How To Install Puppet To Manage Your Server Infrastructure

Create Puppet Master Server Using Ubuntu 14.04 x64 VPS

Install NTP

Because it acts as a certificate authority for agent nodes, the puppet master server must maintain accurate system time to avoid potential problems when it issues agent certificates--certificates can appear to be expired if there are time discrepancies. We will use Network Time Protocol (NTP) for this purpose.

First, do a one-time time synchronization using the `ntpdate` command:

```
sudo ntpdate pool.ntp.org
```

Your system time will be updated, but you need to install the NTP daemon to automatically update the time to minimize time drift. Install it with the following apt command:

```
sudo apt-get update && sudo apt-get -y install ntp
```

It is common practice to update the NTP configuration to use "pools zones" that are geographically closer to your NTP server.

```
sudo nano /etc/ntp.conf
```

Add the time servers from the NTP Pool Project page to the top of the file:

```
server 0.us.pool.ntp.org
server 1.us.pool.ntp.org
server 2.us.pool.ntp.org
server 3.us.pool.ntp.org
```

Save and exit. Restart NTP to add the new time servers.

```
sudo service ntp restart
```

Now that our server is keeping accurate time, let's install the Puppet master software.

Install Puppet Master

There are a variety of ways to install open source Puppet. We will use the debian package called*puppetmaster-passenger*, which is provided by Puppet Labs. The *puppetmaster-passenger* package includes the Puppet master plus production-ready web server (Passenger with Apache), which eliminates a few configuration steps compared to using the basic *puppetmaster* package.

Download the Puppet Labs package:

```
cd ~; wget https://apt.puppetlabs.com/puppetlabs-release-trusty.deb
```

Install the package:

```
sudo dpkg -i puppetlabs-release-trusty.deb
```

Update apt's list of available packages:

```
sudo apt-get update
```

Install the `puppetmaster-passenger` package:

```
sudo apt-get install puppetmaster-passenger
```

The Puppet master, Passenger, Apache, and other required packages are now installed. Because we are using Passenger with Apache, the Puppet master process is controlled by Apache, i.e. it runs when Apache is running.

Before continuing, stop the Puppet master by stopping the `apache2` service:

```
sudo service apache2 stop
```

Next, we want to lock the version of Puppet.

Lock the Version

Changes from version to version can occasionally cause your Puppet environment to stop working properly. For this reason, you will want to maintain a consistent Puppet version across your entire infrastructure. If you decide to upgrade to a newer version, make sure that you upgrade your *master*before any agent nodes, as the master cannot manage agents that have a higher version number.

Let's look up the version of our Puppet installation with the following command:

```
puppet help | tail -n 1
```

During the time of writing, the output from the previous command is `Puppet v3.6.2`. We can use apt's pin feature to lock our Puppet install to `3.6.*`, which will prevent apt from upgrading Puppet to a higher major release. Create a new file called in the apt preferences directory:

```
sudo vi /etc/apt/preferences.d/00-puppet.pref
```

Add the following lines to lock the `puppet`, `puppet-common`, and `puppetmaster-passenger`packages to the `3.6.*` (change this to match your installed version):

```
# /etc/apt/preferences.d/00-puppet.pref
Package: puppet puppet-common puppetmaster-passenger
Pin: version 3.6*
Pin-Priority: 501
```

Save and exit. Your Puppet version is now locked.

The next step is to set up your Puppet master names and certificates.

Set Up Names and Certificates

Puppet uses SSL certificates to authenticate communication between master and agent nodes. The Puppet master acts as a certificate authority (CA), and must generate its own certificates which is used to sign agent certificate requests. We will setup the master's certificates now.

Delete Existing Certificates

Delete any existing SSL certificates that were created during the package install. The default location of Puppet's SSL certificates is /var/lib/puppet/ssl:

```
sudo rm -rf /var/lib/puppet/ssl
```

Configure Certificate

When creating the puppet master's certificate, include every DNS name at which agent nodes can contact the master at. In the case of our example, we will use "puppet" and "puppet.nyc2.example.com", the short hostname and FQDN, respectively.

Edit the master's puppet.conf file:

```
sudo vi /etc/puppet/puppet.conf
```

It will look something like the following:

```
[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
templatedir=$confdir/templates

[master]
# These are needed when the puppetmaster is run by passenger
# and can safely be removed if webrick is used.
ssl_client_header = SSL_CLIENT_S_DN
ssl_client_verify_header = SSL_CLIENT_VERIFY
```

Delete the line with the templatedir option, as that option is deprecated.

