EXPERIMENT-13

NAME: SHAIKH MUBASHIRA TUFEL AHMED
ROLL NO: 612055 **COURSE:** ADVANCE DEVOPS(ITL504)
BRANCH: T.E. INFORMATION TECHNOLOGY (SEM 5)

1. What is Terraform?

➔HashiCorp Terraform is an infrastructure as a code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructures throughout their lifecycle.

Terraform can manage low-level components like computing, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

Terraform creates and manages resources on cloud platforms and other services through its application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.

2. What is Infrastructure as a Code (IaC)?

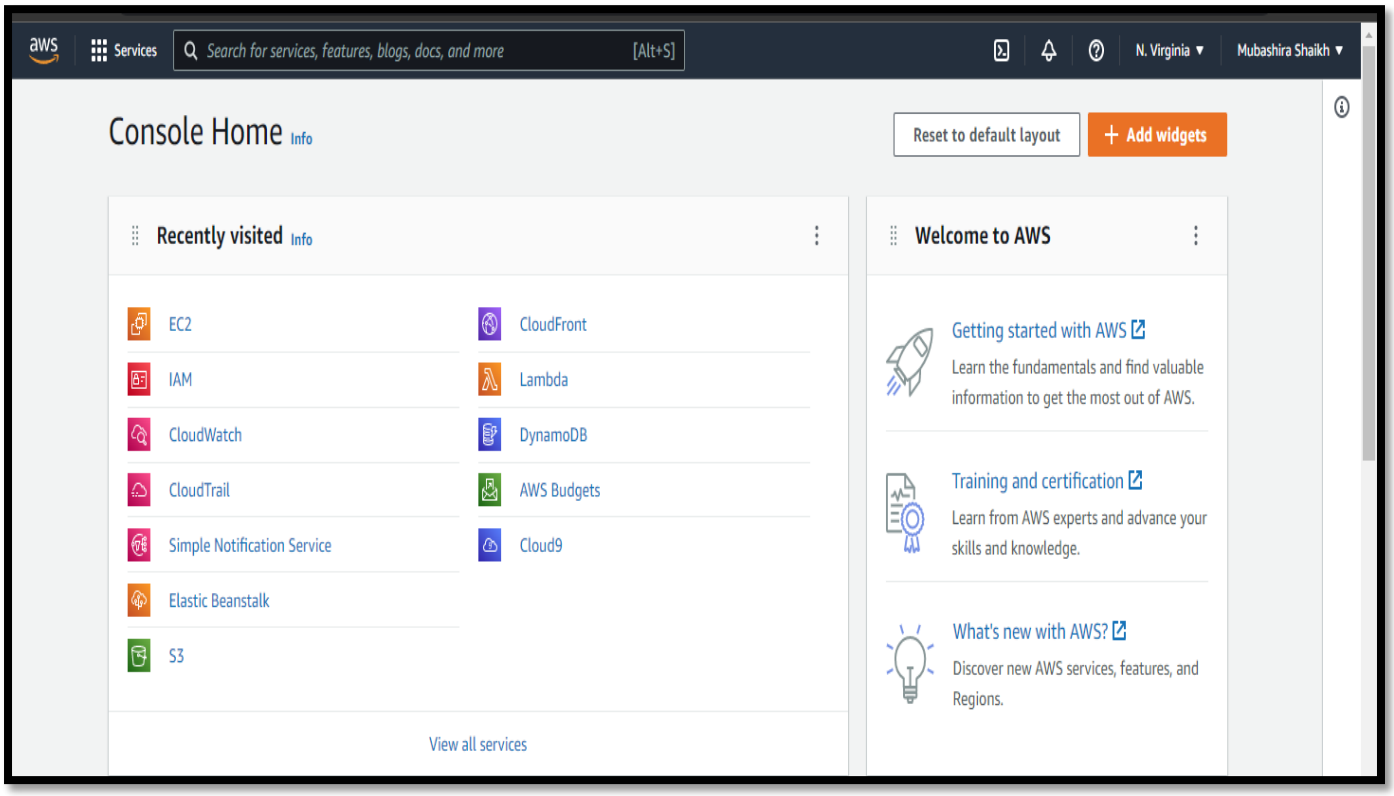
➔Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.

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 provide the same environment every time.

