Experiment No. 11

Aim: To learn Software Configuration Management and provisioning using Puppet Blocks (Manifest, Modules, Class, Function).
LO No. & Statement: (LO6): Synthesize software configuration and provisioning using Ansible.
Theory:
What is Configuration Management?

Configuration management is a system engineering process for establishing consistency of a product's attributes throughout its life. In the technology world, configuration management is an IT management process that tracks individual configuration items of an IT system. IT systems are composed of IT assets that vary in granularity. An IT asset may represent a piece of software, or a server, or a cluster of servers. The following focuses on configuration management as it directly applies to IT software assets and software asset CI/CD.

Software configuration management is a system engineering process that tracks and monitors changes to a software system's configuration metadata. In software development, configuration management is commonly used alongside version control and CI/CD infrastructure. This post focuses on its modern application and uses in agile CI/CD software environments.

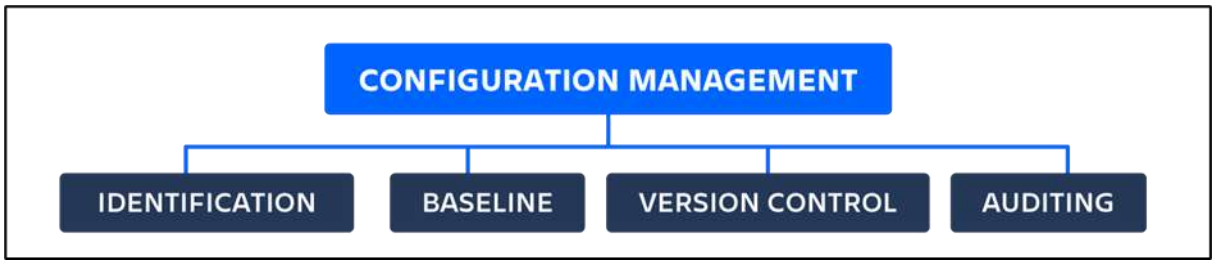

Figure 1: Configuration Management

Puppet:

Puppet is a configuration management tool developed by Puppet Labs in order to automate infrastructure management and configuration. Puppet is a very powerful tool that helps in the concept of Infrastructure as code. This tool is written in Ruby DSL language which helps in converting a complete infrastructure into a code format, which can be easily managed and configured.

Puppet follows the client-server model, where one machine in any cluster acts as the server, known as the puppet master, and the other acts as a client known as a slave on nodes. Puppet has the capability to manage any system from scratch, starting from initial configuration till the end-of-life of any particular machine.

Puppet Features:

The following are the features of Puppet:

- Platform Support: Puppet is compatible with all platforms that support Ruby, like Microsoft Windows, Linux, MacOS X, etc.
- Scalable: Puppet was developed in 2005; therefore, many different organizations, including medium and large, have deployed Puppet, and hence its scalability is very large.

- Documentation: Puppet provides a large number of well-developed wiki pages with detailed documentation.
- Idempotency: Unlike other configuration management tools, in Puppet, we can safely run the same set of configurations multiple times on the same machine. This means, after deploying a configuration on any machine, the puppet keeps verifying those configurations at certain intervals.
- Open-Source: A puppet is an open-source tool, and because of this feature, it is easy to extend it to build custom libraries and modules.
- Reporting Compliance: The enterprise version of the puppet supports graphical reporting with the help of this you can simply visualize the infrastructure, communicate, and quickly respond to the modifications. It provides you the real-time visibility into the effects of changes, which allows you to see what's going on in your infrastructure.
- Cost-Effective: When you have many numbers of systems and want to make some minor code changes, then Puppet helps to reduce the effort and cost.
- Faster: Puppet allows DevOps professionals and System Administrators to work more quickly and effectively.
- Growing Fast: Today, many companies have adopted puppets to manage their infrastructure, such as Google, Red Hat, AT&T, Spotify, AON, US Air Force, etc.

Puppet Blocks:

1. Puppet Manifests: Puppet Manifests are the text documents written in Puppet DSL that describes the end state of the host system. It can be created using any text editor and the extension of it .pp extension.

   Example: The following noalice.pp will ensure that Puppet deletes an account called alice from the system.

   [root@master ~]# vim alice.pp

   [root@master ~]# cat noalice.pp

   user {'alice':

     ensure => 'absent',

   }

2. Puppet Modules: The puppet module is a way of packaging puppet manifests and related files into a single file. The module is a tar archive that has an established hierarchy. The puppet module provides a standard, predictable directory structure for the puppet code and related files that they contain. Some of the essential or most frequently used directories in a module hierarchy are listed below:

- Manifests directory contains the entire puppet manifests defined in the module.
- File directory contains static files
- lib/facter directory contains custom fact definitions
- The test directory contains manifests that are used to test the functionality provided by the module.

Example:

```
file {
  '/etc/vmrc':
    ensure => 'directory',
    owner  => 'root',
    group  => 'root',
    mode   => '755';

  '/etc/rc.d/init.d':
    group  => 'root',
    mode   => '755';
}
    ensure => 'directory',
    owner  => 'root',
```

3. Puppet Classes: Puppet Classes: Puppet Classes helps to ensure that the Puppet resource definitions can be made more robust and reusable so that they can be applied to multiple hosts. A puppet class is usually used to define all the resources that are needed to implement a service or run an application. The following example demonstrates the syntax of the class

Example:

```
class class_name ($param = 'value') {
  resource definitions...
}
```

The following example demonstrates a class called test

```
class test {
  user { 'master':
    ensure  => 'present',
    home    => '/home/master',
    shell   => '/bin/bash',
  }
  file { '/home/master':
```

```
  ensure  => 'directory',
  owner   => 'master',
  group   => 'master',
  mode    => '0770',
  require => User['master'],
 }

 package { 'httpd':
  ensure  => 'present',
 }

 service { 'httpd':
  ensure  => 'running',
  enable  => true,
  require => Package['httpd'],
 }

}
```

include test # include is a function that is used to call the class test in the manifest

4. Puppet Functions: During the catalog compilation on the master, plugins called functions are called. When the puppet manifests calls a function, it returns a value or makes some changes that modify the catalog.
   The code in the function is written in Ruby, which performs a number of things to produce the final value. The functions perform the following tasks:
   - Evaluating the templates
   - Performing mathematical calculations
   - Modifies the catalog

   Puppet includes many built-in functions. Custom functions can also be composed and used in the modules.

   The syntax used to built-in the customized functions is as below:

   function <MODULE NAME>::<NAME>(<PARAMETER LIST>) >> <RETURN TYPE> {

   ... body of function ...

final expression, which will be the returned value of the function

}

Some of the built in functions are template, alert, include, map and many more.

Statement Functions

Statement functions are a group of built in functions which are only used to modify the catalog. The built in statement functions are:

- Catalog Statements like include, require, contain and tag
- Logging statements like debug, info, notice, warning, err.
- Failure statements like fail.

Conclusion:

From this experiment, it is concluded that we have successfully learned Software Configuration Management and provisioning Puppet Blocks (Manifest, Modules, Class, Function).

Hence, with this experiment, we have achieved Lab Outcome LO6.

We have also achieved Program Outcomes PO1, PO2, PO3, PO4, PO5, PO12.