# The InfluxDB-Grafana plugin for Fuel Documentation

***Release 0.9-0.9.0-1***

**Mirantis Inc.**

April 22, 2016

CONTENTS

CHAPTER

ONE

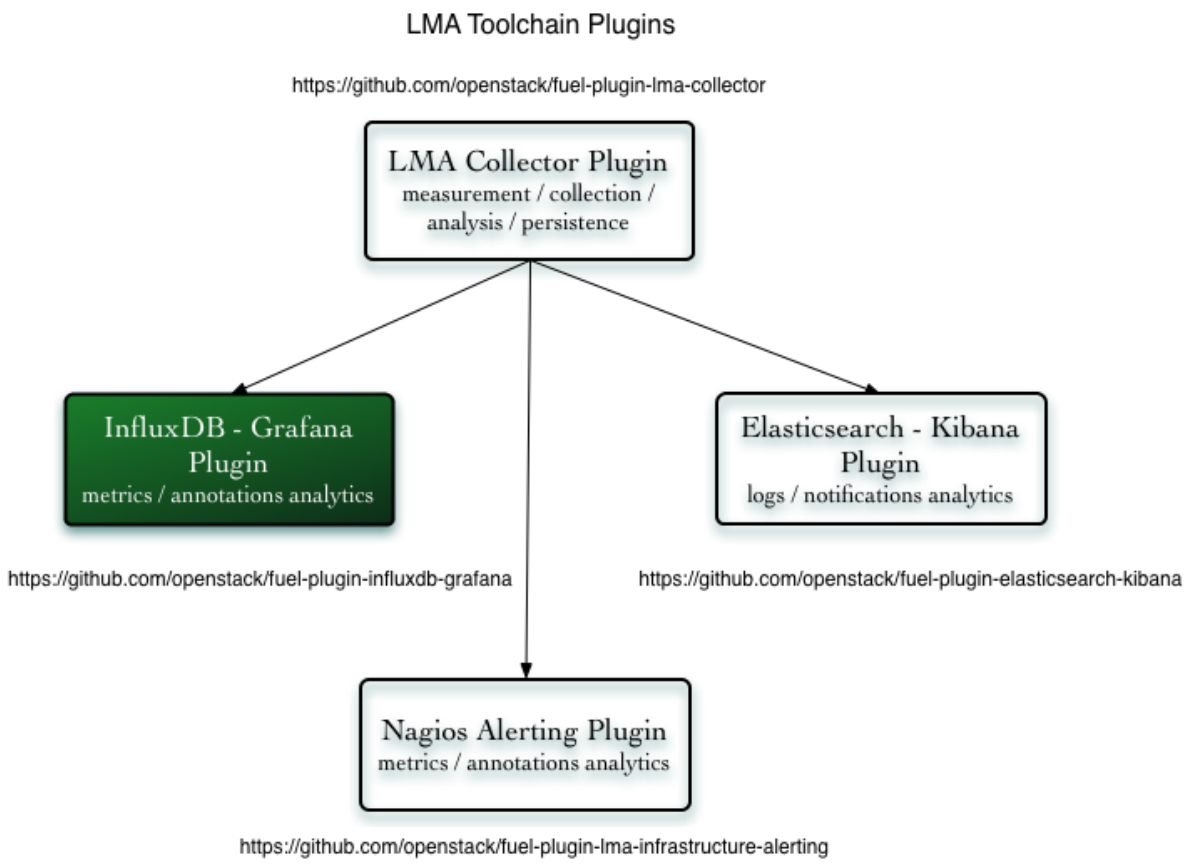
# USER DOCUMENTATION

## 1.1 Overview

The **InfluxDB-Grafana Fuel Plugin** is used to install and configure InfluxDB and Grafana which collectively provide access to the OpenStack metrics analytics. 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 LMA Collector. The annotations contain insightful information about the detected fault or anomaly that triggered a change of state for a node cluster or service cluster as well as textual hints about what might be the root cause of the fault or anomaly.

The InfluxDB-Grafana Plugin is an indispensable tool to answering the questions "what has changed in my 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 key components of the LMA Toolchain project as shown in the figure below.




### 1.1.1 Requirements

| Requirement | Version/Comment |
|---|---|
| Disk space | The plugin's specification requires to provision at least 15GB of disk space for the system, 10GB for the logs and 30GB for the database. The installation of the plugin will fail if there is less than 55GB of disk space available on the node. |
| Mirantis OpenStack | 8.0 |
| 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. Please check the InfluxDB Hardware Sizing Guide for additional sizing information.<br>It is also highly recommended to use a dedicated disk for your data storage. Otherwise, The InfluxDB-Grafana Plugin will use the root filesystem by default. |

### 1.1.2 Limitations

Currently, the size of an InfluxDB cluster the Fuel plugin can deploy is limited to three nodes. In addition to this, 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 using the Fuel plugin, to separate the nodes participating in the Raft consensus cluster from the nodes accessing the data replicas.

### 1.1.3 Key terms, acronyms and abbreviations

| Terms & acronyms | Definition |
|---|---|
| LMA Collector | Logging, Monitoring and Alerting (LMA) Collector. A service running on each node which collects all the logs and the OpenStack notifications. |
| InfluxDB | InfluxDB is a time-series, metrics, and analytics open-source database (MIT license). It's written in Go and has no external dependencies.<br>InfluxDB is targeted at use cases for DevOps, metrics, sensor data, and real-time analytics. |
| Grafana | Grafana is an (Apache 2.0 Licensed) general purpose dashboard and graph composer. It's focused on providing rich ways to visualize metrics time-series, mainly though graphs but supports other ways to visualize data through a pluggable panel architecture.<br>It currently has rich support for Graphite, InfluxDB and OpenTSDB and also supports other data sources via plugins. Grafana is most commonly used for infrastructure monitoring, application monitoring and metric analytics. |

## 1.2 Release Notes

### 1.2.1 Version 0.9.0

- A new dashboard for hypervisor metrics.
- A new dashboard for InfluxDB cluster.
- A new dashboard for Elasticsearch cluster.
- Upgrade to Grafana 2.6.0.
- Upgrade to InfluxDB 0.10.0.
- Add support for InfluxDB clustering (beta state).
- Use MySQL as Grafana backend to support HA.

