpenSSL 0.9.8zh 14 Jan 2016
$ openssl req -x509 -days 365 -nodes -newkey rsa:2048 -config openssl-graylog.cnf -keyout pkcs5-plain.pem
Generating a 2048 bit RSA private key
.....+++
.+++
writing new private key to 'pkcs5-plain.pem'
-----
```

Convert PKCS#5 private key into a *unencrypted* PKCS#8 private key:

```
$ openssl pkcs8 -in pkcs5-plain.pem -topk8 -nocrypt -out pkcs8-plain.pem
```

Convert PKCS#5 private key into an *encrypted* PKCS#8 private key (using the passphrase `secret`):

```
$ openssl pkcs8 -in pkcs5-plain.pem -topk8 -out pkcs8-encrypted.pem -passout pass:secret
```

Converting a PKCS #12 (PFX) file to private key and certificate pair

PKCS #12 key stores (PFX files) are commonly used on Microsoft Windows.

In this example, the PKCS #12 (PFX) file is named `keystore.pfx`:

```
$ openssl pkcs12 -in keystore.pfx -nokeys -out graylog-certificate.pem
$ openssl pkcs12 -in keystore.pfx -nocerts -out graylog-pkcs5.pem
$ openssl pkcs8 -in graylog-pkcs5.pem -topk8 -out graylog-key.pem
```

The resulting `graylog-certificate.pem` and `graylog-key.pem` can be used in the Graylog configuration file.

Converting an existing Java Keystore to private key/certificate pair

This section describes how to export a private key and certificate from an existing Java KeyStore in JKS format.

The starting point is an existing Java KeyStore in JKS format which contains a private key and certificate which should be used in Graylog:

```
$ keytool -list -v -keystore keystore.jks -alias graylog.example.com
Enter keystore password:
Alias name: graylog.example.com
Creation date: May 10, 2016
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=graylog.example.com, OU=Unknown, O="Graylog, Inc.", L=Hamburg, ST=Hamburg, C=DE
Issuer: CN=graylog.example.com, OU=Unknown, O="Graylog, Inc.", L=Hamburg, ST=Hamburg, C=DE
Serial number: 2b33832d
Valid from: Tue May 10 10:02:34 CEST 2016 until: Mon Aug 08 10:02:34 CEST 2016
Certificate fingerprints:
    MD5:  8A:3D:9F:ED:69:93:1B:6C:E3:29:66:EA:82:8D:42:BE
    SHA1: 5B:27:92:25:46:36:BC:F0:82:8F:9A:30:D8:50:D0:ED:32:4D:C6:A0
    SHA256: 11:11:77:F5:F6:6A:20:A8:E6:4A:5D:B5:20:21:4E:B8:FE:B6:38:1D:45:6B:ED:D0:7B:CE:B8:C8:B0
    Signature algorithm name: SHA256withRSA
    Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: AC 79 64 9F A1 60 14 B9    51 F4 F5 0B B3 B5 02 A5    .yd..`..Q.....
0010: B8 07 DC 7B                ....
]
]
```

The Java KeyStore in JKS format has to be converted to a PKCS#12 keystore, so that OpenSSL can work with it:

```
$ keytool -importkeystore -srckeystore keystore.jks -destkeystore keystore.p12 -deststoretype PKCS12
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
```

```
Entry for alias graylog.example.com successfully imported.
Import command completed: 1 entries successfully imported, 0 entries failed or cancelled
```

After the keystore has been successfully converted into PKCS#12 format, OpenSSL can export the X.509 certificate with PEM encoding:

```
$ openssl pkcs12 -in keystore.p12 -nokeys -out graylog-certificate.pem
Enter Import Password:
MAC verified OK
```

The private key can only be exported in PKCS#5 format with PEM encoding:

```
$ openssl pkcs12 -in keystore.p12 -nocerts -out graylog-pkcs5.pem
Enter Import Password:
MAC verified OK
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Graylog currently only supports PKCS#8 private keys with PEM encoding, so OpenSSL has to convert it into the correct format:

```
$ openssl pkcs8 -in graylog-pkcs5.pem -topk8 -out graylog-key.pem
Enter pass phrase for graylog-pkcs5.pem:
Enter Encryption Password:
Verifying - Enter Encryption Password:
```

The working directory should now contain the PKCS#8 private key (graylog-key.pem) and the X.509 certificate (graylog-certificate.pem) to be used with Graylog:

```
$ head graylog-key.pem graylog-certificate.pem
==> graylog-key.pem <==
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIE6TABBgqhkiG9w0BBQMwDgQIwMhLa5bw9vgCaggABIIEyN42AeYJJNBEiqhI
mWqJDot4Jokw2vB4abcIJ5Do4+7tjtMrecVRCDsvBZzjkXjnbuMBHEoxexe5f0/z
wgq6f/UDyTM3uKYQTG91fcqTyMDUlo3Wc8OqSqsNehOAQzA7hMCehqgNJHO0Zfny
EFvrXHurJWi4eA9vLRup86dbm4Wp3o8pmjOLduXieHfcgVtm5jfd7XfL5cRFS8kS
bSFH4v8xDxLNaJmKkK19gPCACMRbO9nGk/Z9q9N8zkj+xG91xlNRMX51SRzg20E0
nyyKTb39tJF35zjroB2HfiFWyrPQ1uF6yGoroGvu0L3eWosjBLjdRs0eBgjJcm5P
ic9zSVqMH6/4CPKJqvB97vP4QhpYcr9jlYJsbN6Zg40IELpM00VLvp0yU9tqTuRR
TDPYA1NMLZ2RrV52CEsh3zO21WHM7r187x4WHgprDFnjkXf02DrFhgCsGwkEQnb3
vj86q13RHhqoXT4W0zugvcv2/NBLMv0HNQBAfEK3X1YBmtQpEJhwSxesza1i7CpU

==> graylog-certificate.pem <==
Bag Attributes
    friendlyName: graylog.example.com
    localKeyID: 54 69 6D 65 20 31 34 36 32 38 36 37 38 32 33 30 39 32
subject=/C=DE/ST=Hamburg/L=Hamburg/O=Graylog, Inc./OU=Unknown/CN=graylog.example.com
issuer=/C=DE/ST=Hamburg/L=Hamburg/O=Graylog, Inc./OU=Unknown/CN=graylog.example.com
-----BEGIN CERTIFICATE-----
MIIDkTCCAnmgAwIBAgIEKzODLTANBgqhkiG9w0BAQsFAADB5MQswCQYDVQQGEwJE
RTEQMA4GA1UECBMHSGFtYnV5ZzEQMA4GA1UEBxMHSGFtYnV5ZzEQMA4GA1UEChMN
R3JheWxvZyZwSW5jLjEjEQMA4GA1UECXMHVW5rbm93bW5jEjEjEQMA4GA1UEAxMTZ3JheWxv
Zy5leGFtcGxlLnNvbTAeFw0xNjA1MTAwODAyMzRaFw0xNjA1MDgwODAyMzRaMHkx
```

The resulting PKCS#8 private key (graylog-key.pem) and the X.509 certificate (graylog-certificate.pem) can now be used to enable encrypted connections with Graylog by enabling TLS for the Graylog REST API and the web interface in the Graylog configuration file:

```
# Enable HTTPS support for the REST API. This secures the communication with the REST API
# using TLS to prevent request forgery and eavesdropping.
rest_enable_tls = true

# The X.509 certificate chain file in PEM format to use for securing the REST API.
rest_tls_cert_file = /path/to/graylog-certificate.pem

# The PKCS#8 private key file in PEM format to use for securing the REST API.
rest_tls_key_file = /path/to/graylog-key.pem

# The password to unlock the private key used for securing the REST API.
rest_tls_key_password = secret

# Enable HTTPS support for the web interface. This secures the communication the web interface
# using TLS to prevent request forgery and eavesdropping.
web_enable_tls = true

# The X.509 certificate chain file in PEM format to use for securing the web interface.
web_tls_cert_file = /path/to/graylog-certificate.pem

# The PKCS#8 private key file in PEM format to use for securing the web interface.
web_tls_key_file = /path/to/graylog-key.pem

# The password to unlock the private key used for securing the web interface.
web_tls_key_password = secret
```

Sample files

This section show the difference between following private key formats with samples.

PKCS#5 plain private key:

```
====-BEGIN RSA PRIVATE KEY====-
MIIBOwIBAAJBANxtmQ1Kccdp7HBNt8zgTai48Vv617bj4SnhkcmN99sCQ2Naj/sp
[...]
NiCYNLiCawBbpZnYw/ztpVACK4EwOpUy+u19cMB0JA==
====-END RSA PRIVATE KEY====-
```

PKCS#8 plain private key:

```
====-BEGIN PRIVATE KEY====-
MIIBVAIBADANBgqhkiG9w0BAQEFAASCAT4wggE6AgEAAkEA6GZN0rQFKRIVaPoz
[...]
LaLGdd9G63kLg85eldSy55uIAXsvqQIgfSYaliVtSbAgyx1Yfs3hJ+CTpNKzTNv/
Fx80EltYV6k=
====-END PRIVATE KEY====-
```

PKCS#5 encrypted private key:

```
====-BEGIN RSA PRIVATE KEY====-
Proc-Type: 4, ENCRYPTED
DEK-Info: DES-EDE3-CBC, E83B4019057F55E9

iIPs59nQn4RSd7ppch9/vNE7PfRSHLoQFmaAjaF0DxjV9oucznUjJq2gphAB2E2H
[...]
y5IT1MZPgN3LNkVSSLPWko08uFZQdfu0JTKcn7NPYRc=
====-END RSA PRIVATE KEY====-
```

PKCS#8 encrypted private key:

```

=====BEGIN ENCRYPTED PRIVATE KEY=====
MIIBpjBAbGkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIU9Y9p2EfWucCAggA
[...]
IjsZNp6zmlqf/RXnETsJjGd0TXRWaEdu+XOOyVyPskX2177X9DUJoD31
=====END ENCRYPTED PRIVATE KEY=====

```

Adding a self-signed certificate to the JVM trust store

Graylog nodes inside a cluster need to communicate with each other using the Graylog REST API. When using HTTPS for the Graylog REST API, the X.509 certificate must be *trusted* by the JVM trust store (similar to the trusted CA bundle in an operating system), otherwise communication will fail.

Important: If you are using different X.509 certificates for each Graylog node, you have to add *all of them* into the JVM trust store of each Graylog node.

The default trust store of an installed Java runtime environment can be found at `$JAVA_HOME/jre/lib/security/cacerts`. In order not to “pollute” the official trust store, we make a copy of it which we will use with Graylog instead:

```
$ cp -a "${JAVA_HOME}/jre/lib/security/cacerts" /path/to/cacerts.jks
```

After the original key store file has been copied, we can add the self-signed certificate (`cert.pem`, see [Creating a self-signed private key/certificate](#)) to the key store (the default password is `changeit`):

```

$ keytool -importcert -keystore /path/to/cacerts.jks -storepass changeit -alias graylog-self-signed -
Owner: CN=graylog.example.com, O="Graylog, Inc.", L=Hamburg, ST=Hamburg, C=DE
Issuer: CN=graylog.example.com, O="Graylog, Inc.", L=Hamburg, ST=Hamburg, C=DE
Serial number: 8c80134cee556734
Valid from: Tue Jun 14 16:38:17 CEST 2016 until: Wed Jun 14 16:38:17 CEST 2017
Certificate fingerprints:
    MD5:  69:D1:B3:01:46:0D:E9:45:FB:C6:6C:69:EA:38:ED:3E
    SHA1: F0:64:D0:1B:3B:6B:C8:01:D5:4D:33:36:87:F0:FB:10:E1:36:21:9E
    SHA256: F7:F2:73:3D:86:DC:10:22:1D:14:B8:5D:66:B4:EB:48:FD:3D:74:89:EC:C4:DF:D0:D2:EC:F8:5D:7
Signature algorithm name: SHA1withRSA
Version: 3

Extensions:

[Other details about the certificate...]

Trust this certificate? [no]: yes
Certificate was added to keystore

```

To verify that the self-signed certificate has indeed been added, it can be listed with the following command:

```

$ keytool -keystore /path/to/cacerts.jks -storepass changeit -list | grep graylog-self-signed -A1
graylog-self-signed, Jun 14, 2016, trustedCertEntry,
Certificate fingerprint (SHA1): F0:64:D0:1B:3B:6B:C8:01:D5:4D:33:36:87:F0:FB:10:E1:36:21:9E

```

The printed certificate fingerprint (SHA1) should match the one printed when importing the self-signed certificate.

In order for the JVM to pick up the new trust store, it has to be started with the JVM parameter `-Djavax.net.ssl.trustStore=/path/to/cacerts.jks`. If you’ve been using another password to encrypt the JVM trust store than the default `changeit`, you additionally have to set the JVM parameter `-Djavax.net.ssl.trustStorePassword=secret`.

Most start and init scripts for Graylog provide a `JAVA_OPTS` variable which can be used to pass the `javax.net.ssl.trustStore` and (optionally) `javax.net.ssl.trustStorePassword` system properties.

Disabling specific TLS ciphers and algorithms

Since [Java 7u76](#) it is possible to disable specific TLS algorithms and ciphers for secure connections.

In order to disable specific TLS algorithms and ciphers, you need to provide a properties file with a list of disabled algorithms and ciphers. Take a look at the example [security.properties](#) in the Graylog source repository.

For example, if you want to disable all algorithms except for TLS 1.2, the properties file has to contain the following line:

```
jdk.tls.disabledAlgorithms=SSLv2Hello, SSLv3, TLSv1, TLSv1.1
```

If additionally you want to disable DSA/RSA key sizes lower than 2048 bits and EC key sizes lower than 160 bits, the properties file has to contain the following line:

```
jdk.tls.disabledAlgorithms=SSLv2Hello, SSLv3, TLSv1, TLSv1.1, EC keySize < 160, RSA keySize < 2048, I
```

To load the properties file into a JVM, you have to pass it to Java using the `java.security.properties` system property:

```
java -Djava.security.properties=/path/to/security.properties -jar /path/to/graylog.jar server
```

Most start and init scripts for Graylog provide a `JAVA_OPTS` variable which can be used to pass the `java.security.properties` system property.

Further reading

- <https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html#DisabledAlgorithms>
- <http://www.oracle.com/technetwork/java/javase/7u76-relnotes-2389087.html>
- http://bugs.java.com/bugdatabase/view_bug.do?bug_id=7133344
- <https://tersesystems.com/2014/01/13/fixing-the-most-dangerous-code-in-the-world/>

Multi-node Setup

This guide doesn't provide a step-by-step tutorial for building a multi-node Graylog cluster but does simply give some advice for questions that might arise during the setup.

It's important for such a project that you understand each step in the setup process and do some planning upfront. Without a proper roadmap of all the things you want to achieve with a Graylog cluster, you will be lost on the way.

Graylog should be the last component you install in this setup. Its dependencies, namely MongoDB and Elasticsearch, have to be up and running first.

Important: This guide doesn't include instructions for running a multi-node Graylog cluster in an untrusted network. We assume that the connection between the hosts is trusted and doesn't have to be secured individually.

Prerequisites

Every server which is part of this setup should have the software requirements installed to run the targeted software. All software requirements can be found in the installation manual.

We highly recommend that the system time on all systems is kept in sync via NTP or a similar mechanism. Needless to say that DNS resolution must be working, too. Because everything is a freaking DNS problem.

In order to simplify the installation process, the servers should have a working Internet connection.

MongoDB replica set

We recommend to [deploy a MongoDB replica set](#).

MongoDB doesn't have to run on dedicated servers for the workload generated by Graylog, but you should follow the recommendations given in the MongoDB documentation about architecture. Most important is that you have an odd number of MongoDB servers in the replica set.

In most setups, each Graylog server will also host an instance of MongoDB which is part of the same replica set and shares the data with all other nodes in the cluster.

Note: To avoid unauthorized access to your MongoDB database, the [MongoDB replica set should be setup with authentication](#).

The correct order of working steps should be as follows:

1. Create the replica set (`rs01`)
2. Create the database (`graylog`)
3. Create a user account for accessing the database, which has the roles `readWrite` and `dbAdmin`.

Elasticsearch cluster

The [Elasticsearch setup documentation](#) should help you to install Elasticsearch with a robust base configuration.

It is important to name the Elasticsearch cluster not simply named *elasticsearch* to avoid accidental conflicts with Elasticsearch nodes using the default configuration. Just choose anything else (we recommend *graylog*), because this is the default name and any Elasticsearch instance that is started in the same network will try to connect to this cluster.

Note: Graylog currently doesn't work with Elasticsearch clusters using the [Shield](#) plugin. Your Elasticsearch cluster need to be secured by design (e. g. segregating network access).

Graylog Multi-node

After the installation of Graylog, you should take care that only one Graylog node is configured to be master with the configuration setting `is_master = true`.

The URI configured in `rest_listen_uri` (or `rest_transport_uri`) must be accessible for all Graylog nodes of the cluster.

Graylog to MongoDB connection

The `mongodb_uri` configuration setting must include all MongoDB nodes forming the replica set, the name of the replica set, as well as the previously configured user account with access to the replica set. The configuration setting is a normal [MongoDB connection string](#).

Finally, the MongoDB connection string in the Graylog configuration file should look like this:

```
mongodb_uri = mongodb://USERNAME:PASSWORD@mongodb-node01:27017,mongodb-node02:27017,mongodb-node03:27017
```

Graylog to Elasticsearch connection

Graylog will connect as [client node](#) to the Elasticsearch Cluster.

To avoid issues with the connection to the Elasticsearch cluster you should add some of the network addresses of the Elasticsearch nodes to `elasticsearch_discovery_zen_ping_unicast_hosts`.

Additionally, `elasticsearch_network_host` must be set to a network interface which can be accessed by the other Elasticsearch nodes in the Elasticsearch cluster.

Graylog web interface

By default, the web interface can be used on every instance of Graylog which hasn't disabled it with the configuration setting `web_enable = false`.

It's possible to use a [loadbalancer](#) in front of all Graylog servers, please refer to [Making the web interface work with load balancers/proxies](#) for more details.

Depending on your setup, it's possible to either use a hardware loadbalancer for TLS/HTTPS termination, a [reverse proxy](#), or to simply enable it [in the Graylog node](#).

Scaling

Each component in this multi-node setup can be scaled on the individual needs.

Depending on the amount of messages ingested and how long messages should be available for direct search, the Elasticsearch cluster will need most of the resources on your setup.

Keep an eye on your Elasticsearch cluster with plugins like [Elastic HQ](#) or [Kopf](#). Those will help you to understand the Elasticsearch cluster health and behavior.

Graylog Metrics should be monitored [with the Graylog Metrics Reporter plugins](#) which are able to send the internal Graylog metrics to your favorite metrics collector (e. g. Graphite or Prometheus).

Up until today, we have almost never faced the issue that the MongoDB replica set needed special attention. But of course you should still monitor it and store its metrics - just to be sure.

Troubleshooting

- After every configuration change or service restart, watch the logfile of the applications you have worked on. Sometimes other log files can also give you hints about what went wrong. For example if you're configuring Graylog and try to find out why the connection to the MongoDB isn't working, the MongoDB logs can help to identify the problem.
- If [HTTPS has been enabled for the Graylog REST API](#), it need to be setup for the Graylog web interface, too.

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.

Important: Graylog currently does not work with Elasticsearch clusters using the License or [Shield](#) plugin.

Warning: Graylog 2.0.x currently **does not** work with Elasticsearch 2.4.x. The latest supported version is [Elasticsearch 2.3.5](#).

Elasticsearch versions

Graylog hosts an embedded Elasticsearch node which is joining the Elasticsearch cluster as a client node.

The following table provides an overview over the Elasticsearch version in Graylog:

Graylog version	Elasticsearch version
1.2.0-1.2.1	1.7.1
1.3.0-1.3.3	1.7.3
1.3.4	1.7.5
2.0.0	2.3.1
2.0.1-2.0.3	2.3.2

Caution: Graylog 2.0.x **does not** work with Elasticsearch 2.4.x or 5.x!

Configuration

Graylog

The most important settings to make a successful connection are the Elasticsearch cluster name, one or more addresses of Elasticsearch master nodes, and the local network bind address.

Graylog needs to know the address of at least one other Elasticsearch master node given in the `elasticsearch_discovery_zen_ping_unicast_hosts` setting. Vice versa, the Elasticsearch nodes need to be able to access the embedded Elasticsearch node in Graylog via the interface given in the `elasticsearch_network_host` setting.

Cluster Name

You need to tell Graylog which Elasticsearch cluster to join. The Elasticsearch default [cluster name](#) is `elasticsearch` and configured for every Elasticsearch node in the `elasticsearch.yml` configuration file with the `cluster.name` name.

Configure the same cluster name in every Graylog configuration file (e. g. `graylog.conf`) with the `elasticsearch_cluster_name` setting (default: `graylog`).

We recommend to call the cluster `graylog-production` or `graylog`, but not `elasticsearch` to prevent accidental cluster name collisions.

The Elasticsearch configuration file is typically located at `/etc/elasticsearch/elasticsearch.yml`.

Network setup

Graylog is using unicast discovery to find all the Elasticsearch nodes in the cluster.

In order for this to work, Graylog has to know some master nodes of the Elasticsearch cluster which can be provided in the `elasticsearch_discovery_zen_ping_unicast_hosts` configuration setting.

For example, add the following lines to your Graylog configuration file for an Elasticsearch cluster which includes the 2 Elasticsearch master nodes `es-node-1.example.org` and `es-node-2.example.org`:

```
# List of Elasticsearch master nodes to connect to
elasticsearch_discovery_zen_ping_unicast_hosts = es-node-1.example.org:9300,es-node-2.example.org:9300
```

Additionally, Graylog has to use a network interface for the embedded Elasticsearch node which the other Elasticsearch nodes in the cluster can connect to:

```
# Public IP address or host name of the Graylog node, accessible for the other Elasticsearch nodes
elasticsearch_network_host = 198.51.100.23
```

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` settings 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`.

Last but not least, make sure that Elasticsearch is binding to a network interface that Graylog can connect to (see `network.host` and [Commonly Used Network Settings](#)).

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 the following settings to the `elasticsearch.yml` file to disable the dynamic scripting feature and prevent possible remote code executions:

```
script.inline: false
script.indexed: false
script.file: false
```

Details about dynamic scripting can be found in [the reference documentation of Elasticsearch](#).

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. But please take care that you **don't cross 32 GB!**

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 for every index it is managing. This is enough tuning for a lot of use cases and setups.

A more detailed guide about tuning Elasticsearch will be published at a later time.

Avoiding split-brain and shard shuffling

Split-brain events

Elasticsearch sacrifices consistency in order to ensure availability, and partition tolerance. The reasoning behind that is that short periods of misbehaviour are less problematic than short periods of unavailability. In other words, when

Elasticsearch nodes in a cluster are unable to replicate changes to data, they will keep serving applications such as Graylog. When the nodes are able to replicate their data, they will attempt to converge the replicas and to achieve *eventual consistency*.

Elasticsearch tackles the previous by electing master nodes, which are in charge of database operations such as creating new indices, moving shards around the cluster nodes, and so forth. Master nodes coordinate their actions actively with others, ensuring that the data can be converged by non-masters. The cluster nodes that are not master nodes are not allowed to make changes that would break the cluster.

The previous mechanism can in some circumstances fail, causing a **split-brain event**. When an Elasticsearch cluster is split into two sides, both thinking they are the master, data consistency is lost as the masters work independently on the data. As a result the nodes will respond differently to same queries. This is considered a catastrophic event, because the data from two masters can not be rejoined automatically, and it takes quite a bit of manual work to remedy the situation.

Avoiding split-brain events

Elasticsearch nodes take a simple majority vote over who is master. If the majority agrees that they are the master, then most likely the disconnected minority has also come to conclusion that they can not be the master, and everything is just fine. This mechanism requires at least 3 nodes to work reliably however, because one or two nodes can not form a majority.

The minimum amount of master nodes required to elect a master must be configured manually in `elasticsearch.yml`:

```
# At least NODES/2+1 on clusters with NODES > 2, where NODES is the number of master nodes in the cluster
discovery.zen.minimum_master_nodes: 2
```

The configuration values should typically for example:

Master nodes	mini-mum_master_nodes	Comments
1	1	
2	1	With 2 the other node going down would stop the cluster from working!
3	2	
4	3	
5	3	
6	4	

Some of the master nodes may be *dedicated master nodes*, meaning they are configured just to handle lightweight operational (cluster management) responsibilities. They will not handle or store any of the cluster's data. The function of such nodes is similar to so called *witness servers* on other database products, and setting them up on dedicated witness sites will greatly reduce the chance of Elasticsearch cluster instability.

A dedicated master node has the following configuration in `elasticsearch.yml`:

```
node.data: false
node.master: true
```

Shard shuffling

When cluster status changes, for example because of node restarts or availability issues, Elasticsearch will start automatically rebalancing the data in the cluster. The cluster works on making sure that the amount of shards and replicas will conform to the cluster configuration. This is a problem if the status changes are just temporary. Moving shards and replicas around in the cluster takes considerable amount of resources, and should be done only when necessary.

Avoiding unnecessary shuffling

Elasticsearch has couple configuration options, which are designed to allow short times of unavailability before starting the recovery process with shard shuffling. There are 3 settings that may be configured in `elasticsearch.yml`:

```
# Recover only after the given number of nodes have joined the cluster. Can be seen as "minimum number of nodes"
gateway.recover_after_nodes: 8
# Time to wait for additional nodes after recover_after_nodes is met.
gateway.recover_after_time: 5m
# Inform Elasticsearch how many nodes form a full cluster. If this number is met, start up immediately.
gateway.expected_nodes: 10
```

The configuration options should be set up so that only *minimal* node unavailability is tolerated. For example server restarts are common, and should be done in managed manner. The logic is that if you lose large part of your cluster, you probably should start re-shuffling the shards and replicas without tolerating the situation.

Custom index mappings

Sometimes it's useful to not rely on Elasticsearch's [dynamic mapping](#) but to define a stricter schema for messages.

Note: If the index mapping is conflicting with the actual message to be sent to Elasticsearch, indexing that message will fail.

Graylog itself is using a default mapping which includes settings for the `timestamp`, `message`, `full_message`, and `source` fields of indexed messages:

```
$ curl -X GET 'http://localhost:9200/_template/graylog-internal?pretty'
{
  "graylog-internal" : {
    "order" : -2147483648,
    "template" : "graylog_*",
    "settings" : { },
    "mappings" : {
      "message" : {
        "_ttl" : {
          "enabled" : true
        },
        "_source" : {
          "enabled" : true
        },
        "dynamic_templates" : [ {
          "internal_fields" : {
            "mapping" : {
              "index" : "not_analyzed",
              "type" : "string"
            },
            "match" : "gl2_*"
          }
        }, {
          "store_generic" : {
            "mapping" : {
              "index" : "not_analyzed"
            },
            "match" : "*"
          }
        }
      ]
    }
  }
}
```

```
    } ],
    "properties" : {
      "full_message" : {
        "analyzer" : "standard",
        "index" : "analyzed",
        "type" : "string"
      },
      "streams" : {
        "index" : "not_analyzed",
        "type" : "string"
      },
      "source" : {
        "analyzer" : "analyzer_keyword",
        "index" : "analyzed",
        "type" : "string"
      },
      "message" : {
        "analyzer" : "standard",
        "index" : "analyzed",
        "type" : "string"
      },
      "timestamp" : {
        "format" : "yyyy-MM-dd HH:mm:ss.SSS",
        "type" : "date"
      }
    }
  },
  "aliases" : { }
}
```

In order to extend the default mapping of Elasticsearch and Graylog, you can create one or more custom index mappings and add them as index templates to Elasticsearch.

Let's say we have a schema for our data like the following:

Field Name	Field Type	Example
http_method	string	GET
http_response_code	long	200
ingest_time	date	2016-06-13T15:00:51.927Z
took_ms	long	56

This would translate to the following additional index mapping in Elasticsearch:

```
"mappings" : {
  "message" : {
    "properties" : {
      "http_method" : {
        "type" : "string",
        "index" : "not_analyzed"
      },
      "http_response_code" : {
        "type" : "long"
      },
      "ingest_time" : {
        "type" : "date",
        "format": "strict_date_time"
      }
    }
  },
}
```



```

    "took_ms" : {
      "type" : "long"
    }
  }
}

```

The format of the `ingest_time` field is described in the Elasticsearch documentation about the [format mapping parameter](#). Also make sure to check the Elasticsearch documentation about [Field datatypes](#).

In order to apply the additional index mapping when Graylog creates a new index in Elasticsearch, it has to be added to an [index template](#). The Graylog default template (`graylog-internal`) has the lowest priority and will be merged with the custom index template by Elasticsearch.

Warning: If the default index mapping and the custom index mapping cannot be merged (e. g. because of conflicting field datatypes), Elasticsearch will throw an exception and won't create the index. So be extremely cautious and conservative about the custom index mappings!

Creating a new index template

Save the following index template for the custom index mapping into a file named `graylog-custom-mapping.json`:

```

{
  "template": "graylog_*",
  "mappings": {
    "message": {
      "properties": {
        "http_method": {
          "type": "string",
          "index": "not_analyzed"
        },
        "http_response_code": {
          "type": "long"
        },
        "ingest_time": {
          "type": "date",
          "format": "strict_date_time"
        },
        "took_ms": {
          "type": "long"
        }
      }
    }
  }
}

```

Finally, load the index mapping into Elasticsearch with the following command:

```

$ curl -X PUT -d @'graylog-custom-mapping.json' 'http://localhost:9200/_template/graylog-custom-mapping'
{
  "acknowledged" : true
}

```

Every Elasticsearch index created from that time on, will have an index mapping consisting of the original `graylog-internal` index template and the new `graylog-custom-mapping` template:

```
$ curl -X GET 'http://localhost:9200/graylog_deflector/_mapping?pretty'
{
  "graylog_2" : {
    "mappings" : {
      "message" : {
        "_ttl" : {
          "enabled" : true
        },
        "dynamic_templates" : [ {
          "internal_fields" : {
            "mapping" : {
              "index" : "not_analyzed",
              "type" : "string"
            },
            "match" : "gl2_*"
          }
        }, {
          "store_generic" : {
            "mapping" : {
              "index" : "not_analyzed"
            },
            "match" : "*"
          }
        }
      ],
      "properties" : {
        "full_message" : {
          "type" : "string",
          "analyzer" : "standard"
        },
        "http_method" : {
          "type" : "string",
          "index" : "not_analyzed"
        },
        "http_response_code" : {
          "type" : "long"
        },
        "ingest_time" : {
          "type" : "date",
          "format" : "strict_date_time"
        },
        "message" : {
          "type" : "string",
          "analyzer" : "standard"
        },
        "source" : {
          "type" : "string",
          "analyzer" : "analyzer_keyword"
        },
        "streams" : {
          "type" : "string",
          "index" : "not_analyzed"
        },
        "timestamp" : {
          "type" : "date",
          "format" : "yyyy-MM-dd HH:mm:ss.SSS"
        },
        "took_ms" : {
          "type" : "long"
        }
      }
    }
  }
}
```

```

    }
  }
}
}
}

```

Deleting custom index templates

If you want to remove an existing index template from Elasticsearch, simply issue a `DELETE` request to Elasticsearch:

```

$ curl -X DELETE 'http://localhost:9200/_template/graylog-custom-mapping?pretty'
{
  "acknowledged" : true
}

```

After you've removed the index template, new indices will only have the original index mapping:

```

$ curl -X GET 'http://localhost:9200/graylog_deflector/_mapping?pretty'
{
  "graylog_3" : {
    "mappings" : {
      "message" : {
        "_ttl" : {
          "enabled" : true
        },
        "dynamic_templates" : [ {
          "internal_fields" : {
            "mapping" : {
              "index" : "not_analyzed",
              "type" : "string"
            },
            "match" : "gl2_*"
          }
        }, {
          "store_generic" : {
            "mapping" : {
              "index" : "not_analyzed"
            },
            "match" : "*"
          }
        }
      ],
      "properties" : {
        "full_message" : {
          "type" : "string",
          "analyzer" : "standard"
        },
        "message" : {
          "type" : "string",
          "analyzer" : "standard"
        },
        "source" : {
          "type" : "string",
          "analyzer" : "analyzer_keyword"
        },
        "streams" : {
          "type" : "string",

```

```
        "index" : "not_analyzed"
      },
      "timestamp" : {
        "type" : "date",
        "format" : "yyyy-MM-dd HH:mm:ss.SSS"
      }
    }
  }
}
```

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 have been 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.

In most cases, this can be solved by adding another Elasticsearch node to the cluster or by reducing the replication factor of the indices (which means less resiliency against node outages, though).

GREEN

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

Backup

When it comes to backup in a Graylog setup it is not easy to answer. You need to consider what type of backup will suite your needs.

Your Graylog Server setup and settings are easy to backup with a [MongoDB dump](#) and a filesystem backup of all configuration files.

The data within your Elasticsearch Cluster can take the advantage of the [Snapshot and Restore](#) function that are offered by Elasticsearch.

Default file locations

Each installation flavor of Graylog will place the configuration files into a specific location on the local files system. The goal of this section is to provide a short overview about the most common and most important default file locations.

DEB package

This paragraph covers Graylog installations on Ubuntu Linux, Debian Linux, and Debian derivatives installed with the *DEB package*.

Graylog

	File system path
Configuration	/etc/graylog/server/server.conf
Logging configuration	/etc/graylog/server/log4j2.xml
Plugins	/usr/share/graylog-server/plugin
JVM settings	/etc/default/graylog-server
Message journal files	/var/lib/graylog-server/journal
Log Files	/var/log/graylog-server/

Elasticsearch

Note: These are only the most common file locations. Please refer to the [Elasticsearch documentation](#) for a comprehensive list of default file locations.

	File system path
Configuration	/etc/elasticsearch
JVM settings	/etc/default/elasticsearch
Data files	/var/lib/elasticsearch/data
Log files	/var/log/elasticsearch/

MongoDB

	File system path
Configuration	/etc/mongod.conf
Data files	/var/lib/mongodb/
Log files	/var/log/mongodb/

RPM package

This paragraph covers Graylog installations on Fedora Linux, Red Hat Enterprise Linux, CentOS Linux, and other Red Hat Linux derivatives installed with the *RPM package*.

Graylog

	File system path
Configuration	/etc/graylog/server/server.conf
Logging configuration	/etc/graylog/server/log4j2.xml
Plugins	/usr/share/graylog-server/plugin
JVM settings	/etc/sysconfig/graylog-server
Message journal files	/var/lib/graylog-server/journal
Log Files	/var/log/graylog-server/

Elasticsearch

Note: These are only the most common file locations. Please refer to the [Elasticsearch documentation](#) for a comprehensive list of default file locations.

	File system path
Configuration	/etc/elasticsearch
JVM settings	/etc/sysconfig/elasticsearch
Data files	/var/lib/elasticsearch/
Log files	/var/log/elasticsearch/

MongoDB

	File system path
Configuration	/etc/mongod.conf
Data files	/var/lib/mongodb/
Log files	/var/log/mongodb/

Omnibus package

This paragraph covers Graylog installations via OVA, on AWS (via AMI), and on Openstack using the [Graylog Omnibus package](#).

Graylog

	File system path
Configuration	/opt/graylog/conf/graylog.conf
Logging configuration	/opt/graylog/conf/log4j2.xml
Plugins	/opt/graylog/plugin
JVM settings	<i>/etc/graylog/graylog-settings.json</i>
Message journal files	/var/opt/graylog/data/journal
Log files	/var/log/graylog/server/

Elasticsearch

Note: These are only the most common file locations. Please refer to the [Elasticsearch documentation](#) for a comprehensive list of default file locations.

	File system path
Configuration	/opt/graylog/conf/elasticsearch/
JVM settings	<i>/etc/graylog/graylog-settings.json</i>
Data files	/var/opt/graylog/data/elasticsearch
Log files	/var/log/graylog/elasticsearch/

MongoDB

	File system path
Configuration	<code>/etc/graylog/graylog-settings.json</code>
Data files	<code>/var/opt/graylog/data/mongodb</code>
Log files	<code>/var/log/graylog/mongodb/</code>

Graylog REST API

The functionality Graylog REST API is very comprehensive; even the Graylog web interface is exclusively using Graylog REST API to interact with the Graylog cluster.

To connect to the Graylog REST API with a web browser, just add `api-browser` to your current `rest_listen_uri` setting or use the **API browser** button on the nodes overview page (*System / Nodes* in the web interface).

For example if your Graylog REST API is listening on `http://192.168.178.26:9000/api/`, the API browser will be available at `http://192.168.178.26:9000/api/api-browser/`.

The screenshot shows the Graylog web interface with the 'Nodes' page selected. The page title is 'Nodes' and it includes a sub-header: 'This page provides a real-time overview of the nodes in your Graylog cluster.' Below this, there is a warning icon and text: 'You can pause message processing at any time. The process buffers will not accept any new messages until you resume it. If the message journal is enabled for a node, which it is by default, incoming messages will be persisted to disk, even when processing is disabled.'

The main content area shows 'There are 3 active nodes'. The first node is highlighted with a red arrow pointing to its 'API browser' button. The node details are as follows:

- Node 1:** `71ab6aaa / gm-01-c.fritz.box` In 1 / Out 1 msg/s. The journal contains 1 unprocessed messages in 1 segment. 0 messages appended, 0 messages read in the last second. Current lifecycle state: Running. Message processing: Enabled. Load balancer indication: ALIVE. The JVM is using 803.8MB of 972.8MB heap space and will not attempt to use more than 972.8MB.
- Node 2:** `ed0ad32d / gm-01-d.fritz.box` In 0 / Out 0 msg/s. The journal contains 0 unprocessed messages in 1 segment. 0 messages appended, 0 messages read in the last second. Current lifecycle state: Running. Message processing: Enabled. Load balancer indication: ALIVE. The JVM is using 650.7MB of 972.8MB heap space and will not attempt to use more than 972.8MB.
- Node 3:** `58c57924 / gm-01-u.fritz.box` In 0 / Out 0 msg/s. The journal contains 0 unprocessed messages in 1 segment. 0 messages appended, 0 messages read in the last second. Current lifecycle state: Running. Message processing: Enabled. Load balancer indication: ALIVE. The JVM is using 824.9MB of 972.8MB heap space and will not attempt to use more than 972.8MB.

At the bottom of the page, there is a footer: 'Graylog 2.1.1+01d50e5 on gm-01-c.fritz.box (Oracle Corporation 1.8.0_102 on Linux 3.16.0-4-amd64)'.

Note: The customized version of Swagger UI used by Graylog does currently only work in Google Chrome and Firefox.

Using the API Browser

After providing the credentials (username and password), you can browse all available HTTP resources of the Graylog REST API.

Graylog REST API browser

192.168.178.26:9000/api/api-browser#/Cluster/get_get_0

GM provide username and password

AlarmCallbackHistories : Manage stream alarm callback histories Show/Hide List Operations Expand Operations Raw

AlarmCallbacks : Manage stream alarm callbacks Show/Hide List Operations Expand Operations Raw

AlertConditions : Manage stream alert conditions Show/Hide List Operations Expand Operations Raw

Alerts : Manage stream alerts for all streams Show/Hide List Operations Expand Operations Raw

Cluster : System information of all nodes in the cluster Show/Hide List Operations Expand Operations Raw

GET /cluster Get system overview of all Graylog nodes

Response Class

Model Model Schema

Map

Response Content Type application/json

Try it out! Hide Response

Request URL

http://192.168.178.26:9000/api/ccluster

Response Body

```
{
  "71ab6aaa-cb39-46be-9dac-4ba99fed3d66": {
    "facility": "graylog-server",
    "codename": "Smuttynose",
    "node_id": "71ab6aaa-cb39-46be-9dac-4ba99fed3d66",
    "cluster_id": "3adaf799-1551-4239-84e5-6ed939b56f62",
    "version": "2.1.1+01d50e5",
    "started_at": "2016-09-23T10:39:00.179Z",
    "hostname": "gm-01-c.fritz.box",
    "lifecycle": "running",
    "lb_status": "alive",
    "timezone": "Europe/Berlin",
    "operating_system": "Linux 3.10.0-327.28.3.el7.x86_64",
    "is_processing": true
  },
  "ed0ad32d-8776-4d25-be2f-a8956eecedcf": {
    "facility": "graylog-server",
    "codename": "Smuttynose",
    "node_id": "ed0ad32d-8776-4d25-be2f-a8956eecedcf",
    "cluster_id": "3adaf799-1551-4239-84e5-6ed939b56f62",
  },
}
```

Response Code

200

Response Headers

```
{"X-Graylog-Node-Id":"58c57924-808a-4fa7-be09-63ca551628cd","Date":"Fri, 14 Oct 2016 13:16:59 GMT","Content-Encoding":"..."}
```

GET /cluster/{nodeId}/jvm Get JVM information of the given node

GET /cluster/{nodeId}/threaddump Get a thread dump of the given node

Cluster/Deflector : Cluster-wide deflector handling Show/Hide List Operations Expand Operations Raw

Interacting with the Graylog REST API

While having a graphical UI for the Graylog REST API is perfect for interactive usage and exploratory learning, the real power unfolds when using the Graylog REST API for automation or integrating Graylog into another system, such as monitoring or ticket systems.

Naturally, the same operations the API browser offers can be used on the command line or in scripts. A very common HTTP client being used for this kind of interaction is `curl`.

Note: In the following examples, the username `GM` and password `superpower` will be used to demonstrate how to

work with the Graylog REST API running at `http://192.168.178.26:9000/api`.

The following command displays Graylog cluster information as JSON, exactly the same information the web interface is displaying on the *System / Nodes* page:

```
curl -u GM:superpower -H 'Accept: application/json' -X GET 'http://192.168.178.26:9000/api/cluster?pretty'
```

The Graylog REST API will respond with the following information:

```
{
  "71ab6aaa-cb39-46be-9dac-4ba99fed3d66" : {
    "facility" : "graylog-server",
    "codename" : "Smuttynose",
    "node_id" : "71ab6aaa-cb39-46be-9dac-4ba99fed3d66",
    "cluster_id" : "3adaf799-1551-4239-84e5-6ed939b56f62",
    "version" : "2.1.1+01d50e5",
    "started_at" : "2016-09-23T10:39:00.179Z",
    "hostname" : "gm-01-c.fritz.box",
    "lifecycle" : "running",
    "lb_status" : "alive",
    "timezone" : "Europe/Berlin",
    "operating_system" : "Linux 3.10.0-327.28.3.el7.x86_64",
    "is_processing" : true
  },
  "ed0ad32d-8776-4d25-be2f-a8956ecebdcf" : {
    "facility" : "graylog-server",
    "codename" : "Smuttynose",
    "node_id" : "ed0ad32d-8776-4d25-be2f-a8956ecebdcf",
    "cluster_id" : "3adaf799-1551-4239-84e5-6ed939b56f62",
    "version" : "2.1.1+01d50e5",
    "started_at" : "2016-09-23T10:40:07.325Z",
    "hostname" : "gm-01-d.fritz.box",
    "lifecycle" : "running",
    "lb_status" : "alive",
    "timezone" : "Europe/Berlin",
    "operating_system" : "Linux 3.16.0-4-amd64",
    "is_processing" : true
  },
  "58c57924-808a-4fa7-be09-63ca551628cd" : {
    "facility" : "graylog-server",
    "codename" : "Smuttynose",
    "node_id" : "58c57924-808a-4fa7-be09-63ca551628cd",
    "cluster_id" : "3adaf799-1551-4239-84e5-6ed939b56f62",
    "version" : "2.1.1+01d50e5",
    "started_at" : "2016-09-30T13:31:39.051Z",
    "hostname" : "gm-01-u.fritz.box",
    "lifecycle" : "running",
    "lb_status" : "alive",
    "timezone" : "Europe/Berlin",
    "operating_system" : "Linux 4.4.0-36-generic",
    "is_processing" : true
  }
}
```

Creating and using Access Token

For security reasons, using the username and password directly on the command line or in some third party application is undesirable.

