Table of Contents

OpenNMS 1
Basic Configuration 1
Database 2
Verify 5
NFS 5
Cluster Applications 8

OpenNMS

Before start configuring the *RedHat Cluster Services*, we should configure *OpenNMS* and verify it works properly with the database cluster.

Optionally, it is also recommended to initialize a local *Git* repository on `/opt/opennms/etc` to track the changes performed on the configuration files.

To know more about *Git*:

https://git-scm.com/book/en/v2

Here is how to setup a local repository.

Be sure that *Git* is installed:

```
[root@onmssrv01 ~]# yum install git -y
[root@onmssrv02 ~]# yum install git -y
```

Then, Initialize the repository:

```
[root@onmssrv01 ~]# cd /opt/opennms/etc/
[root@onmssrv01 etc]# git init .
Initialized empty Git repository in /opt/opennms/etc/.git/
[root@onmssrv01 etc]# git add .
[root@onmssrv01 etc]# git commit -m "Default configuration for OpenNMS X.Y.Z"
```

Remember to replace X.Y.Z with the proper version of *OpenNMS*, for example *Horizon 16.0.2* or *Meridian 2015.1.0*, etc.

Keep in mind that initializing the repository must be done only on *onmssrv01*, as the configuration directory will be moved to a shared *NFS* server when it is ready.

Now, every time you change a file and verify the change works, commit the change to the local *Git* repository.

Basic Configuration

log4j.xml (Meridian)

In *Meridian 2015.0.1*, the default root level and the *defaultThreshold* are configured to be *WARN*. That means, nothing below that will be displayed. In order to use *INFO/DEBUG*, be sure the following lines has *DEBUG*:

```
<root level="DEBUG">
<DynamicThresholdFilter key="prefix" defaultThreshold="DEBUG">
```

opennms.conf

For the cluster services, it is required that the script used to initialize *OpenNMS* wait until *OpenNMS* starts successfully, otherwise this can confuse the cluster, and the cluster service will be bouncing between the cluster nodes.

Also, the *JVM* must have a minimum of 1GB in order to start *OpenNMS*. That means the minimum *RAM* on the OpenNMS machine must be 2GB. For production machines this might not be a problem, but it is important to keep this in mind.

This file doesn't exist by default, so you should create one with at least the following content:

```
Unresolved directive in opennms.adoc - include::../files/opennms/opennms.conf[]
```

IMPORTANT	Note that the start timeout is 3 minutes. In case *OpenNMS* is able to start in less time when it is fully loaded in average, the start timeout can be reduced. It might be the case where the time must be increased, so be sure the time is accurate before start configuring the cluster.
IMPORTANT	do not set `START_TIMEOUT` to be `0`, otherwise the cluster won't work. As mentioned, the timeout must be enough to be sure that *OpenNMS* is running without mistakes.

In case you need more *heap space*, go ahead and change it accordingly to the size of the network you're going to manage; assuming you're not using the *Vagrant VMs*.

Database

Proceed to configure *OpenNMS* to use *Pgpool-II* and initialize the *OpenNMS* database. This must be done only from one *OpenNMS* machine.

The first step is edit the `/opt/opennms/etc/opennms-datasources.xml`, file and make sure the two connections are pointing to the local *pgpool-II*, with the correct port and credentials, for example:

```
[root@onmssrv01 ~]# tail -n 14 /opt/opennms/etc/opennms-datasources.xml
  <jdbc-data-source name="opennms"
                    database-name="opennms"
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:9999/opennms"
                    user-name="opennms"
                    password="opennms" />

  <jdbc-data-source name="opennms-admin"
                    database-name="template1"
                    class-name="org.postgresql.Driver"
                    url="jdbc:postgresql://localhost:9999/template1"
                    user-name="postgres"
                    password="postgres" />
</data-source-configuration>
```

Then, execute the `runjava` script to tell *OpenNMS* where the Java *JDK* is installed:

```
[root@onmssrv01 ~]# /opt/opennms/bin/runjava -s
runjava: Looking for an appropriate JRE...
runjava: Checking for an appropriate JRE in JAVA_HOME...
runjava: skipping... JAVA_HOME not set
runjava: Checking JRE in user's path: "/usr/bin/java"...
runjava: found an appropriate JRE in user's path: "/usr/bin/java"
runjava: value of "/usr/bin/java" stored in configuration file

[root@onmssrv01 ~]# /usr/bin/java -version
java version "1.7.0_75"
Java(TM) SE Runtime Environment (build 1.7.0_75-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.75-b04, mixed mode)
```

If you're using *OpenNMS Horizon 16* or newer, you should see *Oracle JDK 8*, instead of *7*. This tutorial was designed for

Meridian, but should work without issues for *OpenNMS Horizon*.

