
The StackLight Infrastructure Alerting Plugin for Fuel Documentation

Release 1.0.0

Mirantis Inc.

February 14, 2017

CONTENTS

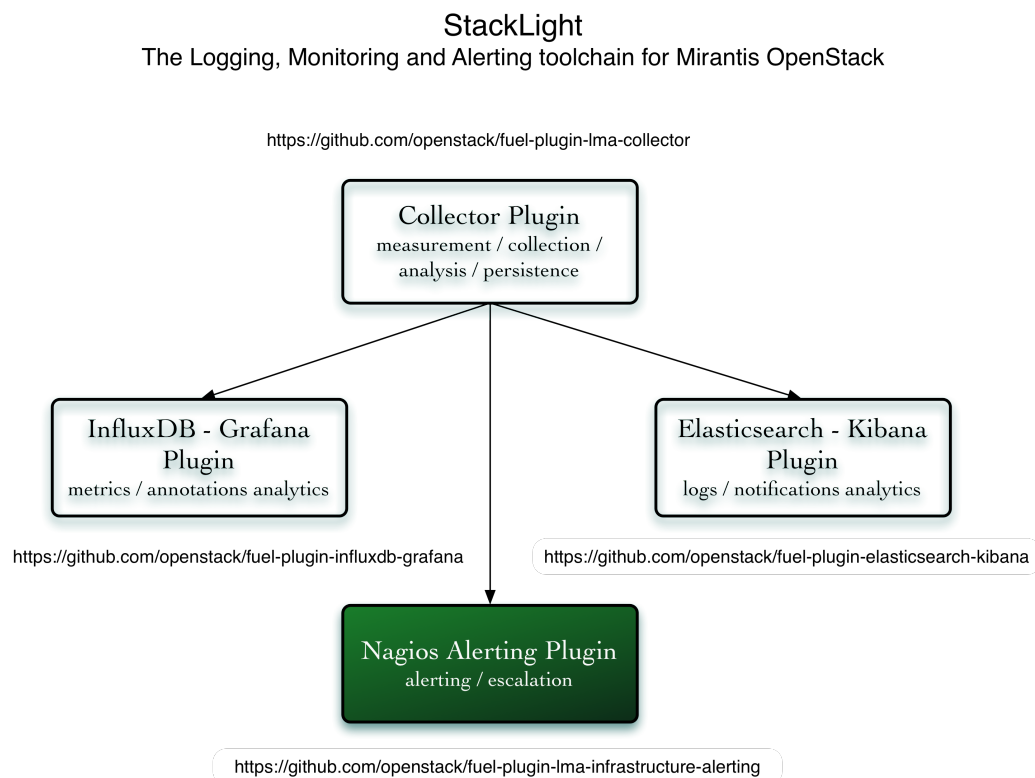
1	Overview	1
1.1	Introduction	1
1.2	Requirements	2
1.3	Limitations	2
1.4	Release notes	2
1.5	Licenses	3
1.6	References	3
2	Installing and configuring StackLight Infrastructure Alerting plugin for Fuel	4
2.1	Introduction	4
2.2	Install using the RPM file	4
2.3	Install from source	5
2.4	Plugin configuration	6
2.5	Plugin verification	9
3	Using StackLight Infrastructure Alerting plugin for Fuel	10
3.1	Using Nagios	10
3.2	Configuring service checks using the InfluxDB metrics	11
3.3	Using an external SMTP server with STARTTLS	13
3.4	Troubleshooting	14

OVERVIEW

1.1 Introduction

The **StackLight Infrastructure Alerting plugin** is used to install and configure Nagios, which provides the alerting and escalation functionality of the so-called Logging, Monitoring, and Alerting Toolchain of Mirantis OpenStack.

Nagios is a key component of the [LMA Toolchain project](#) of Mirantis OpenStack, as shown in the figure below.