To prevent having to use the clear text credentials, Graylog allows to create access tokens which can be used for authentication instead.

In order to create a new access token, you need to send a POST request to the Graylog REST API which includes the username and the name of the new access token.

The following example will create an access token named `icinga` for the user `GM`:

```
curl -u GM:superpower -H 'Accept: application/json' -X POST 'http://192.168.178.26:9000/api/users/GM/tokens'
```

The response will include the access token in the `token` field:

```
{
  "name" : "icinga",
  "token" : "htgi84ut7jpivsrcldd6l4lmcigvfauldm99ofcb4hsfcvdgsru",
  "last_access" : "1970-01-01T00:00:00.000Z"
}
```

The received access token can now be used as username in a request to the Graylog REST API using Basic Auth together with the literal password `token`.

Now the first `curl` example would look as follows:

```
curl -u htgi84ut7jpivsrcldd6l4lmcigvfauldm99ofcb4hsfcvdgsru:token -H 'Accept: application/json' -X GET 'http://192.168.178.26:9000/api/users/GM/tokens'
```

If you need to know which access tokens have already been created by a user, just use `GET /users/{username}/tokens/` on the Graylog REST API to request a list of all access tokens that are present for this user.

The following example will request all access tokens of the user `GM`:

```
curl -u GM:superpower -H 'Accept: application/json' -X GET 'http://192.168.178.26:9000/api/users/GM/tokens'
```

When an access token is no longer needed, it can be deleted on the Graylog REST API via `DELETE /users/{username}/tokens/{token}`.

The following example deletes the previously created access token `htgi84ut7jpivsrcldd6l4lmcigvfauldm99ofcb4hsfcvdgsru` of the user `GM`:

```
curl -u GM:superpower -H 'Accept: application/json' -X DELETE 'http://192.168.178.26:9000/api/users/GM/tokens/htgi84ut7jpivsrcldd6l4lmcigvfauldm99ofcb4hsfcvdgsru'
```

Creating and using Session Token

While access tokens can be used for permanent access, session tokens will expire after a certain time. The expiration time can be adjusted in the user's profile.

Getting a new session token can be obtained via POST request to the Graylog REST API. Username and password are required to get a valid session ID. The following example will create an session token for the user `GM`:

```
curl -i -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' 'http://192.168.178.26:9000/api/users/GM/sessions'
```

The response will include the session token in the field `session_id` and the time of expiration:

```
{
  "valid_until" : "2016-10-24T16:08:57.854+0000",
  "session_id" : "cf1df45c-53ea-446c-8ed7-e1df64861de7"
}
```

The received token can now be used as username in a request to the Graylog REST API using Basic Auth together with the literal password `session`.

Now a `curl` command to get a list of access tokens would look as follows:

```
curl -u cf1df45c-53ea-446c-8ed7-eldf64861de7:session -H 'Accept: application/json' -X GET 'http://192.168.1.100:5443/api/v1/access-tokens'
```


Securing Graylog

Logging user activity

Graylog has been built using a client-server architecture model in which the user interface retrieves all data via a collection of REST APIs. Thus logging relevant user activity, in other words an access log, is simply a matter of enabling a built-in feature. It logs all requests to the Graylog REST API and produces an access log augmented by additional information, like the user name, the remote address, and the user agent.

Configuring the Access Log

The Access Log is configured by adding an appender and logger to the [Log4j2 configuration file](#) (`log4j2.xml`). The following example demonstrates required additions on top of the normal logging configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration packages="org.graylog2.log4j" shutdownHook="disable">
  <Appenders>
    <!-- Simple appender that writes access log to specified file -->
    <File name="RestAccessLog" fileName="/var/log/graylog/server/restaccess.log" append="true">
      <PatternLayout pattern="%d %-5p: %c - %m%n"/>
    </File>
  </Appenders>
  <Loggers>
    <!-- RestAccessLogFilter -->
    <Logger name="org.graylog2.rest.accesslog" level="debug" additivity="false">
      <AppenderRef ref="RestAccessLog" level="debug"/>
      <AppenderRef ref="STDOUT" level="info"/>
    </Logger>
  </Loggers>
</Configuration>
```

The resulting log entries will look similar to the following messages:

```
2016-06-08 18:21:55,651 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-] "GET streams" Mo
2016-06-08 18:21:55,694 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-] "GET system/fie
2016-06-08 18:21:55,698 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-] "GET system/fie
2016-06-08 18:21:55,780 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-] "GET system/inpu
2016-06-08 18:21:56,021 DEBUG: org.graylog2.rest.accesslog - 192.168.122.1 admin [-] "GET search/univ
```

X-Forwarded-For HTTP header support

If there is a proxy server, load balancer, or a network device which hides the client's IP address from Graylog, it can read the information from a supplied X-Forwarded-For HTTP request header. Most HTTP-capable devices support setting such a (semi-) standard HTTP request header.

Since overriding the client address from an externally supplied HTTP request header opens the door for spoofing, the list of trusted proxy servers which are allowed to provide the X-Forwarded-For HTTP request header, can be configured using the `trusted_proxies` setting in the Graylog configuration file (`graylog.conf`):

```
# Comma separated list of trusted proxies that are allowed to set the client address with X-Forwarded-For
# header. May be subnets, or hosts.
trusted_proxies = 127.0.0.1/32, 0:0:0:0:0:0:0:1/128
```

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 [the Graylog Marketplace](#) can be imported using the Graylog web interface.

You can load and even create own content packs from the *System -> Content Packs* section of your Graylog web interface.

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

Sending syslog data from Linux hosts is [described on the Graylog Marketplace](#).

Sending syslog from MacOS X hosts

Sending log messages from MacOS X syslog daemons is easy. Just define a `graylog-server` instance as UDP log target by adding this line in your `/etc/syslog.conf`:

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

Now restart `syslogd`:

```
$ sudo launchctl unload /System/Library/LaunchDaemons/com.apple.syslogd.plist
$ sudo launchctl load /System/Library/LaunchDaemons/com.apple.syslogd.plist
```

Important: If `syslogd` was running as another user you might end up with multiple `syslogd` instances and strange behavior of the whole system. Please check that only one `syslogd` process is running:

```
$ ps aux | grep syslog
lennart      58775   0.0  0.0  2432768    592 s004  S+   6:10PM   0:00.00 grep syslog
root         58759   0.0  0.0  2478772   1020  ??   Ss    6:09PM   0:00.01 /usr/sbin/syslogd
```

That's it! Your MacOS X syslog messages should now appear in your Graylog system.

GELF / Sending from applications

The Graylog Extended Log Format (GELF) is a log format that avoids the shortcomings of classic plain syslog and is perfect to logging from your application layer. It comes with optional compression, chunking and most importantly a clearly defined structure. There are [dozens of GELF libraries](#) for many frameworks and programming languages to get you started.

Read more about [GELF in the specification](#).

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.com"}
```

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.

Using Apache Kafka as transport queue

Graylog supports [Apache Kafka](#) as a transport for various inputs such as GELF, syslog, and Raw/Plaintext inputs. The Kafka topic can be filtered by a regular expression and depending on the input, various additional settings can be configured.

Learn how to use `rsyslog` and Apache Kafka in the [Sending syslog via Kafka into Graylog](#) guide.

Using RabbitMQ (AMQP) as transport queue

Graylog supports [AMQP](#) as a transport for various inputs such as GELF, syslog, and Raw/Plaintext inputs. It can connect to any AMQP broker supporting [AMQP 0-9-1](#) such as [RabbitMQ](#).

Learn how to use rsyslog and RabbitMQ in the [Sending syslog via AMQP into Graylog](#) guide.

Microsoft Windows

Sending syslog data from Windows is [described on the Graylog Marketplace](#).

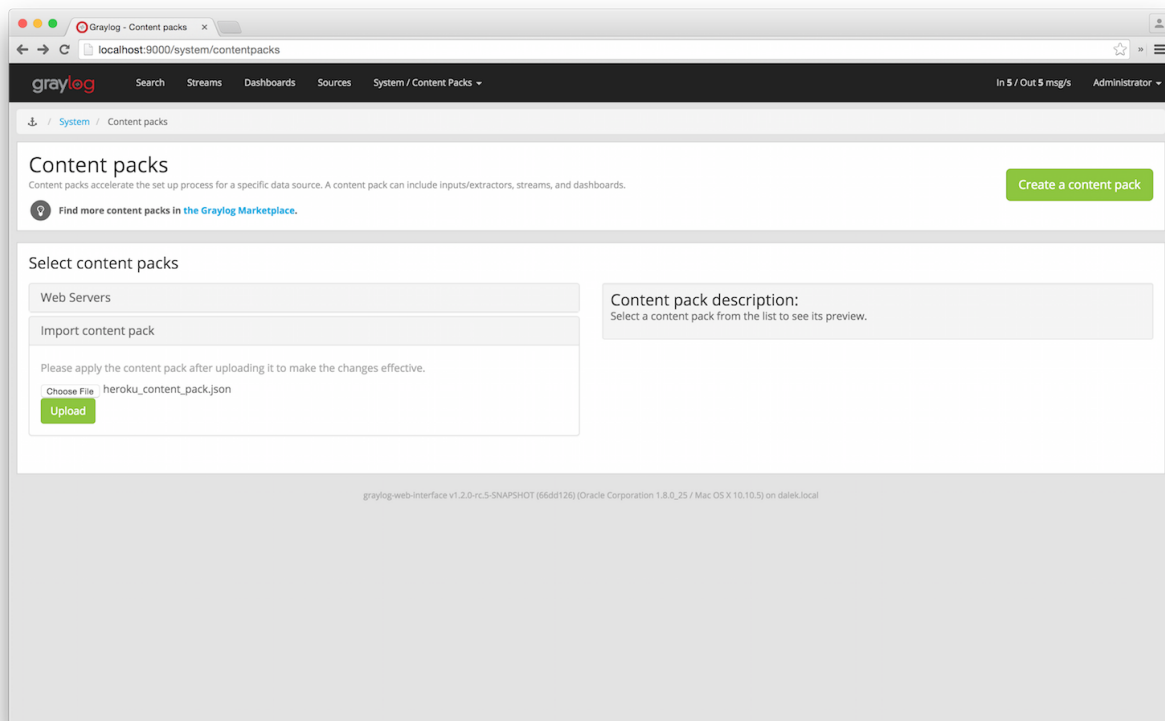
Heroku

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.

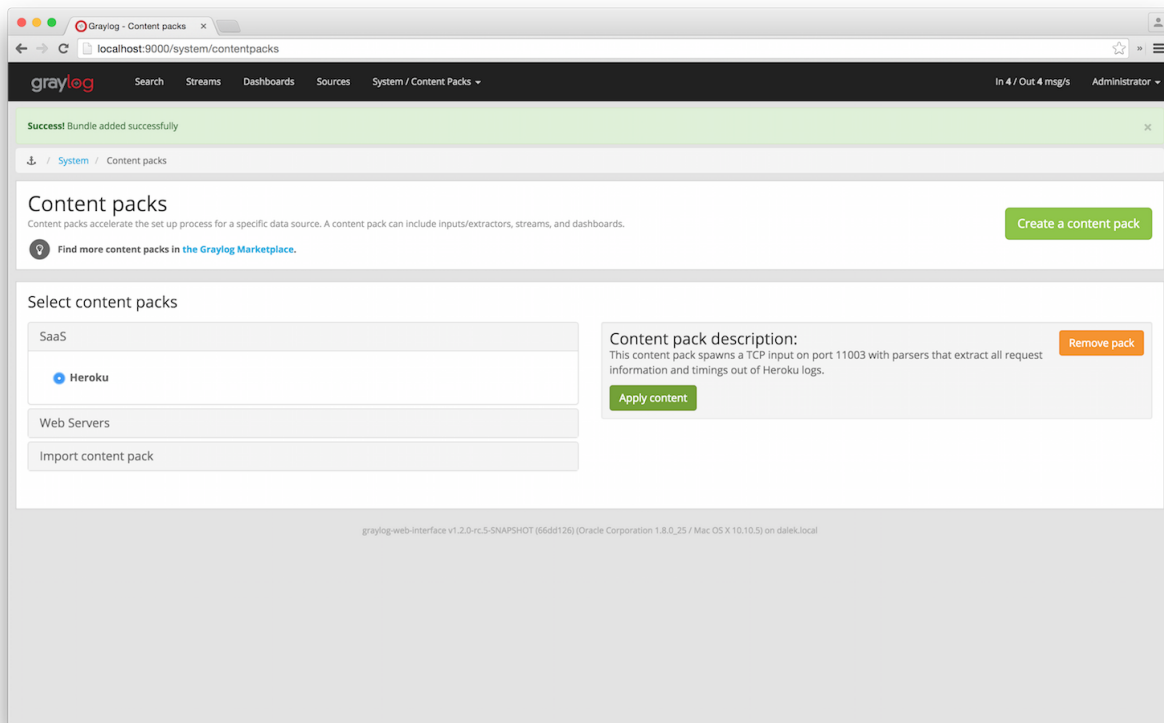
Configuring Graylog to receive Heroku log messages

The [Graylog Marketplace](#) contains a [content pack for Heroku logs](#), including extractors to parse the Heroku log format. You can download and use that [content pack](#) to configure Graylog to be able to receive Heroku logs.

Go to *System -> Content packs*, and click on *Import content pack*. Select the content pack downloaded from the Graylog Marketplace, and click *Upload*



On the same page, select *Heroku* on the *SaaS* category on the left column, and click on *Apply*.



That's it! You can verify that there is a new input for Heroku, containing a set of extractors to parse your log messages. Make sure your firewall setup allows incoming connections on the inputs port!

Local inputs 4 configured on this node

Heroku (Raw/Plaintext TCP) **running**
On node 41283fec / 172.16.0.7

```
recv_buffer_size: 1048576
port: 11003
tls_key_file:
tls_key_password: *****
tls_client_auth_cert_file:
max_message_size: 2097152
tls_client_auth: disabled
override_source:
bind_address: 0.0.0.0
tls_cert_file:
```

Show received messages

Manage extractors

Stop input

More actions ▾

Throughput / Metrics

1 minute average rate: 0 msg/s
Network I/O: 0B 0B (total: 0B 0B)
Active connections: 0 (0 total)

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" })
```

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" })
```

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:



You can use Graylog to analyze 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

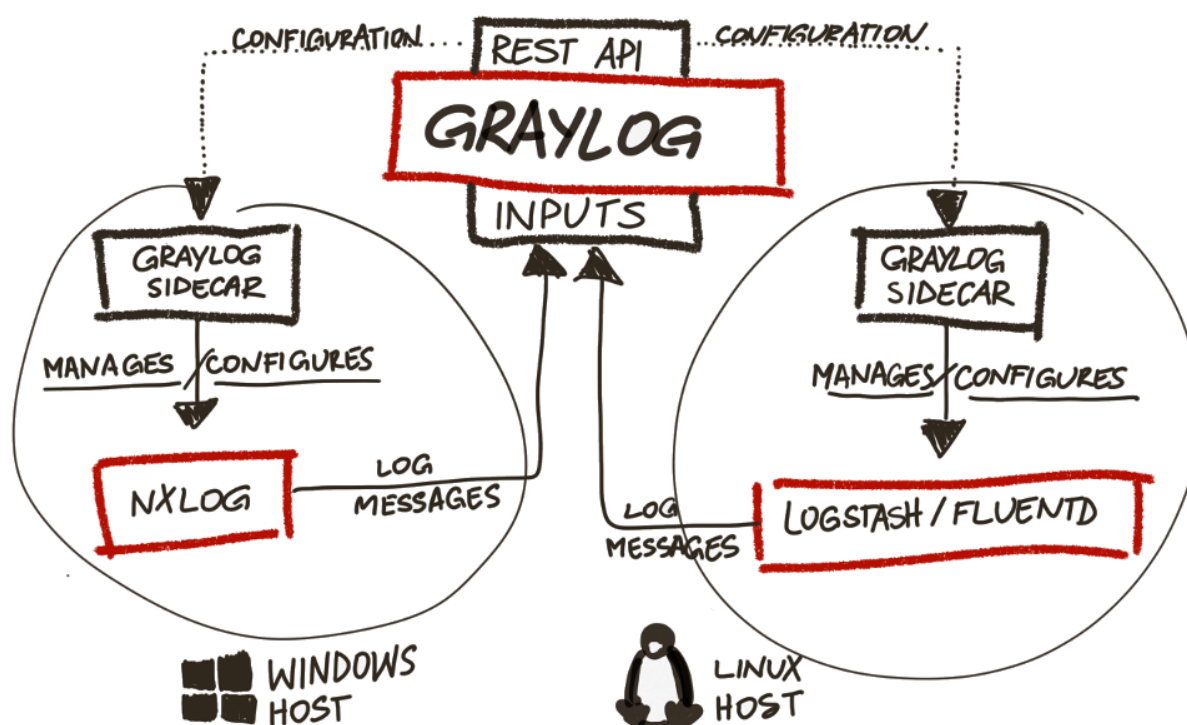
Log files come in a lot of different flavors and formats, much more than any single program could handle.

To support this use case, we provide the *Collector Sidecar* which acts as a supervisor process for other programs, such as `nxlog` and `Filebeats`, which have specifically been built to collect log messages from local files and ship them to remote systems like Graylog.

Of course you can still use any program supporting the GELF or syslog protocol (among others) to send your logs to Graylog.

Graylog Collector Sidecar

Graylog Collector Sidecar is a lightweight supervisor application for various log collectors. It allows the user to centralize the configuration of remote log collectors. Configurations can be maintained through the Graylog web interface in a graphical way. For advanced configurations it's also possible to store the raw content in Graylog. The Sidecar is then fetching the configuration meant for the target host, renders a configuration file and starts the selected log collector on it. It detects changes automatically, performs an update and restarts the necessary process.



Installation

Install the actual log collector first, e.g. NXlog, and disable the system service. The Sidecar will run as a daemon process and take control over NXlog. Download NXlog and follow the [instructions](#), afterwards:

```
$ sudo /etc/init.d/nxlog stop
$ sudo update-rc.d -f nxlog remove
$ sudo gpasswd -a nxlog adm
```

Same on a Windows host:

```
& 'C:\Program Files (x86)\nxlog\nxlog.exe' -u
```

From now on NXlog will not start automatically and the Sidecar can start the collector process without any issues. The Sidecar binary itself doesn't have any dependencies beside an installation of the used collector backend.

Linux/Unix

We offer Debian/Ubuntu packages directly in the [releases](#) section. Download and install the latest version via `dpkg(1)`:

```
$ wget https://github.com/Graylog2/collector-sidecar/releases/download/0.0.7/collector-sidecar_0.0.7-1_amd64.deb
$ sudo dpkg -i collector-sidecar_0.0.7-1_amd64.deb
```

To register and enable a system service use the `-service` option:

```
$ sudo graylog-collector-sidecar -service install
$ sudo service collector-sidecar start
```

Windows

Windows installation works in the same way. [Download](#) the Windows package and register the Sidecar services:

```
& 'C:\Program Files (x86)\graylog\collector-sidecar\graylog-collector-sidecar.exe' -service install
& 'C:\Program Files (x86)\graylog\collector-sidecar\graylog-collector-sidecar.exe' -service start
```

Configuration

On the command line you can provide a path to the configuration file with the `-c` switch. If no path is specified it looks on Linux systems for:

```
/etc/graylog/collector-sidecar/collector_sidecar.yml
```

and on Windows machines:

```
C:\Program Files (x86)\graylog\collector-sidecar\collector_sidecar.yml
```

The configuration file is separated into global options and backend specific options. Global options are:

Parameter	Description
server_url	URL to the Graylog API, e.g. <code>http://127.0.0.1:12900</code>
tls_skip_verify	Ignore errors when the REST API was started with a self-signed certificate
node_id	Name of the Sidecar instance, will also show up in the web interface
collector_id	Unique ID (UUID) of the instance. This can be a string or a path to an ID file
tags	List of configuration tags. All configurations on the server side that match the tag list will be fetched and merged by this instance
log_path	A path to a directory where the Sidecar can store the output of each running collector backend
update_interval	The interval in seconds the sidecar will fetch new configurations from the Graylog server
backends	A list of collector backends the user wants to run on the target host

Currently NXLog is supported as a backend collector, to make it work the Sidecar needs to know where the binary is installed and where it can write a configuration file for it.

Parameter	Description
name	The type name of the collector
enabled	Whether this backend should be started by the Sidecar or not
binary_path	Path to the actual collector binary
configuration_path	Path to the configuration file for this collector

As an example, a complete configuration could look like this:

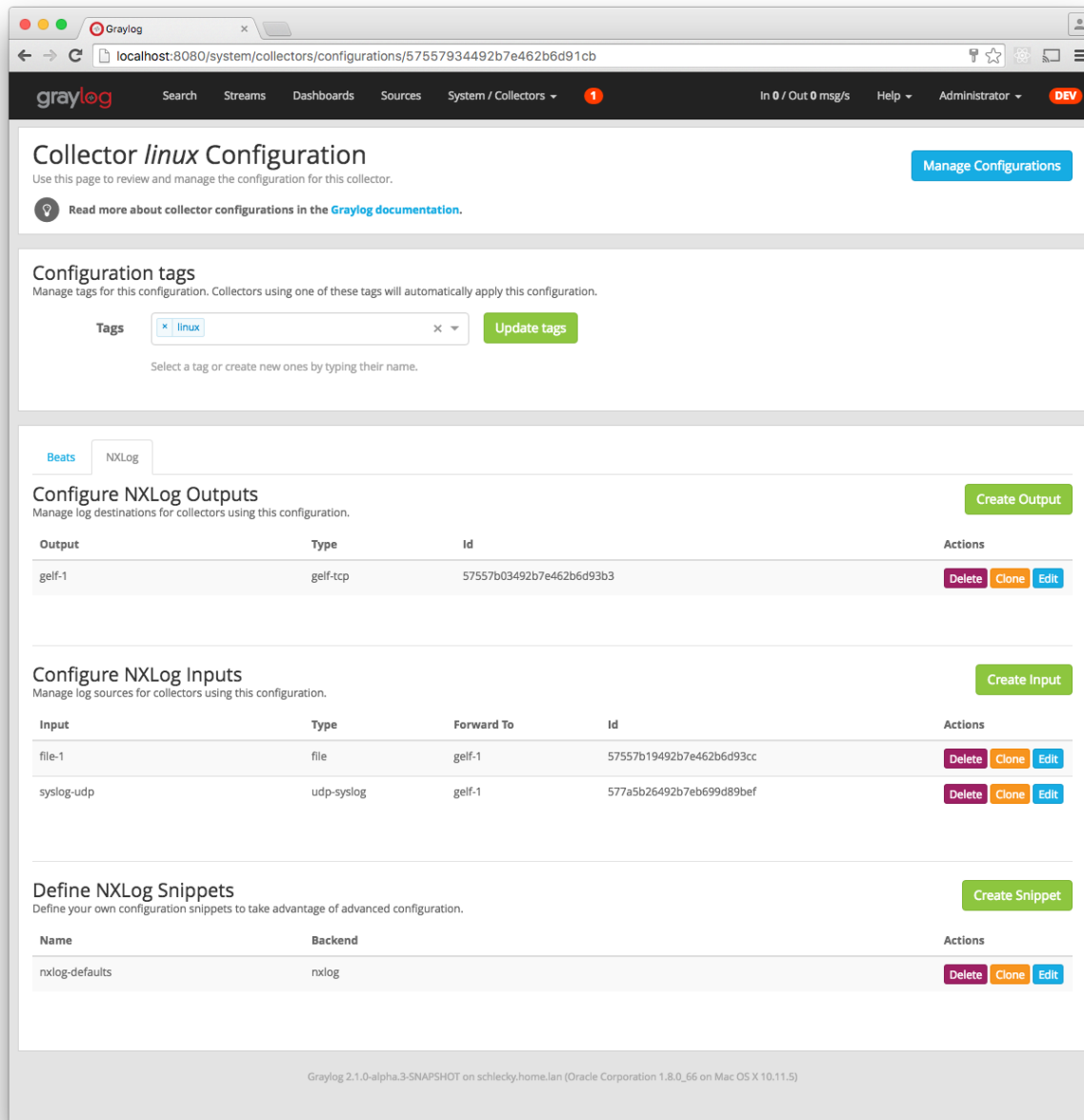
```
server_url: http://10.0.2.2:12900
node_id: graylog-collector-sidecar
collector_id: file:/etc/graylog/collector-sidecar/collector-id
tags:
  - linux
  - apache
  - redis
update_interval: 10
log_path: /var/log/graylog/collector-sidecar
backends:
  - name: nxlog
    enabled: true
    binary_path: /usr/bin/nxlog
    configuration_path: /etc/graylog/collector-sidecar/generated/nxlog.conf
```

Use the Graylog web interface to configure remote collectors

Navigate to `System → Collectors → Manage configurations`, this is the entry point for all Sidecar configurations. Multiple configurations can be created. Because not all connected Sidecars should fetch all configurations, it's essential to provide tags for each configuration. Every Sidecar is only fetching the configuration with the tag it was started with. See also the `tags` parameter in the section before. Each configuration can hold parts for multiple collector backends.

So you can create one configuration with the tag `linux` and this include e.g. an input section for a NXlog collector and one for a Filebeat collector. The Sidecar will then pick the right parts based on the backends that are enabled for the host system.

For some collectors (currently NXlog) we provide a default snippet. This snippet is created by default for every new configuration and contains settings like module paths or other system-wide configurations. This snippet is not meant as an example it's actually needed to generate a working configuration. However it's a normal snippet that can be edited or deleted.



Outputs, Inputs and Snippets

In the example above, Sidecar is instructing NXlog to create a GELF output that writes log messages back to Graylog. The two inputs are for reading in `/var/log/syslog` as a file input and listening on the UDP port 514 for incoming syslog messages. Both inputs route their messages to the GELF output.

There are three sections in a configuration: *Outputs*, *Inputs* and *Snippets*.

Inputs - Data collected by NXLog. Think of this as a source of log data. For example, it could be a file or a syslog.

Outputs - Once data is collected by NXLog, the data is transmitted to this IP or address and port. You need to configure a GELF “Input” (System->Inputs) to capture data on the port.

Snippets - Snippets can be used to represent more complicated collector configurations. Simply paste the whole content of your NXlog configuration into a snippet or use it as an extension to the inputs and outputs defined before. All snippets will be copied directly to the generated collector configuration, no matter if there inputs or outputs defined.

Debug

The Sidecar is writing to the local syslog so take a look into */var/log/syslog* on most systems. The output of the running collectors is written to the `log_path` directory.

You can also start the Sidecar in foreground and monitor the output of the process:

```
$ graylog-collector-sidecar -c /etc/graylog/collector-sidecar/collector_sidecar.yml
```

Graylog Collector (deprecated)

Warning: The Graylog Collector is deprecated and can be replaced with the *Graylog Collector Sidecar*.

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

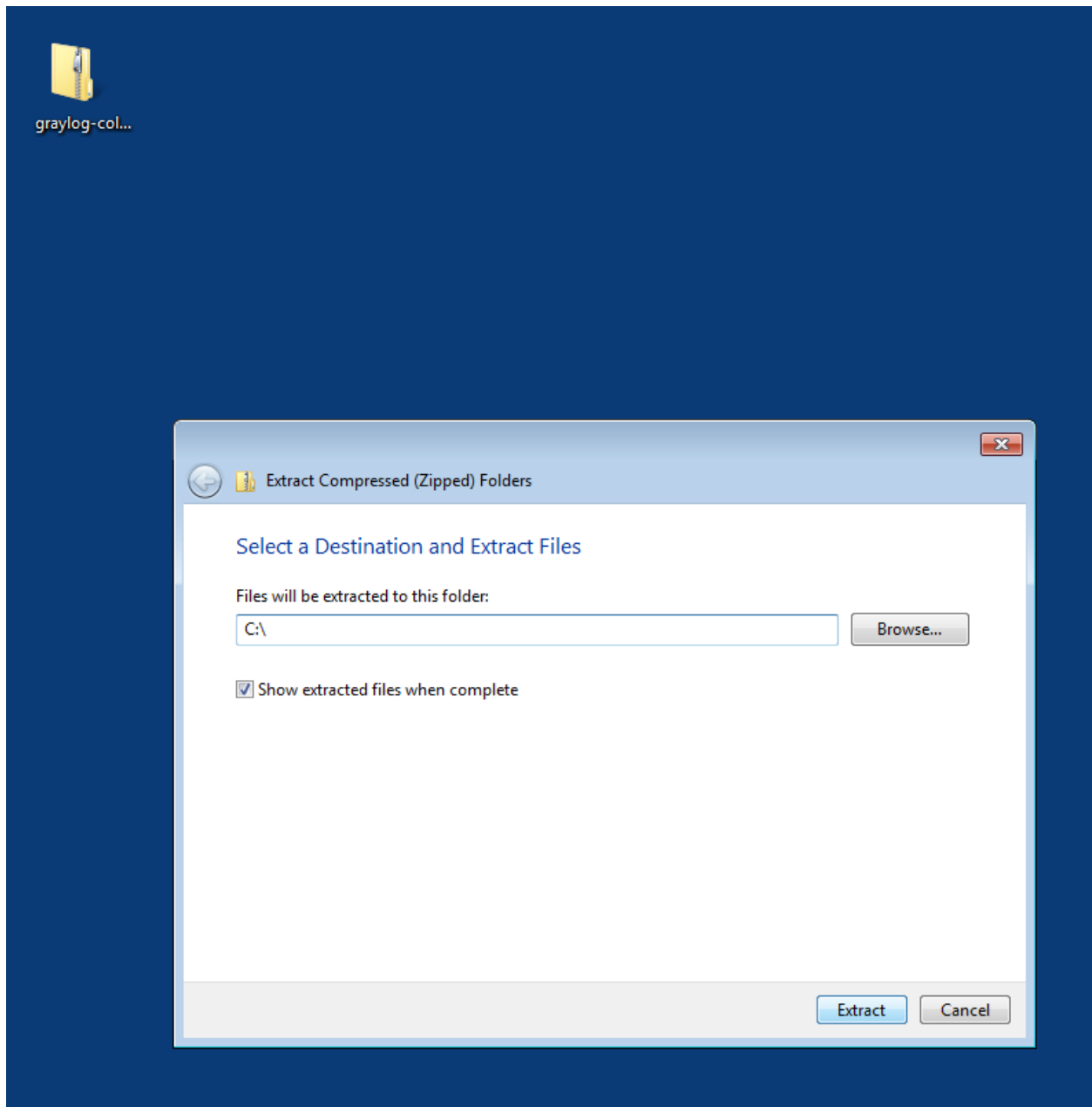
1. Download the latest collector release. (find download links in the [collector repository README](#))
2. Unzip collector tgz file to target location
3. `cp config/collector.conf.example to config/collector.conf`
4. Update `server-url` in `collector.conf` to correct Graylog server address (required 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

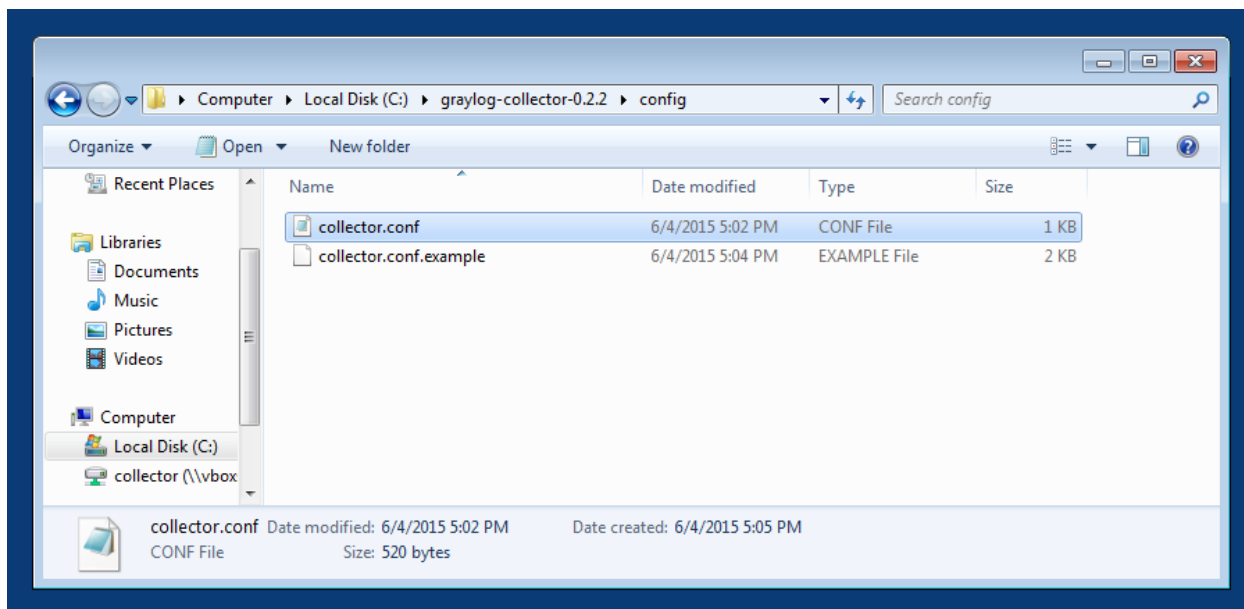
Windows

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.



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



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/"

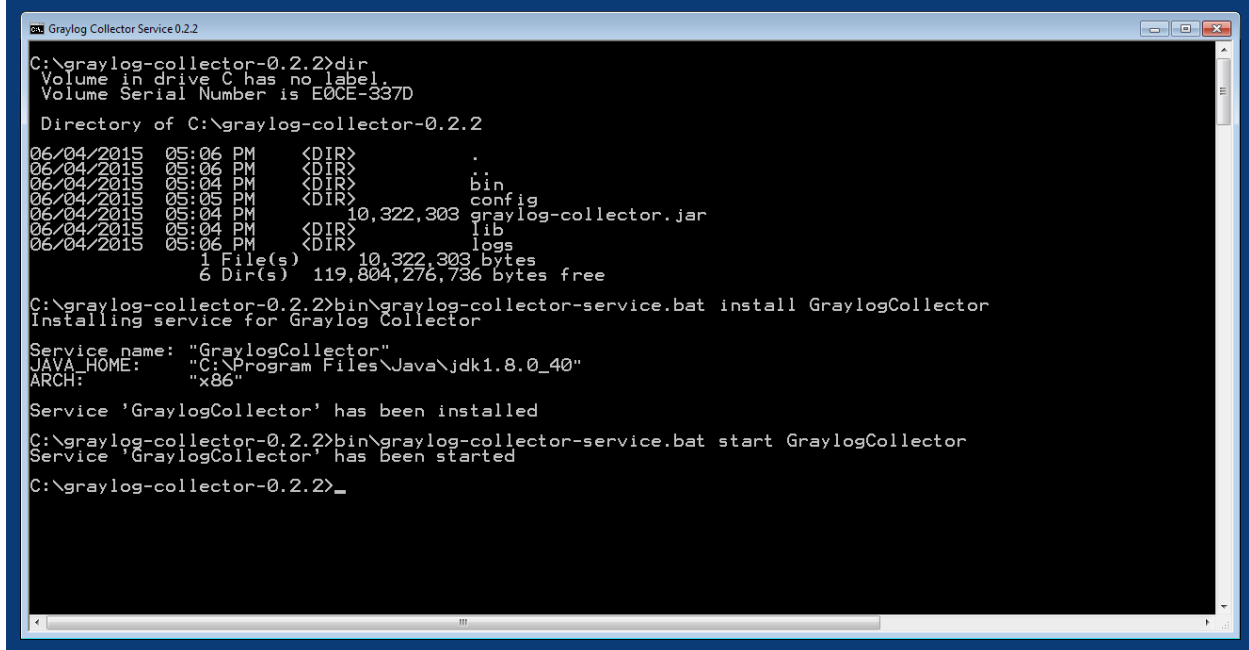
inputs {
  win-eventlog-application {
    type = "windows-eventlog"
    source-name = "Application"
    poll-interval = "1s"
  }
  win-eventlog-system {
    type = "windows-eventlog"
    source-name = "System"
    poll-interval = "1s"
  }
  win-eventlog-security {
    type = "windows-eventlog"
    source-name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



```
Graylog Collector Service 0.2.2
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 an `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 pattern. 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
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 tumble anacron[32426]: Job `cron.daily' terminated # <-- Log message 1
Jul 15 10:27:08 tumble 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=nu
java.lang.NullPointerException
    at org.graylog2.shared.buffer.JournalingMessageHandler$Converter.apply(JournalingMessageHa
    at org.graylog2.shared.buffer.JournalingMessageHandler$Converter.apply(JournalingMessageHa
```

```
at com.google.common.collect.Lists$TransformingRandomAccessList$1.transform(Lists.java:617)
at com.google.common.collect.TransformedIterator.next(TransformedIterator.java:48)
at java.util.AbstractCollection.toArray(AbstractCollection.java:141)
at java.util.ArrayList.<init>(ArrayList.java:177)
at com.google.common.collect.Lists.newArrayList(Lists.java:144)
at org.graylog2.shared.buffering.JournallingMessageHandler.onEvent(JournallingMessageHandler.java:144)
at org.graylog2.shared.buffering.JournallingMessageHandler.onEvent(JournallingMessageHandler.java:144)
at com.lmax.disruptor.BatchEventProcessor.run(BatchEventProcessor.java:128)
at com.codahale.metrics.InstrumentedExecutorService$InstrumentedRunnable.run(InstrumentedExecutorService.java:1142)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:617)
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=nu
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 you 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	<code>sudo start graylog-collector</code>
Debian	systemd	<code>sudo systemctl start graylog-collector</code>
CentOS	systemd	<code>sudo systemctl start graylog-collector</code>

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:

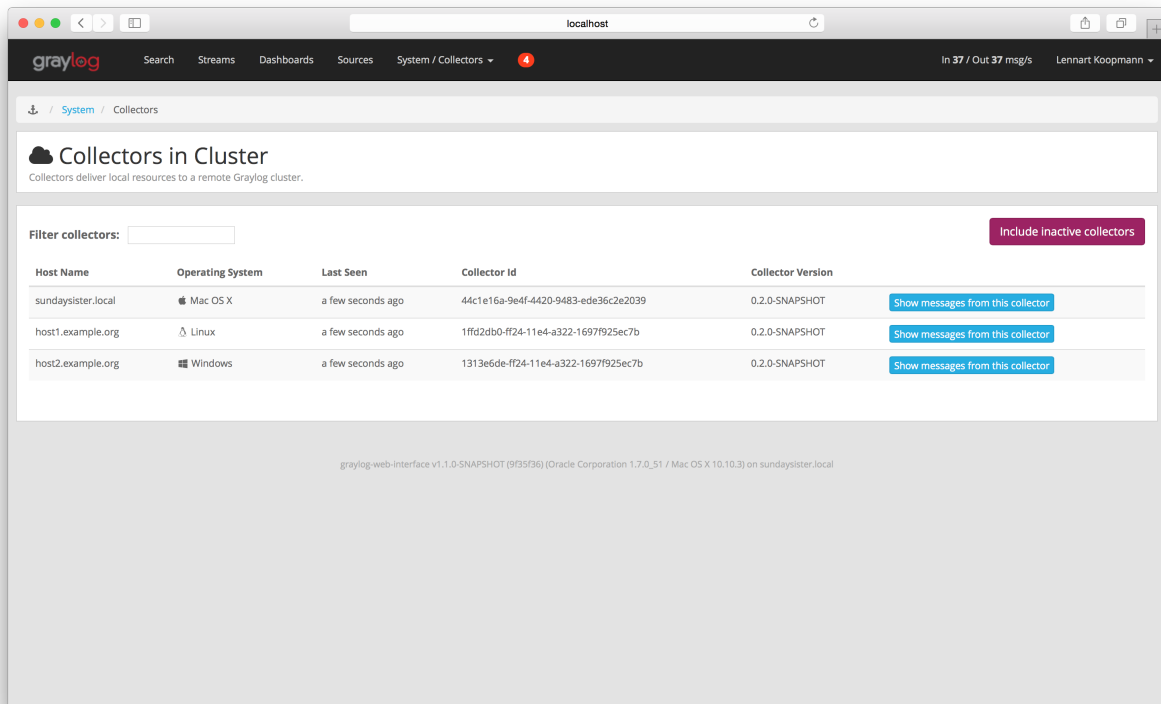
```
C:\> cd graylog-collector-0.2.2
C:\graylog-collector-0.2.2> bin\graylog-collector.bat run -f config\collector.conf
```



```
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-a7d9
2015-06-04T17:37:29.277+0200 INFO [GelfOutput] outputs.gelf.GelfOutput - Starting GELF transport: org.graylog
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.



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 v
2015-05-12T16:00:11.489+0200 INFO [main] o.g.collector.utils.CollectorId - Collector ID: cf4734f7-0
2015-05-12T16:00:11.505+0200 INFO [GelfOutput] o.g.c.outputs.gelf.GelfOutput - Starting GELF transp
2015-05-12T16:00:11.512+0200 INFO [main] o.graylog.collector.cli.commands.Run - Service RUNNING: Bu
2015-05-12T16:00:11.513+0200 INFO [main] o.graylog.collector.cli.commands.Run - Service RUNNING: Met
2015-05-12T16:00:11.515+0200 INFO [main] o.graylog.collector.cli.commands.Run - Service RUNNING: Fi
2015-05-12T16:00:11.516+0200 INFO [main] o.graylog.collector.cli.commands.Run - Service RUNNING: Ge
2015-05-12T16:00:11.516+0200 INFO [main] o.graylog.collector.cli.commands.Run - Service RUNNING: Hea
2015-05-12T16:00:11.516+0200 INFO [main] o.graylog.collector.cli.commands.Run - Service RUNNING: Sto
```

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 -
```

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.

