Installation Guide

Copyright (c) 2015-2016 The OpenNMS Group, Inc.

OpenNMS Horizon 18.0.0-SNAPSHOT, Last updated 2016-04-13 20:59:39 EDT

Table of Contents

Chapter 1. Basic Installation of OpenNMS

The *OpenNMS* platform can be installed in several ways. This guide describes the installation of the platform on *RHEL-*, *Debian-* and *Microsoft Windows* based operation systems. Installable pre-compiled software packages are provided through *RPM* and *Debian* repository servers. Running *OpenNMS* requires the following components:

- Internet access to download and verify installation packages from public repository server
- Installed Oracle Java 8 environment
- PostgreSQL 9.1+ data base
- Set link to section which describes to install with RRDTool. Optional RRDtool to persist long term performance data

OpenJDK 8 can be used, but for production and critical environments *Oracle Java 8* is recommended.

`${OPENNMS_HOME}` is referred to the path *OpenNMS* is installed to. On *RHEL-based* systems it is `/opt/opennms` on *Debian-based* systems it is `/usr/share/opennms`. The environment in *Microsoft Windows* can refer to `C:\Program Files\opennms`

With the *opennms* meta package all dependencies needed for the components mentioned above are maintained. The following sections describe how to install *OpenNMS* on a single system. Dependencies for *Java* and the *PostgreSQL* data base are maintained with the *opennms* meta installation package.

1.1. Repositories for Releases

Installation packages are available for different releases of *OpenNMS*. The configuration of the repository decides which *OpenNMS* release will be installed.

The following releases are available for installation:

Table 1. OpenNMS release name convention

Release	Description
`stable`	Latest stable release
`testing`	Release candidate for next stable
`snapshot`	Latest successful develop build
`branches/${BRANCH-NAME}`	Install from a specific branch name, e.g. `branches/features-newts` installs the repository for the *Newts* development branch. Branches can be found in http://yum.opennms.org/branches/ or http://debian.opennms.org/dists/branches/

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

1.1.1. Specific Release on RHEL-based system

Installation of release specific repositories

```
rpm -Uvh http://yum.opennms.org/repofiles/opennms-repo-${RELEASE}-rhel7.noarch.rpm ①
rpm --import http://yum.opennms.org/OPENNMS-GPG-KEY
```

① Replace `${RELEASE}` with a release name like `testing` or `snapshot`.

Install *OpenNMS* with *YUM* following the normal installation procedure.

Installation of the full OpenNMS application with all dependencies

```
yum install opennms
```

Verify the release of *OpenNMS* packages with `yum info opennms`.

1.1.2. Specific Release on Debian-based system

Create a new apt source file (eg: `/etc/apt/sources.list.d/opennms.list`), and add the following 2 lines:

Package repository configuration for Debian-based systems

```
deb http://debian.opennms.org ${RELEASE} main ①
deb-src http://debian.opennms.org ${RELEASE} main ①
```

① Replace `${RELEASE}` with a release name like `testing` or `snapshot`.

Import the packages' authentication key with the following command:

GPG key import for Debian-based systems

```
wget -O - http://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -
```

Run `apt-get update` and install *OpenNMS* with *apt* following the normal installation procedure.

Verify the release of *OpenNMS* packages with `apt-cache show opennms`.

1.2. Installing on RHEL-based system

This section describes how to install the *OpenNMS* platform on *CentOS 7.1*. The setup process is described in the following steps:

1. Install *OpenNMS YUM* repository server with GPG key to verify packages
2. Installation of the *opennms* meta package which handles all dependencies
3. Initialize *PostgreSQL* database and configure access

4. Initialize *OpenNMS* and first start of the application

1.2.1. Setup OpenNMS YUM repository

Installation of stable repository and GPG key

```
rpm -Uvh http://yum.opennms.org/repofiles/opennms-repo-stable-rhel7.noarch.rpm
rpm --import http://yum.opennms.org/OPENNMS-GPG-KEY
```

1.2.2. Install OpenNMS package

Installation of the full application with all dependencies like PostgreSQL and Java

```
yum -y install opennms
```

The following packages will be automatically installed:

- *opennms*: The platform meta package which handles all dependencies from *OpenNMS* repository.
- *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* platform is installed in the following directory structure:

```
[root@localhost /opt/opennms]# tree -L 2
.
└── opennms
    ├── bin
    ├── contrib
    ├── data
    ├── deploy
    ├── etc
    ├── jetty-webapps
    ├── lib
    ├── logs -> /var/log/opennms
    ├── share -> /var/opennms
    └── system
```

1.2.3. Prepare PostgreSQL

The *CentOS* package installs but doesn't initialize the *PostgreSQL* database directory. Additionally *OpenNMS* requires authentication to access the database and are described in this section. Initialize the database directory with

Initialization of the PostgreSQL database

```
postgresql-setup initdb
```

System startup configuration for PostgreSQL

```
systemctl enable postgresql
```

Startup PostgreSQL database

```
systemctl start postgresql
```

The next step is setting the *postgres* super user password and creating an *opennms* database user with password. Additionally it is required to configure the authentication method to allow authentication from the local network.

Accounting and database management for OpenNMS

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

Set password for Postgres super user

```
su - postgres
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.

To allow *OpenNMS* access to the database over the local network *PostgreSQL* has to be configured.

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

Configuration of network access for PostgreSQL

```
host    all             all             127.0.0.1/32            md5 ①
host    all             all             ::1/128                 md5 ①
```

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

Apply configuration changes for PostgreSQL

```
systemctl reload postgresql
```

In the next step configure the *OpenNMS* database configuration.

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

Configuration for database authentication in OpenNMS

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

<jdbc-data-source name="opennms-admin"
                  database-name="template1"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/template1"
                  user-name="postgres" ③
                  password="** YOUR-POSTGRES-PASSWORD **" /> ④
```

① Set the user name to access the *OpenNMS* database table

② Set the password to access the *OpenNMS* database table

③ Set the *postgres* user for administrative access to PostgreSQL

④ Set the password for administrative access to PostgreSQL

1.2.4. Initialize OpenNMS

OpenNMS is now configured to access the database. It is required to set the *Java* environment running *OpenNMS* and initialize the database schema.

Configuration of Java environment for OpenNMS

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

Initialization of database and system libraries

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

System startup configuration for OpenNMS

```
systemctl enable opennms
```

Startup OpenNMS

```
systemctl start opennms
```

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*.

Change the default admin password to a secure password immediately.

1.3. Install on Debian-based systems

This guide does not apply to OpenNMS Meridian, which can be installed only on Red Hat Enterprise Linux or CentOS systems.

This section describes how to install the *OpenNMS* platform on *Ubuntu 14.04 LTS*. The setup process is described in the following steps:

1. Install *OpenNMS* apt repository server with GPG key to verify packages
2. Installation of the *opennms* meta package which handles all dependencies
3. Initialize *PostgreSQL* database and configure access
4. Initialize *OpenNMS* and first start of the application

1.3.1. Setup OpenNMS Debian repository

OpenNMS can be installed with Installation of stable repository and GPG key

Installation of OpenNMS Debian repository

```
deb http://debian.opennms.org stable main
deb-src http://debian.opennms.org stable main
```

Installation of repository GPG key

```
wget -O - http://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -
```

Update apt repository cache

```
apt-get update
```

1.3.2. Install OpenNMS package

Installation of the full application with all dependencies like PostgreSQL and Java

```
apt-get install -y opennms
```

The following packages will be automatically installed:

- *opennms*: The platform meta package which handles all dependencies from *OpenNMS* repository.
- *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* platform is installed in the following directory structure:

```
[root@localhost /usr/share/opennms]# tree -L 2
.
└── opennms
    ├── bin
    ├── data
    ├── deploy
    ├── etc -> /etc/opennms
    ├── instances
    ├── jetty-webapps
    ├── lib -> ../java/opennms
    ├── logs -> /var/log/opennms
    ├── share -> /var/lib/opennms
    └── system
```

1.3.3. Prepare PostgreSQL

The *Debian* package installs also *PostgreSQL* database and is already initialized and added in the runlevel configuration. It is only necessary to start the *PostgreSQL* database without a restart.