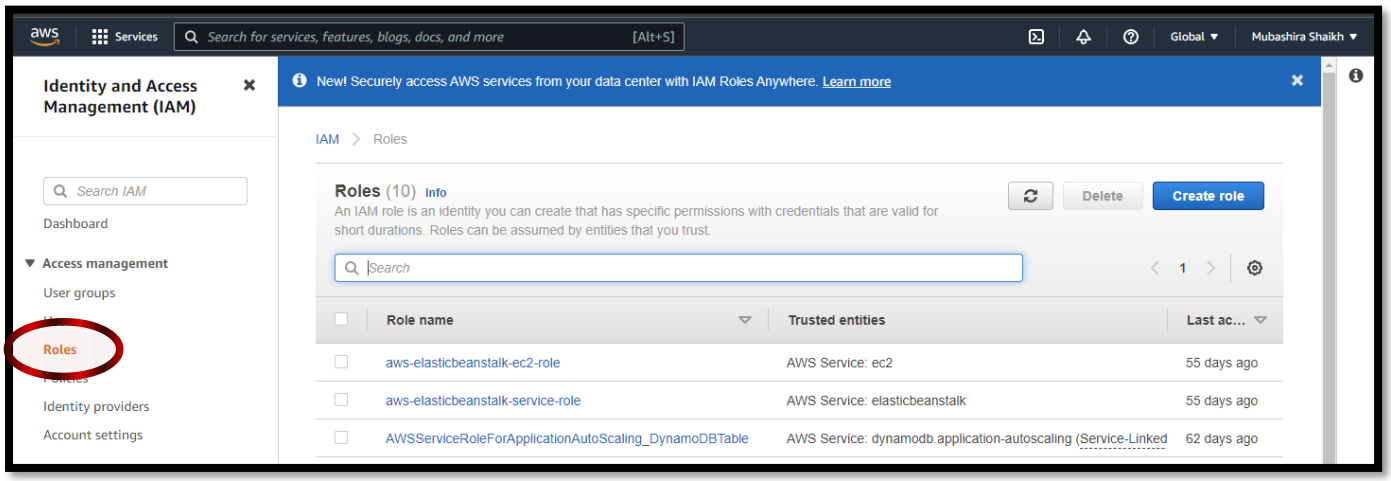
Version control is an important part of IaC, and your configuration files should be under source control just like any other software source code file. Deploying your infrastructure as code also means that you can divide your infrastructure into modular components that can then be combined in different ways through automation.

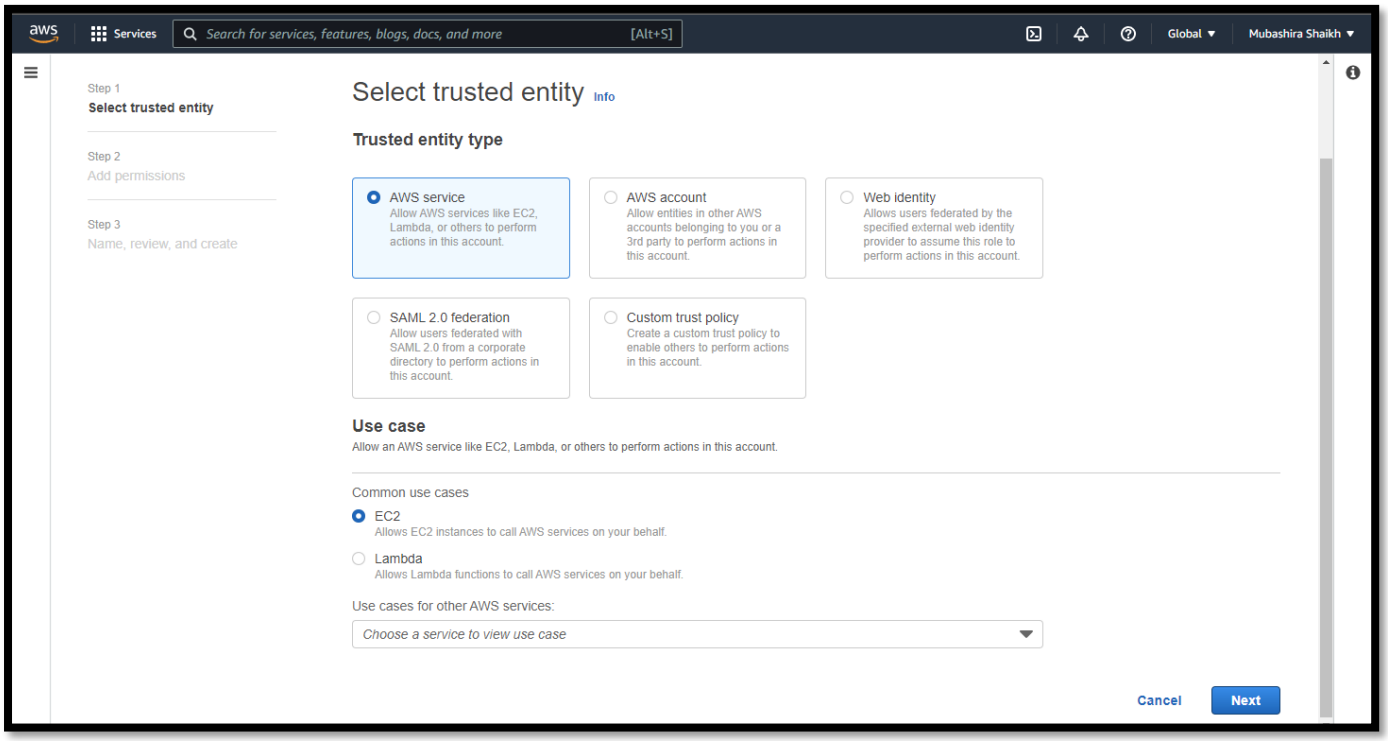
3. Perform an experiment, to understand Terraform's lifecycle, and core concepts/terminologies, and install it on a Linux Machine.