Searching

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 configuration file:

```
allow_leading_wildcard_searches = true
```

Also note that message, full_message, and source are the only fields that are being analyzed by default. While wildcard searches (using * and ?) work on all indexed fields, analyzed fields will behave a little bit different. See [wildcard and regexp queries](#) for details.

Fuzziness: You can search for similar terms:

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

This example is using the [Damerau–Levenshtein distance](#) with a default distance of 2 and will match “ssh login” and “example.org” (intentionally misspelled in the query).

You can change the distance like this:

```
source:exmaple.org~1
```

You can also use the fuzzyness operator to do a **proximity** search where the terms in a phrase can have different/fuzzy distances from each other and don’t have to be in the defined order:

```
"foo bar"~5
```

Numeric fields support **range queries**. Ranges in square brackets are inclusive, curly brackets are exclusive and can even be combined:

```
http_response_code:[500 TO 504]
http_response_code:{400 TO 404}
bytes:{0 TO 64}
http_response_code:[0 TO 64}
```

You can also do searches with one side unbounded:

```
http_response_code:>400
http_response_code:<400
http_response_code:>=400
http_response_code:<=400
```

It is also possible to combine unbounded range operators:

```
http_response_code:(>=400 AND <500)
```

Escaping

The following characters must be escaped with a backslash:

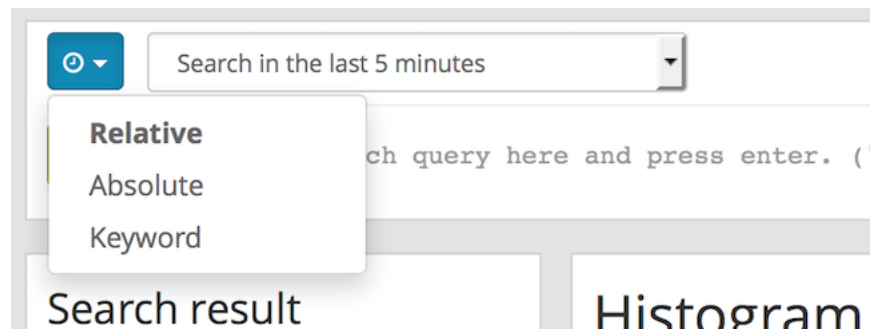
```
&& || : \ / + - ! ( ) { } [ ] ^ " ~ * ?
```

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](#)



Relative time frame selector

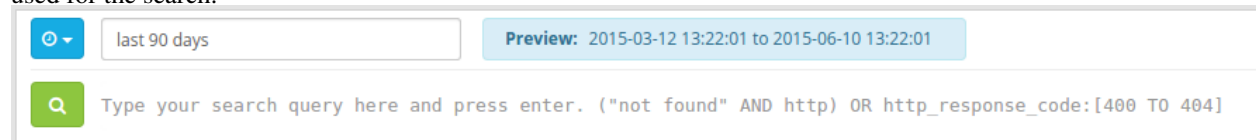
The relative time frame selector lets you look for messages from the selected option to the time you hit the search button. The selector offers a wide set of relative time frames that fit most of your search needs.

Absolute time frame selector

When you know exactly the boundaries of your search, you want to use the absolute time frame selector. Simply introduce the dates and times for the search manually or click in the input field to open up a calendar where you can choose the day with your mouse.

Keyword time frame selector

Graylog offers a keyword time frame selector that allows you to specify the time frame for the search in natural language like *last hour* or *last 90 days*. The web interface shows a preview of the two actual timestamps that will be used for the search.



Here are a few examples for possible values.

- “last month” searches between one month ago and now
- “4 hours ago” searches between four hours ago and now

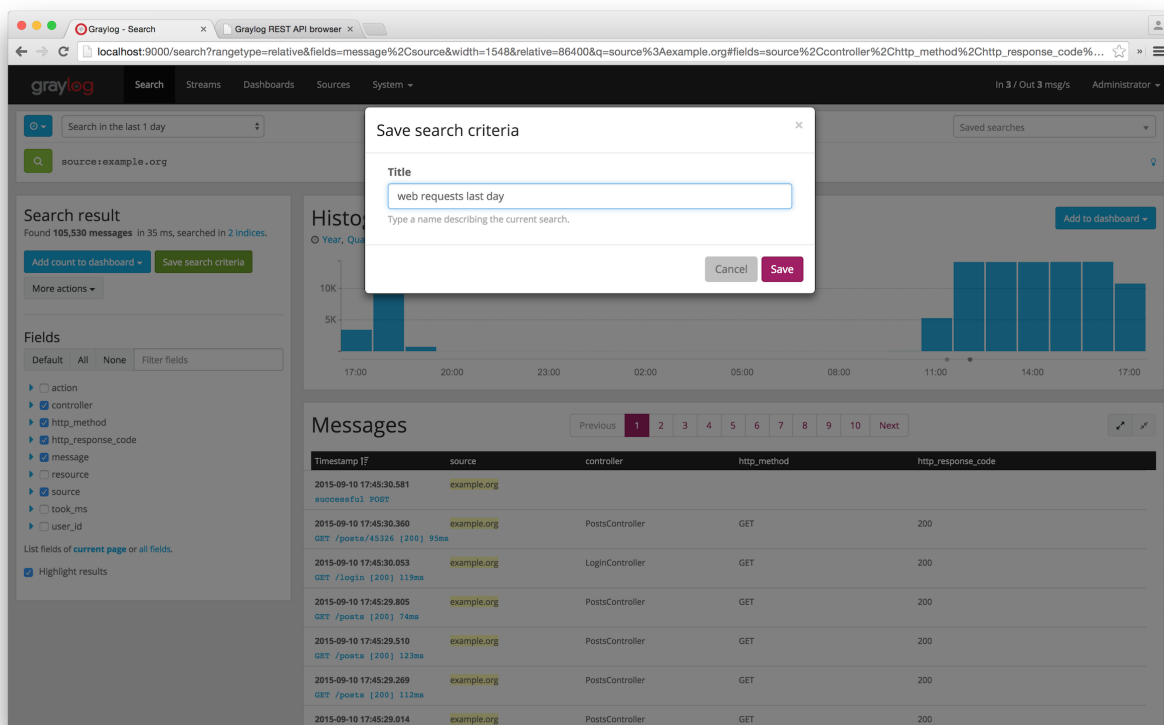
- “1st of april to 2 days ago” searches between 1st of April and 2 days ago
- “yesterday midnight +0200 to today midnight +0200” searches between yesterday midnight and today midnight in timezone +0200 - will be 22:00 in UTC

The time frame is parsed using the [natty natural language parser](#). Please consult its documentation for details.

Saved searches

Sometimes you may want to search a specific search configuration to be used later. Graylog provides a saved search functionality to accomplish exactly that.

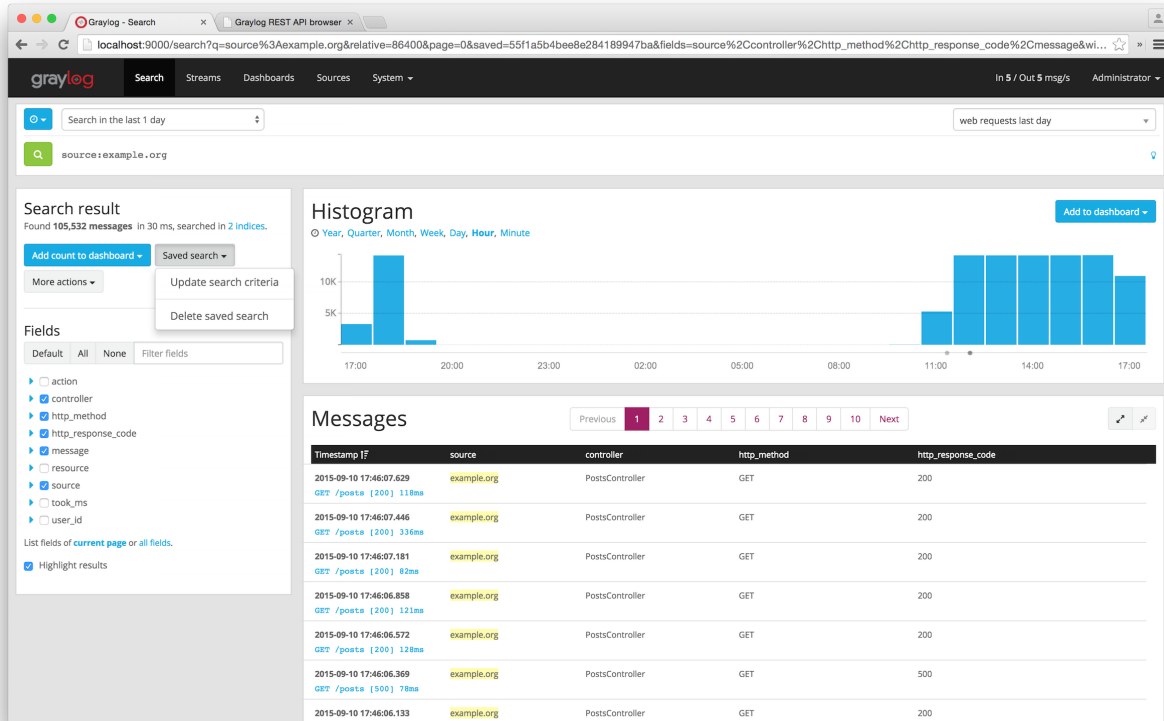
Once you submitted your search, selected the fields you want to show from the search sidebar, and chosen a resolution for the histogram, click on the *Save search criteria* button on the sidebar.



Give a name to the current search and click on save. When you want to use the saved search later on, you only need to select it from the saved search selector.

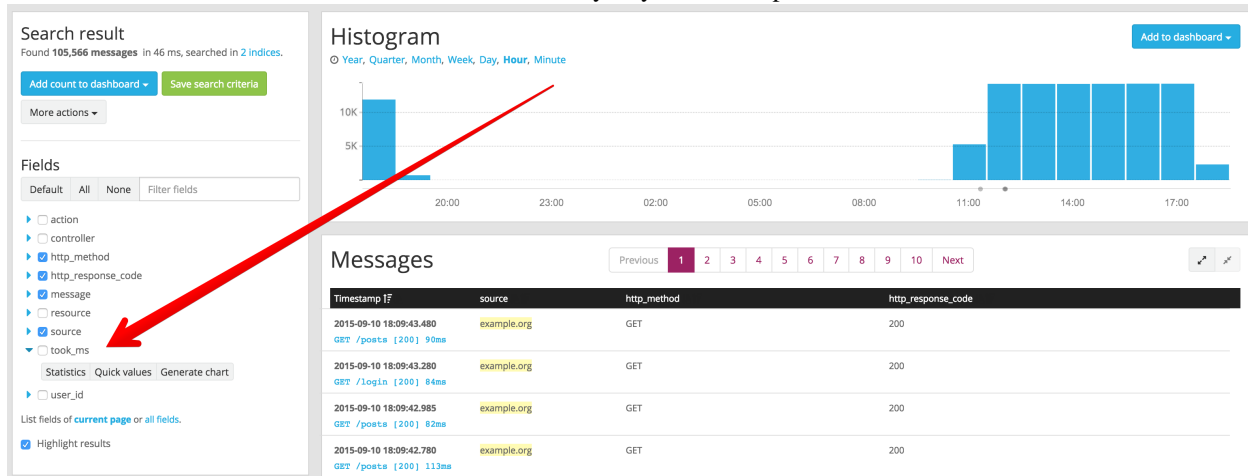


Of course, you can always update the selected fields or name of your saved search. To do so, select the saved search from the saved search selector, update the field selection or histogram resolution, and click on *Saved search -> Update search criteria*. It is also possible to delete the saved search by selecting *Saved search -> Delete saved search*.



Analysis

Graylog provides several tools to analyze your search results. It is possible to save these analysis into dashboards, so you can check them over time in a more convenient way. To analyze a field from your search results, expand the field in the search sidebar and click on the button of the analysis you want to perform.



Field statistics

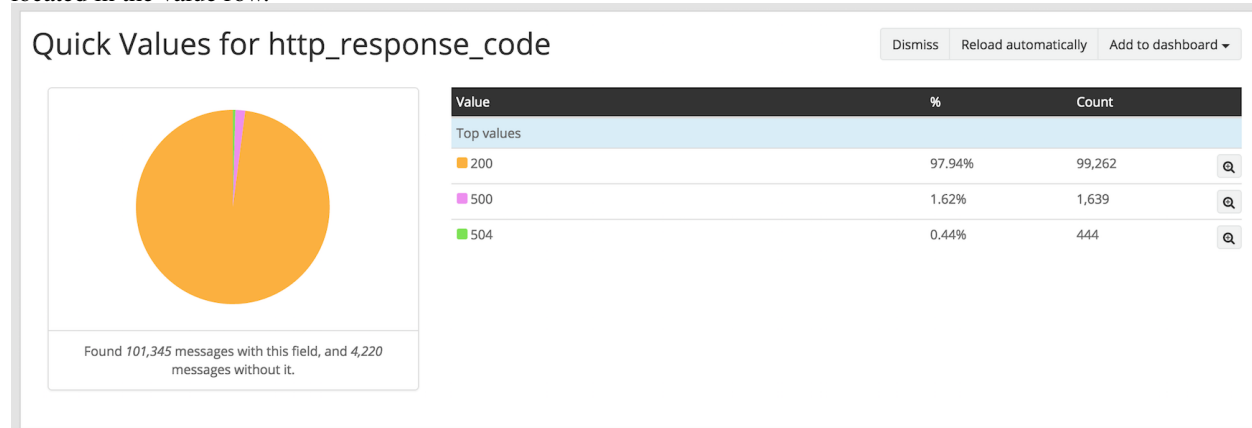
Compute different statistics on your fields, to help you better summarize and understand the data in them.

The statistical information consist of: total, mean, minimum, maximum, standard deviation, variance, sum, and cardinality. On non-numeric fields, you can only see the total amount of messages containing that field, and the cardinality of the field, i.e. the number of unique values it has.

Field Statistics									Dismiss	Reload automatically	Add to dashboard ▾
Field ▲	Total	Mean	Minimum	Maximum	Std. deviation	Variance	Sum	Cardinality			
controller	101,327	NaN	NaN	NaN	NaN	NaN	NaN	3			
resource	101,324	NaN	NaN	NaN	NaN	NaN	NaN	5			
took_ms	101,326	122.23	71	5,450	326.98	106,915.15	12,385,059	134			

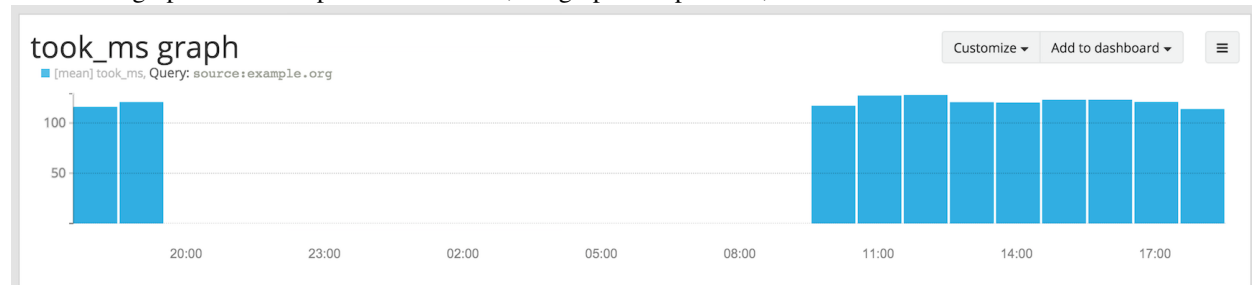
Quick values

Quick values helps you to find out the distribution of values for a field. Alongside a graphic representation of the common values contained in a field, Graylog will display a table with all different values, allowing you to see the number of times they appear. You can include any value in your search query by clicking on the magnifying glass icon located in the value row.

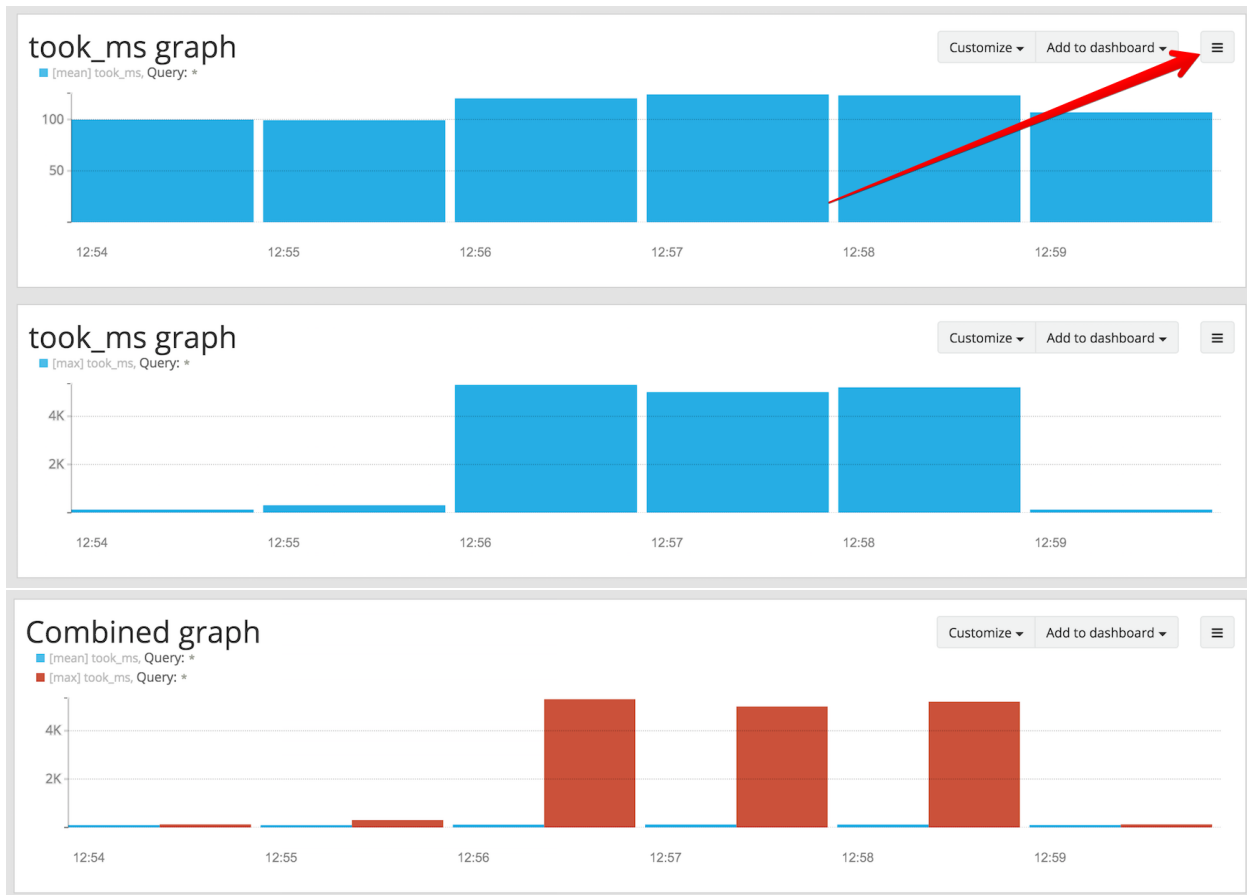


Field graphs

You can create field graphs for any numeric field, by clicking on the *Generate chart* button in the search sidebar. Using the options in the *Customize* menu on top of the field graph, you can change the statistical function used in the graph, the kind of graph to use to represent the values, the graph interpolation, as well as the time resolution.



Once you have customized some field graphs, you can also combine them by dragging them from the hamburger icon on the top corner of the graph, and dropping them into another field graph. You can see the location of the hamburger icon and the end result in the the following screenshots:



Field graphs appear every time you perform a search, allowing you to compare data, or combine graphs coming from different streams.

Export results as CSV

It is also possible to export the results of your search as a CSV document. To do so, select all fields you want to export in the search sidebar, click on the *More actions* button, and select *Export as CSV*.

The screenshot shows the Graylog search interface. On the left, the 'Search result' sidebar displays the search criteria and a list of fields. The 'More actions' button is highlighted, and the 'Export as CSV' option is selected. A red arrow points to this option. On the right, the 'Histogram' and 'Messages' sections are visible. The 'Messages' section shows a table of search results.

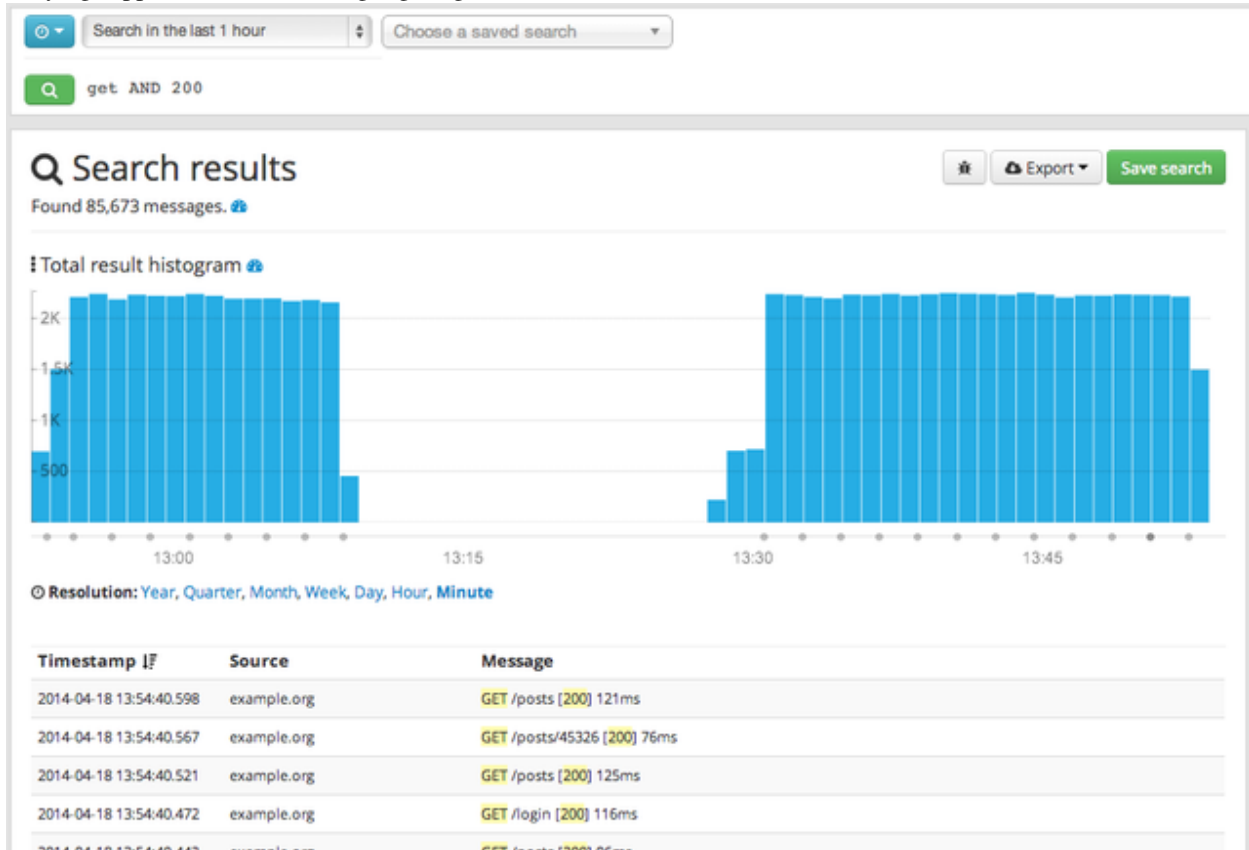
Timestamp	IP	source	http_method	http_response_code
2015-09-11 13:57:25.563	GET	/posts [200] 92ms	example.org	200
2015-09-11 13:57:25.364	GET	/posts [200] 119ms	example.org	200
2015-09-11 13:57:25.149	GET	/posts [200] 95ms	example.org	200
2015-09-11 13:57:24.870	GET	/posts/45326 [200] 101ms	example.org	200

Hint: Some Graylog inputs keep the original message in the `full_message` field. If you need to export the

original message, you can do so by clicking on the *List all fields* link at the bottom of the sidebar, and then selecting the `full_message` field.

Search result highlighting

Graylog supports search result highlighting since v0.20.2:



Enabling/Disabling search result highlighting

Using search result highlighting will result in slightly higher resource consumption of searches. You can enable and disable it using a configuration parameter in the `graylog.conf` of your Graylog nodes:

```
allow_highlighting = true
```

Search configuration

Graylog allows customizing the options allowed to search queries, like limiting the time range users can select or configuring the list of displayed relative time ranges.

graylog
Search
Streams
Dashboards
Sources
System / Configurations ▼
In 19 / Out 19 msg/s

Configurations

You can configure system settings for different sub systems on this page.

Search Configuration

Query time range limit disabled
The maximum time users can query data in the past. This prevents users from accidentally creating queries which span a lot of data and would need a long time and many resources to complete (if at all).

Relative time range options		Surrounding time range options	
PT5M	Search in the last 5 minutes	PT1S	1 second
PT15M	Search in the last 15 minutes	PT5S	5 seconds
PT30M	Search in the last 30 minutes	PT10S	10 seconds
PT1H	Search in the last 1 hour	PT30S	30 seconds
PT2H	Search in the last 2 hours	PT1M	1 minute
PT8H	Search in the last 8 hours	PT5M	5 minutes
P1D	Search in the last 1 day	Surrounding search filter fields	
P2D	Search in the last 2 days	file	
P5D	Search in the last 5 days	source	
P7D	Search in the last 7 days	gl2_source_input	
P14D	Search in the last 14 days	source_file	
P30D	Search in the last 30 days		
PT0S	Search in all messages		

Update

All search configuration settings can be customized using the web interface on the *System -> Configurations* page in the *Search configuration* section.

Query time range limit

Sometimes the amount of data stored in Graylog is quite big and spans a wide time range (e. g. multiple years). In order to prevent normal users from accidentally running search queries which could use up lots of resources, it is possible to limit the time range that users are allowed to search in.

Using this feature, the time range of a search query exceeding the configured query time range limit will automatically be adapted to the given limit.

Update Search Configuration

☒ Enable query limit

Query time range limit (ISO8601 Duration)

P30D

a month

The maximum time range for searches. (i.e. "P30D" for 30 days, "PT24H" for 24 hours)

Relative Timerange Options

Configure the available options for the **relative** time range selector as **ISO8601 duration**

PT5M

5

Search in the last 5 minutes

The query time range limit is a *duration* formatted according to ISO 8601 following the basic format `P<date>T<time>` with the following rules:

Designator	Description
P	Duration designator (for period) placed at the start of the duration representation
Y	Year designator that follows the value for the number of years
M	Month designator that follows the value for the number of months
W	Week designator that follows the value for the number of weeks
D	Day designator that follows the value for the number of days
T	Time designator that precedes the time components of the representation
H	Hour designator that follows the value for the number of hours
M	Minute designator that follows the value for the number of minutes
S	Second designator that follows the value for the number of seconds

Examples:

ISO 8601 duration	Description
P30D	30 days
PT1H	1 hour
P1DT12H	1 day and 12 hours

More details about the format of ISO 8601 durations can be found [on Wikipedia](#).

Relative time ranges

The list of time ranges displayed in the *Relative time frame selector* can be configured, too. It consists of a list of ISO 8601 durations which the users can select on the search page.











The format of the ISO 8601 durations can be looked up [here](#).

Update Search Configuration ×

☐ Enable query limit

Relative Timerange Options

Configure the available options for the **relative** time range selector as **ISO8601 duration**

PT5M	5	Search in the last 5 minutes	
PT15M	15	Search in the last 15 minutes	
PT30M	30	Search in the last 30 minutes	
PT1H	1h	Search in the last 1 hour	
PT2H	2h	Search in the last 2 hours	
PT8H	8h	Search in the last 8 hours	
P1D	1d	Search in the last 1 day	
P2D	2d	Search in the last 2 days	
P5D	5d	Search in the last 5 days	
P7D	7d	Search in the last 7 days	

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, selecting the option to match all rules:

- Field `level` must be greater than 4
- Field `source` must match regular expression `^database-host-\d+`

This will route every new message with a `level` higher than *WARN* and a `source` that matches the database host regular expression into the stream.

A message will be routed into every stream that has all (or any) of its rules matching. This means that a message can be part of many streams and not just one.

The stream is now appearing in the streams list and a click on its title will show you all database errors.

Streams can be used to be alerted in case certain condition happens. We cover more topics related to alerts in [Alerts](#).

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*. The stream is now saved but **not yet activated**.
4. Click on “Edit rules” for the stream you just created. That will open a page where you can manage and test stream rules.
5. Choose how you want to evaluate the stream rules to decide which messages go into the stream:
 - *A message must match all of the following rules* (logical AND): Messages will only be routed into the stream if all rules in the stream are fulfilled. This is the default behavior
 - *A message must match at least one of the following rules* (logical OR): Messages will be routed into the stream if one or more rules in the stream are fulfilled
6. Add stream rules, by indicating the field that you want to check, and the condition that should satisfy. Try the rules against some messages by loading them from an input or manually giving a message ID. Once you are satisfied with the results, click on “I’m done”.
7. The stream is still paused, click on the “Start stream” button to activate the stream.

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 notifications. 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.

Alert callbacks types explained

In this section we explain what the default alert callbacks included in Graylog do, and how to configure them. Alert callbacks are meant to be extensible through [Plugins](#), you can find more types in the [Graylog Marketplace](#) or even create your own.

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

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

The subject of sent out mail alerts

CancelSave

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.

HTTP alert callback

The HTTP alert callback lets you configure an endpoint that will be called when the alert is triggered.

Graylog will send a POST request to the callback URL including information about the alert. Here is an example of the payload included in a callback:

```
{
  "check_result": {
    "result_description": "Stream had 2 messages in the last 1 minutes with trigger condition mor
    "triggered_condition": {
      "id": "5e7a9c8d-9bb1-47b6-b8db-4a3a83a25e0c",
      "type": "MESSAGE_COUNT",
      "created_at": "2015-09-10T09:44:10.552Z",
      "creator_user_id": "admin",
      "grace": 1,
      "parameters": {
        "grace": 1,
        "threshold": 1,
        "threshold_type": "more",
        "backlog": 5,
        "time": 1
      },
      "description": "time: 1, threshold_type: more, threshold: 1, grace: 1",
      "type_string": "MESSAGE_COUNT",
      "backlog": 5
    }
  }
}
```

```
    },
    "triggered_at": "2015-09-10T09:45:54.749Z",
    "triggered": true,
    "matching_messages": [
      {
        "index": "graylog2_7",
        "message": "WARN: System is failing",
        "fields": {
          "gl2_remote_ip": "127.0.0.1",
          "gl2_remote_port": 56498,
          "gl2_source_node": "41283fec-36b4-4352-a859-7b3d79846b3c",
          "gl2_source_input": "55f15092bee8e2841898eb53"
        },
        "id": "b7b08150-57a0-11e5-b2a2-d6b4cd83d1d5",
        "stream_ids": [
          "55f1509dbee8e2841898eb64"
        ],
        "source": "127.0.0.1",
        "timestamp": "2015-09-10T09:45:49.284Z"
      },
      {
        "index": "graylog2_7",
        "message": "ERROR: This is an example error message",
        "fields": {
          "gl2_remote_ip": "127.0.0.1",
          "gl2_remote_port": 56481,
          "gl2_source_node": "41283fec-36b4-4352-a859-7b3d79846b3c",
          "gl2_source_input": "55f15092bee8e2841898eb53"
        },
        "id": "afd71342-57a0-11e5-b2a2-d6b4cd83d1d5",
        "stream_ids": [
          "55f1509dbee8e2841898eb64"
        ],
        "source": "127.0.0.1",
        "timestamp": "2015-09-10T09:45:36.116Z"
      }
    ]
  },
  "stream": {
    "creator_user_id": "admin",
    "outputs": [],
    "matching_type": "AND",
    "description": "test stream",
    "created_at": "2015-09-10T09:42:53.833Z",
    "disabled": false,
    "rules": [
      {
        "field": "gl2_source_input",
        "stream_id": "55f1509dbee8e2841898eb64",
        "id": "55f150b5bee8e2841898eb7f",
        "type": 1,
        "inverted": false,
        "value": "55f15092bee8e2841898eb53"
      }
    ],
    "alert_conditions": [
      {
        "creator_user_id": "admin",
```

```

    "created_at": "2015-09-10T09:44:10.552Z",
    "id": "5e7a9c8d-9bb1-47b6-b8db-4a3a83a25e0c",
    "type": "message_count",
    "parameters": {
      "grace": 1,
      "threshold": 1,
      "threshold_type": "more",
      "backlog": 5,
      "time": 1
    }
  },
  "id": "55f1509dbee8e2841898eb64",
  "title": "test",
  "content_pack": null
}

```

Alert callback history

Sometimes sending alert callbacks may fail for some reason. Graylog provides an alert callback history for those occasions, helping you to debug and fix any problems that may arise.

The screenshot shows the 'Triggered alerts' section in Graylog. It displays a table with columns: Triggered, Condition, and Reason. There are three alerts listed, each with a 'Show callbacks' button. The first alert, triggered 2 minutes ago, shows a failed 'HTTP Alarm Callback' with a red status icon and the message 'Expected successful HTTP response [2xx] but got [404]'. The second alert, triggered 4 minutes ago, shows a successful 'HTTP Alarm Callback' with a green status icon and the message 'Show configuration'. The third alert, triggered an hour ago, also shows a successful 'HTTP Alarm Callback' with a green status icon and the message 'Show configuration'. A pagination bar at the bottom shows 1 page of results.

To check the status of alert callbacks, go to the *Streams* page, and click on the *Manage alerts* button next to the stream containing the alert callbacks. You can find the alert callback history at the bottom of that page, in the *Triggered alerts* section.

On the list of alerts, clicking on *Show callbacks* will open a list of all the callbacks involved in the alert, including their status and configuration at the time the alert was triggered.

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

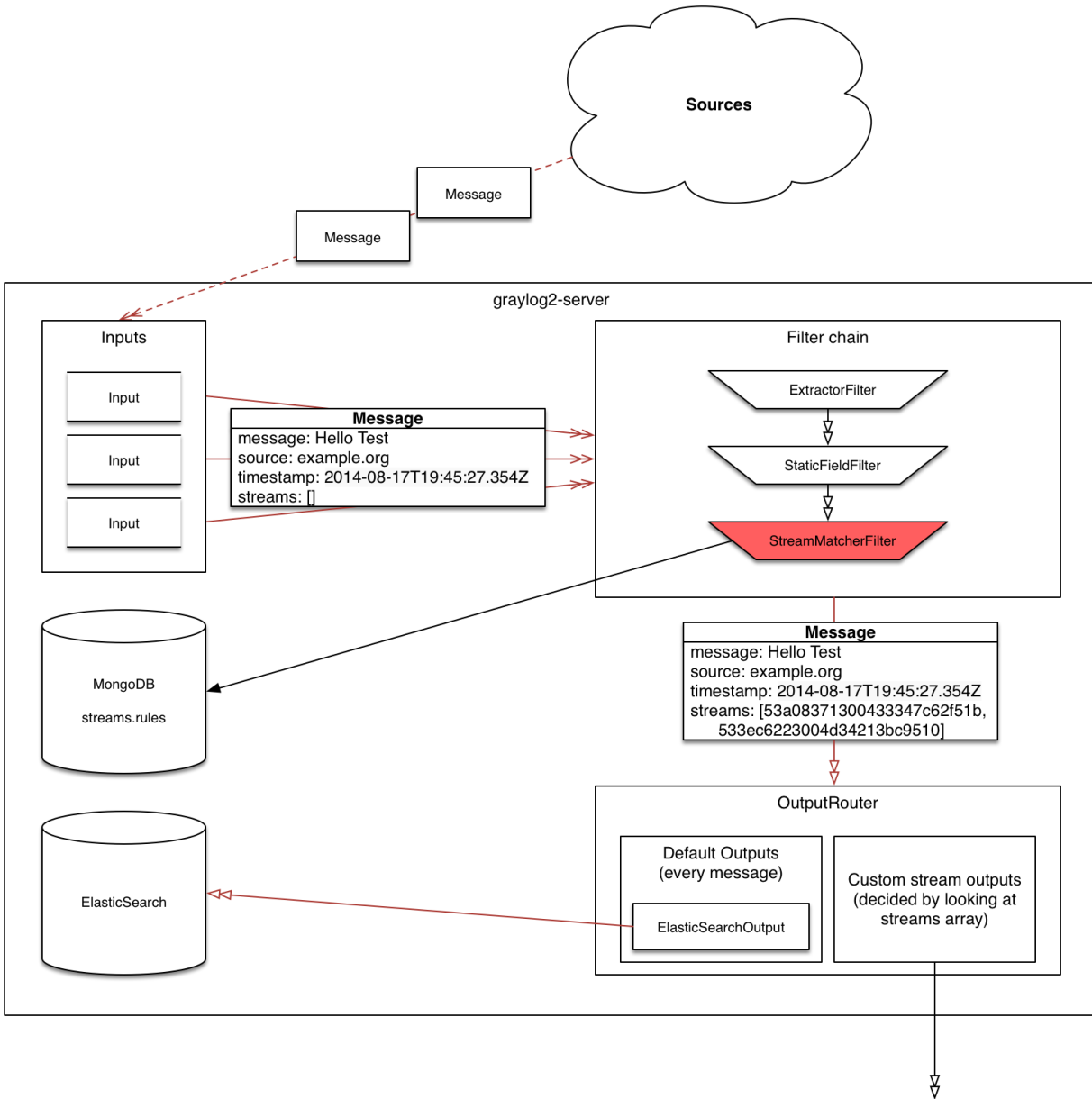
These are a few example use cases for streams:

- 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 were 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 the rules of a stream. For messages satisfying *all* or *at least one* of the stream rules (as configured in the stream), the internal ID of that stream is stored in the `streams` array of the processed message.

All analysis methods and searches that are bound to streams can now easily narrow their operation by searching with a `streams:[STREAM_ID]` limit. This is done automatically by Graylog and does not have to be provided by the user.



Stream Processing Runtime Limits

An important step during the processing of a message is the stream classification. Every message is matched against the user-configured stream rules. The message is added to the stream if all or any rules of a stream matches, depending on what the user chose. Applying stream rules is done during the indexing of a message only, so the amount of time spent for the classification of a message is crucial for the overall performance and message throughput the system can handle.

There are certain scenarios when a stream rule takes very long to match. When this happens for a number of messages, message processing can stall, messages waiting for processing accumulate in memory and the whole system could become non-responsive. Messages are lost and manual intervention would be necessary. This is the worst case scenario.

To prevent this, the runtime of stream rule matching is limited. When it is taking longer than the configured runtime limit, the process of matching this exact message against the rules of this specific stream is aborted. Message processing in general and for this specific message continues though. As the runtime limit needs to be configured pretty high (usually a magnitude higher as a regular stream rule match takes), any excess of it is considered a fault and is recorded for this stream. If the number of recorded faults for a single stream is higher than a configured threshold, the stream rule set of this stream is considered faulty and the stream is disabled. This is done to protect the overall stability and performance of message processing. Obviously, this is a tradeoff and based on the assumption, that the total loss of one or more messages is worse than a loss of stream classification for these.

There are scenarios where this might not be applicable or even detrimental. If there is a high fluctuation of the message load including situations where the message load is much higher than the system can handle, overall stream matching can take longer than the configured timeout. If this happens repeatedly, all streams get disabled. This is a clear indicator that your system is overutilized and not able to handle the peak message load.

