

## Experiment 13

Roll No.	70
Name	MAYURI SHRIDATTA YERANDE
Class	D15-B
Subject	DevOps Lab
LO Mapped	<p>LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements</p> <p>LO2: To obtain complete knowledge of the “version control system” to effectively track changes augmented with Git and GitHub</p>

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

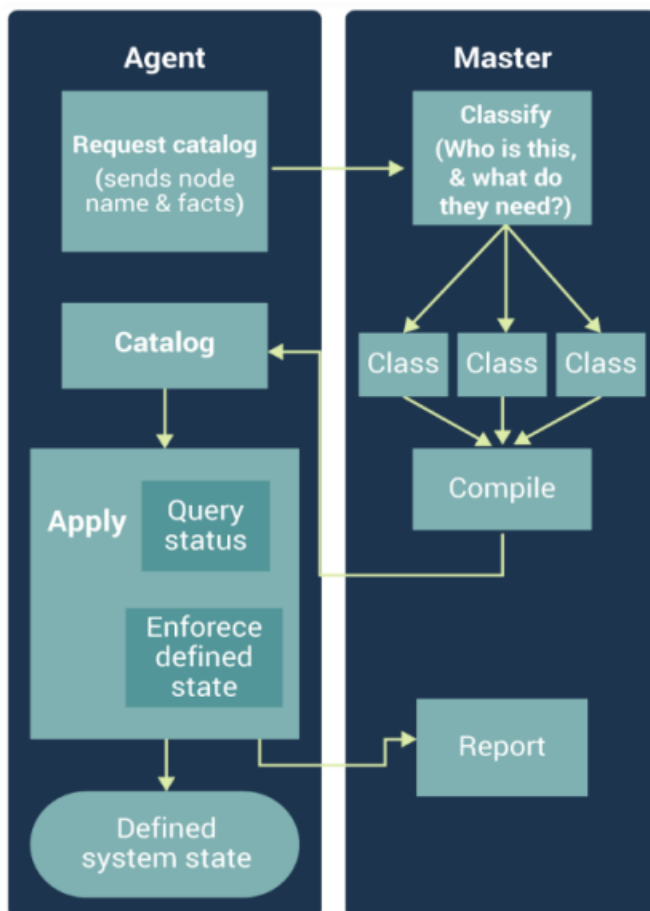
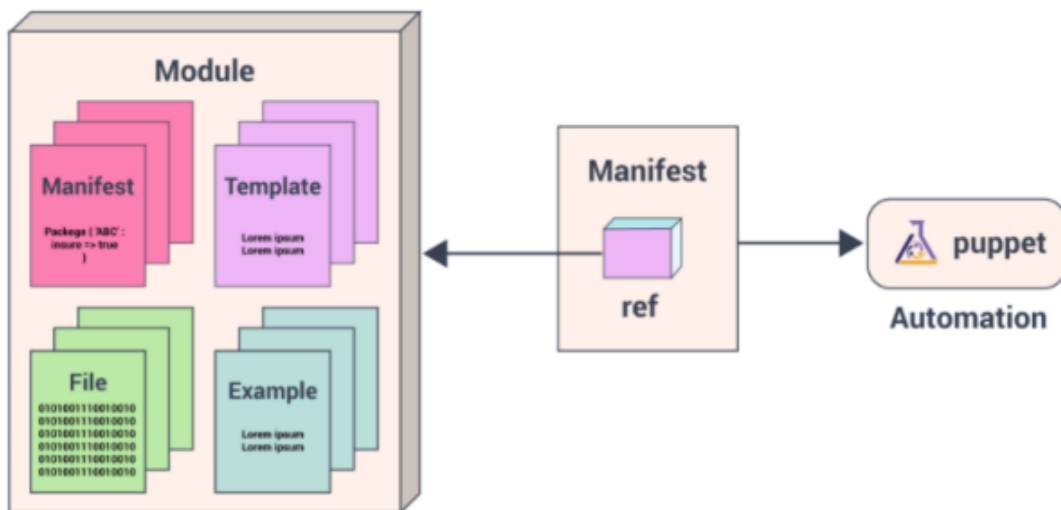
## **Theory:**

### **Introduction to Puppet Programming**

Puppet is one of the popularly used DevOps tools that is widely used for configuration management. It is used to bring about consistency in the Infrastructure. Puppet can define infrastructure as code, manage multiple servers, and enforce system configuration, thus helping in automating the process of infrastructure management. Puppet has its own configuration language, Puppet DSL. As with other DevOps programs, Puppet automates changes, eliminating manual script-driven changes. However, Puppet is not simply another shell language, nor is it a pure programming language, such as PHP. Instead, Puppet uses a declarative model-based approach to IT automation. This enables Puppet to define infrastructure as code and enforce system configuration with programs.