Add the following two lines to the end of the [main] section (replace the highlighted text with the private FQDN):

```
certname = puppet
dns_alt_names = puppet,puppet.nyc2.example.com
```

It is important to use assign certname to "puppet" because the Apache/Passenger configuration expects the certificate to be named "puppet". If you decide that you want a different certname setting, be sure to edit the Apache config file (/etc/apache2/sites-available/puppetmaster.conf) to change the name of the SSL certificate path.

Save and exit.

Generate New Certificate

Now create new CA certificates by running the following command:

```
sudo puppet master --verbose --no-daemonize
```

You will see several lines of output saying that SSL keys and certificates are being created. Once you seeNotice: Starting Puppet master version 3.6.2, the certificate setup is complete. Press CTRL-Cto return to the shell.

Sample Output:

```
Info: Creating a new SSL key for ca
Info: Creating a new SSL certificate request for ca
Info: Certificate Request fingerprint (SHA256):
EC:7D:ED:15:DE:E3:F1:49:1A:1B:9C:D8:04:F5:46:EF:B4:33:91:91:B6:5D:19:AC:21:
D6:40:46:4A:50:5A:29
```

```
Notice: Signed certificate request for ca
...
Notice: Signed certificate request for puppet
Notice: Removing file Puppet::SSL::CertificateRequest puppet at
'/var/lib/puppet/ssl/ca/requests/puppet.pem'
Notice: Removing file Puppet::SSL::CertificateRequest puppet at
'/var/lib/puppet/ssl/certificate_requests/puppet.pem'
Notice: Starting Puppet master version 3.6.2
```

If you want to look at the cert info of the certificate that was just created, type in the following:

```
sudo puppet cert list -all
```

The previous command actually lists all of the signed certificates and unsigned certificate requests. Currently, only the master cert will display, because no other certificates have been added yet:

```
+ "puppet" (SHA256)
05:22:F7:65:64:CF:46:0E:09:2C:5D:FD:8C:AC:9B:31:17:2B:7B:05:93:D5:D1:01:52:
72:E6:DF:84:A0:07:37 (alt names: "DNS:puppet", "DNS:puppet.nyc2.example.com")
```

Our Puppet master service is almost ready to be started. Let's take a look at the master configuration first.

Configure Puppet Master

The main puppet configuration file, `puppet.conf`, consists of three sections: `[main]`, `[master]`, and`[agent]`. As you may have guessed, the "main" section contains global configuration, the "master" section is specific to the puppet master, and the "agent" is used to configure the puppet agent. Aside from the changes we made earlier, the defaults will work fine for a basic setup.

The configuration file has many options which might be relevant to your own setup. A full description of the file is available at Puppet Labs: Main Config File (puppet.conf).

If you want to edit it, run this command:

```
sudo vi /etc/puppet/puppet.conf
```

Let's take a look at the main manifest file.

Main Manifest File

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. The default main manifest file is located at `/etc/puppet/manifests/site.pp`. We will cover some basic manifests later, but we will create a placeholder file for now:

```
sudo touch /etc/puppet/manifests/site.pp
```

Start Puppet Master

We are now ready to start the Puppet master. Start it by running the `apache2` service:

```
sudo service apache2 start
```

Your Puppet master is running, but it isn't managing any agent nodes yet. Let's learn how to install and add Puppet agents!

Install Puppet Agent

The Puppet agent must be installed on any server that the Puppet master will manage. In most cases, this includes every server in your infrastructure. As mentioned in the introduction, the Puppet agent can run on all major Linux distributions, some UNIX platforms, and Windows. Because the installation varies on each OS slightly, we will only cover the installation on Ubuntu and Debian servers.

Instructions to install Puppet on other platforms are located in the Puppet Labs Docs--be sure to follow the "Install Puppet on Agent Nodes" section for your OS of choice.

Note: It is assumed that all of your Puppet nodes, including agent nodes, are configured to use your DNS. If you are creating a brand new server, make sure to add it to your DNS before installing the Puppet agent.

Ubuntu / Debian Agent Node

Note: All of our example agent nodes, *host1*, *host2*, *ns1*, and *ns2*, are Ubuntu 14.04 VPS. We will repeat this step for each server, so each can be managed by the Puppet master.

On your Puppet agent node, download the Puppet Labs package:

```
cd ~; wget https://apt.puppetlabs.com/puppetlabs-release-trusty.deb
```

Install the package:

```
sudo dpkg -i puppetlabs-release-trusty.deb
```

Update apt's list of available packages:

```
sudo apt-get update
```

Install the Puppet agent package (puppet):

```
sudo apt-get install puppet
```

The puppet agent is disabled by default. To change this, update its default file:

```
sudo vi /etc/default/puppet
```

And change value of START to "yes":

```
START=yes
```

Save and exit /etc/default/puppet.

