# Puppet

**Introduction:**

Puppet is tool for configuration management and infrastructure management that is used for deploying, configuring and managing server machines

Can also be used for deployments

- Controls all the steps, right from bootstrapping to end of server life
- Can define configuration at the node level
- Can group them according to the roles
- Example: webserver database application
- Maintain consistency across nodes
- Example: if a change is done locally, it is rolled back to the original configuration.

Puppet is a tool for applying and managing system configurations across all nodes in org

Developed by puppet labs before a decade

A framework for system automation

A declarative Domain Specific Language – DSL

An Open source software written in ruby

Works on Linux, AIX, windows, mac OS

**Note:**

Note: Puppet server on windows is not supported.

But It can manage a windows client/node

**Link:** **https://docs.puppet.com/**

Puppet is free for managing 10 nodes.

It has GUI called puppet console (available in the puppet enterprise edition only)

Puppet word refers to entire puppet eco system (puppet master, slave etc)

It’s a collection of tools

Runs in client server relationship

Server acts a master, and applies configuration to multiple client systems using puppet agent

Chef and salt works in the same way

Mechanics are different but core concepts are identical.

Master is sole repository of information

Puppet is leading market, 70% of people are using puppet.

Developer develops some code, and sysops faces one issue

Developer says, its not supported on some version x.y.z

In olden days it was fine for 2 weeks delay.

But nowadays software delivery itself is for 2 weeks , cuts down productivity

At the end of the day it gets resolved but how long it is taking to resolve is the question.

Operational tasks

Provision

Deploy

Configure

Monitor

Scale

Secure

Application reliability and scalability directly affect business profitability.

We need control flexibility as well as ease of use

Automation script/puppet:

How long it takes to deploy 1000 servers

Puppet allows to deploy configs very quickly.

**Infrastructure as code:**

Track

Test

Deploy

Reproduce

Scale

Dev are expected to test before deploying a change.

System admins assume new change will work and they revert if that does not work.

So system admins has to keep track of version control of configuration files

At any given point of time , we should be able to roll back to a specific version.

Infrastructure changes are also required be tested before deployment

Ability to keep track of all config changes – is a code.

Who did what changes

When was application was rolled off

Puppet modules are on module forge and git hub

Software related to puppet:

**Facter:** system profiling library

Collects info about systems

Ex: Node ip address, interface, OS, disk free, disk used, memory , version, 32 bit 64bit.users etc

**Hiera**: manages key value look up .

| | |
|---|---|
| Memory free | 1GB |
| Data key | data value |

**Mcollective**: True power of Puppet

Allows to do Complex orchestration of actions, sequencing etc…

Run specific action against specific nodes

Example: Configure first 10 nodes     then these 2

CLI driven framework.

**PuppetDB:** Data generated by puppet stored here

We can change db to my sql also

Persistent data storage.

**Puppet Dashboard:** GUI/puppet front end /external node classifier.

The full pledged puppet console is available for only enterprise

**The Foremen:** third party provisioning tool

**Geppato:**  A puppet IDE based on Eclipse, can write puppet specific code.

Infrastructure as code:

Markets are constantly evolving and chaning

Agile – change software very rapidly

We have 5 releases ahead

Good practices:

Versioning of code –

Previously we used to wait till dev team has finished

Nowadays,

Dev changed config file

DevOps knows that

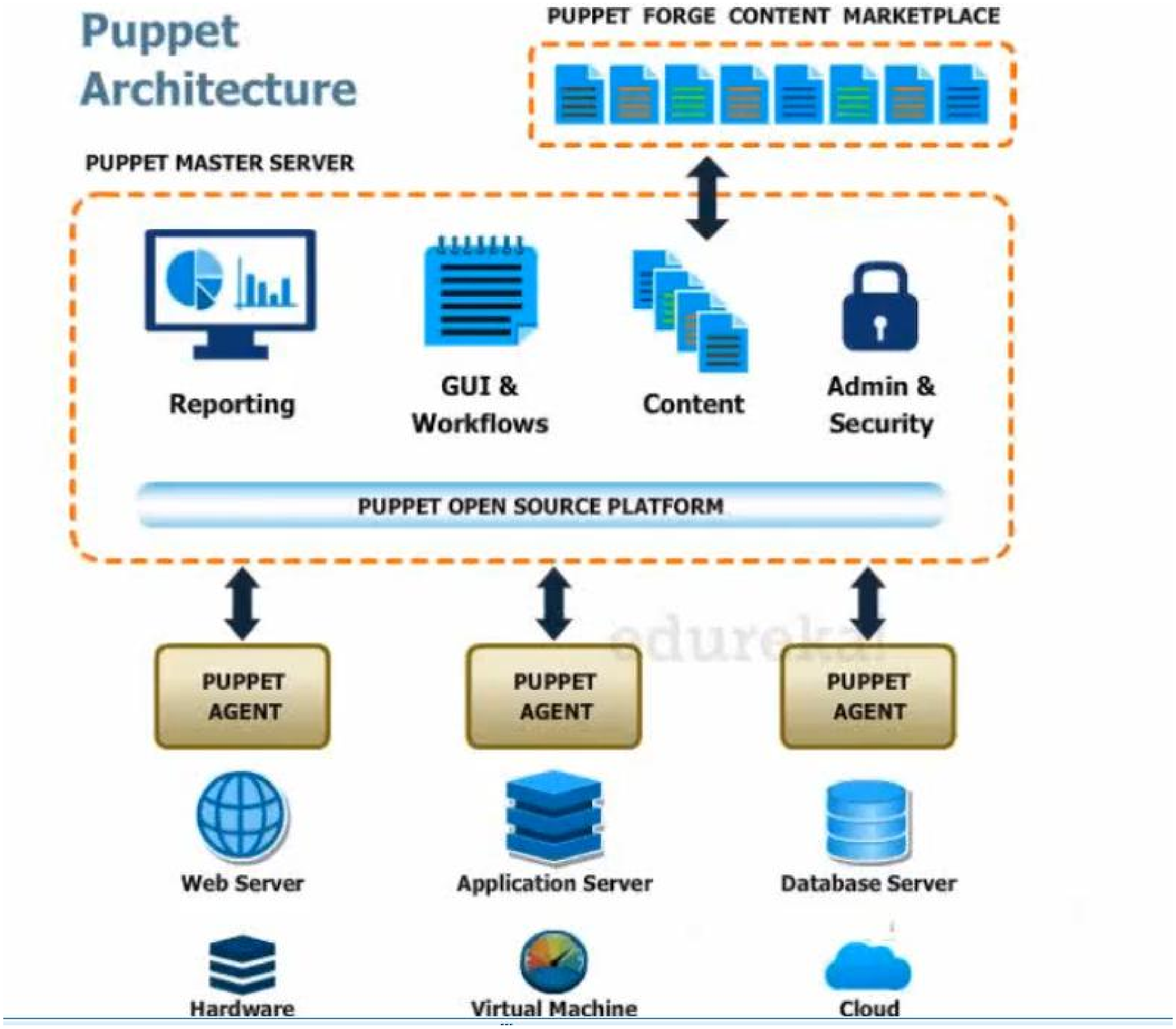
Dev is ready for deployment

Take it deploy it

Inf as code: Both dev and ops team work together to model the required infrastructure as CODE.

All my configurations are code   this should be the idea.

## Puppet Architecture:



Enterprize puppet requires

100GB disk

8GB-16GB RAM available memory

How puppet works:

**Puppet Master:** The machine contains all the config for different hosts. Puppet master will run as daemon on this master server

**Puppet Agent:**This daemon which runs on each node and talks to the master

**Connection:** Connection between these 2 machines is made in a secure encrypted channel with the help of SSL.

Autosign: when node is added to master, node automatically sends key to master.

