**Experiment 13 - Puppet**

| Roll No. | 37 |
| --- | --- |
| Name | Mikil Lalwani |
| Class | D15-B |
| Subject | DevOps Lab |
| LO Mapped | 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.  LO6: To Synthesize software configuration and provisioning using Ansible/Puppet. |
|

**Aim**:

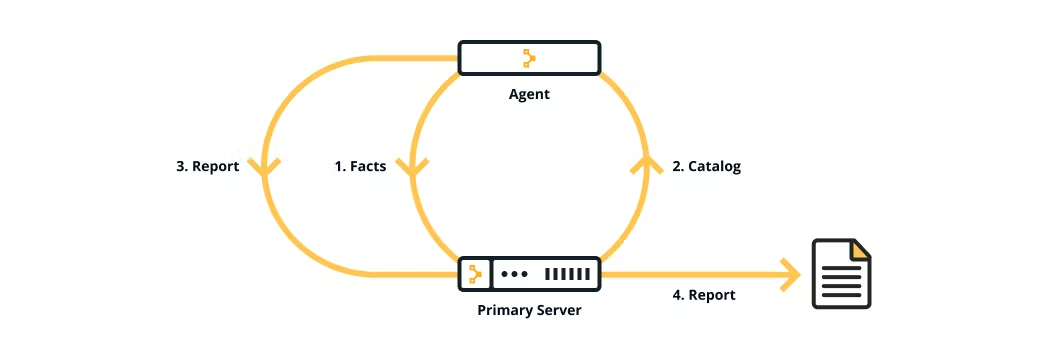
To learn Software Configuration Management and provisioning using Puppet Blocks(Manifest, Modules, Classes, Function)

**Introduction**:

What is Puppet?

Puppet is an open-source Software Configuration Management and Deployment tool. It's mostly used in Linux Cloud environments to automate infrastructure, management of different servers, etc.

Puppet Architecture



Puppet uses a Master-Agent architecture in which the Master and Slave communicate through a secure encrypted channel with the help of SSL.

Puppet Master

Puppet Master is the Hub or the main node in the network of systems that handles all the configuration across systems. It is capable of applying different configurations to other nodes called Puppet Agents.

Puppet Agent

Puppet Agents are working machines in which different servers are set up for different use cases. These servers and their configurations are managed by the Puppet Master.

Use Cases

Server and Database Management

Multiple Servers and Databases can be controlled in one place using Puppet.

System Configuration

A key feature of Puppet is that multiple systems can be configured at once from the Puppet Master device.

Networking

Due to Puppet's architecture, it can be used for secure networking and data transfer across systems.

Container Management

Puppet makes it easy to integrate containers with existing IT infrastructure.

PUPPET MANIFESTS

Puppet programs are called manifests. Manifests are composed of Puppet code and their

filenames use the .pp extension. The default main manifest in Puppet installed via apt is

/etc/puppet/manifests/site.pp.

Classes

In Puppet, classes are code blocks that can be called in a code elsewhere. Using classes allows

you to reuse Puppet code, and can make reading manifests easier.

Class Definition

A class definition is where the code that composes a class lives. Defining a class makes the class

