**Name:** WASIMA QAYYUMUDDIN SHAIKH

**RollNo:**6220071

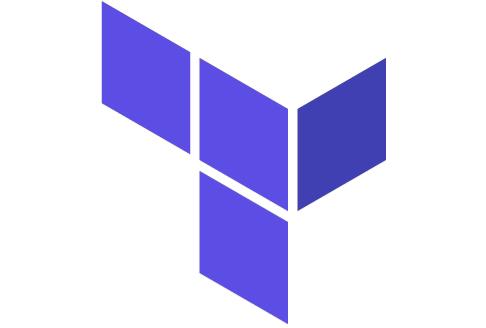
**Class:** T.E.I.T

**Sem:** V

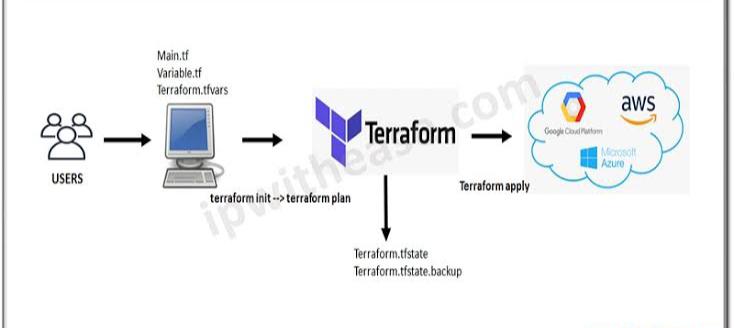
**Subject:** ADVACE DEVOPS LAB **(Addevops**)

**EXPERIMENT NO: 15**

**Q1. What is Terraform?**

Terraform is an open source infrastructure as code software tool that allows DevOps engineers to programmatically provision the physical resources an application requires to run.Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can help with multi-cloud by having one workflow for all clouds. The infrastructure Terraform manages can be hosted on public clouds like Amazon Web Services, Microsoft Azure, and Google Cloud Platform, or on-perm in private clouds such as VMWare vSphere, OpenStack, or CloudStack. Terraform treats infrastructure as code (IaC) so you never have to worry about you infrastructure drifting away from its desired configuration.

Terraform allows users to define their entire infrastructure simply by using configuration files and version control. When a command is given to deploy and run a server, database or load balancer, Terraform parses the code and translates it into an application programming interface (API) call to the resource provider. Because Terraform is open source, developers are always able to extend the tool's usefulness by writing new plugins or compiling different versions of existing plugins.



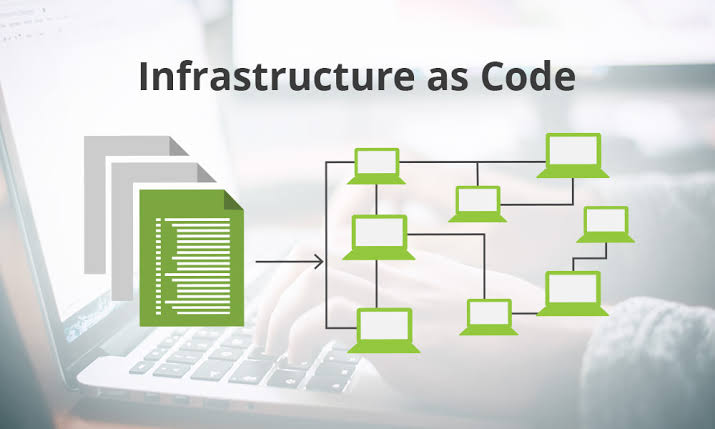
Terraform has two important components: Terraform Core and Terraform Plugins.

Terraform Core oversees the reading and interpolation of resource plan executions, resource graphs, [state management](https://searchapparchitecture.techtarget.com/definition/state-management) features and configuration files. Core is composed of compiled binaries written in the [Go](https://searchitoperations.techtarget.com/definition/Go-programming-language) programming language. Each compiled binary acts as a command-line interface (CLI) for communicating with plugins through remote procedure calls (RPC).

Terraform Plugins are responsible for defining resources for specific services. This includes authenticating infrastructure providers and initializing the libraries used to make API calls. Terraform Plugins are written in Go as executable binaries that can either be used as a specific service or as a [provisioner](https://whatis.techtarget.com/definition/provisioning). (Provisioner plugins are used to execute commands for a designated resource.)