### **Key terms in Puppet Programming**

1. **Manifests:** A puppet program is called manifest and has a filename with .pp extension. Puppet's default main manifest is /etc/puppet/manifests/site.pp. (This defines global system configurations, such as LDAP configuration, DNS servers, or other configurations that apply to every node).
2. **Classes** Within these manifests are code blocks called classes other modules can call. Classes configure large or medium-sized chunks of functionality, such as all the packages, configuration files, and services needed to run an application. Classes make it easier to reuse Puppet code and improve readability.
3. **Resources** Puppet code is made up mostly of resource declarations. A resource describes a specific element about the system's desired state. For example, it can include that a specific file should exist or a package should be installed.
4. **Puppet Modules**
  - Except for the main site.pp manifest, it stores manifests in modules.
  - All of our Puppet code is organized in modules which are the basic building blocks of puppet that we can reuse and share. Each module manages a specific task in the infrastructure, such as installing and configuring a piece of software.
  - Modules contain Puppet classes, defined types, tasks, task plans, capacities, resource types, and plugins, for example, custom types or facts. Install modules in the Puppet module-path. Puppet loads all content from every module in the module-path, making this code available for use



## Implementation:

- On client/agent: mkdir code

```
ubuntu@ip-172-31-89-175:~$ mkdir code
ubuntu@ip-172-31-89-175:~$
```

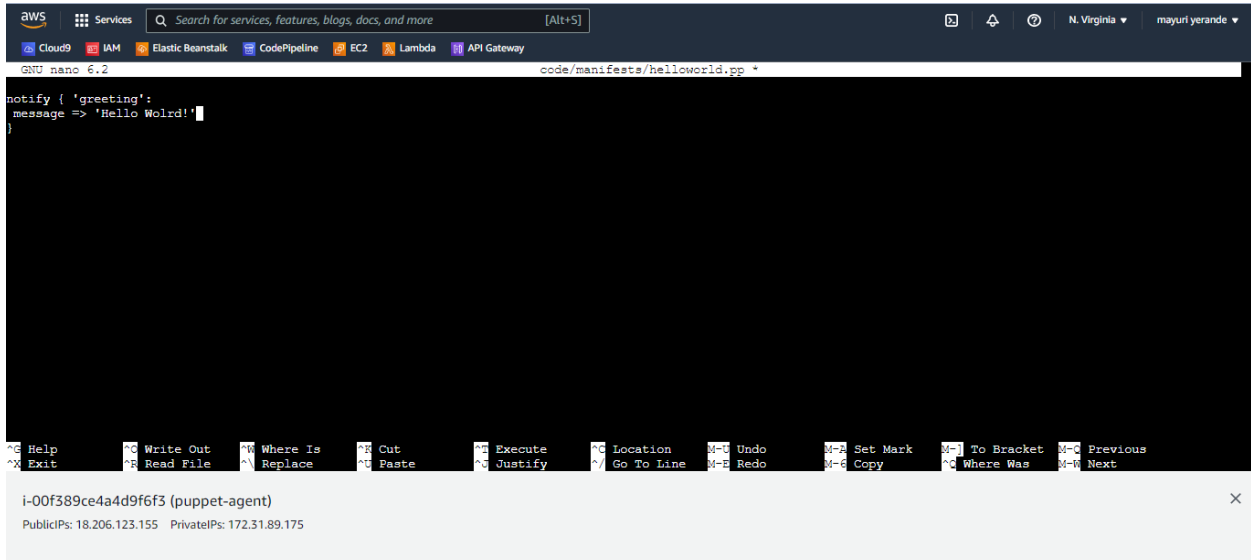
i-00f389ce4a4d9f6f3 (puppet-agent)  
PublicIPs: 18.206.123.155 PrivateIPs: 172.31.89.175

- On client/agent: mkdir code/manifests
- On client/agent: nano code/manifests/helloworld.pp

```
GNU nano 6.2 code/manifests/helloworld.pp
```

