Installation Guide

OpenNMS Horizon 23.0.0, Last updated 2018-10-25 13:06:48 EDT

Table of Contents

1. Java Environment. 1
   1.1. Set JAVA_HOME on Linux 1
   1.2. Set JAVA_HOME on Microsoft Windows 1
2. Setting up a basic OpenNMS Horizon 2
   2.1. Objectives 2
   2.2. Before you begin 2
   2.3. Installing on RHEL 3
   2.4. Installing on Debian 7
   2.5. Installing on Windows 10
3. Monitor isolated location with Minion 14
   3.1. Objectives 14
   3.2. Before you begin 14
   3.3. Installing on RHEL 14
   3.4. Installing on Debian 18
4. Sentinel 23
   4.1. Before you begin 23
   4.2. Installing on RHEL 23
   4.3. Installing on Debian 26
5. Install other versions than stable 30
6. Setup Minion with a config file 31
7. Running in non-root environments 32
   7.1. Send ICMP as non-root 32
   7.2. Trap reception as non-root 32
   7.3. Syslog reception as non-root 33
8. Use R for statistical computing 34
   8.1. Install R on RHEL 34
   8.2. Install R on Debian 34
9. Using a different Time Series Storage 35
   9.1. RRDtool 35
   9.2. Newts 38

Chapter 1. Java Environment

To locate the *Java* system files, applications typically use the `$JAVA_HOME` environment variable. The environment can be set for a specific user or globally for the whole system on boot time.

Example path to Java on RHEL, Debian and Microsoft Windows systems

- RHEL: `/usr/java/jdk1.8.0_71`
- Debian: `/usr/lib/jvm/java-8-oracle`
- Microsoft Windows: `C:\Program Files\Java\jre1.8.0_71`

1.1. Set JAVA_HOME on Linux

Option 1: Set the Java environment for the current user

```
vi ~/.bash_profile
export JAVA_HOME=/path/to/java
```

Option 2: Set the Java environment for all users on boot time

```
vi /etc/profile
export JAVA_HOME=/path/to/java
```

1.2. Set JAVA_HOME on Microsoft Windows

Option 1: Set JAVA_HOME as user specific system variable

```
setx "JAVA_HOME" "path\to\java"
```

Option 2: Set JAVA_HOME as a System variable

```
setx /M "JAVA_HOME" "path\to\java"
```

Chapter 2. Setting up a basic OpenNMS Horizon

The *OpenNMS Horizon* platform can be installed on multiple OS families. This guide provides instructions for installing the platform on *Red Hat Enterprise Linux (RHEL)*-based, *Debian*-based, and *Microsoft Windows* operating systems.

2.1. Objectives

- Installing *OpenNMS Horizon* components on a single node using the built-in *JRobin* as time series storage
- Setup *OpenNMS Horizon* on recommended operating systems
- Login the Web User Interface and change the default admin password

2.2. Before you begin

The following abbreviations will be used to refer to their respective entry through this documentation.

Table 1. Operating Systems

RHEL	Red Hat Enterprise Linux 7 or higher, CentOS 7 or higher
Debian	Debian 9 or higher, Ubuntu 16.04 LTS or higher
Windows	Microsoft Windows Server 2012, Windows 10

It is recommended to meet the following requirements:

Table 2. Installation Requirements

Minimal Hardware	2 CPU, 2 GB RAM, 20 GB disk
Operating System	*RHEL* or *Debian* in a current version is recommended. Please be aware *OpenNMS Horizon* is developed and mostly operated on Linux systems. Community support is limited when you run on *Microsoft Windows* platform. On *Microsoft Windows* the *R* integration for statistical computation on time series data is not supported.
Internet	Access to *{yum,debian}.opennms.org* or *SourceForge* for Microsoft Windows via *https*.
DNS Setup	Please make sure your DNS settings for the OpenNMS server are correct and the localhost name can be resolved. If there is an incorrect or missing *A Resource Record* for the server hostname, OpenNMS might not start correctly. The Java security manager might not initialize and an *RMI class loader disabled* exception will be shown.

Depending on the installed operating system, the path for *OpenNMS Horizon* is different. If the instruction refers to `${OPENNMS_HOME}`, the path is resolved to the following directories:

Table 3. Directory Structure

RHEL	`/opt/opennms`
Debian	`/usr/share/opennms`
Windows	`C:\Program Files\opennms`

2.3. Installing on RHEL

The following steps will be described:

1. Installation of the `opennms` meta package which handles all dependencies
2. Initialize *PostgreSQL* database and configure access
3. Initialize *OpenNMS Horizon* database and start
4. Log in to the Web User Interface and change default admin password

All commands on the command line interface need to be executed with *root* permissions.

Step 1: Install OpenNMS Horizon

Add yum repository and import GPG key

```
yum -y install https://yum.opennms.org/repofiles/opennms-repo-stable-rhel7.noarch.rpm
rpm --import https://yum.opennms.org/OPENNMS-GPG-KEY
```

Installation of with all built-in dependencies

```
yum -y install opennms
```

The following packages will be automatically installed:

- *jicmp6* and *jicmp*: *Java* bridge to allow sending *ICMP messages* from *OpenNMS Horizon* repository.
- *opennms-core*: *OpenNMS Horizon* core services, e.g. *Provisiond*, *Pollerd* and *Collectd* from *OpenNMS Horizon* repository.
- *opennms-webapp-jetty*: *OpenNMS Horizon* web application from *OpenNMS Horizon* repository
- *jdk1.8*: *Oracle Java SE Development Kit 8* environment from *OpenNMS Horizon* respository
- *postgresql*: *PostgreSQL* database server from distribution repository
- *postgresql-libs*: *PostgreSQL* database from distribution repository

With the successful installed packages the *OpenNMS Horizon* is installed in the following directory structure:

```
[root@localhost /opt/opennms]# tree -L 1
.
└──── opennms
    ├──── bin
    ├──── contrib
    ├──── data
    ├──── deploy
    ├──── etc
    ├──── jetty-webapps
    ├──── lib
    ├──── logs -> /var/log/opennms
    ├──── share -> /var/opennms
    └──── system
```

We recommend disabling the OpenNMS Horizon apt repository after installation to prevent upgrades while it is running. *OpenNMS Horizon* requires some manual steps upon upgrade to make sure the database and configuration are consistent in the new version, and on systems with many nodes or lots of events, this can take hours. For this reason, it is recommended to exclude the OpenNMS Horizon packages from update except when you are planning on performing an upgrade.

```
yum-config-manager --disable opennms-repo-stable-common,opennms-repo-stable-rhel7
```

Step 2: Initialize and setup PostgreSQL

Initialization of the PostgreSQL database

```
postgresql-setup initdb
```

System startup configuration for PostgreSQL

```
systemctl enable postgresql
```

Startup PostgreSQL database

```
systemctl start postgresql
```

Create an opennms database user with a password and create an opennms database which is owned by the user opennms

```
su - postgres
createuser -P opennms
createdb -O opennms opennms
```

Set a password for Postgres super user

```
psql -c "ALTER USER postgres WITH PASSWORD 'YOUR-POSTGRES-PASSWORD';"
exit
```

ℹ The super user is required to be able to initialize and change the database schema for installation and updates.

Change the access policy for PostgreSQL

```
vi /var/lib/pgsql/data/pg_hba.conf
```

Allow OpenNMS Horizon accessing the database over the local network with a MD5 hashed password

```
host    all             all             127.0.0.1/32            md5①
host    all             all             ::1/128                 md5①
```

① Change method from `ident` to `md5` for *IPv4* and *IPv6* on localhost.

