**Puppet**

**Edureka ClassRecording 6 from 40th min**

Configuration Management

Using configuration management we write infrastructure as a code(IAC)

Using configuration management we write code to manage our infrastructure. Instead of manually configuring we write code as IAC by using tools like Puppet, chef, Ansible, salt and Terraform

Puppet is a configuration management tool that system administrators use to automate the processes involved in maintaining a company's IT infrastructure. Writing individual Puppet manifest files is sufficient for automating simple tasks. However, when you have an entire workflow to automate, it is ideal to create and use a Puppet module instead. A Puppet module is just a collection of manifests along with files that those manifests require, neatly bundled into a reusable and shareable package.

**Puppet – Agent based**

Ansible – Agent less

**Chef – Agent based**

Salt – Agent less

**Agent based** means if you want to run any machine using puppet configuration we need to make sure that we have one master node or controller and every machine that try to conenect puppet have atleast one agent installed.

**Puppet follows Pull** apporach i.e. always the responsibility of the node agent to start communicate with the master node. In this The puppet node agentor will try to pull the configuration from the master.

**Ansible and Salt follows agent less and push approach i.e. the main machine(master) pushes the configuration to the agent**

In Puppet we use to write the code in DSL (Domain Specific Language).

All the puppet files end with .PP extension

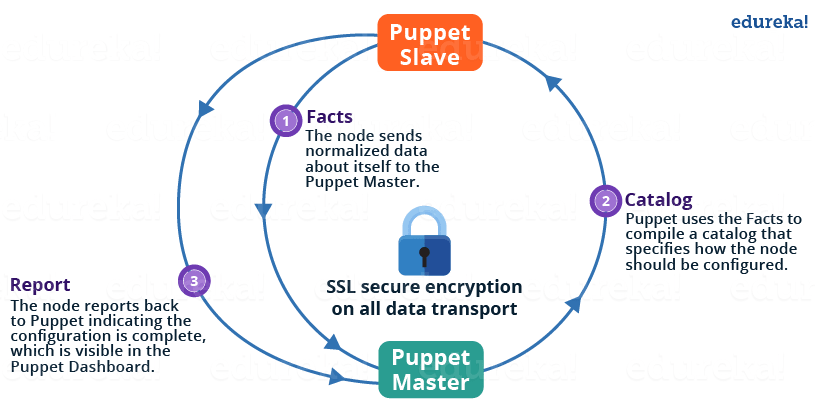
Resource abstraction: **It is a layer which allows us to run our program across multiple operating systems.** using one particular program we can work on various operating systems like linux, mac, unbutu etc.

Note: **Puppet master or controller (agent(slave)) can be deployed only on Linux/Centos/Ubuntu machine.**

**Puppet communicates over SSL protocol using port 8140(defalut port).**

**Puppet master will be installed on controller and we must have only one controller and n number of puppet agents (slaves).**

**Puppet Architecture:**



Puppet follows pull approach i.e. it is always responsibility of the slave or agent to start communication with the puppet master

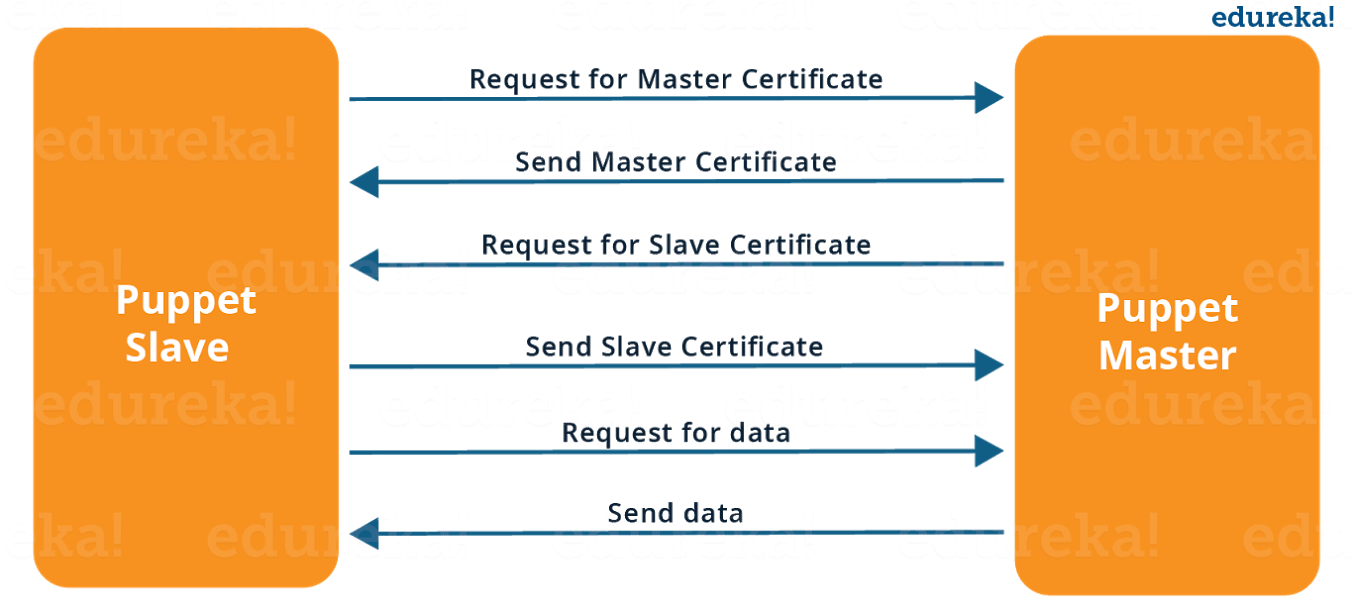
**Step 1:** The Puppet Agent sends the Facts to the Puppet Master. Facts are basically key/value data pair that represents some aspect of Slave state, such as its IP address, up-time, operating system, or whether it’s a virtual machine. These facts are sent in the form of node object.

