The StackLight InfluxDB-Grafana Plugin for Fuel Documentation

Release 0.10.2

Mirantis Inc.

Oct 11, 2016

CONTENTS

1 Overview 1

2 Installing and configuring StackLight InfluxDB-Grafana plugin 6

3 Using StackLight InfluxDB-Grafana plugin 14

CHAPTER

ONE

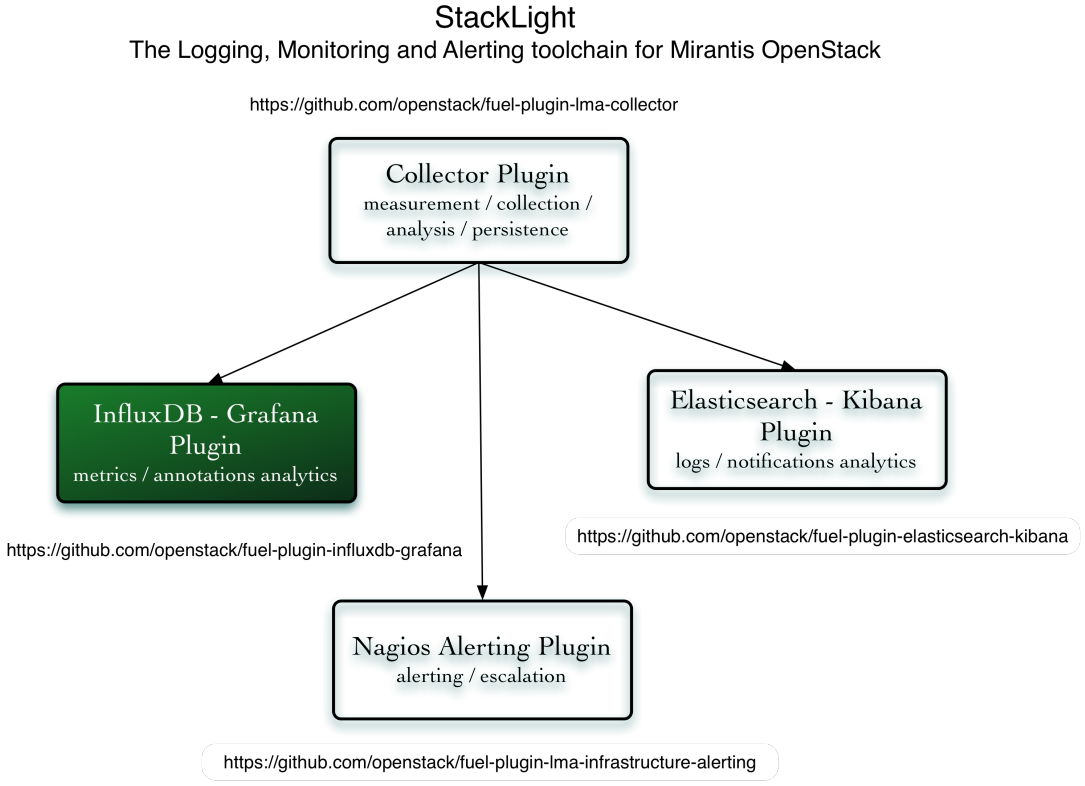
OVERVIEW

Introduction

The **StackLight InfluxDB-Grafana Plugin for Fuel** is used to install and configure InfluxDB and Grafana, which collectively provide access to the metrics analytics of Mirantis OpenStack. InfluxDB is a powerful distributed time-series database to store and search metrics time-series. The metrics analytics are used to visualize the time-series and the annotations produced by the StackLight Collector. The annotations contain insightful information about the faults and anomalies that resulted in a change of state for the clusters of nodes and services of the OpenStack environment.

The InfluxDB-Grafana plugin is an indispensable tool to answer the questions of what has changed in your OpenStack environment, when, and why. Grafana is installed with a collection of predefined dashboards for each of the OpenStack services that are monitored. Among those dashboards, the *Main Dashboard* provides a single pane of glass overview of your OpenStack environment status.

InfluxDB and Grafana are the key components of the LMA Toolchain project as shown in the figure below.



Key terms

The table below lists the key terms and acronyms that are used in this document.

Terms & acronyms	Definition
The Collector	The StackLight Collector is a smart monitoring agent running on every node. It collects and processes the metrics of your OpenStack environment.
InfluxDB	InfluxDB is a time-series, metrics, and analytics open-source database (MIT license). It is written in Go and has no external dependencies. InfluxDB is targeted at use cases for DevOps, metrics, sensor data, and real-time analytics.
Grafana	Grafana is a general-purpose dashboard and graph composer. It is focused on providing rich ways to visualize metrics time-series mainly through graphs but supports other ways to visualize data through a pluggable panel architecture. It has rich support for Graphite, InfluxDB, and OpenTSDB and also supports other data sources through plugins. Grafana is most commonly used for infrastructure monitoring, application monitoring, and metric analytics.

Requirements

Requirement	Version/Comment
Disk space	The plugin's specification requires provisioning at least 15 GB of disk space for the system, 10 GB for the logs, and 30 GB for the database. Therefore, the installation of the plugin will fail if there is less than 55 GB of disk space available on the node.
Mirantis OpenStack	8.0, 9.0
Hardware configuration	The hardware configuration (RAM, CPU, disk(s)) required by this plugin depends on the size of your cloud environment and other factors like the retention policy. An average setup would require a quad-core server with 8 GB of RAM and access to a 500-1000 IOPS disk. For sizeable production deployments it is strongly recommended to use a disk capable of 1000+ IOPS like an SSD. See the InfluxDB Hardware Sizing Guide for additional sizing information. It is highly recommended that you use a dedicated disk for your data storage. Otherwise, the InfluxDB-Grafana Plugin will use the root file system by default.

Limitations

The StackLight InfluxDB-Grafana plugin 0.10.2 has the following limitation:

- The size of an InfluxDB cluster the Fuel plugin can deploy is limited to three nodes. Additionally, each node of the InfluxDB cluster is configured to run under the *meta* node role and the *data* node role. Therefore, it is not possible to separate the nodes participating in the Raft consensus cluster from the nodes accessing the data replicas.

Release notes

Version 0.10.2

The StackLight InfluxDB-Grafana plugin 0.10.2 contains the following updates:

- Add an option to store the InfluxDB WAL in memory. This option is disabled by default.
- Updated the documentation to emphasize the need to use fast disks like an SSD.