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 con",
      "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/32.0.1700.102
```

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.

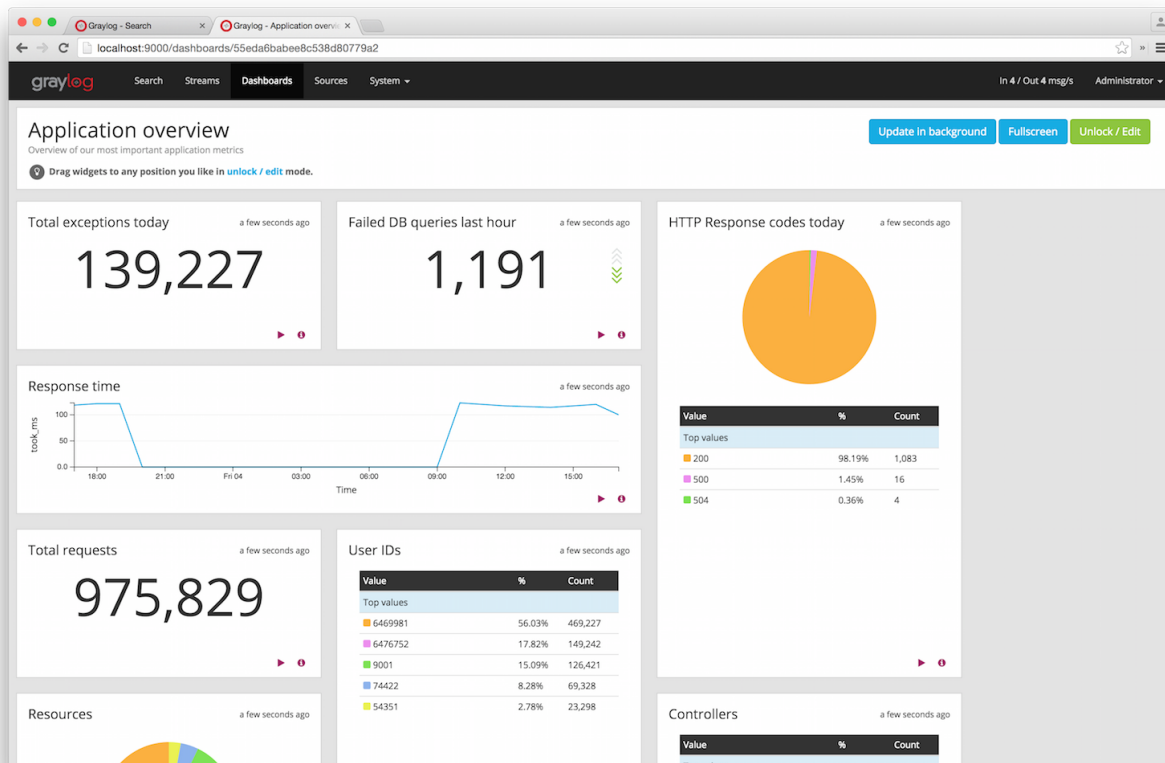
Dashboards

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.



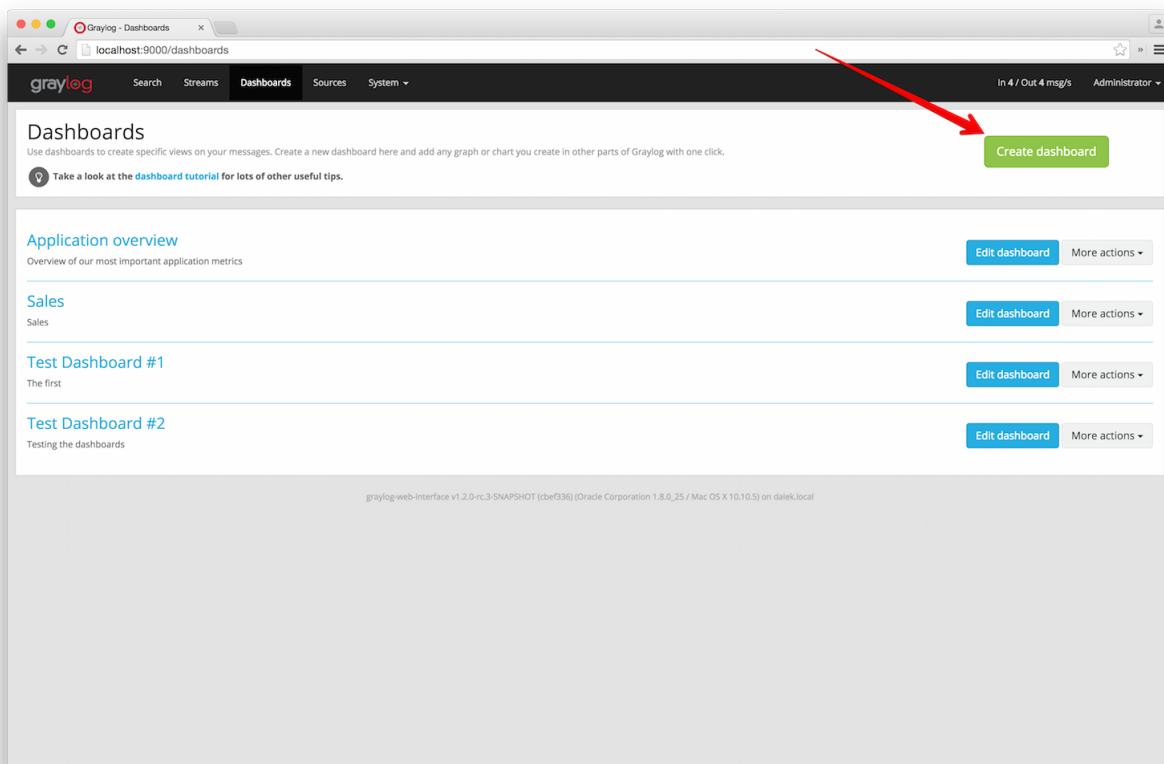
How to use dashboards

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.



Adding widgets

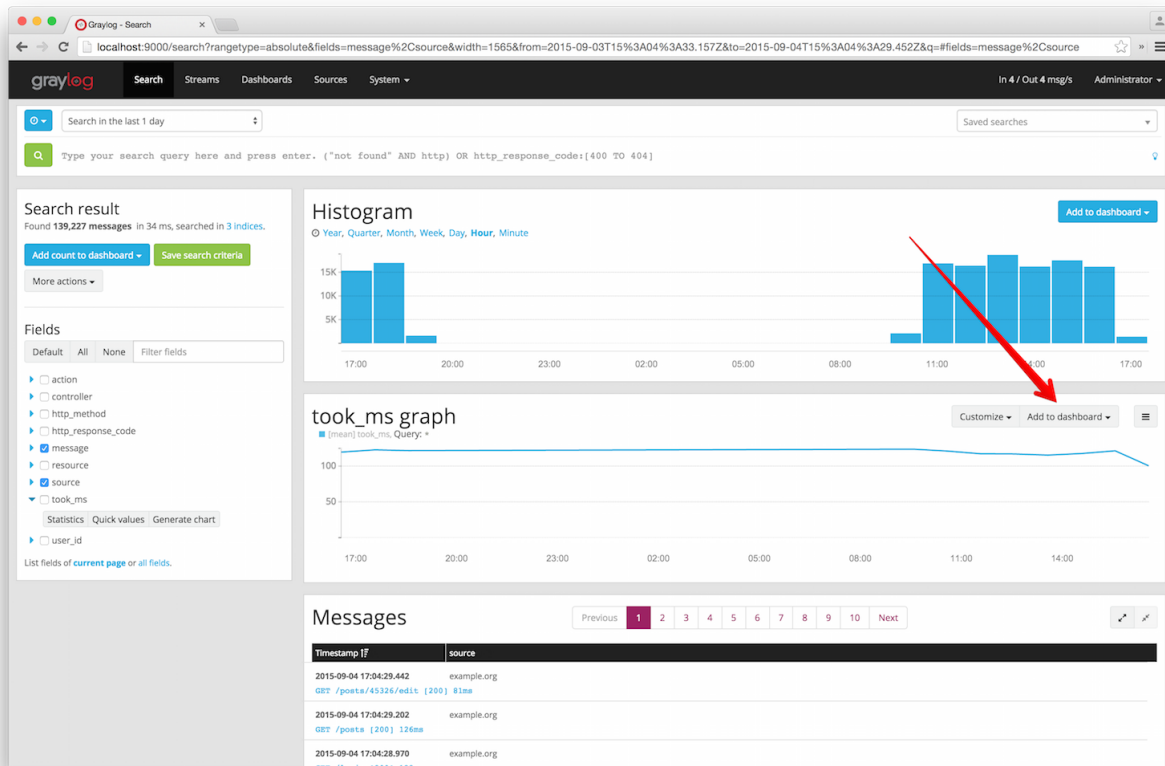
You should have your empty dashboard in front of you. Let's add some widgets! You can add search result information to dashboards with a couple of clicks. The following search result types can be added to dashboards:

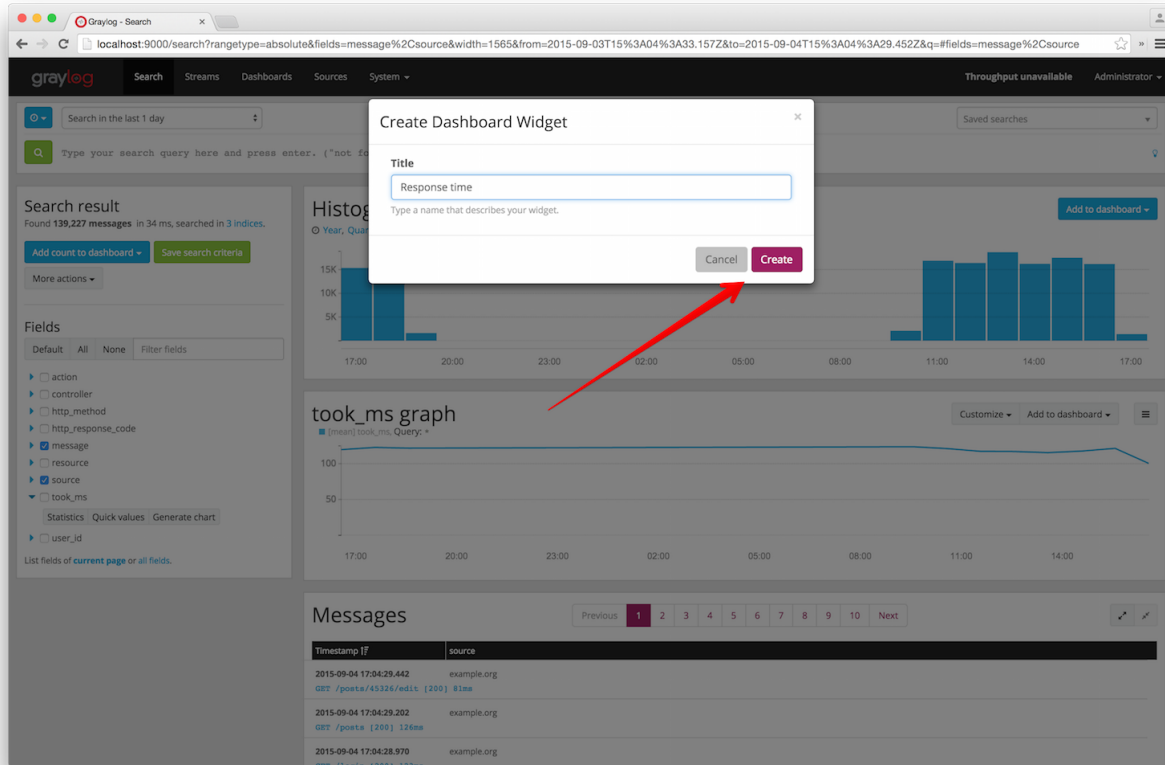
- Search result counts
- Search result histogram charts
- Statistical values
- Field value charts

- Stacked charts
- Quick values results

You can learn more about the different widget types in [Widget types explained](#).

Once you can see the results of your search, you will see buttons with the “Add to dashboard” text, that will allow you to select the dashboard where the widget will be displayed, and configure the widget.





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.

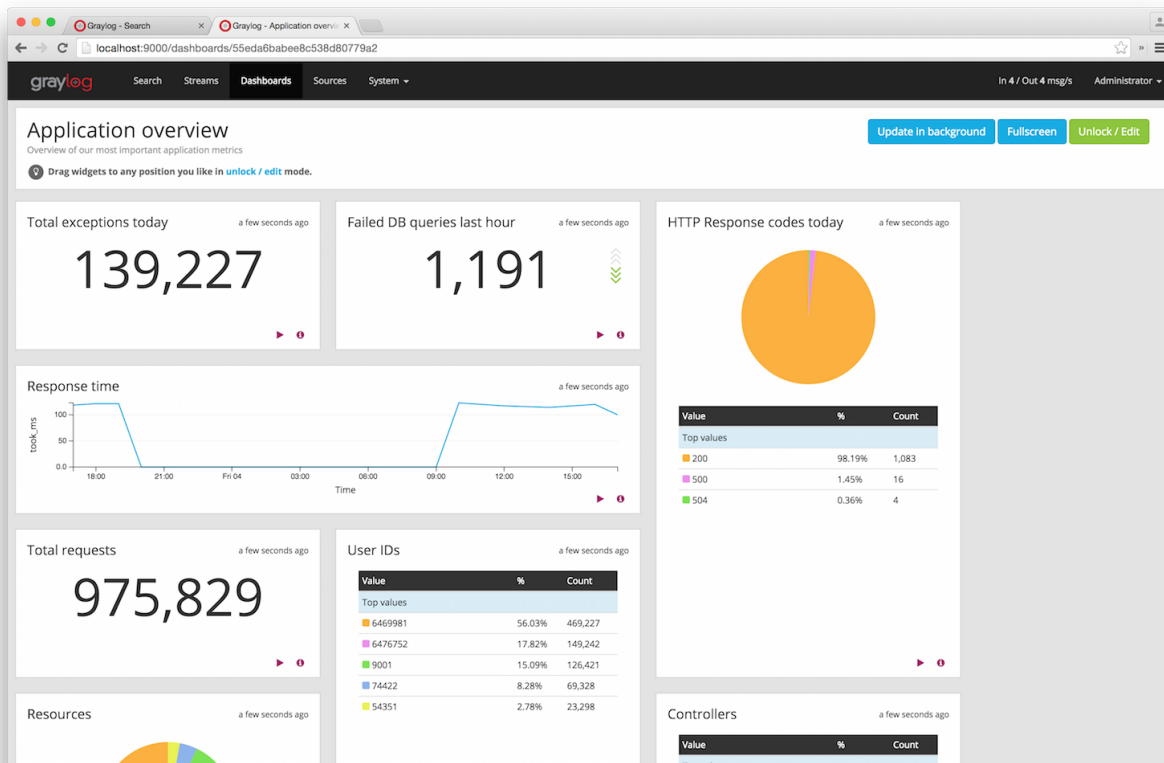
- **Top log sources today**
 - Example search: `*`, timeframe: Last 24 hours
 - Expand the `source` field in the the sidebar and hit *Quick values*
 - Add quick values to dashboard
- **Number of exceptions in a given app today**
 - Example search: `source:myapp AND Exception`, timeframe: Last 24 hours
 - 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, and edit widgets in the next chapter.



Widget types explained

Graylog supports a wide variety of widgets that allow you to quickly visualize data from your logs. This section intends to give you some information to better understand each widget type, and how they can help you to see relevant details from the many logs you receive.

Search result counts

This kind of widget includes a count of the number of search results for a given search. It can help you to quickly visualize things like the number of exceptions an application logs, or the number of requests your site receives.

All search result counts created with a relative time frame can additionally display trend information. The trend is calculated by comparing the count for the given time frame, with the one resulting from going further back the same amount of time. For example, to calculate the trend in a search result count with a relative search of *5 minutes ago*, Graylog will count the messages in the last 5 minutes, and compare that with the count of the previous 5 minutes.

Search result histogram charts

The search result histogram displays a chart using the time frame of your search, graphing the number of search result counts over time. It may help you to visualize how the number of request to your site change over time, or to see how many downloads a file has over time.

Changing the graph resolution, you can decide how much time each bar of the graph represents.

Statistical values

You can add to your dashboard any statistical value calculated for a field. This may help you to see the mean time response for your application, or how many unique servers are handling requests to your application, by using the cardinality value of that field. Please refer to [Field statistics](#) for more information on the available statistical functions and how to display them in your searches.

As with search result counts, you can also add trend information to statistical value widgets created with a relative time frame.

Field value charts

To draw an statistical value over time, you can use a field value chart. They could help you to see the evolution of the number of unique users visiting your site in the last week. In the [Field graphs](#) section we explain how to create these charts and ways you can customize them.

Stacked charts

Stacked charts group several field value charts under the same axes. They let you compare different values in a compact way, like the number of visits to two different websites. As explained in [Field graphs](#), stacked charts are basically field value charts represented in the same axes.

Quick values results

In order to show a list of values a certain field contains and their distribution, you can use a quick value widget. This may help you to see the percentage of failed requests in your application, or which parts of your application experience more problems. Please refer to [Quick values](#) to see how to request this information in your search result page.

The quick values information can be represented as a pie chart and/or as a table, so you can choose what is the best fit for your needs.

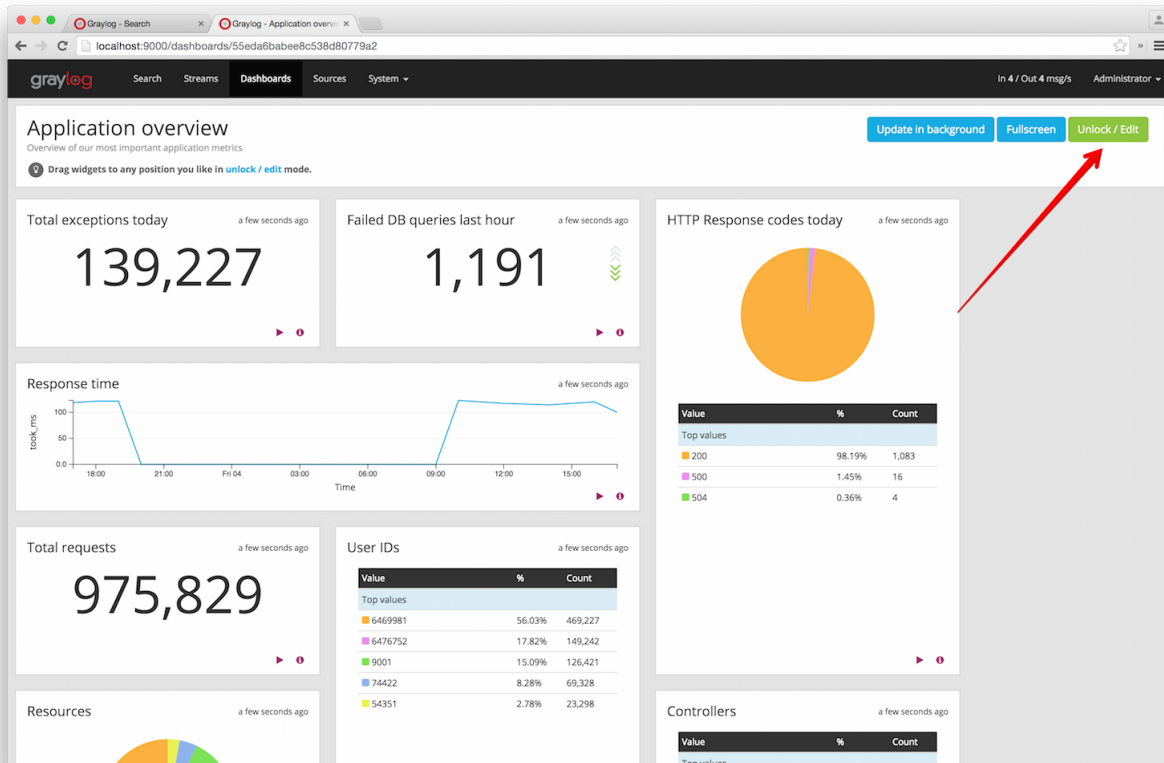
Modifying dashboards

You need to *unlock* dashboards to make any changes to them. Hit the “Unlock/Edit” button in the top right corner of a dashboard to unlock it. You should now see different icons at the bottom of each widget, that allow you to perform more actions.

Unlocked dashboard widgets explained

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

- Delete widget
- Edit widget configuration
- Change widget size (when you hover over the widget)



Widget cache times

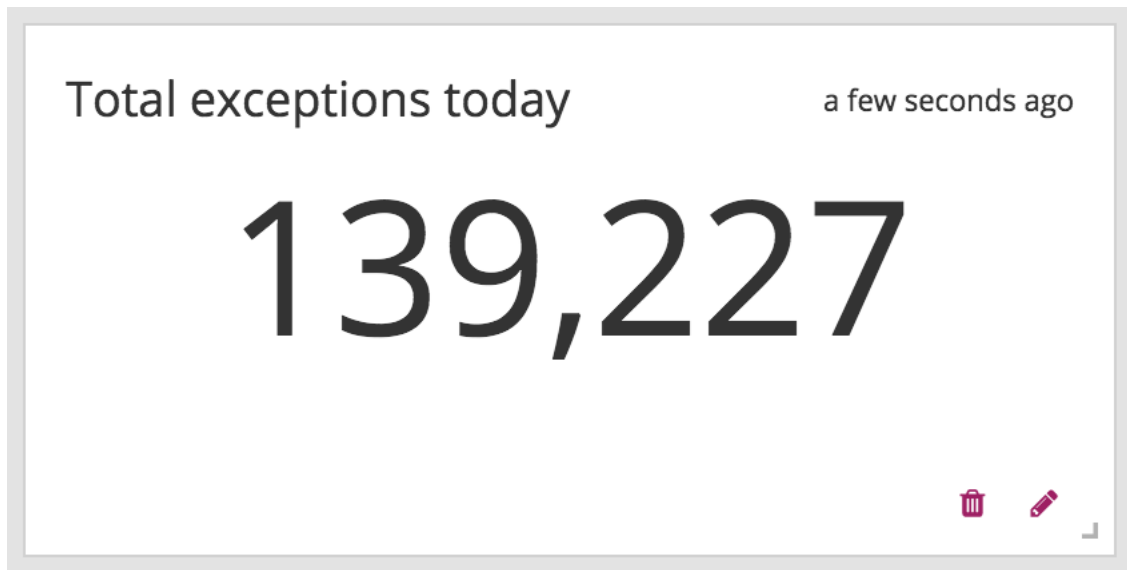
Widget values are cached in `graylog-server` by default. **This means that the cost of value computation does not grow with every new device or even browser tab displaying a dashboard.** Some widgets might need to show real-time information (set cache time to 1 second) and some widgets might be updated 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.

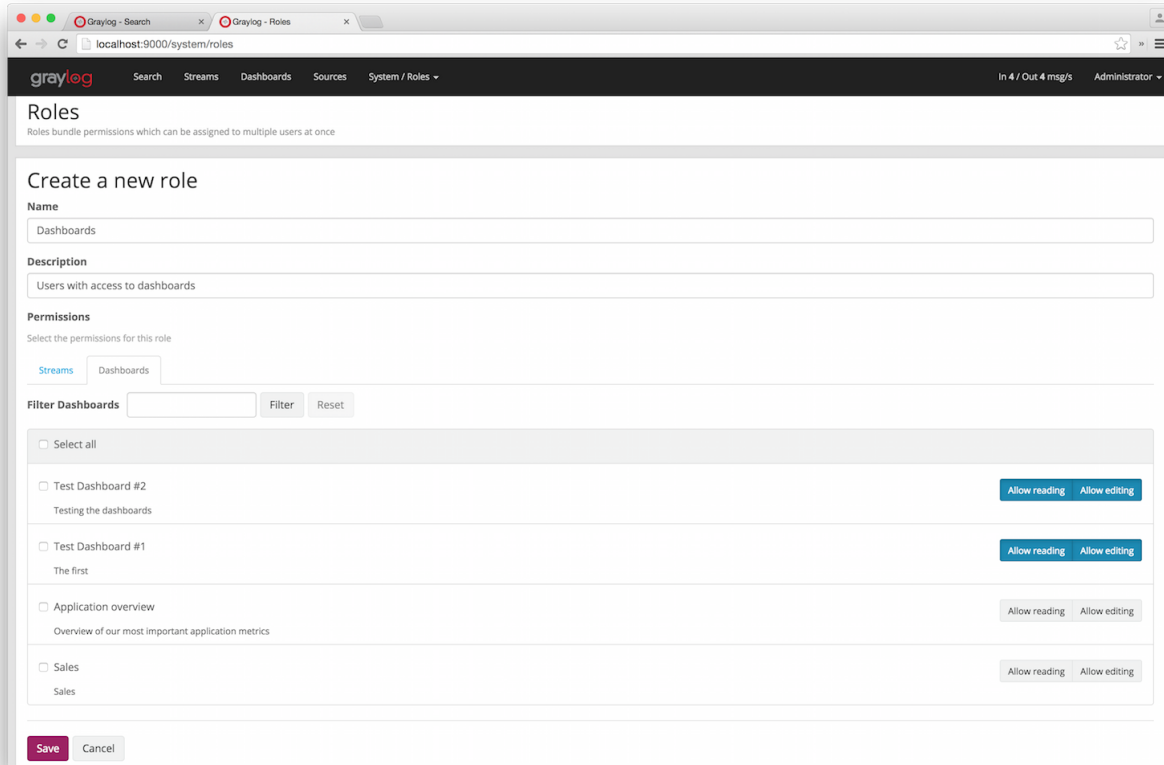
Resizing widgets

When hovering over a widget, you will see that a gray arrow appears in its bottom-right corner. You can use that icon to resize widgets. Their contents will adapt to the new size automatically!



Dashboard permissions

Graylog users in the *Admin* role are always allowed to view and edit all dashboards. Users in the *Reader* role are by default not allowed to view or edit **any** dashboard.



Navigate to *System -> Roles* and create a new role that grant the permissions you wish. You can then assign that new role to any users you wish to give dashboard permissions in the *System -> Users* page.

You can read more about user permissions and roles under *Users and Roles*.

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.

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

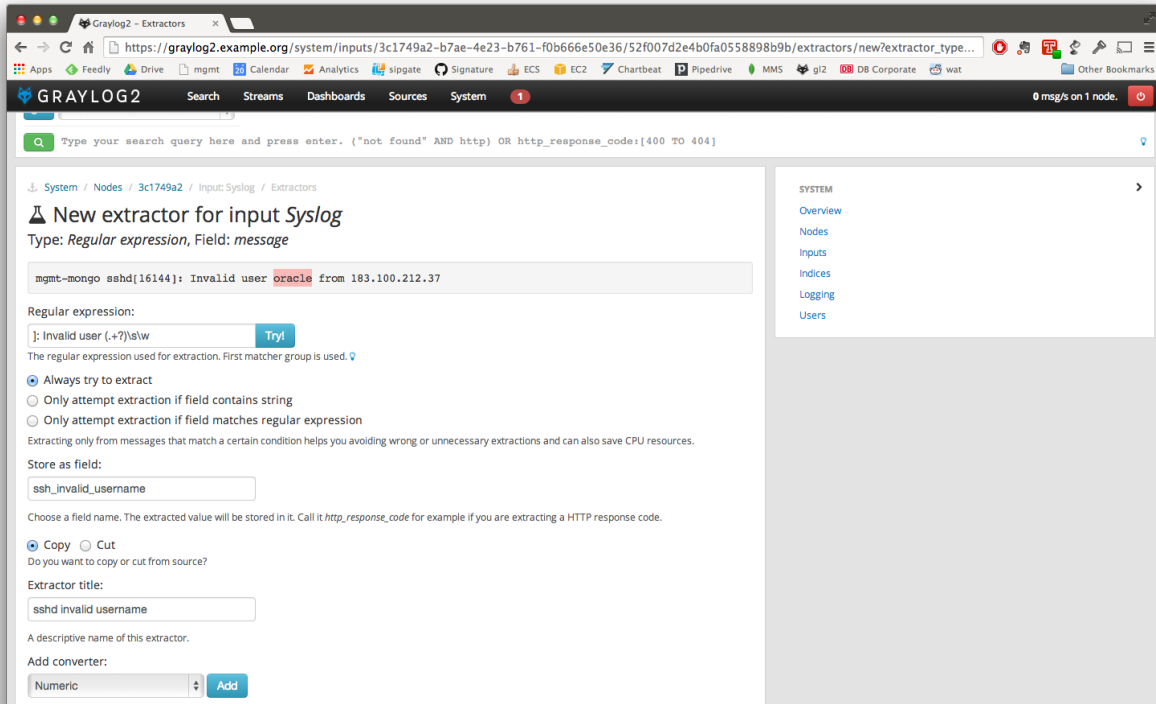
The extractors allow you to instruct Graylog nodes about how to extract data from any text in the received message (no matter from which format or if an already extracted field) to message fields. You may already know why structuring data into fields is important if you are using Graylog: There are a 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.

Attention: Graylog extractors only work on text fields but won’t be executed for numeric fields or anything other than a string.

Creating extractors is possible via either Graylog REST API calls or from the web interface using a wizard. Select a message input on the *System -> Inputs* page and hit *Manage extractors* in the actions menu. The wizard allows you to load a message to test your extractor configuration against. You can extract data using for example regular expressions,

Grok patterns, substrings, or even by splitting the message into tokens by separator characters. The wizard looks like this and should be pretty intuitive:



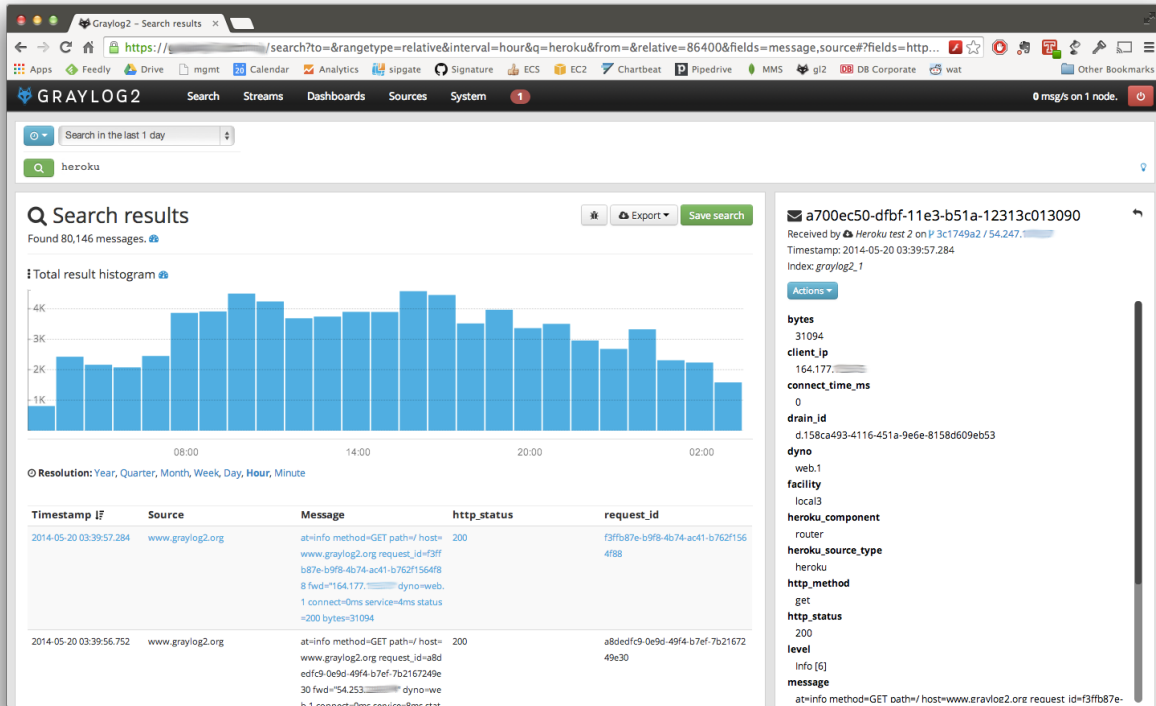
You can also choose to apply so called *converters* on the extracted value to for example convert a string consisting of numbers to an integer or double value (important for range searches later), anonymize IP addresses, lower-/uppercase a string, build a hash value, and much more.

Import extractors

The recommended way of importing extractors in Graylog is using *Content packs*. The [Graylog Marketplace](#) provides access to many content packs that you can easily download and import into your Graylog setup.

You can still import extractors from JSON if you want to. Just copy the JSON extractor export into the import dialog of a message input of the fitting type (every extractor set entry in the directory tells you what type of input to spawn, e. g. syslog, GELF, or Raw/plaintext) and you are good to go. The next messages coming in should already include the extracted fields with possibly converted values.

A message sent by Heroku and received by Graylog with the imported *Heroku* extractor set on a plaintext TCP input looks like this: (look at the extracted fields in the message detail view)



Using regular expressions to extract data

Extractors support matching field values using regular expressions. Graylog uses the `Java Pattern` class to evaluate regular expressions.

For the individual elements of regular expression syntax, please refer to Oracle's documentation, however the syntax largely follows the familiar regular expression languages in widespread use today and will be familiar to most.

However, one key question that is often raised is matching a string in case insensitive manner. Java regular expressions are case sensitive by default. Certain flags, such as the one to ignore case sensitivity can either be set in the code, or as an inline flag in the regular expression.

For example, to create an extractor that matches the browser name in the following user agent string:

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.102
```

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]|1[0-9]|0[0-9])?([0-9]|1[0-9]|2[0-4]|3[0-9])?)?){1,5}(:[0-9A-Fa-f]{1,4})?|::))
IPV4 (?<![0-9]) (?:(?:25[0-5]|2[0-4][0-9]|0[0-1]?[0-9])?([0-9]|1[0-9]|2[0-4]|3[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	<code>%{NUMBER:fieldname;byte}</code>
short	-32768 ... 32767	<code>%{NUMBER:fieldname;short}</code>
int	-2 ³¹ ... 2 ³¹ -1	<code>%{NUMBER:fieldname;int}</code>
long	-2 ⁶³ ... 2 ⁶³ -1	<code>%{NUMBER:fieldname;long}</code>
float	32-bit IEEE 754	<code>%{NUMBER:fieldname;float}</code>
double	64-bit IEEE 754	<code>%{NUMBER:fieldname;double}</code>
boolean	<i>true, false</i>	<code>%{DATA:fieldname;boolean}</code>
string	Any UTF-8 string	<code>%{DATA:fieldname;string}</code>
date	See SimpleDateFormat	<code>%{DATA:timestamp;date;dd/MMM/yyyy:HH:mm:ss Z}</code>
datetime	Alias for <i>date</i>	

There are many resources on 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.

Using the JSON extractor

Since version 1.2, Graylog also supports extracting data from messages sent in JSON format.

Using the JSON extractor is easy: once a Graylog input receives messages in JSON format, you can create an extractor by going to *System -> Inputs* and clicking on the *Manage extractors* button for that input. Next, you need to load a message to extract data from, and select the field containing the JSON document. The following page lets you add some extra information to tell Graylog how it should extract the information. Let's illustrate how a message would be extracted with an example message:

```
{"level": "ERROR", "details": {"message": "This is an example error message", "controller": "IndexController"}}
```

Using the default settings, that message would be extracted into these fields:

details.tags one, two, three

level ERROR

details.controller IndexController

details.message This is an example error message

In the create extractor page, you can also customize how to separate list of elements, keys, and key/values. It is also possible to flatten JSON structures or expand them into multiple fields, as shown in the example above.

Automatically extract all key=value pairs

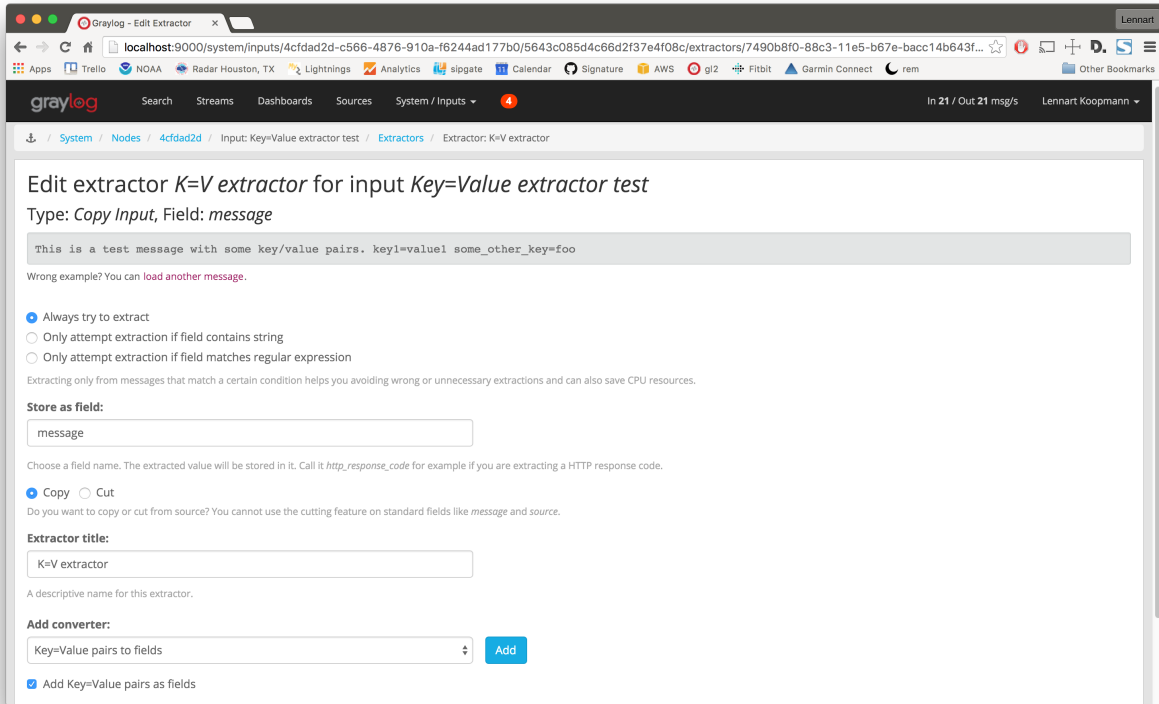
Sometimes you will receive messages like this:

```
This is a test message with some key/value pairs. key1=value1 some_other_key=foo
```

You might want to extract all `key=value` pairs into Graylog message fields without having to specify all possible key names or even their order. This is how you can easily do this:

Create a new extractor of type “Copy Input” and select to read from the field `message`. (Or any other string field that contains `key=value` pairs.) Configure the extractor to store the (copied) field value to the same field. In this case `message`. The trick is to add the “Key=Value pairs to fields” converter as last step. Because we use the “Copy Input” extractor, the converter will run over the complete field you selected and convert all `key=value` pairs it can find.

This is a screenshot of the complete extractor configuration:



... and this is the resulting message:

The screenshot shows the Graylog Messages interface. At the top, there's a 'Messages' header with navigation buttons 'Previous', '1' (selected), and 'Next'. Below the header, a table lists messages with columns 'Timestamp' and 'source'. The selected message is from '2015-11-11 16:27:59.310' and 'sundaysister.local'. The message body is 'This is a test message with some key/value pairs. key1=value1 some_other_key=foo'. Below the message body, there's a search bar and a 'Permalink' button. To the right, there are buttons for 'Copy ID' and 'Test against stream'. Below the message body, there's a list of fields and their values: 'Received by' (Key=Value extractor test on 4cfdad2d / 10.1.10.79), 'Stored in index' (graylog2_2), 'Routed into streams' (Forward to Splunk), 'facility' (gelf-rb), 'file' (irb), 'key1' (value1), 'level' (6), 'line' (5), 'message' (This is a test message with some key/value pairs. key1=value1 some_other_key=foo), 'some_other_key' (foo), 'source' (sundaysister.local), and 'version' (1.0). Each field has a search icon to its right.

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=192.168.70.66
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
```

Extracting most of the data is not a problem and can be done easily. Using the date in the message (*Mar 12 00:45:38*) as Graylog message timestamp however needs to be done with a date parser converter.

Use a copy input extractor rule to select the timestamp and apply the *Date* converter with a format string:

```
MMM dd HH:mm:ss
```

(format string table at the end of this page)

Store as field:

timestamp

Choose a field name. The extracted value will be stored in it. Call it *http_response_code* for example if you are extracting a HTTP response code.

☒ Copy ☐ Cut

Do you want to copy or cut from source?

Extractor title:

Timestamp

A descriptive name of this extractor.

Add converter:

Date

Add


☒ Convert to date type


Format string: ?

MMM dd HH:mm:ss

Please note that you cannot use the cutting feature on standard fields like *message* and *source*.

Create extractor

✉ 4765e370-aa42-11e3-a7dd-4c8d79f2b596 

Received by  Cisco System Messages on [fb66b27e](#) / 10.226.163.44

Timestamp: 2014-03-12 00:45:38.000

Index: *graylog2_356*

Actions ▾

facility

local0

level

Error [3]

local_facility

link

local_level

3

message

Interface GigabitEthernet0/31, changed state to down

source

foo-bar-dc3-org-de01

type

updown

Standard date converter format string table

Symbol	Meaning	Presentation	Examples
G	era	text	AD
C	century of era (≥ 0)	number	20
Y	year of era (≥ 0)	year	1996
x	weekyear	year	1996
w	week of weekyear	number	27
e	day of week	number	2
E	day of week	text	Tuesday; Tue
y	year	year	1996
D	day of year	number	189
M	month of year	month	July; Jul; 07
d	day of month	number	10
a	halfday of day	text	PM
K	hour of halfday (0~11)	number	0
h	clockhour of halfday (1~12)	number	12
H	hour of day (0~23)	number	0
k	clockhour of day (1~24)	number	24
m	minute of hour	number	30
s	second of minute	number	55
S	fraction of second	millis	978
z	time zone	text	Pacific Standard Time; PST
Z	time zone offset/id	zone	-0800; -08:00; America/Los_Angeles
'	escape for text	delimiter	
'	single quote	literal	'

The flexible date converter

Now imagine you had one of those devices that send messages that are not so easy to parse because they do not follow a strict timestamp format. Some network devices for example like to send days of the month without adding a padding 0 for the first 9 days. You'll have dates like `Mar 9` and `Mar 10` and end up having problems defining a parser string for that. Or maybe you have something else that is really exotic like just *last wednesday* as timestamp. The flexible date converter is accepting any text data and tries to build a date from that as good as it can.

Examples:

- **Mar 12**, converted at 12:27:00 UTC in the year 2014: 2014-03-12T12:27:00.000
- **2014-3-12 12:27**: 2014-03-12T12:27:00.000
- **Mar 12 2pm**: 2014-03-12T14:00:00.000

Note that the flexible date converter is using UTC as time zone by default unless you have time zone information in the parsed text or have configured another time zone when adding the flexible date converter to an extractor (see this [comprehensive list of time zones](#) available for the flexible date converter).

Processing Pipelines

Graylog's new processing pipelines plugin allows greater flexibility in routing, blacklisting, modifying and enriching messages as they flow through Graylog.

Warning: This plugin is still under development and will not be as stable and fast as the rest of Graylog at this moment!

Pipelines and rules are no longer configuration for pre-build code (like extractors and stream rules are) but are represented as code, much like Drools rules are. This gives them their great flexibility and extensibility.

The language itself is very simple and can be extended by functions, which are fully pluggable.

The following pages introduce the concepts of pipelines, rules, stream connections and the built in functions.

Pipelines

Warning: This documentation is work in progress

Overview

Pipelines are the central concept which tie together the processing steps Graylog applies to your messages.

They contain rules and can be connected to streams, allowing fine grained control which processing is done on which kinds of messages.

Because processing rules are simply conditions followed by a list of actions and do not have control flow by themselves, pipelines have one additional concept: stages.

Think of stages as groups of conditions and actions which need to run in order. All stages with the same priority run at the same time across all connected pipelines. They provide the necessary control flow to decide whether or not to run the rest of a pipeline.

Pipeline structure

Internally pipelines are represented as code. Let's have a look at a simple example and understand what each part does:

```
pipeline "My new pipeline"
stage 1 match all
  rule "has firewall fields";
  rule "from firewall subnet";
stage 2 match either
  rule "geocode IPs";
  rule "anonymize source IPs";
end
```

This code snippet declares a new pipeline with the name `My new pipeline` which declares two stages.

Stages are run in the order of their given *priority* and aren't otherwise named. Stage priorities can be any integer, even negative ones. This allows flexible ordering to run certain rules before or after others, even without modifying existing pipelines, which can come in handy when dealing with changing data formats.

For example, if there was a second pipeline declared, which contained a stage with the priority 0, that would run before either of the ones from the example. The order in which stages are declared is irrelevant, they are sorted according to their priority.

Stages then list the *rule references* they want to be executed as well as declaring whether *any or all* rules' conditions need to be satisfied to continue to evaluate the pipeline.

In our example, imagine rule *"has firewall fields"* checks for the presence of two message fields: `src_ip` and `dst_ip` but does not have any actions to run. Then for messages that do not have both fields, the condition would be false, and the pipeline would be aborted after stage 1, because the stage requires that *all* rules need to be satisfied. Stage 2 would not begin to run in this case. `match either` acts as an OR operator, only requiring a single (or more) rules to match. Actions are still being run for all matching rules, even if the pipeline stops after the stage.

Rules are referenced by their names and can thus be shared among many different pipelines. The intention is to create building blocks which then make it easier to process the data specific to your organization or use case.

Read more about [Rules](#) in the next section.

Rules

Warning: This documentation is work in progress

Overview

Rules are the cornerstone of the processing pipelines. They contain the logic about how to change, enrich route and drop messages.

To avoid the complexities of a complete programming language, Graylog supports a small rule language to express the processing logic. The rule language is limited on purpose to allow for easier understanding, better runtime optimization and fast learning.

The real work of rules is done in *functions* which are completely pluggable. Graylog already ships with a great number of built-in functions that range from converting data types over string processing, like `substring`, `regex` etc, to JSON parsing.

We expect that special purpose functions will be written and shared by the community, allow for faster innovation and problem solving than previously possible.

Rule structure

Picking up from the previous example in the [Pipelines](#) section, let's look at examples of some of the rules we've referenced:

```
rule "has firewall fields"
when
  has_field("src_ip") && has_field("dst_ip")
then
end
```