### 1.2.2 Version 0.8.0

- Add support for the "influxdb_grafana" Fuel Plugin role instead of the "base-os" role which had several limitations.
- Add support for retention policy configuration.

- Upgrade to InfluxDB 0.9.4 which brings metrics time-series with tagging.
- Upgrade to Grafana 2.5.0.
- Several dashboard visualisation improvements.
- A new self-monitoring dashboard.

### 1.2.3 Version 0.7.0

- Initial release of the plugin. This is a beta version.

## 1.3 Installation Guide

### 1.3.1 InfluxDB-Grafana Fuel Plugin installation using the RPM file of the Fuel Plugins Catalog

To install the InfluxDB-Grafana Fuel Plugin using the RPM file of the Fuel Plugins Catalog, you need to follow these steps:

1. Download the RPM file from the Fuel Plugins Catalog.
2. Copy the RPM file to the Fuel Master node:

```
[root@home ~]# scp influxdb_grafana-0.9-0.9.0-1.noarch.rpm \
root@<Fuel Master node IP address>:
```

3. Install the plugin using the Fuel CLI:

```
[root@fuel ~]# fuel plugins --install influxdb_grafana-0.9-0.9.0-1.noarch.rpm
```

4. Verify that the plugin is installed correctly:

```
[root@fuel ~]# fuel plugins --list
id | name                 | version | package_version
---|----------------------|---------|----------------
1  | influxdb_grafana     | 0.9.0   | 4.0.0
```

### 1.3.2 InfluxDB-Grafana Fuel Plugin installtion 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, modify some built-in configuration or implement your own customization. But note that running a Fuel plugin that you have built yourself is at your own risk.

To install the InfluxDB-Grafana Plugin from source, you first need to 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 won't have to copy that file later on.

**Preparing an environment for building the plugin on the Fuel Master Node**

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, you should 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*. In this case, please refer to the section "Preparing an environment for plugin development" of the Fuel Plugins wiki if you need further instructions about how to build the Fuel Plugin Builder.

4. Clone the plugin git repository:

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

5. Check that the plugin is valid:

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

6. And finally, build the plugin:

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

7. Now that you have created the RPM file, you can install the plugin using the *fuel plugins –install* command:

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

### 1.3.3 Software components installed by the InfluxDB-Grafana Plugin

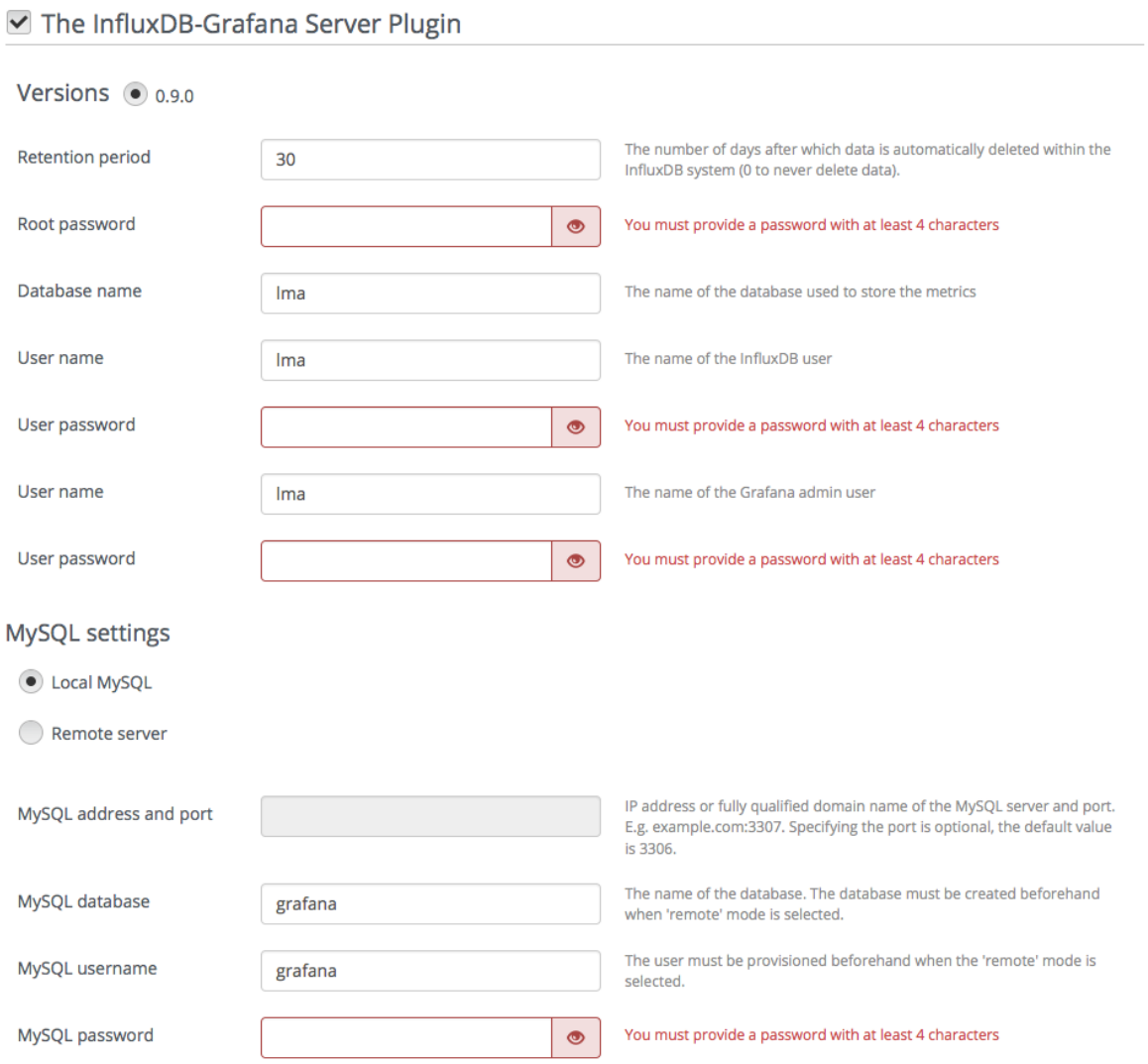
| Components | Version |
|---|---|
| InfluxDB | v0.10.0 for Ubuntu (64-bit) |
| Grafana | v2.6.0 for Ubuntu (64-bit) |