Version 0.10.1

The StackLight InfluxDB-Grafana plugin 0.10.1 contains the following updates:

- Added support for wildcard SSL certificates. See #1608665.
- Fixed the UI issue with the LDAP protocol radio button. See #1599778.

Version 0.10.0

The StackLight InfluxDB-Grafana plugin 0.10.0 contains the following updates:

- Added support for LDAP(S) authentication to access Grafana.
- Added support for TLS encryption to access Grafana. A PEM file obtained by concatenating the SSL certificate with the private key of the server must be provided in the settings of the plugin to configure the TLS termination.
- Upgraded to InfluxDB v0.11.1.
- Upgraded to Grafana v3.0.4.

Version 0.9.0

The StackLight InfluxDB-Grafana plugin 0.9.0 contains the following updates:

- Added a new dashboard for hypervisor metrics.
- Added a new dashboard for InfluxDB cluster.
- Added a new dashboard for Elasticsearch cluster.
- Upgraded to Grafana 2.6.0.
- Upgraded to InfluxDB 0.10.0.
- Added support for InfluxDB clustering (beta state).
- Added the capability to use MySQL as Grafana back end to support HA.

Version 0.8.0

The StackLight InfluxDB-Grafana plugin 0.8.0 contains the following updates:

- Added support for the `influxdb_grafana` Fuel plugin role instead of the `base-os` role which had several limitations.
- Added support for retention policy configuration.
- Upgraded to InfluxDB 0.9.4 which brings metrics time-series with tagging.
- Upgraded to Grafana 2.5.0.
- Improved dashboard visualization.
- Added a new self-monitoring dashboard.

Version 0.7.0

The initial release of the plugin. This is a beta version.

Licenses

Third-party components

Name	Project website	License
InfluxDB	https://influxdb.com/	MIT
Grafana	http://grafana.org/	Apache v2

Puppet modules

Name	Project website	License
Apt	https://github.com/puppetlabs/puppetlabs-apt	Apache v2
Concat	https://github.com/puppetlabs/puppetlabs-concat	Apache v2
Stdlib	https://github.com/puppetlabs/puppetlabs-stdlib	Apache v2
IniFile	https://github.com/puppetlabs/puppetlabs-inifile	Apache v2
Grafana	https://github.com/bfraser/puppet-grafana	Apache v2

References

- The InfluxDB-Grafana plugin project at GitHub
- The official InfluxDB documentation
- The official Grafana documentation

CHAPTER

TWO

INSTALLING AND CONFIGURING STACKLIGHT INFLUXDB-GRAFANA PLUGIN

Introduction

You can install the StackLight InfluxDB-Grafana plugin using one of the following options:

- Install using the RPM file
- Install from source

The following is a list of software components installed by the StackLight InfluxDB-Grafana plugin:

Components	Version
InfluxDB	v0.11.1 for Ubuntu (64-bit)
Grafana	v3.0.4 for Ubuntu (64-bit)

Install using the RPM file of the Fuel plugins catalog

To install the StackLight InfluxDB-Grafana Fuel plugin using the RPM file of the Fuel plugins catalog:

1. Go to the Fuel Plugins Catalog.
2. From the *Filter* drop-down menu, select the Mirantis OpenStack version you are using and the *Monitoring* category.
3. Download the RPM file.
4. Copy the RPM file to the Fuel Master node:

```
[root@home ~]# scp influxdb_grafana-0.10-0.10.2-1.noarch.rpm \
root@<Fuel Master node IP address>:
```

5. Install the plugin using the Fuel Plugins CLI:

```
[root@fuel ~]# fuel plugins --install influxdb_grafana-0.10-0.10.2-1.noarch.rpm
```

6. Verify that the plugin is installed correctly:

```
[root@fuel ~]# fuel plugins --list
id | name                 | version  | package_version
---|----------------------|----------|----------------
1  | influxdb_grafana     | 0.10.2   | 4.0.0
```

Install from source

Alternatively, you may want to build the RPM file of the plugin from source if, for example, you want to test the latest features of the master branch or customize the plugin.

Note: Running a Fuel plugin that you built yourself is at your own risk and will not be supported.

To install the StackLight InfluxDB-Grafana Plugin from source, first prepare an environment to build the RPM file. The recommended approach is to build the RPM file directly onto the Fuel Master node so that you will not have to copy that file later on.

To prepare an environment and build the plugin:

1. Install the standard Linux development tools:

```
[root@home ~] yum install createrepo rpm rpm-build dpkg-devel
```

2. Install the Fuel Plugin Builder. To do that, first get pip:

```
[root@home ~] easy_install pip
```

3. Then install the Fuel Plugin Builder (the *fpb* command line) with *pip*:

```
[root@home ~] pip install fuel-plugin-builder
```

Note: You may also need to build the Fuel Plugin Builder if the package version of the plugin is higher than the package version supported by the Fuel Plugin Builder you get from *pypi*. For instructions on how to build the Fuel Plugin Builder, see the *Install Fuel Plugin Builder* section of the Fuel Plugin SDK Guide.

4. Clone the plugin repository:

```
[root@home ~] git clone https://github.com/openstack/fuel-plugin-influxdb-grafana.
↪git
```

5. Verify that the plugin is valid:

```
[root@home ~] fpb --check ./fuel-plugin-influxdb-grafana
```

6. Build the plugin:

```
[root@home ~] fpb --build ./fuel-plugin-influxdb-grafana
```

To install the plugin:

1. Once you create the RPM file, install the plugin:

```
[root@fuel ~] fuel plugins --install ./fuel-plugin-influxdb-grafana/*.noarch.rpm
```

2. Verify that the plugin is installed correctly:

```
[root@fuel ~]# fuel plugins --list
id | name                 | version   | package_version
---|----------------------|-----------|----------------
1  | influxdb_grafana     | 0.10.2    | 4.0.0
```

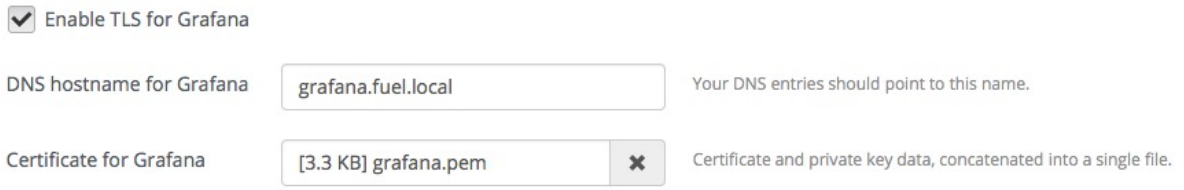
Plugin configuration

To configure the StackLight InfluxDB-Grafana plugin:

1. Create a new environment as described in Create a new OpenStack environment.
2. In the Fuel web UI, click the *Settings* tab and select the *Other* category.
3. Scroll down through the settings until you find *The InfluxDB-Grafana Server Plugin* section. You should see a page like this:



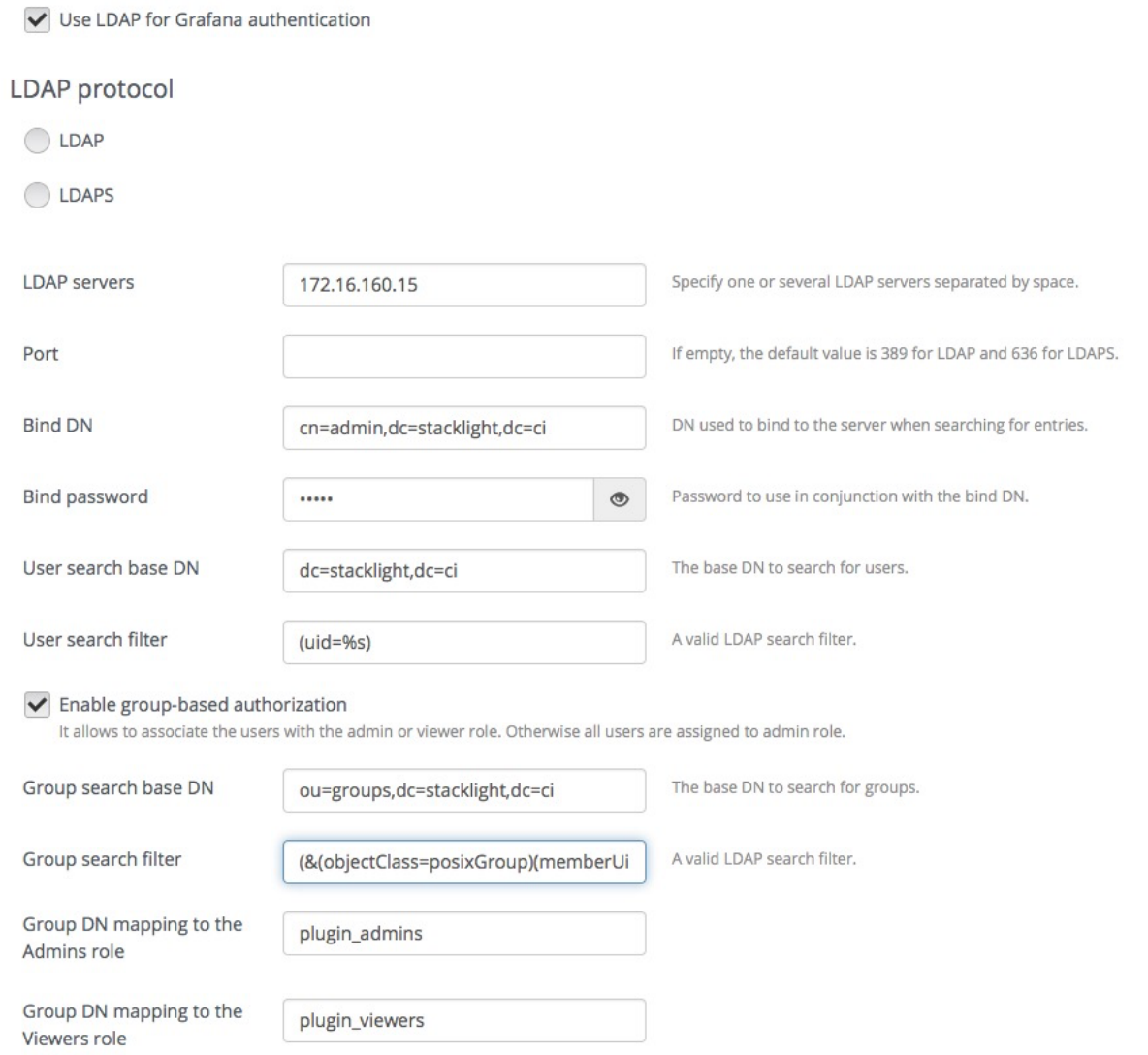
4. Select *The InfluxDB-Grafana Server Plugin* and fill in the required fields as indicated below.
   (a) Specify the number of days of retention for your data.
   (b) Specify the InfluxDB admin password (called root password in the InfluxDB documentation.
   (c) Specify the database name (the default is `lma`).
   (d) Specify the InfluxDB username and password.
   (e) Specify the Grafana username and password.
5. The plugin uses a MySQL database to store its configuration data, such as the dashboard templates.
   (a) Select *Local MySQL* if you want to create the Grafana database using the MySQL server of the OpenStack control plane. Otherwise, select *Remote server* and specify the fully qualified name or the IP address of the MySQL server you want to use.
   (b) Specify the MySQL database name, username, and password that will be used to access that database.
6. Select *Enable TLS for Grafana* if you want to encrypt your Grafana credentials (username, password). Then, fill in the required fields as indicated below.



(a) Specify the DNS name of the Grafana server. This parameter is used to create a link in the Fuel dashboard to the Grafana server.

(b) Specify the location of a PEM file that contains the certificate and the private key of the Grafana server that will be used in TLS handchecks with the client.

7. Select *Use LDAP for Grafana authentication* if you want to authenticate to Grafana through LDAP. Then, fill in the required fields as indicated below.



(a) Select *LDAPS* if you want to enable LDAP authentication over SSL.

(b) Specify one or several LDAP server addresses separated by space. These addresses must be accessible from the node where Grafana is installed. Addresses outside the *management network* are not routable by

default (see the note below).

(c) Specify the LDAP server port number or leave it empty to use the defaults.

(d) Specify the *Bind DN* of a user who has search privileges on the LDAP server.

(e) Specify the password of the user identified by the *Bind DN* above.

(f) Specify the *User search base DN* in the Directory Information Tree (DIT) from where to search for users.

(g) Specify a valid user search filter, for example, `(uid=%s)`. The result of the search should be a unique user entry.

You can further restrict access to Grafana to those users who are members of a specific LDAP group.

(a) Select *Enable group-based authorization*.

(b) Specify the LDAP group *Base DN* in the DIT from where to search for groups.