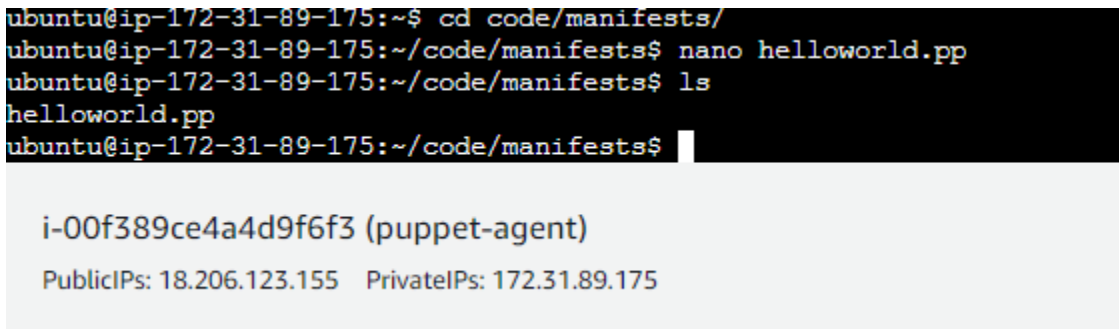
i-00f389ce4a4d9f6f3 (puppet-agent)  
PublicIPs: 18.206.123.155 PrivateIPs: 172.31.89.175

- Add this code



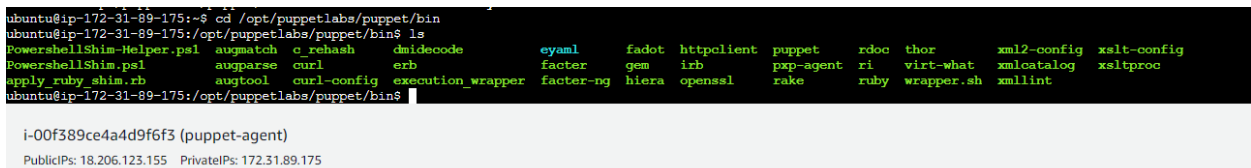
```
aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Cloud9 IAM Elastic Beanstalk CodePipeline EC2 Lambda API Gateway
GNU nano 6.2 code/manifests/helloworld.pp
notify { 'greeting':
  message => 'Hello Wolrd!'
}
Help Write Out Where Is Cut Execute Location Undo Set Mark To Bracket Previous
Exit Read File Replace Paste Justify Go To Line Redo Copy Where Was Next
i-00f389ce4a4d9f6f3 (puppet-agent)
PublicIPs: 18.206.123.155 PrivateIPs: 172.31.89.175
```

- Ctrl s to save, ctrl x to exit
- Enter the following commands



```
ubuntu@ip-172-31-89-175:~$ cd code/manifests/
ubuntu@ip-172-31-89-175:~/code/manifests$ nano helloworld.pp
ubuntu@ip-172-31-89-175:~/code/manifests$ ls
helloworld.pp
ubuntu@ip-172-31-89-175:~/code/manifests$
i-00f389ce4a4d9f6f3 (puppet-agent)
PublicIPs: 18.206.123.155 PrivateIPs: 172.31.89.175
```

- On client/agent: cd /opt/puppetlabs/puppet/bin
- On client/agent: ls



```
ubuntu@ip-172-31-89-175:~$ cd /opt/puppetlabs/puppet/bin
ubuntu@ip-172-31-89-175:/opt/puppetlabs/puppet/bin$ ls
PowershellShim-Helper.ps1 augmatch c_rehash dmidecode eyaml fadot httpclient puppet rdoc thor xml2-config xslt-config
PowershellShim.ps1 augparse curl erb factor facter-gem irb pcp-agent ri virt-what xmllcatalog xsltproc
apply_ruby_shim.rb augtool curl-config execution_wrapper factor-ng hiera openssl rake ruby wrapper.sh xmllint
ubuntu@ip-172-31-89-175:/opt/puppetlabs/puppet/bin$
i-00f389ce4a4d9f6f3 (puppet-agent)
PublicIPs: 18.206.123.155 PrivateIPs: 172.31.89.175
```

- On client/agent: `./puppet apply /home/ubuntu/code/manifests/helloworld.pp`

```
ubuntu@ip-172-31-89-175:/opt/puppetlabs/puppet/bin$ ./puppet apply /home/ubuntu/code/manifests/helloworld.pp
Notice: Compiled catalog for ip-172-31-89-175.ec2.internal in environment production in 0.01 seconds
Notice: Hello World!
Notice: /Stage[main]/Main/Notify[greeting]/message: defined 'message' as 'Hello World!'
Notice: Applied catalog in 0.01 seconds
ubuntu@ip-172-31-89-175:/opt/puppetlabs/puppet/bin$
```

```
i-00f389ce4a4d9f6f3 (puppet-agent)
PublicIPs: 18.206.123.155 PrivateIPs: 172.31.89.175
```

- Thus we successfully executed a puppet program using puppet blocks.
- We can see the “hello world” output.

**Conclusion:** We have learnt Software Configuration Management and provisioning using Puppet Blocks(Manifest, Modules, Classes, Function)