## 1.4 User Guide

### 1.4.1 Plugin configuration

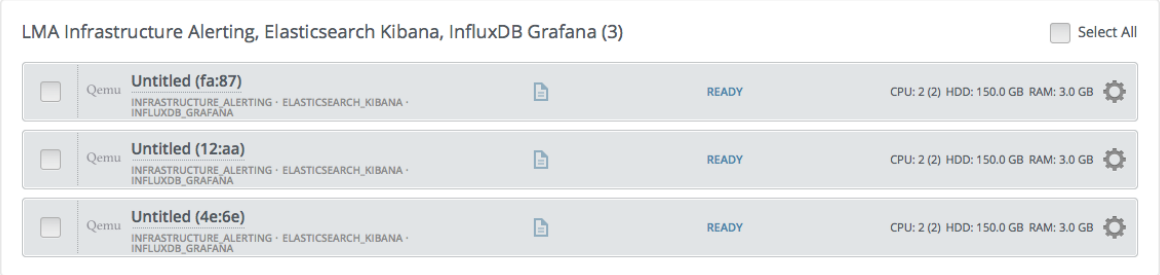
To configure the plugin, you need to follow these steps:

1. Create a new environment from the Fuel web user interface.
2. Click on 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. Tick the **InfluxDB-Grafana Plugin** box 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 (default is lma).
   (d) Specify the InfluxDB username and password.
   (e) Specify the Grafana username and password.
5. With the introduction of Grafana 2.6.0, the plugin now uses a MySQL database to store its configuration 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 IP address of the MySQL server you want to use.
   (b) Then, specify the MySQL database name, username and password that will be used to access that database.
6. Scroll down to the bottom of the page and click the **Save Settings** button when you are done with the settings.

7. Assign the *InfluxDB_Grafana* role to either one node (no HA) or three nodes if you want to run the InfluxDB and Grafana servers in an HA cluster. Note that installing the InfluxDB and Grafana servers on more than three nodes is currently not possible. Similarly, installing the InfluxDB and Grafana servers 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 of the VIP address failover. To be also noted, it is possible to add or remove a node with the *InfluxDB_Grafana* role in the cluster after deployment.




---

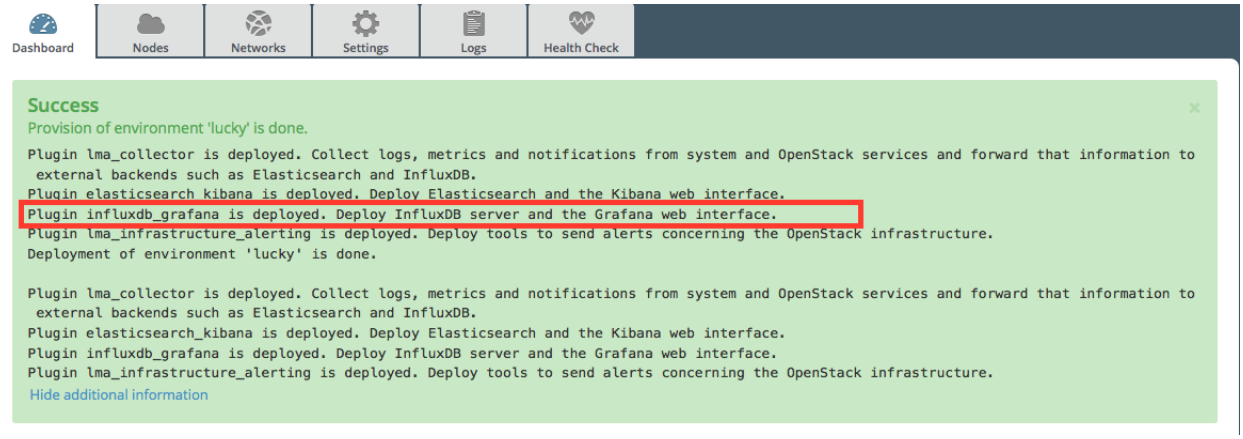
**Note:** You can see in the example above that the *InfluxDB_Grafana* role is assigned to three different nodes along with the *Infrastructure_Alerting* role and the *Elasticsearch_Kibana* role. This means that the three plugins of the LMA toolchain can be installed on the same nodes.

---

8. Click on **Apply Changes**
9. Adjust the disk configuration for your plugin if necessary (see the Fuel User Guide for details). By default, the InfluxDB-Grafana Plugin allocates:
   - 20% of the first available disk for the operating system by honoring a range of 15GB minimum to 50GB maximum.
   - 10GB for */var/log*.
   - At least 30 GB for the InfluxDB database in */var/lib/influxdb*.
10. Configure your environment as needed.
11. Verify the networks.
12. And finally, deploy your changes.