1.2 Requirements

The StackLight Infrastructure Alerting plugin 1.0.0 has the following 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 20 GB for Nagios. Therefore, the installation of the plugin will fail if there is less than 45 GB of disk space available on the node.
Hardware configuration	The hardware configuration (RAM, CPU, disk) required by this plugin depends on the size of your cloud environment and other parameters like the retention period of the data. A typical setup would at least require a quad-core server with 8 GB of RAM and fast disks (ideally, SSDs).
Mirantis OpenStack	8.0, 9.x
The StackLight Collector Plugin	1.0
The StackLight InfluxDB Grafana Plugin	1.0 This is optional and only needed if you want to create alarms in Nagios for time-series stored in InfluxDB.

1.3 Limitations

The StackLight Infrastructure Alerting plugin 1.0.0 has the following limitation:

- If Nagios is installed on several nodes for high availability, the alerts history will be lost in case of a server failover.

1.4 Release notes

1.4.1 Version 1.0.0

The StackLight Infrastructure Alerting plugin 1.0.0 contains the following updates:

- Fixed an issue to allow the configuration of a list of LDAP servers. See [#1624002](#).
- Modified the cron job to use a specific version of the plugin. See [#1622628](#).
- Added support for wildcard SSL certificates. See [#1608665](#).
- Fixed the UI issue with the LDAP protocol radio button. See [#1599778](#).
- Fixed wrong log rotation of Nagios WSGI application logs. See [#1635222](#).
- Fixed wrong Email address for Nagios hosts. See [#1650543](#).

1.4.2 Version 0.10.0

The StackLight Infrastructure Alerting plugin 0.10.0 contains the following updates:

- Added support for LDAP(S) authentication to access the Nagios web UI. The *nagiosadmin* user is still created statically and is the only user who has the *admin* privileges by default.
- Added Support for TLS encryption to access the Nagios web UI. A PEM file (obtained by concatenating the SSL certificates with the private key) must be provided in the settings of the plugin to configure the TLS termination.

- Bug fixes:
 - Fixed the issue with Apache that could not handle the passive checks workload for large deployments. See [#1552772](#).

1.4.3 Version 0.9.0

The StackLight Infrastructure Alerting plugin 0.9.0 contains the following updates:

- Added support for Nagios clustering for high availability.
- Bug fixes:
 - Specified contact_groups for HTTP checks. See [#1559151](#).
 - Specified contact_groups for SSH checks. See [#1559153](#).

1.4.4 Version 0.8.0

The initial release of the plugin.

1.5 Licenses

1.5.1 Third-party components

Name	Project website	License
Nagios	https://www.nagios.org/	GPLv2
Apache HTTP server	http://httpd.apache.org	Apache v2

1.5.2 Puppet modules

Name	Project website	License
puppetlabs-apache	https://github.com/puppetlabs/puppetlabs-apache	Apache v2
puppetlabs-concat	https://github.com/puppetlabs/puppetlabs-concat	Apache v2
puppetlabs-stdlib	https://github.com/puppetlabs/puppetlabs-stdlib	Apache v2
leinaddm-htpasswd	https://github.com/leinaddm/puppet-htpasswd	Apache v2

1.6 References

- The StackLight Infrastructure Alerting Plugin project at GitHub
- The StackLight Collector Plugin project at GitHub
- The StackLight InfluxDB-Grafana Plugin project at GitHub
- The official Nagios documentation

INSTALLING AND CONFIGURING STACKLIGHT INFRASTRUCTURE ALERTING PLUGIN FOR FUEL

2.1 Introduction

You can install the StackLight Infrastructure Alerting 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 Infrastructure Alerting plugin:

Component	Version
Nagios	v3.5.1 for Ubuntu (64-bit)
Apache	Version coming with the Ubuntu distribution

2.2 Install using the RPM file

To install the StackLight Infrastructure Alerting 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 lma_infrastructure_alerting-1.0-1.0.0-0.noarch.rpm \  
root@<Fuel Master node IP address>:
```

5. Install the plugin using the [Fuel Plugins CLI](#):

```
[root@fuel ~]# fuel plugins --install \  
lma_infrastructure_alerting-1.0-1.0.0-0.noarch.rpm
```

6. Verify that the plugin is installed correctly:

```
[root@fuel ~]# fuel plugins --list  
id | name | version | package_version  
---|-----|-----|-----  
1 | lma_infrastructure_alerting | 1.0.0 | 4.0.0
```

2.3 Install from source

Alternatively, you may want to build the plugin RPM file 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 Infrastructure Alerting 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 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-lma-infrastructure-alerting.git
```

5. Verify that the plugin is valid:

```
[root@home ~] fpb --check ./fuel-plugin-lma-infrastructure-alerting
```

6. Build the plugin:

```
[root@home ~] fpb --build ./fuel-plugin-lma-infrastructure-alerting
```

To install the plugin:

1. Once you have created the RPM file, install the plugin:

```
[root@fuel ~] fuel plugins --install ./fuel-plugin-lma-infrastructure-alerting/*.rpm
```

2. Verify that the plugin is installed correctly:

```
[root@fuel ~]# fuel plugins --list
id | name | version | package_version
---|-----|-----|-----
1  | lma_infrastructure_alerting | 1.0.0 | 4.0.0
```


2.4 Plugin configuration

To configure the StackLight Infrastructure Alerting 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 StackLight Infrastructure Alerting Plugin* section.
4. Select *The StackLight Infrastructure Alerting Plugin* and fill in the required fields as indicated below.

☒ The StackLight Infrastructure Alerting Plugin ⚠

Versions ☒ 0.10.0

Nagios HTTP password  The password to access the Nagios Web Interface (username: "nagiosadmin")

☒ Receive CRITICAL notifications by email

☒ Receive WARNING notifications by email

☒ Receive UNKNOWN notifications by email

☒ Receive RECOVERY notifications by email

The recipient email address The recipient for the alert notifications

The sender email address

SMTP server IP and port ie: 10.2.2.3:25

SMTP authentication method


☒ None

☐ Login

☐ Plain

☐ CRAMMD5

SMTP user

SMTP password 

- (a) If required, override the Nagios web interface self-generated password.
 - (b) Select the types of notifications that you would like to be alerted for by email (*CRITICAL*, *WARNING*, *UNKNOWN*, *RECOVERY*).
 - (c) Specify the recipient email address for the alerts.
 - (d) Specify the sender email address for the alerts.
 - (e) Specify the SMTP server address and port.
 - (f) Specify the SMTP authentication method.
 - (g) Specify the SMTP username and password. This is not required if the authentication method is *None*.
5. Select *Enable TLS for Nagios* if you want to encrypt your Nagios web UI credentials (username, password). Then, fill in the required fields as indicated below.

☒ Enable TLS for Nagios

DNS hostname for Nagios UI Your DNS entries should point to this name

Certificate for Nagios UI  Certificate and private key data, concatenated into a single file

- (a) Specify the DNS name of the Nagios web UI. This parameter is used to create a link from within the Fuel dashboard to the Nagios web UI.
 - (b) Specify the location of the PEM file, which contains the certificate and the private key of the server that will be used in TLS handchecks with the client.
6. Select *Use LDAP for Nagios Authentication* if you want to authenticate through LDAP to the Nagios Web UI. Then, fill in the required fields as indicated below.

☒ Use LDAP for Nagios authentication

LDAP protocol

☒ LDAP

☐ LDAPS

LDAP servers Specify one or several LDAP servers separated by space.

Port If empty, the default value is 389 for LDAP and 636 for LDAPS.

Bind DN DN used to bind to the server when searching for entries.

Bind password  Password to use in conjunction with the bind DN.

User search base DN The base DN to search for users.

User attribute to search for It's a good idea to choose an attribute that will be unique across all entries.

User search filter A valid LDAP search filter.

☒ Enable group-based authorization
It allows to authorized only users for a specific group.

LDAP group attribute LDAP attribute used to identify the user members of group.

Group DN mapping to the Admins role

- (a) Select the *LDAPS* if you want to enable LDAP authentication over SSL.
- (b) Specify one or several LDAP server addresses separated by a space. These addresses must be accessible from the node where Nagios 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 *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* to search for users. The search should return a unique user entry.