Finally, initialize the *OpenNMS* database. Prior doing that, be sure that *pgpool-II* is running:

On *RHEL/CentOS 6*:

```
[root@onmssrv01 ~]# service pgpool start
```

On *RHEL/CentOS 7*:

```
[root@onmssrv01 ~]# systemctl start pgpool
```

Then, execute the install script:

```
[root@onmssrv01 ~]# /opt/opennms/bin/install -dis
=============================================================================
OpenNMS Installer
=============================================================================

Configures PostgreSQL tables, users, and other miscellaneous settings.

DEBUG: Platform is IPv6 ready: true
- searching for libjicmp.so:
  - trying to load /usr/lib64/libjicmp.so: OK
- searching for libjicmp6.so:
  - trying to load /usr/lib64/libjicmp6.so: OK
- searching for libjrrd.so:
  - trying to load /usr/lib64/libjrrd.so: OK
16:44:44.698 [Main] WARN  org.opennms.install.Installer - Could not create file:
/opt/opennms/etc/libraries.properties
- using SQL directory... /opt/opennms/etc
- using create.sql... /opt/opennms/etc/create.sql
16:44:44.707 [Main] INFO  org.opennms.core.schema.Migrator - validating database version
* using 'postgres' as the PostgreSQL user for OpenNMS
* using 'opennms' as the PostgreSQL database name for OpenNMS
16:44:44.785 [Main] INFO  org.opennms.core.schema.Migrator - validating database version
16:44:44.812 [Main] INFO  org.opennms.core.schema.Migrator - adding PL/PgSQL support to the database, if
necessary
16:44:44.825 [Main] INFO  org.opennms.core.schema.Migrator - PL/PgSQL call handler exists
16:44:44.827 [Main] INFO  org.opennms.core.schema.Migrator - PL/PgSQL language exists
- checking if database "opennms" is unicode... ALREADY UNICODE
- Checking for old import files in /opt/opennms/etc... DONE
Running migration for changelog: URL [jar:file:/opt/opennms/lib/org.opennms.core.schema-2015.1.0-
liquibase.jar!/changelog.xml]
...
INFO 7/13/15 4:44 PM:liquibase: Successfully acquired change log lock
INFO 7/13/15 4:45 PM:liquibase: Reading from databasechangelog
INFO 7/13/15 4:45 PM:liquibase: Reading from databasechangelog
INFO 7/13/15 4:45 PM:liquibase: Successfully released change log lock
- checking if iplike is usable... YES
- checking if iplike supports IPv6... YES
checking for stale eventtime.so references... OK
...
Installer completed successfully!

=============================================================================
OpenNMS Upgrader
=============================================================================
...
Upgrade completed successfully!
```

Most of the output is omitted to simplify the lecture, but you should see that the installation code can reach the database at the beginning.

If the credentials are not correct, you should see an exception complaining about the password:

```
Caused by: org.postgresql.util.PSQLException: ERROR: md5 authentication failed
  Detail: password does not match
```

If you see a different error, for example:

```
Caused by: org.postgresql.util.PSQLException: ERROR: pgpool is not accepting any new connections
  Detail: all backend nodes are down, pgpool requires atleast one valid node
  Hint: repair the backend nodes and restart pgpool
```

Restart the *pgpool* service and then use the `psql` command to be sure you can connect to the databases through *pgpool* on port 9999.

If the install script has been executed correctly, that means you've created the *OpenNMS* database successfully. You can now try to see if the database was successfully replicated on *pgdbsrv02*.

Verify

Start *OpenNMS* to be sure it works as expected:

On *RHEL/CentOS 6*:

```
[root@onmssrv01 ~]# service opennms start
```

On *RHEL/CentOS 7*:

```
[root@onmssrv01 ~]# systemctl start opennms
```

Use a *Web Browser* and verify that the *OpenNMS WebUI* works when opening the following *URI*:

http://onmssrv01:8980/opennms

After verifying that *OpenNMS* works, stop the applications:

On *RHEL/CentOS 6*:

```
[root@onmssrv01 ~]# service opennms stop
[root@onmssrv01 ~]# service pgpool stop
```

On *RHEL/CentOS 7*:

```
[root@onmssrv01 ~]# systemctl stop opennms
[root@onmssrv01 ~]# systemctl stop pgpool
```

IMPORTANT Do not make *opennms* or *pgpool* to start with the machine, as these applications will be controlled by the cluster services, not the operating system.

NFS

If you're using a test *VM* for the *NFS* server, make sure the *nfs* service is installed, configured and activated.

On *RHEL/CentOS 6*:

```
[root@nfssrv01 ~]# yum -y install nfs-utils rsync
[root@nfssrv01 ~]# chkconfig nfs on
[root@nfssrv01 ~]# service nfs start
```

On *RHEL/CentOS 7*:

```
[root@nfssrv01 ~]# yum -y install nfs-utils rsync
[root@nfssrv01 ~]# systemctl enable rpcbind nfs-server
[root@nfssrv01 ~]# systemctl start rpcbind nfs-server
```

We need to prepare the shared devices with the appropriate information and test it before create the cluster. The *NFS* server must have the following content on its `/etc/exports` file:

```
/opt/opennms/etc      192.168.205.0/24(rw,sync,no_root_squash)
/opt/opennms/share    192.168.205.0/24(rw,sync,no_root_squash)
/opt/opennms/pgpool   192.168.205.0/24(rw,sync,no_root_squash)
```

The above content is based on the *Vagrant lab*. On a real deployment, the networks must be different. The above paths must be accessible form the *OpenNMS* servers.