### 1.4.2 Plugin verification

Be aware that depending on the number of nodes and deployment setup, deploying a Mirantis OpenStack environment can typically take anything from 30 minutes to several hours. But once your deployment is complete, you should see a notification message indicating that you deployment is complete as in the figure below.




### Verifying InfluxDB

You should verify that the InfluxDB cluster is running properly. First, you need first to retreive the InfluxDB cluster VIP address. Here is how to proceed.

1. On the Fuel Master node, find the IP address of a node where the InfluxDB server is installed using the following command:

```
[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. Then *ssh* to anyone of these nodes (ex. *node-1*) and type the command:

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

   This tells you that the VIP address of your InfluxDB cluster is *10.109.1.4*.

3. With that VIP address type the 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 0.10.0
InfluxDB shell 0.10.0
>
```

   As you can see, executing */usr/bin/influx* will start an interactive CLI and automatically connect to the InfluxDB server. Then if you type:

```
> show series
```

   You should see a dump of all the time-series collected so far. Then, if you type:

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

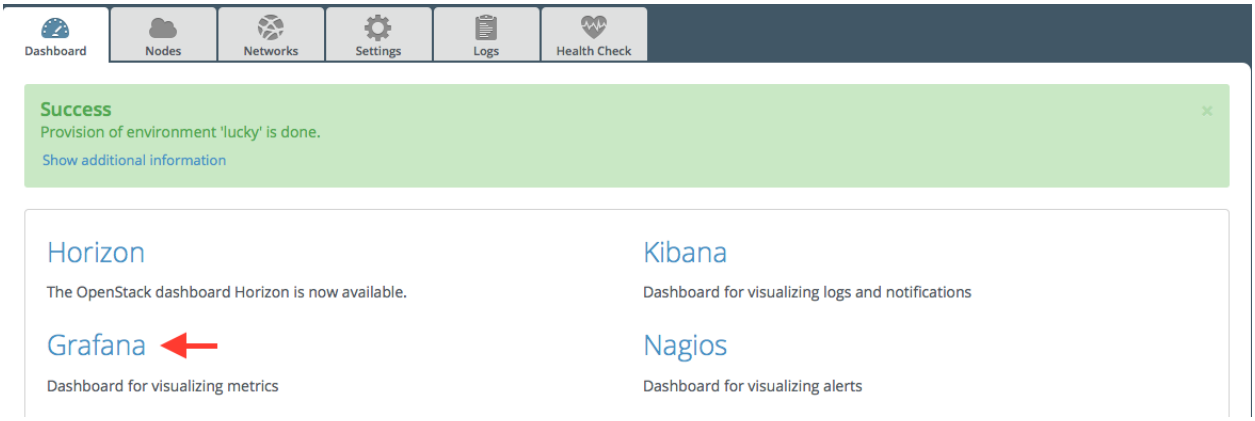
```
3       node-2:8086     node-2:8088
5       node-3:8086     node-3:8088

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
```

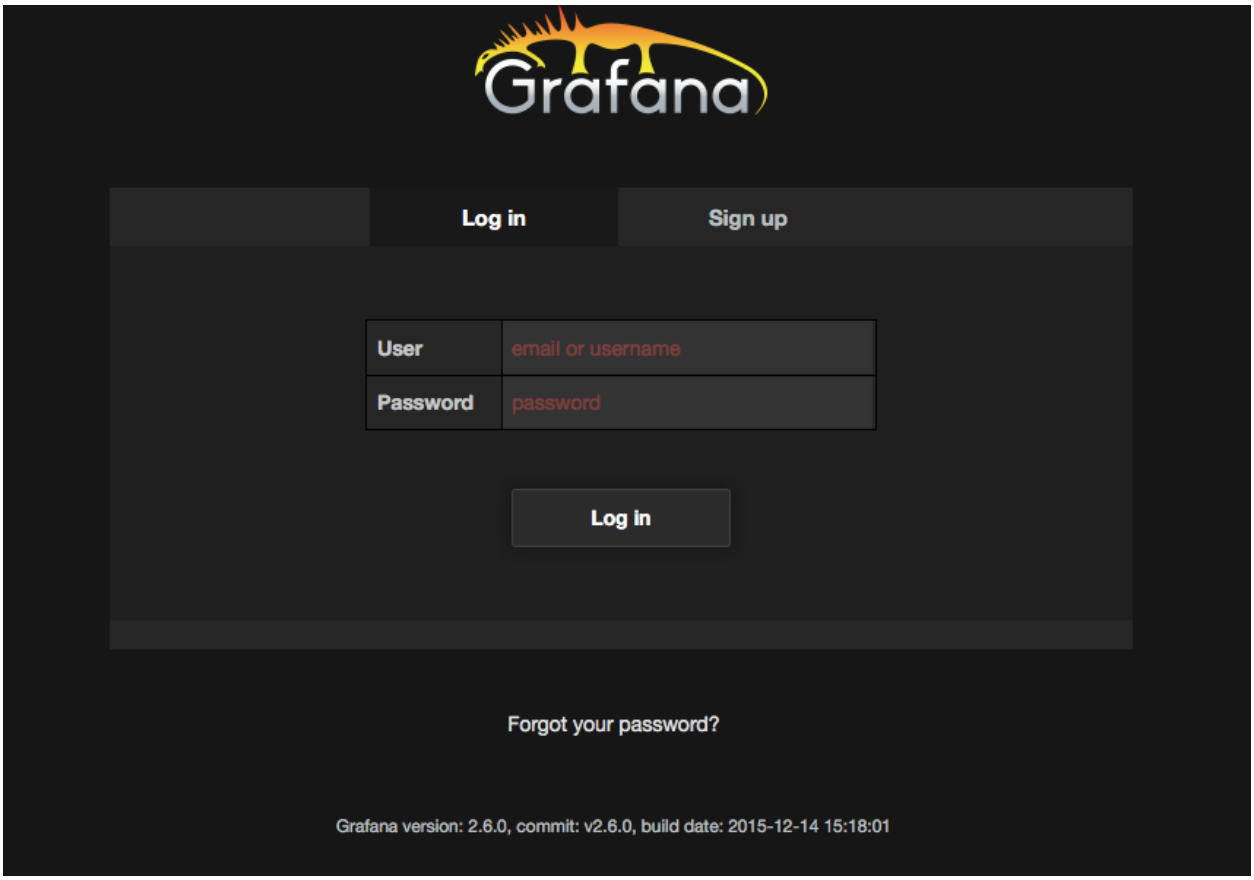
You should see a list of the nodes participating in the InfluxDB cluster with their roles (data or meta).