Step 1: AWS Management Console

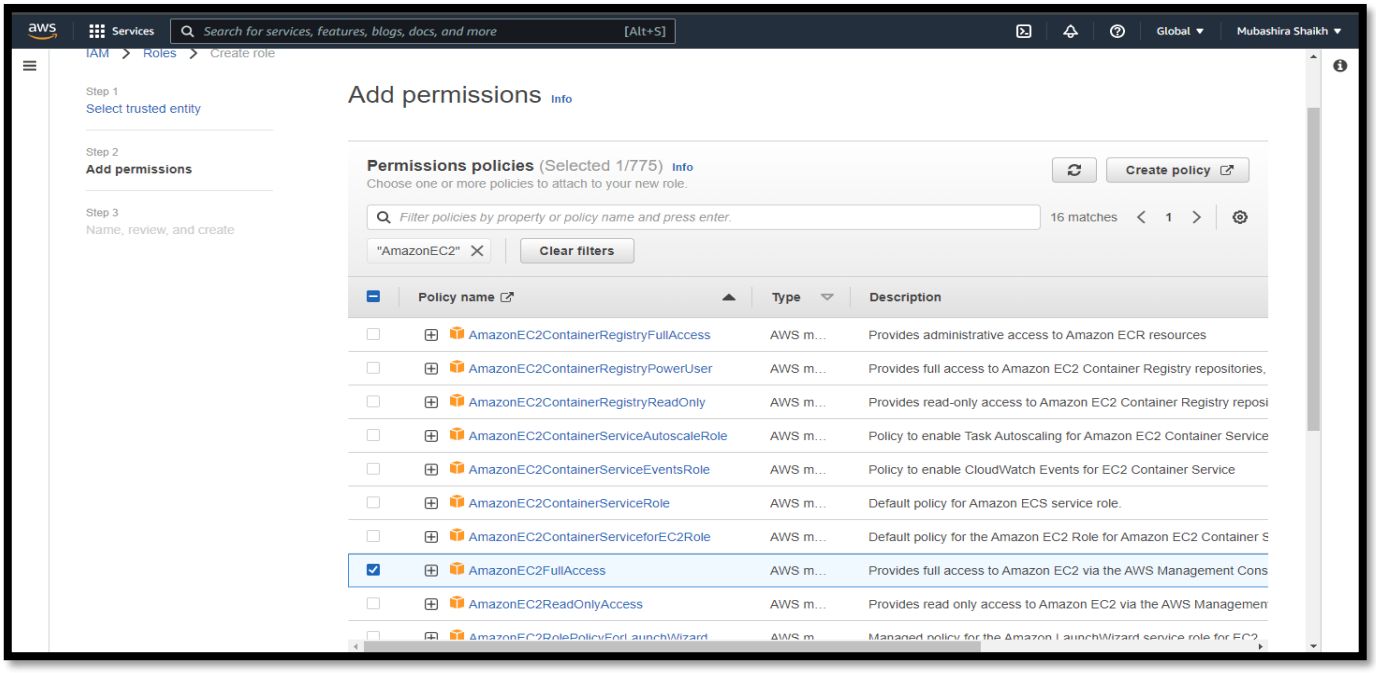


Step 2: Search for IAM → Select Roles → Create role → Set Trusted entity type to AWS service, select the use case to EC2.