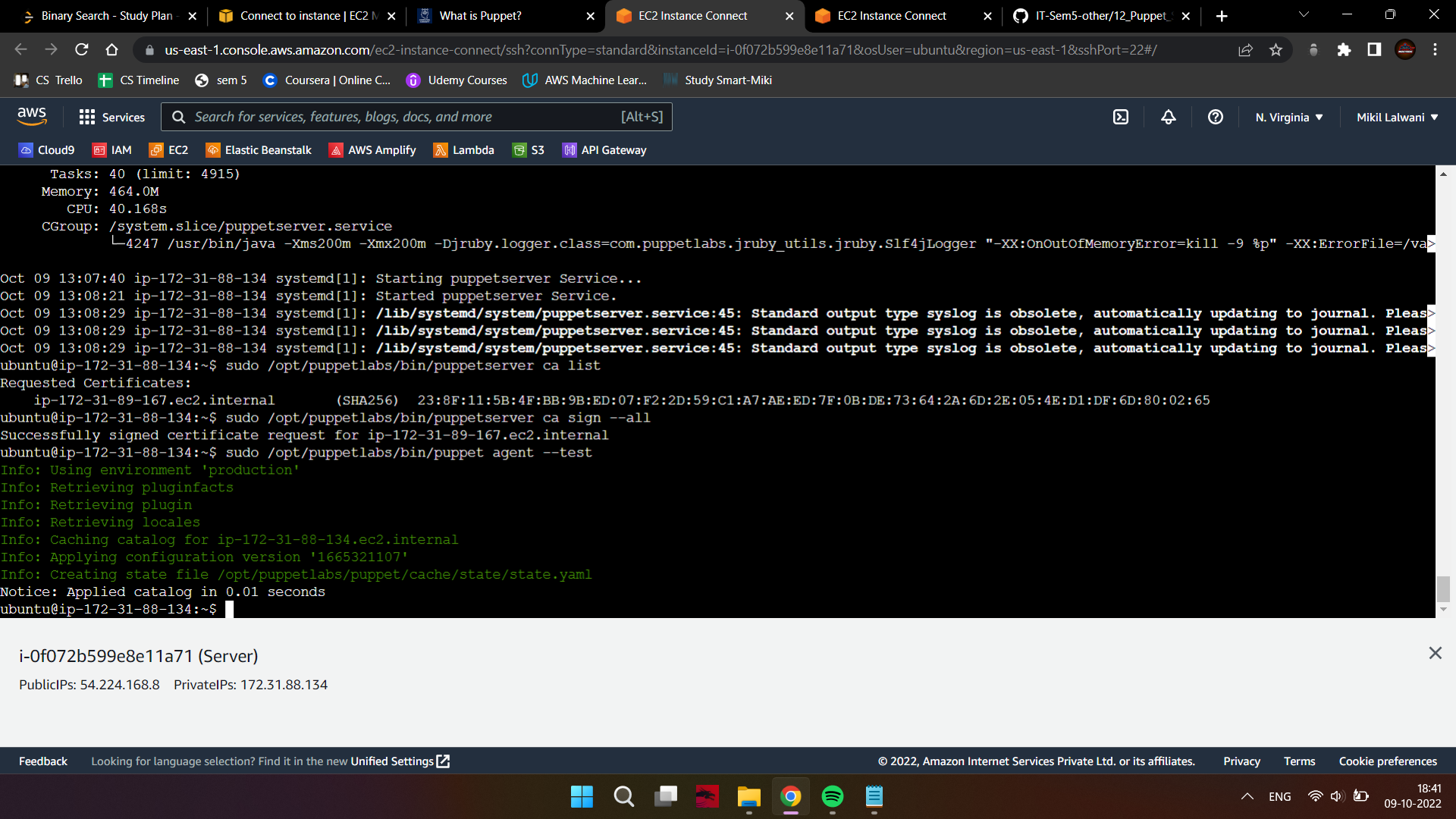
available to be used in manifests, but does not actually evaluate anything.

The above defines a class named “example\_class”, and the Puppet code would go between the

curly braces.

1. To test your cluster setup, run this command -

sudo /opt/puppetlabs/bin/puppet agent --test



If the output is normal, you can proceed.