As mentioned before, the *postgres* user must exist on the *NFS* server:

```
[root@nfssrv01 ~]# mkdir /var/lib/pgsql
[root@nfssrv01 ~]# groupadd -r -g 26 postgres
[root@nfssrv01 ~]# useradd -r -u 26 -M -d /var/lib/pgsql -n -g postgres postgres
[root@nfssrv01 ~]# chown postgres:postgres /var/lib/pgsql/
```

Also, be sure that the domain for your deployment (in this case `local`) is declared in `/etc/idmapd.conf` on the *NFS* server and all *NFS* clients. In some cases, it might be required to restart the *NFS* services on the *NFS* server, and clear the *idmap* cache on the clients with `nfsidmap -c`.

After modifying `/etc/exports`, restart the nfs service:

On *RHEL/CentOS 6*:

```
[root@nfssrv01 ~]# service nfs restart
```

On *RHEL/CentOS 7*:

```
[root@nfssrv01 ~]# systemctl restart rpcbind nfs-server
```

We just make sure that the local copy of `/opt/opennms/etc`, `/var/opennms` and `/etc/pgpool-II/` are synchronized with the *NFS* server:

```
[root@onmssrv01 ~]# rsync -avr --delete /opt/opennms/etc/ nfssrv01:/opt/opennms/etc/
[root@onmssrv01 ~]# rsync -avr --delete /var/opennms/ nfssrv01:/opt/opennms/share/
[root@onmssrv01 ~]# rsync -avr --delete /etc/pgpool-II/ nfssrv01:/opt/opennms/pgpool/
```

If you cannot perform the synchronization, check the credentials and the target mount point with the administrators of the *NFS* server.

Remove all the content form the directories that will be mounted through *NFS*:

```
[root@onmssrv01 ~]# rm -rf /opt/opennms/etc/*
[root@onmssrv01 ~]# rm -rf /opt/opennms/etc/.git*
[root@onmssrv01 ~]# rm -rf /var/opennms/*
[root@onmssrv01 ~]# rm -rf /etc/pgpool-II/*

[root@onmssrv02 ~]# rm -rf /opt/opennms/etc/*
[root@onmssrv02 ~]# rm -rf /var/opennms/*
[root@onmssrv02 ~]# rm -rf /etc/pgpool-II/*
[root@onmssrv02 ~]# rm -rf /etc/pgpool-II/*
```

Make sure that the `nfs-utils` package is installed on the *OpenNMS* servers:

```
[root@onmssrv01 ~]# yum install nfs-utils -y
[root@onmssrv02 ~]# yum install nfs-utils -y
```

Mount the filesystems on each server to make sure that works.

```
[root@onmssrv01 ~]# mount -t nfs -o vers=4 nfssrv01:/opt/opennms/etc/ /opt/opennms/etc/
[root@onmssrv01 ~]# mount -t nfs -o vers=4 nfssrv01:/opt/opennms/share/ /var/opennms/
[root@onmssrv01 ~]# mount -t nfs -o vers=4 nfssrv01:/opt/opennms/pgpool/ /etc/pgpool-II/

[root@onmssrv02 ~]# mount -t nfs -o vers=4 nfssrv01:/opt/opennms/etc/ /opt/opennms/etc/
[root@onmssrv02 ~]# mount -t nfs -o vers=4 nfssrv01:/opt/opennms/share/ /var/opennms/
[root@onmssrv02 ~]# mount -t nfs -o vers=4 nfssrv01:/opt/opennms/pgpool/ /etc/pgpool-II/
```

IMPORTANT

make sure that the permissions on `/etc/pgpool-II/` are correct. If they are not correct (i.e. like showing nobody instead of *postgres* as the owner of the files), validate with the *NFS* administrators why it is not working. Besides the *nobody* user, you should see the following message on `/var/log/messages`:

```
Jul 16 17:48:04 onmssrv01 nfsidmap[5353]: nss_getpwnam: name 'nobody' does not map into domain 'local'
```

As a workaround, do not use the cluster resource associated with `/etc/pgpool-II/`, synchronize the `/etc/pgpool-II/` directory back to each *OpenNMS* server, and fix the permissions.

Now, unmount the filesystem on each server, as they will be managed by the cluster.

```
[root@onmssrv01 ~]# umount /opt/opennms/etc/
[root@onmssrv01 ~]# umount /var/opennms/
[root@onmssrv01 ~]# umount /etc/pgpool-II/

[root@onmssrv02 ~]# umount /opt/opennms/etc/
[root@onmssrv02 ~]# umount /var/opennms/
[root@onmssrv02 ~]# umount /etc/pgpool-II/
```

IMPORTANT

Do not declare the external filesystem on `/etc/fstab`, the mount/umount operation of the shared filesystems will be managed through the cluster services, not the operating system.

Cluster Applications

Do not enable the *opennms* service or the *pgpool* service to start with the operating system. Same restriction applies to all shared mounts.

The reason for this is because these resources will be managed through the cluster, not the operating system.