**Puppet Installation and Configuration:**

Puppet, from [Puppet Labs](https://puppet.com/), is a configuration management tool that helps system administrators automate the **provisioning, configuration, and management** of a server infrastructure. Planning ahead and using config management tools like **Puppet** can cut down on time spent repeating basic tasks, and help ensure that your configurations are consistent and accurate across your infrastructure. Once you get the hang of managing your servers with **Puppet** and other **automation** tools, you will have more free time that can be used to improve other aspects of your setup.

|  |
| --- |
| The official Puppet documentation can be found on the Puppet web site:  <https://docs.puppet.com/puppet/4.5/>  There are a number of tutorials on Internet including:  <https://www.unixmen.com/install-puppet-server-centos-7/>  <https://www.digitalocean.com/community/tutorials/how-to-install-puppet-4-in-a-master-agent-setup-on-centos-7> |

|  |
| --- |
| During this installation, you will go back and forth between the Puppet Client and the Puppet Server. I recommend opening two two terminals and keeping an ssh connection open to the Puppet Client. |

Network time:

It is very important that all the machines are synchronized in time. When the operating system was installed, the package cronyd was installed. You can check the status of chronyd using:

|  |
| --- |
| systemctl status chronyd |

You can also see that your host’s time is the same as the main server’s time using:

|  |
| --- |
| ssh htc180 date; date |

You should see output similar to:

|  |
| --- |
| [root@htc184 install]# ssh htc180 date; date  root@htc180's password:  Mon Mar 19 11:42:19 IST 2018  Mon Mar 19 11:42:19 IST 2018 |

Beginning to install Puppet:

**Puppet** is a client server system. We will start by installing the server.

**For this section all commands are issued on the** **Cobbler/Puppet Server**.

Usually the first step when installing Puppet is to install the repo for Puppet. We did this during the initial installation when we used the **InstallRepos** command. We did this so that we could access the Puppet files from the local server. If we haven’t done this, we would use the following:

|  |
| --- |
| **Do not issue this command:**  rpm -ivh https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm |

With the repo installed, we can now Install the puppet server software. Notice that we are installing a few additional packages:

* python-pip - This contains the pip command. **pip** is a recursive acronym that can stand for either "Pip Installs Packages" or "Pip Installs Python". We use it to install python libraries.
* libyaml - YAML is is a human-readable data serialization language. For Puppet, it is used for part of the configuration language.
* pyyaml - This is the python binding to YAML. This enables us to use the YAML library inside of python code.

|  |
| --- |
| **NOTE**: In the early part of using **Puppet**, you will need to know very little **YAML** This falls into the category of topics that you learn it when you need and generally only as much as you need. You can work with **Puppet** without knowing any **python**. Of course, **python** is a very **popular** language with a large support community. If you are planning to use **Linux**, **you should plan to learn python**. |

Install the **Puppet Server** software packages:

|  |
| --- |
| yum -y install puppetserver python-pip libyaml  pip install pyyaml |

Configuring the Puppet Server:

The Puppet Server configuration files are installed in:

|  |
| --- |
| /etc/puppetlabs/puppet |

The two main configuration files are:

|  |
| --- |
| * /etc/puppetlabs/puppet/puppet.conf - We will only add two additional lines * /etc/puppetlabs/puppet/hiera.yaml - The default file should work |

and we are going to add two additional files:

* node.sh - A simple **bash** shell script that prints to **STDOUT** the base configuration for the nodes in the system. It takes no arguments and will only work for a simple configuration.
* node.py - A python script that takes a single argument, the **hostname** of a **Puppet** **Client**, and prints to **STDOUT** the base configuration for that particular host. It is expandable to any number of different Puppet Client configurations.

|  |
| --- |
| * /etc/puppetlabs/puppet/node.py * /etc/puppetlabs/puppet/node.sh |

|  |
| --- |
| * **node.sh** will be **sufficient** for our two node cluster. It could be modified to take an input argument and return different configurations based on the argument. * For larger systems, **node.py** will work but eventually it will also not scale well. * For even larger systems, a management system such as [**Foreman**](https://www.theforeman.org/) would be used. [**Foreman**](https://www.theforeman.org/) maintains a database of hosts, all the packages that are installed and how all the files are configured for each host.   NOTE: I know of fairly large systems (thousands of computers) that are maintained with a python script similar to node.py. |

