Table of Contents

Upgrading OpenNMS on a cluster environment

PostgreSQL

The only possible way to perform a live upgrade (i.e. without shutting down *pgpool-II* or the *OpenNMS* cluster) is when upgrading to the same code base. For example, this document was designed for PostgreSQL 9.4, so you can upgrade to any 9.4 release but not to a newer release like 9.5. Upgrading to a 9.5 is not considered an upgrade, it is considered a migration because the data structure can be different.

If we're talking about the same code base, here is the procedure:

Standby server

1. Log into the standby server (check the notes to verify which database server is not the master/primary server).
2. Shutdown the postgresql application
3. Perform the upgrade of the postgresql packages (or the full operating system if apply).
4. Reboot the server, to ensure it will properly starts.
5. Verify that *postgresql* is running after the reboot and it is still registered as standby/slave with *repmgr*.
6. Log into the active *OpenNMS* server and re-attach the standby server to *pgpool-II*.

Primary server

Perform a failover to promote the standby server. Once the secondary has been promoted, perform the upgrade using the procedure for the standby.

Pgpool-II

Pgpool-II is mandatory in order to run *OpenNMS*. For this reason, you can only upgrade a standby server, on a similar way as explained for the database.

Like the database, it is recommended to keep the code base of the version you're using. On this document the version 3.4.2 is used. That means, you can upgrade to any 3.4 version, but not to 3.5 or higher, as the configuration file could contain changes not compatible with the old version. For this reason, upgrading to 3.5 or higher is considered a migration.

Standby server

1. Log into the standby server (use the cluster commands to verify that *pgpool/opennms* are not running).
2. Perform the upgrade of the *pgpool* packages (or the full operating system if apply).
3. Reboot the server, to ensure it will properly starts.
4. Remove the content of `/etc/pgpool-II`, or move it to a different location.

Primary server

Use the cluster command to migrate the cluster resources to the standby machine. Once all the services are running on the other server, upgrade the packages using the procedure for the standby.

Meridian

The main reason why I separated the upgrade procedure of *Meridian* and *Horizon*, is because by default, *Meridian* is a long term support release. That means, upcoming upgrade for the code base you're using will be available for several years, which is not the case of *Horizon*. The following procedure assumes that you're not planing to upgrade to a major version of *Meridian*.

Standby Server

Log into the standby server (use the cluster commands to verify that *pgpool/opennms* are not running). Perform the upgrade of the *Meridian* packages (be sure that the licensing is valid, otherwise the access to the *Meridian* repository will be denied). At this time, the `/opt/opennms/etc` and `/var/opennms` has been populated with the files on the version you've installed. It is important to understand which are the changes performed on the configuration files in order to merge them into the production configuration.

Mount temporarily the `/opt/opennms/share` directory from the *NFS* server into the upgraded machine, for example: `/mnt/onms_var`. Execute `diff -ur /var/opennms/etc-pristine/ /mnt/onms_var > /mnt/config-changes.diff` Check the `config_changes.diff` file to understand the changes added from the version of *Meridian* you used to have and the current one you're installing. If the content doesn't have conflicts with your configuration changes, continue with (d). Otherwise, go to (4) Mount temporarily the `/opt/opennms/etc` directory from the *NFS* server into the upgraded machine, for example: `/mnt/onms_etc`. Copy the content of `/mnt/onms_etc` to another location to avoid change the files on the *NFS server* (because the active server is still using them): `rsync -avr /mnt/onms_etc /tmp/onms_etc`. Go to `/tmp/onms_etc` Execute: `patch -p1 < /tmp/config-changes.diff`, and analyze the results. If there are no conflicts, `/tmp/onms_etc` contains your changes and the new changes. If not, you can either fix the conflicts manually, or discard the changes. If you have fixed the conflicts, or the patch was applied without errors, you can copy the changes back to the *NFS* mount point. I know the new server is still using it but it won't be a problem if no one is making changes on the active *OpenNMS* machine. Unmount the temporary filesystems if any. Reboot the server, to ensure it will properly starts. Remove the content of `/opt/opennms/etc` and `/var/opennms`, or move them to a different location.