–

**Installation & Configuration:**

On CentOS/RHEL 6.5:

***rpm -ivh https://yum.puppetlabs.com/el/6.5/products/x86_64/puppetlabs-release-6-10.noarch.rpm***

On CentOS/RHEL 7:

***rpm -ivh https://yum.puppetlabs.com/el/7/products/x86_64/puppetlabs-release-7-10.noarch.rpm***

Install server

***yum install puppet-server***

***chkconfig puppetmaster on***

***service puppetmaster start***

On CentOS/RHEL 6, where iptables is used as firewall, add following line into section ":OUTPUT ACCEPT" of /etc/sysconfig/iptables.

**I -A INPUT -m state --state NEW -m tcp -p tcp --dport 8140 -j ACCEPT**

```
# service iptables restart
```

On CentOS/RHEL 7, where firewalld is used, the same thing can be achieved by:

```
# firewall-cmd --permanent --zone=public --add-port=8140/tcp

# firewall-cmd --reload
```

**Client Installation**

```
  # yum install puppet
  # chkconfig puppet on
```

On client change the puppet server in below file

```
  bash-4.1# cat  /etc/sysconfig/puppet
# The puppetmaster server
PUPPET_SERVER=ctx1p14.in.ibm.com

# If you wish to specify the port to connect to do so here
#PUPPET_PORT=8140

# Where to log to. Specify syslog to send log messages to the system log.
#PUPPET_LOG=/var/log/puppet/puppet.log

# You may specify other parameters to the puppet client here
#PUPPET_EXTRA_OPTS=--waitforcert=500
```

```
  bash-4.1# cat /etc/puppet/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=ctx1p14.in.ibm.com

use_cached_catalog=true

    # Where puppetd caches the local configuration.  An

    # extension indicating the cache format is added automatically.

    # The default value is '$confdir/localconfig'.

    localconfig = $vardir/localconfig
```

```
    service puppet start

    puppet agent --test
```

With agent –test command nodes generate a certificate and sent it to server, server has to validate to establish communication.

on the pupper server

```
puppet cert list
```

## Note : To establish the communication between master and the node, ntp time sync should be there between master and the agent.

here there will be a entry for client request

```
puppet cert sign client-node
```

As each puppet agent runs for the first time it submits certificate signing request (CSR) to the ceritificate authority (CA) puppet master

To check outstanding requests:

```
puppet cert list
```

to sign a ceritificate

```
Puppet cert sign <name of client node>
```

**Puppet DSL:**

Resources

Variable and Facts

Resource relationships

Classes and modules

Classification

Other Terminology

DSL: Domain Specific Language

Puppet code is written in manifests (files with .pp extension)

Resources are grouped in classes

Classes and Configuration files are organized in modules.

When a client connects , the puppet master generates a catalog with the list of resources that the clients have to apply locally.

The puppet master has to classify the nodes and define it for each of them

The classes to include

Parameters to pass

Puppet env to use

By default all nodes are classified as production nodes

The catalog is generated by master according to the logic of our puppet code and data

Resource types are single units of configuration composed by

A type (package,service,file,user,mount,exec …)

A title (how is called and referred)

Zero or more arguments

```
Type { ‘title’:

argument => value,

Orhter_arg=>value

}
```

Example for a file resource:

```
file  {‘motd’:

path =>’/etc/motd’,

content =>’Tomorrow is another day’,

}
```

Install openssh :

Package { ‘openssh’ :

Ensure     present ,

}

Service { ‘httpd’

Ensure     running,

Enable -> true,

}

Variables:

Variable names are started with $

Assigned with = sign

Nested arrays can be used.

Strict mode=true, if a variable is not having value, compilation will fail.