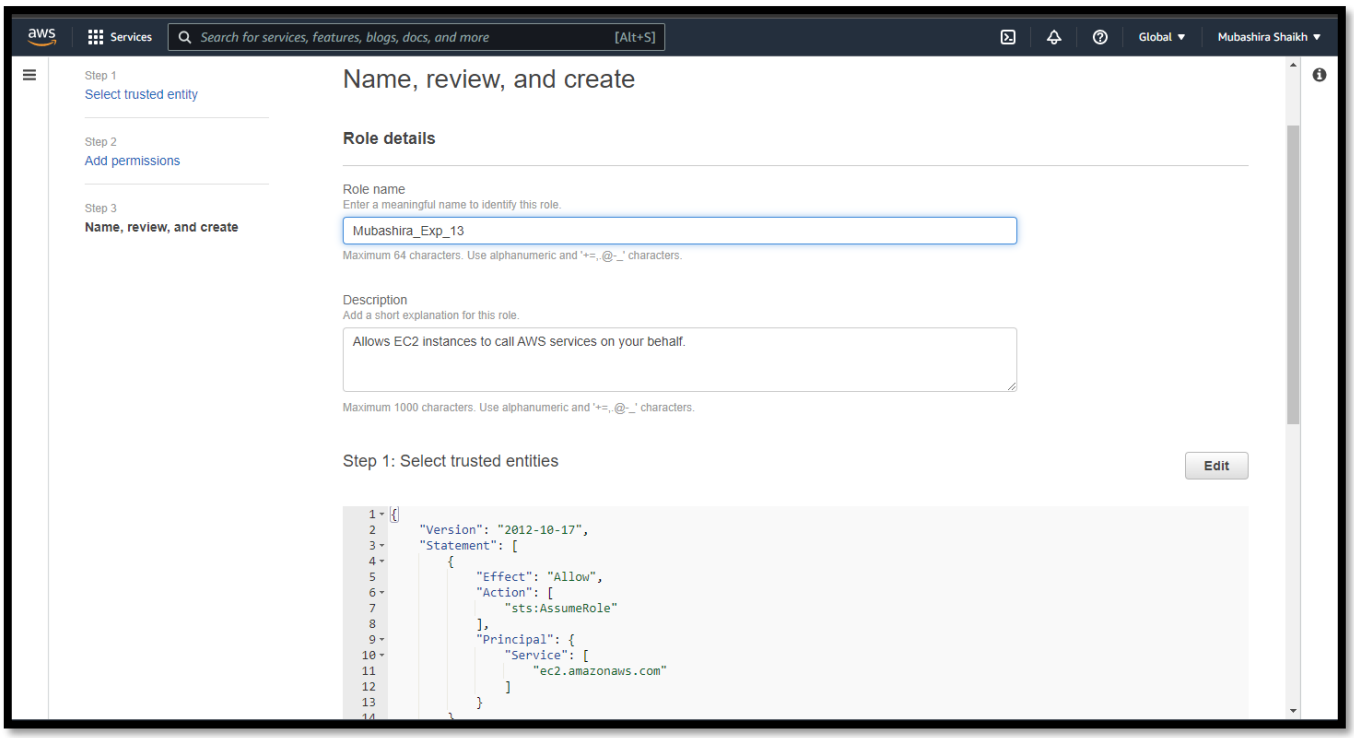




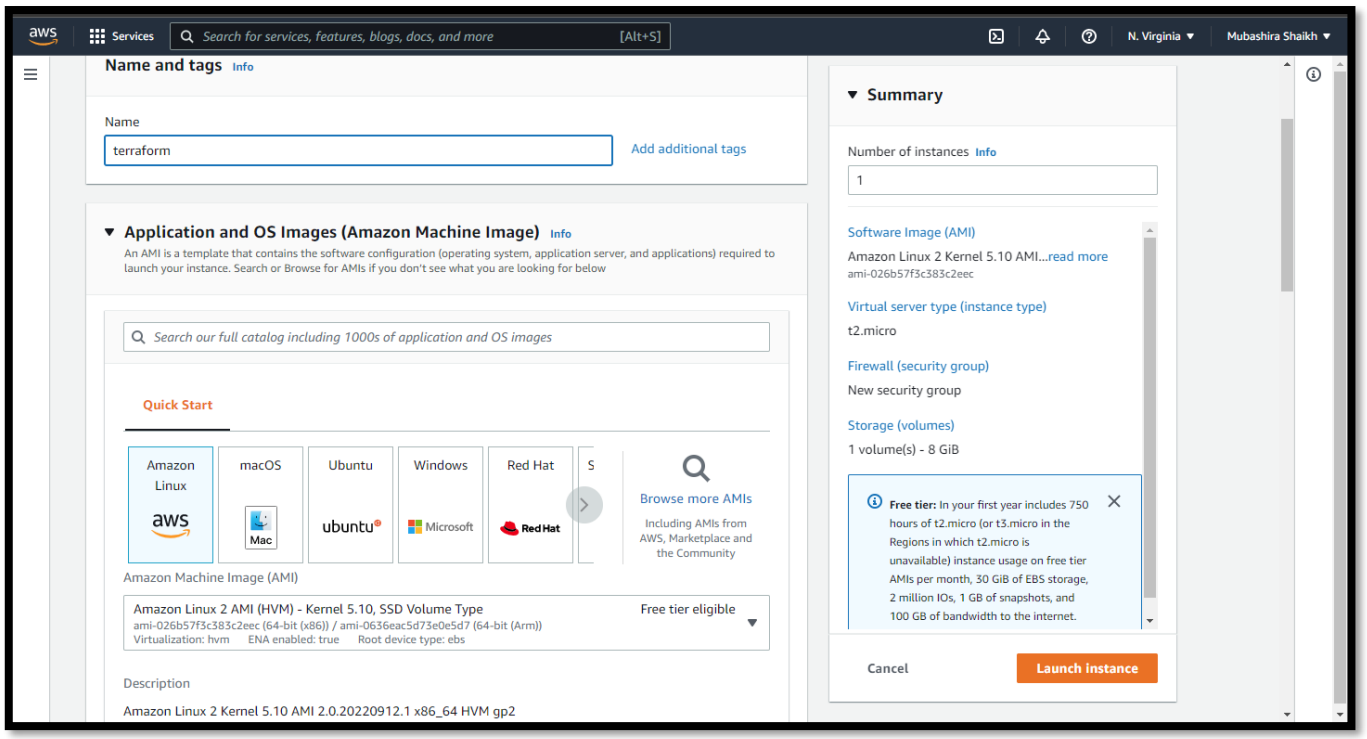
Step 3: Search for AmazonEC2FullAccess in Permission policies → Select the policy → Next