**Q2. What is Infrastructure as a Code (IaC) ?**

Infrastructure as code is an IT practice that manages an application's underlying IT infrastructure through programming. This approach to resource allocation allows developers to logically manage, monitor and provision resources -- as opposed to requiring that an operations team manually configure each required resource.



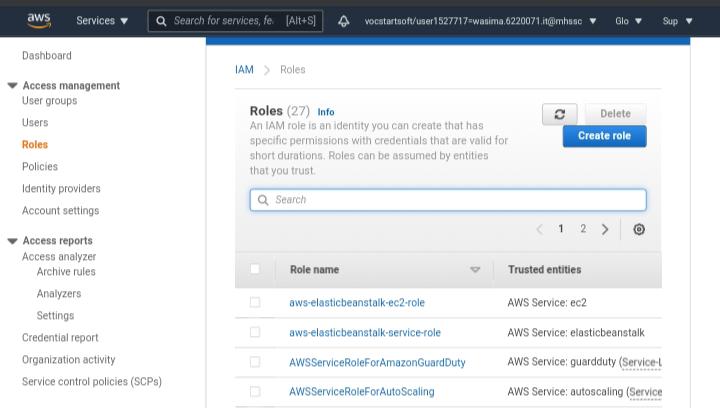
Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.

With IaC, configuration files are created that contain your infrastructure specifications, which makes it easier to edit and distribute configurations. It also ensures that you provision the same environment every time.

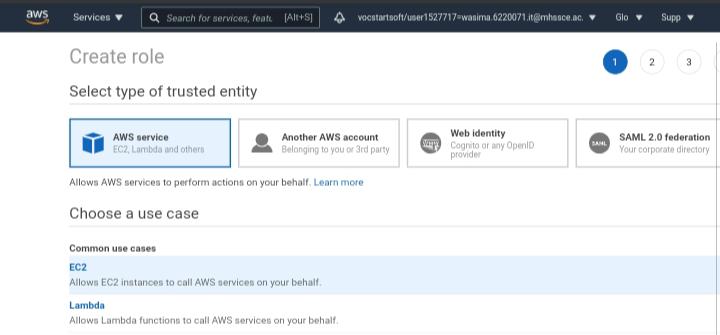
**Q3. Perform an experiment, to understand Terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.**