**Step 2:** Once the puppet master receives the facts it will identify which particular node/agent is trying to communicate to. Once the master identifies the particular slave node and identifies the facts it will create a compiler catalogue. Catalogue is nothing but a document with details of the configuration that are to be done on the node. For example if want to push or deploy some software or if we want to start the service those details will be specifeed in the catalogue. These catalogue will be compiled by the puppet master.

**Step 3:**once the catalogue is compiled by the master, these catalogues will be placed on the slave node.Then slave node will deploy the configuration that it got from the catalogue. Puppet Slave reports back to Master indicating that Configuration is complete, which is visible in the Puppet dashboard.

**Puppet Master and Slave Communication:**

Puppet Master and Slave communicates through a secure encrypted channel with the help of SSL. The diagram below depicts the same:



As you can see from the above Image:

* Puppet Slave asks for Puppet Master certificate.
* After receiving Puppet Master certificate, Master requests for Slave certificate.
* Once Master has signed the Slave certificate, Slave requests for configuration/data.
* Finally, Puppet Master will send the configuration to Puppet Slave.

**Components of Puppet:**

**Manifests:** Every Slave has got its configuration details in Puppet Master, written in the native Puppet language. These details are written in the language which Puppet can understand and are termed as Manifests. They are composed of Puppet code and their filenames use the *.pp* extension. These are basically Puppet programs.   
For example: You can write a Manifest in Puppet Master that creates a file and installs Apache server on all Puppet Slaves connected to the Puppet Master.

**Module:** A Puppet Module is a collection of Manifests and data (such as facts, files, and templates), and they have a specific directory structure. Modules are useful for organizing your Puppet code, because they allow you to split your code into multiple Manifests. Modules are self-contained bundles of code and data.

**Resource:**Resources are the fundamental unit for modeling system configurations. Each Resource describes some aspect of a system, like a specific service or package.

**Facter:**Facter gathers basic information (facts) about Puppet Slave such as hardware details, network settings, OS type and version, IP addresses, MAC addresses, SSH keys, and more. These facts are then made available in Puppet Master’s Manifests as variables.

**Mcollective:** It is a framework that allows several jobs to be executed in parallel on multiple Slaves. It performs various functions like:

* Interact with clusters of Slaves, whether in small groups or very large deployments.
* Use a broadcast paradigm to distribute requests. All Slaves receive all requests at the same time, requests have filters attached, and only Slaves matching the filter will act on requests.
* Use simple command-line tools to call remote Slaves.
* Write custom reports about your infrastructure.

**Catalogs:**A Catalog describes the desired state of each managed resource on a Slave. It is a compilation of all the resources that the Puppet Master applies to a given Slave, as well as the relationships between those resources. Catalogs are compiled by a Puppet Master from Manifests and Slave-provided data (such as facts, certificates, and an environment if one is provided), as well as an optional external data (such as data from an external Slave classifier, exported resources, and functions). The Master then serves the compiled Catalog to the Slave when requested.

**Puppet Installation**

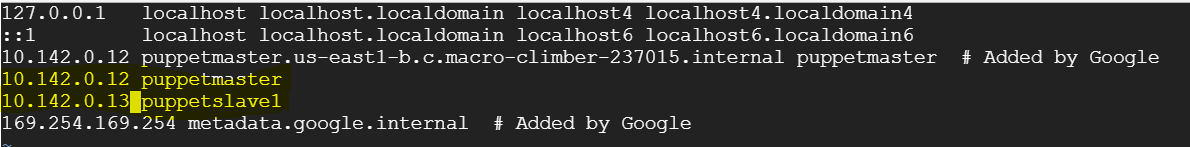
**First requirement:** Both the master and slave machines are talk to each other

Step 1: Go to /etc/hosts in Master machine i.e. Linux machine and

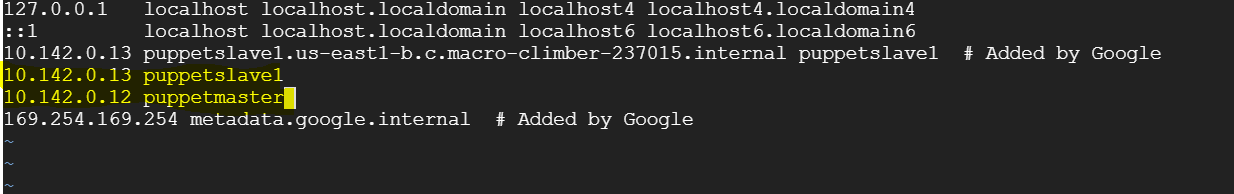
Vi /etc/hosts

Step 2:Enter internal ip address of master(10.142.0.12) and its name(puppetmaster)