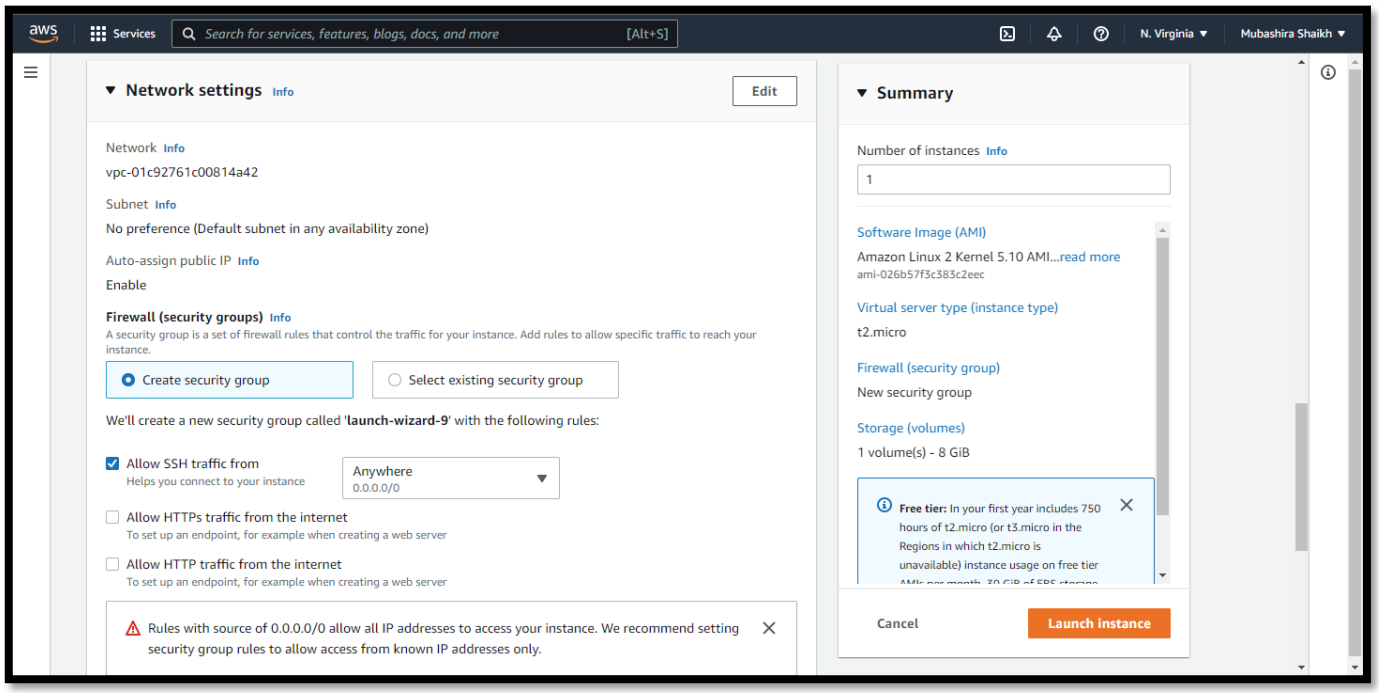
Step 4: Assign a name to your role → Create a role