You can further restrict access to the Nagios web UI to those users who are members of a specific LDAP group. However, with the Nagios web UI there is no notion of privileged (admin) access.

- (a) Select *Enable group-based authorization* to restrict the access to a group of users.
- (b) Specify the LDAP attribute in the user entry to identify the group of users.
- (c) Specify the DN of the LDAP group that has access to the Nagios web UI.

7. 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 a dedicated network for StackLight should improve performance and reduce the monitoring footprint. It will also facilitate access to the Nagios web UI after deployment.

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

The example below shows that the *Infrastructure_Alerting* role is assigned to three nodes alongside with the *Elasticsearch_Kibana* role and the *InfluxDB_Grafana* role. The three plugins of the LMA toolchain back-end servers are installed on the same node.

StackLight Infrastructure Alerting, Elasticsearch Kibana, InfluxDB Grafana (3)						<input type="checkbox"/> Select All
<input type="checkbox"/>	stacklight1 SUPER MICRO INFRASTRUCTURE_ALERTING · ELASTICSEARCH_KIBANA · INFLUXDB_GRAFANA		READY	CPU: 1 (12) RAM: 64.0 GB HDD: 1.8 TB		
<input type="checkbox"/>	stacklight2 SUPER MICRO INFRASTRUCTURE_ALERTING · ELASTICSEARCH_KIBANA · INFLUXDB_GRAFANA		READY	CPU: 1 (12) RAM: 64.0 GB HDD: 1.8 TB		
<input type="checkbox"/>	stacklight3 SUPER MICRO INFRASTRUCTURE_ALERTING · ELASTICSEARCH_KIBANA · INFLUXDB_GRAFANA		READY	CPU: 1 (12) RAM: 64.0 GB HDD: 1.8 TB		

Note: Nagios clustering for high availability requires assigning the *Infrastructure_Alerting* role to three different nodes. You can add or remove nodes with the *Infrastructure_Alerting* role after deployment.

9. If required, adjust the disk partitioning as described in [Configure disk partitioning](#).

By default, the StackLight Infrastructure Alerting plugin allocates:

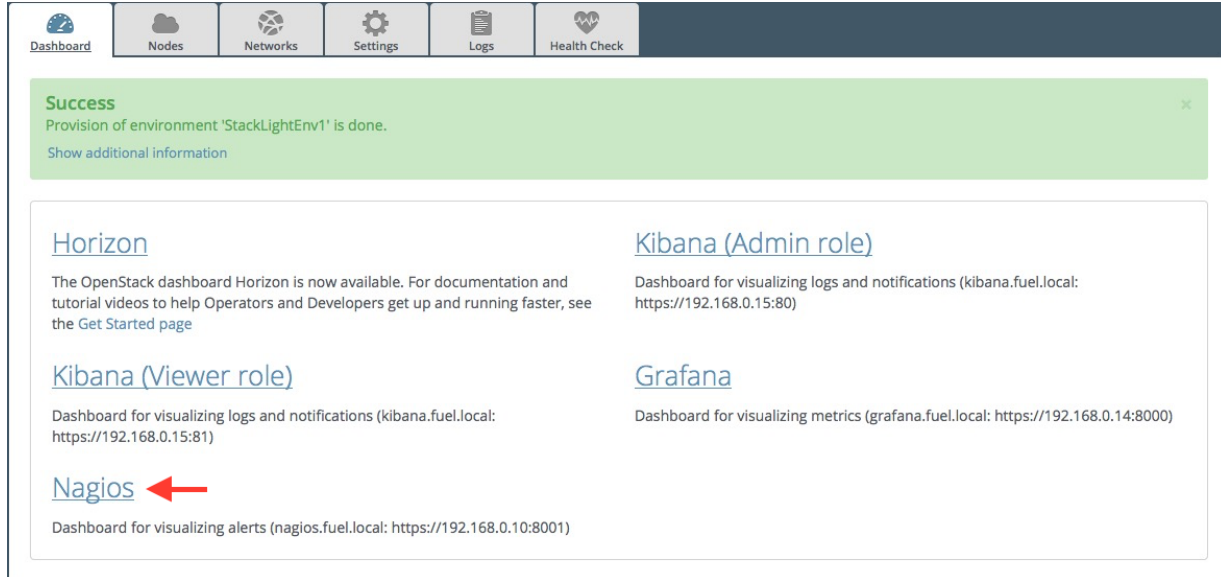
- 20% of the first available disk for the operating system by honoring a range of 15 GB minimum and 50 GB maximum
- 10 GB for `/var/log`
- At least 20 GB for the Nagios data in `/var/nagios`