(c) Specify the LDAP group search filter. For example, `(&(objectClass=posixGroup)(memberUid=%s))`.

(d) Specify the CN of the LDAP group that will be mapped to the *admin role*.

(e) Specify the CN of the LDAP group that will be mapped to the *viewer role*.

Users who have the *admin role* can modify the Grafana dashboards or create new ones. Users who have the *viewer role* can only visualize the Grafana dashboards.

8. Configure your environment as described in Configure your Environment.

Note: By default, StackLight is configured to use the *management network* of the so-called Default Node Network Group. While this default setup may be appropriate for small deployments or evaluation purposes, it is recommended that you not use this network for StackLight in production. Instead, create a network dedicated to StackLight using the networking templates Fuel capability. Using a dedicated network for StackLight will improve performance and reduce the monitoring footprint on the control plane. It will also facilitate access to the Gafana UI after deployment, as the *management network* is not routable.

---

9. Click the *Nodes* tab and assign the *InfluxDB_Grafana* role to the node or multiple nodes where you want to install the plugin.

The example below shows that the *InfluxDB_Grafana* role is assigned to three nodes alongside with the *Alerting_Infrastructure* and the *Elasticsearch_Kibana* roles. The three plugins of the LMA toolchain back-end servers are installed on the same nodes. You can assign the *InfluxDB_Grafana* role to either one node (standalone install) or three nodes for HA.



Note: Currently, installing the InfluxDB server on more than three nodes is not possible using the Fuel plugin. Similarly, installing the InfluxDB server on two nodes is not recommended to avoid split-brain situations in the Raft consensus of the InfluxDB cluster, as well as the *Pacemaker* cluster, which is responsible for the VIP

address failover. It is possible to add or remove nodes with the *InfluxDB_Grafana* role in the cluster after deployment.

10. If required, adjust the disk partitioning as described in Configure disk partitioning.

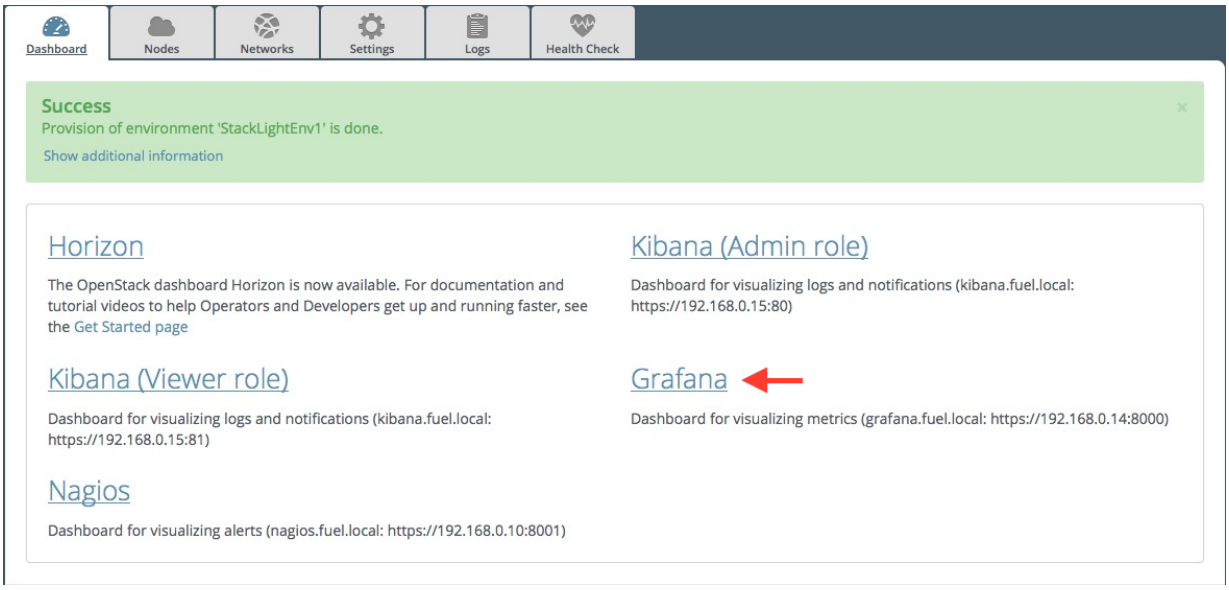
    By default, the InfluxDB-Grafana Plugin allocates:

    - 20% of the first available disk for the operating system by honoring a range of 15 GB minimum to 50 GB maximum.
    - 10 GB for `/var/log`.
    - At least 30 GB for the InfluxDB database in `/var/lib/influxdb`.

11. Deploy your environment as described in Deploy an OpenStack environment.

Plugin verification

Depending on the number of nodes and deployment setup, deploying a Mirantis OpenStack environment may take 30 minutes to several hours. Once the deployment is complete, you should see a deployment success notification with a link to the Grafana web UI as shown below.



Verify InfluxDB

To verify that the InfluxDB cluster is running properly, first retrieve the InfluxDB cluster VIP address:

1. On the Fuel Master node, find the IP address of a node where the InfluxDB server is installed using the **`fuel nodes`** command. For example:

```
[root@fuel ~]# fuel nodes
id | status   | name              | cluster | ip         | mac | roles            |
---|----------|-------------------|---------|------------|-----|------------------|
1  | ready    | Untitled (fa:87)  | 1       | 10.109.0.8 | ... | influxdb_grafana |
2  | ready    | Untitled (12:aa)  | 1       | 10.109.0.3 | ... | influxdb_grafana |
3  | ready    | Untitled (4e:6e)  | 1       | 10.109.0.7 | ... | influxdb_grafana |
```

2. Log in to any of these nodes through SSH, for example, to `node-1`
3. Run the following command:

```
root@node-1:~# hiera lma::influxdb::vip
10.109.1.4
```

Where `10.109.1.4` is the virtual IP address (VIP) of your InfluxDB cluster.