Array:

[$a, $b, $c] = [1,2,3]

[$a, [$b,$c]] = [1,[$2,$3]]

[$a, $b] = [1,[2]]

$a=2

$b= [2]

Hashes:

[$a, $b] = {a    10, b    20}

By default undefined variable will have undef

But puppet will stop execution

```
File { “${homedir}/.vim”:

Ensure     directory.

}
```

```
$apache::params::vhostdir
```

To access a variable vhostdir variable from apache::params class

Scope:

Scope can be overridden in the local

Facts:

The command tool factor runs on clients and collects data that the server can use as variable

Facts are the system level values that cannot be changed

Ex:

```
Architecture     x86_64
```

Classes:

Classes are containers of different resources.

```
Class mysql (

{

Root_password     ‘default_value’,

Port     3306,

}

Package { ‘mysql-server’:
```

Ensure     present,

}

Ser vice { ‘mysql’:

Ensure     running

}

)

Packaging all the resources into one block

All packages services are put together in a class

Class definition

Class declaration ->  2 ways     include or class {‘my class’}

include my class

Resource relationships:

If A and B resources are defined

A

B

So default order is executed

We can defined relationship of resources using

1. Relationship metaparameters
2. Chaining arrows
3. Require() function

Before

A is before b

Automatically A will be applied first before b gets done

A requires B

Automatically B is applied first then A is applied

Notify –

A notify B

If A is changed, B gets notified, (may be B will be restarted/flushed)

Change configuration A

Restart apache B

If config is change, apache is restarted

B subscribe A

B says If A changes let me know

B will restart

Chaining operators

A     B   first run A then run B

A ~> B  if A is changed, refresh/restart/notify B

Require

Class A{

Require B

Require C}

Declaring a class

Class {         ‘mysql ‘:

Root_password     ‘my_value’

Port    ‘789’

}

Manifests can contain 4 or 5 or 10 classes.

If we want to separate functions and code data together

Modules can be used

Every single puppet manifest belong to a module

Only exception is site.pp

**Module tools:**

Install

Puppet module install puppetlabs-apache –version 0.0.2

List

Puppet module list

Puppet module uninstall  <>

Puppet module search <>  -    searches forge site

Create a directory production

Under /etc/puppe/environments/production/modules

Add a config file environment.conf

**Node.pp     pp stands for puppet programs**

**Cert dir:**

**/var/lib/puppet/ssl**

```
puppet-master#cat environment.conf

modulepath = /etc/puppet/environments/production/modules

environment_timeout = 5s

puppet-master#pwd

/etc/puppet/environments/production

puppet-master#
```

```
Node ‘web01’ {

Include apache

}
```

If this is include in site.pp

The node web01 starts executing class apache

```
Node ‘web01’, ‘web02’

{

Include apache

}
```

```
Node /^web\d+S/

{
```

```
Indlude apache

}
```

Web01,Web02 ,web03 .. web100

Use if ip address is discouraged

Need to use FQDN

**Manifests:**

Puppet programs are called manifests

Manifests are composed of puppet code and their file names use .pp extension

The default manifest installed via apt is:

/etc/puppet/manifests/site.pp

```
    mkdir -p /etc/puppet/manifests
```

## Site.pp vs node.pp

By defaults clients get the changes from the master on pull request mode, for every 30 min

If the changes have to be reflected automatically use puppet agent –t on the clients

```
[root@ctx1p21 manifests]# cat site.pp

import 'nodes.pp'
```

While pulling requests, master checks for the manifests in site.pp

Here site.pp is importing nodes.pp

Nodes.pp can have individual nodes manifests as below

```
[root@ctx1p21 manifests]# cat nodes.pp

node 'ctx1p13.XX.XXX.com'

{

        file { '/etc/motd':

               content => "this is a file puppet MMMMaster\n"

        }

        package { 'tigervnc':

                ensure => present,

        }

        user { 'john1':

                ensure => present,

                comment => 'John user' ,

                home => '/home/john1' ,

                managehome => true,

                password =>
'$6$TEwb3UaixYC5.L2a$KunE.GvPMz2QEHIKBrP/v0G6vGEdJntuiZLzt8p3EdMaZ6V7S
LL2WpPaY6.RsZ4IG3zI9TLQVfVNZlSyuz23g0'

        }

}
```