```
rule "from firewall subnet"
when
  cidr_match("10.10.10.0/24", to_ip($message.gl2_remote_ip))
then
end
```

Firstly, apart from naming the rule, their structure follows a simple *when, then* pattern. In the *when* clause we specify a boolean expression which is evaluated in the context of the current message in the pipeline. These are the conditions that are being used by the pipeline processor to determine whether to run a rule and collectively whether to continue in a pipeline.

Note that we are already calling the built-in function `has_field` with a field name. In the rule *has firewall fields* we make sure the message contains both `src_ip` as well as `dst_ip` as we want to use them in a later stage.

The rule has no actions to run, because we are only interested in using it as a condition at this point.

The second rule uses another built-in function `cidr_match`. That function takes a [CIDR pattern](#) and an IP address. In this case we reference a field from the currently processed message using the message reference syntax `$message`.

The field `gl2_remote_ip` is always set by Graylog upon receiving a message, so we do not check whether that field exists, otherwise we would have used another `has_field` function call to make sure it is there.

However, note the call to `to_ip` around the field reference. This is necessary because the field is stored as a *string* internally. In order to successfully match the CIDR pattern, we need to convert it to an IP address.

This is an important feature of Graylog's rule language, it enforces type safety to ensure that you end up with the data in the correct format. All too often everything is treated as a string, which wastes enormous amounts of cycles to convert data all the time as well as preventing to do proper analysis over the data.

Again we have no actions to immediately run, so the *then* block is empty.

Data Types

As we have seen in the previous section, we need to make sure to use the proper data types when calling functions.

Graylog's rule language parser rejects invalid use of types, making it safe to write rules.

The six built-in types in Graylog are `string` (a UTF-8 string), `double` (corresponds to Java's `Double`), `long` (Java's `Long`), `boolean` (Boolean), `void` (indicating a function has no return value to prevent it being used in a condition) and `ip` (a subset of `InetAddress`), but plugins are free to add additional types as they see fit. The rule processor takes care of ensuring that values and functions agree on the types being used.

Conventionally functions that convert types start with the prefix `to`, please refer to the [Functions](#) index for a list.

Conditions

In Graylog's rules the **when** clause is a boolean expression, which is evaluated against the processed message.

Expressions support the common boolean operators AND (or `&&`), OR (`||`), NOT (`!`) and comparison operators (`<`, `<=`, `>`, `>=`, `==`, `!=`).

Additionally any function that returns a value can be called (e.g. `route_to_stream` does not return a value) but the resulting expression must eventually be a boolean.

The condition must not be empty, but can instead simply use the boolean literal `true` in case you always want to execute the actions inside the rule.

If a condition calls a function which is not present, e.g. due to a missing plugin, the call evaluates to false instead.

Actions

A rule's **then** clause contains a list of actions which are evaluated in the order they appear.

There are two different types of actions:

Function calls # Variable assignments

Function calls look exactly like they do in conditions and all of the functions in the system can be used, including the functions that do not return a value.

Variable assignments have the following form:

```
let name = value;
```

They are useful to avoid recomputing expensive parsing of data, holding on to temporary values or making rules more readable.

Variables need to be defined before they can be used and can be accessed using the `name.field` notation in any place where a value is required.

The list of actions can also be empty, turning the rule into a pure condition which can be useful in combination with stages to guide the processing flow.

Stream connections

Warning: This documentation is work in progress

Overview

Pipelines by themselves do not process any messages. Instead they can be connected to streams, allowing fine grained control over which message types are being processed by which pipelines.

Using the built-in function `route_to_stream` a rule action can cause a message to be routed to the given stream. The pipeline engine will then look up the pipelines connected to that stream and start evaluating the additional pipelines.

To begin processing pipelines the incoming messages must therefore be directed to the initial set of pipelines.

The default stream

All messages received by Graylog are initially on what is called the *default stream*. This merely means they aren't routed to any specific stream yet. The default stream by itself is no explicit stream, it doesn't show up anywhere, but the pipeline processor still allows you to connect pipelines to it.

These initial pipelines are the entry to pipeline processing, allowing messages to be routed to more streams and being processed subsequently.

However, if you prefer to use the original stream matching functionality, you can configure the pipeline processor to run after the *message filter chain* and connect pipelines to existing streams. This gives you fine grained control over the extraction, conversion and enrichment process.

Functions

Warning: This documentation is work in progress

Overview

Functions are the means of how to interact with the messages Graylog processes.

They are written in Java and are pluggable, allowing extending the capabilities of Graylog in a simple manner.

Conceptually a function receives parameters, the current message context and returns a value. The data types of its return value and parameters determines where in the rules it can be used. Graylog makes sure the rules are sound from a data type perspective.

A function's parameters can be passed as named pairs or by position, as long as optional parameters are declared as coming last. The function's documentation below indicates which parameters are optional.

Let's look at a small example to illustrate these properties:

```
rule "function howto"
when
  has_field("transaction_date")
then
  // the following date format assumes there's no time zone in the string
  let new_date = parse_date(to_string($message.transaction_date), "yyyy-MM-dd HH:mm:ss");
  set_field("transaction_year", new_date.year);
end
```

In this example, we check if the current message contains the field `some_date` and then, after converting it to a string, try to parse it according to the format string `yyyy-MM-dd HH:mm:ss`, so for example the string `2016-03-05 14:45:02` would match. `Parse date` returns a `DateTime` object from the Java Joda-Time library, allowing easier access to the date's components.

We then add the transaction's year as a new field to the message.

You'll note that we haven't said in which time zone the timestamp is in, but still Graylog had to pick one (Graylog never relies on the local time of your server as that makes it nearly impossible to figure out why date handling came up with its result).

The reason Graylog knows which timezone to pick is because `parse_date` actually takes three parameters rather than the two we've given it in this example. The third one is a `String` called `timezone` and is optional. It has a default value of `"UTC"`.

Let's assume we have another field in the message, called `transaction_timezone`. It is send by the application and contains the time zone ID the transaction was done in (hopefully no application in the world sends its data like this, though):

```
rule "function howto"
when
  has_field("transaction_date") && has_field("transaction_timezone")
then
  // the following date format assumes there's no time zone in the string
  let new_date = parse_date(
    to_string($message.transaction_date),
    "yyyy-MM-dd HH:mm:ss",
    to_string($message.transaction_timezone)
  );
  set_field("transaction_year", new_date.year);
end
```

Now, if the time zone is specified in the other field, we pass it to `parse_date`.

In this case we only have a single optional parameter, which makes it easy to simply omit it from the end of the function call. However, if there are multiple ones, or if there are many parameters and it gets difficult to keep track of which positions correspond to which parameters, you can also use the named parameter variant of function calls. In this mode the order of the parameters does not matter, but all required ones still need to be there.

In our case the alternative version of calling `parse_date` would look like this:

```
rule "function howto"
when
  has_field("transaction_date") && has_field("transaction_timezone")
then
  // the following date format assumes there's no time zone in the string
  let new_date = parse_date(
    value: to_string($message.transaction_date),
    pattern: "yyyy-MM-dd HH:mm:ss",
    timezone: to_string($message.transaction_timezone)
  );
  set_field("transaction_year", new_date.year);
end
```

All parameters in Graylog's processing functions are named, please refer to the function index.

Function Index

The following list describes the built in functions that ship with Graylog. Additional third party functions are available via other plugins in the marketplace.

Table 14.1: Built-in Functions

Name	Description
<i>to_bool</i>	Converts the single parameter to a boolean value using its string value.
<i>to_double</i>	Converts the first parameter to a double floating point value.
<i>to_long</i>	Converts the first parameter to a long integer value.
<i>to_string</i>	Converts the first parameter to its string representation.
<i>abbreviate</i>	Abbreviates a String using ellipses.
<i>capitalize</i>	Capitalizes a String changing the first letter to title case.
<i>uncapitalize</i>	Uncapitalizes a String changing the first letter to lower case.

Continued on next page

Table 14.1 – continued from previous page

Name	Description
<i>uppercase</i>	Converts a String to upper case.
<i>lowercase</i>	Converts a String to lower case.
<i>swapcase</i>	Swaps the case of a String.
<i>contains</i>	Checks if a string contains another string.
<i>substring</i>	Returns a substring of <i>value</i> with the given start and end offsets.
<i>regex</i>	Match a regular expression against a string, with matcher groups.
<i>crc32</i>	Returns the hex encoded CRC32 digest of the given string.
<i>crc32c</i>	Returns the hex encoded CRC32C (RFC 3720, Section 12.1) digest of the given string.
<i>md5</i>	Returns the hex encoded MD5 digest of the given string.
<i>murmur3_32</i>	Returns the hex encoded MurmurHash3 (32-bit) digest of the given string.
<i>murmur3_128</i>	Returns the hex encoded MurmurHash3 (128-bit) digest of the given string.
<i>sha1</i>	Returns the hex encoded SHA1 digest of the given string.
<i>sha256</i>	Returns the hex encoded SHA256 digest of the given string.
<i>sha512</i>	Returns the hex encoded SHA512 digest of the given string.
<i>now</i>	Returns the current date and time.
<i>parse_date</i>	Parses a date and time from the given string, according to a strict pattern.
<i>flex_parse_date</i>	Attempts to parse a date and time using the Natty date parser.
<i>format_date</i>	Formats a date and time according to a given formatter pattern.
<i>parse_json</i>	Parse a string into a JSON tree.
<i>select_jsonpath</i>	Selects one or more named JSON Path expressions from a JSON tree.
<i>to_ip</i>	Converts the given string to an IP object.
<i>cidr_match</i>	Checks whether the given IP matches a CIDR pattern.
<i>from_input</i>	Checks whether the current message was received by the given input.
<i>route_to_stream</i>	Assigns the current message to the specified stream.
<i>create_message</i>	Currently incomplete Creates a new message which will be evaluated by the entire processing pipeline.
<i>drop_message</i>	This currently processed message will be removed from the processing pipeline after the rule finishes.
<i>has_field</i>	Checks whether the currently processed message contains the named field.
<i>remove_field</i>	Removes the named field from the currently processed message.
<i>set_field</i>	Sets the name field to the given value in the currently processed message.
<i>set_fields</i>	Sets multiple fields to the given values in the currently processed message.

to_bool

```
to_bool(any)
```

Converts the single parameter to a boolean value using its string value.

to_double

```
to_double(any, [default: double])
```

Converts the first parameter to a double floating point value.

to_long

```
to_long(any, [default: long])
```

Converts the first parameter to a long integer value.

to_string

```
to_string(any, [default: string])
```

Converts the first parameter to its string representation.

abbreviate

```
abbreviate(value: string, width: long)
```

Abbreviates a String using ellipses, the width defines the maximum length of the resulting string.

capitalize

```
capitalize(value: string)
```

Capitalizes a String changing the first letter to title case.

uncapitalize

```
uncapitalize(value: string)
```

Uncapitalizes a String changing the first letter to lower case.

uppercase

```
uppercase(value: string, [locale: string])
```

Converts a String to upper case. The locale (IETF BCP 47 language tag) defaults to “en”.

lowercase

```
lowercase(value: string, [locale: string])
```

Converts a String to lower case. The locale (IETF BCP 47 language tag) defaults to “en”.

swapcase

```
swapcase(value: string)
```

Swaps the case of a String changing upper and title case to lower case, and lower case to upper case.

contains

```
contains(value: string, search: string, [ignore_case: boolean])
```

Checks if value contains search, optionally ignoring the case of the search pattern.

substring

```
substring(value: string, start: long, [end: long])
```

Returns a substring of `value` starting at the `start` offset (zero based indices), optionally ending at the `end` offset. Both offsets can be negative, indicating positions relative to the end of `value`.

regex

```
regex(pattern: string, value: string, [group_names: array[string]])
```

Match the regular expression in `pattern` against `value`. Returns a match object, with the boolean property `matches` to indicate whether the regular expression matched and, if requested, the matching groups as `groups`. The groups can optionally be named using the `group_names` array. If not named, the groups names are strings starting with "0".

crc32

```
crc32(value: string)
```

Creates the hex encoded CRC32 digest of the `value`.

crc32c

```
crc32c(value: string)
```

Creates the hex encoded CRC32C (RFC 3720, Section 12.1) digest of the `value`.

md5

```
md5(value: string)
```

Creates the hex encoded MD5 digest of the `value`.

murmur3_32

```
murmur3_32(value: string)
```

Creates the hex encoded MurmurHash3 (32-bit) digest of the `value`.

murmur3_128

```
murmur3_128(value: string)
```

Creates the hex encoded MurmurHash3 (128-bit) digest of the `value`.

sha1

```
sha1(value: string)
```

Creates the hex encoded SHA1 digest of the `value`.

sha256

```
sha256(value: string)
```

Creates the hex encoded SHA256 digest of the `value`.

sha512

```
sha512(value: string)
```

Creates the hex encoded SHA512 digest of the `value`.

now

```
now([timezone: string])
```

Returns the current date and time. Uses the default time zone UTC.

parse_date

```
parse_date(value: string, pattern: string, [timezone: string])
```

Parses the `value` into a date and time object, using the `pattern`. If no timezone is detected in the pattern, the optional `timezone` parameter is used as the assumed timezone. If omitted the `timezone` defaults to UTC.

The format used for the `pattern` parameter is identical to the pattern of the [Joda-Time DateTimeFormat](#).

Symbol	Meaning	Presentation	Examples
G	era	text	AD
C	century of era (≥ 0)	number	20
Y	year of era (≥ 0)	year	1996
x	weekyear	year	1996
w	week of weekyear	number	27
e	day of week	number	2
E	day of week	text	Tuesday; Tue
y	year	year	1996
D	day of year	number	189
M	month of year	month	July; Jul; 07
d	day of month	number	10
a	halfday of day	text	PM
K	hour of halfday (0~11)	number	0
h	clockhour of halfday (1~12)	number	12
H	hour of day (0~23)	number	0
k	clockhour of day (1~24)	number	24
m	minute of hour	number	30
s	second of minute	number	55
S	fraction of second	millis	978
z	time zone	text	Pacific Standard Time; PST
Z	time zone offset/id	zone	-0800; -08:00; America/Los_Angeles
'	escape for text	delimiter	
' '	single quote	literal	'

flex_parse_date

```
flex_parse_date(value: string, [default: DateTime], [timezone: string])
```

Uses the [Natty date parser](#) to parse a date and time value. If no timezone is detected in the pattern, the optional timezone parameter is used as the assumed timezone. If omitted the timezone defaults to UTC.

In case the parser fails to detect a valid date and time the `default` date and time is being returned, otherwise the expression fails to evaluate and will be aborted.

format_date

```
format_date(value: DateTime, format: string, [timezone: string])
```

Returns the given date and time value formatted according to the `format` string. If no timezone is given, it defaults to UTC.

parse_json

```
parse_json(value: string)
```

Parses the `value` string as JSON, returning the resulting JSON tree.

select_jsonpath

```
select_jsonpath(json: JsonNode, paths: Map<string, string>)
```

Evaluates the given `paths` against the `json` tree and returns the map of the resulting values.

to_ip

```
to_ip(ip: string)
```

Converts the given `ip` string to an `IpAddress` object.

cidr_match

```
cidr_match(cidr: string, ip: IpAddress)
```

Checks whether the given `ip` address object matches the `cidr` pattern.

from_input

```
from_input(id: string | name: string)
```

Checks whether the currently processed message was received on the given input. The input can be looked up by either specifying its `name` (the comparison ignores the case) or the `id`.

route_to_stream

```
route_to_stream(id: string | name: string, [message: Message])
```

Routes the message to the given stream. The stream can be looked up by either specifying its name or the id.

If message is omitted, this function uses the currently processed message.

This causes the message to be evaluated on the pipelines connected to that stream, unless the stream has already been processed for this message.

create_message

```
create_message([message: string], [source: string], [timestamp: DateTime])
```

Creates a new message with from the given parameters. If any of them is omitted, its value is taken from the currently processed message. If timestamp is omitted, the timestamp of the created message will be the timestamp at that moment.

drop_message

```
drop_message(message: Message)
```

The processing pipeline will remove the given message after the rule is finished executing.

If message is omitted, this function uses the currently processed message.

This can be used to implement flexible blacklisting based on various conditions.

has_field

```
has_field(field: string, [message: Message])
```

Checks whether the given message contains a field with the name field.

If message is omitted, this function uses the currently processed message.

remove_field

```
remove_field(field: string, [message: Message])
```

Removes the given field with the name field from the given message, unless the field is reserved.

If message is omitted, this function uses the currently processed message.

set_field

```
set_field(field: string, value: any, [message: Message])
```

Sets the given field named field to the new value. The field name must be valid, and specifically cannot include a . character. It is trimmed of leading and trailing whitespace. String values are trimmed of whitespace as well.

If message is omitted, this function uses the currently processed message.

set_fields

```
set_fields(fields: Map<string, any>, [message: Message])
```

Sets all of the given name-value pairs in `field` in the given message. This is a convenience function acting like *set_field*. It can be helpful for using the result of a function like *select_jsonpath* or *regex* in the currently processed message especially when the key names are the result of a regular expression.

If `message` is omitted, this function uses the currently processed message.

Message rewriting with Drools

Note: Since Graylog 2.0 you can use the *processing pipelines* for more flexible message rewriting.

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:

```
rule "Overwrite localhost"
  when
    m : Message( source == "localhost" )
  then
    m.addField("source", "localhost.example.com" );
    log.info("[Overwrite localhost] rule fired: {}", m);
  end

rule "Drop UDP and ICMP Traffic from firewall"
  when
```

```
m : Message( getField("full_message") matches "(?i).*(ICMP|UDP) Packet (.|\n|\r)*" && source =
then
  m.setFilterOut(true);
  log.info("[Drop UDP and ICMP Traffic from firewall] rule fired: {}", m);
end
```

The log object being used to write log messages from within Drools rules is an instance of the [SLF4J Logger](#) interface.

Parsing Message and adding fields

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

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

// Raw Syslog
// Example:
// 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=
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 address of the Squid proxy, which not very useful. This rule overwrites the source and adds other fields:

```
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+\\.\\d+\\.\\d+)");

  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
```

Blacklisting messages

You can also use Drools rules to blacklist messages. How to do this is described [here](#).

Blacklisting

Note: Since Graylog 2.0 you can use the *processing pipelines* for blacklisting.

If you have messages coming into Graylog that should be discarded before being written to Elasticsearch or forwarded to another system you can use *Drools rules* to perform custom filtering.

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](#).

How to

The general idea is simple: Any Message marked with `setFilterOut(true)` will be discarded when processed in the Graylog filter chain. You can either *write and load your own filter plugin* that can execute any Java code to mark messages or just use the *Drools rules*. The following example shows how to do this.

Based on regular expressions

Put this into your `rules_file`:

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

rule "Blacklist all messages that start with 'firewall'"
  when
    m : Message( message matches "^firewall.*" )
  then
    System.out.println("DEBUG: Blacklisting message."); // Don't do this in production.
    m.setFilterOut(true);
  end
```

This rule will blacklist any message that starts with the string “firewall” (matches `^firewall.*`).

Geolocation

Graylog lets you extract and visualize geolocation information from IP addresses in your logs. Here we will explain how to install and configure the geolocation resolution, and how to create a map with those geolocation.

Setup

The [Graylog Map Widget](#) is the plugin providing geolocation capabilities to Graylog. The plugin is compatible with Graylog 2.0.0 and higher, and it is installed by default, although **some configuration is still required on your side**. This section explains how to configure the plugin in detail.

In case you need to reinstall the plugin for some reason, you can find it inside the Graylog tarball in our [downloads page](#). Follow the instructions in *Installing and loading plugins* to install it.

Configure the database

In first place, you need to download a geolocation database. We currently support **MaxMind City databases** in the **MaxMind DB format**, as the [GeoIP2 City Database](#) or [GeoLite2 City Database](#) that MaxMind provides.

The next step is to store the geolocation database in all servers running Graylog. As an example, if you were using the Graylog OVA, you could save the database in the `/var/opt/graylog/data` folder, along with other data used by Graylog. Make sure you grant the right permissions so the user running Graylog can read the file.

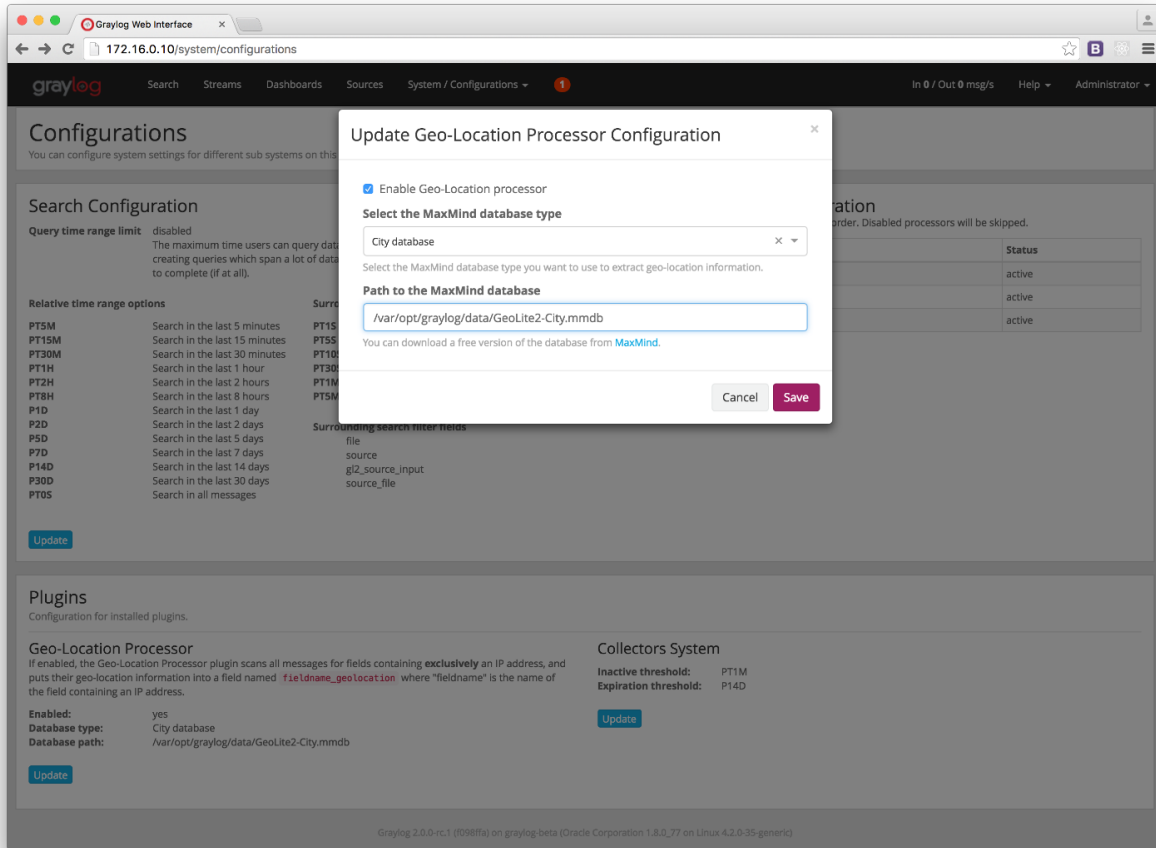
Then you need to configure Graylog to start using the geolocation database to resolve IPs in your logs. To do that, open Graylog web interface in your favourite browser, and go to *System -> Configurations*. You can find the geolocation configuration under the *Plugins / Geo-Location Processor* section, as seen in the screenshot.

The screenshot shows the Graylog Web Interface at the URL `172.16.0.10/system/configurations`. The page is titled "Configurations" and contains several sections:

- Search Configuration:** Includes a "Query time range limit" set to "disabled" and "Relative time range options" with a list of time ranges (PT5M to PT30D). It also has "Surrounding time range options" (PT1S to PT5M) and "Surrounding search filter fields" (file, source, gl2_source_input, source_file). An "Update" button is at the bottom.
- Message Processors Configuration:** A table showing three processors: GeolP Resolver (active), Pipeline Processor (active), and Message Filter Chain (active). An "Update" button is below the table.
- Plugins:** A section for installed plugins, including the "Geo-Location Processor". It has fields for "Enabled" (checked), "Database type" (set to "No database"), and "Database path" (set to "tmp/GeoLite2-City.mmdb"). An "Update" button is at the bottom.
- Collectors System:** Shows "Inactive threshold" (PT1M) and "Expiration threshold" (PT14D). An "Update" button is below.

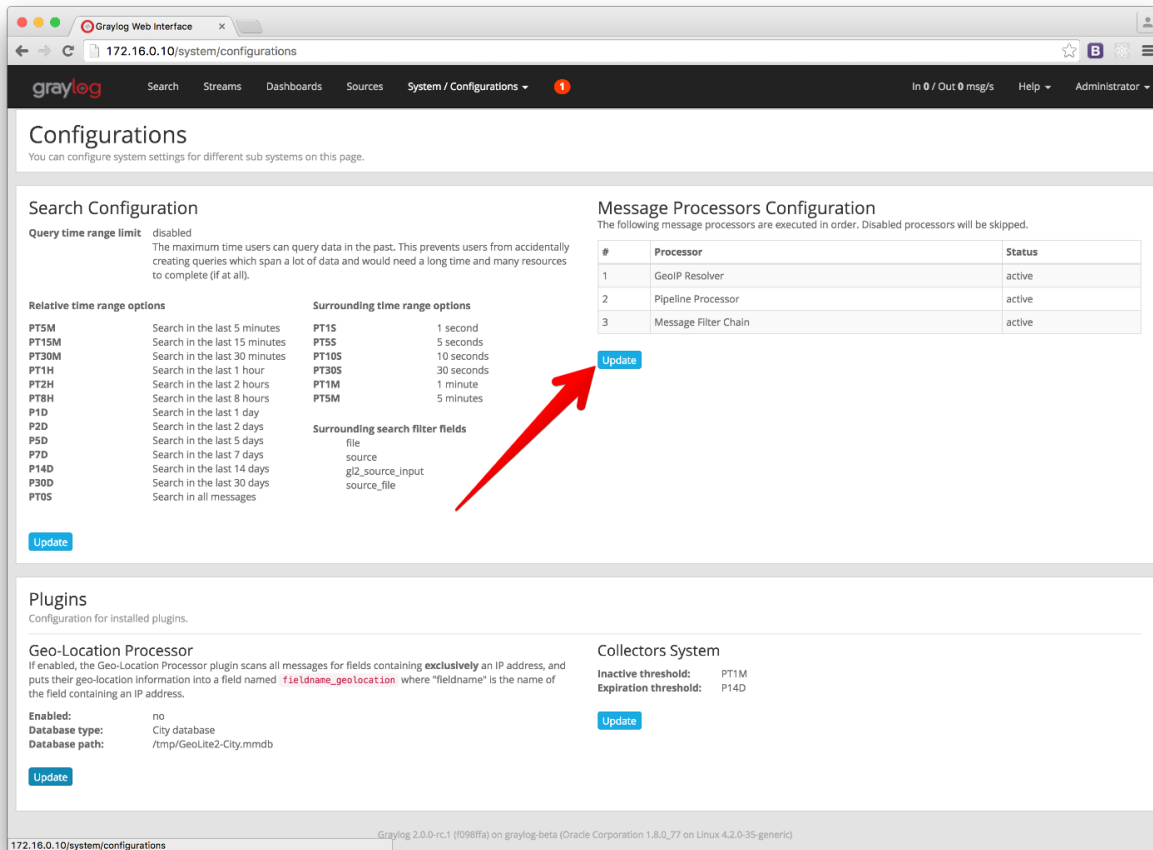
A red arrow points from the "Update" button in the "Geo-Location Processor" section to the "Update" button in the "Message Processors Configuration" section.

In the configuration modal, you need to check the *Enable geolocation processor*, and enter the path to the geolocation database you use. Once you are all set, click on save to store the configuration changes.



Configure the message processor

The last step before being able to resolve locations from IPs in your logs, is to activate the GeoIP Resolver processor. In the same *System -> Configurations* page, update the configuration in the *Message Processors Configuration* section.

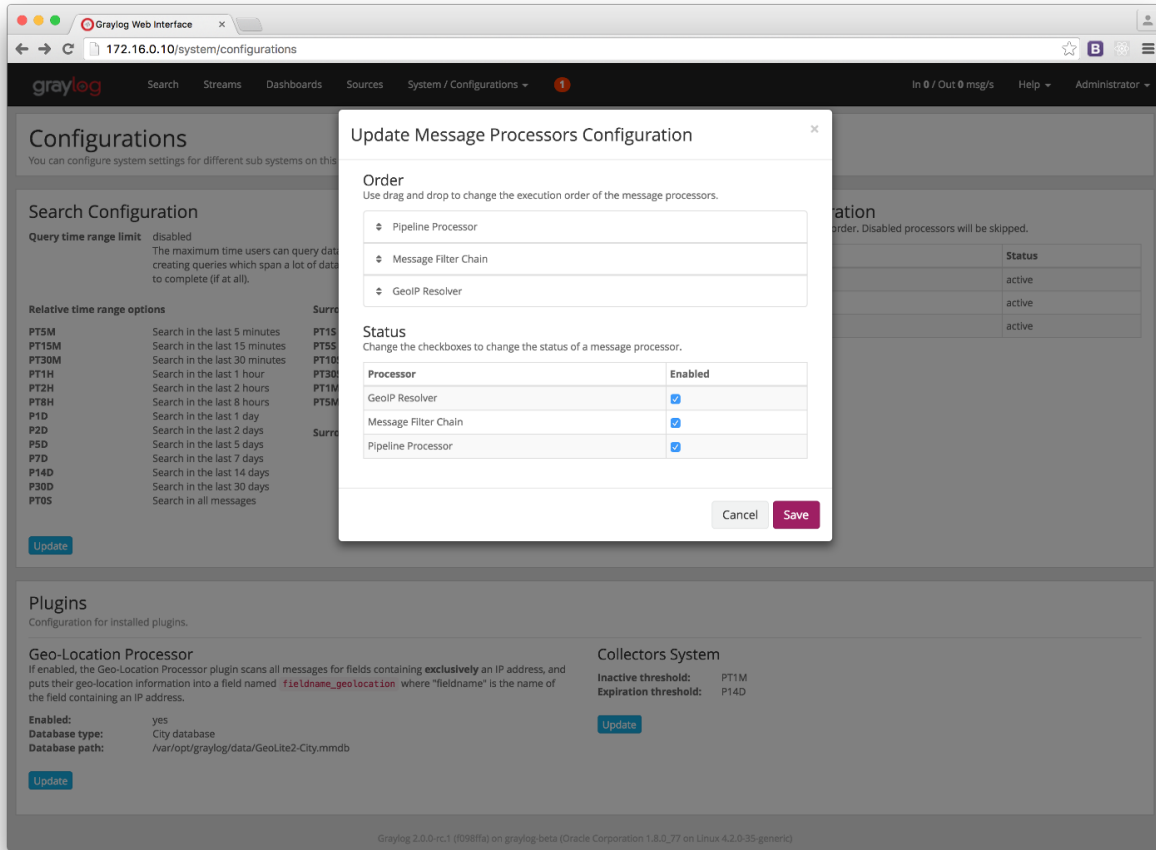


The screenshot shows the Graylog Web Interface at the URL `172.16.0.10/system/configurations`. The page title is "Configurations" with a subtitle "You can configure system settings for different sub systems on this page." The navigation bar includes "Search", "Streams", "Dashboards", "Sources", and "System / Configurations". The main content area is divided into three sections:

- Search Configuration**: Includes "Query time range limit" (disabled) and "Relative time range options" (PT5M, PT15M, PT30M, PT1H, PT2H, PT8H, P1D, P2D, P5D, P7D, P14D, P30D, PTOS). It also has "Surrounding time range options" (PT1S, PT5S, PT10S, PT30S, PT1M, PT5M) and "Surrounding search filter fields" (file, source, gl2_source_input, source_file). An "Update" button is at the bottom.
- Message Processors Configuration**: Includes a table of message processors and an "Update" button. A red arrow points to this button.
- Plugins**: Includes "Geo-Location Processor" (Enabled: no, Database type: City database, Database path: /tmp/GeoLite2-City.mmdb) and "Collectors System" (Inactive threshold: PT1M, Expiration threshold: P14D). An "Update" button is at the bottom.

The footer shows "Graylog 2.0.0-rc.1 (f098ffa) on graylog-beta (Oracle Corporation 1.8.0_77 on Linux 4.2.0-35-generic)".

In that screen, you need to **enable the GeoIP Resolver**, and you must also **set the GeoIP Resolver as the last message processor to run**, if you want to be able to resolve geolocation from fields coming from extractors.



That's it, at this point Graylog will start looking for fields **containing exclusively** an IPv4 or IPv6 address, and extracting their geolocation into a `<field>_geolocation` field.

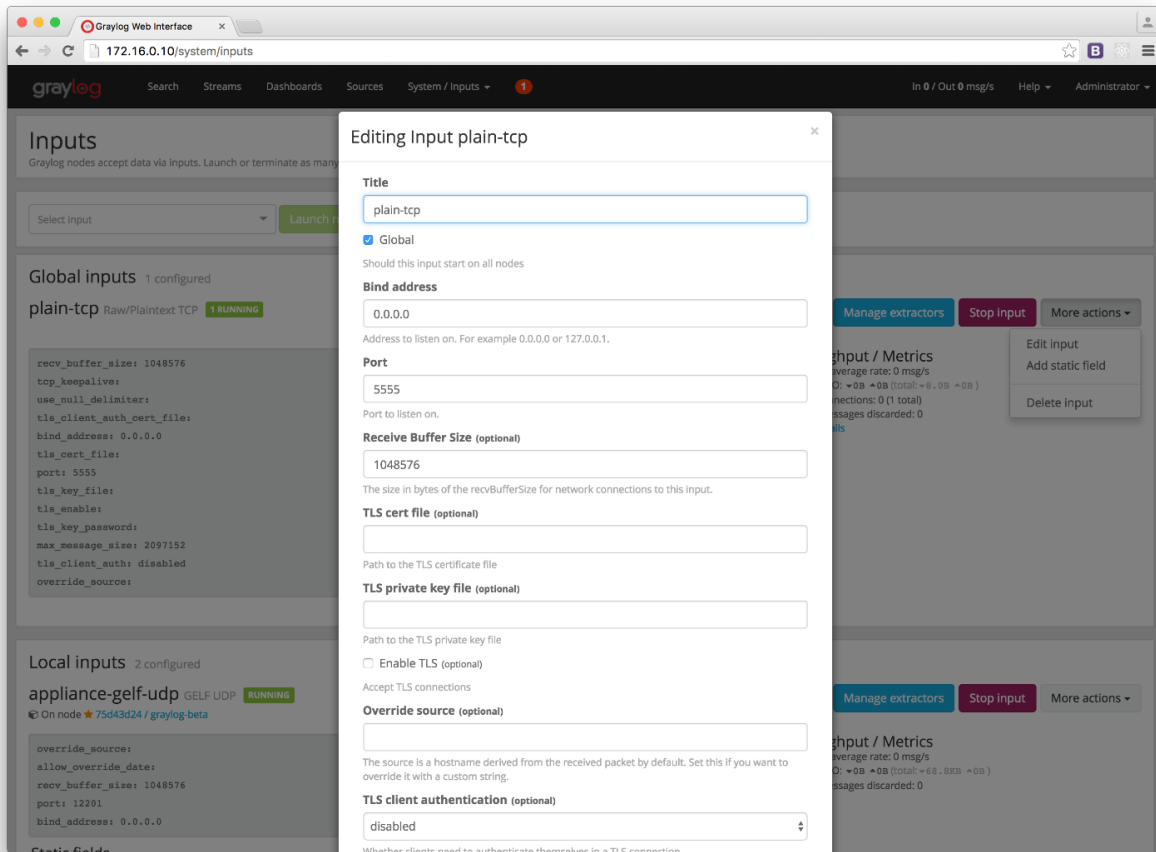
Note: In case you are not sending structured logs to Graylog, you can use extractors to store the IP addresses in your messages into their own fields. Check out the [Extractors](#) documentation for more information.

Important: The GeoIP Resolver processor will **not process** any internal message fields, i. e. any field starting with `gl2_` such as `gl2_remote_ip`.

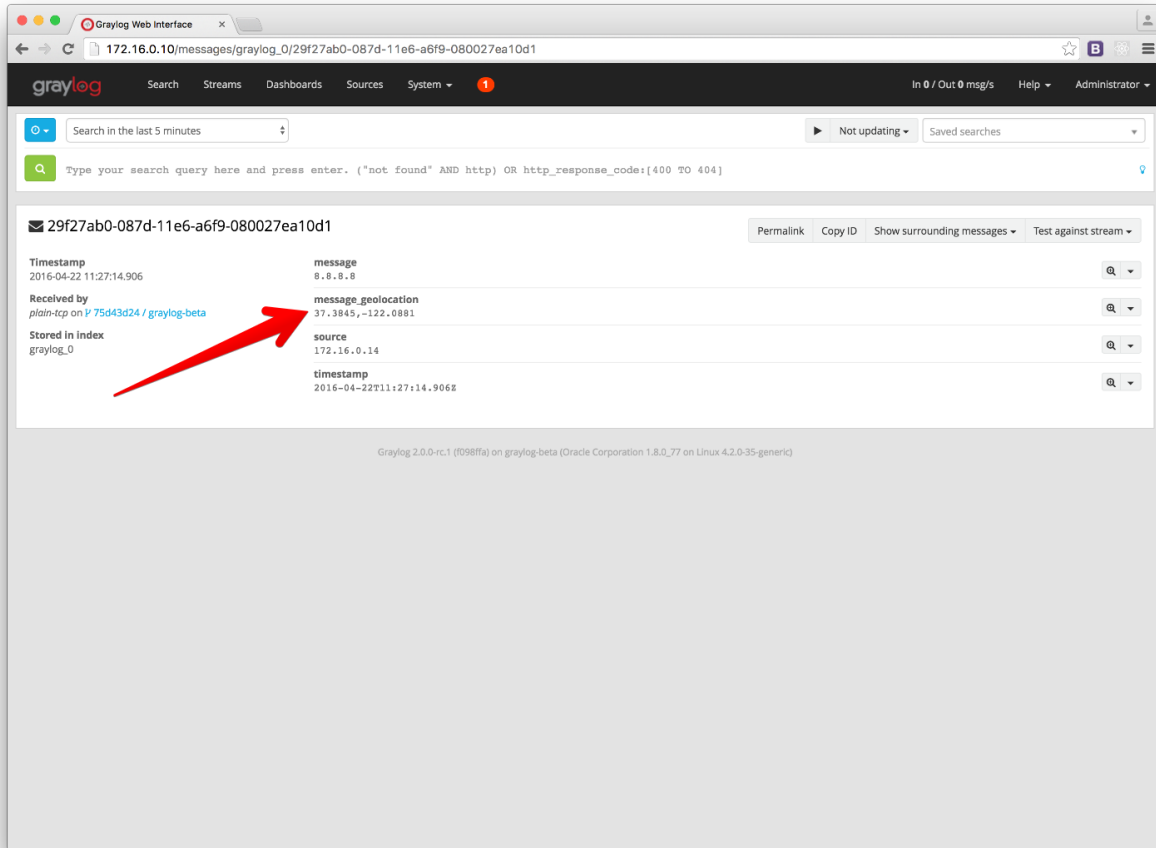
Verify the geolocation configuration (Optional)

To ensure the geolocation resolution is working as expected, you can do the following:

1. Create a TCP Raw/Plaintext input:



2. Send a message only containing an IP to the newly created input. As an example, we will be using the `nc` command:
`nc -w0 <graylog_host> 5555 <<< '8.8.8.8'`
3. Verify that the message contains a `message_geolocation` field:



4. Delete the input if you don't need it any more

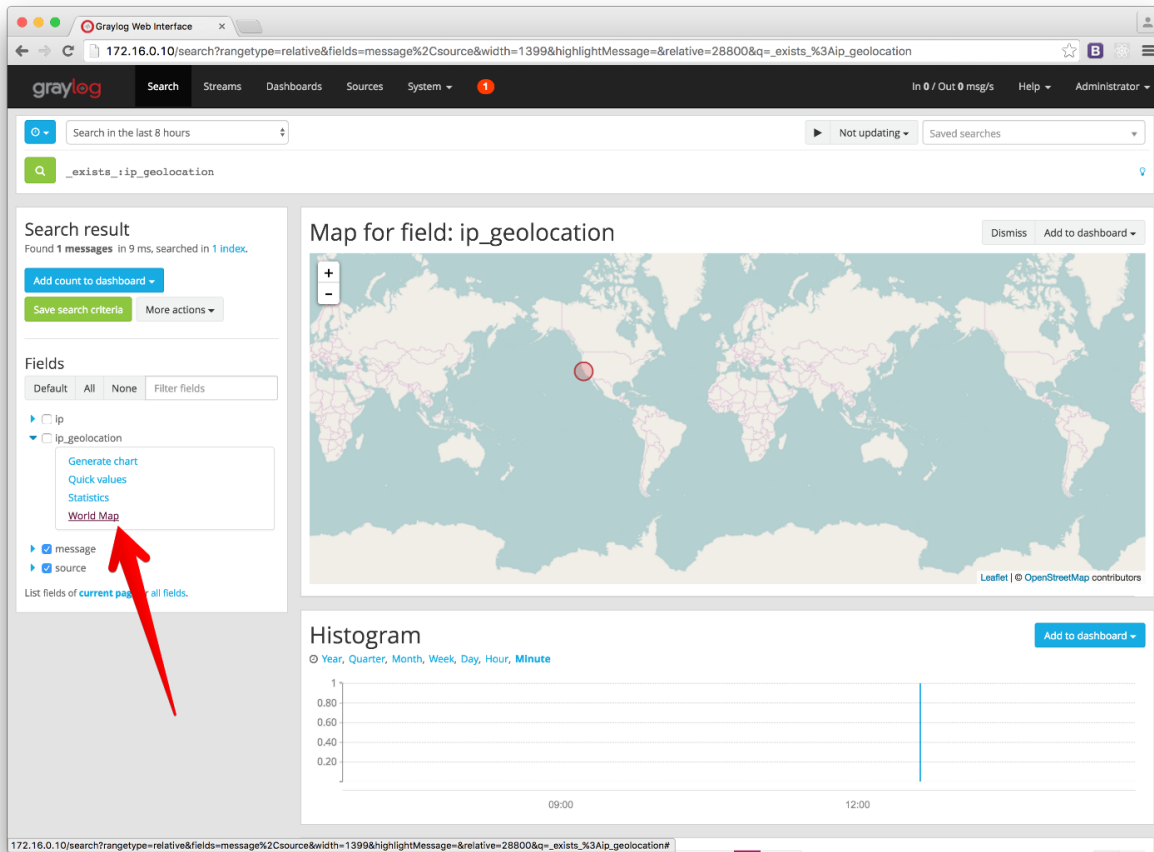
In case the message does not contain a `message_geolocation` field, please check your Graylog server logs, and ensure you followed the steps in the [Configure the database](#) section.