Lock the Version

As with the Puppet master, we will want to use the apt pin feature to lock the version of the Puppet agent:

```
sudo vi /etc/apt/preferences.d/00-puppet.pref
```

Add the following lines to lock the puppet and puppet-common packages to the 3.6.* (change this to match your installed version):

```
# /etc/apt/preferences.d/00-puppet.pref
Package: puppet puppet-common
Pin: version 3.6*
Pin-Priority: 501
```

Save and exit. Your Puppet version is now locked.

Configure Agent

Before running the agent, we must make a few configuration changes.

Edit the agent's puppet.conf:

```
sudo vi /etc/puppet/puppet.conf
```

It will look exactly like the Puppet master's initial configuration file.

Again, delete the templatedir line. Then delete the [master] section, and all of the lines below it.

Assuming that the Puppet master is reachable at "puppet", the agent should be able to connect to the master. If the master is not available at "puppet", you will need to add the Puppet master's FQDN. We recommend configuring this regardless (substitute the FQDN with your own):

```
[agent]
server = puppet.nyc2.example.com
```

Save and exit.

The Puppet agent is ready to run. Do so by running the following command:

```
sudo service puppet start
```

If everything is configured properly, you should not see any output. The first time you run the Puppet agent, it generates an SSL certificate and sends a signing request to the Puppet master. After the Puppet master signs the agent's certificate, it will be able to communicate with the agent node.

Note: If this is your first Puppet agent, it is recommended that you attempt to sign the certificate on the Puppet master before adding your other agents. Once you have verified that everything works properly, then you can go back and add the remaining agent nodes with confidence.

Sign Request On Master

The first time Puppet runs on an agent node, it will send a certificate signing request to the Puppet master. Before the master will be able to communicate and control the agent node, it must sign that particular agent node's certificate. We will describe how to sign and check for signing requests.

List Current Certificate Requests

On the Puppet master, run the following command to list all unsigned certificate requests:

```
sudo puppet cert list
```

If you just set up your first agent node, you will see one request. It will look something like the following, with the agent node's FQDN as the hostname:

```
"host1.nyc2.example.com" (SHA256)
B1:96:ED:1F:F7:1E:40:53:C1:D4:1B:3C:75:F4:7C:0B:A9:4C:1B:5D:95:2B:79:C0:08:
DD:2B:F4:4A:36:EE:E3
```

Note that there is no + in front of it. This indicates that it has not been signed yet.

Sign A Request

To sign a certificate request, use the `puppet cert sign` command, with the hostname of the certificate you want to sign. For example, to sign `host1.nyc2.example.com`, you would use the following command:

```
sudo puppet cert sign host1.nyc2.example.com
```

You will see the following output, which indicates that the certificate request has been signed:

```
Notice: Signed certificate request for host1.nyc2.example.com
Notice: Removing file Puppet::SSL::CertificateRequest host1.nyc2.example.com at
'/var/lib/puppet/ssl/ca/requests/host1.nyc2.example.com.pem'
```

The Puppet master can now communicate and control the node that the signed certificate belongs to.

If you want to sign all of the current requests, use the -all option, like so:

```
sudo puppet cert sign --all
```

Revoke Certificates

You may want to remove a host from Puppet, or rebuild a host then add it back to Puppet. In this case, you will want to revoke the host's certificate from the Puppet master. To do this, you will want to use the cleanaction:

```
sudo puppet cert clean hostname
```

The specified host's associated certificates will be removed from Puppet.

View All Signed Requests

If you want to view all of the requests, signed and unsigned, run the following command:

```
sudo puppet cert list --all
```

You will see a list of all of the requests. Signed requests are preceded by a + and unsigned requests do not have the +.

```
 "ns2.nyc2.example.com"   (SHA256)
E4:F5:26:EB:B1:99:1F:9D:6C:B5:4B:BF:86:14:40:23:E0:50:3F:C1:45:D0:B5:F0:68:6
E:B2:0F:41:C7:BA:76
+ "host1.nyc2.example.com" (SHA256)
71:A2:D3:82:15:0D:80:20:D4:7E:E3:42:C2:35:87:83:79:2B:57:1D:D5:5A:EC:F6:8B:
```

```
EE:51:69:53:EB:6B:A1
+ "host2.nyc2.example.com" (SHA256)
F4:79:A6:7C:27:0C:EA:8E:BC:31:66:FF:F2:01:AB:B1:35:7B:9F:5E:C8:C9:CE:82:EE:E
8:2F:23:9F:0C:2B:ED
+ "puppet"                (SHA256)
05:22:F7:65:64:CF:46:0E:09:2C:5D:FD:8C:AC:9B:31:17:2B:7B:05:93:D5:D1:01:52:
72:E6:DF:84:A0:07:37 (alt names: "DNS:puppet", "DNS:puppet.nyc2.example.com")
```

Congrats! Your infrastructure is now ready to be managed by Puppet!