**Step - 1 Create an IAM role to provision EC2 instance in AWS**

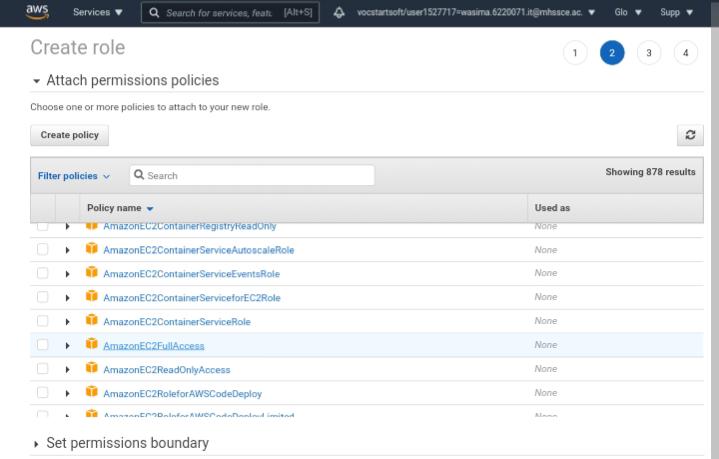
Go to AWS console, click on IAM



Select AWS service, EC2, Click on Next Permissions

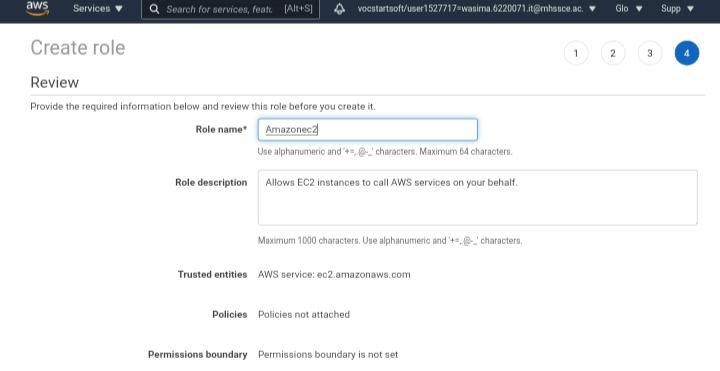


Type EC2 and choose AmazonEC2FullAccess as policy



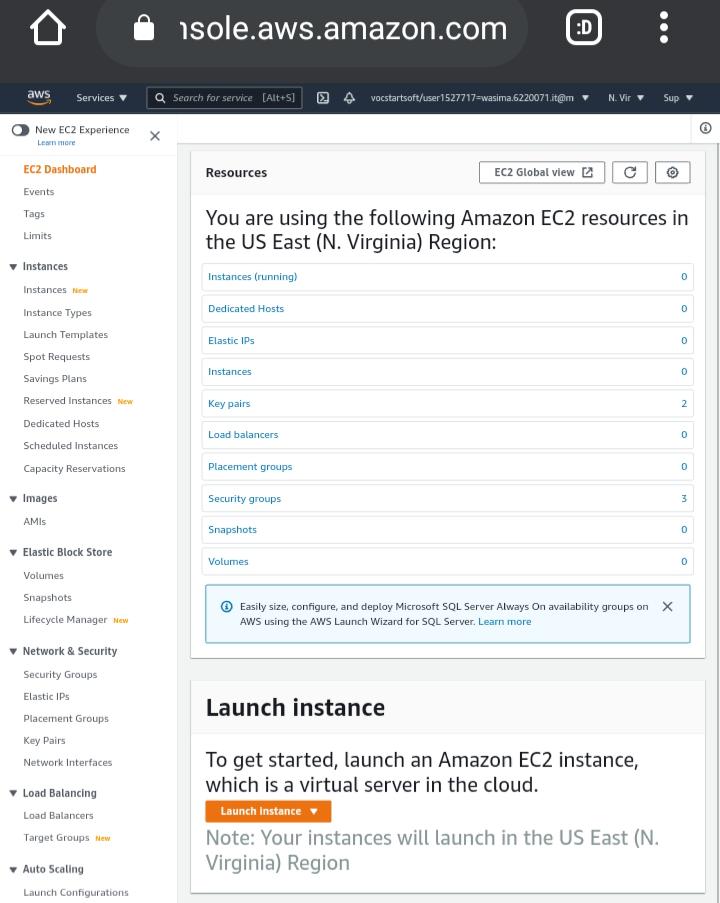
Click on Next tags, Next Review

give some role name and click on Create role.

