Semester	Semester VIII	
Subject	DevOps Lab	
Subject Professor In-charge	Prof. Yash Shah	
Laboratory	DevOps	
Student Name	Ashwini Jadhav	
Roll Number	17101B0038	
Grade and Subject Teacher's Signature		

Experiment Number	8	
Experiment Title	To use conditional statements, loops in Puppet.	
Resources / Apparatus Required	Hardware: Compatible Computer System	Software: Linux, Docker, Puppet
Objectives	Writing Script in puppet using loops and conditional statements.	
Theory	**Puppet:-** o 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. o 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. o Puppet tool deploys, configures, and manages the servers. This is used particularly for the automation of hybrid infrastructure delivery and management. o With the help of automation, Puppet enables system administrators to operate easier and faster. o Puppet can also be used as a deployment tool as it 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 other works as a client on nodes called a slave.

Puppet Coding Style:

In Puppet, the coding style describes all the requirements that must be followed when attempting to transform infrastructure on the system configuration into code. Puppet requires resources to work and execute all of its defined tasks.

As we know, the puppet employs **Ruby language** as its encoding language, which provides several predefined features, and with the help of these features, it is very easy to complete the things with the simple configuration on code.

Resources:

In puppet, resources are the basic unit used for modeling the system configurations. Resources are the building blocks of a puppet. Each resource describes the desired state for some aspects of a system, such as service, file, and package.

Resources are the predefined functions that allow the users or developers to develop custom resources, with the help of which we can manage any particular unit of a system. Resources in the puppet are aggregated together by using either "define" or "classes." This feature provides help in organizing a module.

Every resource declaration at least contains a resource type, a title, and a set of attributes.

Syntax:

1. <TYPE> {'<TITLE>':
2. <ATTRIBUTE> => <VALUE>,}

	Puppet Manifest: In puppet, all the programs are written in Ruby programming language and added with an extension of .pp is 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. Puppet manifest has the following components: o **Files:** Files are the plain text files that can be directly deployed on your puppet clients. o **Resources:** Resources are the elements that we need to evaluate or change. Resources can be packages, files, etc. o **Nodes:** Block of code where all the information and definition related to the client are defined here. o **Classes:** Classes are used to group different types of resources.

Output	**1. Working with Resources**
	
	Master Side:
	
	• **File & package**
	
	root@fac5ae6c2631:/etc/puppet/code/environments/production/manifests# cat site.pp
	file{ '/etc/sss.txt':
	ensure ⇒ present,
	content ⇒ "Hello There",
	mode ⇒ '0644',
	}
	
	package{'apache2':
	{
	ensure ⇒ installed,
	}
	
	}

- **Exec**

```
root@fac5ae6c2631:/etc/puppet/code/environments/production/manifests# cat site.pp
node default {

file{ '/etc/sss.txt':
ensure ⇒ present,
content ⇒ "Hello There",
mode ⇒ '0644',
}

exec{ 'sss':
command ⇒ "/sbin/ifconfig > /etc/sss.txt "
}

}
```

Slave Side:

- **File & package**

```
root@a55f9f2059ed:/etc# service apache2 status
 * apache2 is not running
root@a55f9f2059ed:/etc# cat sss.txt
Hello Thereroot@a55f9f2059ed:/etc#
```

- **Exec**

```
root@a55f9f2059ed:/etc# cat sss.txt
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:04  txqueuelen 0  (Ethernet)
        RX packets 2496  bytes 9282211 (9.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1266  bytes 369328 (369.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. Working with Variables

Puppet provides many in-built variables that we can use in the manifest. As well as we can create our own variable to define in puppet manifest. Puppet provides different types of variables. Some frequently used variables are strings or an array of string.

Ex:

Master Side:

```
root@fac5ae6c2631:/etc/puppet/code/environments/production/manifests# cat site.pp
node default {

$tmp = "This is from master"

file{ '/etc/sss.txt':
ensure => present,
content => $tmp,
mode => '0644',
}

}
```

```
root@fac5ae6c2631:/etc/puppet/code/environments/production/manifests# cat site.pp
node default {

$tmp = "mysql-server"

package{ $tmp:
ensure => installed,
}

}
```

Slave Side:

```
root@a55f9f2059ed:/# cat /etc/sss.txt
This is from masterroot@a55f9f2059ed:/#
```

```
root@a55f9f2059ed:/etc# mysql -V
mysql  Ver 8.0.23-0ubuntu0.20.04.1 for Linux on x86_64 ((Ubuntu))
root@a55f9f2059ed:/etc#
```

3. Working with List

Master Side:

```
root@fac5ae6c2631:/etc/puppet/code/environments/production/manifests# cat site.pp
node default {

$tmp = "mysql-server"

package{ ['apache2','nginx']:
ensure => installed,
}

}
```

Slave Side:

```
root@a55f9f2059ed:/etc# service apache2 status
 * apache2 is not running
root@a55f9f2059ed:/etc# service nginx status
 * nginx is not running
root@a55f9f2059ed:/etc# mysql -V
mysql  Ver 8.0.23-0ubuntu0.20.04.1 for Linux on x86_64 ((Ubuntu))
```

4. **Working with only-if (Conditional Statement)**

Master Side:

- **If condition is TRUE**

```
root@fac5ae6c2631:/etc/puppet/code/environments/production/manifests# cat site.pp
node default {

exec{ 'sss':
command => '/sbin/ifconfig > /etc/sss.txt',
onlyif => '/bin/grep -c is /etc/sss.txt'
}

}
```

- **If condition is FALSE**

```
root@fac5ae6c2631:/etc/puppet/code/environments/production/manifests# cat site.pp
node default {

exec{ 'sss':
command => '/sbin/ifconfig > /etc/sss.txt',
onlyif => '/bin/grep -c hello /etc/sss.txt'
}

}
```

Slave Side:

- **If condition is TRUE**

```
root@a55f9f2059ed:/# cat /etc/sss.txt
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:04  txqueuelen 0  (Ethernet)
        RX packets 22696  bytes 77947238 (77.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7811  bytes 1221049 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

	• **If condition is FALSE**
	
	`root@a55f9f2059ed:/# cat /etc/sss.txt`
	`This is from masterroot@a55f9f2059ed:/#`

Conclusion	Thus, we successfully write our script in puppet with the help of loops and conditional statements.