Enter internal ip address of slave(10.142.0.13) and its name(puppetslave1)



**Do the same steps in Slave machine**



**Verify the machines are communicating each other**

In Master : ping ipaddress of the slavenode

**Eg: ping** 10.142.0.13

**In Slave:** ping ipaddress of the master

Eg: ping 10.142.0.12

**Second Requirement:** Install the dependencies before installing the puppet in master

Run the below command in master machine

rpm -Uvh <https://yum.puppetlabs.com/puppet5/puppet5-release-el-7.noarch.rpm>

**note:** this dependency command will work in CENTOS or Linux 7 version. If you are using ubuntu or Linux 6 the command will be different.

Check your OS by run the command : uname -a

Run the same dependency command on slave node

rpm -Uvh <https://yum.puppetlabs.com/puppet5/puppet5-release-el-7.noarch.rpm>

**Third Requirement:** Now install the puppet master/server on the master machine.

Command: yum install -y puppetserver

Install the puppet agent on the slave machine.

Command: yum install -y puppet

Now both puppet server and agent installed in master and slave respectively.

**Now configure the below settings:**

**In puppet master/server machine:**

Go to the folder structure: /etc/puppetlabs/puppet

Open the file puppet.conf

Commands:

cd /etc/puppetlabs/puppet

ll

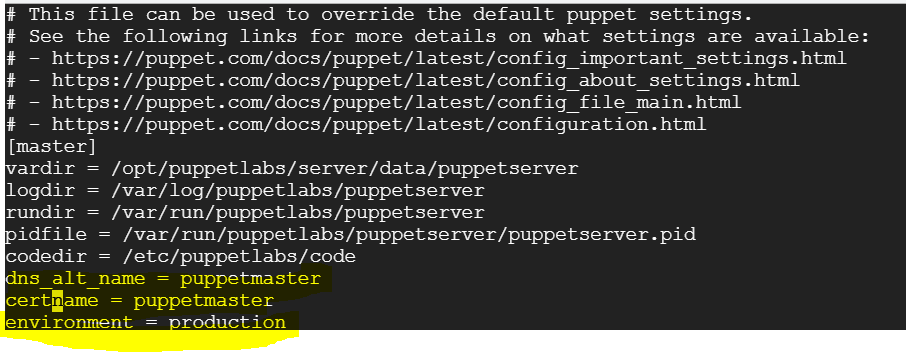
vi puppet.conf

And enter the below details in the puppet.conf file and save.

dns\_alt\_name = puppetmaster 🡪puppetmaster = master machine name

cetername = puppetmaster 🡪certificaiton name: you can given any name

environment = production



**In puppet agent/node machine:**

Go to the folder structure: /etc/puppetlabs/puppet

Open the file puppet.conf

Commands:

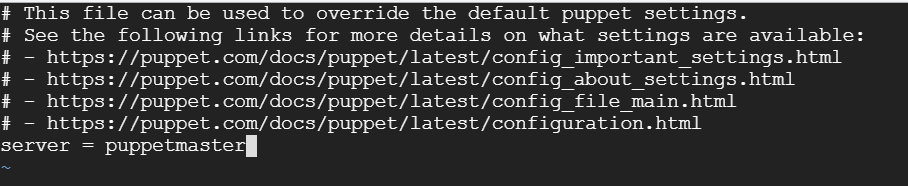
cd /etc/puppetlabs/puppet

ll

vi puppet.conf

And enter the below details in the puppet.conf file and save.

Server = puppetmaster 🡪server = name of master machine



Restart the master and slave machines.

**Start the puppetservice in Master machine:**

Command: Systemctl start puppetserver

**Go to Slave Machine:**

Comand to install first time your puppet service slave node is:

Command: puppet resource service puppet ensure=running enable=true

**To validate the connection between Slave node and master/server machine in slave node:**

Command: puppet agent –test

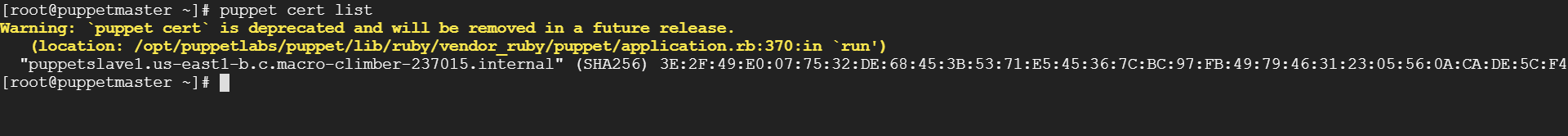
If you get the message “Exiting; no certificate found and waitforcert is disabled” i.e. there is no signed certificate