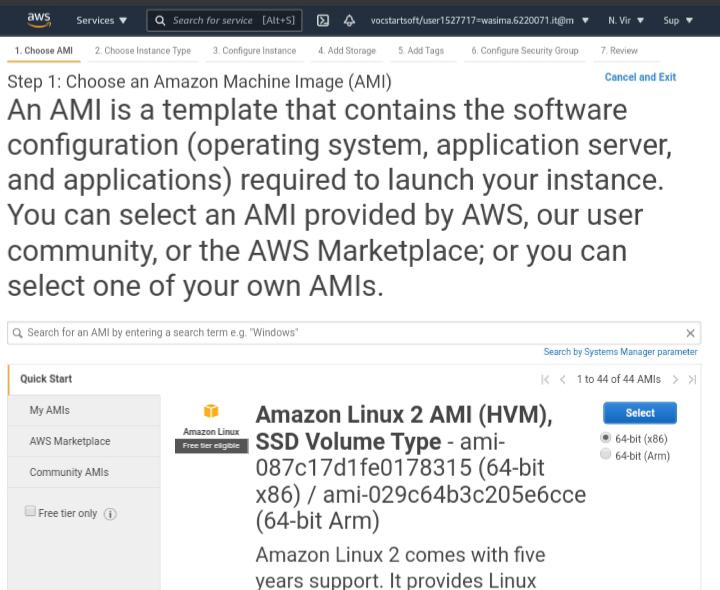


**Step-2**

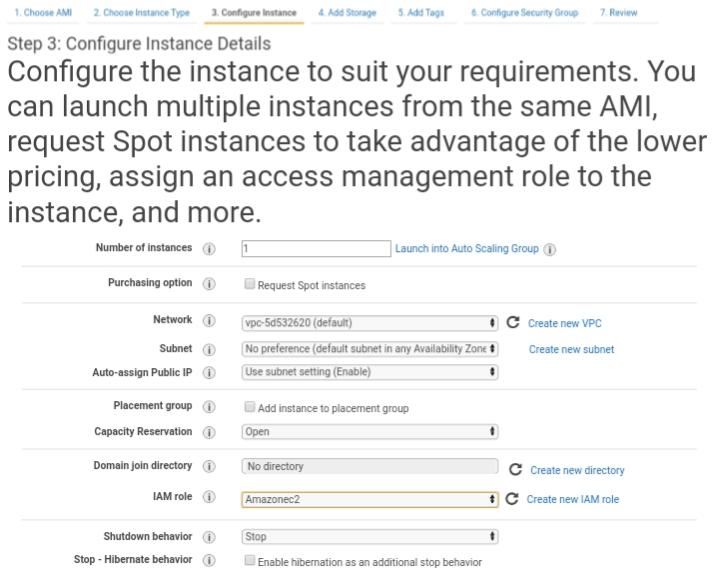
1. After successfully creating role we have to create EC2 instance Service 🡪 EC2 🡪 Launch Instance



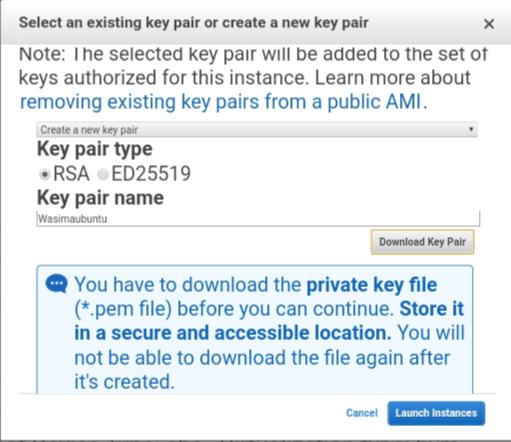
1. Then choose AMI here we are choosing Amazon Linux 2



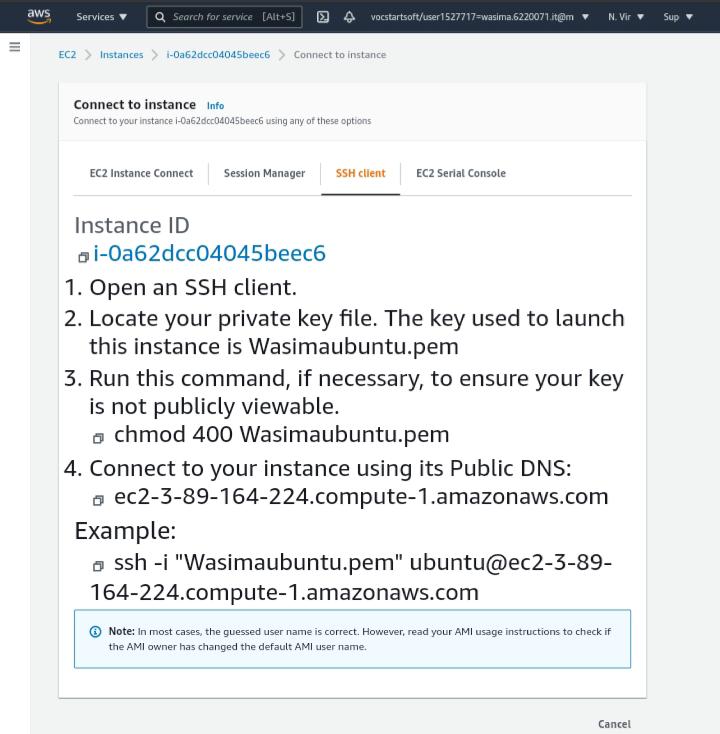
1. Then in configure instance detail 🡪 IAM role section choose the IAM role which we have already created by above step.