4. Using that VIP address, run the following command:

```
root@node-1:~# /usr/bin/influx -database lma -password lmapass \
--username root -host 10.109.1.4 -port 8086
Visit https://enterprise.influxdata.com to register for updates,
InfluxDB server management, and monitoring.
Connected to http://10.109.1.4:8086 version 0.11.1
InfluxDB shell 0.11.1
>
```

The example above shows that executing `/usr/bin/influx` starts an interactive CLI and automatically connects to the InfluxDB server. Then run the following command:

```
> show series
```

You should see a dump of all the time-series collected so far. Then run:

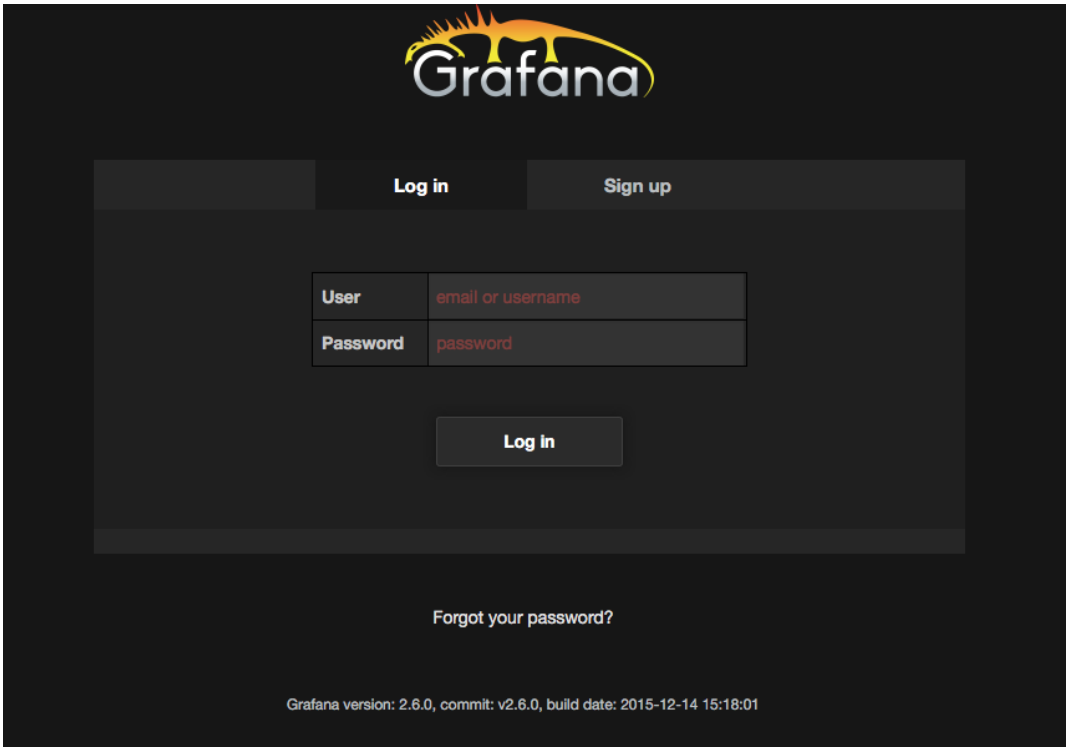
```
> show servers
name: data_nodes
----------------
id      http_addr       tcp_addr
1       node-1:8086     node-1:8088
3       node-2:8086     node-2:8088
5       node-3:8086     node-3:8088

name: meta_nodes
----------------
id      http_addr       tcp_addr
1       node-1:8091     node-1:8088
2       node-2:8091     node-2:8088
4       node-3:8091     node-3:8088
```

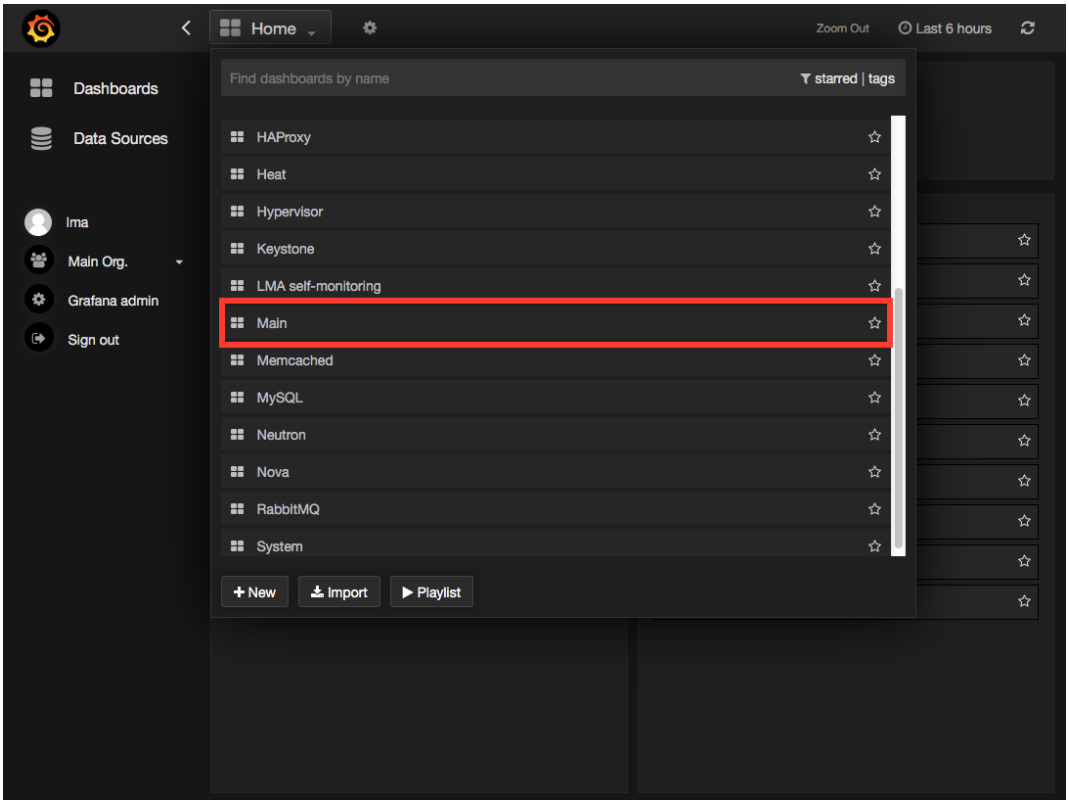
You should see a list of nodes participating in the InfluxDB cluster with their roles (data or meta).

Verify Grafana

1. Log in to the Fuel web UI.
2. In the *Dashboard* tab, click *Grafana*. If your DNS is not set up, enter the IP address and the port number.
3. Authenticate using your credentials.



You should be redirected to the *Grafana Home Page* where you can select a dashboard as shown below.