Both **node.py** and **node.sh** are available on the class server. You can get them in the usual way:

|  |
| --- |
| curl -o /etc/puppetlabs/puppet/node.py \  [http://htc180.najah.edu/software/AnNajah-Files/node.py](http://htc180.najah.edu/AnNajah-Files/node.py)  curl -o /etc/puppetlabs/puppet/node.sh \  [http://htc180.najah.edu/software/AnNajah-Files/node.sh](http://htc180.najah.edu/AnNajah-Files/node.py)  chmod +x /etc/puppetlabs/puppet/node.py  chmod +x /etc/puppetlabs/puppet/node.sh |

The contents of node.sh is:

|  |
| --- |
| #!/bin/bash  echo "---"  echo "classes:"  echo " role::htcxxx: {}"  echo ""  exit |

and running it produces the output:

|  |
| --- |
| ---  classes:  role::htcxxx: {} |

|  |
| --- |
| **Explanation**: When the **Puppet Clien**t runs, it sends a request to the **Puppet Server** for configuration information based on its hostname. The client’s hostname is passed to node.sh and the output is **YAML** configuration data. These data are them processed by the **Puppet Master** in order to send configuration commands back to the client.  **YAML** data always begins with three dashes:  ---  This then followed by **YAML** formatted data. The information above defines **Puppet classes** that make up the configuration information for the host. In the above output, there is a single class with no arguments:  role::htcxxx: {}  The output could be many classes, but it is best practices to return a single class and then have the **Puppet Manifests** (or configuration) files define the data, dependencies and actions that should be taken.  An important feature of **Puppet** is that the **Manifests** and data are stored on the **Puppet Server**; there is no need for a common file system. This means that **Puppet Servers** can manage **Puppet Clients** across a **WAN**.  On the experiment I work on, we have a **Puppet Server** that manages the configurations for **Puppet Clients** that are on different ends of the country and one **Puppet Client in Puerto** Rico is managed by a **Puppet Server in California**. These two systems are separated by about **10,000 Km**. |

The script **node.sh** will be used in the Puppet configuration file:

|  |
| --- |
| /etc/puppetlabs/puppet/puppet.conf |

When we installed the **Puppet Server software**, a default **puppet.conf** was installed. We will make two changes to it so that it uses **node.sh.** At bottom of the file **/etc/puppetlabs/puppet/puppet.conf** add the following two lines:

|  |
| --- |
| external\_nodes = /etc/puppetlabs/puppet/node.sh  node\_terminus = exec |

The complete /etc/puppetlabs/puppet/puppet.conf should contain:

|  |
| --- |
| # This file can be used to override the default puppet settings.  # See the following links for more details on what settings are available:  # - https://docs.puppetlabs.com/puppet/latest/reference/config\_important\_settings.html  # - https://docs.puppetlabs.com/puppet/latest/reference/config\_about\_settings.html  # - https://docs.puppetlabs.com/puppet/latest/reference/config\_file\_main.html  # - https://docs.puppetlabs.com/puppet/latest/reference/configuration.html  [master]  vardir = /opt/puppetlabs/server/data/puppetserver  logdir = /var/log/puppetlabs/puppetserver  rundir = /var/run/puppetlabs/puppetserver  pidfile = /var/run/puppetlabs/puppetserver/puppetserver.pid  codedir = /etc/puppetlabs/code  external\_nodes = /etc/puppetlabs/puppet/node.py  node\_terminus = exec |

Configuring the Puppet Server Firewall:

The **Puppet Clients** need to be able to contact the **Puppet Server** on **port 8140.** This requires opening that port:

|  |
| --- |
| firewall-cmd --zone=public --add-port=8140/tcp --permanent firewall-cmd --reload |

Starting Puppet Server:

We are now ready to start the **Puppet Server software**:

|  |
| --- |
| systemctl start puppetserver  systemctl enable puppetserver |

Configuring the Puppet Client:

|  |
| --- |
| For this part of the installation, you are working on the **Puppet Client.** |

The **Puppet Client** is fairly easy to configure:

* Install Puppet Client Software - puppet-agent
* Create Puppet Client configuration file - /etc/puppetlabs/puppet/puppet.conf
* Start the Puppet Client Software
* Initialize Puppet Agent - This sends a request to the Puppet Server for a certificate.