**In Master machine:**

To show the list of certificates in master: puppet cert list

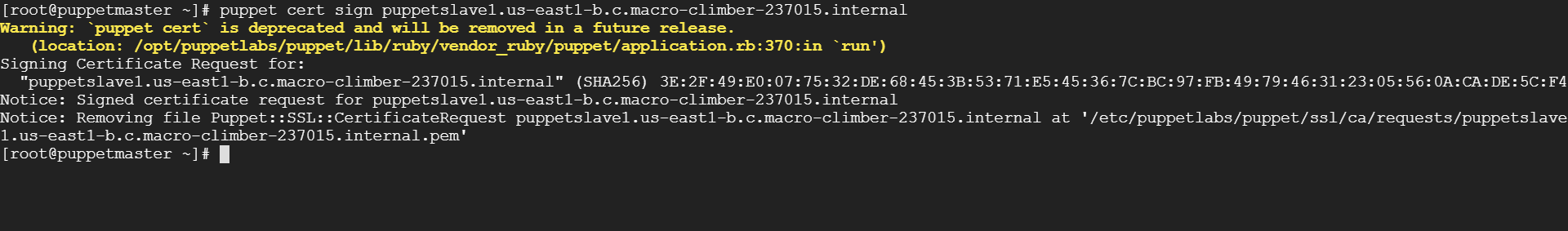
If you see the below message i.e one certificate is available but it is not signed and authorised.

**Warning: `puppet cert` is deprecated and will be removed in a future release. (location: /opt/puppetlabs/puppet/lib/ruby/vendor\_ruby/puppet/application.rb:370:in `run')** "puppetslave1.us-east1-b.c.macro-climber-237015.internal" (SHA256) 3E:2F:49:E0:07:75:32:DE:68:45:3B:53:71:E5:45:36:7C:BC:97:FB:49:79:46:31:23:05:56:0A:CA:DE:5C:F4[root@puppetmaster ~]# puppet sign cert puppetslave1.us-east1-b.c.macro-climber-237015.internal

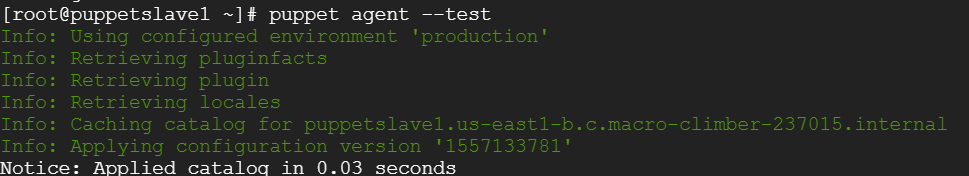


**To sign and authorized the certificate in master machine:**

Command: Puppet cert sign puppetslave1.us-east1-b.c.macro-climber-237015.internal



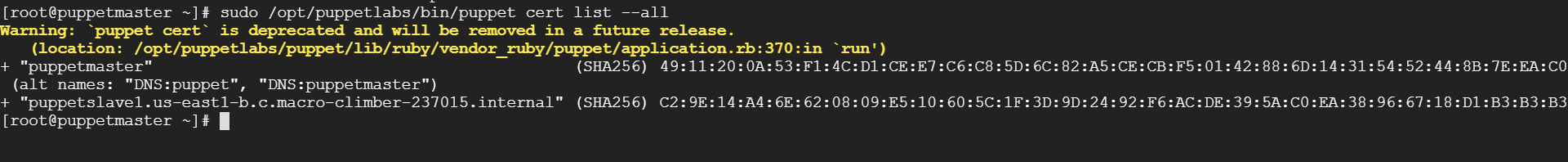
Run the command in slave node and see the connectivity between master and slave: puppet agent –test



**Command to check how many slave nodes are connected to master machine:**

Run this command in master machine:

Sudo /opt/puppetlabs/bin/puppet cert list –all



To see what are facts being sent from puppet agent/slave to master

Run the command in slave: facter

Command: facter

Puppet Manifest: Puppet programs are written in puppet DSL (Domain specific language).

Manifests are the puppet programs written in puppet DSL.

Manifest files are end with .pp

Manifest file contain resource declaration, file, user ,software , service etc.

Syntax for creating manifest file:

{‘title’:

Attribute ==> value,

}

Puppet follows a strict directory structre. We cannot create puppet manifest file in random directory .

We should create manifest file in master machine in below directory only:

cd **/etc/puppetlabs/code/environments/production/manifests/**

Interview question: Default location to create puppet manifest file: cd **/etc/puppetlabs/code/environments/production/manifests/**

Class 7 Recording:

Creating a file in root directory using manifest file:

node 'puppetslave1' {

file{'/info.txt':

ensure => 'present',

content => "This is our first file",

}

}

**Interview question**:

**To validate the syntax of the mainfest file:**

puppet parser validate main.pp

**To run the manifest file in slave i.e to communticate from slave to master :** puppet agent --test

**To install a package using manifest file**

Example: httpd package

node 'puppetslave1' {

file{'/info.txt':

ensure => 'present',

content => "This is our first file",

}

package{'httpd':

ensure => 'present',

}

}

**To start the service automatically after installing the package**

node 'puppetslave1' {

file{'/info.txt':

ensure => 'present',

content => "This is our first file",

}

package{'httpd':

ensure => 'present',

}

**service{'httpd':**

**ensure => 'running',**

**enable => 'true',**

}

}

Here to start the service automatically after installing ensure => ‘running’

To make sure the service is up and running even the slave machine is rebooted or restarted : enable => ‘true’

**To install Java using manifest file:**

use the below code

package{'java':

ensure => 'present',

}

**To change the defalut runinterval from 30 min to 1min:**

By default, a Puppet agent polls after every 30 minutes for any configuration changes on Puppet master.