### Verifying Grafana

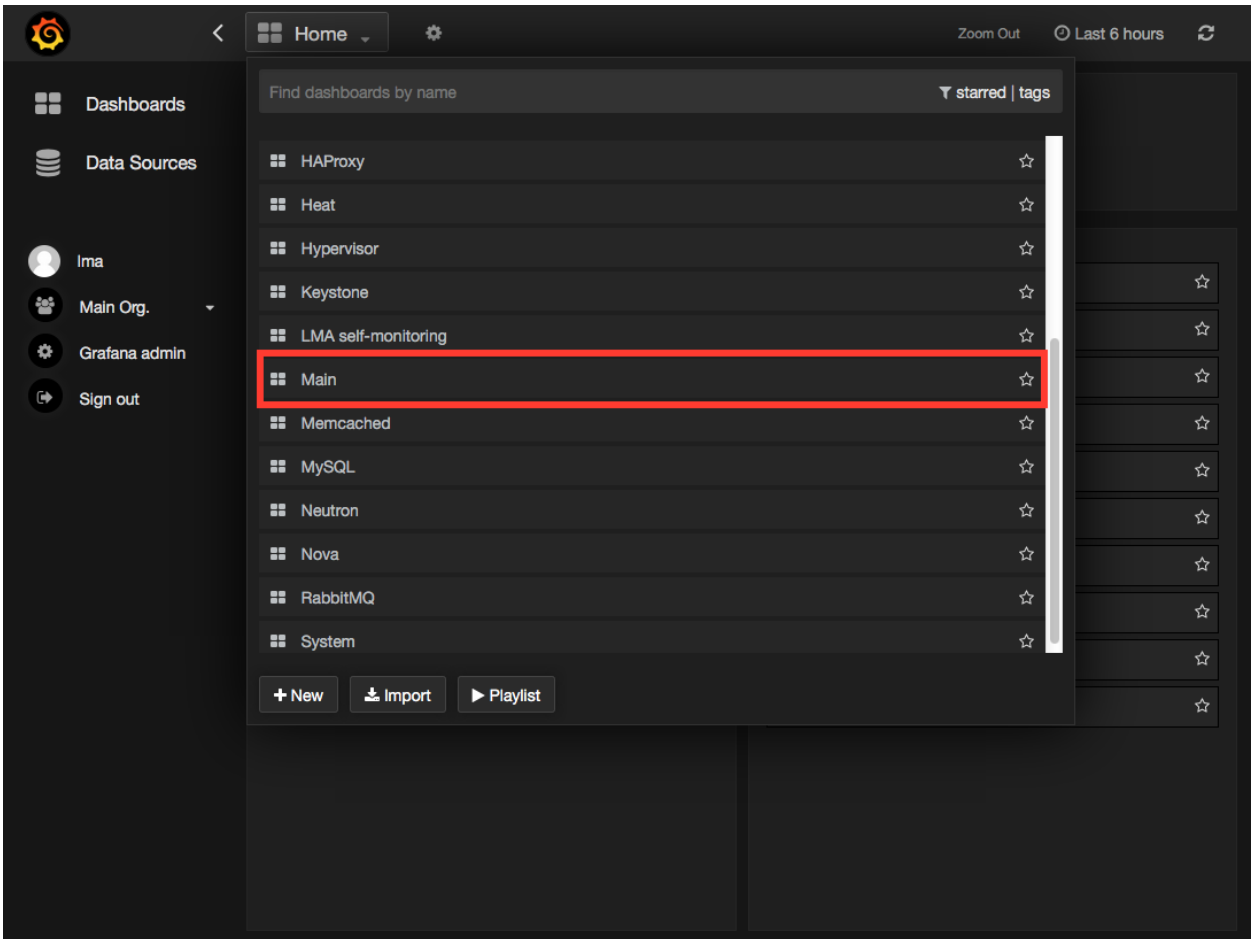
From the Fuel web UI **Dashboard** view, click on the **Grafana** link as shown in the figure below.




The first time you access Grafana, you are requested to authenticate using the credentials you defined in the plugin's settings.




Once you have authenticated, you should be automatically redirected to the **Home Page** from where you can select a dashboard as shown below.




**Note:** Be aware that Grafana is attached to the *management network*. Your desktop machine must have access to the OpenStack environment's *management network* you just created, to get access to the Grafana dashboard.