Apply configuration changes for PostgreSQL

```
systemctl reload postgresql
```

Configure database access in OpenNMS Horizon

```
vi ${OPENNMS_HOME}/etc/opennms-datasources.xml
```

Set credentials to access the PostgreSQL database

```
<jdbc-data-source name="opennms"
                    database-name="opennms"①
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:5432/opennms"
                    user-name="** YOUR-OPENNMS-USERNAME **"②
                    password="** YOUR-OPENNMS-PASSWORD **" />③

<jdbc-data-source name="opennms-admin"
                    database-name="template1"
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:5432/template1"
                    user-name="postgres"④
                    password="** YOUR-POSTGRES-PASSWORD **" />⑤
```

① Set the database name *OpenNMS Horizon* should use

② Set the user name to access the *opennms* database table

③ Set the password to access the *opennms* database table

④ Set the *postgres* user for administrative access to PostgreSQL

⑤ Set the password for administrative access to PostgreSQL

Step 3: Initialize and start OpenNMS Horizon

Detect of Java environment and persist in /opt/opennms/etc/java.conf

```
${OPENNMS_HOME}/bin/runjava -s
```

Initialize the database and detect system libraries persisted in /opt/opennms/etc/libraries.properties

```
${OPENNMS_HOME}/bin/install -dis
```

Configure systemd to start OpenNMS Horizon on system boot

```
systemctl enable opennms
```

Start OpenNMS Horizon

```
systemctl start opennms
```

Step 4: First Login and change default password

After starting OpenNMS the web application can be accessed on http://<ip-or-fqdn-of-your-server>:8980/opennms. The default login user is *admin* and the password is initialized to *admin*.

1. Open in your browser http://<ip-or-fqdn-of-your-server>:8980/opennms
2. Login with with admin/admin
3. Click in main navigation menu on "admin → Change Password → Change Password"
4. Set as current password *admin* and set a new password and confirm your newly set password
5. Click "Submit"
6. Logout and login with your new password

Next Steps

Additional information can be found in these follow up documents:

- Getting Started Guide

  Learn the first steps to setup, configure, and maintain an *OpenNMS Horizon*.

- Reference Guide

  Find in-depth information on the detecters, monitors, collectors, and configuration files used by the *OpenNMS Horizon* platform.

2.4. Installing on Debian

The following steps will be described:

1. Installation of the `opennms` meta package which handles all dependencies
2. Initialize *PostgreSQL* database and configure access
3. Initialize *OpenNMS Horizon* database and start
4. Log in to the Web User Interface and change default admin password

All commands on the command line interface need to be executed with *root* permissions.

Step 1: Install OpenNMS Horizon

Add apt repository in /etc/apt/sources.list.d/opennms.list and add GPG key

```
cat << EOF | sudo tee /etc/apt/sources.list.d/opennms.list
deb https://debian.opennms.org stable main
deb-src https://debian.opennms.org stable main
EOF
wget -O - https://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -
apt update
```

Installation of OpenNMS Horizon with all built-in dependencies

```
apt -y install opennms
```

The following packages depend on the `opennms` package and will be automatically installed:

- *jicmp6* and *jicmp*: *Java* bridge to allow sending *ICMP messages* from *OpenNMS* repository.
- *opennms-core*: *OpenNMS* core services, e.g. *Provisiond*, *Pollerd* and *Collectd* from *OpenNMS* repository.
- *opennms-webapp-jetty*: *OpenNMS* web application from *OpenNMS* repository
- *jdk1.8*: *Oracle Java 8* environment from *OpenNMS* respository
- *postgresql*: *PostgreSQL* database server from distribution repository
- *postgresql-libs*: *PostgreSQL* database from distribution repository

With the successful installed packages the *OpenNMS Horizon* is installed in the following directory structure:

```
[root@localhost /usr/share/opennms]# tree -L 1
.
 └──── opennms
    ├──── bin
    ├──── data
    ├──── deploy
    ├──── etc -> /etc/opennms
    ├──── instances
    ├──── jetty-webapps
    ├──── lib -> ../java/opennms
    ├──── logs -> /var/log/opennms
    ├──── share -> /var/lib/opennms
    └──── system
```

We recommend disabling the OpenNMS Horizon apt repository after installation to prevent upgrades while it is running. *OpenNMS Horizon* requires some manual steps upon upgrade to make sure the database and configuration are consistent in the new version, and on systems with many nodes or lots of events, this can take hours. For this reason, it is recommended to exclude the OpenNMS Horizon packages from update except when you are planning on performing an upgrade.

```
apt-mark hold libopennms-java \
              libopennmsdeps-java \
              opennms-common \
              opennms-db
```

Step 2: Initialize and setup PostgreSQL

The *Debian* package installs the *PostgreSQL* database and is already initialized. The *PostgreSQL* service is already added in the runlevel configuration for system startup.

Startup PostgreSQL database

```
systemctl start postgresql
```

Create an opennms database user with a password and create an opennms database which is owned by the user opennms

```
su - postgres
createuser -P opennms
createdb -O opennms opennms
```

Set a password for Postgres super user

```
psql -c "ALTER USER postgres WITH PASSWORD 'YOUR-POSTGRES-PASSWORD';"
exit
```

The super user is required to be able to initialize and change the database schema for installation and updates.

Configure database access in OpenNMS Horizon

```
vi ${OPENNMS_HOME}/etc/opennms-datasources.xml
```

Set credentials to access the PostgreSQL database

```
<jdbc-data-source name="opennms"
                    database-name="opennms"①
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:5432/opennms"
                    user-name="** YOUR-OPENNMS-USERNAME **"②
                    password="** YOUR-OPENNMS-PASSWORD **" />③

<jdbc-data-source name="opennms-admin"
                    database-name="template1"
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:5432/template1"
                    user-name="postgres"④
                    password="** YOUR-POSTGRES-PASSWORD **" />⑤
```

① Set the database name *OpenNMS Horizon* should use

② Set the user name to access the *opennms* database table

③ Set the password to access the *opennms* database table

④ Set the *postgres* user for administrative access to PostgreSQL

⑤ Set the password for administrative access to PostgreSQL

Step 3: Initialize and start OpenNMS Horizon

Detect of Java environment and persist in /usr/share/opennms/etc/java.conf

```
${OPENNMS_HOME}/bin/runjava -s
```

Initialize the database and detect system libraries persisted in /opt/opennms/etc/libraries.properties

```
${OPENNMS_HOME}/bin/install -dis
```

Configure systemd to start OpenNMS Horizon on system boot

```
systemctl enable opennms
```

Start OpenNMS Horizon

```
systemctl start opennms
```

Step 4: First Login and change default password

After starting OpenNMS the web application can be accessed on http://<ip-or-fqdn-of-your-server>:8980/opennms. The default login user is *admin* and the password is initialized to *admin*.

1. Open in your browser http://<ip-or-fqdn-of-your-server>:8980/opennms
2. Login with with admin/admin
3. Click in main navigation menu on "admin → Change Password → Change Password"
4. Set as current password *admin* and set a new password and confirm your newly set password
5. Click "Submit"
6. Logout and login with your new password

Next Steps

Additional information can be found in these follow up documents:

- Getting Started Guide

  Learn the first steps to setup, configure, and maintain an *OpenNMS Horizon*.