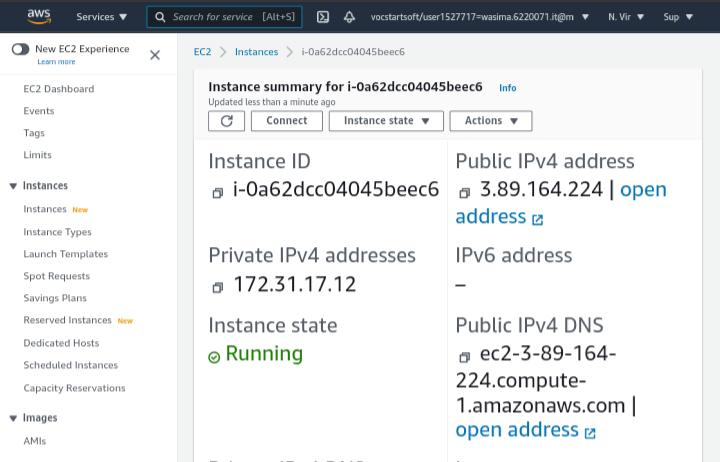


1. Then review and launch by downloading key-value pair

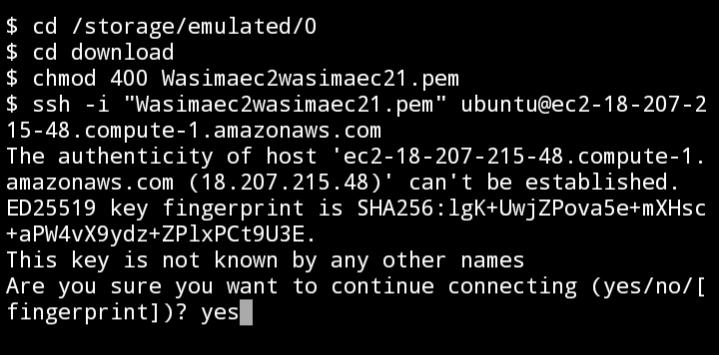
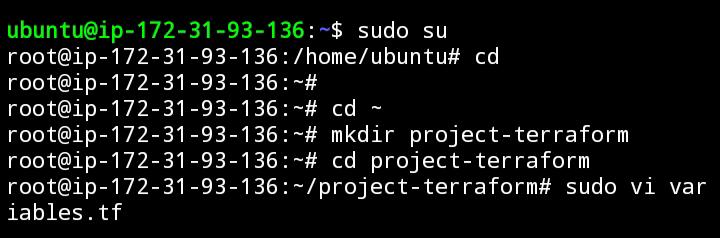


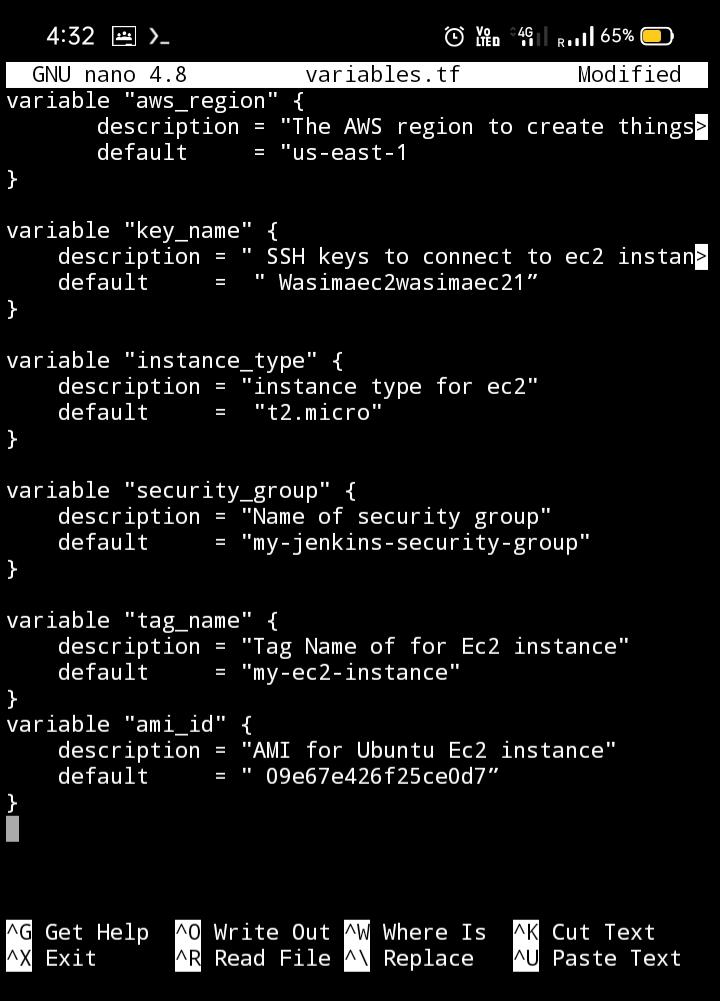
1. After launching the instance, we need to connect to our instance by clicking on connect. Then we need to select SSH client.