The deployment will fail if the above requirements are not met.

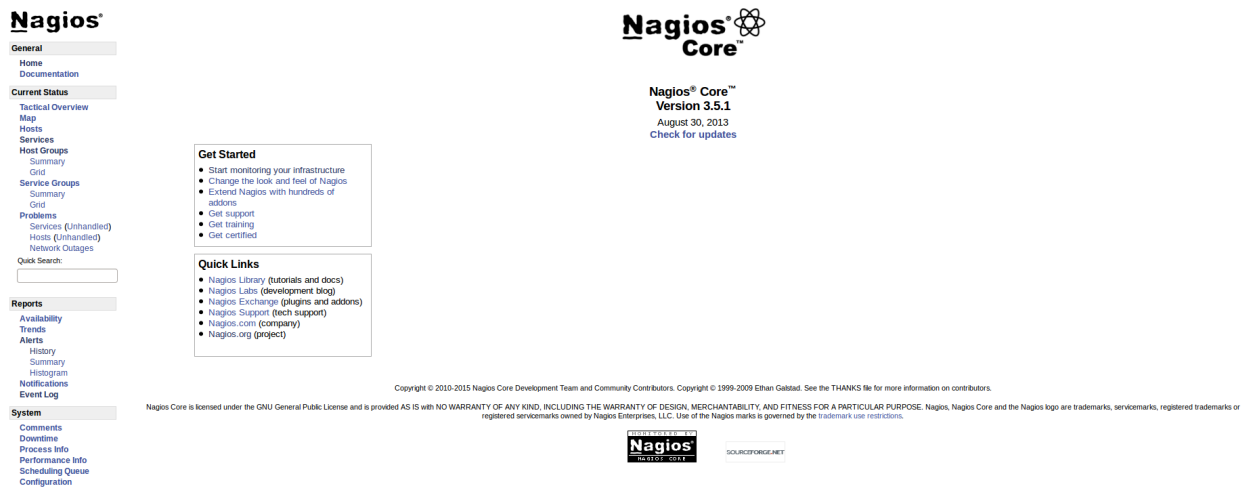
10. Deploy your environment as described in [Deploy an OpenStack environment](#).

2.5 Plugin verification

Depending on the number of nodes and deployment setup, deploying a Mirantis OpenStack environment may take 20 minutes to several hours. Once the deployment is complete, you should see a deployment success notification message with a link to the Nagios web UI as shown below.



Click *Nagios*. Once authenticated, you should be redirected to the Nagios home page as shown below.



Note: The username is `nagiosadmin` by default, the password is defined in the settings.

Note: If Nagios is installed on the *management network*, you may not have direct access to the Nagios web UI. Extra network configuration may be required to create an SSH tunnel to the *management network*.