1. Create new directories

mkdir puppet\_code

mkdir puppet\_code/manifests

1. Create a new puppet manifest file and open it using nano.

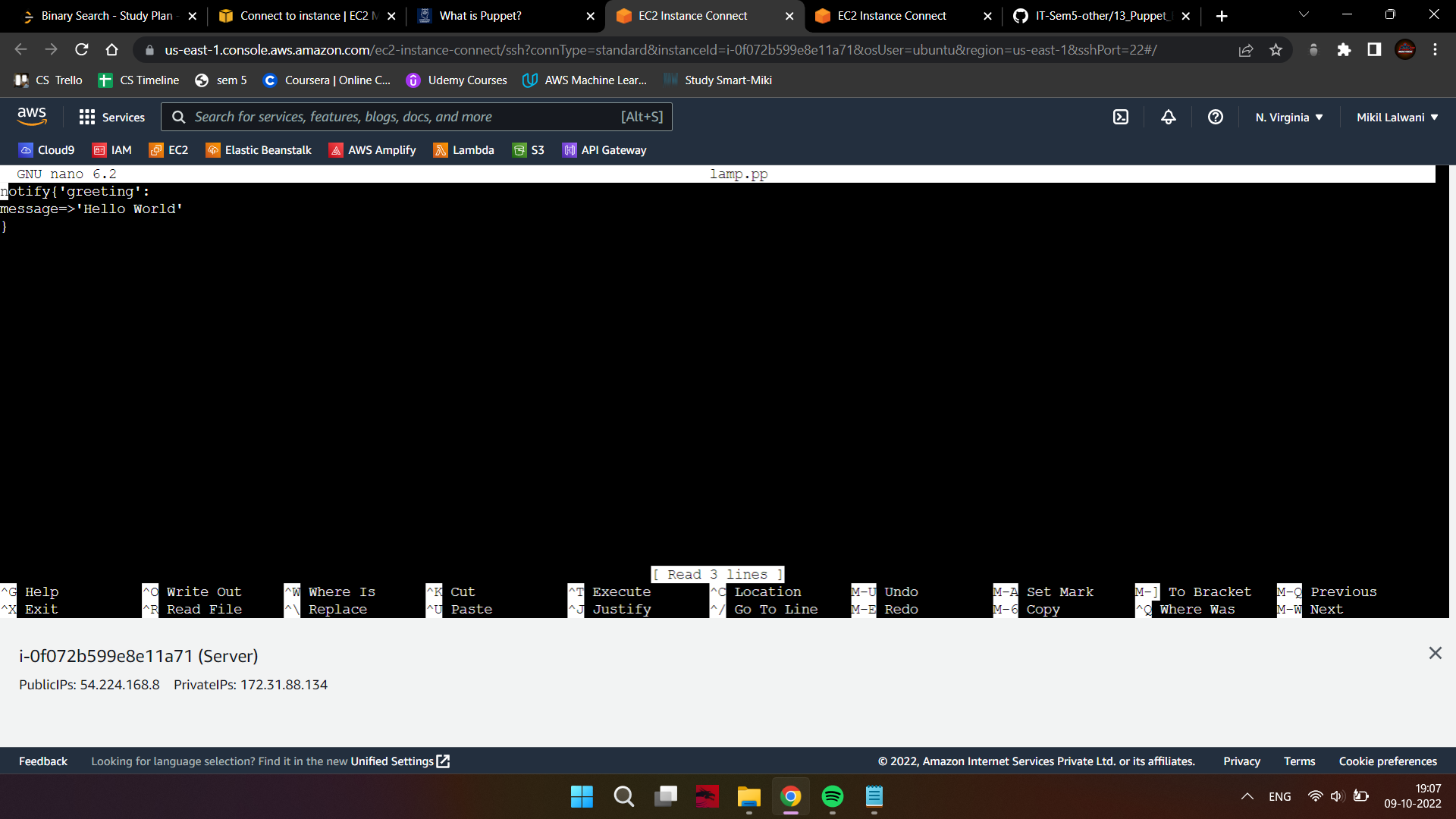
nano /puppet\_code/manifests/helloworld.pp

1. Enter the following contents inside the file.

notify { 'greeting':

message => 'Hello, world!'