- Reference Guide

  Find in-depth information on the detecters, monitors, collectors, and configuration files used by the *OpenNMS Horizon* platform.

2.5. Installing on Windows

The installer for *Microsoft Windows* does not handle *PostgreSQL* and *Java* dependencies as on *Linux* operating systems.

> ❗ Ensure you have installed *Oracle Java Development Kit 8 (JDK)* which is available on the Oracle page.

The following steps will be described:

1. Install *PostgreSQL* on *Microsoft Windows*
2. Install *OpenNMS Horizon* with GUI installer

3. Initialize *PostgreSQL* database and configure access
4. Log in to the Web User Interface and change default admin password

It is required to have local administration permission to install *OpenNMS Horizon*.

> To edit *OpenNMS* configuration files on *Microsoft Windows* the tool Notepad++ can deal with the formatting of *.property* and *.xml* files.

Step 1: Install PostgreSQL

PostgreSQL is available for *Microsoft Windows* and latest version can be downloaded from Download PostgreSQL page. Follow the on-screen instructions of the graphical installer.

> The placeholder `{PG-VERSION}` represents the *PostgreSQL* version number. Check the Compatibility Matrix to find a suited *PostgreSQL* version.

During the installation of *PostgreSQL* the following information need to be provided:

- Installation directory for *PostgreSQL*, e.g. `C:\Program Files\PostgreSQL{PG-VERSION}`
- Password for the database superuser (*postgres*), this password will be used during the *OpenNMS* setup.
- Port to listen for *PostgreSQL* connections, default is `5432` and can normally be used.
- Locale for the database, keep `[Default locale]`, if you change the locale, *OpenNMS* may not be able to initialize the database.

> It is not required to install anything additional from the *PostgreSQL Stack Builder*.

Step 2: Install OpenNMS with GUI installer

For *Microsoft Windows* environments download the *standalone-opennms-installer-{ONMS-VERSION}.zip* file from the OpenNMS SourceForge repository. Extract the downloaded *ZIP* file.

> The `{ONMS-VERSION}` has to be replaced with the latest stable version number.

Start the graphical installer and follow the on screen instructions. The following information has to be provided:

- Path to *Oracle JDK*, e.g. `C:\Program Files\Java\jdk1.8.0_71`
- Installation path for *OpenNMS*, e.g. `C:\Program Files\OpenNMS`
- Select packages which has to be installed, the minimum default selection is *Core* and *Docs*
- PostgreSQL Database connection
  - Host: Server with *PostgreSQL* running, e.g. `localhost`
  - Name: Database name for *OpenNMS*, e.g. `opennms`
  - Port: *TCP* port connecting to *PostgreSQL* server, e.g. `5432`

    - Username (administrative superuser): *PostgreSQL* superuser, e.g. `postgres`
    - Password (administrative superuser): Password given during *PostgreSQL* setup for the superuser
    - Username (runtime user for opennms): Username to connect to the *OpenNMS* database, e.g. `opennms`
    - Password (runtime user for opennms): Password to connect to the *OpenNMS* database, e.g. `opennms`
- Configure a discovery range for an initial node discovery. If you don't want any discovery set begin and end to the same unreachable address.

> Choose secure passwords for all database users and don't use the example passwords above in production.

Step 3: Configure PostgreSQL access for OpenNMS Horizon

Set credentials to access the PostgreSQL database

```
<jdbc-data-source name="opennms"
                    database-name="opennms"①
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:5432/opennms"
                    user-name="** YOUR-OPENNMS-USERNAME **"②
                    password="** YOUR-OPENNMS-PASSWORD **" />③

<jdbc-data-source name="opennms-admin"
                    database-name="template1"
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:5432/template1"
                    user-name="postgres"④
                    password="** YOUR-POSTGRES-PASSWORD **" />⑤
```

① Set the database name *OpenNMS Horizon* should use

② Set the user name to access the *opennms* database table

③ Set the password to access the *opennms* database table

④ Set the *postgres* user for administrative access to PostgreSQL

⑤ Set the password for administrative access to PostgreSQL

After setting the username and passwords in `opennms-datasources.xml` re-run the graphical installer and also initialize the database. *OpenNMS* can be started and stopped with the `start.bat` and `stop.bat` script located in `%OPENNMS_HOME%\bin` directory.

> The Wiki article Configuring OpenNMS as Windows Service describes how to create a *Windows Service* from the `start.bat` files. There is also a Java Wrapper which allows to install *Java* applications as *Windows Service*.

Step 4: First Login and change default password

After starting OpenNMS the web application can be accessed on http://<ip-or-fqdn-of-your-server>:8980/opennms. The default login user is *admin* and the password is initialized to *admin*.

1. Open in your browser http://<ip-or-fqdn-of-your-server>:8980/opennms
2. Login with with admin/admin
3. Click in main navigation menu on "admin → Change Password → Change Password"
4. Set as current password *admin* and set a new password and confirm your newly set password
5. Click "Submit"
6. Logout and login with your new password

Next Steps

Additional information can be found in these follow up documents:

- Getting Started Guide

  Learn the first steps to setup, configure, and maintain an *OpenNMS Horizon*.

- Reference Guide

  Find in-depth information on the detecters, monitors, collectors, and configuration files used by the *OpenNMS Horizon* platform.

Chapter 3. Monitor isolated location with Minion

This section describes how to install the *Minion* to monitor devices and services in a location which can’t be reached from an *OpenNMS Horizon* instance.

3.1. Objectives

- Install a *Minion* to monitor devices and services unreachable from an *OpenNMS Horizon* instance
- Configure an authenticated unencrypted communication between *Minion* and *OpenNMS Horizon* using *ActiveMQ* and *REST*

3.2. Before you begin

Setting up a *OpenNMS Horizon* with *Minions* requires:

- Instance of *OpenNMS Horizon* needs to be exact same version as *Minion* packages
- Packages are available as *RPMs* for *RHEL*-based systems and *DEBs* for *Debian*-based systems
- *OpenNMS Horizon* needs to be installed and communication to the *REST (8980/tcp)* and *ActiveMQ (616161/tcp)* endpoints is possible

Depending on the installed operating system, the path for *Minion* is different. If the instruction refers to `${MINION_HOME}`, the path is resolved to the following directories:

Table 4. Directory Structure

RHEL	`/opt/minion`
Debian	`/usr/share/minion`

3.3. Installing on RHEL

1. Setup *OpenNMS Horizon* to allow *Minion* communication
2. Installation of the `opennms-minion` meta package which handles all dependencies
3. Starting *Minion* and access the *Karaf* console over *SSH*
4. Configure *Minion* to communicate with *OpenNMS Horizon*
5. Verify the connectivity between *Minion* and *OpenNMS Horizon*

All commands on the command line interface need to be executed with *root* permissions.

Step 1: Setup OpenNMS Horizon to allow Minion communication

Communication between a *Minion* and *OpenNMS Horizon* uses *REST API* and a messaging system, by default *ActiveMQ*. An authenticated user in *OpenNMS Horizon* is required for these communication channels. The security role *ROLE_MINION* includes the minimal amount of permissions required for a *Minion* to operate.

> As an example we use in this guide the user name *minion* with password *minion*. Change the credentials accordingly.