Startup PostgreSQL database

```
service postgresql start
```

The next step is creating an *opennms* database user with password and configure the authentication method.

Accounting and database management for OpenNMS

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

It is not necessary to change the authentication method in `pg_hba.conf`, it is by default set to `md5` for localhost connections.

Set password for Postgres super user

```
su - postgres
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.

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

Configuration for database authentication in OpenNMS

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

<jdbc-data-source name="opennms-admin"
                  database-name="template1"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/template1"
                  user-name="postgres" ③
                  password="** YOUR-POSTGRES-PASSWORD **" /> ④
```

① Set the user name to access the *OpenNMS* database table

② Set the password to access the *OpenNMS* database table

③ Set the *postgres* user for administrative access to PostgreSQL

④ Set the password for administrative access to PostgreSQL

1.3.4. Initialize OpenNMS

OpenNMS is now configured to access the database. It is required to set the *Java* environment running *OpenNMS* and initialize the database schema.

Configuration of Java environment for OpenNMS

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

Initialization of database and system libraries

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

It is not necessary to add *OpenNMS* to the run level manually, it is automatically added after setup.

Startup OpenNMS

```
service opennms start
```

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*.

Change the default admin password to a secure password immediately.

1.4. Install on Microsoft Windows Systems

This guide does not apply to OpenNMS Meridian, which can be installed only on Red Hat Enterprise Linux or CentOS systems.

OpenNMS is mostly developed on Unix/Linux based systems, nevertheless it is possible to install the platform on *Microsoft Windows* operating systems. To install the application a graphical installer is provided and can be used to install *OpenNMS* on *Microsoft Windows*. This section describes how to install the *OpenNMS* platform on *Microsoft Windows 2012 Server*.

The standalone installer for *Microsoft Windows* is only available for the most recent stable version of *OpenNMS*.

It is required to have Oracle JDK 8 installed. The *JRE* is **NOT** sufficient.

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

The setup process is described in the following steps:

1. Installation of *PostgreSQL* database service
2. Download and install the graphical *OpenNMS* installer
3. First start of the *OpenNMS* application

1.4.1. Installation 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. A version of *9.1+* is required for *OpenNMS*.

The following information has 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*.

The database data directory is automatically initialized during the setup and the *postgresql-x64-{PG-VERSION}* is already added as service and automatically started at system boot.

It is not necessary to change the authentication method in `pg_hba.conf`, it is by default set to `md5` for localhost connections.

1.4.2. 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.

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_51`
- 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.

There is currently an open issue in the installer NMS-7831. Username and password are not written to the `opennms-datasources.xml` file and has to be changed manually. The initialize of the database will fail with an authentication error.

Configuration for database authentication in OpenNMS

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

<jdbc-data-source name="opennms-admin"
                  database-name="template1"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/template1"
                  user-name="postgres" ③
                  password="** YOUR-POSTGRES-PASSWORD **" /> ④
```

① Set the user name to access the *OpenNMS* database table

② Set the password to access the *OpenNMS* database table

③ Set the *postgres* user for administrative access to PostgreSQL

④ Set the password for administrative changes of the *OpenNMS* database table

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.

After starting *OpenNMS* with the `start.bat` file 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*.

Change the default admin password to a secure password immediately.

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*.

Chapter 2. Installing Oracle Java Environment

Installing *Oracle Java 8* requires external installation packages. These packages are provided from *Oracle* or 3rd party maintainer for the *Debian* and *Ubuntu-based Linux Distributions*. The following tools should be installed to follow this installation manual:

- download files and tools with `wget` and `curl`
- extract archives with `tar`
- text manipulation with `sed`
- Editor, e.g. `vi`, `nano` or `joe`
- internet access

By downloading the *Oracle Java 8 RPM* installer you'll accept the license agreement from *Oracle* which can be found on the Java distribution web site.

2.1. Setup on RHEL-based systems

This section describes how to install *Oracle Java 8* on a *RPM-based* system like *Red Hat Enterprise Linux 7* or *CentOS 7.1*.

Download Oracle JDK RPM