}

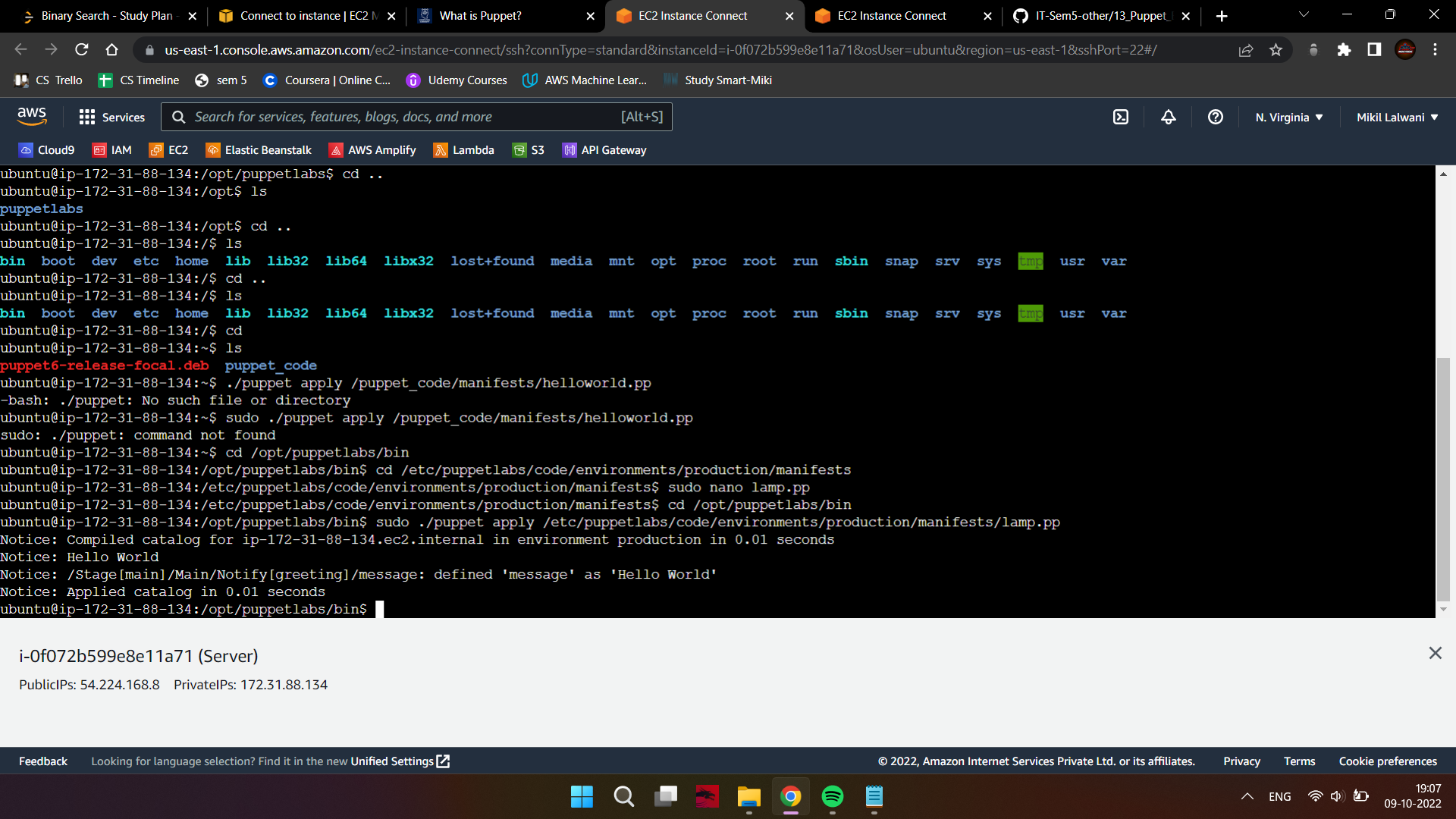


1. Change directory to /opt/puppetlabs/puppet/bin

cd /opt/puppetlabs/puppet/bin

1. Now, apply the file.

./puppet apply /home/ubuntu/manifests/helloworld.pp



**Conclusion:**

Thus, we successfully learned about the syntax of manifest files and created our first puppet script to print a ‘hello world’ message.