Create a user minion in the OpenNMS Horizon web user interface

1. Login the web user interface with a user which has administrative permissions
2. Go in the main navigation to *"Login Name → Configure OpenNMS → Configure Users, Groups and On-Call Roles → Configure Users"*
3. Add a new user with login name *minion* and password *minion* and click *Ok*
4. Assign the security role *ROLE_MINION*, optional fill in a comment for what location and purpose the user is used for and click *Finish*
5. The *minion* user should now be listed in the *User List*

Configure ActiveMQ to allow communication on public network interface

```
vi ${OPENNMS_HOME}/etc/opennms-activemq.xml
```

Remove comments for the transport connector listening on 0.0.0.0 and save

```
<transportConnector name="openwire" uri="tcp://0.0.0.0:61616?useJmx=false
&amp;maximumConnections=1000&amp;wireformat.maxFrameSize=104857600"/>
```

Restart OpenNMS Horizon

```
systemctl restart opennms
```

Verify if port 61616/tcp is listening on all interfaces

```
ss -lnpt sport = :61616
State    Recv-Q  Send-Q  Local Address:Port   Peer  Address:Port
LISTEN   0       128     *:61616              *:*   users:(("java",pid=1,fd=706))
```

Step 2: Install the repository and Minion package

Connect with *SSH* to your remote *RHEL* system where you need a *Minion* to be installed.

Install the Yum repository

```
yum -y install https://yum.opennms.org/repofiles/opennms-repo-stable-rhel7.noarch.rpm
rpm --import https://yum.opennms.org/OPENNMS-GPG-KEY
```

Install the Minion package

```
yum -y install opennms-minion
```

The following packages will be automatically installed:

- *opennms-minion*: The Minion meta package
- *opennms-minion-container*: The *Karaf* OSGi container with *Minion* branding and additional management extensions
- *opennms-minion-features-core*: Core utilities and services required by the *Minion* features
- *opennms-minion-features-default*: Service-specific features

With the successful installed packages the *Minion* is installed in the following directory structure:

```
[root@localhost /opt/minion]# $ tree -L 1
.
├── bin
├── deploy
├── etc
├── lib
├── repositories
└── system
```

The Minion's startup configuration can be changed by editing the `/etc/sysconfig/minion` file. It allows to override the defaults used at startup including:

- Location of the JDK
- Memory usage
- User to run as

Step 3: Starting the Minion and test access to Karaf Shell

Configure systemd to start Minion on system boot

```
systemctl enable minion
```

Startup Minion

```
systemctl start minion
```

Test access to Karaf shell with user admin and password admin and exit with <ctrl-d>

```
ssh -p 8201 admin@localhost
```

Step 4: Configure Minion to communicate with OpenNMS Horizon

Login to the Karaf Shell on the system where your Minion is installed with SSH

```
ssh -p 8201 admin@localhost
```

Configure the Minion's location and endpoint URLs for communication with OpenNMS Horizon

```
[root@localhost /root]# $ ssh -p 8201 admin@localhost
...
admin@minion()> config:edit org.opennms.minion.controller
admin@minion()> config:property-set location Office-Pittsboro
admin@minion()> config:property-set http-url http://opennms-fqdn:8980/opennms
admin@minion()> config:property-set broker-url failover:tcp://opennms-fqdn:61616
admin@minion()> config:update
```

Include the `failover:` portion of the broker URL to allow the *Minion* to re-establish connectivity on failure. For a reference on the different URL formats, see ActiveMQ URI Protocols.

Configure the credentials to use when communicating with OpenNMS Horizon

```
admin@minion()> scv:set opennms.http minion minion
admin@minion()> scv:set opennms.broker minion minion
```

Another way to configure credentials is to use the `scvcli` utility in your *Minion* `bin` directory.

Example of configuring credentials with the command line utility `scvcli`

```
[root@localhost /root]# $ cd /opt/minion
[root@localhost /opt/minion]# $ ./bin/scvcli set opennms.http minion minion
[root@localhost /opt/minion]# $ ./bin/scvcli set opennms.broker minion minion
```

Restart the Minion after updating the credentials

```
[root@localhost /root]# $ systemctl restart minion
```

The credentials are configured separately since they are encrypted on disk.

Step 5: Verifying Connectivity

Connect to Karaf Shell of the Minion

```
ssh -p 8201 admin@localhost
```

Verify connectivity with the OpenNMS Horizon

```
admin@minion()> minion:ping
Connecting to ReST...
OK
Connecting to Broker...
OK
admin@minion()>
```

3.4. Installing on Debian

1. Setup *OpenNMS Horizon* to allow *Minion* communication
2. Installation of the `opennms-minion` meta package which handles all dependencies
3. Starting *Minion* and access the *Karaf* console over *SSH*
4. Configure *Minion* to communicate with *OpenNMS Horizon*
5. Verify the connectivity between *Minion* and *OpenNMS Horizon*

All commands on the command line interface need to be executed with *root* permissions.

Step 1: Setup OpenNMS Horizon to allow Minion communication

Communication between a *Minion* and *OpenNMS Horizon* uses *REST API* and a messaging system, by default *ActiveMQ*. An authenticated user in *OpenNMS Horizon* is required for these communication channels. The security role *ROLE_MINION* includes the minimal amount of permissions required for a *Minion* to operate.

> ❗ As an example we use in this guide the user name *minion* with password *minion*. Change the credentials accordingly.

Create a user minion in the OpenNMS Horizon web user interface

1. Login the web user interface with a user which has administrative permissions
2. Go in the main navigation to *"Login Name → Configure OpenNMS → Configure Users, Groups and On-Call Roles → Configure Users"*
3. Add a new user with login name *minion* and password *minion* and click *Ok*
4. Assign the security role *ROLE_MINION*, optional fill in a comment for what location and purpose the user is used for and click *Finish*
5. The *minion* user should now be listed in the *User List*

Configure ActiveMQ to allow communication on public network interface

```
vi ${OPENNMS_HOME}/etc/opennms-activemq.xml
```

Remove comments for the transport connector listening on 0.0.0.0 and save

```
<transportConnector name="openwire" uri="tcp://0.0.0.0:61616?useJmx=false
&amp;maximumConnections=1000&amp;wireformat.maxFrameSize=104857600"/>
```

Restart OpenNMS Horizon

```
systemctl restart opennms
```

Verify if port 61616/tcp is listening on all interfaces

```
ss -lnpt sport = :61616
State   Recv-Q  Send-Q  Local Address:Port  Peer  Address:Port
LISTEN  0       128     *:61616             *:*   users:(("java",pid=1,fd=706))
```

Step 2: Install the repository and Minion package

Add apt repository in /etc/apt/sources.list.d/opennms.list and add GPG key

```
echo 'deb https://debian.opennms.org stable main \
      deb-src https://debian.opennms.org stable main' >
/etc/apt/sources.list.d/opennms.list
wget -O - https://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -
apt update
```

Install the Minion package

```
apt -y install opennms-minion
```

The following packages will be automatically installed:

- *opennms-minion*: The Minion meta package
- *opennms-minion-container*: The *Karaf* OSGi container with *Minion* branding and additional management extensions
- *opennms-minion-features-core*: Core utilities and services required by the *Minion* features
- *opennms-minion-features-default*: Service-specific features

The *Minion* packages setup the following directory structure:

```
[root@localhost /usr/share/minion]# $ tree -L 1
.
├── bin
├── deploy
├── etc
├── lib
├── repositories
└── system
```

Additionally, symbolic links are set up pointing to `/etc/minion` and `/var/log/minion` to match Debian's expected filesystem layout.