Run interval could be checked using single command:

puppet agent --configprint runinterval

important link: <https://projects.puppetlabs.com/issues/11407>

Add the runinterval = 1m in puppet.conf file

Use the below commands in slave node

Cd /etc/puppetlbas/puppet/

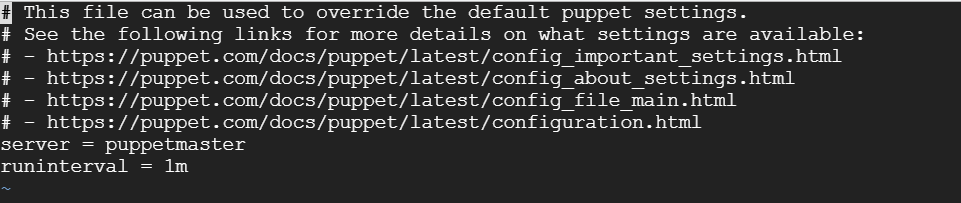
Ls

Vi puppet.conf

# This file can be used to override the default puppet settings.# See the following links for more details on what settings are available:# - <https://puppet.com/docs/puppet/latest/config_important_settings.html># - <https://puppet.com/docs/puppet/latest/config_about_settings.html># - <https://puppet.com/docs/puppet/latest/config_file_main.html># - <https://puppet.com/docs/puppet/latest/configuration.html>

server = puppetmaster

runinterval = 1m



**Class in Puppet:**

In puppet , to create reusable code/content we use class.

For example if we have 100 nodes/slave machines and if we want to push same configuration in 100 machines instead of writing everytime code we create class and we can reuse that class.

Example:

node 'puppetslave1' {

class{'linux1':}

}

class linux1{

file{'/info.txt':

ensure => 'present',

content => "This is our first file",

}

package{'httpd':

ensure => 'present',

}

package{'java':

ensure => 'present',

}

service{'httpd':

ensure => 'running',

enable => 'true',

}

}

**Array in Puppet:**

To install multiple packages at same time

class{'linux1':}

class{'linux':}

}

class linux1{

file{'/info.txt':

ensure => 'present',

content => "This is our first file",

}

package{'httpd':

ensure => 'present',

}

service{'httpd':

ensure => 'running',

enable => 'true',

}

}

class linux{

**$pkg\_list = ["git", "nano" , "java"]**

**package{ $pkg\_list:**

ensure => 'present',

}

}

**Puppet Modules:**

It’s a self bundled puppet code which is already inbuilt.

Modules are self-contained bundles of code and data. These reusable, shareable units of Puppet code are a basic building block for Puppet.

Modules follow directory structure.

/etc/puppetlabs/code/environments/production/modules

**Default location of Puppet Modules:** /etc/puppetlabs/code/environments/production/modules

Real time application Example: Deploying an application

Front end – php browser

Webserver – apache

Database – mysql

Version Control – GIT

Network setting: Firewall

**Example for Version Control:**

<https://github.com/wikimedia/mediawiki/tree/REL1_23>

**Steps:**

1)Create a custom module

2)Download inbuilt Modules from forge.puppet.com

3)use the downloaded module inside our custom module

Step 1: Creating Custom module with name mediawiki

Go to directory /etc/puppetlabs/code/environments/production/modules

Command to creat a custom module:

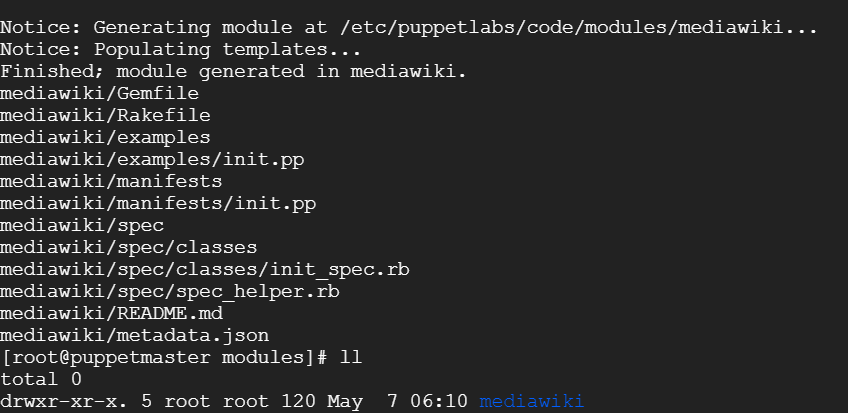
Syntax: puppet module generate authorname-modulename --environment=environmentname

Example: puppet module generate narayana-mediawiki --environment=production

It will ask some questions. If you want you can give else press enter

Give yes for the last question [About to generate this metadata; continue? [n/Y]]