CHAPTER

THREE

USING STACKLIGHT INFLUXDB-GRAFANA PLUGIN

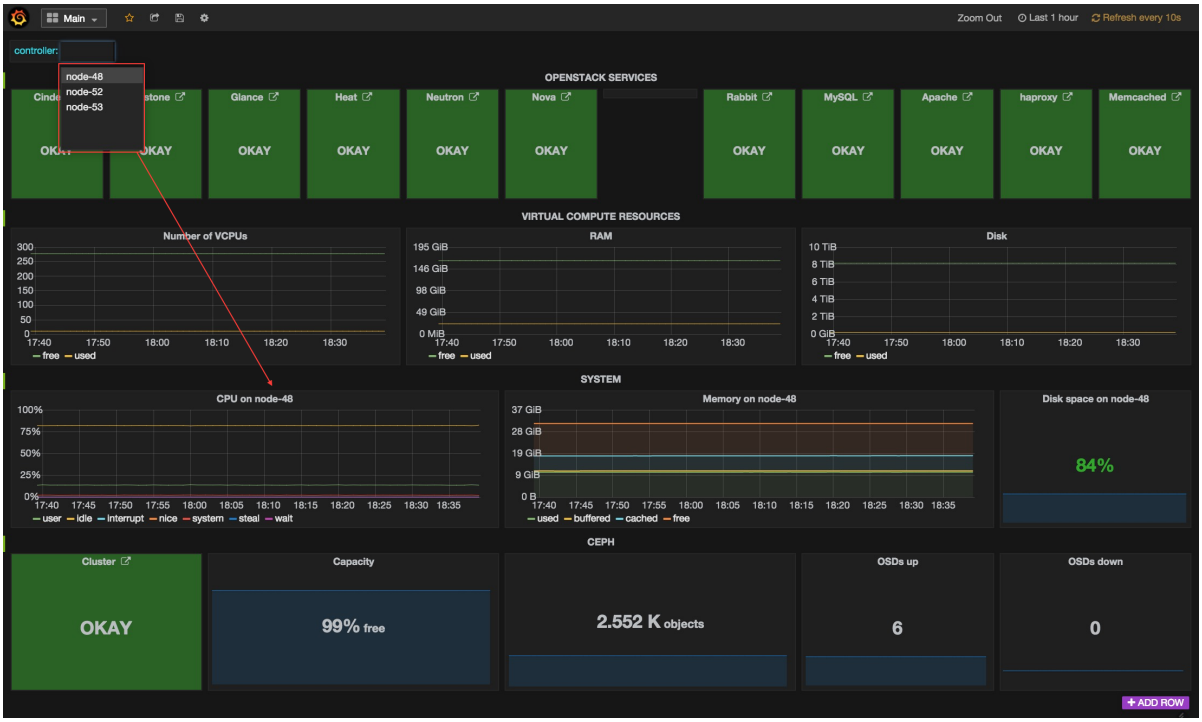
Exploring your time-series with Grafana

The InfluxDB-Grafana Plugin comes with a collection of predefined dashboards you can use to visualize the time-series stored in InfluxDB.

For a complete list of all the metrics time-series that are collected and stored in InfluxDB, see the *List of metrics* section of the StackLight Collector documentation.

The Main dashboard

We recommend that you start with the **Main dashboard**, as shown below, as an entry to other dashboards. The **Main dashboard** provides a single pane of glass from where you can visualize the overall health status of your OpenStack services, such as Nova, Cinder, HAProxy, MySQL, RabbitMQ, and others.



The **Main dashboard**, like most dashboards, provides a drop-down menu in the upper left corner from where you can pick a particular metric dimension, such as the *controller name* or the *device name* you want to select.

In the example above, the dashboard displays the system metrics of *node-48*.

Within the **OpenStack Services** section, each of the services represented can be assigned five different statuses.

Note: The precise determination of a service health status depends on the correlation policies implemented for that service by a Global Status Evaluation (GSE) plugin. See the *Configuring alarms* section in the StackLight Collector documentation.

The service health statuses can be as follows:

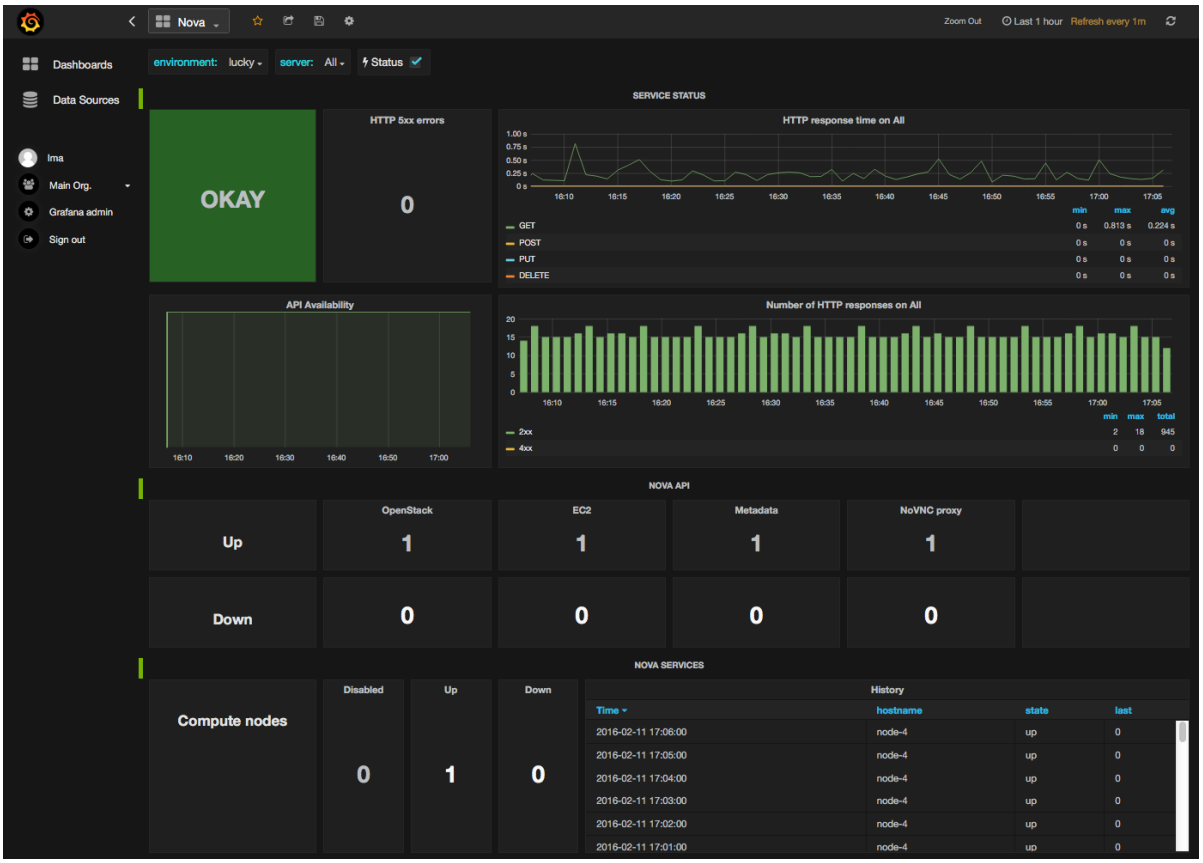
- **Down**: One or several primary functions of a service cluster has failed. For example, all API endpoints of a service cluster like Nova or Cinder failed.
- **Critical**: One or several primary functions of a service cluster are severely degraded. The quality of service delivered to the end user is severely impacted.
- **Warning**: One or several primary functions of a service cluster are slightly degraded. The quality of service delivered to the end user is slightly impacted.
- **Unknown**: There is not enough data to infer the actual health status of a service cluster.
- **Okay**: None of the above was found to be true.