The Minion's startup configuration can be changed by editing the `/etc/default/minion` file. It allows to override the defaults used at startup including:

- Location of the JDK
- Memory usage
- User to run as

Step 3: Starting the Minion and test access to Karaf Shell

Configure systemd to start Minion on system boot

```
systemctl enable minion
```

Startup Minion

```
systemctl start minion
```

Test access to Karaf shell with user admin and password admin and exit with <ctrl-d>

```
ssh -p 8201 admin@localhost
```

Step 4: Configure Minion to communicate with OpenNMS Horizon

Login to the Karaf Shell on the system where your Minion is installed with SSH

```
ssh -p 8201 admin@localhost
```

Configure the Minion's location and endpoint URLs for communication with OpenNMS Horizon

```
[root@localhost /root]# $ ssh -p 8201 admin@localhost
...
admin@minion()> config:edit org.opennms.minion.controller
admin@minion()> config:property-set location Office-Pittsboro
admin@minion()> config:property-set http-url http://opennms-fqdn:8980/opennms
admin@minion()> config:property-set broker-url failover:tcp://opennms-fqdn:61616
admin@minion()> config:update
```

Include the `failover:` portion of the broker URL to allow the *Minion* to re-establish connectivity on failure. For a reference on the different URL formats, see ActiveMQ URI Protocols.

Configure the credentials to use when communicating with OpenNMS Horizon

```
admin@minion()> scv:set opennms.http minion minion
admin@minion()> scv:set opennms.broker minion minion
```

Another way to configure credentials is to use the `scvcli` utility in your *Minion* `bin` directory.

Example of configuring credentials with the command line utility `scvcli`

```
[root@localhost /root]# $ cd /opt/minion
[root@localhost /opt/minion]# $ ./bin/scvcli set opennms.http minion minion
[root@localhost /opt/minion]# $ ./bin/scvcli set opennms.broker minion minion
```

Restart the Minion after updating the credentials

```
[root@localhost /root]# $ systemctl restart minion
```

The credentials are configured separately since they are encrypted on disk.

Step 5: Verifying Connectivity

Connect to Karaf Shell of the Minion

```
ssh -p 8201 admin@localhost
```

Verify connectivity with the OpenNMS Horizon

```
admin@minion()> minion:ping
Connecting to ReST...
OK
Connecting to Broker...
OK
admin@minion()>
```

Chapter 4. Sentinel

This section describes how to install the *Sentinel* to scale individual components of OpenNMS Horizon.

ℹ At the moment only flows can be distributed using *Sentinel*. In the future more components will follow.

4.1. Before you begin

Setting up a *OpenNMS Horizon* with *Sentinel* requires:

- Instance of *OpenNMS Horizon* needs to be exact same version as *Sentinel* packages
- Packages are available as *RPMs* for *RHEL*-based systems and *DEBs* for *Debian*-based systems
- *OpenNMS Horizon* needs to be installed and communication to the *REST (8980/tcp)* and *ActiveMQ (616161/tcp)* endpoints is possible
- At least one *Minion* needs to be installed and successful communicate with the *OpenNMS Horizon*

Depending on the installed operating system, the path for *Sentinel* is different. If the instruction refers to `${SENTINEL_HOME}`, the path is resolved to the following directories:

Table 5. Directory Structure

RHEL	`/opt/sentinel`
Debian	`/usr/share/sentinel`

4.2. Installing on RHEL

1. Setup *OpenNMS Horizon* to allow *Sentinel* communication
2. Installation of the `opennms-sentinel` meta package which handles all dependencies
3. Starting *Sentinel* and access the *Karaf* console over *SSH*
4. Configure *Sentinel* to communicate with *OpenNMS Horizon*
5. Verify the connectivity between *Sentinel* and *OpenNMS Horizon*

All commands on the command line interface need to be executed with *root* permissions.

Step 1: Setup OpenNMS Horizon to allow Sentinel communication

This step is exactly the same as for *Minion*. Even the role name `ROLE_MINION` can be used, as there does not exist a dedicated role `ROLE_SENTINEL` yet.

Therefore, please refer to section Setup OpenNMS Horizon to allow Minion communication.

> Even if we have to configure the communication to the *OpenNMS Horizon* exactly the same as for *Minion* no ReST requests are made and may be removed at a later state.

Step 2: Install the repository and Sentinel package

Connect with *SSH* to your remote *RHEL* system where the *Sentinel* should be installed.

Install the Yum repository

```
yum install -y https://yum.opennms.org/repofiles/opennms-repo-stable-rhel7.noarch.rpm
rpm --import https://yum.opennms.org/OPENNMS-GPG-KEY
```

Install the Sentinel package

```
yum -y install opennms-sentinel
```

With the successful installed packages the *Sentinel* is installed in the following directory structure:

```
[root@localhost /opt/sentinel]# $ tree -L 1
.
|-- bin
|-- COPYING
|-- data
|-- deploy
|-- etc
|-- lib
`-- system
```

The Sentinel's startup configuration can be changed by editing the `/etc/sysconfig/sentinel` file. It allows to override the defaults used at startup including:

- Location of the JDK
- Memory usage
- User to run as

Step 3: Starting the Sentinel and test access to Karaf Shell

Configure systemd to start Sentinel on system boot

```
systemctl enable sentinel
```

Startup Sentinel

```
systemctl start sentinel
```

Test access to Karaf shell with user admin and password admin and exit with <ctrl-d>

```
ssh -p 8301 admin@localhost
```

Step 4: Configure Sentinel to communicate with OpenNMS Horizon

Login to the Karaf Shell on the system where your Sentinel is installed with SSH

```
ssh -p 8301 admin@localhost
```

Configure the Sentinel's location and endpoint URLs for communication with OpenNMS Horizon

```
[root@localhost /root]# $ ssh -p 8201 admin@localhost
...
admin@sentinel()> config:edit org.opennms.sentinel.controller
admin@sentinel()> config:property-set location Office-Pittsboro
admin@sentinel()> config:property-set http-url http://opennms-fqdn:8980/opennms
admin@sentinel()> config:property-set broker-url failover:tcp://opennms-fqdn:61616
admin@sentinel()> config:update
```

> 💡 Include the `failover:` portion of the broker URL to allow the *Sentinel* to re-establish connectivity on failure. For a reference on the different URL formats, see ActiveMQ URI Protocols.

> ℹ Even if the id, location and http-url must be set the same ways as for *Minion*, this may change in future versions of *Sentinel*.

Configure the credentials to use when communicating with OpenNMS Horizon

```
admin@sentinel()> scv:set opennms.http minion minion
admin@sentinel()> scv:set opennms.broker minion minion
```

Username and password is explicitly set to `minion` as it is assumed that they share the same credentials and roles.

> 💡 Another way to configure credentials is to use the `scvcli` utility in your *Sentinel* `bin` directory.

Example of configuring credentials with the command line utility `scvcli`

```
[root@localhost /root]# $ cd /opt/sentinel
[root@localhost /opt/sentinel]# $ ./bin/scvcli set opennms.http minion minion
[root@localhost /opt/sentinel]# $ ./bin/scvcli set opennms.broker minion minion
```

Restart the Sentinel after updating the credentials

```
[root@localhost /root]# $ systemctl restart sentinel
```

The credentials are configured separately since they are encrypted on disk.