Module will be created with name mediawiki



Go to mediawiki folder : cd mediawiki

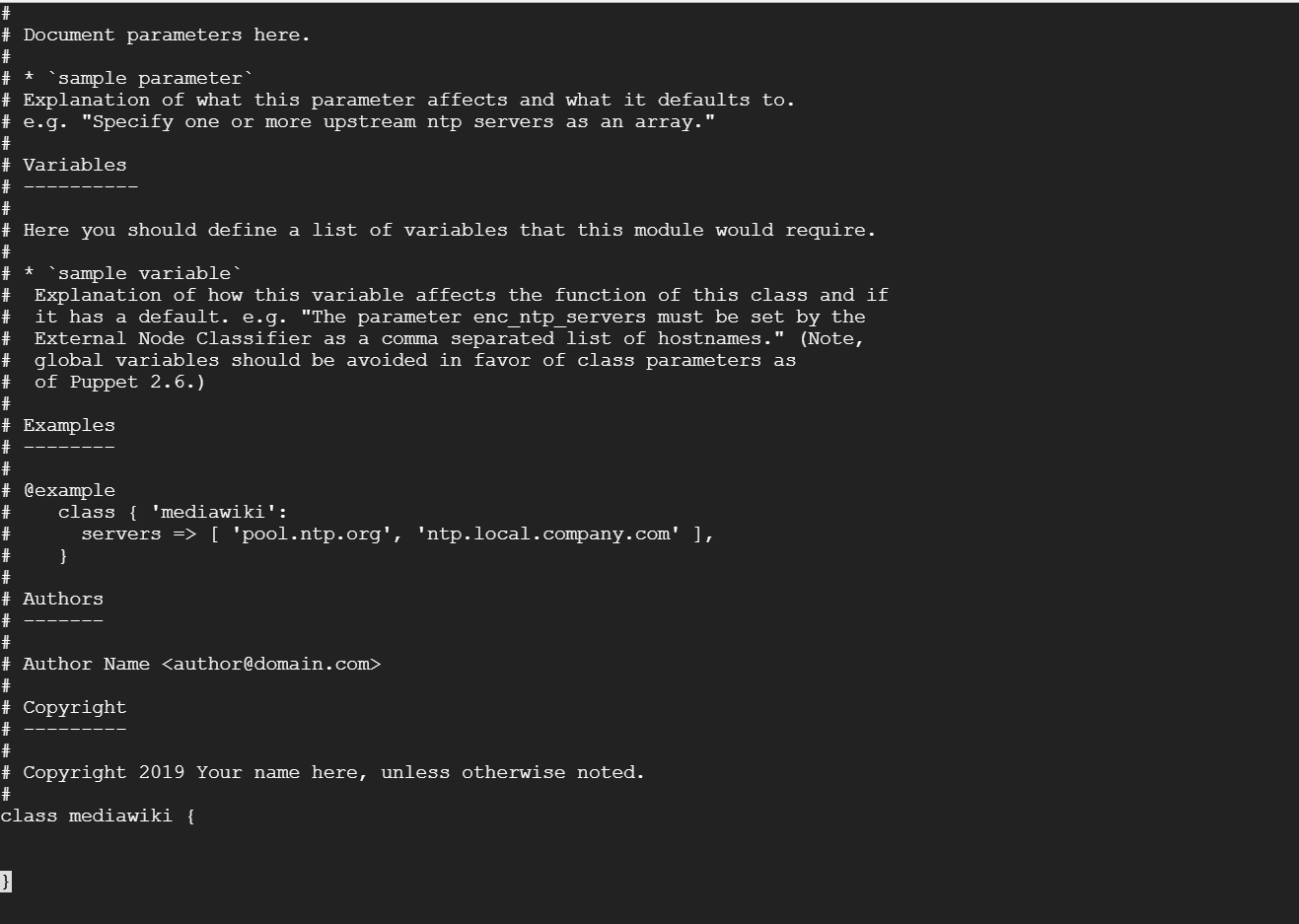
And see the manifest file get created

Go to manifests folder: cd manifests

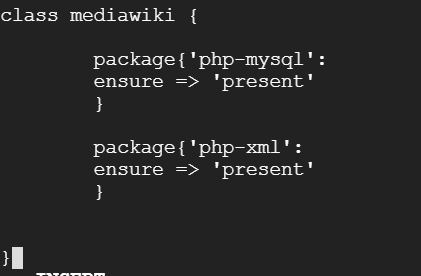
And see the init.pp file get created

Go to the init.pp file: vi init.pp

And see the class get created with the name of the module. Whenever we created custom module , a class will be created automatically with the modulename.



For php browser, it requries two resource declaration php-mysql and php-xml.