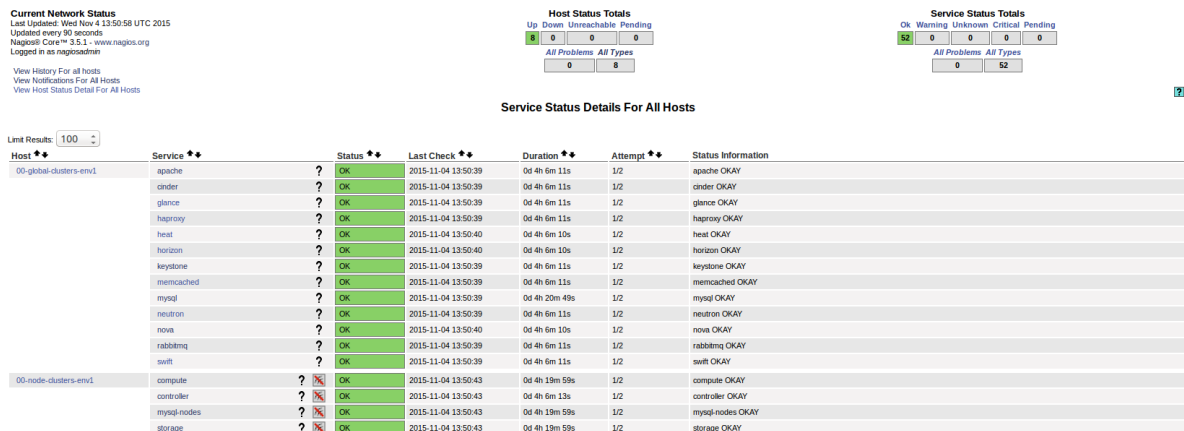
USING STACKLIGHT INFRASTRUCTURE ALERTING PLUGIN FOR FUEL

3.1 Using Nagios

The StackLight Infrastructure Alerting plugin configures Nagios to display the health status of all the nodes and services running in the OpenStack environment. The alarms, or service checks in Nagios terms, are created in **passive mode**, which means that the actual checks are not performed by Nagios itself, but by the Collector and Aggregator agents of the LMA toolchain.

To get an overview of your OpenStack environment:

1. Log in to the Fuel web UI.
2. Click *Dashboard*.
3. Click *Nagios*.
4. Click the *Services* link in the left panel of the Nagios web UI. You should see the following page:



In this dashboard, there are two virtual hosts representing the health status of the so-called **global clusters** and **node clusters** entities:

- *00-global-clusters-env\${ENVID}* is used to represent the aggregated health status of global clusters, such as ‘Nova’, ‘Keystone’, ‘RabbitMQ’, and others.
- *00-node-clusters-env\${ENVID}* is used to represent the aggregated health status of node clusters, such as ‘Controller’, ‘Compute’, and ‘Storage’.

The virtual hosts section contains a list of checks received for each of the nodes provisioned in the environment. These checks may vary depending on the role of the node being monitored.

Alerting is enabled by default for the global cluster entities. For the nodes and clusters of nodes alerting is disabled by default to avoid the alert fatigue, since these alerts should not be representative of a critical condition affecting the overall health status of the global cluster entities.

To enable alerting for nodes and clusters:

1. Click a particular service.
2. Click the *Enable notifications for this service* link within the *Service Commands* panel as shown below.

Service Information
 Last Updated: Thu Feb 18 17:36:09 UTC 2016
 Updated every 90 seconds
 Nagios® Core™ 3.5.1 - www.nagios.org
 Logged in as nagiosadmin

View Information For This Host
 View Status Detail For This Host
 View Alert History For This Service
 View Trends For This Service
 View Alert Histogram For This Service
 View Availability Report For This Service
 View Notifications For This Service

Service
compute
 On Host
00-node-clusters-env1
 (00-node-clusters-env1)

Member of
 No servicegroups.

10.109.1.3

Service State Information

Current Status: **OK** (for 7d 12h 46m 36s)
 Status Information: compute OKAY
 no details

Performance Data:
 Current Attempt: 1/2 (HARD state)
 Last Check Time: 2016-02-18 17:35:42
 Check Type: PASSIVE
 Check Latency / Duration: N/A / 0.000 seconds
 Next Scheduled Check: N/A
 Last State Change: 2016-02-11 04:49:33
 Last Notification: N/A (notification 0)
 Is This Service Flapping? **NO** (0.00% state change)
 In Scheduled Downtime? **NO**
 Last Update: 2016-02-18 17:36:01 (0d 0h 0m 8s ago)