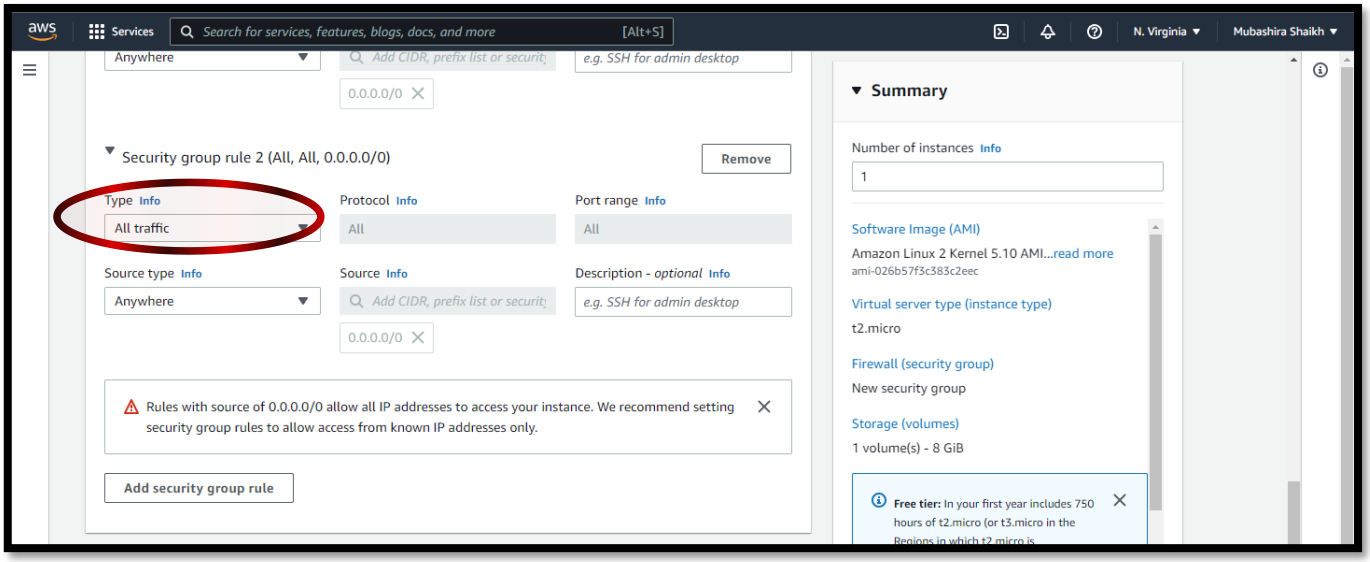


Step 5: Search for EC2 → Select 'Launch an instance. Assign a name to your instance → Choose the 'Amazon Linux machine → Select the 'Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type' machine image.

