Table of Contents

Pre-requisites

The RHEL/CentOS Linux Distribution will be used for this solution. The instruction provides details for the last 2 mayor versions: 6 and 7.

For references purposes, the following machines will used:

Name	IP address
onmssrv01	192.168.205.151
onmssrv02	192.168.205.152
pgdbsrv01	192.168.205.153
pgdbsrv02	192.168.205.154
nfssrv01	192.168.205.155

The virtual IP will be:

onmssrvvip : `192.168.205.150`

The IP addresses must be configured statically on each machine prior starting configuring the cluster.

Testing Lab Setup

Ignore this section if you want to use real servers for this deployment.

For the testing environment the following tools must be downloaded and installed on your system:

Vagrant (www.vagrantup.com) VirtualBox (www.virtualbox.org)

Your testing machine must have at least 8 GB of RAM with a quad-core processor, as the testing environment will require 5 VMs, and two of them requires 2GB, for a total of 7GB.

Create a directory and inside of it create a file called Vagrantfile, with the following content:

```
Unresolved directive in prerequisites.adoc - include::../files/prerequisites/Vagrantfile[]
```

The above file will setup 5 machines using *CentOS 6.6*. In order to use *CentOS 7*, replace the value of the `config.vm.box` attribute from `chef/centos-6.6` to `chef/centos-7.1`. If you want to use *CentOS 6* on some machines and *CentOS 7* on other machines, just add a `:box` attribute on the *JSON* configuration for the boxes on which you want a different OS than the default (the default box is `config.vm.box`).

To startup the VMs, execute the following command:

```
vagrant up
```

This will take several minutes to download the base image, create the 5 machines, and execute the provisioning on each of them.

To *SSH* a specific box, use the `vagrant ssh` command follow by the name of the VM, for example:

```
vagrant ssh onmssrv01
```

Vagrant automatically generates a user called `vagrant` with *sudo* access without password.

As almost all the commands used on this tutorial requires *root* access, you could either become *root* (and change the *root* password on each VM), or use the *vagrant* user with `sudo` for all the commands.

By default, *SELinux* is configured as *permissive* in `/etc/sysconfig/selinux`. For full security, it is recommended to change it to be *enforcing* and restart the VMs, as this is the default option in *RHEL/CentOS 6* and *7*.

To restart all the VMs with Vagrant:

```
$ vagrant reload
```

Also, the internal firewall must be enabled by default. There is a section designated to explain how to properly configure the internal firewall on each *Linux* distribution.

Hostnames

Before we will get started with the setup, it's important that all nodes in the setup can connect to each other via their hostname. This could be achieve by either adding the involved hosts to the `/etc/hosts` file, or use the *FQDN* registered on the *DNS* servers.

If *DNS* is not an option, the `/etc/hosts` should look like the following on each server:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.205.151  onmssrv01.local      onmssrv01
192.168.205.152  onmssrv02.local      onmssrv02
192.168.205.153  pgdbsrv01.local      pgdbsrv01
192.168.205.154  pgdbsrv02.local      pgdbsrv02
192.168.205.155  nfssrv01.local       nfssrv01
```

Be sure that the domain is properly declared at `/etc/idmapd.conf` for all the hosts, including the *NFS* server. In this particular case, the domain is `local`:

```
[root@nfssrv01]# more /etc/idmapd.conf
[General]
#Verbosity = 0
# The following should be set to the local NFSv4 domain name
# The default is the host's DNS domain name.
Domain = local
```

To restart all the VMs with *Vagrant*:

```
$ vagrant reload
```

PostgreSQL Repository

The standard *RHEL/CentOS* repositories do not contain the packages for *pgpool-II* or *repmgr*. Also, the supported *PostgreSQL* version for *RHEL/CentOS 6* is *8.4*, and for *RHEL/CentOS 7* is *9.2*.

Because this solution required the usage of *PostgreSQL 9.4*, *repmgr 3.0* and *pgpool-II 3.4*, we need to add the *PGDG* repository.

The *PGDG* provides packages for *pgpool-II*, but we are going to use a different repository because I found a dependency issue with the *RPM* provided on the *PGDG* repository.

PGDG is a project that maintains *RPM* builds of *PostgreSQL* and related components. You can find more information and repo-URL's for *PGDG* here:

http://yum.postgresql.org/repopackages.php

The *pgpool-II* maintainers, provides information about their *YUM* repository here:

http://www.pgpool.net/mediawiki/index.php/Yum_Repository

Let's add the *PGDG* repo on all nodes:

```
[root@pgdbsrv01 ~]# yum install http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-
1.noarch.rpm -y
[root@pgdbsrv02 ~]# yum install http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-
1.noarch.rpm -y
[root@onmssrv01 ~]# yum install http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-
1.noarch.rpm -y
[root@onmssrv02 ~]# yum install http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-
1.noarch.rpm -y
```

For *RHEL/CentOS 7*, replace `rhel-6` with `rhel-7`.

As mentioned before, we're going to use the following repository for *pgpool-II*, which must be installed only on the *Pgpool-II* server (in this case, the *OpenNMS* servers).

```
[root@onmssrv01 ~]# yum install http://www.pgpool.net/yum/rpms/3.4/redhat/rhel-6-x86_64/pgpool-II-release-
3.4-1.noarch.rpm -y
[root@onmssrv02 ~]# yum install http://www.pgpool.net/yum/rpms/3.4/redhat/rhel-6-x86_64/pgpool-II-release-
3.4-1.noarch.rpm -y
```

For *RHEL/CentOS 7*, replace `rhel-6` with `rhel-7`.

To avoid version conflicts between the two versions of *pgpool-II* and *libmemcached*, edit the `/etc/yum.repos.d/pgdg-94-centos.repo` file (i.e. the *PGDG* repository), to exclude *libmemcached* by adding the following line at the end of the section called `pgdb94`, on the *OpenNMS* servers:

```
[pgdg94]
name=PostgreSQL 9.4 $releasever - $basearch
baseurl=http://yum.postgresql.org/9.4/redhat/rhel-$releasever-$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG-94
exclude=libmemcached*
```

If you forget this step, you'll get errors when executing `yum update`.

OpenNMS Repository

Horizon

The first step is to install the *opennms-repo RPM* appropriate for your distribution. This contains the information *YUM* needs to get *OpenNMS* package information for installing.

To do so, find the appropriate release *RPM* from yum.opennms.org.

Then all you should need to do is install that repo package (as `root`):

```
[root@onmssrv01 ~]# yum install http://yum.opennms.org/repofiles/opennms-repo-stable-rhel6.noarch.rpm -y
[root@onmssrv01 ~]# yum install http://yum.opennms.org/repofiles/opennms-repo-stable-rhel6.noarch.rpm -y
```

For *CentOS/RHEL 7*, replace `rhel6` with `rhel7`.

This solution has been tested with *OpenNMS Horizon 14*, *15* and *16*.

Meridian

When you buy a *Meridian* license, you'll get the credentials to access the private repository that contains the *RPM* packages. Once you have the credentials, create a *YUM* repository file called `/etc/yum.repos.d/opennms-meridian.repo`, with the following content on each *OpenNMS* server:

```
[meridian]
name=Meridian for Red Hat Enterprise Linux and CentOS
baseurl=https://username:password@meridian.opennms.com/packages/2015/stable/rhel$releasever
enabled=1
gpgcheck=1
gpgkey=http://yum.opennms.org/OPENNMS-GPG-KEY
```

Replace `username` and `password` with the proper credentials.

This solution has been tested with *OpenNMS Meridian 2015.1.0*.

RPM Packages Installation

Database Packages

The *Pgpool-II* servers (i.e. the *OpenNMS* servers) are not going to run *PostgreSQL*, but it is recommended to install the `postgresql` packages on these servers. For this reason, install the following packages on the *pgpoo-II/OpenNMS* servers:

```
[root@onmssrv01 ~]$ yum install postgresql94 postgresql94-server postgresql94-libs pgpool-II-pg94 rsync -y
  [root@onmssrv02 ~]$ yum install postgresql94 postgresql94-server postgresql94-libs pgpool-II-pg94 rsync
-y
```

On the database servers, install the packages for PostgreSQL Server and repmgr:

```
[root@pgdbsrv01 ~]$ yum install postgresql94 postgresql94-server postgresql94-contrib repmgr94 rsync -y
[root@pgdbsrv02 ~]$ yum install postgresql94 postgresql94-server postgresql94-contrib repmgr94 rsync -y
```

At this point, the `postgres` user must exist on all the 4 machines. You can check it with the following command:

```
[root@pgdbsrv01 ~]$ # grep postgres /etc/passwd
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
```

The user will be created automatically as part of the installation of the `postgresql94-server` package.

IMPORTANT Do not make *Pgpool-II* to start with the operating system, because it will be managed by the cluster services.

OpenNMS Packages

Starting with *OpenNMS Horizon 16*, *OpenNMS* requires *Oracle JDK 8*. For *Meridian* and older versions of *Horizon*, *Oracle JDK 7* is required. The appropriate *JDK* will be installed with *OpenNMS*, as they are provided through the *OpenNMS YUM* repositories.

For Horizon:

```
[root@onmssrv01 ~]# yum install opennms-core opennms-webapp-jetty -y
[root@onmssrv01 ~]# yum install 'perl(LWP)' 'perl(XML::Twig)' -y
[root@onmssrv02 ~]# yum install opennms-core opennms-webapp-jetty -y
[root@onmssrv02 ~]# yum install 'perl(LWP)' 'perl(XML::Twig)' -y
```

If you're planning to use *RRDtool* with *Horizon* and you're using *RHEL/CentOS 6*, it is recommended to download the `rrdtool` and `rrdtool-perl` packages (version 1.4.7) from http://packages.express.org/rrdtool/, instead of using the packages from the distribution. For *RHEL/CentOS 7*, use the `rrdtool` packages available for the distribution.

For Meridian:

```
[root@onmssrv01 ~]# yum install meridian-core meridian-webapp-jetty -y
[root@onmssrv01 ~]# yum install 'perl(LWP)' 'perl(XML::Twig)' -y
[root@onmssrv02 ~]# yum install meridian-core meridian-webapp-jetty -y
[root@onmssrv02 ~]# yum install 'perl(LWP)' 'perl(XML::Twig)' -y
```

Meridian provides *RRDtool* and it is enabled by default.

IMPORTANT Do not make *OpenNMS* to start with the operating system, because it will be managed by the cluster services.

RedHat Cluster Packages

The *High Availability YUM group* must be installed on each cluster member (i.e. the *OpenNMS* machines):

```
[root@onmssrv01 ~]# yum groupinstall "High Availability" -y
[root@onmssrv02 ~]# yum groupinstall "High Availability" -y
```

IMPORTANT By default, the above group is going to install the default cluster services for the selected *Linux* distribution. For *CentOS/RHEL 6*, *CMAN/Ricci*; for *CentOS/RHEL 7 Pacemaker*.

NFS Packages

On all the servers, make sure the following packages are installed:

- `nfs-utils`
- `nfs-utils-lib`

Then, on the *NFS* server, enable and start the *NFS* service:

On *RHEL/CentOS 6*:

```
[root@nfssrv01 ~]# chkconfig nfs on
[root@nfssrv01 ~]# service nfs start
```

On *RHEL/CentOS 7*:

```
[root@nfssrv01 ~]# systemctl enable nfs-server
[root@nfssrv01 ~]# systemctl start nfs-server
```

OS Updates

It is always recommended to keep the OS up to date, so at this point it is a good time to perform a *YUM* update on all the servers, and reboot them.

```
[root@onmssrv01 ~]# yum update -y
[root@onmssrv02 ~]# yum update -y
[root@pgdbsrv01 ~]# yum update -y
[root@pgdbsrv02 ~]# yum update -y
```

After updating all packages, it is recommended to reboot all machines. If you're using the *Vagrant lab*, use the following command:

```
$ vagrant reload
```

Customization of Initialization Scripts

Pgpool and *OpenNMS* expects that *PostgreSQL* is up and running, and all the connections are available prior starting the services.

To avoid potential problems with the *OpenNMS* cluster when a machine dies, the following steps are necessary.

On each *OpenNMS* server, create a script called `dbcleanup` at `/usr/local/bin/` with the following content:

```
Unresolved directive in prerequisites.adoc - include::../files/prerequisites/dbcleanup.sh[]
```

WARNING Update the variables according to your environment. Make sure the permissions on the file are 0700:

```
[root@onmssrv01 ~]# chmod 0700 /usr/local/bin/dbcleanup.sh
[root@onmssrv02 ~]# chmod 0700 /usr/local/bin/dbcleanup.sh
```

On each *PostgreSQL* server, create a script called `cleanup.sh` at `/var/lib/psql/9.4/` with the following content:

```
Unresolved directive in prerequisites.adoc - include::../files/prerequisites/cleanup.sh[]
```

Make sure up update the ownership and the permissions on the file:

```
[root@pgdbsrv01 ~]# chmod 0700 /var/lib/pgsql/9.4/cleanup.sh
[root@pgdbsrv01 ~]# chown postgres:postgres /var/lib/pgsql/9.4/cleanup.sh

[root@pgdbsrv02 ~]# chmod 0700 /var/lib/pgsql/9.4/cleanup.sh
[root@pgdbsrv02 ~]# chown postgres:postgres /var/lib/pgsql/9.4/cleanup.sh
```

Now, it is important to tune the initialization scripts properly.

SysV Initialization Scripts (RHEL/CentOS 6)

Edit the file `/etc/init.d/pgpool`, and inside the start function, call the `dbcleanup.sh` script before start *pgpool*. Here is how the content should look like:

```
Unresolved directive in prerequisites.adoc - include::../files/prerequisites/pgpool[]
```

WARNING Be careful when upgrading the installed packages because the files modified above might be overridden and will require to be updated again.

Systemd Initialization Scripts (RHEL/CentOS 7)

The *systemd* initialization script for *pgpool-II* is not correct and it must be fixed prior start using *pgpool*. Edit the `/lib/systemd/system/pgpool.service` file and make sure the content looks like the following:

```
Unresolved directive in prerequisites.adoc - include::../files/prerequisites/pgpool.service[]
```

Then, edit `/usr/lib/tmpfiles.d/pgpool-II-pg94.conf` and make sure it contents look like the following:

```
Unresolved directive in prerequisites.adoc - include::../files/prerequisites/pgpool-II-pg94.conf[]
```

To avoid reboot the system, fix the ownership of the directory:

```
[root@onmssrv01 ~]# chown postgres:postgres /var/run/pgpool
```

Similarly, by default the *systemd* initialization script for *OpenNMS* will force `START_TIMEOUT=0`. In order to avoid potential problems when checking the *OpenNMS* state, it is better to disable this option.

Edit the `/usr/lib/systemd/system/opennms.service` file, remove the `-Q` from the `ExecStart` method; add `TimeoutStartSec=0` and replace `postgresql` with `pgpool`. The content should look like this:

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

Also, it is important to update the default options used by `pgpool` when it initializes. These options are defined on `/etc/sysconfig/pgpool`. Be sure they look like the following:

```
OPTS=" -n --discard-status"
```

After updating the files, execute the following command:

```
[root@onmssrv01 ~]# systemctl daemon-reload
```

Remember to do the same on *onmssrv02*.

WARNING Be careful when upgrading the installed packages because the files modified above might be overridden and will require to be updated again.

Public Key Authentication

All nodes must connect to each other over *SSH* without a password prompt for the *postgres* user. *SSH* will be used to *rsync* the data from the primary to the standby, and to initiate a failover from *pgpool-II* servers. Password-less *SSH* can be achieved with public key authentication.

On each of the 4 servers (both *OpenNMS* servers and both *DB* servers), generate a new *SSH* key for the *postgres* user without passphrase:

```
[root@pgdbsrv01 ~]# su - postgres -c "ssh-keygen -t rsa"
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/pgsql/.ssh/id_rsa):
Created directory '/var/lib/pgsql/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/pgsql/.ssh/id_rsa.
Your public key has been saved in /var/lib/pgsql/.ssh/id_rsa.pub.
The key fingerprint is:
22:fd:26:d2:0b:a4:30:62:a8:99:37:03:b9:2e:25:89 postgres@pgdbsrv01.local
```

Once you have the *SSH* keys, to allow all machines to connect to each other an accept each other's key, we'll need to add the generated public keys from all hosts to `/var/lib/pgsql/.ssh/authorized_keys`.

To show the public *RSA-key*, use the following command on each server:

```
[root@pgdbsrv01 ~]$ cat /var/lib/pgsql/.ssh/id_rsa.pub
```

Then, create the `authorized_keys` file, and put all the public keys inside (one per line). The content of the file should look like this:

```
[root@pgdbsrv01 ~]# cat /var/lib/pgsql/.ssh/authorized_keys
ssh-rsa AAAAB3...== postgres@onmssrv01.local
ssh-rsa AAAAB3...== postgres@onmssrv02.local
ssh-rsa AAAAB3...== postgres@pgdbsrv01.local
ssh-rsa AAAAB3...== postgres@pgdbsrv02.local
```

The actual keys have been omitted for simplification purposes.

Since we really want unattended access (so no password or any other question), I'll also add all hosts to the `known_hosts`-file. This prevents the question to add the hosts fingerprint on the first connection (be sure to execute the following command on all the servers):

```
[root@pgdbsrv01 ~]# su - postgres -c "ssh-keyscan -H {onmssrv01,onmssrv02,pgdbsrv01,pgdbsrv02}| tee
~/.ssh/known_hosts"
```

The `authorized_keys` and `known_host` on *pgdbsrv01* are fine for all other hosts. Use `scp` to copy these files to the other nodes:

```
[root@pgdbsrv01 ~]# scp /var/lib/pgsql/.ssh/{authorized_keys,known_hosts} pgdbsrv02:/var/lib/pgsql/.ssh/
[root@pgdbsrv01 ~]# scp /var/lib/pgsql/.ssh/{authorized_keys,known_hosts} onmssrv01:/var/lib/pgsql/.ssh/
[root@pgdbsrv01 ~]# scp /var/lib/pgsql/.ssh/{authorized_keys,known_hosts} onmssrv02:/var/lib/pgsql/.ssh/
```

Be sure the to fix the permissions of the files on each server:

```
[root@pgdbsrv01 ~]# chmod 600 /var/lib/pgsql/.ssh/authorized_keys
[root@pgdbsrv01 ~]# chown postgres:postgres -R /var/lib/pgsql/.ssh/*
[root@pgdbsrv01 ~]# restorecon -R  /var/lib/pgsql/
```

It's a good thing to test the authentication now, by just trying to *SSH* between all nodes (as the *postgres* user). You shouldn't get any prompt at all and be presented with the other host's command prompt:

```
[root@pgdbsrv01 ~]# su - postgres
-bash-4.1$ ssh onmssrv01
Last login: Mon Jul 13 09:22:42 2015 from 192.168.205.163
-bash-4.1$ hostname
onmssrv01.local
-bash-4.1$ exit
logout
Connection to onmssrv01 closed.
-bash-4.1$ ssh onmssrv02
Last login: Mon Jul 13 09:22:53 2015 from 192.168.205.163
-bash-4.1$ hostname
onmssrv02.local
-bash-4.1$ exit
logout
Connection to onmssrv02 closed.
-bash-4.1$ ssh pgdbsrv02
Last login: Mon Jul 13 09:23:27 2015 from 192.168.205.163
-bash-4.1$ hostname
pgdbsrv02.local
-bash-4.1$ exit
logout
Connection to pgdbsrv02 closed.
-bash-4.1$ exit
logout
```

Repeat the above set of commands for each host on each server to be sure that *postgres* user can *SSH* without passwords.

Internal Firewall

RHEL/CentOS 6

The first thing to do is enable and start `iptables` on all servers:

```
[root@onmssrv01 ~]# /etc/init.d/iptables start
[root@onmssrv01 ~]# /etc/init.d/iptables save
[root@onmssrv01 ~]# chkconfig iptables on
```

Now, the file `/etc/sysconfig/iptables` should exist. Edit this file on both servers, and add the following content before the `COMMIT` instruction and save the file:

```
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
```

The above instructions are the default content for a standard *CentOS 6* box, so it could be possible that the above lines already exist on that file. But, if you're using the *Vagrant lab*, *iptables* is not enabled or configured by default, the lines are needed.

Copy `/etc/sysconfig/iptables` from *onmssrv01* to *onmssrv02*, *pgdbsrv01* and *pgdbsrv02*. Then restart the *iptables* service on all the machines. This will ensure that the basic firewall configuration is the same on all servers.

The following are the ports to be opened for OpenNMS:

Table 1. Opened ports for OpenNMS

Port	Protocol	Description
5817	TCP	OpenNMS event listener
8980	TCP	OpenNMS WebUI
18980	TCP	OpenNMS JMX Management (internal monitoring)
8181	TCP	OpenNMS JMX Configuration
9999	TCP	pgpool-II (PostgreSQL cluster)
162	UDP	SNMP Traps (To receive traps in OpenNMS)

RHEL 6 uses *CMAN/Ricci* and optionally *luci* for managing the cluster. The following are the ports to open on each cluster member (i.e. the *OpenNMS* machines):

Table 2. Opened ports for Cluster management

Port	Protocol	Description
5404, 5405	UDP	corosync/cman (Cluster Manager)
11111	TCP	ricci (propagates updated cluster information)
21064	TCP	dlm (Distributed Lock Manager)
16851	TCP	modclusterd
8084	TCP	*Optional*: luci (WebUI to configure the cluster)

On each *OpenNMS* servers, edit the `/etc/sysconfig/iptables` file and add the following rules before the first `REJECT` entry:

```
|----
-A INPUT -p udp -m multiport --dports 5404,5405 -j ACCEPT
-A INPUT -p tcp --dport 11111 -j ACCEPT
-A INPUT -p tcp --dport 21064 -j ACCEPT
-A INPUT -p tcp --dport 16851 -j ACCEPT
-A INPUT -p tcp --dport 8084 -j ACCEPT
```

-A INPUT -p tcp --dport 5817 -j ACCEPT -A INPUT -p tcp --dport 8980 -j ACCEPT -A INPUT -p tcp --dport 18980 -j ACCEPT -A INPUT -p tcp --dport 8181 -j ACCEPT -A INPUT -p tcp --dport 9999 -j ACCEPT -A INPUT -p udp --dport 162 -j ACCEPT |----

On each *PostgreSQL* servers, open the port *TCP 5432*. To do this, edit the `/etc/sysconfig/iptables` file, and add the following rules before the first `REJECT` entry:

```
-A INPUT -p tcp --dport 5432 -j ACCEPT
```

Repeat the process on the standby *PostgreSQL* server (*pgdbsrv02*).

On the *NFS* server, open the port *TCP 2049*. To do this, edit the `/etc/sysconfig/iptables` file, and add the following rules before the first `REJECT` entry:

```
-A INPUT -p tcp --dport 2049 -j ACCEPT
```

Finally, restart *iptables* on all the servers:

```
[root@onmssrv01 ~]# service iptables restart
[root@onmssrv02 ~]# service iptables restart
[root@pgdbsrv01 ~]# service iptables restart
[root@pgdbsrv02 ~]# service iptables restart
[root@nfssrv01 ~]# service iptables restart
```

RHEL/CentOS 7

The first thing to do is enable and start firewalld on all servers:

```
[root@onmssrv01 ~]# systemctl enable firewalld
ln -s '/usr/lib/systemd/system/firewalld.service' '/etc/systemd/system/dbus-
org.fedoraproject.FirewallD1.service'
ln -s '/usr/lib/systemd/system/firewalld.service'
'/etc/systemd/system/basic.target.wants/firewalld.service'
[root@onmssrv01 ~]# systemctl start firewalld
[root@onmssrv01 ~]# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
   Active: active (running) since Thu 2015-07-16 18:55:59 UTC; 4s ago
 Main PID: 29635 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─29635 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

Jul 16 18:55:59 testsrv.local systemd[1]: Started firewalld - dynamic firewall daemon.
```

All firewall changes must be executed against *firewalld* using `firewall-cmd` and not through *iptables*.

Repeat the above commands on the rest of the servers: *onmssrv02*, *pgdbsrv01*, *pgdbsrv02* and *nfssrv01*.

On the *OpenNMS* servers, create a new file called `/etc/firewalld/services/opennms.xml`, and put the following content inside of it:

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

Then, reload firewall configuration:

```
[root@onmssrv01 ~]# firewall-cmd --reload
success
```

You can use the following command to verify that the new service called *opennms* is listed:

```
[root@onmssrv01 ~]# firewall-cmd --get-services
RH-Satellite-6 amanda-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns ftp high-availability http
https imaps ipp ipp-client ipsec kerberos kpasswd ldap ldaps libvirt libvirt-tls mdns mountd ms-wbt mysql
nfs ntp opennms openvpn pmcd pmproxy pmwebapi pmwebapis pop3s postgresql proxy-dhcp radius rpc-bind samba
samba-client smtp ssh telnet tftp tftp-client transmission-client vnc-server wbem-https
```

Enable the *opennms* service

```
[root@onmssrv01 ~]# firewall-cmd --permanent --add-service=opennms
success
[root@onmssrv01 ~]# firewall-cmd --add-service=opennms
success
```

RHEL 7 uses *Pacemaker* for managing the cluster. There is already a service group for high availability applications that includes *Pacemaker* ports. To enable it on *firewalld*:

```
[root@onmssrv01 ~]# firewall-cmd --permanent --add-service=high-availability
success
[root@onmssrv01 ~]# firewall-cmd --add-service=high-availability
success
```

Then, reload the firewall configuration

```
[root@onmssrv01 ~]# firewall-cmd --reload
success
```

Repeat all the process on the standby *OpenNMS* server (*onmssrv02*).

On the *PostgreSQL* servers, execute the following commands:

```
[root@pgdbsrv01 ~]# firewall-cmd --permanent --add-service=postgresql
success
[root@pgdbsrv01 ~]# firewall-cmd --add-service=postgresql
success
[root@pgdbsrv01 ~]# firewall-cmd --reload
success
```

Repeat the process on the standby *PostgreSQL* server (*pgdbsrv02*).

On the *NFS* server, execute the following commands:

```
[root@nfssrv01 ~]# firewall-cmd --permanent --add-service=nfs
success
[root@nfssrv01 ~]# firewall-cmd --add-service=nfs
success
[root@nfssrv01 ~]# firewall-cmd --reload
success
```

Syslogd in OpenNMS

In case you're interested on using *Syslog* to receive *syslog* messages and convert them into events, you should open the port you're going to use for this purpose. By default the port is 10514, but it could be different. This port is configured on `/opt/opennms/etc/syslogd-configuration.xml`.

If the port is changed, then the firewall rules for *OpenNMS* must be updated at `/etc/firewalld/services/opennms.xml`.

If the standard *syslog* server on the *OpenNMS* servers will be used as a gateway to receive *syslog* messages from all other servers and then forward them to *OpenNMS*, the port UDP 514 must be opened. You can either create a firewall *XML* file for *syslog*, or add the port to `opennms.xml`.

If the `opennms.xml` is updated (or another firewall rule *XML* file is created), you should reload the firewall configuration:

```
[root@onmssrv01 ~]# firewall-cmd --reload
success
```

External Firewalls

It is important to allow communication between *OpenNMS* and the nodes that are going to be monitored through the appropriate protocols. Typical applications to open are *ICMP*, *SNMP* (UDP 161) and *SNMP-Traps* (UDP 162). For other kind of monitor requirements, you might need to open other ports like *HTTP* (TCP 80).

In order to access *OpenNMS*, the *WebUI* port must be opened. By default it is 8980, but it can be changed. In case the port is changed, the internal firewall rules for *OpenNMS* must be updated.

Also, in some use cases, it is important to inject events directly to *OpenNMS*. This is done through the port TCP 5417.

Multicast and Networking

Corosync which is the common cluster service used on any version of *RHEL/CentOS* requires that multicast communication is open across all the cluster members. This is already covered by the firewall rules, but in case of a real environment, make sure that the IP rules on switches (or Virtual Networks, in case of *VMWare vCloud*) allows multicast; otherwise, the cluster won’t work.

Also, it is extremely important to reduce as much as possible the network devices between the cluster members, to reduce potential failure points.