Visualize geolocations in a map

Graylog can display maps from geolocation stored in any field, as long as the geo-points are using the latitude, longitude format.

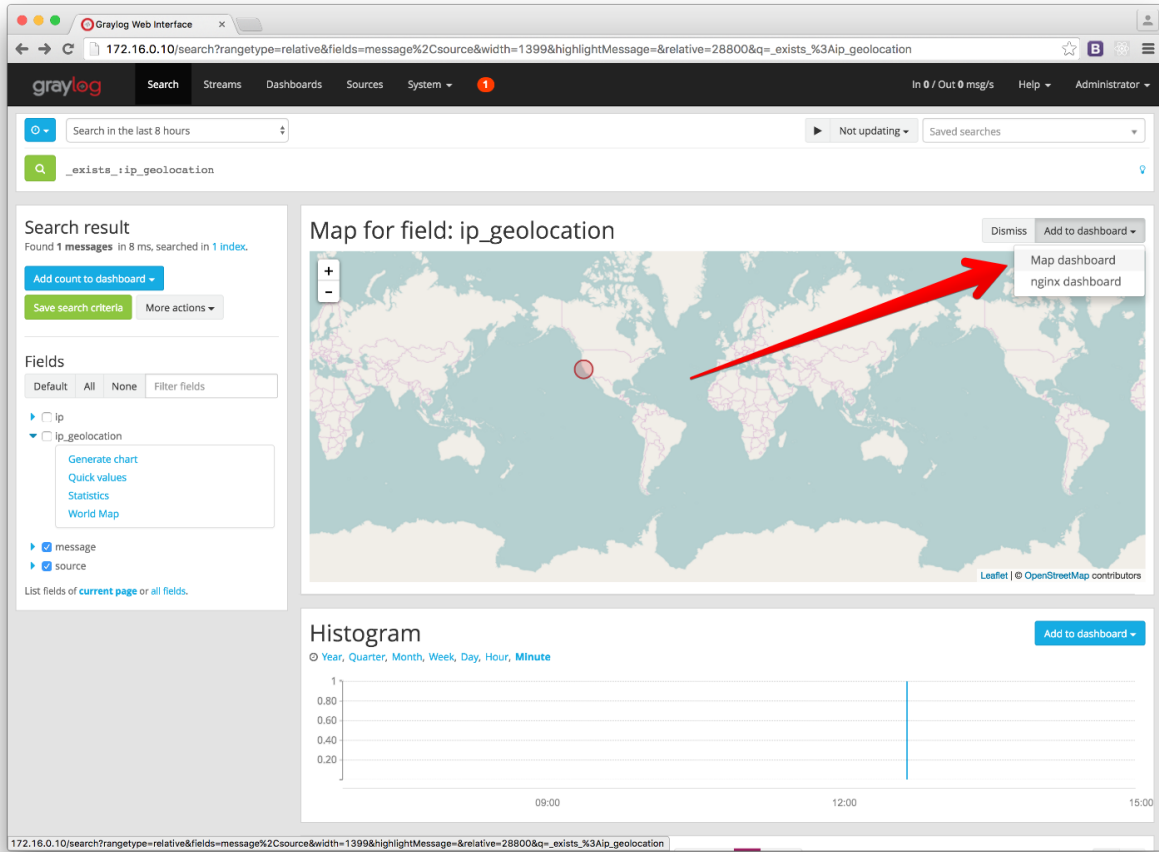
Display a map in the search results page

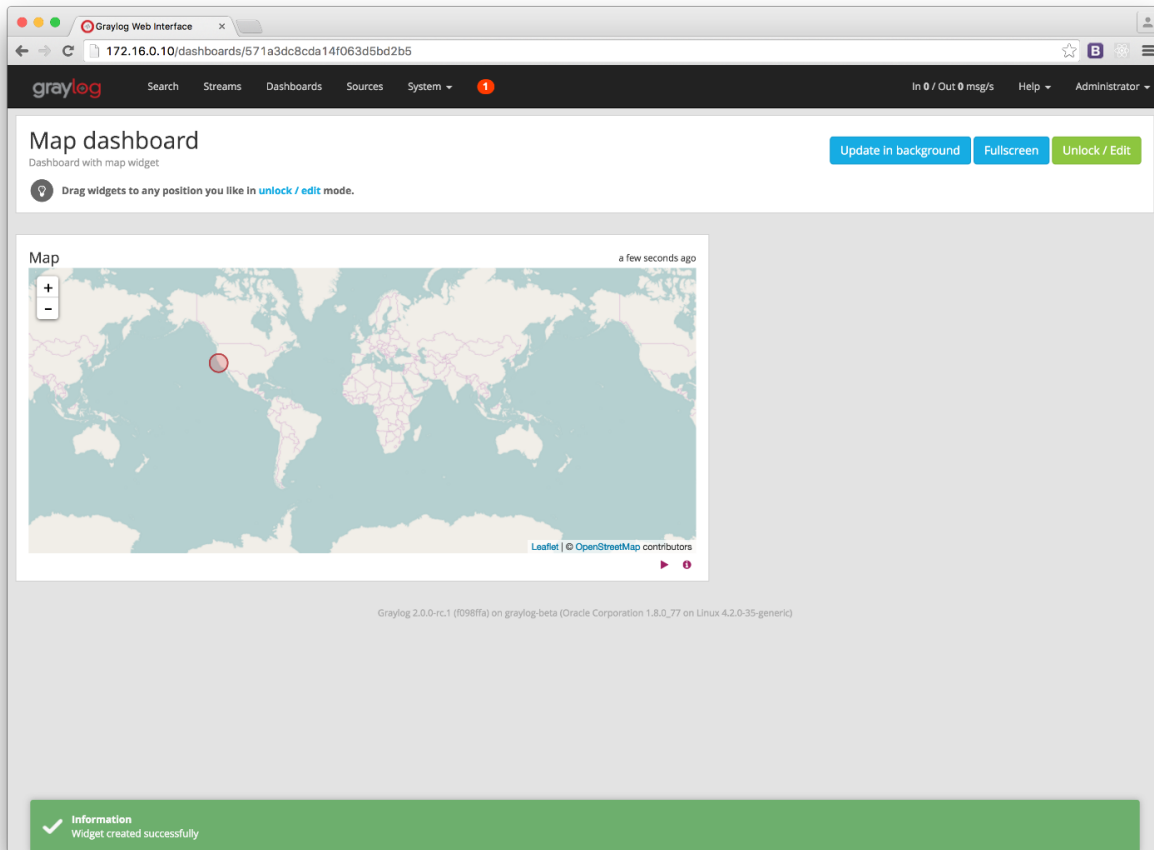
On any search result page, you can expand the field you want to use to draw a map in the search sidebar, and click on the *World Map* link. That will show a map with all different points stored in that field.



Add map to a dashboard

You can add the map visualization into any dashboards as you do with other widgets. Once you displayed a map in the search result page, click on *Add to dashboard*, and select the dashboard where you want to add the map.





FAQs

Will Graylog extract IPs from all fields?

Yes, as long as they contain exclusively an IP address.

Where are the extracted geolocations stored?

Extracted geolocations are stored in a new field, named as the original field, with `_geolocation` appended to it. That is, if the original field was called `ip_address`, the extracted geolocation will be stored in the `ip_address_geolocation` field.

Which geo-points format does Graylog use to store geolocation information?

Graylog stores the geolocation information in the `latitude, longitude` format.

I have a field in my messages with geolocation information already, can I use it in Graylog?

Yes, as long as it contains geolocation information in the `latitude, longitude` format.

Not all fields containing IP addresses are resolved. Why does this happen?

Most likely it is a misconfiguration issue. Please ensure that **the IPs you want to get geolocation information from are in their own fields**, and also ensure that **the GeoIP Resolver is enabled, and in the right order** in the *Message Processors Configuration*, as explained in *Configure the message processor*.

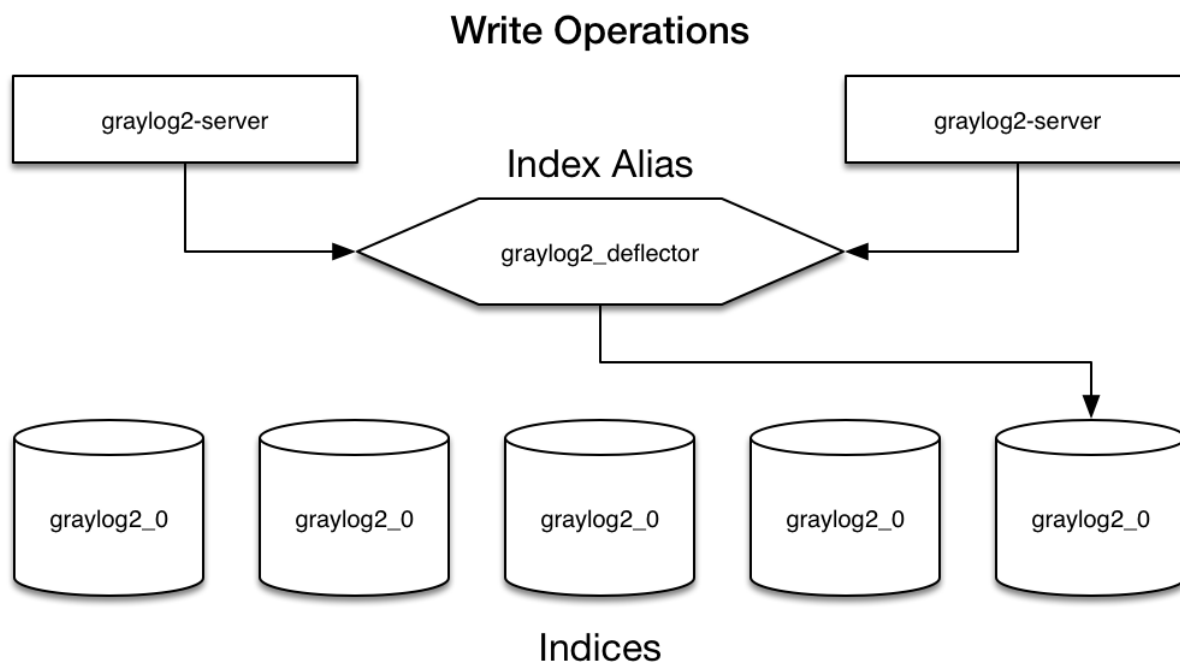
The Graylog index model explained

Overview

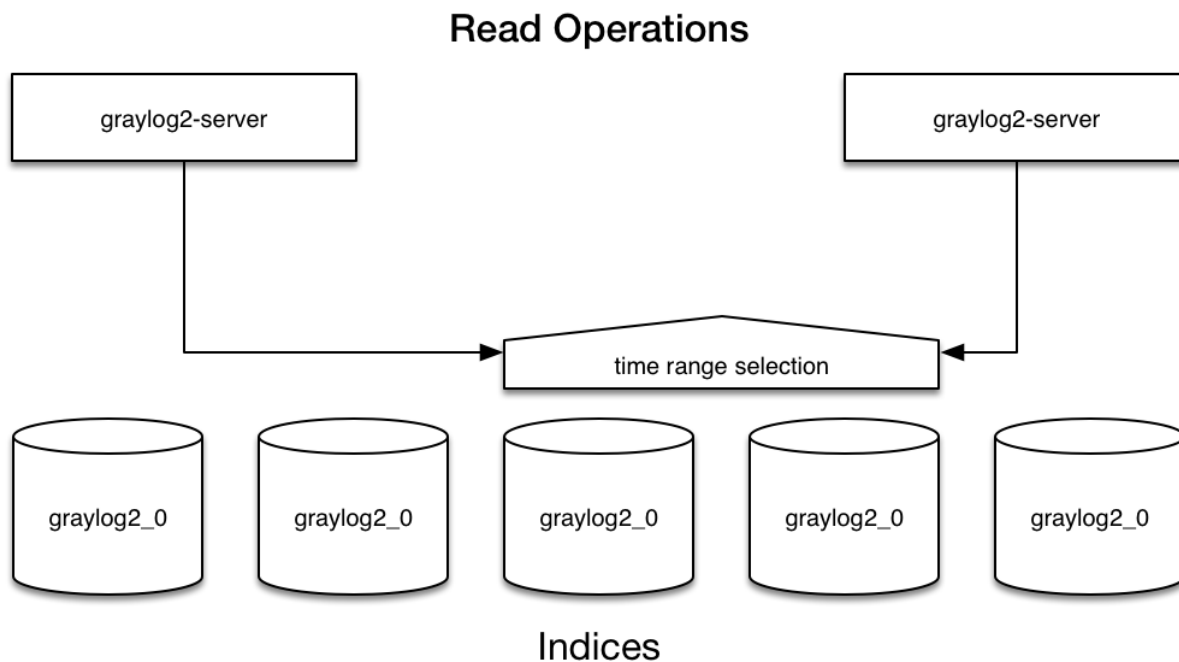
Graylog is transparently managing a set of indices to optimize search and analysis operations for speed and low resource utilisation.

The system is maintaining an **index alias** called `graylog_deflector` which is always pointing to the current write-active index. There is always exactly one index to which new messages are written until the configured rotation criterion (number of documents, index size, or index age) has been reached.

A background task checks regularly if the rotation criterion has been reached and 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` index alias 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 write-active index is.



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

There's a configuration setting for the maximum number of indices Graylog is managing. Depending on the configured retention strategy, the oldest indices will automatically be closed, deleted, or exported when the maximum number of indices has been reached. The deletion is performed by the Graylog master node in a background thread that is continuously comparing the number of indices with the configured maximum.

The following index rotation settings are available:

- **Message count:** Rotates the index after a specific number of messages have been written.
- **Index size:** Rotates the index after an approximate size (before optimization) has been reached.
- **Index time:** Rotates the index after a specific time (e. g. 1 hour or 1 week).

Update Index Settings

Index Rotation Configuration

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

Index Time

Rotation period (ISO8601 Duration)

PT1H

an hour

How long an index gets written to before it is rotated. (i.e. "P1D" for 1 day, "PT6H" for 6 hours)

Index Retention Configuration

Graylog uses a retention strategy to clean up old indices.

Close Index

Max number of indices

20

Maximum number of indices to keep before **closing** the oldest ones

Cancel

Save

Index:

8,120 ops (took a few seconds)

The following index retention settings are available:

- **Delete:** [Delete indices](#) in Elasticsearch to minimize resource consumption.
- **Close:** [Close indices](#) in Elasticsearch to reduce resource consumption.
- **Do nothing**
- **Archive:** Commercial feature, see [Archiving](#).

Keeping the metadata in synchronisation

Graylog will 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 or delete messages from already “closed” indices. 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>
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Re-calculating index ranges.
INFO : org.graylog2.indexer.ranges.RebuildIndexRangesJob - Calculated range of [graylog2_56] in [640m]
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
INFO : org.graylog2.system.jobs.SystemJobManager - SystemJob <ef7057c0-5ae3-11e3-b935-4c8d79f2b596>
```

Manually cycling the deflector

Sometimes you might want to cycle the deflector manually and not wait until the configured rotation criterion for the latest index has been reached. You can do this either via an HTTP request against the REST API of the Graylog master node or via the web interface:

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

The screenshot shows the Graylog web interface. The top navigation bar includes the Graylog logo and links for Search, Streams, Dashboards, Sources, and System / Indices. The main content area is titled 'Indices' and includes a description: 'This is an overview of all indices (message stores) Graylog is currently taking in account for searches and analysis.' Below this, there's a light blue box with a lightbulb icon and the text 'You can learn more about the index model in the documentation'. To the right, a 'Maintenance' dropdown menu is open, showing options: 'Recalculate index ranges' and 'Manually cycle deflector'. Below the maintenance menu, there's a 'Settings' section with a table of configuration options:

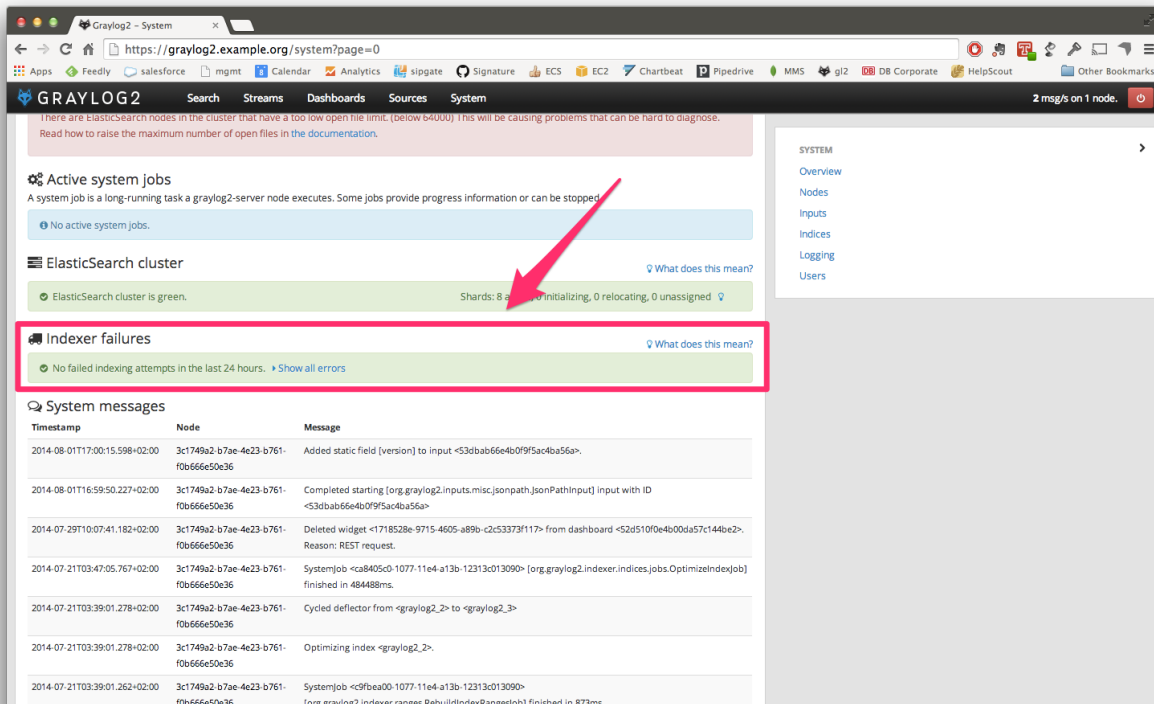
Index rotation strategy:	Index Time	Index retention strategy:	Archive
Rotation period:	P1D (1d, a day)	Max number of indices:	5
		Index action:	DELETE

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>
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>
```

Indexer failures

Every Graylog node is constantly keeping track about every indexing operation it performs. This is important for making sure that you are not silently losing any messages. The web interface can show you a number of write operations that failed and also a list of failed operations. Like any other information in the web interface this is also available via the REST APIs so you can hook it into your own monitoring systems.



Information about the indexing failure is stored in a capped MongoDB collection that is limited in size. A lot (many tens of thousands) of failure messages should fit in there but it should not be considered a complete collection of all errors ever thrown.

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: "s"]
```

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.

Users and Roles

Graylog has a granular permission system which secures the access to its features. Each interaction which can look at data or change configuration in Graylog must be performed as an authenticated user.

Each user can have varying levels of access to Graylog's features, which can be controlled with assigning roles to users.

The following sections describe the capabilities of users and roles **and also how to use LDAP for authentication**.

Users

It is recommended to create an account for each individual user accessing Graylog.

User accounts have the usual properties such as a login name, email address, full name, password etc. In addition to these fields, you can also configure the session timeout, roles and timezone.

The screenshot shows the 'Create new user' page in the Graylog web interface. The page has a dark header with the Graylog logo and navigation links: Search, Streams, Dashboards, Sources, and System / Users. The top right shows 'In 36 / Out 36 msg/s' and 'Administrator'. Below the header is a breadcrumb trail: / System / Users / New. The main heading is 'Create new user', followed by a subtext: 'Use this page to create new Graylog users. The users and their permissions created here are not limited to the web interface but valid and required for the REST APIs of your Graylog server nodes, too.'

The form contains the following fields and options:

- Username:** A text input field with a note: 'Select a unique user name used to log in with.'
- Full Name:** A text input field with a note: 'Give a descriptive name for this account, e.g. the full name.'
- Email Address:** A text input field with a note: 'Give the contact email address.'
- Password:** Two text input fields labeled 'Password' and 'Repeat password'. A note below states: 'Passwords must be at least 6 characters long. We recommend using a strong password.'
- Roles:** A dropdown menu showing 'Reader' with a close icon. A note below explains: 'Assign the relevant roles to this user to grant them access to the relevant streams and dashboards. The Reader role grants basic access to the system and will be enabled. The Admin role grants access to everything in Graylog.'
- Sessions do not time out:** A checkbox with a note: 'When checked sessions never time out due to inactivity.'
- Timeout:** A text input field and a dropdown menu set to 'Seconds'. A note below states: 'Session automatically end after this amount of time, unless they are actively used.'
- Time Zone:** A dropdown menu with the text 'Pick your time zone'. A note below states: 'Choose your local time zone or leave it as it is to use the system's default.'

A green 'Create User' button is located at the bottom of the form.

Sessions

Each login for a user creates a session, which is bound to the browser the user is currently using. Whenever the user interacts with Graylog this session is extended.

For security reasons you will want to have Graylog expire sessions after a certain period of inactivity. Once the interval specified by `timeout` expires the user will be logged out of the system. Requests like displaying throughput statistics do not extend the session, which means that if the user keeps Graylog open in a browser tab, but does not interact with it, their session will expire as if the browser was closed.

Logging out explicitly terminates the session.

Timezone

Since Graylog internally processes and stores messages in the UTC timezone, it is important to set the correct timezone for each user.

Even though the system defaults are often enough to display correct times, in case your team is spread across different timezones, each user can be assigned and change their respective timezone setting. You can find the current timezone

settings for the various components on the *System -> Overview* page of your Graylog web interface.

Initial Roles

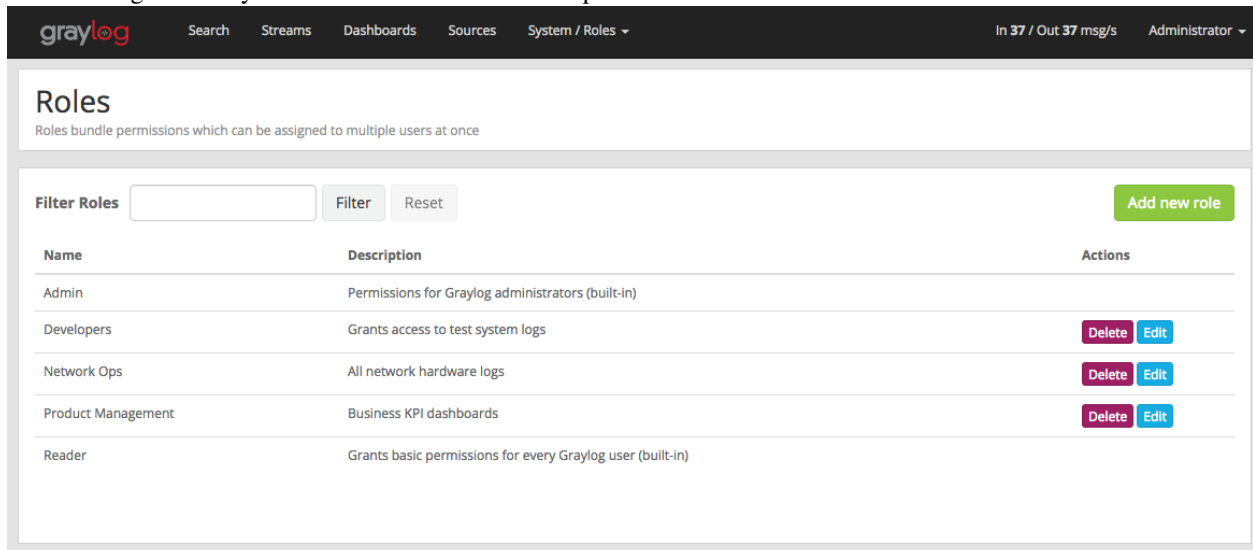
Each user needs to be assigned at least one role, which governs the basic set of permissions this user has in Graylog.

Normal users, which do not need to create inputs, outputs or perform administrative tasks like managing access control etc, should be assigned the built in `Reader` role in addition to the custom roles which grant access to streams and dashboards.

Roles

In Graylog, roles are named collections of individual permissions which can be assigned to users. Previous Graylog versions could only assign individual permissions to each user in the system, making updating stream or dashboard permissions for a large group of users difficult to deal with.

Starting in Graylog 1.2 you can create roles which bundle permissions to streams and dashboards. These roles can then be assigned to any number of users and later be updated to include new streams and dashboards.



The screenshot shows the Graylog web interface for managing roles. The top navigation bar includes the Graylog logo, links for Search, Streams, Dashboards, Sources, and System / Roles (with a dropdown arrow). On the right, it shows 'In 37 / Out 37 msg/s' and the user 'Administrator' with a dropdown arrow. The main content area is titled 'Roles' with a subtitle 'Roles bundle permissions which can be assigned to multiple users at once'. Below this is a 'Filter Roles' section with an input field, 'Filter', and 'Reset' buttons. A green 'Add new role' button is in the top right. A table lists the roles:

Name	Description	Actions
Admin	Permissions for Graylog administrators (built-in)	
Developers	Grants access to test system logs	Delete Edit
Network Ops	All network hardware logs	Delete Edit
Product Management	Business KPI dashboards	Delete Edit
Reader	Grants basic permissions for every Graylog user (built-in)	

The two roles `Admin` and `Reader` are built in and cannot be changed. The `Admin` role grants all permissions and should only be assigned to users operating Graylog. The `Reader` role grants the basic permissions every user needs to be able to use Graylog. The interface will ensure that every user at least has the `Reader` role in addition to more business specific roles you create.

Roles cannot be deleted as long as users are still assigned to them, to prevent accidentally locking users out.

Create a role

In order to create a new role, choose the green *Add new role* button on the *System -> Roles* page.

This will display a dialog allowing you to describe the new role and select the permissions it grants.

graylog

SearchStreamsDashboardsSourcesSystem / Roles ▾

In 37 / Out 37 msg/sAdministrator ▾

Roles

Roles bundle permissions which can be assigned to multiple users at once

Create a new role

Name

Description

Permissions

Select the permissions for this role

Streams

Dashboards

Filter Streams

<input type="checkbox"/> Select all	
<input type="checkbox"/> nginx requests All requests that were logged into the nginx_access_log	<input type="button" value="Allow reading"/> <input type="button" value="Allow editing"/>
<input type="checkbox"/> nginx HTTP 4XXs All requests that were answered with a HTTP code in the 400 range by nginx	<input type="button" value="Allow reading"/> <input type="button" value="Allow editing"/>
<input type="checkbox"/> nginx HTTP 5XXs All requests that were answered with a HTTP code in the 500 range by nginx	<input type="button" value="Allow reading"/> <input type="button" value="Allow editing"/>
<input type="checkbox"/> nginx errors All requests that were logged into the nginx_error_log	<input type="button" value="Allow reading"/> <input type="button" value="Allow editing"/>

Please name the role and select at least one permission to save it.

After naming the role, select the permissions you want to grant using the buttons to the right of the respective stream or dashboard names. For each stream or dashboard you can select whether to grant `edit` or `read` permissions, but note that edit permissions always imply read permissions as well.

In case you have many streams or dashboards you can use the filter to narrow the list down, and use the checkboxes on the left hand side of the table to select multiple items. You can then use the bulk action buttons on the right hand side to toggle the permissions for all of the selected items at once.

Roles
Roles bundle permissions which can be assigned to multiple users at once

Create a new role

Name
Developer

Description
Grants access to error logs for debugging

Permissions
Select the permissions for this role

Streams Dashboards

Filter Streams Filter Reset

<input type="checkbox"/> Select all	Toggle read permissions Toggle edit permissions
<input type="checkbox"/> nginx requests All requests that were logged into the nginx access_log	Allow reading Allow editing
<input checked="" type="checkbox"/> nginx HTTP 4XXs All requests that were answered with a HTTP code in the 400 range by nginx	Allow reading Allow editing
<input checked="" type="checkbox"/> nginx HTTP 5XXs All requests that were answered with a HTTP code in the 500 range by nginx	Allow reading Allow editing
<input checked="" type="checkbox"/> nginx errors All requests that were logged into the nginx error_log	Allow reading Allow editing

Please name the role and select at least one permission to save it.

Save Cancel

Once you are done, be sure to save your changes. The save button is disabled until you select at least one permission.

Editing a role

Administrators can edit roles to add or remove access to new streams and dashboards in the system. The two built in `Admin` and `Reader` roles cannot be edited or deleted because they are vital for Graylog's permission system.

Simply choose the *edit* button on the *System -> Roles* page and change the settings of the role in the following page:

Roles
Roles bundle permissions which can be assigned to multiple users at once

Edit role Developers

Name
Developers

Description
Grants access to test system logs

Permissions
Select the permissions for this role

Streams Dashboards

Filter Streams Filter Reset

☐ Select all

- ☐ nginx requests
All requests that were logged into the nginx access_log
- ☐ nginx HTTP 4XXs
All requests that were answered with a HTTP code in the 400 range by nginx
- ☐ nginx HTTP 5XXs
All requests that were answered with a HTTP code in the 500 range by nginx
- ☐ nginx errors
All requests that were logged into the nginx error_log

Allow reading Allow editing

Allow reading Allow editing

Allow reading Allow editing

Allow reading Allow editing

Save Cancel

You can safely rename the role as well as updating its description, the existing role assignment for users will be kept.

Deleting a role

Deleting roles checks whether a role still has users assigned to it, to avoid accidentally locking users out. If you want to remove a role, please remove it from all users first.

System users

The Graylog permission system is extremely flexible and allows you to create users that are only allowed to perform certain REST calls. The [Roles](#) UI allows you to create roles based on stream or dashboard access but does not expose permissions on a REST call level yet. This guide describes how to create those roles using the Graylog REST API.

Let's imagine we want to create a role that is only allowed to start or stop message processing on `graylog-server` nodes.

REST call permissions

Almost every REST call in Graylog has to be authenticated or it will return a HTTP 403 (Forbidden). In addition to that the requesting user also has to have the permissions to execute the REST call. A Graylog admin user can always

execute all calls and roles based on the standard stream or dashboard permissions can execute calls related to those entities.

If you want to create a user that can only execute calls to start or stop message processing you have to find the name of the required permission first. Until we include this functionality in the Graylog UI you'll have to look directly into the code. The permissions are listed in the `RestPermissions` class which you can find on [GitHub](#). (Make sure to select a branch that reflects your current `graylog-server` version.)

The permission you are searching for in this case is:

```
public static final String PROCESSING_CHANGE_STATE = "processing:changestate";
```

Creating the role

You can create a new role using the REST API like this:

```
curl -v -XPOST -H 'Content-Type: application/json' 'http://ADMIN:PASSWORD@graylog.example.org:12900/roles'
```

Notice the `processing:changestate` permission that we assigned. Every user with this role will be able to execute only calls that start or stop processing on `graylog-server` nodes. (and also the standard reader permissions that do not provide any access to data or maintenance functionalities though.)

This is the POST body in an easier to read formatting:

```
{
  "name": "Change processing state",
  "description": "Permission to start or stop processing on graylog-server nodes",
  "permissions": [
    "processing:changestate"
  ],
  "read_only": false
}
```

Assigning the role to a user

Create a new user in the Graylog Web Interface and assign the new role to it:

Every user needs to at least have the standard `reader` permissions but those do not provide any access to data or maintenance functionalities.

Now request the user information to see what permissions have been assigned:

```
$ curl -XGET 'http://ADMIN:PASSWORD@graylog.example.org:12900/users/maintenanceuser?pretty=true'
{
  "id" : "563d1024d4c63709999c4ac2",
  "username" : "maintenanceuser",
  "email" : "it-ops@example.org",
  "full_name" : "Rock Solid",
  "permissions" : [
    "indexercluster:read",
    "messagecount:read",
    "journal:read",
    "inputs:read",
    "metrics:read",
    "processing:changestate",
    "savedsearches:edit",
    "fieldnames:read",
    "buffers:read",
    "system:read",
    "users:edit:maintenanceuser",
    "users:passwordchange:maintenanceuser",
    "savedsearches:create",
    "jvmstats:read",
    "throughput:read",
    "savedsearches:read",
    "messages:read"
  ],
  "preferences" : {
```



```
    "updateUnfocussed" : false,
    "enableSmartSearch" : true
  },
  "timezone" : "America/Chicago",
  "session_timeout_ms" : 300000,
  "read_only" : false,
  "external" : false,
  "startpage" : { },
  "roles" : [
    "Change processing state",
    "Reader"
  ]
}
```

Now you can use this user in your maintenance scripts or automated tasks.

External authentication

LDAP / Active Directory

It is possible to use an external LDAP or Active Directory server to perform user authentication in Graylog. Since version 1.2, you can also use LDAP groups to perform authorization by mapping them to Graylog roles.

Configuration

To set up your LDAP or Active Directory server, go to *System -> Users -> Configure LDAP*. Once LDAP is enabled, you need to provide some details about the server. Please test the server connection before continuing to the next steps.

User mapping

In order to be able to look for users in the LDAP server you configured, Graylog needs to know some more details about it: the base tree to limit user search queries, the pattern used to look for users, and the field containing the full name of the user. You can test the configuration any time by using the login test form that you can find at the bottom of that page.

Login Test

Loads the LDAP entry for the given user name. If you omit the password, no authentication attempt will be made.

✓ User check ✓ Login check

LDAP attributes of the user

uid
foobar
uidnumber
1000
mail
foobar@graylog.test
homedirectory
/home/foo
givenname
Foo
gidnumber
501
sn
Bar
cn
Foo Bar
objectclass
posixAccount

LDAP Groups of the user

The login test information will indicate if Graylog was able to load the given user (and perform authentication, if a password was provided), and it will display all LDAP attributes belonging to the user, as you can see in the screenshot.

That's it for the basic LDAP configuration. Don't forget to save your settings at this point!

Group mapping

You can additionally control the default permissions for users logging in with LDAP or Active Directory by mapping LDAP groups into Graylog roles. That is extremely helpful if you already use LDAP groups to authorize users in your organization, as you can control the default permissions members of LDAP groups will have.

Once you configure group mapping, Graylog will rely on your LDAP groups to assign roles into users. That means that each time an LDAP user logs into Graylog, their roles will be assigned based on the LDAP groups their belong to.

In first place, you need to fill in the details in the *Group Mapping* section under *System -> Users -> Configure LDAP*, by giving the base where to limit group searches, a pattern used to look for groups, and the group name attribute.

Then you need to select which default user role will be assigned to any users authenticated with the LDAP server should have. It is also possible to assign additional roles to any users logging in with LDAP. Please refer to [Roles](#) for more details about user roles.

Note: Graylog only synchronizes with LDAP when users log in. After changing the default and additional roles for LDAP users, you may need to modify existing users manually or delete them in order to force them to log in again.

You can test the group mapping information by using the login test form, as it will display LDAP groups that the test user belongs to. Save the LDAP settings once you are satisfied with the results.

graylog Search Streams Dashboards Sources System / Overview ▾

LDAP Group mapping

Map LDAP groups to Graylog roles

💡 LDAP groups with no defined mapping will use the defaults set in your [LDAP settings](#).

Admins	Admin
Developers	Dashboards
QA	Reader

[Save](#) [Cancel](#)

Finally, in order to map LDAP groups into roles, you need to go to *System -> Users -> LDAP group mapping*. That page will load all available LDAP groups using the configuration you previously provided, and will allow you to select a Graylog role that defines the permissions that group will have inside Graylog.

Note: Loading LDAP groups may take some time in certain configurations, specially if you have many groups. In those cases, creating a better filter for groups may help with the loading times.

Note: Remember that Graylog only synchronizes with LDAP when users log in, so you may need to modify existing users manually after changing the LDAP group mapping.

Troubleshooting

LDAP referrals for groups can be a problem during group mapping. Referral issues are most likely to come up with larger AD setups. The Active Directory servers literally refer to other servers in search results, and it is the client's responsibility to follow all referrals. Support for that is currently not implemented in Graylog.

Referral issues can be detected by warnings in the server logs about group mapping failing, for example:

```
2016-04-11T15:52:06.045Z WARN [LdapConnector] Unable to iterate over user's groups,
unable to perform group mapping. Graylog does not support LDAP referrals at the moment.
Please see http://docs.graylog.org/en/2.0/pages
```

These issues may be resolved by either managing the groups manually, or configuring the LDAP connection to work against the [global catalog](#). The first solution means simply that the LDAP group settings must not be set, and the groups are managed locally. The global catalog solution requires using the 3268/TCP, or 3269/TCP (TLS) port of eligible Active Directory server. The downside is that using the global catalog service consumes slightly more server resources.

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 step for writing a plugin is creating a skeleton that is the same for each type of plugin. The next chapter is explaining how to do this and will then go over to chapters explaining plugin types in detail.

Prerequisites

What you need in your development environment before starting is:

- `git`
- `maven`

There are lots of different ways to get those on your local machine, unfortunately we cannot list all of them, so please refer to your operating system-specific documentation,

Creating a plugin skeleton

The easiest way to get started is to use our [Graylog meta project](#), which will create a complete plugin project infrastructure with all required classes, build definitions, and configurations. Using the meta project allows you to have the [Graylog server project](#) and your own plugins (or 3rd party plugins) in the same project, which means that you can run and debug everything in your favorite IDE or navigate seamlessly in the code base.

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

Use it like this:

```
$ git clone git@github.com:Graylog2/graylog-project.git
```

This will create a checkout of the meta project in your current work dir. Now change to the `graylog-project` directory and execute the step which to download the necessary base modules:

```
$ scripts/bootstrap
```

Now you can bootstrap the plugin you want to write from here, by doing:

```
$ scripts/bootstrap-plugin jira-alarmcallback
```

It will ask you a few questions about the plugin you are planning to build. Let's say you work for a company called ACMECorp and want to build an alarm callback plugin that creates a JIRA ticket for each alarm that is triggered:

```
groupId: com.acmecorp
version: 1.0.0
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 `graylog-plugin-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 `graylog-project` directory as a fully configured Java project, which should include the Graylog server and your plugin as submodules.

Please pay close attention to the [README file](#) of the Graylog meta project and follow any further instructions listed there to set up your IDE properly.