The above example modifies /etc/motd

Installs tigervnc packages

And creates a user called john

We can also have multiple nodes in the node.pp

```
node ‘ctx1p13’ , ‘ctx1p14’, ‘ctx1p15’

{

Include apache

}


node /^ctx\d+$/

{

Include apache

}
```

and add below code:

```
  node 'client-node' {

        include custom_utils

}


class custom_utils {

        package { ["nmap","telnet","vim-enhanced","traceroute"]:

                ensure => latest,

                allow_virtual => false,

        }
}
```

```
   service puppetmaster restart
```

```
puppet agent -t

puppet agent -t --debug

puppet agent -t --noop
```

Puppet terminalogy:

Example of a manifest

Resource declaration for user john:

To list default resource types that are available to puppet, enter following

Puppet resources –types

```
user { 'mitchell':
  ensure     => present,
  uid        => '1000',
  gid        => '1000',
  shell      => '/bin/bash',
  home       => '/home/mitchell'
}
```

The below example stops a process

```
service { 'multipathd':
  ensure => 'stopped',
  enable => 'false',
}
```

The below example removes the package

```
Package { ‘ntp’ :
Ensure => absent,
}
```

```
import "templates.pp"
import "nodes.pp"
import "classes/*"
import "groups/*"
import "users/*"
import "os/*"
```

/manifests/classes/ – Directory containing all classes

/manifests/site.pp – the primary manifest file

/manifests/templates.pp – Contains template nodes

/manifests/nodes.pp – Contains node definitions

/manifests/definitions/ – Contains all definitions

/manifests/groups/ – Contains manifests configuring groups

/manifests/os/ – Contains classes designed to configure nodes with particular operating systems

/manifests/users/ – Contains manifests configuring users

/manifest/files/ – Contains file server modules for Puppet distributable files

```
file { ‘/tmp/x/x.txt’ :
          Ensure => present,
}
```

This manifest will fail if the directory /tmp/x does not exists

```
file {‘/tmp/x/x.txt’:
          Ensure => present,
}
file { ‘/tmp/x’  :
          Ensure => directory,
}
```

Ideally this manifest should fail as /tmp/x does not exists for the first time

But puppet is intellingent enough to create directory first then create file.

Between 2 resources puppet finds the relationship between them and creates directory

```
file { ‘x’:
          Ensure =>directory,
          Path => ‘/tmp/x’
}
file {  ‘r’  :
          ensure =>present,
          path => ‘/tmp/i/x.txt’,
}

file {  ‘i’  :
          ensure => directory,
          path => ‘/tmp/i’,
          before  => File [‘r’],
}
```

So here with use of before keyword,

File r is created before creating file called i

So the sequence is

Create file x

Create directory r

Create file i

**Class:**

**Classes are containers of different resources.**

We write code in classes

We don’t have to use classes, but we recommend to use classes

Because its easy to read manifests with classes

Code can be resued

Class defination:

Does not execute any code:

```
Class example

{

Code

}
```

Class declaration : executes the code

Normal class – occurs when inlcude keyword is called

Resource like class – declaration occurs when a class is declared like a resource like class {‘example’}

```
class apache {

     package { 'httpd':

       ensure => 'present',

           }

     service {'httpd':

       ensure => 'running',

       require => Package["httpd"],
          }
}
include apache
```

In production the site.pp looks like this

**Modules:**

Path: /etc/puppet/modules

Modules are for modularity, reusability.

Modules are self-contained bundles of code and data

Modules can be downloaded from puppet forge.

