Name: Mrunal Vaidya XIE ID: 202003060
Roll : 68 Batch: C

Experiment 11

Aim: To learn Software Configuration Management and provisioning using Puppet Blocks (Manifest, Modules, Class, Function).

LO: **LO6:** Synthesize software configuration and provisioning using Ansible.

Theory:

Configuration Management:
Configuration management is a systems 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, 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 systems engineering process that tracks and monitors changes to software systems 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.

Puppet:

- Puppet is a **DevOps configuration management tool**. This is developed by Puppet Labs and is available for both open-source and enterprise versions. It is used to centralize and automate the procedure of configuration management.
- This tool is developed using Ruby DSL (domain-specific language), which allows you to change a complete infrastructure in code format and can be easily managed and configured.
- The puppet tool deploys, configures, and manages the servers. This is used particularly for the automation of hybrid infrastructure delivery and management.
- With the help of automation, Puppet enables system administrators to operate easier and faster.
- Puppets can also be used as a deployment tool as they can deploy software on the system automatically. Puppet implements infrastructure as a code, which means that you can test the environment for accurate deployment.
- puppet supports many platforms such as Microsoft Windows, Debian/Ubuntu, Red Hat/CentOS/Fedora, MacOS X, etc.

- Puppet uses the client-server paradigm, where one system in any cluster works as the server, called the puppet master, and the other works as a client on nodes called a slave.

Puppet Blocks

Puppet provides the flexibility to integrate Reports with third-party tools using Puppet APIs. **Four types of Puppet building blocks are**

- Resources
- Classes
- Manifest
- Modules

Puppet Resources:

- Puppet Resources are the building blocks of Puppet.
- Resources are the **inbuilt functions** that run at the back end to perform the required operations in puppet.

Puppet Classes:

- Puppet classes are a set of puppet resources that are grouped together as a single unit. Classes are used to model the fundamental aspects of the node. Puppet uses classes to make the structure reusable and organized. Classes can only be evaluated once per node.

- Classes are described within the manifest file, located inside Puppet modules. The main reason for using a class is to decrease the duplication of the same code inside any manifest file or other puppet code.

- Before using a class, we have to define it, which is done by the class keyword, the name of the class, curly braces, and a set of code. This part of the code does not automatically evaluate the code.

- Syntax for defining and declaring a class:

Puppet Manifest:

- In puppet, all the programs are written in Ruby programming language and added with an extension of .pp known as manifests. The full form of .pp is the puppet program.
- Manifest files are puppet programs. This is used to manage the target host system. All the puppet programs follow the puppet coding style.

- We can use a set of different kinds of resources in any manifest, which is grouped by definition and class.
- Puppet manifest also supports the conditional statement. The default manifest file is available in the /etc/puppet/manifests/site.pp location.
- **Manifest Components**

    - Puppet manifest has the following components:
        - **Files:** Files are the plain text files that can be directly deployed on your puppet clients. Such as yum.conf, httpd.con, etc.
        - **Resources:** Resources are the elements that we need to evaluate or change. Resources can be packages, files, etc.
        - **Templates:** These are used to create configuration files on nodes and which we can reuse later.
        - **Nodes:** Block of code where all the information and definition related to the client are defined here.
        - **Classes:** Classes are used to group different types of resources.

Puppet Modules:

- The puppet Module is a collection of files, classes, templates, and resources. Each module handles a specific task in your infrastructure, such as installing and configuring a piece of software.
- Since modules allow you to divide your code into multiple manifests, it is very helpful in organizing your puppet code. Modules are the reusable and shareable units in the puppet ● Modules must be installed in the puppet module path. And the module path is the /etc/puppet/modules directory.
- We have two partitions in any Puppet module, which allows us to define the structure of code and control the denominations.
- The modules search path is configured in the puppetmaster or mastered and uses a colon separated list of directories. The other sections are configured with the module path parameter. i.e.
- And the second partition is access control settings in fileserver.conf for the file server modules. We don't need to specify the path for that module, and if you specify the path, it will give a warning.

Name: Mrunal Vaidya XIE ID: 202003060
Roll : 68 Batch: C

Puppet Function

- As we know, Puppet uses Ruby programming language, and like other programming languages, Ruby also supports the function. We can write the functions in Ruby language which can be distributed in puppet modules. Puppet provides two different types of functions:
- **statement:** This type of function did not return any value and is used to perform standalone tasks. It can be used to import Puppet modules in the new manifest file.
- **rvalue:** In Puppet, if you want to define a function with its return type, then you can use rvalue functions. rvalue can only be used when the statement needs a value, like a case statement or assignment. A function can only take two parameters as an argument.

Features of Puppet

1. **Platform Support:** Puppet is compatible with all platforms that support Ruby, like Microsoft Windows, Linux, MacOS X, etc.

2. **Scalable:** Puppet was developed in 2005; therefore, many different organizations, including medium and large, have deployed Puppet, and hence its scalability is very large.

3. **Documentation:** Puppet provides a large number of well-developed wiki pages with detailed documentation.

4. **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.

5. **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.

6. **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 real-time visibility into the effects of changes, which allows you to see what's going on in your infrastructure.

7. **Cost-Effective:** When you have many systems and want to make some minor code changes, then Puppet helps to reduce the effort and cost.

8. **Faster:** Puppet allows DevOps professionals and System Administrators to work more quickly and effectively.

9. **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.

Conclusion:

From the above experiment, I understood what a Puppet block is i.e Manifest, resource, classes, and modules, and also learned about the Puppet function. And hence, with this experiment we have achieved the Lab Outcome Six (LO6).

POs Achieved: PO1, PO5, PO12.