Install Puppet Client Software:

|  |
| --- |
| yum -y install puppet-agent |

Create Puppet Client Configuration:

The configuration file is **/etc/puppetlabs/puppet/puppet.conf** and only needs to contain:

|  |
| --- |
| [main]  server = htcYYY.najah.edu  [agent] |

|  |
| --- |
| **YYY** is the I**P address** for the **Puppet Server** |

Start the Puppet Client Software:

|  |
| --- |
| systemctl enable puppet  systemctl start puppet |

Initialize Puppet Agent:

You can initialize the **Puppet Client Agent** using the **“puppet agent”** command. The first time you run it, it will exit because it does not have a **certificate** issued by the **Puppet Server** yet. The **Puppet Client Agent** runs periodically to check for updates. If you wait, it will do this initialization step automatically. Regardless, you must go **back to the Puppet Server and approve the request.**

Running the Puppet Client Agent:

|  |
| --- |
| puppet agent -t |

|  |
| --- |
| NOTE: If the above fails because the puppet command is not found, try running the command:  source /etc/profile.d/puppet-agent.sh  Then rerun the command above. |

If the certificate has not been approved yet, you will see output similar to:

|  |
| --- |
| Info: Creating a new SSL key for htc192.najah.edu  Info: csr\_attributes file loading from /etc/puppetlabs/puppet/csr\_attributes.yaml  Info: Creating a new SSL certificate request for htc192.najah.edu  Info: Certificate Request fingerprint (SHA256): F5:03:90:E2:C2:3B:5E:02:F3:06:14:E3:B7:51:21:08:07:69:29:F1:9C:39:99:34:B1:76:2F:DF:77:F0:39:7A  Exiting; no certificate found and waitforcert is disabled |

Approving the Puppet Client’s certificate request:

Now you must go back to the Puppet Server and approve the request:

|  |
| --- |
| puppet cert list |

You will see output similar to:

|  |
| --- |
| “htc192.najah.edu" (SHA256) F5:03:90:E2:C2:3B:5E:02:F3:06:14:E3:B7:51:21:08:07:69:29:F1:9C:39:99:34:B1:76:2F:DF:77:F0:39:7A |

Approve the certificate using:

|  |
| --- |
| puppet cert sign htc192.najah.edu --all |

Your Puppet Server and Client are now installed.

Your first Puppet Manifests (configurations):

Puppet **configurations** are called **manifests**. For your first two **manifests**, you will just take the them from the class server. In the next class, we will start to look at how to write your own manifests.

**Puppet Client:**

Before installing the manifests, on the Puppet Client run:

|  |
| --- |
| puppet agent -t |

You will notice that it encounters an error similar to:

|  |
| --- |
| Info: Using configured environment 'production'  Info: Retrieving pluginfacts  Info: Retrieving plugin  Error: Could not retrieve catalog from remote server: Error 500 on SERVER: Server Error: Could not find class role::htcxxx for cuhepdj.colorado.edu on node cuhepdj.colorado.edu  Warning: Not using cache on failed catalog  Error: Could not retrieve catalog; skipping run |

This error occurs because we have not installed any **manifests**. In particular, recall from the explanation above about **/etc/puppetlabs/puppet/node.sh** that the **Puppet Server** wants to use the **Manifest** class **role::htc192{}** to configure the **client**. Since we have not installed this or any manifests, we get this error.

|  |
| --- |
| It should be noted that in Puppet errors are notoriously obscure and difficult to read.  I have no explanation why the authors do not improve this. |

**Puppet Server:**

Now working back on the **Puppet Server**, copy and install the first **manifests** from the **class server**:

|  |
| --- |
| curl -o /tmp/htc-first-manifests.tar.gz \  <http://htc180.najah.edu/AnNajah-Files/htc-first-manifests.tar.gz>  tar -C /etc/puppetlabs/code/environments/production \  -xzvf /tmp/htc-first-manifests.tar.gz |

This installed the modules **motd** and **roles**.

* roles - Act the main guide for htcXXX
* motd - Manages the file /etc/motd.

**Puppet Client:**

Now move back to the **Puppet Client**, and issue the following commands:

|  |
| --- |
| cat /etc/motd  puppet agent -t  cat /etc/motd |

If you ran puppet agent before the periodic running by the system, the first cat command did not produce any output and the second one did.

**Puppet Server:**

Back on the Puppet Server, we will change the module motd which will change the contents of /etc/motd.

|  |
| --- |
| cd /etc/puppetlabs/code/environments/production/modules/motd/manifests |

And edit **init.pp** changing:

|  |
| --- |
| class motd { -> class motd1 {  …  class motd2 { -> class motd { |

**Puppet Client:**

On the **Puppet Client** run:

|  |
| --- |
| cat /etc/motd  puppet agent -t  cat /etc/motd |

This should work without an error and the contents of /etc/motd should have changed.

**What is /etc/motd?**

Try logging out of the **Puppet Client** and then logging back in.

motd - Stands for Message of the Day.

|  |
| --- |
| **An important observation:**  If you configure the Puppet Client using the kickstart file in Cobbler, you never need to log into the Cobbler Client. All the configuration would be done on the Cobbler Server auto-magically. |