Every single manifest belong to a module.

**Build small single-purpose modules**

The site.pp manifest, contains site wide node specific code, does not belong to any module

Some more examples of manifests

**How to execute a command using puppet**

```
Class command_exec
{
exec  { ‘some command’
command => ‘some command’,
path => ‘/usr/sbin/’
}

}
```

**Facter:**

Facter –p is the command to show all values

## Hiera

A key value lookup tool

Can be organized and ordered nicely without touching the actual code

Just give hiera the data your module need

Hiera makes data separate from modules, so that module code can be untouched

Hiera data separation can be best explained with below example:

Ssh module without hiera:

```
class sshdconfig {

    case $::osfamily {

        Debian: {

            $serviceName = 'ssh'

        }

        RedHat: {
            $serviceName = 'sshd'
        }
    }

    file { "/etc/ssh/sshd_config":
        owner   => 'root',
        group   => 'root',
        mode    => '0644',
        #content => template("$module_name/sshd_config.erb"),
        content => template("sshdconfig/sshd_config.erb"),
        notify  => Service[$serviceName],
    }

    service { $serviceName:
        ensure => 'running',
        enable => 'true',
    }
}
include sshdconfig
```

Here the data is written in the manifest

If there are several os flavours we need to change the code every time, instead we can use hiera to get the data

As below:

```
class sshdconfig ( $serviceName = hiera("sshservicename") ){
    file { "/etc/ssh/sshd_config":
        owner   => 'root',
        group   => 'root',
        mode    => '0644',
        source => "puppet:///modules/sshdconfig/sshd_config",
        notify  => Service[$serviceName],
    }

    service { $serviceName:
        ensure => 'running',
        enable => 'true',
    }
}
include sshdconfig
```

here hiera is a function which can be used in manifests

:backends – Hiera supports yaml, json and puppet class backends.

```
[root@ctx1p13 sshdconfig]# pwd
/etc/puppet/modules/sshdconfig
[root@ctx1p13 sshdconfig]# puppet apply --test manifests/init.pp
Notice: Compiled catalog for ctx1p13.in.ibm.com in environment production in 0.20 seconds
Info: Applying configuration version '1478739772'
```

```
Notice: Finished catalog run in 0.09 seconds

[root@ctx1p13 sshdconfig]#
```

If there is no change in config file the module is never executed

Make change sshd_config and it executes module.

```
[root@ctx1p13 sshdconfig]# puppet apply --test manifests/init.pp

Notice: Compiled catalog for ctx1p13.in.ibm.com in environment production in 0.20 seconds

Info: Applying configuration version '1478739821'

Notice: /Stage[main]/Sshdconfig/File[/etc/ssh/sshd_config]/content:

--- /etc/ssh/sshd_config        2016-11-10 06:32:43.452946133 +0530

+++ /tmp/puppet-file20161110-30843-1ccq8q8-0    2016-11-10 06:33:41.595946106 +0530

@@ -1,4 +1,4 @@

-    #10th Nov 2nd run This file is managed by Puppet - Local changes will be DESTROYED.

+    #10th Nov 3rd rdun This file is managed by Puppet - Local changes will be DESTROYED.

 #

 #       $OpenBSD: sshd_config,v 1.80 2008/07/02 02:24:18 djm Exp $



Info: Computing checksum on file /etc/ssh/sshd_config
```

```
Info: /Stage[main]/Sshdconfig/File[/etc/ssh/sshd_config]: Filebucketed
/etc/ssh/sshd_config to puppet with sum
22dea011d255c836b6cc32ce7cfd273a

Notice: /Stage[main]/Sshdconfig/File[/etc/ssh/sshd_config]/content: content
changed '{md5}22dea011d255c836b6cc32ce7cfd273a' to
'{md5}fa81e2e095f2b34e7a7b4a153d98abe8'

Info: /Stage[main]/Sshdconfig/File[/etc/ssh/sshd_config]: Scheduling refresh
of Service[sshd]

Notice: /Stage[main]/Sshdconfig/Service[sshd]: Triggered 'refresh' from 1
events

Notice: Finished catalog run in 0.33 seconds

[root@ctx1p13 sshdconfig]# ls -lrt /etc/ssh/sshd_config

-rw-r--r-- 1 root root 3997 Nov 10 06:33 /etc/ssh/sshd_config

[root@ctx1p13 sshdconfig]# date

Thu Nov 10 06:33:54 IST 2016

[root@ctx1p13 sshdconfig]# pwd

/etc/puppet/modules/sshdconfig

[root@ctx1p13 sshdconfig]#
```

