**Prerequisites**

To follow this tutorial, you will need **three Ubuntu 16.04 servers**, each with a non-root user with sudo privileges.

**One Puppet master**

One server will be the **Puppet master**. The Puppet master will run Puppet Server, which is resource intensive and requires:

* at least 1GB of memory
* at least 1 CPU core

To manage larger infrastructures, the Puppet master will require more resources.

**Two Puppet agents**

The other two servers will be **Puppet agent nodes**, managed by the Puppet master. We'll call them db1 and web1.

When these three servers are in place, you're ready to begin.

**Step 1 — Configuring /etc/hosts**

Puppet master servers and the nodes they manage need to be able to communicate with each other. In most situations, this will be accomplished using DNS, either configured on an externally hosted service or on self-hosted DNS servers maintained as part of the infrastructure.

DNS is its own domain of expertise, however, even on hosted services, so in order to focus on the fundamentals of Puppet itself and eliminate potential complexity in troubleshooting while we're learning, in this tutorial we'll use the /etc/hosts file instead.

**On every machine**

On each machine, edit the /etc/hosts file. At the end of the file, specify the Puppet master server as follows, substituting the IP address for *your* Puppet master:

$sudo nano /etc/hosts

/etc/hosts

. . .

puppet\_ip\_address puppet

. . .

When you're done, save and exit.

**Note:** By default, Puppet agents will look for the Puppet master at puppet to make it easier to get Puppet set up. This means we *must* use the name puppet in /etc/hosts. If puppet does not resolve to the Puppet master, the agents will not be able to make contact without [configuring the server value in the agent's puppet.conf](https://docs.puppet.com/puppet/latest/config_file_main.html#example-agent-config).

**Step 2 — Installing Puppet Server**

Puppet Server is the software that pushes configuration from the Puppet master to the other servers. It runs only on the Puppet master; the other hosts will run the Puppet Agent.

**Note:** The Ubuntu package manager *does* contain packages for Puppet, but many administrators need to manage multiple operating systems and versions. In this case, working with the official Puppet Labs repositories can simplify administration by allowing you to maintain the same Puppet version on all systems.

We'll enable the official Puppet Labs collection repository with these commands:

$curl -O https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb

$sudo dpkg -i puppetlabs-release-pc1-xenial.deb

$sudo apt-get update

When apt-get update is complete, ensuring that we'll be pulling from the Puppet Labs repository, we'll install the puppetserver package:

$sudo apt-get install puppetserver

Press Y to proceed. Once installation is complete, and before we start the server, we'll take a moment to configure the memory.

**Configure memory allocation**

By default, Puppet Server is configured to use 2 GB of RAM. You can customize this setting based on how much free memory the master server has and how many agent nodes it will manage.

To customize it, open /etc/default/puppetserver:

$sudo nano /etc/default/puppetserver

Then find the JAVA\_ARGS line, and use the -Xms and -Xmx parameters to set the memory allocation. We'll increase ours to 3 gigabytes:

/etc/default/puppetserver

JAVA\_ARGS="-Xms3g -Xmx3g -XX:MaxPermSize=256m"

Save and exit when you're done.

**Open the firewall**

When we start Puppet Server, it will use port 8140 to communicate, so we'll ensure it's open:

$sudo ufw allow 8140

Next, we'll start Puppet server.

**Start Puppet server**

We'll use systemctl to start Puppet server:

$sudo systemctl start puppetserver

This will take some time to complete.

Once we're returned to the command prompt, we'll verify we've succeeded since systemctl doesn't display the outcome of all service management commands:

$sudo systemctl status puppetserver

We should see a line that says "active (running)" and the last line should look something like:

Output

Dec 07 16:27:33 puppet systemd[1]: Started puppetserver Service.

Now that we've ensured the server is running, we'll configure it to start at boot:

$sudo systemctl enable puppetserver

With the server running, now we're ready to set up Puppet Agent on our two agent machines, db1 and web1.

**Step 3 — Installing the Puppet Agent**

The Puppet agent software must be installed on any server that the Puppet master will manage. In most cases, this will include every server in your infrastructure.

**Note:** The Puppet agent can run on all major Linux distributions, some UNIX platforms, and Windows. Installation instructions vary on each OS. Directions to install the Puppet agent on CentOS are available [here](https://www.digitalocean.com/community/tutorials/how-to-install-puppet-4-in-a-master-agent-setup-on-centos-7#install-puppet-agent), and you can find directions for the complete set of installation targets in the [Puppet Reference Manual](https://docs.puppet.com/puppet/4.8/install_linux.html).

**Enable the official Puppet Labs repository**

First we'll enable the official Puppet Labs collection repository with these commands:

* wget https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb
* sudo dpkg -i puppetlabs-release-pc1-xenial.deb
* sudo apt-get update

**Install the Puppet agent package**

Then, we'll install the puppet-agent package:

$sudo apt-get install puppet-agent

We'll start the agent and enable it to start on boot:

$sudo systemctl start puppet

$sudo systemctl enable puppet

Finally, we'll repeat these steps on web1:

$wget https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb

$sudo dpkg -i puppetlabs-release-pc1-xenial.deb

$sudo apt-get update

$sudo apt-get install puppet-agent

$sudo systemctl enable puppet

$sudo systemctl start puppet

Now that both agent nodes are running the Puppet agent software, we will sign the certificates on the Puppet master.