Active Checks: **DISABLED**
 Passive Checks: **ENABLED**
 Obsessing: **ENABLED**
 Notifications: **DISABLED**
 Event Handler: **ENABLED**
 Flap Detection: **ENABLED**

Service Commands

- ✓ Enable active checks of this service
- ⌚ Re-schedule the next check of this service
- ? Submit passive check result for this service
- ✗ Stop accepting passive checks for this service
- ✗ Stop obsessing over this service
- ✓ Enable notifications for this service
- 📧 Send custom service notification
- 🕒 Schedule downtime for this service
- ✗ Disable event handler for this service
- ✗ Disable flap detection for this service

There is a direct dependency between the configuration of the passive checks in Nagios and the configuration of the alarms in the Collectors. For details, see the *Configuring alarms* section in the [StackLight Collector documentation](#). A change in `/etc/hiera/override/alarming.yaml` or `/etc/hiera/override/gse_filters.yaml` on any of the nodes monitored by StackLight would require reconfiguring Nagios. It also implies that these two files should be maintained rigorously identical on all the nodes of the environment **including those where Nagios is installed**. StackLight provides Puppet artifacts to help you out with that task. To reconfigure the passive checks in Nagios when `/etc/hiera/override/alarming.yaml` or `/etc/hiera/override/gse_filters.yaml` are modified, run the following command on all the nodes where Nagios is installed:

```
# puppet apply --modulepath=/etc/fuel/plugins/\
lma_infrastructure_alerting-<version>/puppet/modules:/etc/puppet/modules \
/etc/fuel/plugins/lma_infrastructure_alerting-<version>/puppet/manifests/nagios.pp
```

3.2 Configuring service checks using the InfluxDB metrics

You could also configure Nagios to perform active checks, which are not performed by StackLight by default, using the metrics stored in InfluxDB's time-series. For example, you could define active checks to be notified when the CPU activity of particular process is too high.

Consider the following scenario:

- You want to monitor the Elasticsearch server.

- The CPU activity of the Elasticsearch server is captured in a time-series stored in InfluxDB.
- You want to receive an alert at the ‘warning’ level when the CPU load exceeds 30% of system activity.
- You want to receive an alert at the ‘critical’ level when the CPU load exceeds 50% of system activity.

The steps to create such alarms in Nagios are as follows:

1. Connect to each of the nodes running Nagios.
2. Install the Nagios plugin for querying InfluxDB:

```
[root@node-13 ~]# pip install influx-nagios-plugin
```

3. Define the command and the service check in the `/etc/nagios3/conf.d/influxdb_services.conf` file:

```
# Replace <INFLUXDB_HOST>, <INFLUXDB_USER> and <INFLUXDB_PASSWORD> by
# the appropriate values for your deployment
define command {
    command_line /usr/local/bin/check_influx \
        -h <INFLUXDB_HOST> -u <INFLUXDB_USER> -p <INFLUXDB_PASSWORD> -d lma \
        -q "select max(value) from lma_components_cputime_syst \
        where time > now() - 5m and service='$ARG1$' \
        group by time(5m) limit 1" \
        -w $ARG2$ -c $ARG3$
    command_name check_cpu_metric
}

define service {
    service_description Elasticsearch system CPU
    host_name            node-13
    check_command         check_cpu_metric!elasticsearch!30!50:
    use                   generic-service
}
```

4. Verify that the Nagios configuration is valid:

```
[root@node-13 ~]# nagios3 -v /etc/nagios3/nagios.cfg
```

```
[snip]
```

```
Total Warnings: 0
Total Errors:   0
```

No serious problems were detected during the pre-flight check.

5. Restart the Nagios server:

```
[root@node-13 ~]# crm resource restart nagios3
```

6. Go to the Nagios Web UI to verify that the service check has been added.

You can define additional service checks for different nodes or node groups using the same **check_influx** command. To define new service checks, provide the following required arguments:

- A valid InfluxDB query that should return only one row with a single value. See [InfluxDB documentation](#) to learn how to use the InfluxDB’s query language.
- A range specification for the warning threshold.
- A range specification for the critical threshold.