Step 5: Verifying Connectivity

Connect to Karaf Shell of the Sentinel

```
ssh -p 8301 admin@localhost
```

Verify connectivity with the OpenNMS Horizon

```
admin@sentinel()> feature:install sentinel-core
admin@sentinel> health:check
Verifying the health of the container

Verifying installed bundles      [ Success  ]
Connecting to OpenNMS ReST API   [ Success  ]

=> Everything is awesome
admin@sentinel()>
```

The `health:check` command is a newer and more flexibel version of the original `minion:ping` command. Therefore on *Sentinel* there is no equivalent such as `sentinel:ping`.

4.3. Installing on Debian

1. Setup *OpenNMS Horizon* to allow *Sentinel* communication
2. Installation of the `opennms-sentinel` meta package which handles all dependencies
3. Starting *Sentinel* and access the *Karaf* console over *SSH*
4. Configure *Sentinel* to communicate with *OpenNMS Horizon*
5. Verify the connectivity between *Sentinel* and *OpenNMS Horizon*

All commands on the command line interface need to be executed with *root* permissions.

Step 1: Setup OpenNMS Horizon to allow Sentinel communication

This step is exactly the same as for *Minion*. Even the role name `ROLE_MINION` can be used, as there does not exist a dedicated role `ROLE_SENTINEL` yet.

Therefore, please refer to section Setup OpenNMS Horizon to allow Minion communication.

> ℹ Even if we have to configure the communication to the *OpenNMS Horizon* exactly the same as for *Minion* no ReST requests are made and may be removed at a later state.

Step 2: Install the repository and Sentinel package

Add apt repository in /etc/apt/sources.list.d/opennms.list and add GPG key

```
echo 'deb https://debian.opennms.org stable main \
      deb-src https://debian.opennms.org branches/features-sentinel main' >
/etc/apt/sources.list.d/opennms.list
wget -O - https://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -
apt update
```

Install the Sentinel package

```
apt -y install opennms-sentinel
```

The *Sentinel* packages setup the following directory structure:

```
[root@localhost /usr/share/sentinel]# $ tree -L 1
.
|-- bin
|-- COPYING
|-- data
|-- debian
|-- deploy
|-- etc
|-- lib
`-- system
```

Additionally, symbolic links are set up pointing to `/etc/sentinel` and `/var/log/sentinel` to match Debian's expected filesystem layout.

The Minion's startup configuration can be changed by editing the `/etc/default/sentinel` file. It allows to override the defaults used at startup including:

- Location of the JDK
- Memory usage
- User to run as

Step 3: Starting the Sentinel and test access to Karaf Shell

Configure systemd to start Sentinel on system boot

```
systemctl enable sentinel
```

Startup Sentinel

```
systemctl start sentinel
```

Test access to Karaf shell with user admin and password admin and exit with <ctrl-d>

```
ssh -p 8301 admin@localhost
```

Step 4: Configure Sentinel to communicate with OpenNMS Horizon

Login to the Karaf Shell on the system where your Sentinel is installed with SSH

```
ssh -p 8301 admin@localhost
```

Configure the Sentinel's location and endpoint URLs for communication with OpenNMS Horizon

```
[root@localhost /root]# $ ssh -p 8201 admin@localhost
...
admin@sentinel()> config:edit org.opennms.sentinel.controller
admin@sentinel()> config:property-set location Office-Pittsboro
admin@sentinel()> config:property-set http-url http://opennms-fqdn:8980/opennms
admin@sentinel()> config:property-set broker-url failover:tcp://opennms-fqdn:61616
admin@sentinel()> config:update
```

💡 Include the `failover:` portion of the broker URL to allow the *Sentinel* to re-establish connectivity on failure. For a reference on the different URL formats, see ActiveMQ URI Protocols.

ℹ Even if the id, location and http-url must be set the same ways as for *Minion*, this may change in future versions of *Sentinel*.

Configure the credentials to use when communicating with OpenNMS Horizon

```
admin@sentinel()> scv:set opennms.http minion minion
admin@sentinel()> scv:set opennms.broker minion minion
```

Username and password is explicitly set to `minion` as it is assumed that they share the same credentials and roles.

💡 Another way to configure credentials is to use the `scvcli` utility in your *Sentinel* `bin` directory.

Example of configuring credentials with the command line utility `scvcli`

```
[root@localhost /root]# $ cd /opt/sentinel
[root@localhost /usr/share/sentinel]# $ ./bin/scvcli set opennms.http minion minion
[root@localhost /usr/share/sentinel]# $ ./bin/scvcli set opennms.broker minion minion
```

Restart the Sentinel after updating the credentials

```
[root@localhost /root]# $ systemctl restart sentinel
```

> The credentials are configured separately since they are encrypted on disk.

Step 5: Verifying Connectivity

Connect to Karaf Shell of the Sentinel

```
ssh -p 8301 admin@localhost
```

Verify connectivity with the OpenNMS Horizon

```
admin@sentinel()> feature:install sentinel-core
admin@sentinel> health:check
Verifying the health of the container

Verifying installed bundles      [ Success  ]
Connecting to OpenNMS ReST API   [ Success  ]

=> Everything is awesome
admin@sentinel()>
```

> The `health:check` command is a newer and more flexibel version of the original `minion:ping` command. Therefore on *Sentinel* there is no equivalent such as `sentinel:ping`.

Chapter 5. Install other versions than stable

Installation packages are available for different releases of *OpenNMS Horizon* or *Minion*. You will need to choose which release you would like to run and then configure your package repository to point to that release. Configuring a package repository will enable you to install and update the software by using standard Linux software update tools like *yum* and *apt*.

The following package repositories are available:

Table 6. OpenNMS package repositories

Release	Description
`stable`	Latest stable release. This version is recommended for all users.
`testing`	Release candidate for the next stable release.
`snapshot`	Latest successful development build, the "nightly" build.
`branches/${BRANCH-NAME}`	Install from a specific branch name for testing a specific feature that is under development. Available branches can be found in https://yum.opennms.org/branches/ or https://debian.opennms.org/dists/branches/.

To install a different release the repository files have to be installed and manually modified.

In *Debian* systems modify the repository file in `/etc/apt/sources.list.d/opennms.list`.

```
deb https://debian.opennms.org snapshot main①
deb-src https://debian.opennms.org snapshot main①
EOF
wget -O - https://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -
apt update
```

① Change from `stable` to `snapshot`

On *RHEL* systems you can install a snapshot repository with:

```
yum -y install https://yum.opennms.org/repofiles/opennms-repo-snapshot-
rhel7.noarch.rpm
```

For branches use `repofiles/opennms-repo-branches-${branch-name}-rhel7.noarch.rpm`.

The installation procedure is the same as with the stable version.

Chapter 6. Setup Minion with a config file

Beside manually configuring a *Minion* instance via the *Karaf CLI* it is possibleto modify and deploy its configuration file through configuration management tools. The configuration file is located in `${MINION_HOME}/etc/org.opennms.minion.controller.cfg`. All configurations set in *Karaf CLI* will be persisted in this configuration file which can also be populated through configuration management tools.

Configuration file for Minion

```
id = 00000000-0000-0000-0000-deadbeef0001
location = MINION
broker-url = tcp://myopennms.example.org:61616
http-url = http://myopennms.example.org:8980/opennms
```

The *Minion* needs to be restarted when this configuration file is changed.

> 💡 In case the credentials needs to be set through the *CLI* with configuration management tools or scripts, the `${MINION_HOME}/bin/client` command can be used which allows to execute *Karaf* commands through the Linux shell.