```
wget --no-cookies \
     --no-check-certificate \
     --header \
              "Cookie: oraclelicense=accept-securebackup-cookie" \
              "http://download.oracle.com/otn-pub/java/jdk/8u45-b14/jdk-8u45-linux-
x64.rpm" \
     -O /tmp/jdk-8-linux-x64.rpm
```

Install Oracle JDK RPM file

```
yum install /tmp/jdk-8-linux-x64.rpm
```

2.2. Setup on *Debian-based* systems

This section describes how to install *Oracle Java 8* on a *Debian-based* system like *Debian 8* or *Ubuntu 14.04 LTS*.

Add Java repository from webupd8 maintainer

```
su -
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee
/etc/apt/sources.list.d/webupd8team-java.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a
/etc/apt/sources.list.d/webupd8team-java.list
```

Add repository key server and update repository

```
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
apt-get update
```

Install Oracle Java 8 installer

```
apt-get install -y oracle-java8-installer
```

2.3. Setup on Windows Server

This section describes how to install *Oracle Java 8* on a system running the *Microsoft Windows Server 2012* operating system.

Download the Microsoft Windows Java 8 installer with PowerShell or a browser

```
cd C:\Users\Administrator\Downloads
Invoke-WebRequest http://javadl.sun.com/webapps/download/AutoDL?BundleId=107944
-Outfile java8-installer.exe
```

Start the `java8-installer.exe` from the command line or with *Windows Explorer* from the Administrator's *Download* folder.

The setup requires administrative privileges.

2.4. Java Environment

To provide *Java*, applications 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_51`
- Debian: `/usr/lib/jvm/java-8-oracle`
- Windows Server 2012: `C:\Program Files\Java\jre1.8.0_51`

2.4.1. Set Java home in 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
```

2.4.2. Set Java home in Windows Server 2012

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 3. RRDtool as Time Series Database

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

3.1. RRDtool Installation

RRDtool can be installed from the official package repositories provided by *RHEL* and *Debian* based *Linux* distributions.

Installation on RHEL/CentOS

```
yum install rrdtool
```

Installation of RRDtool on Debian/Ubuntu

```
apt-get install rrdtool
```

If you want to install the latest *RRDtool* from source, make sure the `rrdtool` binary is in search path. To make the setup easier, you can link the binary to `/usr/bin/rrdtool` which is the location *OpenNMS* will expect the executable binary.

3.2. Install jrrd2 Interface

To get access from the *OpenNMS Java Virtual Machine* you have to install *jrrd2* as an interface. You can install it from the *OpenNMS* package repository with:

Installation of jrrd2 on RHEL/CentOS

```
yum install jrrd2
```

Installation of jrrd2 on Debian/Ubuntu

```
apt-get install jrrd2
```

With OpenNMS 17.0.0 it is preferred to use *jrrd2* instead of *jrrd*. The *jrrd2* module is improved for performance by adding multithreading capabilities.

3.3. Configuration of OpenNMS

To configure *OpenNMS* to use *RRDtool* instead of *JRobin* configure the following properties in `rrd-`

`configuration.properties`.

Configuration of RRDtool in OpenNMS on RHEL/CentOS

```
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
```

Configuration of RRDtool in OpenNMS on Debian/Ubuntu

```
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
```

OpenNMS expects the *RRDtool* binary in `/usr/bin/rrdtool`.

Table 2. 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`

Chapter 4. Installing Time Series database 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.

4.1. Setting up Cassandra

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 for evaluating and testing *Newts*. These steps are not suitable for a high performance production *Cassandra Cluster*. For further information see Cassandra Getting Started Guide. If you already have a running cluster you can skip this section.

4.1.1. Installing on RHEL-based systems

This section describes how to install the latest *Cassandra 2.1.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 *RHEL 7* and *CentOS 7.1*.