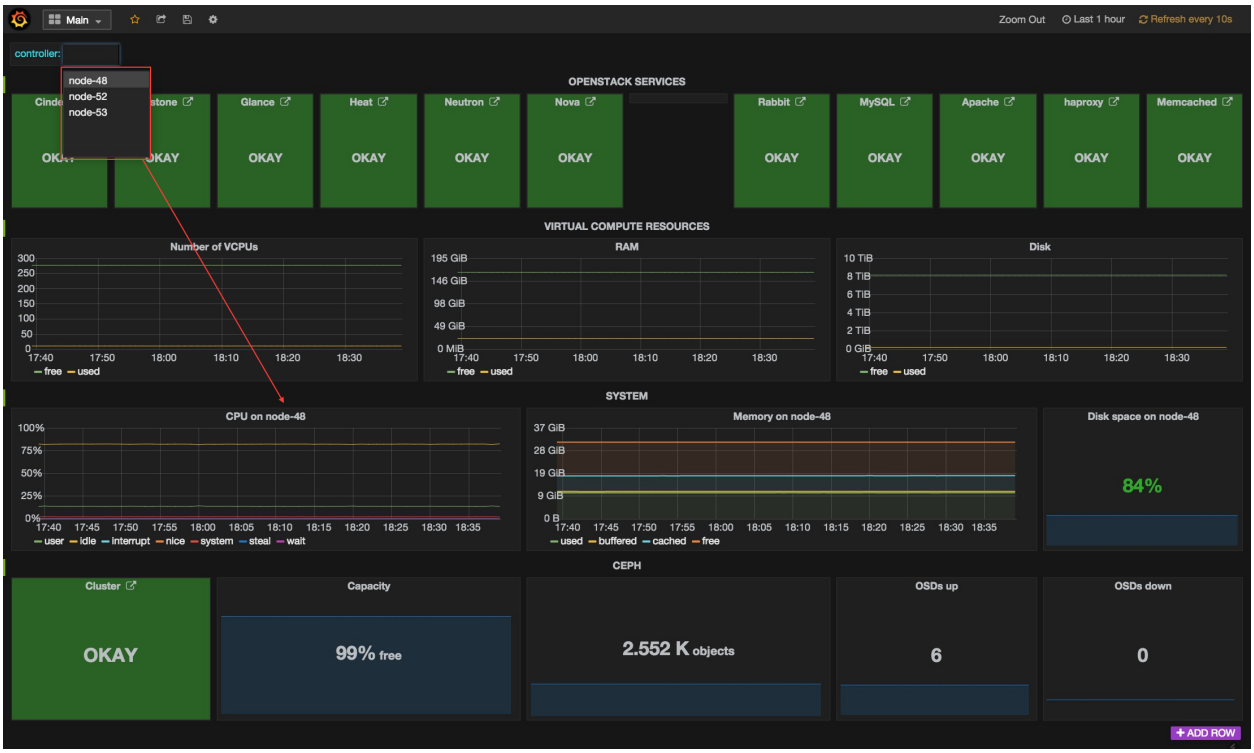
### 1.4.3 Exploring your time-series with Grafana

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

Please check the LMA Collector documentation for a complete list of all the metrics time-series that are collected and stored in InfluxDB.

#### The Main Dashboard

We suggest you start with the **Main Dashboard**, as shown below, as an entry to the other dashboards. The **Main Dashboard** provides a single pane of glass from where you can visualize the overall health state of your OpenStack services such as Nova and Cinder but also HAProxy, MySQL and RabbitMQ to name a few..




As you can see, the **Main Dashboard** (as most dashboards) provides a drop down menu list in the upper left corner of the window 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 system metrics of *node-48* are being displayed in the dashboard.

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

---

**Note:** The precise determination of a service health state depends on the correlation policies implemented for that service by a Global Status Evaluation (GSE) plugin.

---

The meaning associated with a service health state is the following:

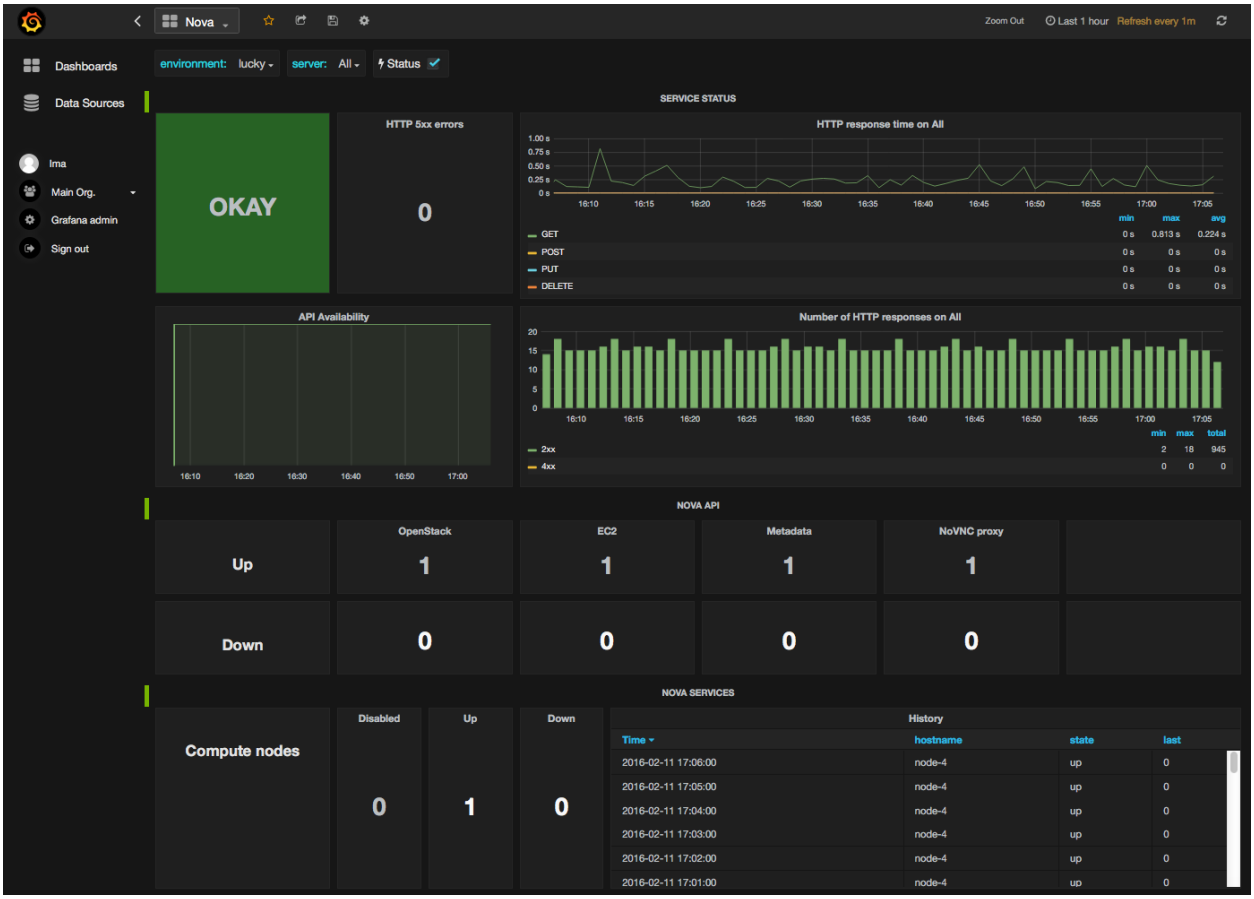
- **Down**: One or several primary functions of a service cluster has failed. For example, all API endpoints of a service cluster like Nova or Cinder are failed.
- **Critical**: One or several primary functions of a service cluster are severely degraded. The quality of service delivered to the end-user should be severely impacted.
- **Warning**: One or several primary functions of a service cluster are slightly degraded. The quality of service delivered to the end-user should be 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** row 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" row 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” row provides an overview of the resources usage and current health state 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 on 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.