Chapter 7. Running in non-root environments

This section provides information running *OpenNMS Horizon* and *Minions* processes in non-root environments. Running with a system user have restricted possibilites. This section describes how to configure your *Linux* system related to:

- sending *ICMP* packages as an unprivileged user
- receiving *Syslog* on ports < 1023, e.g. 514/udp
- receiving *SNMP Trap* on ports < 1023,e.g. 162/udp

7.1. Send ICMP as non-root

By default, *Linux* does not allow regular users to perform `ping` operations from arbitrary programs (including *Java*). To enable the *Minion* or *OpenNMS Horizon* to ping properly, you must set a `sysctl` option.

Enable User Ping (Running System)d

```
# run this command as root to allow ping by any user (does not survive reboots)
sysctl net.ipv4.ping_group_range='0 429496729'
```

If you wish to restrict the range further, use the *GID* for the user the *Minion* or *OpenNMS Horizon* will run as, rather than `429496729`.

To enable this permanently, create a file in `/etc/sysctl.d/` to set the range:

/etc/sysctl.d/99-zzz-non-root-icmp.conf

```
# we start this filename with "99-zzz-" to make sure it's last, after anything else
that might have set it
net.ipv4.ping_group_range=0 429496729
```

7.2. Trap reception as non-root

If you wish your *Minion* or *OpenNMS Horizon* to listen to *SNMP Traps*, you will need to configure your firewall to port forward from the privileged trap port (162) to the Minion’s default trap listener on port 1162.

Forward 162 to 1162 with Firewalld

```
# enable masquerade to allow port-forwards
firewall-cmd --add-masquerade
# forward port 162 TCP and UDP to port 1162 on localhost
firewall-cmd --add-forward-port=port=162:proto=udp:toport=1162:toaddr=127.0.0.1
firewall-cmd --add-forward-port=port=162:proto=tcp:toport=1162:toaddr=127.0.0.1
```

7.3. Syslog reception as non-root

If you wish your *Minion* or *OpenNMS Horizon* to listen to syslog messages, you will need to configure your firewall to port forward from the privileged *Syslog* port (514) to the Minion's default syslog listener on port 1514.

Forward 514 to 1514 with Firewalld

```
# enable masquerade to allow port-forwards
firewall-cmd --add-masquerade
# forward port 514 TCP and UDP to port 1514 on localhost
firewall-cmd --add-forward-port=port=514:proto=udp:toport=1514:toaddr=127.0.0.1
firewall-cmd --add-forward-port=port=514:proto=tcp:toport=1514:toaddr=127.0.0.1
```

Chapter 8. Use R for statistical computing

R is a free software environment for statistical computing and graphics. *OpenNMS Horizon* can leverage the power of *R* for forecasting and advanced calculations on collected time series data.

OpenNMS Horizon interfaces with *R* via *stdin* and *stdout*, and for this reason, *R* must be installed on the same host as *OpenNMS Horizon*. Note that installing *R* is optional, and not required by any of the core components.

❗ The *R* integration is not currently supported on *Microsoft Windows* systems.

8.1. Install R on RHEL

Install the EPEL repositories

```
yum install epel-release
```

Install R

```
yum install R
```

8.2. Install R on Debian

Install R

```
apt -y install r-recommended
```

Chapter 9. Using a different Time Series Storage

OpenNMS Horizon stores performance data in a time series storage which is by default JRobin. For different scenarios it is useful to switch to a different time series storage. The following implementations are supported:

Table 7. Supported Time Series Databasees

JRobin	*JRobin* is a clone of *RRDTool* written in *Java*, it does not fully cover the latest feature set of *RRDTool* and is the default when you install *OpenNMS Horizon*. Data is stored on the local file system of the OpenNMS Horizon node. Depending on I/O capabilities it works good for small to medium sized installations.
RRDTool	*RRDTool* is active maintained and the de-facto standard dealing with time series data. Data is stored on the local file system of the OpenNMS Horizon node. Depending on I/O capabilities it works good for small to medium sized installations.
Newts	Newts is a database schema for Cassandra. The time series is stored on a dedicated *Cassandra* cluster which gives growth flexibility and allows to persist time series data in a large scale.

This section describes how to configure *OpenNMS Horizon* to use *RRDTool* and *Newts*.

> The way how data is stored in the different time series databases makes it extremely hard to migrate from one technology to another. Data loss can't be prevented when you switch from one to another.

9.1. RRDtool

In most *Open Source* applications, RRDtool is often used and is the de-facto open standard for *Time Series Data*. The basic installation of *OpenNMS Horizon* comes with *JRobin* but it is simple to switch the system to use *RRDtool* to persist *Time Series Data*. This section describes how to install *RRDtool*, the *jrrd2 OpenNMS Java Interface* and how to configure *OpenNMS Horizon* to use it.

9.1.1. Install RRDTool on RHEL

> Following this guide does not cover data migration from *JRobin* to *RRDTool*.

> To install *jrrd2* enable the OpenNMS YUM repository ensure the repositories are enabled. You can enable them with `yum-config-manager --enable opennms-repo-stable-common,opennms-repo-stable-rhel7`.

A more current version of *RRDTool* is in the *OpenNMS* YUM repository. The provided versions can be shown with `yum info rrdtool`. This guide uses the *RRDTool* provided in the *OpenNMS* repository. When using the *CentOS* provided *RRDTool* package verify the path to the *rrdtool* binary file.

Step 1: Install RRDTool and the jrrd2 interface

Installation on RHEL

```
yum -y install rrdtool jrrd2
```

Step 2: Configure OpenNMS Horizon to use RRDTool

```
cat << EOF | sudo tee /opt/opennms/etc/opennms.properties.d/timeseries.properties
org.opennms.rrd.strategyClass=org.opennms.netmgt.rrd.rrdtool.MultithreadedJniRrdStrate
gy
org.opennms.rrd.interfaceJar=/usr/share/java/jrrd2.jar
opennms.library.jrrd2=/usr/lib64/libjrrd2.so
org.opennms.web.graphs.engine=rrdtool # optional, unset if you want to keep Backshift
as default
EOF
```

The visualization with the graph engine is optional. You can still use the default graphing engine `backshift` by not setting the `org.opennms.web.graphs.engine` property and use the system default.

Step 3: Restart OpenNMS Horizon and verify setup

```
find /opt/opennms/share/rrd -iname "*.rrd"
```

With the first data collection, *RRDTool* files with extension *.rrd* will be created. The *JRobin* files with extension *.jrb* are not used anymore and are not deleted automatically.

9.1.2. Reference

The following configuration files have references to the *RRDTool* binary and may be changed if you have a customized *RRDTool* setup.

Table 8. References to the RRDtool binary

Configuration file	Property
`opennms.properties`	`rrd.binary=/usr/bin/rrdtool`
`response-adhoc-graph.properties`	`command.prefix=/usr/bin/rrdtool`

Configuration file	Property

`response-graph.properties`	`command.prefix=/usr/bin/rrdtool`
	`info.command=/usr/bin/rrdtool`

`snmp-adhoc-graph.properties`	`command.prefix=/usr/bin/rrdtool`

`snmp-graph.properties`	`command.prefix=/usr/bin/rrdtool`
	`command=/usr/bin/rrdtool info`

9.1.3. Install RRDTool on Debian

Following this guide does not cover data migration from *JRobin* to *RRDTool*.