Getting Started with Puppet

Now that your infrastructure is set up to be managed with Puppet, we will show you how to do a few basic tasks to get you started.

How Facts Are Gathered

Puppet gathers facts about each of its nodes with a tool called *facter*. Facter, by default, gathers information that is useful for system configuration (e.g. OS names, hostnames, IP addresses, SSH keys, and more). It is possible to add custom facts if you need other facts to perform you configurations.

The facts gathered can be useful in many situations. For example, you can create an web server configuration template and automatically fill in the appropriate IP addresses for a particular virtual host. Or you can determine that your server's OS is "Ubuntu", so you should run the `apache2` service instead of`httpd`. These are basic examples, but they should give you an idea of how facts can be used.

To see a list of facts that are automatically being gathered on your agent node, run the following command:

```
facter
```

How The Main Manifest Is Executed

The puppet agent periodically checks in with the puppet master (typically every 30 minutes). During this time, it will send facts about itself to the master, and pull a current catalog--a compiled list of resources and their desired states that are relevant to the agent, determined by the main manifest. The agent node will then attempt to make the appropriate changes to achieve its desired state. This cycle will continue as long as the Puppet master is running and communicating with the agent nodes.

Immediate Execution on a Particular Agent Node

It is also possible initiate the check for a particular agent node manually, by running the following command (on the agent node in question):

```
puppet agent --test
```

Running this will apply the main manifest to the agent running the test. You might see output like the following:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts in /var/lib/puppet/lib/facter/pe_version.rb
Info: Loading facts in /var/lib/puppet/lib/facter/puppet_vardir.rb
Info: Loading facts in /var/lib/puppet/lib/facter/root_home.rb
Info: Loading facts in /var/lib/puppet/lib/facter/facter_dot_d.rb
Info: Caching catalog for host1.nyc2.example.com
Info: Applying configuration version '1407966707'
```

This command is useful for seeing how the main manifest will affect a single server immediately.

One-off Manifests

The puppet apply command allows you to execute manifests that are not related to the main manifest, on demand. It only applies the manifest to the node that you run the apply from. Here is an example:

```
sudo puppet apply /etc/puppet/modules/test/init.pp
```

Running manifests in this fashion is useful if you want to test a new manifest on an agent node, or if you just want to run a manifest once (e.g. to initialize an agent node to a desired state).

A Simple Manifest

As you may recall, the main manifest file on the Puppet master is located at/etc/puppet/manifests/site.pp.

On the *master*, edit it now:

```
sudo vi /etc/puppet/manifests/site.pp
```

Now add the following lines to describe a file resource:

```
file {'/tmp/example-ip':                                            # resource type file and filename
  ensure  => present,                                               # make sure it exists
  mode    => 0644,                                                  # file permissions
  content => "Here is my Public IP Address: ${ipaddress_eth0}.\n",  # note the
ipaddress_eth0 fact
}
```

Now save and exit. The inline comments should explain the resource that we are defining. In plain English, this will make *ensure* that all agent nodes will have a file at /tmp/example-ip, with -rw-r--r--permissions, and text that contains the node's public IP address.

You can either wait until the agent checks in with the master automatically, or you can run the `puppet agent --test` command (from one of your agent nodes). Then run the following command to print the file:

```
cat /tmp/example-ip
```

You should see output that looks like the following (with that node's IP address):

```
Here is my Public IP Address: 128.131.192.11.
```

Specify a Node

If you want to define a resource for specific nodes, define a node in the manifest.

On the master, edit site.pp:

```
sudo vi /etc/puppet/manifests/site.pp
```

Now add the following lines:

```
node 'ns1', 'ns2' {    # applies to ns1 and ns2 nodes
  file {'/tmp/dns':    # resource type file and filename
```

```
    ensure => present, # make sure it exists
    mode => 0644,
    content => "Only DNS servers get this file.\n",
  }
}

node default {}       # applies to nodes that aren't explicitly defined
```

Save and exit.

Now Puppet will ensure that a file at `/tmp/dns` will exist on *ns1* and *ns2*. You may want to run the `puppet agent --test` command (from ns1 or ns2), if you do not want to wait for the scheduled Puppet agent pull.

Note that if you do not define a resource, Puppet will do its best not to touch it. So if you deleted these resources from the manifest, Puppet will not delete the files it created. If you want to have it delete the files, change `ensure` to `absent`.

These examples don't do anything useful, but they do prove that Puppet is working properly.