Note: Threshold ranges are defined following the [Nagios format](#).

3.3 Using an external SMTP server with STARTTLS

If your SMTP server requires STARTTLS, perform some manual adjustments to the Nagios configuration after the deployment of your environment.

Note: Prior to enabling STARTTLS, configure the *SMTP Authentication method* parameter in the plugin's settings to use either *Plain*, *Login* or *CRAM-MD5*.

1. Log in to the *LMA Infrastructure Alerting* node.
2. Open the `cmd_notify-service-by-smtp-with-long-service-output.cfg` file in the `/etc/nagios3/conf.d/` directory for editing.
3. Add the `-S smtp-use-starttls` option to the **mail** command. For example:

```
define command{
  command_name      notify-service-by-smtp-with-long-service-output
  command_line      /usr/bin/printf "%b" "***** Nagios *****\n\n"\
    "Notification Type: $NOTIFICATIONTYPE$\n\n"\
    "Service: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\n"\
    "State: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\n"\
    "Additional Info:\n\n$SERVICEOUTPUT$\n$LONGSERVICEOUTPUT$\n" | \
    /usr/bin/mail -s "** $NOTIFICATIONTYPE$ " \
    "Service Alert: $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" \
    -r 'nagios@localhost' \
    -S smtp="smtp://<SMTP_HOST>" \
    -S smtp-auth=<SMTP_AUTH_METHOD> \
    -S smtp-auth-user='<SMTP_USER>' \
    -S smtp-auth-password='<SMTP_PASSWORD>' \
    -S smtp-use-starttls \
    $CONTACTEMAIL$
}
```

Note: If the server certificate is not present in the standard directory, for example, `/etc/ssl/certs` on Ubuntu, specify its location by adding the `-S ssl-ca-file=<FILE>` option.

To disable the verification of the SSL/TLS server certificate altogether, add the `-S ssl-verify=ignore` option instead.

4. Verify that the Nagios configuration is correct:

```
[root@node-13 ~]# nagios3 -v /etc/nagios3/nagios.cfg
```

5. Restart the Nagios service:

```
[root@node-13 ~]# crm resource restart nagios3
```

3.4 Troubleshooting

If you cannot access the Nagios web UI, use the following troubleshooting tips.

1. Verify that the StackLight Collector is able to connect to the Nagios VIP address on port 80.
2. Verify that the Nagios configuration is valid:

```
[root@node-13 ~]# nagios3 -v /etc/nagios3/nagios.cfg

[snip]

Total Warnings: 0
Total Errors:   0
```

No serious problems were detected during the pre-flight check.

1. Verify that the Nagios server is up and running:

```
[root@node-13 ~]# crm resource status nagios3
resource nagios3 is NOT running
```

2. If Nagios is not running, start it:

```
[root@node-13 ~]# crm resource start nagios3
```

3. Verify that Apache is up and running:

```
[root@node-13 ~]# crm resource status apache2-nagios
```

4. If Apache is not running, start it:

```
[root@node-13 ~]# crm resource start apache2-nagios
```

5. Look for errors in the Nagios `/var/nagios/nagios.log` log file:

6. Look for errors in the Apache log files:

- `/var/log/apache2/nagios_error.log`
- `/var/log/apache2/nagios_wsgi_access.log`
- `/var/log/apache2/nagios_wsgi_error.log`

Nagios may report a host or service state as *UNKNOWN*, for example:

- ‘UNKNOWN: No datapoint have been received ever’
- ‘UNKNOWN: No datapoint have been received over the last X seconds’

Both cases indicate that Nagios does not receive regular passive checks from the StackLight Collector. This may be due to different issues, for example:

- The ‘hekad’ process fails to communicate with Nagios
- The ‘collectd’ and/or ‘hekad’ process have crashed
- One or several alarm rules are misconfigured

For solutions, see the *Troubleshooting* section in the [StackLight Collector plugin documentation](#).