**Catalog:**

Complete list of resources and their relationships that the puppet master generates for the client

Applied on the client after it has been received from master

Client uses RAL – Resource abstraction layer to execute actual sytem commands that convert abstract resources likes

Package {‘openssh’}

To their actual fulfillment on th system (apt-get, yum)

Catalog is saved by default:

/var/lib/puppet/client_data/catalog/$certname.json

Creating a custom module:

***puppet module generate santosh-devops***

go inside Santosh-devops

cd manifests

open init.pp

```
[root@reviewb manifests]# cat site.pp
import 'node11.pp'
[root@reviewb manifests]# cat node11.pp
node 'ctx2p06.in.ibm.com'
{

class {'apache':}
class {'devops':}
}

class apache {
      package { 'httpd':
        ensure => 'present',
                }
        service {'httpd':
           ensure => 'running',
           require => Package["httpd"],
           }
}

[root@reviewb modules]# cd devops/
[root@reviewb devops]# ls
Gemfile  manifests  metadata.json  Rakefile  README.md  spec  tests
[root@reviewb devops]# tree
.
├── Gemfile
├── manifests
│   └── init.pp
├── metadata.json
├── Rakefile
├── README.md
├── spec
│   ├── classes
│   │   └── init_spec.rb
│   └── spec_helper.rb
└── tests
    └── init.pp
```

```
4 directories, 8 files


[root@reviewb devops]# cd manifests/

[root@reviewb manifests]# ls -lrt

total 4

-rw-r--r--. 1 root root 1194 Mar 19 06:32 init.pp

[root@reviewb manifests]# cat init.pp

# == Class: devops

#

# Full description of class devops here.

#

# === Parameters

#

# Document parameters here.

#

# [*sample_parameter*]

#   Explanation of what this parameter affects and what it defaults to.

#   e.g. "Specify one or more upstream ntp servers as an array."

#

# === Variables

#

# Here you should define a list of variables that this module would require.

#

# [*sample_variable*]

#   Explanation of how this variable affects the funtion of this class and if
#   it has a default. e.g. "The parameter enc_ntp_servers must be set by the
#   External Node Classifier as a comma separated list of hostnames." (Note,
#   global variables should be avoided in favor of class parameters as
#   of Puppet 2.6.)
#
# === Examples
#
#  class { 'devops':
#    servers => [ 'pool.ntp.org', 'ntp.local.company.com' ],
#  }
#
# === Authors
#
# Author Name <author@domain.com>
#
# === Copyright
#
# Copyright 2017 Your name here, unless otherwise noted.
#
class devops {
$phpmysql = $osfamily ? {
'RedHat' => 'php-mysql',
'debian' => 'php5-mysql',
default => 'php-mysql',
}
package {$phpmysql:
ensure =>  'present',
}

if $osfamily == 'RedHat'{
package {'php-xml':
ensure =>  'present',
}
}
}


class {'::apache':
docroot => '/var/www/html',
mpm_module => 'prefork',
subscribe => Package[$phpmysql],
}

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

**Commonly used modules:**

vcsrepo – change source code for version control like git

mysql

php

**How to download a module:**

Go to puppet forge site

Search in modules, download using the module command given