**Step 4 — Signing Certificates on Puppet Master**

The first time Puppet runs on an agent node, it sends a certificate signing request to the Puppet master. Before Puppet Server will be able to communicate with and control the agent node, it must sign that particular agent node's certificate.

**List current certificate requests**

To list all unsigned certificate requests, run the following command on the Puppet master:

$sudo /opt/puppetlabs/bin/puppet cert list

There should be one request for each host you set up, that looks something like the following:

Output:

"db1.localdomain" (SHA256) 46:19:79:3F:70:19:0A:FB:DA:3D:C8:74:47:EF:C8:B0:05:8A:06:50:2B:40:B3:B9:26:35:F6:96:17:85:5E:7C

"web1.localdomain" (SHA256) 9D:49:DE:46:1C:0F:40:19:9B:55:FC:97:69:E9:2B:C4:93:D8:A6:3C:B8:AB:CB:DD:E6:F5:A0:9C:37:C8:66:A0

A + in front of a certificate indicates it has been signed. The absence of a plus sign indicates our new certificate has not been signed yet.

**Sign requests**

To sign a single certificate request, use the puppet cert sign command, with the hostname of the certificate as it is displayed in the certificate request.

For example, to sign db1's certificate, you would use the following command:

sudo /opt/puppetlabs/bin/puppet cert sign db1.localdomain

Output similar to the example below indicates that the certificate request has been signed:

Output:

Notice: Signed certificate request for db.localdomain

Notice: Removing file Puppet::SSL::CertificateRequest db1.localdomain at '/etc/puppetlabs/puppet/ssl/ca/requests/db1.localdomain.pem'

The Puppet master can now communicate and control the node that the signed certificate belongs to. You can also sign all current requests at once.

We'll use the --all option to sign the remaining certificate:

sudo /opt/puppetlabs/bin/puppet cert sign --all

Now that all of the certificates are signed, Puppet can manage the infrastructure. You can learn more about managing certificates in the [How to Manage Puppet 4 Certificates](https://www.digitalocean.com/community/tutorials/how-to-manage-puppet-4-certificates) cheat sheet.

**Step 5 — Verifying the Installation**

Puppet uses a domain-specific language to describe system configurations, and these descriptions are saved to files called "manifests", which have a .pp file extension. You can learn more about these in the [Getting Started with Puppet Code: Manifests and Modules](https://www.digitalocean.com/community/tutorials/getting-started-with-puppet-code-manifests-and-modules) guide, but for now we'll create a brief directive to verify that the Puppet Server can manage the Agents as expected.

We'll begin by creating the default manifest, site.pp, in the default location:

sudo nano /etc/puppetlabs/code/environments/production/manifests/site.pp

We'll use Puppet's domain-specific language to create a file called it\_works.txt on agent nodes located in the tmp directory which contains the public IP address of the agent server and sets the permissions to-rw-r--r--:

site.pp example

file {'/tmp/it\_works.txt': # resource type file and filename

ensure => present, # make sure it exists

mode => '0644', # file permissions

content => "It works on ${ipaddress\_eth0}!\n", # Print the eth0 IP fact

}

By default Puppet Server runs the commands in its manifests by default every 30 minutes. If the file is removed, the ensure directive will cause it to be recreated. The mode directive will set the file permissions, and the content directive add content to the directive.

We can also test the manifest on a single node using puppet agent --test. Note that --test is not a flag for a dry run; if it's successful, it will change the agent's configuration.

Rather than waiting for the Puppet master to apply the changes, we'll apply the manifest now on db1:

$sudo /opt/puppetlabs/bin/puppet agent --test

The output should look something like:

Output

Info: Using configured environment 'production'

Info: Retrieving pluginfacts

Info: Retrieving plugin

Info: Loading facts

Info: Caching catalog for db1.localdomain

Info: Applying configuration version '1481131595'

Notice: /Stage[main]/Main/File[/tmp/it\_works.txt]/ensure: defined content as '{md5}acfb1c7d032ed53c7638e9ed5e8173b0'

Notice: Applied catalog in 0.03 seconds

When it's done, we'll check the file contents:

$cat /tmp/it\_works.txt

Output

It works on 203.0.113.0!

Repeat this for web1 or, if you prefer, check back in half an hour or so to verify that the Puppet master is running automatically.

**Note:** You can check the log file on the Puppet master to see when Puppet last [compiled the catalog](https://docs.puppet.com/puppet/latest/subsystem_catalog_compilation.html) for an agent, which indicates that any changes required should have been applied.

$tail /var/log/puppetlabs/puppetserver/puppetserver.log

Output excerpt

. . .

2016-12-07 17:35:00,913 INFO [qtp273795958-70] [puppetserver] Puppet Caching node for web1.localdomain

2016-12-07 17:35:02,804 INFO [qtp273795958-68] [puppetserver] Puppet Caching node for web1.localdomain

2016-12-07 17:35:02,965 INFO [qtp273795958-68] [puppetserver] Puppet Compiled catalog for web1.localdomain in environment production in 0.13 seconds

. . .

Congratulations! You've successfully installed Puppet in Master/Agent mode.

**Conclusion**

Now that you have a basic agent/master Puppet installation, you are ready to learn more about how to use Puppet to manage your server infrastructure.