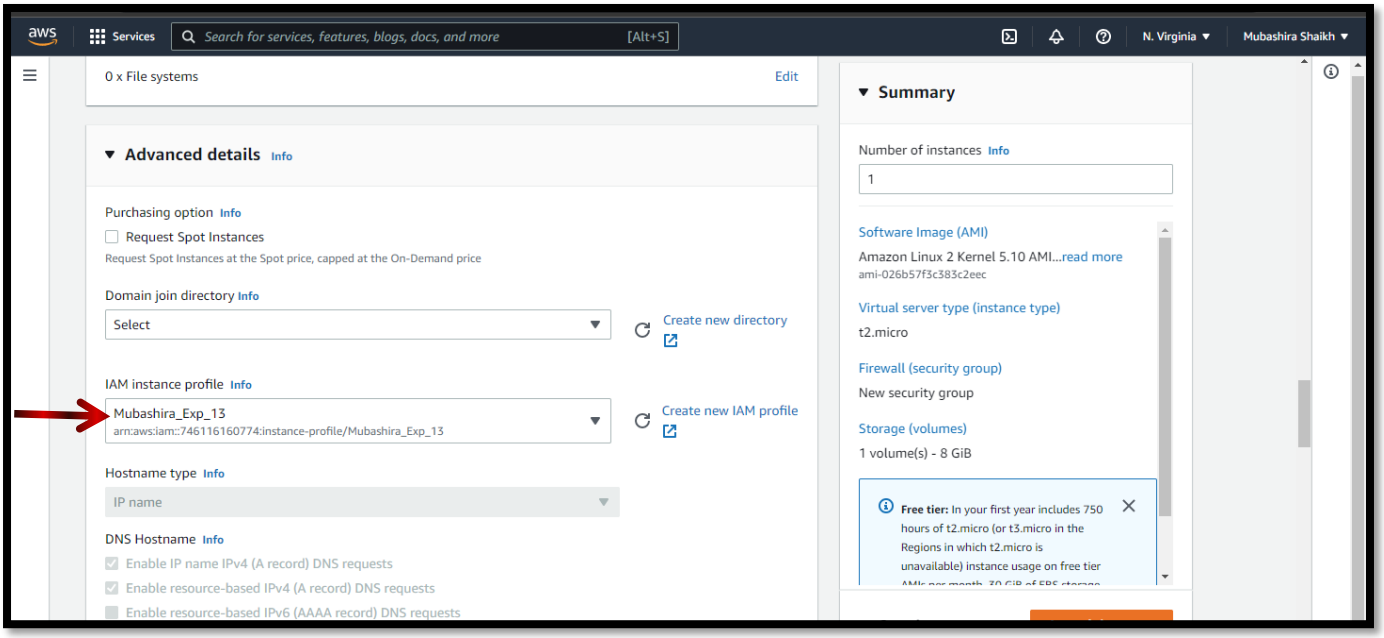


Step 6: Create a key pair → In the network settings, allow SSH traffic from anywhere → Click on edit → Add a security group for 'All traffic' and select the source type as 'anywhere'.

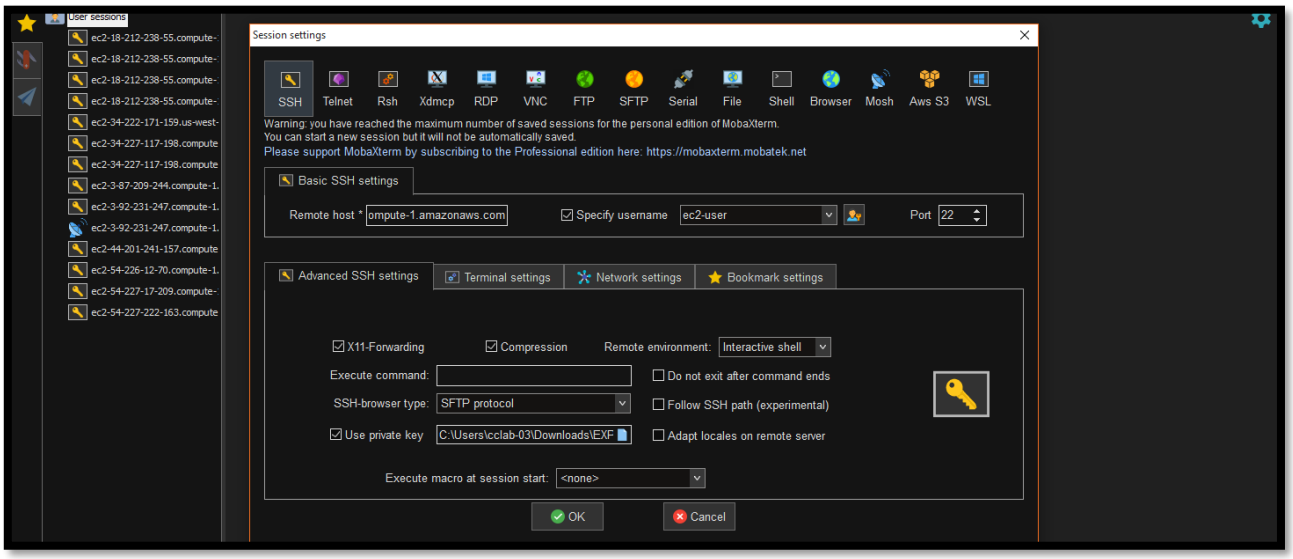




Step 7: In the advanced details, select the IAM role created. Finally, launch the instance.