Primary Server

Use the cluster command to migrate the cluster resources to the standby machine. Once all the services are running on the other server, upgrade the packages using the procedure for the standby.

Horizon

Horizon has a very aggressive release cycle, an upgrade for a major version to another is considered a migration not an upgrade, for example upgrading from 14 to 15 or from 15 to 17, etc.

If you're planning to keep the same code base, you can follow the procedure explained for *Meridian*; otherwise you most follow the migration procedure.

OpenNMS Install Script

As you probably noticed, the procedure explained above doesn't include the execution of the install script (or the execution of the upgrade script with OpenNMS up and running). This is because that will alter the database which might introduce potential problems on the currently running machine.

For this reason if this is not a migration (as explained), you can run the install script after you upgrade both machines, even if *OpenNMS* is running. In theory this should not perform drastic changes, as you're on the same code base.

To avoid potential problems:

1. Stop the cluster.
2. Mount the resources manually on one machine.
3. Execute the install script.
4. Umount the resources.
5. Start the cluster.

Migration

If you want to upgrade the *DB*, *Pgpool*, or *OpenNMS* from one major version to another, you must stop the entire cluster and perform an offline upgrade. The reason for this is that the data from one version could not be compatible with the newer major release.

In the case of *PostgreSQL*, you must create a full binary dump and then restore it after upgrading *PostgreSQL*, or use the command line tools for the same purpose. *PostgreSQL* intentionaly uses different directory names for each major version precisely because the data directory is not compatible. So, you currently have the data on `/var/lib/pgsql/9.4` and *9.5* will put the data on `/var/lib/pgsql/9.5`. Follow the recommendation at http://www.postgresql.org to upgrade your *PostgreSQL* server. Do a similar thing for *PgPool*.

For *OpenNMS*, it is extremely recommended that you prepare your configuration files for the new version. This can be done offline without touching the current cluster, but you will need a spare server.

Because the amount of changes between a major version and the next one can be massive. The procedure to apply the changes and prepare the configuration files is different.

The first thing you should do is go to the active *OpenNMS* server (or the *NFS* server), and perform a *DIFF* between the default configuration files and your current files:

`diff -ur /var/opennms/etc-pristine/ /opt/opennms/etc/ > /tmp/opennms-current-config-changes.diff`

This will give you the changes you've done in *OpenNMS* since it was installed.

Now, on the test machine, install the new version of *OpenNMS* you're planning to use in production. It is important to use the same packages you've used, as your installation might have additional packages installed. To know which packages have been installed:

For *Horizon*:

```
rpm -qa | grep opennms
```

For *Meridian*:

```
rpm -qa | grep meridian
```

Then copy the `opennms-current-config-changes.diff` file from the active production machine (or the *NFS* server) to the test machine, and apply the changes to `/opt/opennms/etc` directory.

```
cd /opt/opennms/etc
patch -p1 < /tmp/opennms-current-config-changes.diff
```

Considering how aggressive could be the changes in *Horizon*, do not expect the above command will pass cleanly. You might need to do a lot of manual merge.

For *Meridian*, even if it is a major release, the amount of manual changes might be a lot less than Horizon due to the way on which the configuration files have been created for *Meridian*.

When you have the configuration files merged without problems, start *OpenNMS* on the test machine to ensure it works. In case the test server doesn't have the same settings like the production server, be sure to remove `opennms.conf` before start *OpenNMS*.

If everything works fine, you can perform the migration.

1. Stop the cluster.
2. Rsync all the files from `/opt/opennms/etc` on the test server to the proper location on the shared device (i.e. the *NFS* server).
3. Upgrade OpenNMS on each cluster node.
4. Remove everything from `/opt/opennms/etc` and `/var/opennms` on each node.
5. On one server, mount the shared resources manually and start *pgpool-II*.
6. Execute the install script. If everything went well, mount the resources umount the resources.
7. Start the cluster. Now, if you want to be 100% sure that the above procedure won't have issues, you can create a copy of the production database, and then test the install script on the test server prior performing the migration.

Having the files ready on *NFS* after stopping the cluster, and pre-testing the schema update will minimize the downtime of the cluster.

If you have an *OpenNMS Support Contract*, we can help with the merge procedure, either if you're performing a simple upgrade or a migration.