If you want to continue working on the command line, you can do the following to compile the server and your plugin:

```
$ mvn package
```

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 practice 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 logic 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);
}
```

Creating a plugin for the web interface

Sometimes your plugin is not only supposed to work under the hood inside a Graylog server as an input, output, alarm callback, etc. but you also want to contribute previously nonexistent functionality to Graylog's web interface. Since version 2.0 this is now possible. When using the most recent *Graylog meta project* <<https://github.com/Graylog2/graylog-project>> to bootstrap the plugin skeleton, you are already good to go for this. Otherwise please see our chapter about [Creating a plugin skeleton](#).

The Graylog web interface is written in JavaScript, based on [React](#). It is built using [webpack](#), which is bundling all JavaScript code (and other files you use, like stylesheets, fonts, images, even audio or video files if you need them)

into chunks digestable by your browser and [npm](#), which is managing our external (and own) dependencies. During the build process all of this will be bundled and included in the jar file of your plugin.

This might be overwhelming at first if you are not accustomed to JS-development, but fortunately we have set up a lot to make writing plugins easier for you!

Prerequisites

If you use our proposed way for *Creating a plugin skeleton*, and followed the part about the *Prerequisites*, you are already good to go for building a plugin with a web part. **All you need is a running Graylog server on your machine.** Everything else is fetched at build time!

How to start development

Getting up and running with a web development environment is as easy as this:

```
$ git clone https://github.com/Graylog2/graylog-project.git
[...]
$ cd graylog-project
$ scripts/bootstrap
[...]
$ scripts/bootstrap-plugin your-plugin
[...]
$ scripts/start-web-dev
[...]
$ open http://localhost:8080
```

This clones the meta project repository, bootstraps the required modules and starts the web server. It even tries to open a browser window going to it (probably working on Mac OS X only).

If your Graylog server is not running on `http://localhost:12900`, then you need to edit `graylog2-server/graylog2-web-interface/config.js` (in your `graylog-project` directory) and adapt the `gl2ServerUrl` parameter.

Web Plugin structure

These are the relevant files and directories in your plugin directory for the web part of it:

webpack.config.js This is the configuration file for the [webpack](#) module bundler. Most of it is already preconfigured by our `PluginWebpackConfig` class, so the file is very small. You can override/extend every configuration option by passing a webpack snippet though.

build.config.js.sample In this file you can customize some of the parameters of the build. There is one mandatory parameter named `web_src_path` which defines the absolute or relative location to a checkout of the [Graylog source repository](#).

package.json This is a standard [npm](#) JSON file describing the web part of your plugin, especially its dependencies. You can read more about its format [here](#).

src/web This is where the actual code for the web part of your plugin goes to. For the start there is a simple `index.jsx` file, which shows you how to register your plugin and the parts it provides with the Graylog web interface. We will get to this in detail later.

Best practices for web plugin development

Using ESLint

ESLint is an awesome tool for linting JavaScript code. It makes sure that any written code is in line with general best practises and the project-specific coding style/guideline. We at Graylog are striving to make the best use of this tools as possible, to help our developers and you to generate top quality code with little bugs. Therefore we highly recommend to enable it for a Graylog plugin you are writing.

Code Splitting

Both the web interface and plugins for it depend on a number of libraries like React, RefluxJS and others. To prevent those getting bundled into *both* the web interface *and* plugin assets, therefore wasting space or causing problems (especially React does not like to be present more than once), we extract those into a commons chunk which is reused by the web interface and plugins.

This has no consequences for you as a plugin author, because the configuration to make use of this is already generated for you when using the meta project or the maven archetype. But here are some details about it:

Common libraries are built into a separate `vendor` bundle using an own configuration file named `webpack.vendor.js`. Using the `DLLPlugin` a `manifest` is extracted which allow us to reuse the generated bundle. This is then imported in our main web interface `webpack` configuration file and the corresponding generated `webpack config file` for plugins.

Building plugins

Building the plugin is easy because the meta project has created all necessary files and settings for you. Just run `mvn package` either from the meta project's directory (to build the server *and* the plugin) or from the plugin directory (to build the plugin only):

```
$ mvn package
```

This will generate a `.jar` file in `target/` that is the complete plugin file:

```
$ ls target/jira-alarmcallback-1.0.0-SNAPSHOT.jar
target/jira-alarmcallback-1.0.0-SNAPSHOT.jar
```

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.

This is a list of default plugin locations for the different installation methods.

Table 21.1: Plugin Installation Locations

Installation Method	Directory
<i>Virtual Machine Appliances</i>	<code>/opt/graylog/plugins/</code>
<i>Operating System Packages</i>	<code>/usr/share/graylog-server/plugin/</code>
<i>Manual Setup</i>	<code>/<extracted-graylog-tarball-path>/plugin/</code>

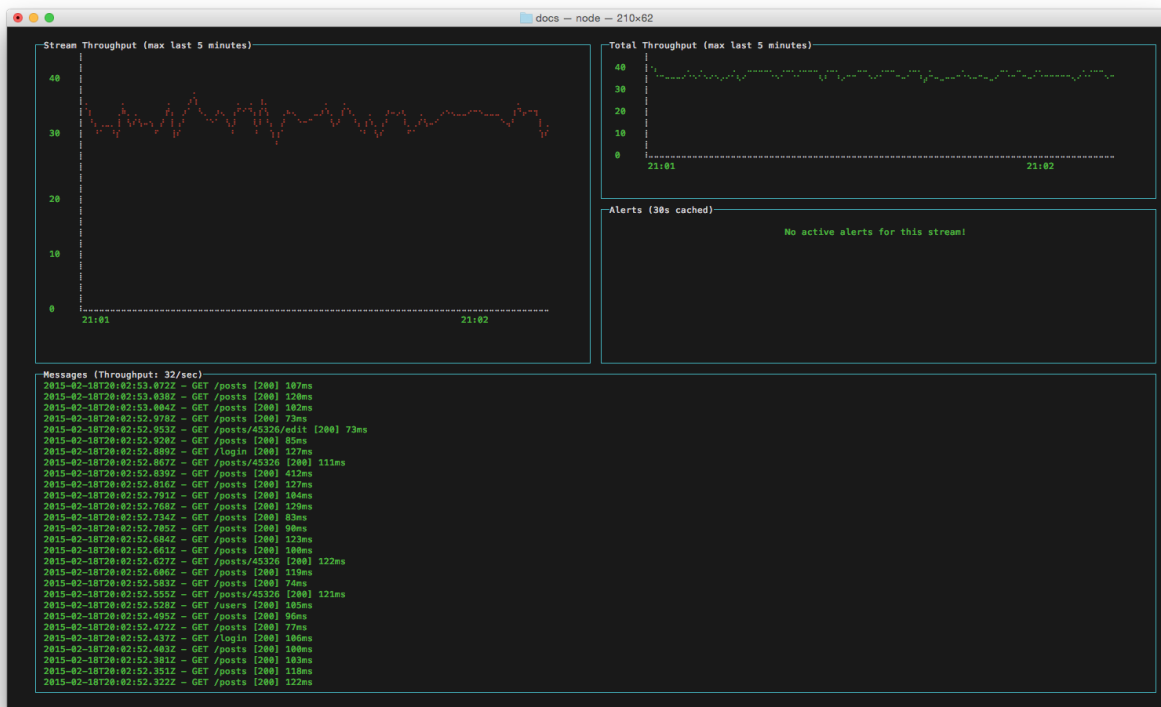
Restart `graylog-server` and the plugin should be available to use from the web interface immediately.

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

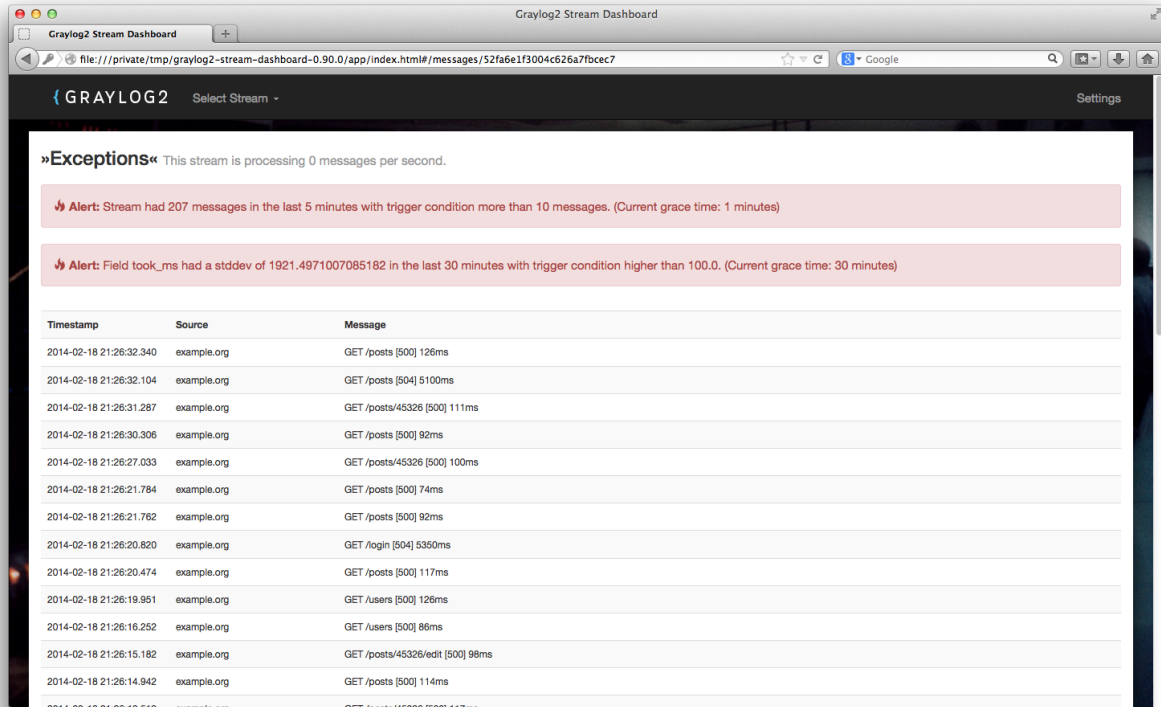
This official Graylog dashboard which is developed by us is showing live information of a specific stream in your terminal. For example it is the perfect companion during a deployment of your platform: Run it next to the deployment output and show information of a stream that is catching all errors or exceptions on your systems.



The CLI stream dashboard documentation is available on [GitHub](#).

Browser stream dashboard

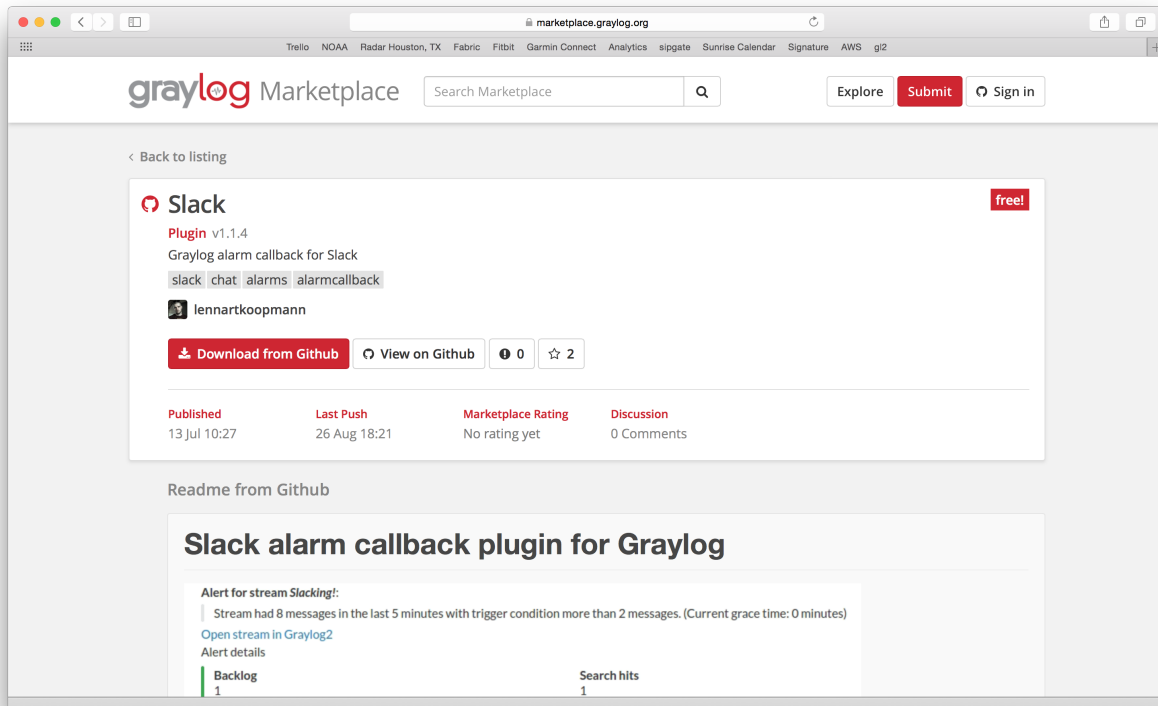
This official Graylog dashboard is showing live information of a specific stream in a web browser. It will display and automatically reload the most recent messages and alerts of a stream and is perfect to display on large screens in your office.



The browser stream dashboard documentation is available on [GitHub](#).

Graylog Marketplace

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



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.

4 Types of Add-Ons

Plug-Ins: Code that extends Graylog to support a specific use case that it doesn't support out of the box.

Content Pack: A file that can be uploaded into your Graylog system that sets up streams, inputs, extractors, dashboards, etc. to support a given log source or use case.

GELF Library: A library for a programming language or logging framework that supports sending log messages in GELF format for easy integration and pre-structured messages.

Other Solutions: Any other content or guide that helps you integrate Graylog with an external system or device. For example, how to configure a specific device to support a format Graylog understands out of the box.

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.

Frequently asked questions

General

Do I need to buy a license to use Graylog?

We believe software should be open and accessible to all. You should not have to pay to analyze your own data, no matter how much you have.

Graylog is licensed under the [GNU General Public License](#). We do not require license fees for production or non-production use.

How long do you support older versions of the Graylog product?

For our commercial support customers, we support older versions of Graylog up to 12 months after the next major release is available. So if you're using 1.X, you will continue to receive 1.X support up to a full year after 2.0 has been released.

Architecture

What is MongoDB used for?

Graylog uses MongoDB to store your configuration data, not your log data. Only metadata is stored, such as user information or stream configurations. None of your log messages are ever stored in MongoDB. This is why MongoDB does not have a big system impact, and you won't have to worry too much about scaling it. With our recommended setup architecture, MongoDB will simply run alongside your graylog-server processes and use almost no resources.

Can you guide me on how to replicate MongoDB for High Availability?

MongoDB actually supplies this information as part of their documentation. Check out :

- About [MongoDB Replica Sets](#).
- How to [convert a standalone MongoDB node to a replica set](#).

After you've done this, add all MongoDB nodes into the `replica_set` configuration in all `graylog-server.conf` files.

I have datacenters across the world and do not want logs forwarding from everywhere to a central location due to bandwidth, etc. How do I handle this?

You can have multiple graylog-server instances in a federated structure, and forward select messages to a centralized GL server.

Which load balancers do you recommend we use with Graylog?

You can use any. We have clients running AWS ELB, HAProxy, F5 BIG-IP, and KEMP.

Isn't Java slow? Does it need a lot of memory?

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

Give it a try, you might be surprised!

Does Graylog encrypt log data?

All log data is stored in Elasticsearch. They recommend you use *dm-crypt* at the file system level. Please see [here](#).

Where are the log files Graylog produces?

You can find the log data for Graylog under the below directory with timestamps and levels and exception messages. This is useful for debugging or when the server won't start.

```
/var/log/graylog-server/server.log
```

If you use the pre-build appliances, take a look into

```
/var/log/graylog/<servicename>/current
```

Installation / Setup

Should I download the OVA appliances or the separate packages?

If you are downloading Graylog for the first time to evaluate it, go for the appliance. It is really easy, and can be quickly setup so you can understand if Graylog is right for you. If you are wanting to use Graylog at some scale in production, and do things like high availability (Mongo replication) we recommend you go for the separate packages.

How do I find out if a specific log source is supported?

We support many log sources – and more are coming everyday. For a complete list, check out [Graylog Marketplace](#), the central repository of Graylog extensions. There are 4 types of content on the Marketplace:

- Plug-Ins: Code that extends Graylog to support a specific use case that it doesn't support out of the box.
- Content Pack: A file that can be uploaded into your Graylog system that sets up streams, inputs, extractors, dashboards, etc. to support a given log source or use case.

- GELF Library: A library for a programming language or logging framework that supports sending log messages in GELF format for easy integration and pre-structured messages.
- Other Solutions: Any other content or guide that helps you integrate Graylog with an external system or device. For example, how to configure a specific device to support a format Graylog understands out of the box.

Can I install the Graylog Server on Windows?

Even though our engineers say it is “technically possible”, don’t do it. The Graylog server is built using Java, so technically it can run anywhere. But we currently have it optimized to run better on other operating systems. If you don’t feel comfortable running your own Linux system, we recommend you use our Linux virtual appliance which will run under VMWare.

Functionality

Can Graylog automatically clean old data?

Absolutely we have data retention features, please see [here](#).

Does Graylog support LDAP / AD and its groups?

Yup, we’re all over this too with read/write roles and group permissions. To start, see [this](#). If you want to get very granular, you can go through the Graylog REST API.

Do we have a user audit log for compliance?

Coming soon in a future release – stay tuned!

It seems like Graylog has no reporting functionality?

That’s correct. We currently don’t have built-in reporting functionality that sends automated reports. However, you can 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.

Can I filter inbound messages before they are processed by the Graylog server?

Yes, check out our page on how to use [blacklisting](#).

Dedicated Partition for the Journal

If you create a dedicated Partition for your Kafka Journal, you need to watch that this is a clean directory. Even *lost+found* can break it, for [your reference](#).

Raise the Java Heap

If you need to raise the Java Heap of the Graylog Server or Elasticsearch in a System that runs as virtual appliances you can use *the advanced settings*.

On Systems that are installed with *DEB / APT* this setting can be made in `/etc/default/graylog-server`.

Systems that are installed with *RPM / YUM / DNF* the file is found in `/etc/sysconfig/graylog-server`.

Graylog & Integrations

What is the best way to integrate my applications to Graylog?

We recommend that you use *GELF*. It's easy for your application developers and eliminates the need to store the messages locally. Also, GELF can just send what app person wants so you don't have to build extractors or do any extra processing in Graylog.

I have a log source that creates dynamic syslog messages based on events and subtypes and grok patterns are difficult to use - what is the best way to handle this?

Not a problem! Use our *key=value extractor*.

I want to archive my log data. Can I write to another database, for example HDFS / Hadoop, from Graylog?

Yes, you can output data from Graylog to a different database. We currently have an HDFS output *plug-in* in the Marketplace - thank you *sivasamyk*!

It's also easy and fun to *write your own*, which you can then add to Graylog Marketplace for others to use.

I don't want to use Elasticsearch as my backend storage system – can I use another database, like MySQL, Oracle, etc?

You can, but we don't suggest you do. You will not be able to use our query functionality or our analytic engine on the dataset outside the system. We only recommend another database if you want it for secondary storage.

How can I create a restricted user to check internal Graylog metrics in my monitoring system?

You can create a restricted user which only has access to the `/system/metrics` resource on the Graylog REST API. This way it will be possible to integrate the internal metrics of Graylog into your monitoring system. Giving the user only restricted access will minimize the impact of these credentials getting compromised.

Send a POST request via the Graylog API Browser or curl to the `/roles` resource of the Graylog REST API:

```
{
  "name": "Metrics Access",
  "description": "Provides read access to all system metrics",
  "permissions": ["metrics:*"],
}
```

```
"read_only": false
}
```

The following curl command will create the required role (modify the URL of the Graylog REST API, here `http://127.0.0.1:12900`, and the user credentials, here `admin/admin`, according to your setup):

```
$ curl -u admin:admin -H "Content-Type: application/json" -X POST -d '{"name": "Metrics Access", "des
```

Troubleshooting

I'm sending in messages, and I can see they are being accepted by Graylog, but I can't see them in the search. What is going wrong?

A common reason for this issue is that the timestamp in the message is wrong. First, confirm that the message was received by selecting 'all messages' as the time range for your search. Then identify and fix the source that is sending the wrong timestamp.

I have configured an SMTP server or an output with TLS connection and receive handshake errors. What should I do?

Outbound TLS connections have CA (*certification authority*) certificate verification enabled by default. In case the target server's certificate is not signed by a CA found from trust store, the connection will fail. A typical symptom for this is the following error message in the server logs:

```
Caused by: javax.mail.MessagingException: Could not convert socket to TLS; nested exception is: java
```

This should be corrected by either adding the missing CA certificates to the Java default trust store (typically found at `$JAVA_HOME/jre/lib/security/cacerts`), or a custom store that is configured (by using `-Djavax.net.ssl.trustStore`) for the Graylog server process. The same procedure applies for both missing valid CAs and self-signed certificates.

For Debian/Ubuntu-based systems using OpenJDK JRE, CA certificates may be added to the systemwide trust store. After installing the JRE (including `ca-certificates-java`, ergo `ca-certificates` packages), place `name-of-certificate-dot-crt` (in PEM format) into `/usr/local/share/ca-certificates/` and run `/usr/sbin/update-ca-certificates`. The hook script in `/etc/ca-certificates/update.d/` should automatically generate `/etc/ssl/certs/java/cacerts`.

Fedora/RHEL-based systems may refer to [Shared System Certificates in the Fedora Project Wiki](#).

Suddenly parts of Graylog did not work as expected

If you notice multiple different non working parts in Graylog and found something like `java.lang.OutOfMemoryError: unable to create new native thread` in your Graylog Server logfile, you need to raise the process/thread limit of the graylog user. The limit can be checked with `ulimit -u` and you need to check how you can raise `nproc` in your OS.

Have another troubleshooting question?

See below for some additional support options where you can ask your question.

Support

I think I've found a bug, how do I report it?

Think you spotted a bug? Oh no! Please report it in our issue trackers so we can take a look at it. All issue trackers are hosted on [GitHub](#), tightly coupled to our code and milestones. Don't hesitate to open issues – we'll just close them if there is nothing to do. We have GitHub repos for the [web interface](#) and the [server](#).

I'm having issues installing or configuring Graylog, where can I go for support?

Check out the [Graylog Community Forums](#) – you can search for your problem which may already have an answer, or post a new question.

Another source is the [Graylog channel on Matrix.org](#) or the [#graylog IRC chat channel on freenode](#) (both are bridged, so you'll see messages from either channels). Our developers and a lot of community members hang out here. Just join the channel and add any questions, suggestions or general topics you have.

If you're looking for professional commercial support from the Graylog team, we do that too. Please [get in touch here](#) for more details.

Structured events from anywhere. Compressed and chunked.

The Graylog Extended Log Format (GELF) is a log format that avoids the shortcomings of classic plain syslog:

- Limited to length of 1024 bytes – Not much space for payloads like backtraces
- No data types in structured syslog. You don't know what is a number and what is a string.
- The RFCs are strict enough but there are so many syslog dialects out there that you cannot possibly parse all of them.
- No compression

Syslog is okay for logging system messages of your machines or network gear. GELF is a great choice for logging from within applications. There are libraries and appenders for many programming languages and logging frameworks so it is easy to implement. You could use GELF to send every exception as a log message to your Graylog cluster. You don't have to care about timeouts, connection problems or anything that might break your application from within your logging class because GELF can be sent via UDP.

GELF via UDP

Chunking

UDP datagrams are usually limited to a size of 8192 bytes. A lot of compressed information fits in there but you sometimes might just have more information to send. This is why Graylog supports chunked GELF.

You can define chunks of messages by prepending a byte header to a GELF message including a message ID and sequence number to reassemble the message later.

Most GELF libraries support chunking transparently and will detect if a message is too big to be sent in one datagram.

Of course TCP would solve this problem on a transport layer but it brings other problems that are even harder to tackle: You would have to care about slow connections, timeouts and other nasty network problems.

With UDP you may just lose a message while with TCP it could bring your whole application down when not designed with care.

Of course TCP makes sense in some (especially high volume environments) so it is your decision. Many GELF libraries support both TCP and UDP as transport. Some do even support HTTP.

Prepend the following structure to your GELF message to make it chunked:

- **Chunked GELF magic bytes - 2 bytes:** 0x1e 0x0f
- **Message ID - 8 bytes:** Must be the same for every chunk of this message. Identifying the whole message and is used to reassemble the chunks later. Generate from millisecond timestamp + hostname for example.
- **Sequence number - 1 byte:** The sequence number of this chunk. Starting at 0 and always less than the sequence count.
- **Sequence count - 1 byte:** Total number of chunks this message has.

All chunks **MUST** arrive within 5 seconds or the server will discard all already arrived and still arriving chunks. A message **MUST NOT** consist of more than 128 chunks.

Compression

When using UDP as transport layer, GELF messages can be sent uncompressed or compressed with either GZIP or ZLIB.

Graylog nodes detect the compression type in the GELF magic byte header automatically.

Decide if you want to trade a bit more CPU load for saving a lot of network bandwidth. GZIP is the protocol default.

GELF via TCP

At the current time, GELF TCP only supports uncompressed and non-chunked payloads. Each message needs to be delimited with a null byte (`\0`) when sent in the same TCP connection.

Attention: GELF TCP **does not support** compression due to the use of the null byte (`\0`) as frame delimiter.

GELF Payload Specification

Version 1.1 (11/2013)

A GELF message is a JSON string with the following fields:

- **version string (UTF-8)**
 - GELF spec version – “1.1”; **MUST** be set by client library.
- **host string (UTF-8)**
 - the name of the host, source or application that sent this message; **MUST** be set by client library.
- **short_message string (UTF-8)**
 - a short descriptive message; **MUST** be set by client library.
- **full_message string (UTF-8)**
 - a long message that can i.e. contain a backtrace; optional.
- **timestamp number**
 - Seconds since UNIX epoch with optional decimal places for milliseconds; *SHOULD* be set by client library. Will be set to the current timestamp (now) by the server if absent.
- **level number**

- the level equal to the standard syslog levels; optional, default is 1 (ALERT).
- **facility string (UTF-8)**
 - optional, deprecated. Send as additional field instead.
- **line number**
 - the line in a file that caused the error (decimal); optional, deprecated. Send as additional field instead.
- **file string (UTF-8)**
 - the file (with path if you want) that caused the error (string); optional, deprecated. Send as additional field instead.
- **_[additional field] string (UTF-8) or number**
 - every field you send and prefix with an underscore (_) will be treated as an additional field. Allowed characters in field names are any word character (letter, number, underscore), dashes and dots. The verifying regular expression is: `^[\\w\\.\\-]*$`. Libraries SHOULD not allow to send id as additional field (`_id`). Graylog server nodes omit this field automatically.

Example payload

This is an example GELF message payload. Any graylog-server node accepts and stores this as a message when GZIP/ZLIB compressed or even when sent uncompressed over a plain socket (without newlines):

```
{
  "version": "1.1",
  "host": "example.org",
  "short_message": "A short message that helps you identify what is going on",
  "full_message": "Backtrace here\n\nmore stuff",
  "timestamp": 1385053862.3072,
  "level": 1,
  "_user_id": 9001,
  "_some_info": "foo",
  "_some_env_var": "bar"
}
```

Sending GELF messages via UDP using netcat

Sending an example message to a GELF UDP input (running on host `graylog.example.com` on port 12201):

```
echo -n '{ "version": "1.1", "host": "example.org", "short_message": "A short message", "level": 5, '}
```

Sending GELF messages via TCP using netcat

Sending an example message to a GELF TCP input (running on host `graylog.example.com` on port 12201):

```
echo -n -e '{ "version": "1.1", "host": "example.org", "short_message": "A short message", "level": 5, '}
```

Sending GELF messages via TCP using curl

Sending an example message to a GELF HTTP input (running on `http://graylog.example.com:12201/gelf`):

```
curl -X POST -H 'Content-Type: application/json' -d '{ "version": "1.1", "host": "example.com" }
```

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 administrator needs to be able to log into the web interface of a log management product and select log files of a remote host (that has a forwarder running) for ingestion into the tool.

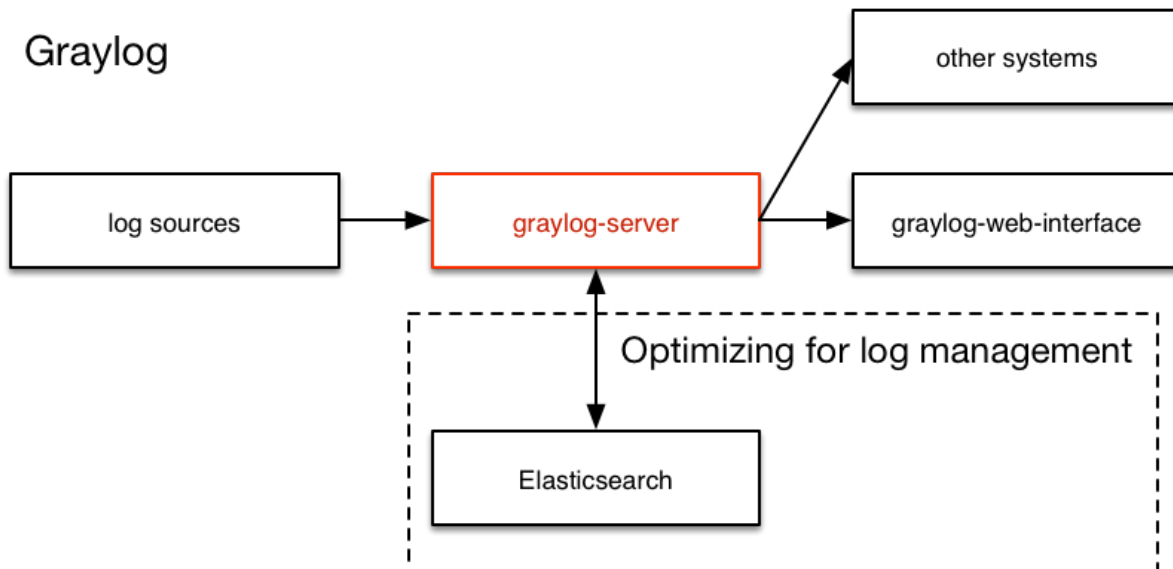
You also want to be able to see the health and configuration of all forwarders, data processors and indexers in a central place because the whole log management stack can easily involve thousands of machines if you include the log

emitting clients into this calculation. You need to be able to see which clients are forwarding log data and which are not to make sure that you are not missing any important data.

Graylog is coming the closest to the Splunk architecture:

- **Graylog was solely built as a log management system from the first line of code.** This makes it very efficient and easy to use.
- The `graylog-server` component sits in the middle and works around shortcomings of Elasticsearch (a full text search engine, not a log management system) for log management. It also builds an abstraction layer on top of it to make data access as easy as possible without having to select indices and write tedious time range selection filters, etc. - Just submit the search query and Graylog will take care of the rest for you.
- All parts of the system are tightly integrated and many parts speak to each other to make your job easier.
- Like Wordpress makes MySQL a good solution for blogging, Graylog makes Elasticsearch a good solution for logging. You should never have a system or frontend query Elasticsearch directly for log management so we are putting `graylog-server` in front of it.

Graylog



ELK



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.

Changelog

Graylog 2.0.3

Released 2016-06-20

<https://www.graylog.org/blog/58-graylog-v2-0-3-released>

Improvements

- Make Message#getStreamIds() more reliable. [Graylog2/graylog2-server#2378](#)
- Disabling a configured proxy for requests to localhost/127.0.0.1/:1. [Graylog2/graylog2-server#2305](#)

Bug fixes

- Update search query on auto refresh [Graylog2/graylog2-server#2385](#) [Graylog2/graylog2-server#2379](#)
- Fix permission checks for non admin users [Graylog2/graylog2-server#2366](#) [Graylog2/graylog2-server#2358](#)
- Fix display of total count of indices. [Graylog2/graylog2-server#2365](#) [Graylog2/graylog2-server#2359](#)
- Fix base URI for API documentation [Graylog2/graylog2-server#2362](#) [Graylog2/graylog2-server#2360](#)
- Fix link to API Browser on Node pages [Graylog2/graylog2-server#2361](#) [Graylog2/graylog2-server#2360](#)
- Calculate keyword from and to values on the fly [Graylog2/graylog2-server#2335](#) [Graylog2/graylog2-server#2301](#)
- Make MemoryAppender thread-safe [Graylog2/graylog2-server#2307](#) [Graylog2/graylog2-server#2302](#)
- Use right metrics to display buffer usage [Graylog2/graylog2-server#2300](#) [Graylog2/graylog2-server#2299](#)
- Check if props actually contain configuration fields before copying them [Graylog2/graylog2-server#2298](#) [Graylog2/graylog2-server#2297](#)

Graylog 2.0.2

Released: 2016-05-27

<https://www.graylog.org/blog/57-graylog-v2-0-2-released>

Improvements

- Improved user form. [Graylog2/graylog2-server#2261](#)
- Improved logging of plugin list on server startup. [Graylog2/graylog2-server#2290](#)

- Forbid empty passwords when using LDAP. [Graylog2/graylog2-server#2214](#) [Graylog2/graylog2-server#2283](#)
- Improved metrics page. [Graylog2/graylog2-server#2250](#) [Graylog2/graylog2-server#2255](#)
- Improved search histogram resolution auto selection. [Graylog2/graylog2-server#2148](#) [Graylog2/graylog2-server#2289](#)
- Improved cluster overview page. [Graylog2/graylog2-server#2291](#)

Bug Fixes

- Fixed concurrency issue with Drools. [Graylog2/graylog2-server#2119](#) [Graylog2/graylog2-server#2188](#) [Graylog2/graylog2-server#2231](#)
- Fixed problems with Internet Explorer. [Graylog2/graylog2-server#2246](#)
- Fixed issues with old dashboards. [Graylog2/graylog2-server#2262](#) [Graylog2/graylog2-server#2163](#)
- Fixed changing log levels via REST API. [Graylog2/graylog2-server#1904](#) [Graylog2/graylog2-server#2277](#)
- Fixed plugin inter-dependencies by using one class loader for all plugins. [Graylog2/graylog2-server#2280](#)

Plugin: Pipeline Processor

- Add syslog related rule functions. [Graylog2/graylog-plugin-pipeline-processor#19](#)
- Add concat rule functions. [Graylog2/graylog-plugin-pipeline-processor#20](#)
- Fixed problem with IP address function. [Graylog2/graylog-plugin-pipeline-processor#28](#) [Graylog2/graylog-plugin-pipeline-processor#32](#)
- Properly unescape strings in raw literals. [Graylog2/graylog-plugin-pipeline-processor#30](#) [Graylog2/graylog-plugin-pipeline-processor#31](#)

Graylog 2.0.1

Released: 2016-05-11

<https://www.graylog.org/blog/56-graylog-v2-0-1-released>

Improvements

- Improved session handling. [Graylog2/graylog2-server#2157](#)
- Included UPGRADING file in the build artifact. [Graylog2/graylog2-server#2170](#)
- Added rotation/retention settings back to the config file. [Graylog2/graylog2-server#2181](#)
- Improved proxy setup configuration settings. [Graylog2/graylog2-server#2156](#)
- Forbid wildcard host in `rest_transport_uri`. [Graylog2/graylog2-server#2205](#)
- Improved robustness for unreachable nodes. [Graylog2/graylog2-server#2206](#)
- Use a more lightweight API to get all index names and aliases. [Graylog2/graylog2-server#2194](#) [Graylog2/graylog2-server#2210](#)

Bug Fixes

- Fixed some documentation links.
- Fixed inverted stream rules. [Graylog2/graylog2-server#2160](#) [Graylog2/graylog2-server#2172](#)
- Fixed swallowed LDAP authentication exception. [Graylog2/graylog2-server#2176](#) [Graylog2/graylog2-server#2178](#)

- Fixed insecure handling of PID files. Thanks @juergenhoetzel! [Graylog2/graylog2-server#2174](#)
- Fixed alert conditions that have been created in Graylog 1.x. [Graylog2/graylog2-server#2169](#) [Graylog2/graylog2-server#2182](#)
- Fixed setting of application context. [Graylog2/graylog2-server#2191](#) [Graylog2/graylog2-server#2208](#)
- Fixed setting of custom Elasticsearch analyzer. [Graylog2/graylog2-server#2209](#)
- Fixed masking of password config values in the web interface. [Graylog2/graylog2-server#2198](#) [Graylog2/graylog2-server#2203](#)
- Fixed URL handling. [Graylog2/graylog2-server#2200](#) [Graylog2/graylog2-server#2213](#)

Plugin: Collector

- Rotate nxlog logfiles once a day by default.
- Add GELF TCP output for nxlog.

Graylog 2.0.0

Released: 2016-04-27

<https://www.graylog.org/blog/55-announcing-graylog-v2-0-ga>

Note: Please make sure to read the *Upgrade Guide* before upgrading to Graylog 2.0. There are breaking changes!

Feature Highlights

See the release announcement for details on the new features.

- Web interface no longer a separate process
- Support for Elasticsearch 2.x
- Live tail support
- Message Processing Pipeline
- Map Widget Plugin
- Collector Sidecar
- Streams filter UI
- Search for surrounding messages
- Query range limit
- Configurable query time ranges
- Archiving (commercial feature)

Bug Fixes

There have been lots of bug fixes since the 1.3 releases. We only list the ones that we worked on since the 2.0 alpha phase.

- Fixed issues with search page pagination and number of returned results: [Graylog2/graylog2-server#1759](#), [Graylog2/graylog2-server#1775](#), and [Graylog2/graylog2-server#1802](#)
- Avoid creating MongoDB collection multiple times: [Graylog2/graylog2-server#1747](#)

- Removed number of connected nodes in login page: [Graylog2/graylog2-server#1732](#)
- Fix dynamic search result histogram resolution: [Graylog2/graylog2-server#1764](#)
- Show overlay in Graylog web interface when Graylog server is not available: [Graylog2/graylog2-server#1762](#)
- Fix metric types: [Graylog2/graylog2-server#1784](#)
- Only load all metrics on demand: [Graylog2/graylog2-server#1782](#)
- Activate search refresh after selecting a refresh interval: [Graylog2/graylog2-server#1796](#)
- Fix circular dependencies: [Graylog2/graylog2-server#1789](#)
- Only render input forms when input type is available: [Graylog2/graylog2-server#1798](#)
- Document web interface configuration settings in graylog.conf. [Graylog2/graylog2-server#1777](#)
- Fix roles link to documentation. [Graylog2/graylog2-server#1805](#)
- Fix issue with field graphs. [Graylog2/graylog2-server#1811](#)
- Fix search result pagination. [Graylog2/graylog2-server#1812](#)
- Fix add to query button on quick values. [Graylog2/graylog2-server#1797](#)
- Fix URL to Graylog marketplace on content pack export page. [Graylog2/graylog2-server#1817](#)
- Fix elasticsearch node name for the Graylog client node. [Graylog2/graylog2-server#1814](#) and [Graylog2/graylog2-server#1820](#)
- Fix widget sorting for dashboards.
- Use `_` as default key separator in JSON Extractor. [Graylog2/graylog2-server#1841](#)
- Clarify that Graylog Collector needs access to `rest_listen_uri`. [Graylog2/graylog2-server#1847](#)
- Fix potential memory leak in GELF UDP handler. [Graylog2/graylog2-server#1857](#) [Graylog2/graylog2-server#1862](#)
- Fix user with correct permissions not allowed to view stream: [Graylog2/graylog2-server#1887](#), [Graylog2/graylog2-server#1902](#)
- Make pattern to check Graylog-managed indices stricter: [Graylog2/graylog2-server#1882](#), [Graylog2/graylog2-server#1888](#)
- Fix throughput counter: [Graylog2/graylog2-server#1876](#)
- Fix replay search link in dashboards: [Graylog2/graylog2-server#1835](#)
- Render server unavailable page more reliably: [Graylog2/graylog2-server#1867](#)
- Fix build issue with maven. [Graylog2/graylog2-server#1907](#) (Thanks @gitfrederic)
- Fix username in REST API access logs. [Graylog2/graylog2-server#1815](#) [Graylog2/graylog2-server#1918](#) (Thanks @mikkolehtisalo)
- Fix alert annotations in message histogram. [Graylog2/graylog2-server#1921](#)
- Fix problem with automatic input form reload. [Graylog2/graylog2-server#1870](#) [Graylog2/graylog2-server#1929](#)
- Fix asset caching. [Graylog2/graylog2-server#1924](#) [Graylog2/graylog2-server#1930](#)
- Fix issue with cursor jumps in the search bar. [Graylog2/graylog2-server#1911](#)
- Fix import of Graylog 1.x extractors. [Graylog2/graylog2-server#1831](#) [Graylog2/graylog2-server#1937](#)

- Field charts will now use the stream and time range of the current search. [Graylog2/graylog2-server#1785](#) [Graylog2/graylog2-web-interface#1620](#) [Graylog2/graylog2-web-interface#1618](#) [Graylog2/graylog2-web-interface#1485](#) [Graylog2/graylog2-server#1938](#)
- Improve browser validations. [Graylog2/graylog2-server#1885](#)
- Fix Internet Explorer support. [Graylog2/graylog2-server#1935](#)
- Fix issue where a user was logged out when accessing an unauthorized resource. [Graylog2/graylog2-server#1944](#)
- Fix issue with surrounding search. [Graylog2/graylog2-server#1946](#)
- Fix problem deleting dashboard widget where the plugin got removed. [Graylog2/graylog2-server#1943](#)
- Fix permission issue on user edit page. [Graylog2/graylog2-server#1964](#)
- Fix histogram time range selection via mouse. [Graylog2/graylog2-server#1895](#)
- Fix problems with duplicate Reflux store instances. [Graylog2/graylog2-server#1967](#)
- Create PID file earlier in the startup process. [Graylog2/graylog2-server#1969](#) [Graylog2/graylog2-server#1978](#)
- Fix content type detection for static assets. [Graylog2/graylog2-server#1982](#) [Graylog2/graylog2-server#1983](#)
- Fix caching of static assets. [Graylog2/graylog2-server#1982](#) [Graylog2/graylog2-server#1983](#)
- Show error message on malformed search query. [Graylog2/graylog2-server#1896](#)
- Fix parsing of GELF chunks. [Graylog2/graylog2-server#1986](#)
- Fix problems editing reader users profile. [Graylog2/graylog2-server#1984](#) [Graylog2/graylog2-server#1987](#)
- Fix problem with lost extractors and static fields on input update. [Graylog2/graylog2-server#1988](#) [Graylog2/graylog2-server#1923](#)
- Improve fetching cluster metrics to avoid multiple HTTP calls. [Graylog2/graylog2-server#1974](#) [Graylog2/graylog2-server#1990](#)
- Properly handle empty messages. [Graylog2/graylog2-server#1584](#) [Graylog2/graylog2-server#1995](#)
- Add 100-Continue support to HTTP inputs. [Graylog2/graylog2-server#1939](#) [Graylog2/graylog2-server#1998](#)
- Fix setting dashboard as start page for reader users. [Graylog2/graylog2-server#2005](#)
- Allow dots (".") in LDAP group name mappings. [Graylog2/graylog2-server#1458](#) [Graylog2/graylog2-server#2009](#)
- Update user edit form when username changes. [Graylog2/graylog2-server#2000](#)
- Fix issue with permissions in user form. [Graylog2/graylog2-server#1989](#)
- Update extractor example when message is loaded. [Graylog2/graylog2-server#1957](#) [Graylog2/graylog2-server#2013](#)
- Disable log4j2 shutdown hooks to avoid exception on shutdown. [Graylog2/graylog2-server#1795](#) [Graylog2/graylog2-server#2015](#)
- Fix styling issue with map widget. [Graylog2/graylog2-server#2003](#)
- Fix openstreetmap URL in map widget. [Graylog2/graylog2-server#1994](#)
- Fix problem with collector heartbeat validation. [Graylog2/graylog2-server#2002](#) [Graylog2/graylog2-web-interface#1726](#) [Graylog2/graylog-plugin-collector#3](#)
- Remove unused command line parameters. [Graylog2/graylog2-server#1977](#)
- Fixed timezone issues for date time processing in JSON parser. [Graylog2/graylog2-server#2007](#)

- Fixed JavaScript error with field truncation. [Graylog2/graylog2-server#2025](#)
- Fixed redirection if user is not authorized. [Graylog2/graylog2-server#1985](#) [Graylog2/graylog2-server#2024](#)
- Made changing the sort order in search result table work again. [Graylog2/graylog2-server#2028](#) [Graylog2/graylog2-server#2031](#)
- Performance improvements on “System/Indices” page. [Graylog2/graylog2-server#2017](#)
- Fixed content-type settings for static assets. [Graylog2/graylog2-server#2052](#)
- Fixed return code for invalid input IDs. [Graylog2/graylog2-server#1718](#) [Graylog2/graylog2-server#1767](#)
- Improved field analyzer UI. [Graylog2/graylog2-server#2022](#) [Graylog2/graylog2-server#2023](#)
- Fixed login with LDAP user. [Graylog2/graylog2-server#2045](#) [Graylog2/graylog2-server#2046](#) [Graylog2/graylog2-server#2069](#)
- Fixed issue with bad message timestamps to avoid data loss. [Graylog2/graylog2-server#2064](#) [Graylog2/graylog2-server#2065](#)
- Improved handling of Elasticsearch indices. [Graylog2/graylog2-server#2058](#) [Graylog2/graylog2-server#2062](#)
- Extractor form improvements for JSON and Grok extractors. [Graylog2/graylog2-server#1883](#) [Graylog2/graylog2-server#2020](#)
- Used search refresh to refresh field statistics. [Graylog2/graylog2-server#1961](#) [Graylog2/graylog2-server#2068](#)
- Fixed clicking zoom button in quick values. [Graylog2/graylog2-server#2040](#) [Graylog2/graylog2-server#2067](#)
- Web interface styling improvements.
- Replaced . in message field keys with a _ for ES 2.x compatibility. [Graylog2/graylog2-server#2078](#)
- Fixed unprocessed journal messages reload in node list. [Graylog2/graylog2-server#2083](#)
- Fixed problems with stale sessions on the login page. [Graylog2/graylog2-server#2073](#) [Graylog2/graylog2-server#2059](#) [Graylog2/graylog2-server#1891](#)
- Fixed issue with index retention strategies. [Graylog2/graylog2-server#2100](#)
- Fixed password change form. [Graylog2/graylog2-server#2103](#) [Graylog2/graylog2-server#2105](#)
- Do not show search refresh controls on the sources page. [Graylog2/graylog2-server#1821](#) [Graylog2/graylog2-server#2104](#)
- Wait for index being available before calculating index range. [Graylog2/graylog2-server#2061](#) [Graylog2/graylog2-server#2098](#)
- Fixed issue with sorting extractors. [Graylog2/graylog2-server#2086](#) [Graylog2/graylog2-server#2088](#)
- Improve DataTable UI component. [Graylog2/graylog-plugin-pipeline-processor#11](#)
- Move TCP keepalive setting into AbstractTcpTransport to simplify input development. [Graylog2/graylog2-server#2112](#)
- Fixed issue with Elasticsearch index template update. [Graylog2/graylog2-server#2089](#) [Graylog2/graylog2-server#2097](#)
- Ensure that tmpDir is writable when generating self-signed certs in TCP transports. [Graylog2/graylog2-server#2054](#) [Graylog2/graylog2-server#2096](#)
- Fixed default values for plugin configuration forms. [Graylog2/graylog2-server#2108](#) [Graylog2/graylog2-server#2114](#)
- Dashboard usability improvements. [Graylog2/graylog2-server#2093](#)
- Include default values in pluggable entities forms. [Graylog2/graylog2-server#2122](#)

- Ignore empty authentication tokens in `LdapUserAuthenticator`. [Graylog2/graylog2-server#2123](#)
- Add REST API authentication and permissions. [Graylog2/graylog-plugin-pipeline-processor#15](#)
- Require authenticated user in REST resources. [Graylog2/graylog-plugin-pipeline-processor#14](#)
- Lots of UI improvements in the web interface. [Graylog2/graylog2-server#2136](#)
- Fixed link to REST API browser. [Graylog2/graylog2-server#2133](#)
- Fixed CSV export skipping first chunk. [Graylog2/graylog2-server#2128](#)
- Fixed updating content packs. [Graylog2/graylog2-server#2138](#) [Graylog2/graylog2-server#2141](#)
- Added missing 404 page. [Graylog2/graylog2-server#2139](#)

Graylog 1.3.4

Released: 2016-03-16

<https://www.graylog.org/blog/49-graylog-1-3-4-is-now-available>

- Fix security issue which allowed redirecting users to arbitrary sites on login [Graylog2/graylog2-web-interface#1729](#)
- Fix issue with time-based index rotation strategy [Graylog2/graylog2-server#725](#) [Graylog2/graylog2-server#1693](#)
- Fix issue with `IndexFailureServiceImpl` [Graylog2/graylog2-server#1747](#)
- Add default `Content-Type` to `GettingStartedResource` [Graylog2/graylog2-server#1700](#)
- Improve OS platform detection [Graylog2/graylog2-server#1737](#)
- Add prefixes `GRAYLOG_` (environment variables) and `graylog.` (system properties) for overriding configuration settings [Graylog2/graylog2-server@48ed88d](#)
- Fix URL to Graylog Marketplace on `Extractor/Content Pack` pages [Graylog2/graylog2-server#1817](#)
- Use monospace font on message values [Graylog2/graylog2-web-interface@3cce368](#)

Graylog 1.3.3

Released: 2016-01-14

<https://www.graylog.org/graylog-1-3-3-is-now-available/>

- Absolute and relative time spans give different results [Graylog2/graylog2-server#1572](#) [Graylog2/graylog2-server#1463](#) [Graylog2/graylog2-server#1672](#) [Graylog2/graylog2-server#1679](#)
- Search result count widget not caching [Graylog2/graylog2-server#1640](#) [Graylog2/graylog2-server#1681](#)
- Field Value Condition Alert, does not permit decimal values [Graylog2/graylog2-server#1657](#)
- Correctly handle null values in nested structures in `JsonExtractor` [Graylog2/graylog2-server#1676](#) [Graylog2/graylog2-server#1677](#)
- Add `Content-Type` and `X-Graylog2-No-Session-Extension` to CORS headers [Graylog2/graylog2-server#1682](#) [Graylog2/graylog2-server#1685](#)
- Discard Message Output [Graylog2/graylog2-server#1688](#)

Graylog 1.3.2

Released: 2015-12-18

<https://www.graylog.org/graylog-1-3-2-is-now-available/>

- Deserializing a blacklist filter (`FilterDescription`) leads to `StackOverflowError` [Graylog2/graylog2-server#1641](#)

Graylog 1.3.1

Released: 2015-12-17

<https://www.graylog.org/graylog-1-3-1-is-now-available/>

- Add option to AMQP transports to bind the queue to the exchange [Graylog2/graylog2-server#1599](#) [Graylog2/graylog2-server#1633](#)
- Install a Graylog index template instead of set mappings on index creation [Graylog2/graylog2-server#1624](#) [Graylog2/graylog2-server#1628](#)

Graylog 1.3.0

Released: 2015-12-09

<https://www.graylog.org/graylog-1-3-ga-is-ready/>

- Allow index range calculation for a single index. [Graylog2/graylog2-server#1451](#) [Graylog2/graylog2-server#1455](#)
- Performance improvements for index ranges.
- Make internal server logs accessible via REST API. [Graylog2/graylog2-server#1452](#)
- Make specific configuration values accessible via REST API. [Graylog2/graylog2-server#1484](#)
- Added Replace Extractor. [Graylog2/graylog2-server#1485](#)
- Added a default set of Grok patterns. [Graylog2/graylog2-server#1495](#)
- Log operating system details on server startup. [Graylog2/graylog2-server#1244](#) [Graylog2/graylog2-server#1553](#)
- Allow reader users to set a dashboard as start page. [Graylog2/graylog2-web-interface#1681](#)
- Auto content pack loader – download and install content packs automatically
- Appliance pre-configured for log ingestion and analysis
- Show a getting started guide on first install. [Graylog2/graylog2-web-interface#1662](#)
- Include role permissions in “/roles/{rolename}/members” REST API endpoint. [Graylog2/graylog2-server#1549](#)
- Fixed `NullPointerException` in GELF output. [Graylog2/graylog2-server#1538](#)
- Fixed `NullPointerException` in GELF input handling. [Graylog2/graylog2-server#1544](#)
- Use the root user’s timezone for LDAP users by default. [Graylog2/graylog2-server#1000](#) [Graylog2/graylog2-server#1554](#)
- Fix display of JSON messages. [Graylog2/graylog2-web-interface#1686](#)

- Improve search robustness with missing Elasticsearch indices. [Graylog2/graylog2-server#1547](#)
[Graylog2/graylog2-server#1533](#)
- Fixed race condition between index creation and index mapping configuration. [Graylog2/graylog2-server#1502](#)
[Graylog2/graylog2-server#1563](#)
- Fixed concurrency problem in GELF input handling. [Graylog2/graylog2-server#1561](#)
- Fixed issue with widget value calculation. [Graylog2/graylog2-server#1588](#)
- Do not extend user sessions when updating widgets. [Graylog2/graylog2-web-interface#1655](#)
- Fixed compatibility mode for Internet Explorer. [Graylog2/graylog2-web-interface#1661](#) [Graylog2/graylog2-web-interface#1668](#)
- Fixed whitespace issue in extractor example. [Graylog2/graylog2-web-interface#1650](#)
- Fixed several issues on the indices page. [Graylog2/graylog2-web-interface#1691](#) [Graylog2/graylog2-web-interface#1692](#)
- Fixed permission issue for stream alert management. [Graylog2/graylog2-web-interface#1659](#)
- Fixed deletion of LDAP group mappings when updating LDAP settings. [Graylog2/graylog2-server#1513](#)
- Fixed dangling role references after deleting a role [Graylog2/graylog2-server#1608](#)
- Support LDAP Group Mapping for Sun Directory Server (new since beta.2) [Graylog2/graylog2-server#1583](#)

Graylog 1.2.2

Released: 2015-10-27

<https://www.graylog.org/graylog-1-2-2-is-now-available/>

- Fixed a whitespace issue in the extractor UI. [Graylog2/graylog2-web-interface#1650](#)
- Fixed the index description on the indices page. [Graylog2/graylog2-web-interface#1653](#)
- Fixed a memory leak in the GELF UDP handler code. (Analysis and fix contributed by @lightpriest and @onyx-master on GitHub. Thank you!) [Graylog2/graylog2-server#1462](#), [Graylog2/graylog2-server#1488](#)
- Improved the LDAP group handling code to handle more LDAP setups. [Graylog2/graylog2-server#1433](#), [Graylog2/graylog2-server#1453](#), [Graylog2/graylog2-server#1491](#), [Graylog2/graylog2-server#1494](#)
- Fixed email alerts for users with multiple email addresses. (LDAP setups) [Graylog2/graylog2-server#1439](#), [Graylog2/graylog2-server#1492](#)
- Improve index range handling performance. [Graylog2/graylog2-server#1465](#), [Graylog2/graylog2-server#1493](#)
- Fixed JSON extractor with null values. [Graylog2/graylog2-server#1475](#), [Graylog2/graylog2-server#1505](#)
- Fixed role assignment when updating user via REST API. [Graylog2/graylog2-server#1456](#), [Graylog2/graylog2-server#1507](#)

Graylog 1.2.1

Released: 2015-09-22

<https://www.graylog.org/graylog-1-2-1-is-now-available/>

- Fixed various issues around importing and applying content packs [Graylog2/graylog2-server#1423](#), [Graylog2/graylog2-server#1434](#), [Graylog2/graylog2-web-interface#1605](#), [Graylog2/graylog2-web-interface#1614](#)
- Fixed loading existing alarm callbacks that had been created with Graylog 1.0.x or earlier [Graylog2/graylog2-server#1428](#)
- Fixed compatibility problem with Elasticsearch 1.5.x and earlier [Graylog2/graylog2-server#1426](#)
- Fixed handling of statistical functions in field graphs [Graylog2/graylog2-web-interface#1604](#)
- Use correct title when adding quick values to a dashboard [Graylog2/graylog2-web-interface#1603](#)

Graylog 1.2.0

Released: 2015-09-14

<https://www.graylog.org/announcing-graylog-1-2-ga-release-includes-30-new-features/>

- Make sure existing role assignments survive on LDAP account sync. [Graylog2/graylog2-server#1405](#) | [Graylog2/graylog2-server#1406](#)
- Use memberOf query for ActiveDirectory to speed up LDAP queries. [Graylog2/graylog2-server#1407](#)
- Removed disable_index_range_calculation configuration option. [Graylog2/graylog2-server#1411](#)
- Avoid potentially long-running Elasticsearch cluster-level operations by only saving an index range if it actually changed. [Graylog2/graylog2-server#1412](#)
- Allow editing the roles of LDAP users. [Graylog2/graylog2-web-interface#1598](#)
- Improved quick values widget. [Graylog2/graylog2-web-interface#1487](#)

Graylog 1.2.0-rc.4

Released: 2015-09-08

<https://www.graylog.org/announcing-graylog-1-2-rc-4/>

- Deprecated MongoDB storage of internal metrics feature.
- Added customizable LDAP filter for user groups lookup. [Graylog2/graylog2-server#951](#)
- Allow usage of count and cardinality statistical functions in dashboard widgets. [Graylog2/graylog2-server#1376](#)
- Disabled index range recalculation on every index rotation. [Graylog2/graylog2-server#1388](#)
- Added automatic migration of user permissions to admin or reader roles. [Graylog2/graylog2-server#1389](#)
- Fixed widget problem with invalid timestamps. [Graylog2/graylog2-web-interface#1390](#)
- Added config option to enable TLS certificate validation in REST client. [Graylog2/graylog2-server#1393](#)
- Fixed rule matching issue in stream routing engine. [Graylog2/graylog2-server#1397](#)
- Changed default titles for stream widgets. [Graylog2/graylog2-web-interface#1476](#)
- Changed data filters to be case insensitive. [Graylog2/graylog2-web-interface#1585](#)
- Improved padding for stack charts. [Graylog2/graylog2-web-interface#1568](#)
- Improved resiliency when Elasticsearch is not available. [Graylog2/graylog2-web-interface#1518](#)

- Redirect to user edit form after updating a user. [Graylog2/graylog2-web-interface#1588](#)
- Improved dashboard widgets error handling. [Graylog2/graylog2-web-interface#1590](#)
- Fixed timing issue in streams UI. [Graylog2/graylog2-web-interface#1490](#)
- Improved indices overview page. [Graylog2/graylog2-web-interface#1593](#)
- Fixed browser back button behavior. [Graylog2/graylog2-web-interface#1594](#)
- Fixed accidental type conversion for number configuration fields in alarmcallback plugins. [Graylog2/graylog2-web-interface#1596](#)
- Fixed data type problem for extracted timestamps via grok. [Graylog2/graylog2-server#1403](#)

Graylog 1.2.0-rc.2

Released: 2015-08-31

<https://www.graylog.org/announcing-graylog-1-2-rc/>

- Implement global Elasticsearch timeout and add `elasticsearch_request_timeout` configuration setting. [Graylog2/graylog2-server#1220](#)
- Fixed lots of documentation links. [Graylog2/graylog2-server#1238](#)
- Groovy shell server removed. [Graylog2/graylog2-server#1266](#)
- Lots of index range calculation fixes. [Graylog2/graylog2-server#1274](#)
- New Raw AMQP input. [Graylog2/graylog2-server#1280](#)
- New Syslog AMQP input. [Graylog2/graylog2-server#1280](#)
- Updated bundled Elasticsearch to 1.7.1.
- The fields in configuration dialogs for inputs and outputs are now ordered. [Graylog2/graylog2-server#1282](#)
- Allow server startup without working Elasticsearch cluster. [Graylog2/graylog2-server#1136](#), [Graylog2/graylog2-server#1289](#)
- Added OR operator to stream matching. [Graylog2/graylog2-server#1292](#), [Graylog2/graylog2-web#1552](#)
- New stream router engine with better stream matching performance. [Graylog2/graylog2-server#1305](#), [Graylog2/graylog2-server#1309](#)
- Grok pattern import/export support for content packs. [Graylog2/graylog2-server#1300](#), [Graylog2/graylog2-web#1527](#)
- Added MessageListCodec interface for codec implementations that can decode multiple messages from one raw message. [Graylog2/graylog2-server#1307](#)
- Added keepalive configuration option for all TCP transports. [Graylog2/graylog2-server#1287](#), [Graylog2/graylog2-server#1318](#)
- Support for roles and LDAP groups. [Graylog2/graylog2-server#1321](#), [Graylog2/graylog2-server#951](#)
- Added timezone configuration option to date converter. [Graylog2/graylog2-server#1320](#), [Graylog2/graylog2-server#1324](#)
- Added alarmcallback history feature. [Graylog2/graylog2-server#1313](#), [Graylog2/graylog2-web#1537](#)
- Added more configuration options to GELF output. (TCP settings, TLS support) [Graylog2/graylog2-server#1337](#), [Graylog2/graylog2-server#979](#)

- Store timestamp and some other internal fields in Elasticsearch as doc values. Removed “elastic-search_store_timestamps_as_doc_values” option from configuration file. [Graylog2/graylog2-server#1335](#), [Graylog2/graylog2-server#1342](#)
- Added TLS support for GELF HTTP input. [Graylog2/graylog2-server#1348](#)
- Added JSON extractor. [Graylog2/graylog2-server#632](#), [Graylog2/graylog2-server#1355](#), [Graylog2/graylog2-web#1555](#)
- Added support for TLS client certificate authentication to all TCP based inputs. [Graylog2/graylog2-server#1357](#), [Graylog2/graylog2-server#1363](#)
- Added stacked chart widget. [Graylog2/graylog2-server#1284](#), [Graylog2/graylog2-web#1513](#)
- Added cardinality option to field histograms. [Graylog2/graylog2-web#1529](#), [Graylog2/graylog2-server#1303](#)
- Lots of dashboard improvements. [Graylog2/graylog2-web#1550](#)
- Replaced Gulp with Webpack. [Graylog2/graylog2-web#1548](#)
- Updated to Play 2.3.10.

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, despite editing permissions. [Graylog2/graylog2-web-interface#1456](#)
- Choose quick values widget size dynamically. [Graylog2/graylog2-web-interface#1422](#)
- Default field sort order is not guaranteed after reload. [Graylog2/graylog2-web-interface#1436](#)
- Toggling all fields in search list throws error and breaks pagination. [Graylog2/graylog2-web-interface#1434](#)
- Improve multi-line log messages support. [Graylog2/graylog2-web-interface#612](#)

- NPE when clicking a message from a deleted input on a stopped node. [Graylog2/graylog2-web-interface#1444](#)
- Auto created search syntax must use quotes for values with whitespaces in them. [Graylog2/graylog2-web-interface#1448](#)
- Quick Values doesn't update for new field. [Graylog2/graylog2-web-interface#1438](#)
- New Quick Values list too large. [Graylog2/graylog2-web-interface#1442](#)
- Unloading referenced alarm callback plugin breaks alarm callback listing. [Graylog2/graylog2-web-interface#1450](#)
- Add to search button doesn't work as expected for "level" field. [Graylog2/graylog2-web-interface#1453](#)
- Treat "*" query as empty query. [Graylog2/graylog2-web-interface#1420](#)
- Improve title overflow on widgets. [Graylog2/graylog2-web-interface#1430](#)
- Convert NaN to 0 on histograms. [Graylog2/graylog2-web-interface#1417](#)
- "<>" values in fields are unescaped and don't display in Quick Values. [Graylog2/graylog2-web-interface#1455](#)
- New quickvalues are not showing number of terms. [Graylog2/graylog2-web-interface#1411](#)
- Default index for split & index extractor results in an error. [Graylog2/graylog2-web-interface#1464](#)
- Improve behaviour when field graph fails to load. [Graylog2/graylog2-web-interface#1276](#)
- Unable to unpause streams, despite editing permissions. [Graylog2/graylog2-web-interface#1456](#)
- Wrong initial size of quick values pie chart. [Graylog2/graylog2-web-interface#1469](#)
- Problems refreshing data on quick values pie chart. [Graylog2/graylog2-web-interface#1470](#)
- Ignore streams with no permissions on message details. [Graylog2/graylog2-web-interface#1472](#)

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 `Dwebsockets.enabled=false` 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/radioctrl` 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 elastic-search_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.

Introduction

Graylog Enterprise, built on top of the Graylog open source platform, offers additional features that enable users to deploy Graylog at enterprise scale and apply Graylog to processes and workflows across the whole organization.

Please see the [Graylog Enterprise Page](#) for details.

Setup

Graylog Enterprise comes as a set of Graylog server plugins which need to be installed in addition to the Graylog open source setup.

Requirements

The following list shows the minimum required Graylog versions for the Graylog Enterprise plugins.

Table 29.1: Enterprise Version Requirements

Enterprise Version	Required Graylog Version
1.0.x	$\geq 2.0.2$, $< 2.1.0$

Installation

Once you [purchase a license for Graylog Enterprise](#), you will get download links for the tarballs and DEB/RPM packages in the confirmation mail.

Note: The Graylog Enterprise plugins need to be installed on all your Graylog nodes!

Tarball

The tarball includes the enterprise plugin JAR files.