The **Virtual compute resources** section provides an overview of the amount of virtual resources being used by the compute nodes including the number of virtual CPUs, the amount of memory and disk space being used, as well as the amount of virtual resources remaining available to create new instances.

The **System** section provides an overview of the amount of physical resources being used on the control plane (the controller cluster). You can select a specific controller using the controller's drop-down list in the left corner of the toolbar.

The **Ceph** section provides an overview of the resources usage and current health status of the Ceph cluster when it is deployed in the OpenStack environment.

The **Main dashboard** is also an entry point to access more detailed dashboards for each of the OpenStack services that are monitored. For example, if you click the **Nova** box, the **Nova dashboard** is displayed.



The Nova dashboard

The **Nova** dashboard provides a detailed view of the Nova service's related metrics and consists of the following sections:

Service status – information about the Nova service cluster overall health status, including the status of the API front end (the HAProxy public VIP), a counter of HTTP 5xx errors, the HTTP requests response time and status code.

Nova API – information about the current health status of the API back ends, for example, nova-api, ec2-api, and others.

Nova services – information about the current and historical status of the Nova *workers*.

Instances – information about the number of active instances in error and instances creation time statistics.

Resources – various virtual resources usage indicators.

Self-monitoring dashboards

The **Self-Monitoring** dashboard brings operational insights about the overall monitoring system (the toolchain) performance. It provides information about the *hekad* and *collectd* processes. In particular, the **Self-Monitoring** dashboard provides information about the amount of system resources consumed by these processes, the time allocated to the Lua plugins running within *hekad*, the number of messages being processed, and the time it takes to process those messages.

You can select a particular node view using the drop-down menu.

Since StackLight 0.9, there are two new dashboards:

- The **Elasticsearch Cluster** dashboard provides information about the overall health status of the Elasticsearch cluster including the state of the shards, the number of pending tasks, and various resources usage metrics.
- The **InfluxDB Cluster** dashboard provides statistics about the InfluxDB processes running in the InfluxDB cluster including various resources usage metrics.

The hypervisor dashboard

The **Hypervisor** dashboard brings operational insights about the virtual instances managed through *libvirt*. As shown in the figure below, the **Hypervisor** dashboard assembles a view of various *libvirt* metrics. Use the drop-down menu to pick a particular instance UUID running on a particular node. The example below shows the metrics for the instance ID `ba844a75-b9db-4c2f-9cb9-0b083fe03fb7` running on *node-4*.



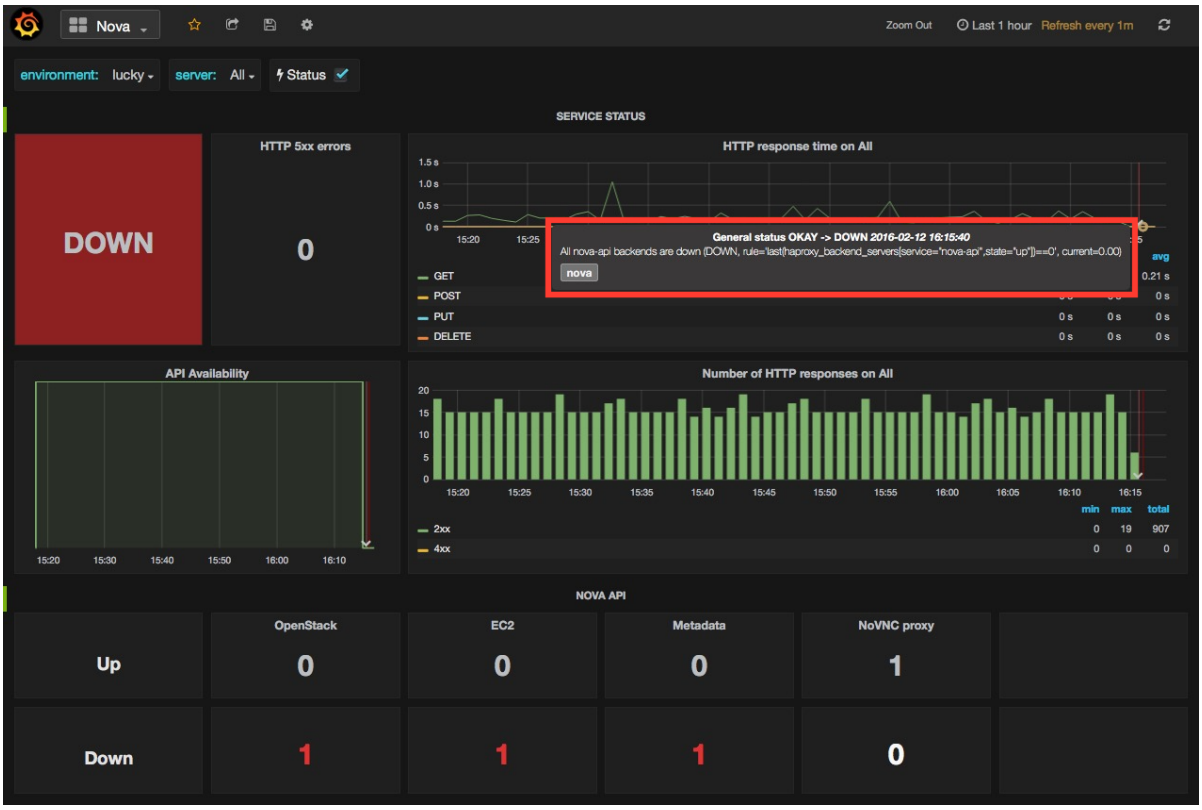
For additional information on the *libvirt* metrics that are displayed in the **Hypervisor** dashboard, see the *List of metrics* section of the StackLight Collector documentation.