Whenever we run the command in slave puppet agent –test It will look for the puppet manifest files in /etc/puppetlabs/code/environments/production/manifests/

But here in this scenario when we made custom module mediawiki we have written our class in the init.pp file which is in folder /etc/puppetlabs/code/environments/production/modules/mediawiki/manifests

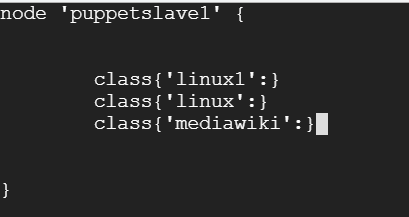
So inorder to run the file we need to call the class which is written inside the init.pp in main manifest file i.e. main.pp file

Go to the directory : cd /etc/puppetlabs/code/environments/production/manifests/

Open the file : vi main.pp

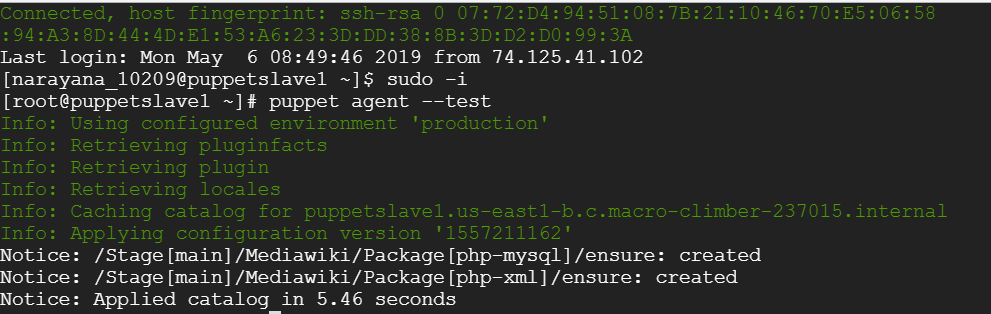
And give the class name: class{‘mediawiki’:}

Save the file

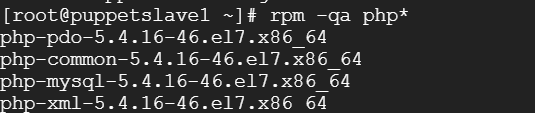


Now if we run the puppet agent –test it will look for the class inside the folder /etc/puppetlabs/code/environments/production and when it finds mediawiki class it will execute that class.

Run the command in slave node: puppet agent –test

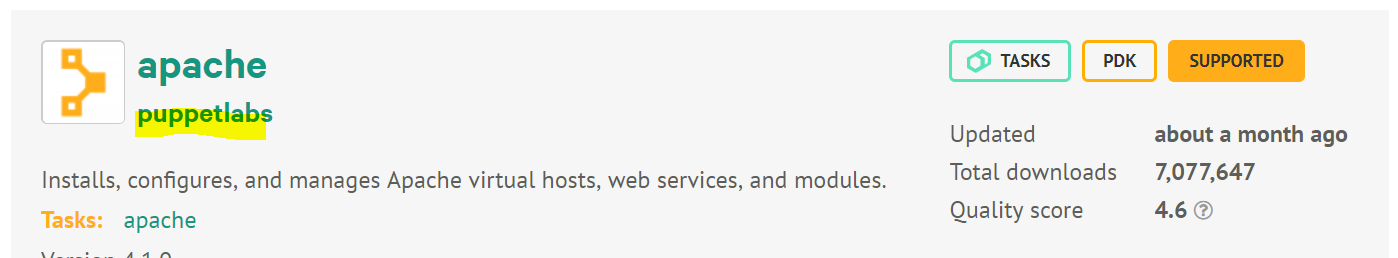


To validate whethere php is installed or not:



Step 2: Download the apache webserver module from forge.puppet.com

Use the module from puppetlabs only



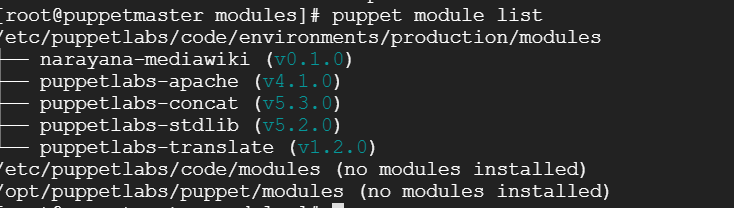
Go to the puppet module directory

cd /etc/puppetlabs/code/environments/production/modules

Run the command to intall apache webserver from forge.puppet.com

puppet module install puppetlabs-apache

To see the module list: puppet module list



**Step 3: use the downloaded module in custom module:**

Go to directory /etc/puppetlabs/code/environments/production/modules/mediawiki/manifests

Vi init.pp

class mediawiki {

package{'php-mysql':

ensure => 'present',

}

package{'php-xml':

ensure => 'present',

}

class{'::apache':

docroot => '/var/www/html',

mpm\_module => 'prefork',

subscribe => Package['php-mysql'],

}

class{'::apache::mod::php':}

}

class{'::apache':} –Here are calling the apache class

why we use :: is - rightnow we are in the custom module i.e. mediawiki. How the custom module work is if we declare any class in the custom module it will look for that class inside that module only. So if we don’t declare :: it will look for the class in mediawiki folder but the apache module we have downloaded in step 2 is outside the mediawiki module.So if we want to look for the class outside the module we need to give :: (called as top score)

**docroot => ‘/var/www/html ‘:** default location where the apache to be installed

**mpm\_module => ‘prefork’ :** if we have some other webserver other than apache running , now I always want to make sure that apache webserver is called. For that we use prefork**.** eventhough if any other webbrowser present it will only look for apache. Mpm means multi processing module.