```
$ tar -tzf graylog-enterprise-plugins-1.0.0.tgz
graylog-enterprise-plugins-1.0.0/plugin/graylog-plugin-archive-1.0.0.jar
graylog-enterprise-plugins-1.0.0/plugin/graylog-plugin-license-1.0.0.jar
```

Depending on the Graylog setup method you have used, you have to install the plugins into different locations.

Table 29.2: Plugin Installation Locations

Installation Method	Directory
<i>Virtual Machine Appliances</i>	<code>/opt/graylog/plugins/</code>
<i>Operating System Packages</i>	<code>/usr/share/graylog-server/plugin/</code>
<i>Manual Setup</i>	<code>/<extracted-graylog-tarball-path>/plugin/</code>

Also check the `plugin_dir` config option in your Graylog server configuration file. The default might have been changed.

Make sure to install the enterprise plugin JAR files alongside the other Graylog plugins. Your plugin directory should look similar to this after installing the enterprise plugins.

```
plugin/
-- graylog-plugin-archive-1.0.0.jar
-- graylog-plugin-collector-1.0.2.jar
-- graylog-plugin-enterprise-integration-1.0.2.jar
-- graylog-plugin-license-1.0.0.jar
-- graylog-plugin-map-widget-1.0.2.jar
-- graylog-plugin-pipeline-processor-1.0.0-beta.4.jar
-- usage-statistics-2.0.2.jar
```

DEB / RPM Package

We also provide DEB and RPM packages for the enterprise plugins.

Note: These packages can **only** be used when you installed Graylog via the *Operating System Packages!*

DEB

Use the following command to install the plugins via `dpkg`.

```
$ sudo dpkg -i graylog-enterprise-plugins-1.0.0.deb
```

RPM

Use the following command to install the plugins via `rpm`.

```
$ sudo rpm -Uvh graylog-enterprise-plugins-1.0.0-1.noarch.rpm
```

Server Restart

After you installed the Graylog Enterprise plugins you have to restart each of your Graylog servers to load the plugins.

Note: We recommend restarting one server at a time!

You should see something like the following in your Graylog server logs. It indicates that the plugins have been successfully loaded.

```
2016-05-30 12:06:34,606 INFO : org.graylog2.bootstrap.CmdLineTool - Loaded plugin: ArchivePlugin 1.0
2016-05-30 12:06:34,607 INFO : org.graylog2.bootstrap.CmdLineTool - Loaded plugin: License Plugin 1.0
```


License Installation

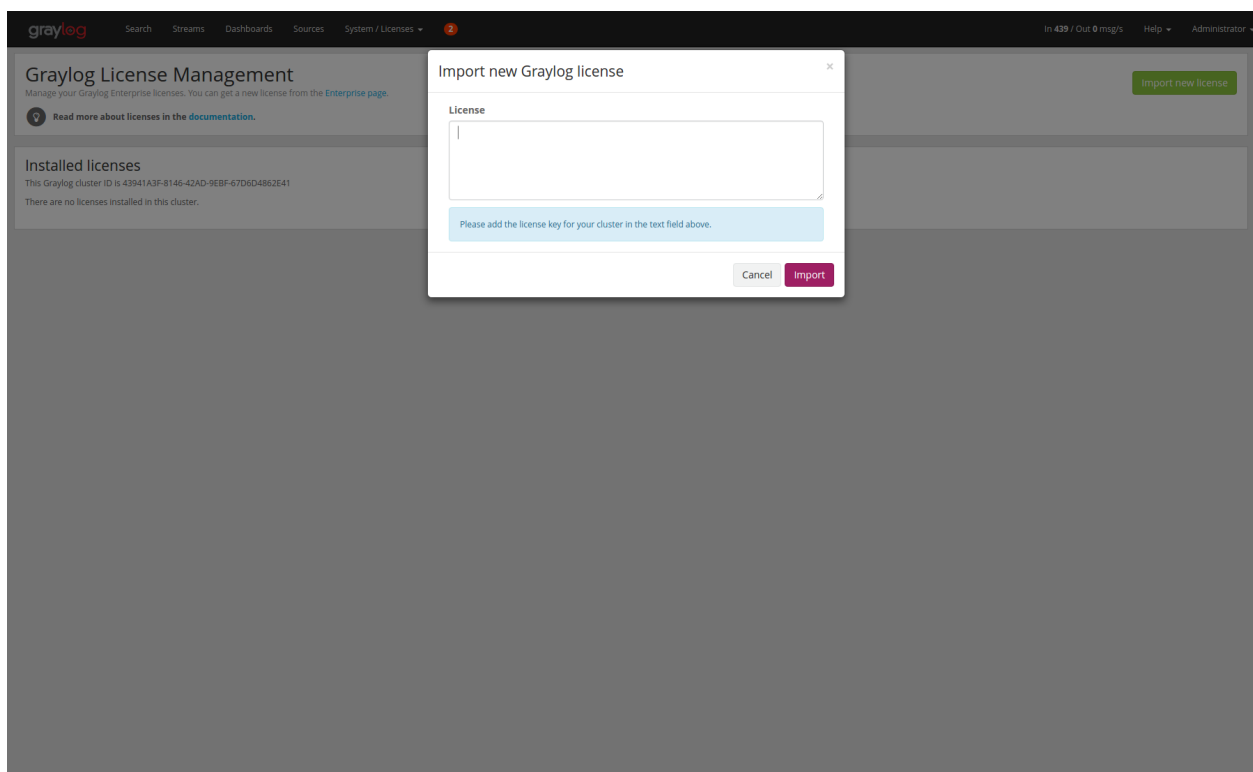
The Graylog Enterprise plugins require a valid license to use the additional features.

Once you have [purchased a license](#) you can import it into your Graylog setup by going through the following steps.

1. As an admin user, open the System/License page from the menu in the web interface.
2. Click the Import new license button in the top right hand corner.
3. Copy the license text from the confirmation email and paste it into the text field.
4. The license should be valid and a preview of your license details should appear below the text field.
5. Click Import to activate the license.

The license automatically applies to all nodes in your cluster without the need to restart your server nodes.

Note: If there are errors, please check that you copied the entire license from the email without line breaks. The same license is also attached as a text file in case it is wrongly formatted in the email.



Archiving

Graylog enables you to configure a retention period to automatically delete older messages - this is to help you control the costs of storage in Elasticsearch. But we know it's not ideal deciding between keeping less messages in Graylog or paying more for hardware. Additionally, many of you are required to store data for long periods of time due to compliance requirements like PCI or HIPAA.

The Archiving functionality allows you to archive log messages until you need to re-import them into Graylog for analysis. You can instruct Graylog to automatically archive log messages to compressed flat files on the local filesystem before retention cleaning kicks in and messages are deleted from Elasticsearch. Archiving also works through a REST call or the web interface if you don't want to wait until retention cleaning to happen. We chose flat files for this because they are vendor agnostic so you will always be able to access your data.

You can then do whatever you want with the archived files: move them to cheap storage, write them on tape, or even print them out if you need to! If you need to search through archived data in the future, you can move any selection of archived messages back into the Graylog archive folder, and the web interface will enable you to temporarily import the archive so you can analyze the messages again in Graylog.

Note: The archive plugin is a commercial feature.

Setup

The archive plugin is a commercial Graylog feature that can be installed in addition to the Graylog open source server.

Installation

Please see the [Graylog Enterprise setup page](#) for details on how to install the Archive plugin.

Configuration

The archive plugin can be configured via the Graylog web interface and does not need any changes in the Graylog server configuration file.

In the web interface menu navigate to "System/Archives" for the configuration.

The screenshot shows the Graylog web interface. At the top, there's a navigation bar with links to Search, Streams, Dashboards, Sources, and System / Archives. The main header says "Archives" and includes a sub-header: "The Graylog archive feature allows you to create archives from indices." Below this, there's a button for "Archive Documentation". A section titled "Create Archive for Index" allows selecting an index to be archived. The "Configuration" section displays the current settings: Archive path: /graylog-archive, Max segment size: 500.0MB, Compression type: Snappy, and Restore index batch size: 10000. An "Update configuration" button is present. A list of archives is shown below, including graylog_557, graylog_556, graylog_555, graylog_554, graylog_553, and graylog_552, each with creation time and document counts.

The “Configuration” section on the page shows the current configuration values. You can change the configuration by pressing “Update configuration”.

This screenshot shows the same Graylog Archives page as the previous one, but with the "Update Archive Configuration" dialog box open. The dialog box contains the following fields: "Archive Path" (set to /graylog-archive), "Max Segment Size (bytes)" (set to 524288000), "Compression Type" (set to Snappy), and "Restore index batch size" (set to 10000). Each field has a description of its purpose. The dialog box has "Cancel" and "Save" buttons at the bottom.

Archive Options

There are several configuration options to configure the archive plugin.

Table 30.1: Configuration Options

Name	Description
Archive Path	Directory on the master node where the archive files will be stored.
Max Segment Size	Maximum size (in <i>bytes</i>) of archive segment files.
Compression Type	Compression type that will be used to compress the archives.
Restore index batch size	Elasticsearch batch size when restoring archive files.

Archive Path

The archived indices will be stored in the *Archive Path* directory. This directory **needs to be writable for the Graylog server process** so the files can be stored.

Note: Only the **master** node needs access to the *Archive Path* directory because the archiving process runs on the master node.

We recommend to put the *Archive Path* directory onto a **separate disk or partition** to avoid any negative impact on the message processing should the archiving fill up the disk.

Max Segment Size

When archiving an index, the archive job writes the data into segments. The *Max Segment Size* setting sets the size limit for each of these data segments.

This allows control over the file size of the segment files to make it possible to process them with tools which have a size limit for files.

Once the size limit is reached, a new segment file will be started.

Example:

```
/path/to/archive/
graylog_201/
  archive-metadata.json
  archive-segment-0.gz
  archive-segment-1.gz
  archive-segment-2.gz
```

Compression Type

Archives will be compressed with gzip by default. This option can be changed to use a different compression type.

The selected compression type has a big impact on the time it takes to archive an index. Gzip for example is pretty slow but has a great compression rate. Snappy and LZ4 are way faster but the archives will be bigger.

Here is a comparison between the available compression algorithms with test data.

Table 30.2: Compression Type Comparison

Type	Index Size	Archive Size	Duration
gzip	1 GB	134 MB	15 minutes, 23 seconds
Snappy	1 GB	291 MB	2 minutes, 31 seconds
LZ4	1 GB	266 MB	2 minutes, 25 seconds

Note: Results with your data may vary! Make sure to test the different compression types to find the one that is best for your data.

Warning: The current implementation of LZ4 is not compatible with the LZ4 CLI tools, thus decompressing the LZ4 archives outside of Graylog is currently not possible.

Restore Index Batch Size

This setting controls the batch size for re-indexing archive data into Elasticsearch. When set to 1000, the restore job will re-index the archived data in document batches of 1000.

You can use this setting to control the speed of the restore process and also how much load it will generate on the Elasticsearch cluster. The **higher** the batch size, the **faster** the restore will progress and the **more** load will be put on your Elasticsearch cluster in addition to the normal message processing.

Make sure to tune this **carefully** to avoid any negative impact on your message indexing throughput and search speed!

Index Retention

Graylog is using configurable index retention strategies to delete old indices. By default indices can be *closed* or *deleted* if you have more than the configured limit.

The archive plugin offers a new index retention strategy that you can configure to automatically archive an index before closing or deleting it.

Index retention strategies can be configured in the system menu under “System/Indices”. Click “Update configuration” to change the index rotation and retention strategies.

The screenshot displays the Graylog web interface with the 'Update Index Settings' modal open. The modal is divided into two sections: 'Index Rotation Configuration' and 'Index Retention Configuration'. In the rotation section, the 'Index Time' is set to 'Index Time' and the 'Rotation period (ISO8601 Duration)' is set to 'PT2H' (2 hours). In the retention section, the 'Archive Index' strategy is selected, and the 'Max number of indices' is set to 20. The background shows the 'Indices' page with a list of indices and their details.

As with the regular index retention strategies, you can configure a max number of Elasticsearch indices. Once there are more indices than the configured limit, the oldest ones will be archived to the *Archive Path* and then closed or deleted. You can also decide to not do anything (*NONE*) after archiving an index. In that case **no cleanup of old indices will happen** and you have to take care of that yourself!

Usage

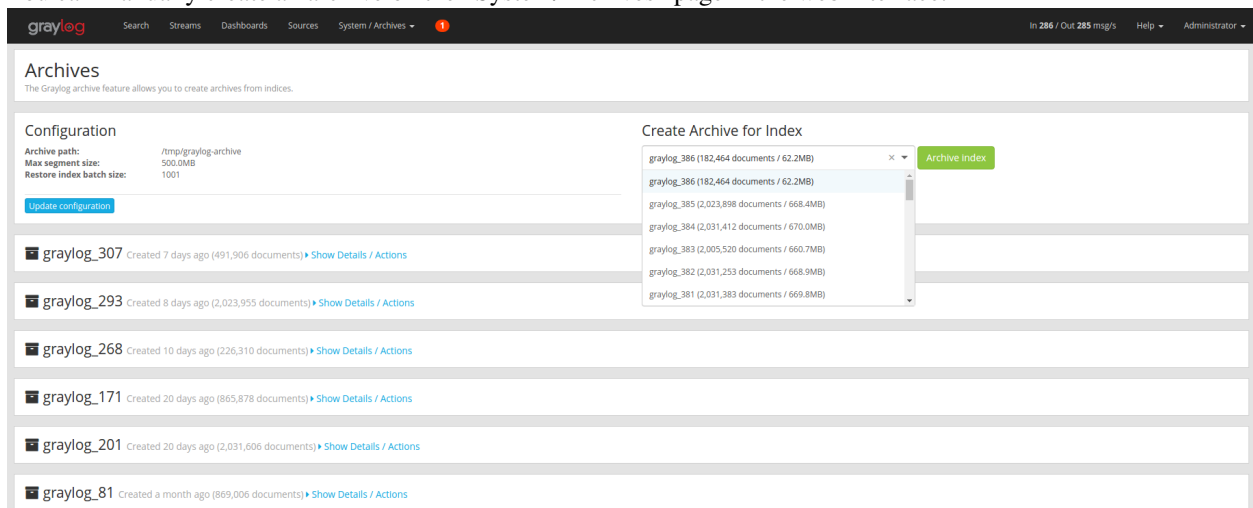
Creating Archives

There are three ways to create archives from the Graylog Elasticsearch indices.

- *Web Interface*
- *Index Retention*
- *REST API*

Web Interface

You can manually create an archive on the “System/Archives” page in the web interface.



On the “Create Archive for Index” section of the page is a form where you can select an index and archive it by pressing “Archive Index”.

Using this will just archive the index to disk and does not close it or delete it. This is a great way to test the archiving feature without changing your *index retention configuration*.

Index Retention

The archive plugin ships with an index retention strategy that can be used to automatically create archives before closing or deleting Elasticsearch indices.

This is the easiest way to automatically create archives without custom scripting.

Please see the *Index Retention Configuration* on how to configure it.

REST API

The archive plugin also offers a REST API that you can use to automate archive creation if you have some special requirements and need a more flexible way to do this.

Plugins/Archive/Archives : Manage index archives

Show/Hide | List Operations | Expand Operations | Raw

GET	/plugins/org.graylog.plugins.archive/archives	Returns all existing archives
POST	/plugins/org.graylog.plugins.archive/archives/{indexName}	Archive the given index
DELETE	/plugins/org.graylog.plugins.archive/archives/{indexName}	Delete the archive for the given index
POST	/plugins/org.graylog.plugins.archive/archives/{indexName}/restore	Restore the given index

An index can be archived with a simple curl command:

```
$ curl -s -u admin -X POST http://127.0.0.1:12900/plugins/org.graylog.plugins.archive/archives/graylog_386
Enter host password for user 'admin': *****
{
  "archive_job_config" : {
    "archive_path" : "/tmp/graylog-archive",
    "max_segment_size" : 524288000,
    "segment_filename_prefix" : "archive-segment",
    "metadata_filename" : "archive-metadata.json",
    "source_histogram_bucket_size" : 86400000,
    "restore_index_batch_size" : 1001,
    "segment_compression_type" : "SNAPPY"
  },
  "system_job" : {
    "id" : "cd7ebfa0-079b-11e6-9e1b-fa163e6e9b8a",
    "description" : "Archives indices and deletes them",
    "name" : "org.graylog.plugins.archive.job.ArchiveCreateSystemJob",
    "info" : "Archiving documents in index: graylog_386",
    "node_id" : "c5df7bff-cafd-4546-ac0a-5ccd2ba4c847",
    "started_at" : "2016-04-21T08:34:03.034Z",
    "percent_complete" : 0,
    "provides_progress" : true,
    "is_cancelable" : true
  }
}
```

That command started a system job in the Graylog server to create an archive for index `graylog_386`. The `system_job.id` can be used to check the progress of the job.

The REST API can be used to automate other archive related tasks as well, like restoring and deleting archives or updating the archive config. See the REST API browser on your Graylog server for details.

Restoring Archives

Note: The restore process adds load to your Elasticsearch cluster because all messages are basically **re-indexed**. Please make sure to keep this in mind and test with smaller archives to see how your cluster behaves. Also use the *Restore Index Batch Size* setting to control the Elasticsearch batch size on re-index.

The archive plugin offer two ways to restore archived indices.

- *Web Interface*

- *REST API*

A restored index has a `_restored_archive` in the index name to avoid conflicts with the original index. (if that still exists)

The screenshot shows the Graylog web interface with a list of indices. The index `graylog_307_restored_archive` is selected and its details are displayed. The details include the index name, a status of `reopened`, and a description: "Contains messages from 7 days ago up to 7 days ago (174.0MB / 491,906 messages)". Below this, there are two sections: "Primary shard operations" and "Total shard operations", each showing a table of operations (Index, Flush, Merge, Query, Fetch, Get, Refresh) and their respective counts and durations. At the bottom, there is a "Shard routing" section showing a list of shards (S0, S1, S2, S3) and a note about primary and replica shards. There are also buttons for "Recalculate index ranges", "Close index", and "Delete index".

Restored indices are also marked as reopened so they are **ignored** by index retention jobs and are not closed or deleted. That means you have to manually delete any restored indices **manually** once you do not need them anymore.

Web Interface

In the web interface you can restore an archive on the “System/Archives” page by selecting an archive from the list, open the archive details and clicking the “Restore Index” button.

REST API

As with archive creation you can also use the REST API to restore an archived index into the Elasticsearch cluster:

```
$ curl -s -u admin -X POST http://127.0.0.1:12900/plugins/org.graylog.plugins.archive/archives/graylog
Enter host password for user 'admin': *****
{
  "archive_metadata": {
    "archive_id": "graylog_307",
    "index_name": "graylog_307",
    "document_count": 491906,
    "created_at": "2016-04-14T14:31:50.787Z",
    "creation_duration": 142663,
    "timestamp_min": "2016-04-14T14:00:01.008Z",
    "timestamp_max": "2016-04-14T14:29:27.639Z",
    "id_mappings": {
      "streams": {
        "56fbafe0fb121a5309cef297": "nginx requests"
      },
      "inputs": {
        "56fbafe0fb121a5309cef290": "nginx error_log",
        "56fbafe0fb121a5309cef28d": "nginx access_log"
      },
      "nodes": {
        "c5df7bfff-cafd-4546-ac0a-5ccd2ba4c847": "graylog.example.org"
      }
    },
    "source_histogram": {
      "2016-04-14T00:00:00.000Z": {
        "example.org": 227567
      }
    },
    "segments": [
```

```

    {
      "path": "archive-segment-0.gz",
      "size": 21653755,
      "raw_size": 2359745839,
      "compression_type": "SNAPPY"
    }
  ],
  "system_job": {
    "id": "e680dcc0-07a2-11e6-9e1b-fa163e6e9b8a",
    "description": "Restores an index from the archive",
    "name": "org.graylog.plugins.archive.job.ArchiveRestoreSystemJob",
    "info": "Restoring documents from archived index: graylog_307",
    "node_id": "c5df7bff-cafd-4546-ac0a-5ccd2ba4c847",
    "started_at": "2016-04-21T09:24:51.468Z",
    "percent_complete": 0,
    "provides_progress": true,
    "is_cancelable": true
  }
}

```

The returned JSON payload contains the archive metadata and the system job description that runs the index restore process.

Restore into a separate cluster

As said earlier, restoring archived indices slow down your indexing speed because of added load. If you want to completely avoid adding more load to your Elasticsearch cluster, you can restore the archived indices on a different cluster.

To do that, you only have to transfer the archived indices to a different machine and put them into the configured *Archive Path*.

Each index archive is in a separate directory, so if you only want to transfer one index to a different machine, you only have to copy the corresponding directory into the archive path.

Example:

```

$ tree /tmp/graylog-archive
/tmp/graylog-archive
-- graylog_171
|   -- archive-metadata.json
|   -- archive-segment-0.gz
-- graylog_201
|   -- archive-metadata.json
|   -- archive-segment-0.gz
-- graylog_268
|   -- archive-metadata.json
|   -- archive-segment-0.gz
-- graylog_293
|   -- archive-metadata.json
|   -- archive-segment-0.gz
-- graylog_307
|   -- archive-metadata.json
|   -- archive-segment-0.gz
-- graylog_386
|   -- archive-metadata.json
|   -- archive-segment-0.gz

```

```
-- graylog_81
-- archive-metadata.json
-- archive-segment-0.gz
7 directories, 14 files
```

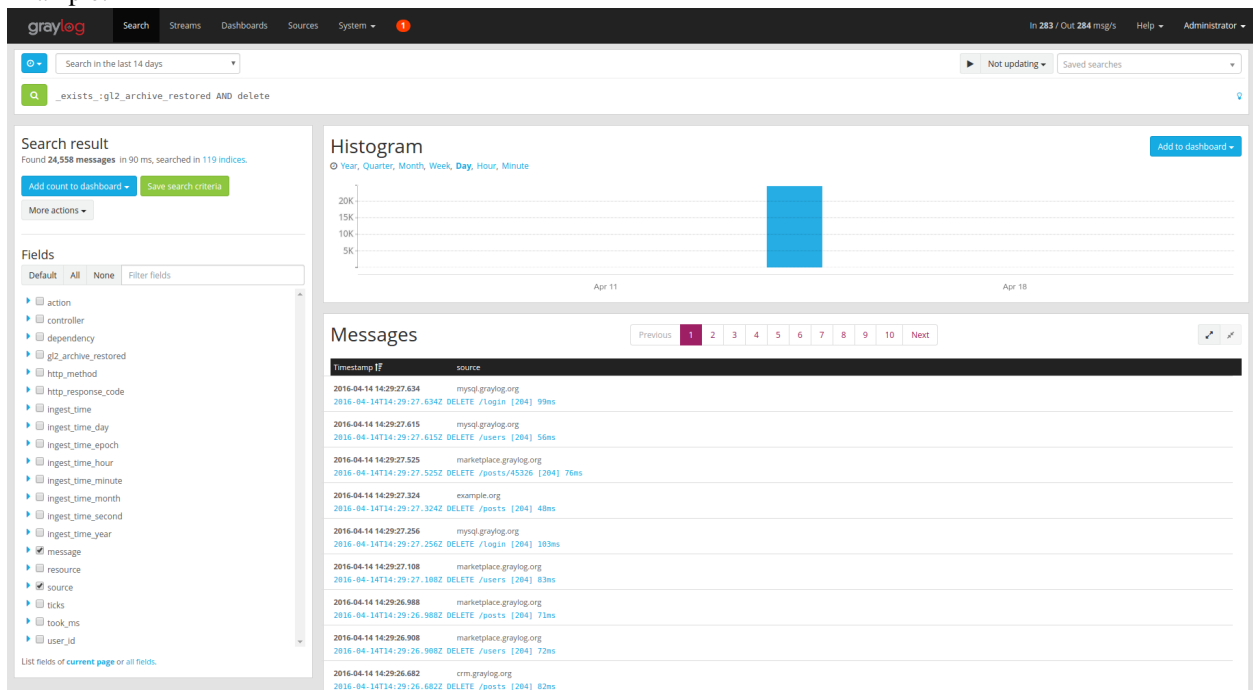
Searching in Restored Indices

Once an index has been restored from an archive it will be used by search queries automatically.

Every message that gets restored into an Elasticsearch index gets a special `gl2_archive_restored` field with value `true`. This allows you to only search in restored messages by using a query like:

```
_exists_:gl2_archive_restored AND <your search query>
```

Example:



If you want to exclude all restored messages from you query you can use:

```
_missing_:gl2_archive_restored AND <your search query>
```

Changelog

Graylog Enterprise 1.0.1

Released: 2016-06-08

Bugfix release for the archive plugin.

Plugin: Archive

Fixed problem when writing multiple archive segments

There was a problem when exceeding the max segment size so that multiple archive segments are written. The problem has been fixed and wrongly written segments can be read again.

Graylog Enterprise 1.0.0

Released: 2016-05-27

Initial Release including the Archive plugin.

Plugin: Archive

New features since the last beta plugin:

- Support for multiple compression strategies. (Snappy, LZ4, Gzip, None)