The **Service Status** row provides information about the Nova service cluster health state as a whole including the state of the API frontend (the HAProxy public VIP), a counter of HTTP 5xx errors, the HTTP requests response time and status code.

The **Nova API** row provides information about the current health state of the API backends (nova-api, ec2-api, ...).

The **Nova Services** row provides information about the current and historical state of the Nova *workers*.

The **Instances** row provides information about the number of active instances in error and instances creation time statistics.

The **Resources** row provides various virtual resources usage indicators.

## Self-Monitoring Dashboards

The first **Self-Monitoring Dashboard** was introduced in LMA 0.8. The intent of the self-monitoring dashboards is to bring operational insights about how the monitoring system itself (the toolchain) performs overall.

The **Self-Monitoring Dashboard**, provides information about the *hekad* and *collectd* processes. In particular, it gives information about the amount of system resources consumed by these processes, the time allocated to the Lua plugins running within *hekad*, the amount of messages being processed and the time it takes to process those messages.

Again, it is possible to select a particular node view using the drop down menu list.

With LMA 0.9, we have introduced two new dashboards.

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

### The Hypervisor Dashboard

LMA 0.9 introduces a new **Hypervisor Dashboard** which 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. A dropdown menu list allows to pick a particular instance UUID running on a particular node. In the example below, the metrics for the instance id *ba844a75-b9db-4c2f-9cb9-0b083fe03fb7* running on *node-4* are displayed.

