Open Source Puppet Master & Agent Setup

Puppet master – station1.example.com
Puppet agent – station2.example.com

OS – RHEL 6.5
Selinux & Firewall – Disabled

Pre-requisites – The hostnames must be resolved by DNS server and The hosts must be in sync with NTP server.

Important Note : Please do not try these steps on a production server as this is only done for classroom lab purpose.

Install puppet master on station1:

- Setup the puppetlab yum repository.

```
wget https://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm

#cd /etc/yum.repos.d/
#vi puppetlabs.repo

( enable the disabled repository if any )

#yum repolist

#yum install puppet-server
```

Update /etc/puppet/puppet.conf for the puppet master setup.

We are making auto sign true for our lab setup so that the puppet master CA will auto sign the cert signing requests. The sample file below. I have only modified the [master] and [agent] section to reflect our hostnames.

```
[root@station1 puppet]#cd /etc/puppet
[root@station1 puppet]# cat puppet.conf
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
    ssldir = $vardir/ssl
```

```
[master]
   certname = station1.exmaple.com
   autosign = true
   dns_alt_names = station1.example.com

[agent]
   # The file in which puppetd stores a list of the classes
   # associated with the retrieved configuratiion.  Can be loaded in
   # the separate ``puppet`` executable using the ``--loadclasses``
   # option.
   # The default value is '$confdir/classes.txt'.
   classfile = $vardir/classes.txt
   server = station1.example.com

   # Where puppetd caches the local configuration.  An
   # extension indicating the cache format is added automatically.
   # The default value is '$confdir/localconfig'.
   localconfig = $vardir/localconfig
```

```
[root@station1 manifests]# puppet --version
3.8.2

[root@station1 manifests]# puppet master --verbose --no-daemonize
```

The above command will generate the CA server and certs. Once command is completed, you need to Ctrl+c to come back to the prompt. Now we will start the puppetmaster service.

```
[root@station1 ~]# service puppetmaster start
Starting puppetmaster:                                     [  OK  ]
```

Verify a agent run, even though we have no classes defined at this point of time.

```
[root@station1 ~]# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for station1.example.com
Info: Applying configuration version '1440930745'
```

Setup Agent host:

Setup the puppetlabs repository and install the package on station2.example.com.

```
#rpm -ivh https://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
```

```
#yum install puppet
```

Update /etc/puppet/puppet.conf to point to to the puppet master server (staion1.example.com). For a default configuration we have to only change server = station1.example.com.

Sample file below.

```
[root@station2 puppet]# cat puppet.conf
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
    ssldir = $vardir/ssl

[agent]
    # The file in which puppetd stores a list of the classes
    # associated with the retrieved configuratiion.  Can be
loaded in
    # the separate ``puppet`` executable using the ``--
loadclasses``
    # option.
    # The default value is '$confdir/classes.txt'.
    classfile = $vardir/classes.txt
    server = station1.example.com
    # Where puppetd caches the local configuration.  An
    # extension indicating the cache format is added
automatically.
    # The default value is '$confdir/localconfig'.
    localconfig = $vardir/localconfig
```

- Start the puppet agent

```
[root@station2 ~]# service puppet start
Starting puppet agent:                                 [  OK  ]
```

Verify a sample puppet agent run. As no classes defined yet, it should not apply anything.

```
[root@station2 ~]# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for station2.example.com
```

```
Info: Applying configuration version '1440930745'
Notice: Finished catalog run in 0.04 seconds
```

Create our first module and verify:

Let us create a sample module “hello_world” which will create a file /root/hello_world with some content.

The default location of the puppet module in puppet master is /etc/puppet/modules

```
#cd /etc/puppet/modules
#mkdir hello_world
#cd hello_world
#mkdir {files,manifests,templates,tests}
#cd manifests
#cat >init.pp
class hello_world {
             file { "/root/hello_world":
                 ensure => 'file',
         source =>
"puppet:///modules/hello_world/hello_world",
}

}
```

Save & exit

init.pp is the default name of the manifest file. We will learn about complex module configuration later.

Create the hello_world file that is to be pushed to agents as per our module requirement.

```
#cd /etc/puppet/modules/hello_world/
#cd files
#cat >hello_world
Hello World !!

This is my first file using puppet
```

Save & exit

You can instantiate the newly created class to verify if this class is working or not. The below step is optional and is used just to test the module manifests before we really push them to agent.

```
#cd /etc/puppet/modules/hello_world/tests/
```

```
#cat > init.pp
include hello_world
```

Save & exit. Now you can test the module locally on puppet master by doing below. The –noop option does not apply anything but simulate which would have been applied.

```
# puppet apply --noop init.pp
Notice: Compiled catalog for station1.example.com in
environment production in 0.18 seconds
Notice:
/Stage[main]/Hello_world/File[/root/hello_world]/ensure:
current_value absent, should be file (noop)
Notice: Class[Hello_world]: Would have triggered 'refresh'
from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 1
events
Notice: Finished catalog run in 0.12 seconds
```

Now we are good with the module setup. However, in order to apply the same class on the puppet agent node, we must create the node definition. We will use the default node definition for now.

On puppet master, we have to edit site.pp file for node definition.

```
[root@station1 ~]# cd /etc/puppet/manifests/

[root@station1 ~]# cat > site.pp
node default {
       class { 'hello_world': }
}
```

Save & exit.

Run the puppet agent on station2 to verify if it is working.

```
[root@station2 ~]# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for station2.example.com
Info: Applying configuration version '1440932461'
```

puppet agent is executed successfully and you can see the file hello_world got created in /root and it has the sample content.

All the Best !!