A more current version of *RRDTool* is in the *OpenNMS* YUM repository. The provided versions can be shown with `apt show rrdtool`. This guide uses the *RRDTool* provided in the *OpenNMS* repository. When using the *Debian/Ubuntu* provided *RRDTool* package verify the path to the *rrdtool* binary file.

Step 1: Install RRDTool and the jrrd2 interface

Installation on RHEL

```
apt -y install rrdtool jrrd2
```

Step 2: Configure OpenNMS Horizon to use RRDTool

```
cat << EOF | sudo tee
/usr/share/opennms/etc/opennms.properties.d/timeseries.properties
org.opennms.rrd.strategyClass=org.opennms.netmgt.rrd.rrdtool.MultithreadedJniRrdStrate
gy
org.opennms.rrd.interfaceJar=/usr/share/java/jrrd2.jar
opennms.library.jrrd2=/usr/lib/jni/libjrrd2.so
org.opennms.web.graphs.engine=rrdtool # optional, unset if you want to keep Backshift
as default
EOF
```

The visualization with the graph engine is optional. You can still use the default graphing engine `backshift` by not setting the `org.opennms.web.graphs.engine` property and use the system default.

Step 3: Restart OpenNMS Horizon and verify setup

```
find /usr/share/opennms/share/rrd -iname "*.rrd"
```

With the first data collection, *RRDTool* files with extension *.rrd* will be created. The *JRobin* files with extension *.jrb* are not used anymore and are not deleted automatically.

9.1.4. Reference

The following configuration files have references to the *RRDTool* binary and may be changed if you have a customized *RRDTool* setup.

Table 9. References to the RRDtool binary

Configuration file	Property
`opennms.properties`	`rrd.binary=/usr/bin/rrdtool`
`response-adhoc-graph.properties`	`command.prefix=/usr/bin/rrdtool`

`response-graph.properties`	`command.prefix=/usr/bin/rrdtool`
	`info.command=/usr/bin/rrdtool`

`snmp-adhoc-graph.properties`	`command.prefix=/usr/bin/rrdtool`

`snmp-graph.properties`	`command.prefix=/usr/bin/rrdtool`
	`command=/usr/bin/rrdtool info`

9.2. Newts

Newts is a time-series data store based on Apache Cassandra. *Newts* is a persistence strategy, that can be used as an alternative to JRobin or RRDtool.

> ❗ It is currently not supported to initialize the *Newts* keyspace from *Microsoft Windows Server* operating system. *Microsoft Windows* based *Cassandra* server can be part of the cluster, but keyspace initialization is only possible using a _Linux-_based system.

9.2.1. Setting up Cassandra

> ⚠ *Cassandra* is only required when using *Newts*. If your *OpenNMS Horizon* system is not using *Newts*, you can skip this section.

It is recommended to install *Cassandra* on a dedicated server, but is also possible to run a node on the *OpenNMS Horizon* server itself. This installation guide describes how to set up a single *Cassandra* instance on the same system as *OpenNMS Horizon* for the purpose of evaluating and testing *Newts*. These steps are not suitable for a production *Cassandra Cluster*. If you already have a running cluster you can skip this section.

For further information see Cassandra Getting Started Guide. Before setting up a production cluster make sure to consult Anti-patterns in Cassandra.

RHEL

This section describes how to install the latest *Cassandra 3.0.x* release on a *RHEL* based systems for *Newts*. The first step is to add the *DataStax* community repository and install the required *GPG Key* to verify the integrity of the *RPM packages*. After that install the package with *yum* and the *Cassandra* service is managed by *Systemd*.

> This description was built on *CentOS 7.2*.

> Cassandra 3.x requires Java 8+.

Add the DataStax repository

```
vi /etc/yum.repos.d/datastax.repo
```

Content of the datastax.repo file

```
[datastax]
name = "DataStax Repo for Apache Cassandra"
baseurl = https://rpm.datastax.com/community
enabled = 1
gpgcheck = 1
```

Install GPG key to verify RPM packages

```
rpm --import https://rpm.datastax.com/rpm/repo_key
```

Install latest Cassandra 3.0.x package

```
yum install dsc30
```

Enable Cassandra to start on system boot

```
chkconfig cassandra on
```

Start cassandra service

```
service cassandra start
```

> Verify whether the *Cassandra* service is automatically started after rebooting the server.

Debian

This section describes how to install the latest *Cassandra 3.0.x* release on a *Debian*-based system for *Newts*. The first step is to add the *DataStax* community repository and install the required *GPG Key* to verify the integrity of the *DEB packages*. After that install the packages with *apt* and the *Cassandra* service is added to the runlevel configuration.

This description was built on *Debian 8.3* and *Ubuntu 16.04 LTS*.

Cassandra 3.x requires Java 8+.

Add the DataStax repository

```
vi /etc/apt/sources.list.d/cassandra.sources.list
```

Content of the cassandra.sources.list file

```
deb https://debian.datastax.com/community stable main
```

Install GPG key to verify DEB packages

```
wget -O - https://debian.datastax.com/debian/repo_key | apt-key add -
```

Install latest Cassandra 3.0.x package

```
apt-get update
apt-get install dsc30
```

The *Cassandra* service is added to the runlevel configuration and is automatically started after installing the package.

Verify whether the *Cassandra* service is automatically started after rebooting the server.

Microsoft Windows

This section describes how to install the latest *Cassandra 3.0.x* release on a *Microsoft Windows Server* based systems for *Newts*. The first step is to download the graphical installer and register *Cassandra* as a *Windows Service* so it can be manged through the *Service Manager*.

This description was built on *Windows Server 2012*.

Cassandra 3.x requires Java 8+.

Download the DataStax graphical installer for Cassandra from PowerShell or a Browser

```
cd C:\Users\Administrator\Downloads
Invoke-WebRequest https://downloads.datastax.com/community/datastax-community-
64bit_3.0.6.msi -Outfile datastax-community-64bit_3.0.6.msi
```

Run the Windows Installer file from *PowerShell* or through *Windows Explorer* and follow the setup wizard to install. During the installation, accept the options to automatically start the services. By default the *DataStax Server*, *OpsCenter Server* and the *OpsCenter Agent* will be automatically installed and started.

> ℹ The *DataStax OpsCenter Server* is only required to be installed once per *Cassandra Cluster*.

> ❗ If you install the *DataStax OpsCenter* make sure you have *Chrome* or *Firefox* installed.

9.2.2. Configure OpenNMS Horizon

Once *Cassandra* is installed, *OpenNMS Horizon* can be configured to use *Newts*.

```
cat << EOF | sudo tee /opt/opennms/etc/opennms.properties.d/timeseries.properties
# Configure storage strategy
org.opennms.rrd.storeByForeignSource=true
org.opennms.timeseries.strategy=newts

# Configure Newts time series storage connection
org.opennms.newts.config.hostname=$ipaddress$
org.opennms.newts.config.keyspace=newts
org.opennms.newts.config.port=9042
EOF
```

> ℹ The `org.opennms.newts.config.hostname` property also accepts a comma separated list of hostnames and or IP addresses.

Once *Newts* has been enabled, you can initialize the *Newts* schema in *Cassandra* with the following:

Initialize Newts keyspace in Cassandra

```
${OPENNMS_HOME}/bin/newts init
```

Optionally, you can now connect to your *Cassandra* cluster and verify that the keyspace has been properly initialized:

Verify if the keyspace is initialized with cqlsh

```
cqlsh
use newts;
describe table terms;
describe table samples;
```

Restart *OpenNMS Horizon* to apply the changes.