If we give postfork it will look for all webservers other than apache webserver.

**subscribe => Package['php-mysql' :** It is a dependency parameter or meta parameter. When the apache webserver gets installed our front end service i.e. php web browser should gets restarted.

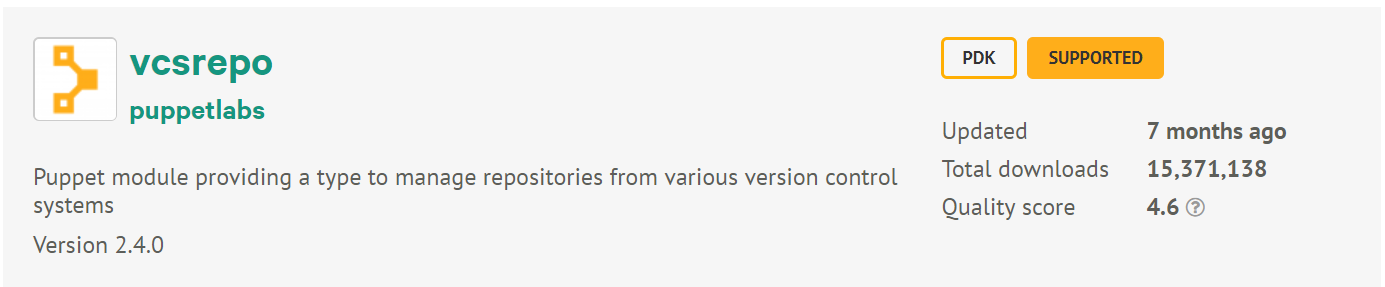
**Run the below command in slave**: puppet agent --test

Class 8 Recording:

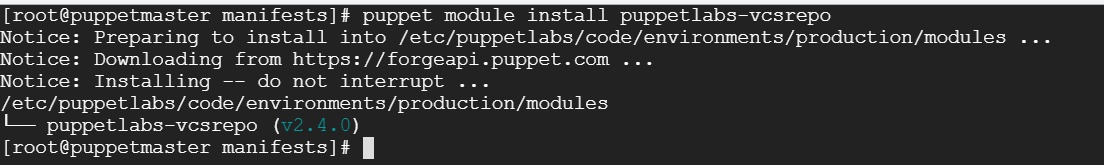
**How to import the source code from GIT:**

**EG:** <https://github.com/wikimedia/mediawiki/tree/REL1_23>

Use the module for version control GIT/GITLAB “vcsrepo”



Command to install vcsrepo: puppet module install puppetlabs-vcsrepo



Go the directory /etc/puppetlabs/code/environments/production/modules/mediawiki/manifests

Vi init.pp

Write the below code in init.pp file

vcsrep { '/var/www/html':

ensure => present,

provider => git,

source => 'https://github.com/wikimedia/mediawiki.git',

revision => 'REL1\_23',

}

‘/Var/www/hmtl’ : directory to store repo

Provider=>git means the version control system like GIT, GITLAB

Source : path of the repository

Revision : branch name of the repository

Save the file and run the command to validate: puppet parser validate init.pp

Entire code:

class mediawiki {

package{'php-mysql':

ensure => 'present',

}

package{'php-xml':

ensure => 'present',

}

class{'::apache':

docroot => '/var/www/html',

mpm\_module => 'prefork',

subscribe => Package['php-mysql'],

}

class{'::apache::mod::php':}

vcsrep { '/var/www/html':

ensure => present,

provider => git,

source => 'https://github.com/wikimedia/mediawiki.git',

revision => 'REL1\_23',

}

}

**Download the module mysql database from forge.puppet.com**

Puppet module install puppetlabs-mysql

Write the below code in init.pp

class{'::mysql::server':

root\_password => 'training',

}

Setting Firewall:

If we want to access the application by specific users or with specific ip address we can use firewall module.

Install the firewall module: puppet module install puppetlabs-firewall

Go the directory /etc/puppetlabs/code/environments/production/modules/mediawiki/manifests

Vi init.pp

class{'::firewall':}

firewall{'000 allow http access':

port => '80',

proto => 'tcp',

action => 'accept',

}

Firewall{'000 allow http access': 🡺 it is a message, we can give anytthing

Port=>80 if we want to run our application on port we can give any port

Proto 🡺 prototype like tcp, utp etc.

Action => if we want to run application on only port 80(given above) then give accept

Run the command in slave : puppet agent --test

**Important links:**

<https://www.digitalocean.com/community/tutorials/getting-started-with-puppet-code-manifests-and-modules>

<https://puppet.com/blog/managing-websphere-application-server-puppet>

<https://www.digitalocean.com/community/tutorials/how-to-create-a-puppet-module-to-automate-wordpress-installation-on-ubuntu-14-04>

<https://www.digitalocean.com/community/tutorials?q=puppet>

<https://puppet.com/blog/magic-directories-guide-to-puppet-directory-structure>

<https://codingcompiler.com/puppet-interview-questions-answers/>