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 = http://rpm.datastax.com/community
enabled = 1
gpgcheck = 1
```

Install GPG key to verify RPM packages

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

Install latest Cassandra 2.1.x package

```
yum install dsc21
```

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.

4.1.2. Installing on Debian-based systems

This section describes how to install the latest *Cassandra 2.1.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* and *Ubuntu 14.04 LTS*.

Add the DataStax repository

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

Content of the cassandra.sources.list file

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

Install GPG key to verify DEB packages

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

Install latest Cassandra 2.1.x package

```
apt-get update
apt-get install dsc21=2.1.10-1 cassandra=2.1.10
```

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.

4.1.3. Installing on Windows Server systems

This section describes how to install the latest *Cassandra 2.1.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*.

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

```
cd C:\Users\Administrator\Downloads
Invoke-WebRequest http://downloads.datastax.com/community/datastax-community-
64bit_2.1.10.msi -Outfile datastax-community-64bit_2.1.10.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.

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.

4.2. Configure OpenNMS Horizon

Once *Cassandra* is installed, *OpenNMS Horizon* can be configured to use *Newts*. To enable and configure *Newts*, set the following properties in `${OPENNMS_HOME}/etc/opennms.properties`:

Configuration for OpenNMS Horizon

```
# 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
```

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.

Chapter 5. Installing R

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

OpenNMS interfaces with *R* via *stdin* and *stdout*, and for this reason, *R* must be installed on the same host. 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.

5.1. Installing on RHEL-based systems

This section describes how to install *R* on a *RHEL* based system.

This description was built on *RHEL 7* and *CentOS 7.1*.

Install the EPEL repositories

```
yum install epel-release
```

Install R

```
yum install R
```

5.2. Installing on Debian-based systems

This section describes how to install *R* on a *Debian*-based system.

This description was built on *Debian 8* and *Ubuntu 14.04 LTS*.

Install R

```
sudo apt-get install r-recommended
```

Chapter 6. Installing Minion

Minion enables enterprises with the ability to create a globally distributed and scalable monitoring fabric.

Support for *Minion* is currently experimental and packages are only available for RHEL based systems.

Before attempting to setup *Minion* you must have an instance of *OpenNMS* setup using the same version of the packages.

A *Minion* can be installed on the same system as *OpenNMS* or on other system systems provided that it can communicate with:

1. The OpenNMS REST interface
2. The ActiveMQ broker used by OpenNMS

OpenNMS embeds an *ActiveMQ* broker which is used by default, however the port is bound to `127.0.0.1`. In order to make the *ActiveMQ* broker accessible remotely, must edit `$OPENNMS_HOME/etc/opennms-activemq.xml` and configure the `transportConnector` to bind to `0.0.0.0` (or another suitable address):

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

6.1. Installing on RHEL-based systems

This section describes how to install *Minion* on a *RHEL* based system.

This description was built on *RHEL 7* and *CentOS 7.1*.

Start by setting up the OpenNMS YUM repository and installing Java.

The Minion currently requires a JDK. See HZN-620 for details.

Once the *YUM* repository has been configured:

Install the Minion packages

```
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

The *Minion* packages setup the following directory structure:

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

6.1.1. Initialize Minion

System startup configuration for Minion

```
systemctl enable minion
```

Startup Minion

```
systemctl start minion
```

After starting *Minion* the shell can be accessed locally on ssh://localhost:8201. The default login user is *admin* and the password is initialized to *admin*.

```
[root@localhost /root]# $ ssh -oPort=8201 admin@localhost
```

6.2. Configuring Minion

This section describes how to configure *Minion* once installed and started.

Once the *Minion* service is started and the *Karaf* shell is accessible, you can configure the *Minion* to point to your *OpenNMS* instance.

By default the *Minion* is configured to communicate with *OpenNMS* via `localhost`.

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

```
admin@minion()> config:edit org.opennms.minion.controller
admin@minion()> config:property-set http-url http://opennms-fqdn:8980/opennms
admin@minion()> config:property-set broker-url tcp://opennms-fdqn:61616
admin@minion()> config:property-set location RDU
admin@minion()> config:update
```

Configure the credentials to use when communicating with OpenNMS

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

Restart the Minion after updating the credentials

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

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

6.2.1. Verifying Connectivity

Once the URLs and credentials for communicating with the *OpenNMS* instance are configured, you can verify connectivity using:

Verify connectivity with the OpenNMS controller

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