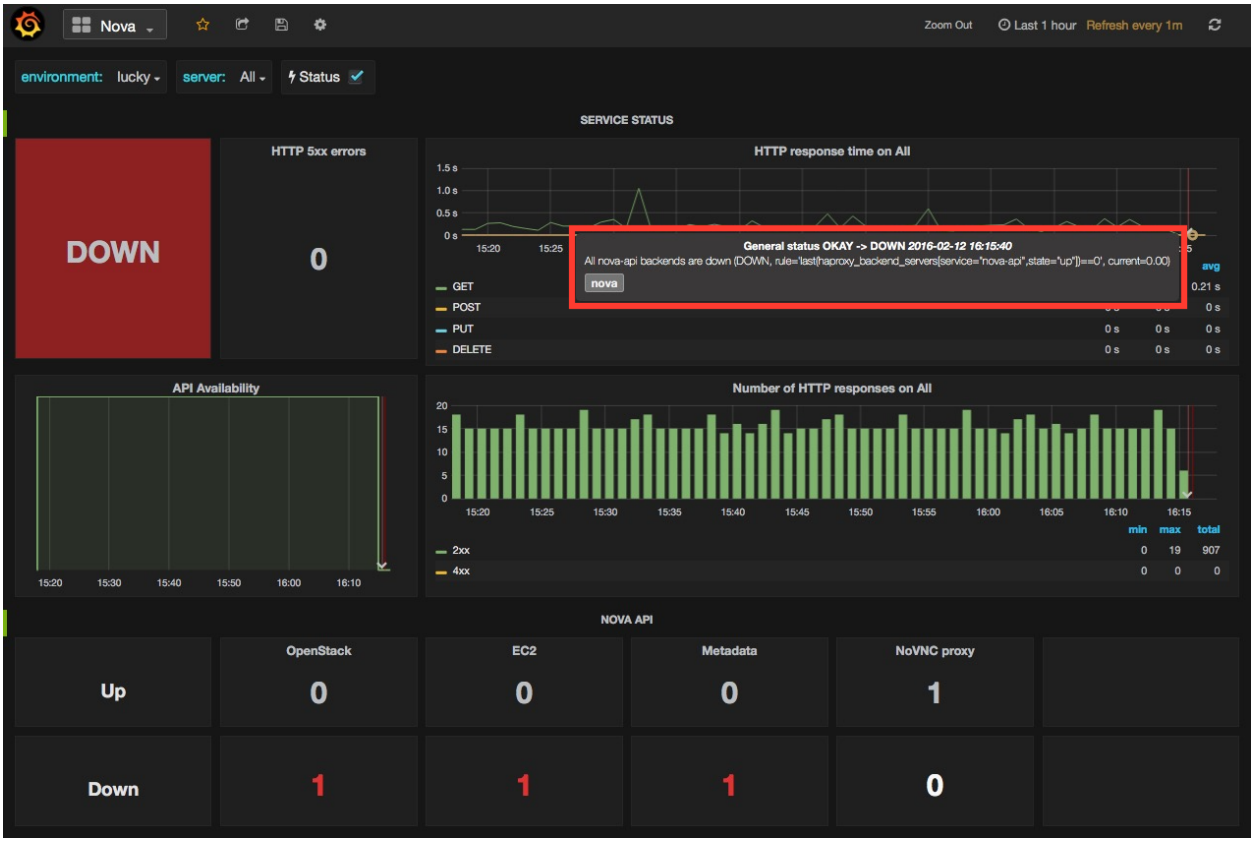

Check the LMA Collector documentation for additional information about the *libvirt* metrics that are displayed in the **Hypervisor Dashboard**.

### Other Dashboards

In total there are 19 different dashboards 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 occured in your OpenStack environment. Those 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 could use to diagnose and troubleshoot problems. 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 state of an entity strictly depends on the health state 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 state of an entity doesn't strictly depends on the health state 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 state of Nova will not be directly impacted by the health state of Neutron but the annotation will provide a root cause analysis hint. Let's assume a situation where Nova has changed from *okay* to *critical* state (because of 5xx HTTP errors) and that Neutron has been in *down* state for a while. In this case, the Nova dashboard will display an annotation showing that Nova has changed to a *warning* state because the system has detected 5xx errors and that it may be due to the fact that Neutron is *down*. An example of what an annotation looks like is shown below.

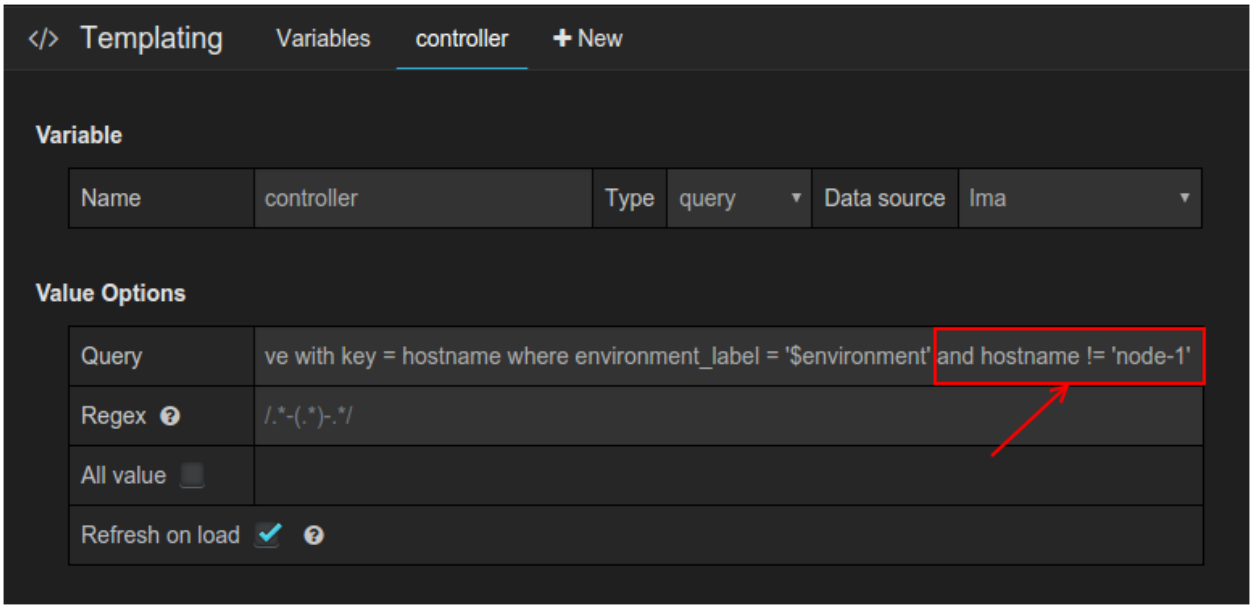



This annotation shows that the health state of Nova is *down* because there is no *nova-api* service backend (viewed from HAProxy) that is *up*.

### 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 you need to edit the associated InfluxDB query in the *templating* section. For example, if you want to remove *node-1*, you need to add the following condition to the *where* clause:

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




If you want to hide more than one node you can add more conditions like this:

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

This should be done for all dashboards that display the deleted node and you need to save them afterwards.

## 1.4.4 Troubleshooting

If you get no data in Grafana, follow these troubleshooting tips.

1. First, check that the LMA Collector is running properly by following the LMA Collector troubleshooting instructions in the LMA Collector Fuel Plugin User Guide.
2. Check that the nodes are able to connect to the InfluxDB cluster via the VIP address (see above 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: 0.10.0
   Date: Fri, 12 Feb 2016 15:32:19 GMT
   ```

3. Check 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. Check that the InfluxDB service is started on all nodes of the cluster:

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

5. If not, (re)start it:

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

6. Check that Grafana server is running:

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

7. If not, (re)start it:

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

8. If none of the above solves the problem, check the logs in `/var/log/influxdb/influxdb.log` and `/var/log/grafana/grafana.log` to find out what might have gone wrong.

## 1.5 Licenses

### 1.5.1 Third Party Components

| Name | Project Web Site | License |
|---|---|---|
| InfluxDB | https://influxdb.com/ | MIT |
| Grafana | http://grafana.org/ | Apache v2 |

### 1.5.2 Puppet modules

| Name | Project Web Site | 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 |

## 1.6 Appendix

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

CHAPTER

TWO

# INDICES AND TABLES

- search