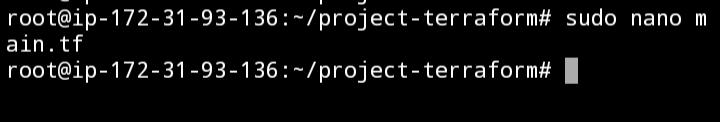


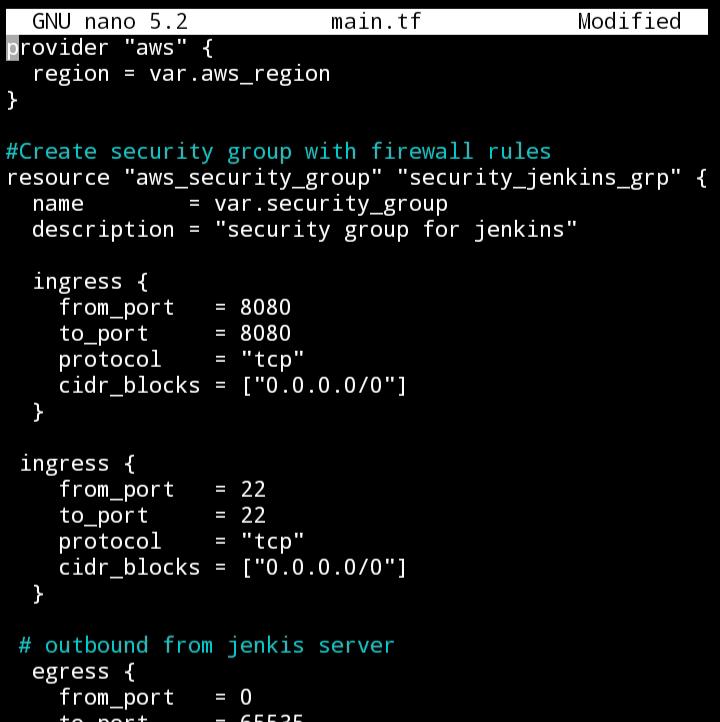
**Q4. Using Terraform , create an EC2 instance on AWS cloud**

1. After that open Termux qpp -> locate private key file by using cd command 🡪 Execute chmod 400 wasima12.pem for publically viewable to connect instance execute command ssh -I “wasimaec2wasimaec21.pem” and it’s Public DNS.
2. Then type ‘sudo su’ for entering root user.Then create one directory named as project terraform🡪 change directory to project-terraform in this direactory create one terraform file by using nano editor named as variables.tf.

****

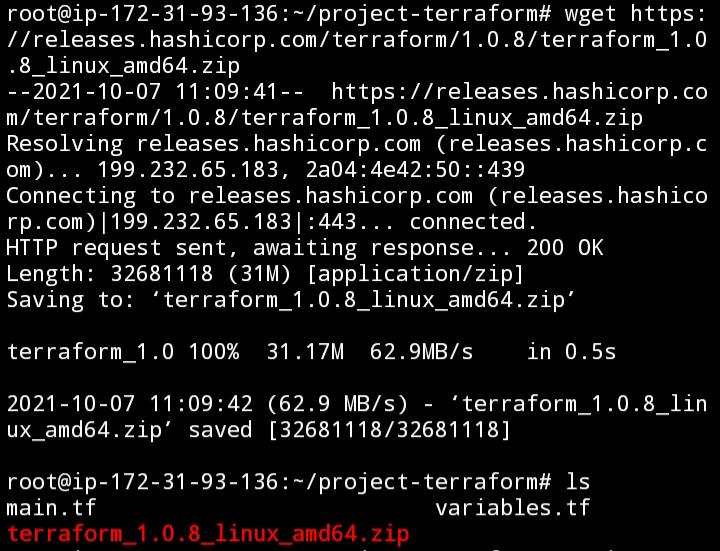
1. Create one more terraform file named as main.tf

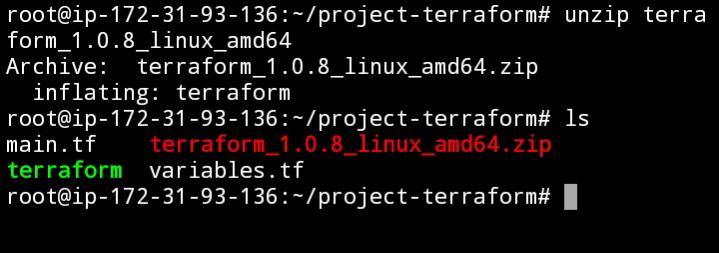




1. **Execute Terraform Commands**

Install Terraform [wget and unzip]