Other dashboards

There are 19 different dashboards in total that you can use to explore different time-series facets of your OpenStack environment.

Viewing faults and anomalies

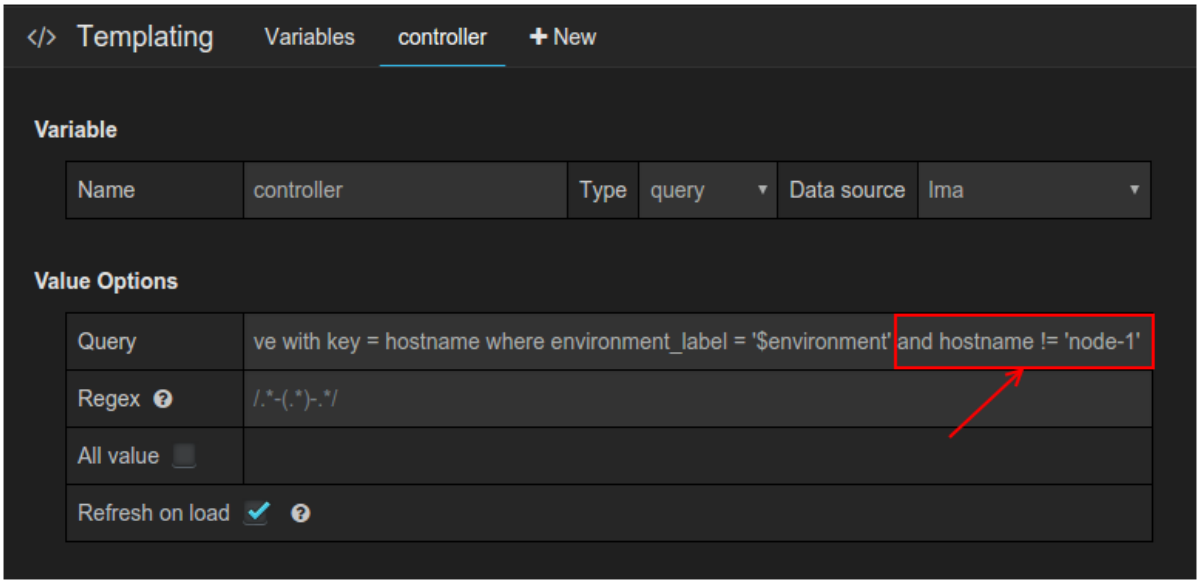
The LMA Toolchain is capable of detecting a number of service-affecting conditions, such as the faults and anomalies that occurred in your OpenStack environment. These conditions are reported in annotations that are displayed in Grafana. The Grafana annotations contain a textual representation of the alarm (or set of alarms) that were triggered by the Collectors for a service. In other words, the annotations contain valuable insights that you can use to diagnose and troubleshoot issues. Furthermore, with the Grafana annotations, the system makes a distinction between what is estimated as a direct root cause versus what is estimated as an indirect root cause. This is internally represented in a dependency graph. There are first degree dependencies used to describe situations, whereby the health status of an entity strictly depends on the health status of another entity. For example, Nova as a service has first-degree dependencies with the nova-api endpoints and the nova-scheduler workers. But there are also second-degree dependencies, whereby the health status of an entity does not strictly depend on the health status of another entity, although it might, depending on other operations being performed. For example, by default, we declared that Nova has a second-degree dependency with Neutron. As a result, the health status of Nova will not be directly impacted by the health status of Neutron, but the annotation will provide a root cause analysis hint. Consider a situation where Nova has changed from *okay* to the *critical* status (because of 5xx HTTP errors) and that Neutron has been in the *down* status for a while. In this case, the Nova dashboard will display an annotation showing that Nova has changed to a *warning* status because the system has detected 5xx errors and that it may be due to the fact that Neutron is *down*. Below is an example of an annotation, which shows that the health status of Nova is *down* because there is no *nova-api* service back end (viewed from HAProxy) that is *up*.



Hiding nodes from dashboards

When you remove a node from the environment, it is still displayed in the *server* and *controller* drop-down lists. To hide it from the list, edit the associated InfluxDB query in the *Templating* section. For example, if you want to remove *node-1*, add the following condition to the *where* clause:

```
and hostname != 'node-1'
```



To hide more than one node, add more conditions. For example:

```
and hostname != 'node-1' and hostname != 'node-2'
```

Perform these actions for all dashboards that display the deleted node and save them afterward.

Troubleshooting

If Grafana contains no data, use the following troubleshooting tips:

1. Verify that the StackLight Collector is running properly by following the troubleshooting instructions in the StackLight Collector Fuel plugin documentation.
2. Verify that the nodes are able to connect to the InfluxDB cluster through the VIP address (See the *Verify InfluxDB* section for instructions on how to get the InfluxDB cluster VIP address) on port *8086*:

   ```
   root@node-2:~# curl -I http://<VIP>:8086/ping
   ```

   The server should return a 204 HTTP status:

   ```
   HTTP/1.1 204 No Content
   Request-Id: cdc3c545-d19d-11e5-b457-000000000000
   X-Influxdb-Version: 0.11.1
   Date: Fri, 12 Feb 2016 15:32:19 GMT
   ```

3. Verify that InfluxDB cluster VIP address is up and running:

   ```
   root@node-1:~# crm resource status vip__influxdb
   resource vip__influxdb is running on: node-1.test.domain.local
   ```

4. Verify that the InfluxDB service is running on all nodes of the cluster:

   ```
   root@node-1:~# service influxdb status
   influxdb Process is running [ OK ]
   ```

5. If the InfluxDB service is not running, restart it:

```
root@node-1:~# service influxdb start
Starting the process influxdb [ OK ]
influxdb process was started [ OK ]
```

6. Verify that the Grafana server is running:

```
root@node-1:~# service grafana-server status
* grafana is running
```

7. If the Grafana server is not running, restart it:

```
root@node-1:~# service grafana-server start
* Starting Grafana Server
```

8. If none of the above solves the issue, look for errors in the following log files:
   - InfluxDB – `/var/log/influxdb/influxdb.log`
   - Grafana – `/var/log/grafana/grafana.log`