1. Execute following commands

cp Terraform /bin/

check Terraform version by typing Terraform --version

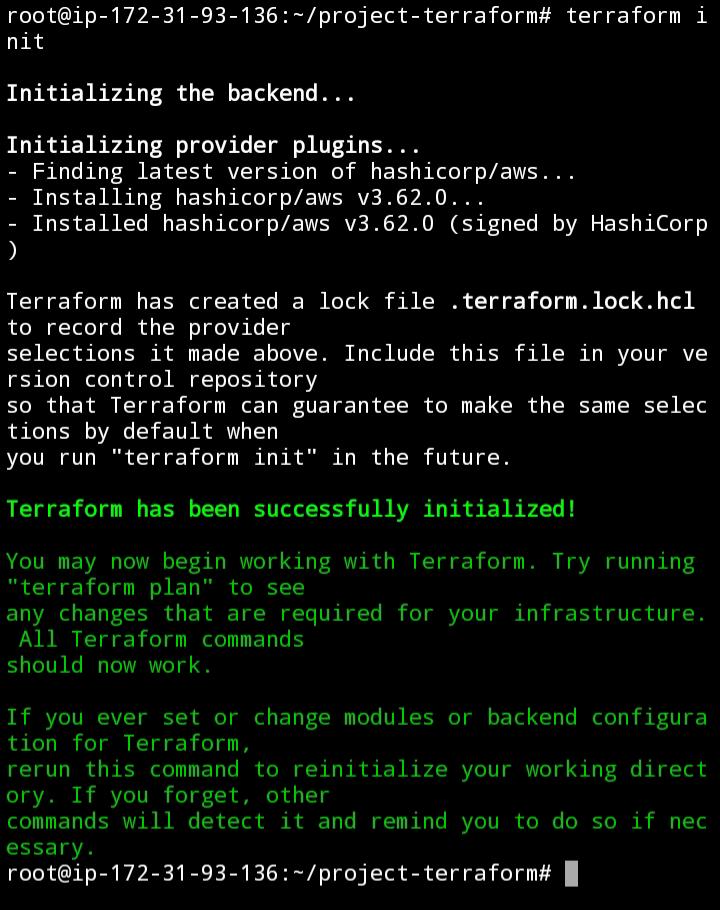
cd project-terraform



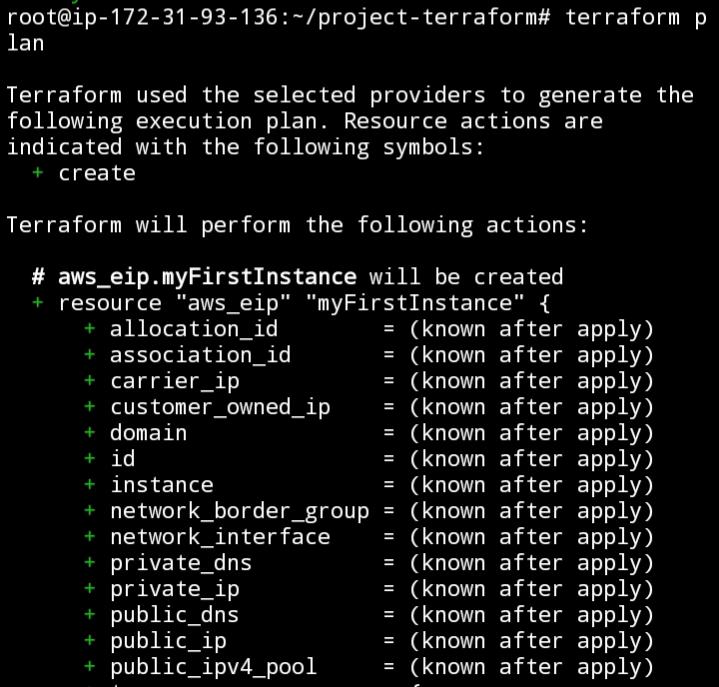
1. **Execute Terraform Commands**

Now execute the below command: to initialize the tarraform

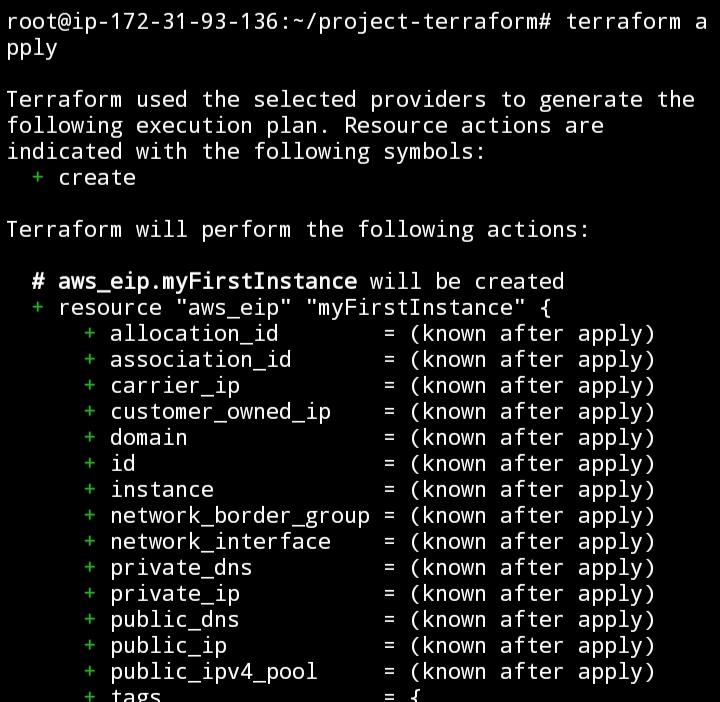
terraform init



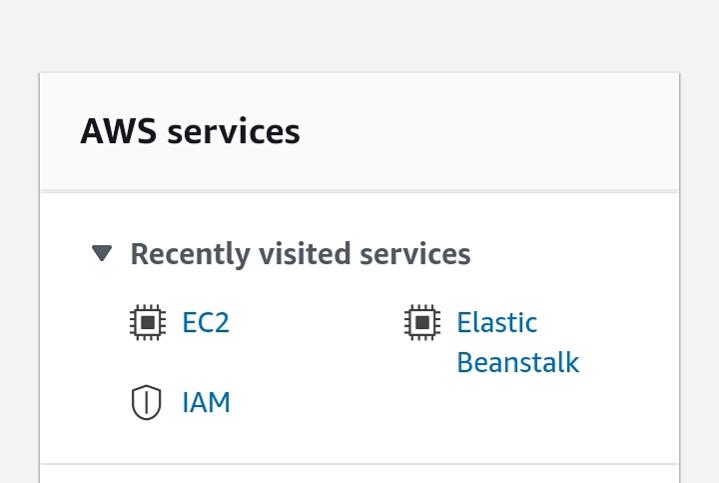
Execute the below command

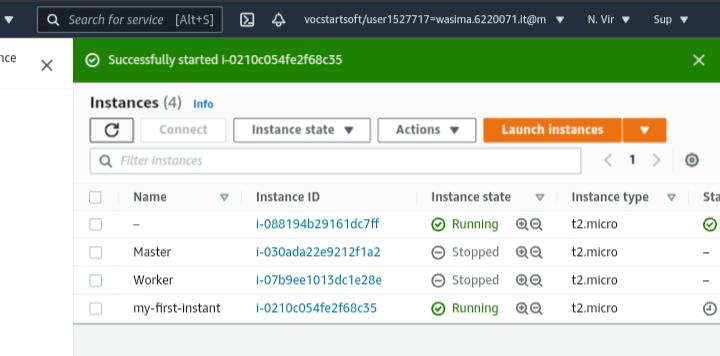
terraform plan

Execute the below command to apply.

terraform apply

1. Now login to EC2 console, to see the new instances up and running





**Q5. Explain following Terraform commands in one line**

* terraform init:

The terraform init command is used to initialize a working directory containing Terraform configuration files.

* terraform validate:

The terraform validate command validates the configuration files in a directory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs, etc.

* terraform plan:

The terraform plan command creates an execution plan.

* terraform apply:

The terraform apply command executes the actions proposed in a Terraform plan.

* terraform destroy:

The terraform destroy command is a convenient way to destroy all remote objects managed by a particular Terraform configuration.

**Note: Terminate the resources after performing the practical- terminate Ec2**