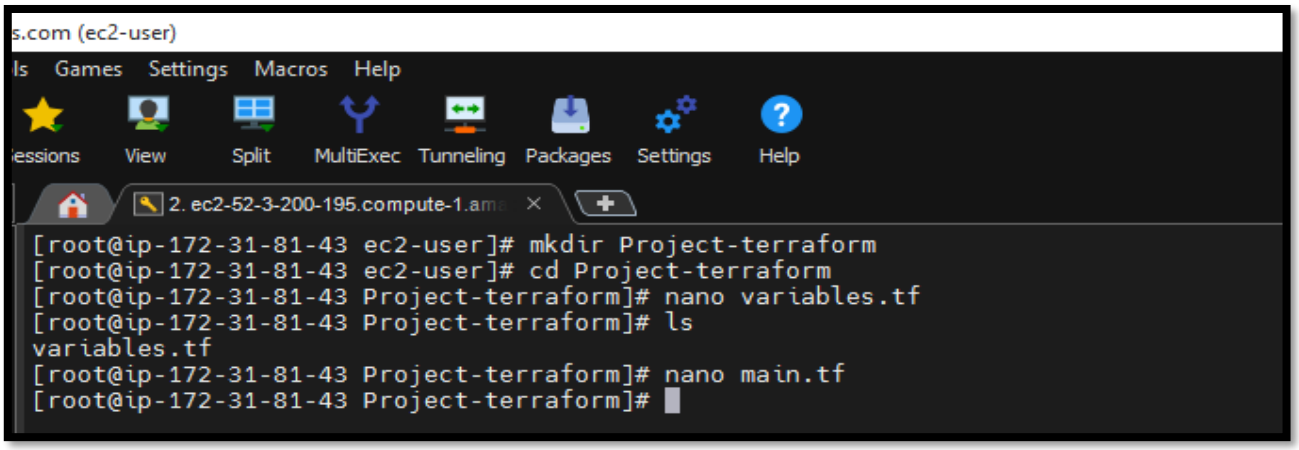


Step 8: Launch MobaXterm → Select SSH session → Copy the public DNS of your instance and paste it into the remote host. Use the downloaded key pair as the private key.

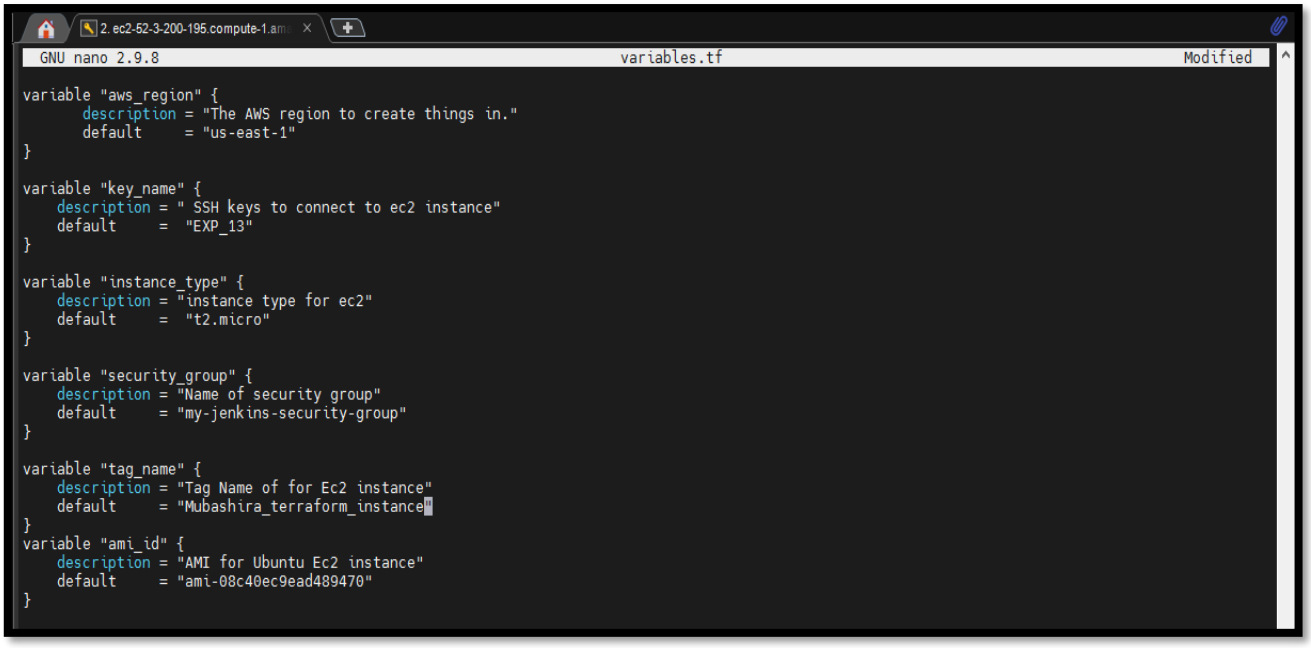


Step 9: Create Terraform files by executing the following commands –

- ⇒ **sudo su**
- ⇒ **mkdir Project-terraform**
- ⇒ **cd Project-terraform**
- ⇒ **nano variables.tf**
- ⇒ **nano main.tf**



variables.tf:



main.tf:

```
GNU nano 2.9.8                          main.tf                          Modified

  }

 # outbound from jenkis server
  egress {
    from_port   = 0
    to_port     = 65535
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags= {
    Name = var.security_group
  }
}

resource "aws_instance" "myFirstInstance" {
  ami           = var.ami_id
  key_name = var.key_name
  instance_type = var.instance_type
  security_groups= [var.security_group]
  tags= {
    Name = var.tag_name
  }
}

# Create Elastic IP address
resource "aws_eip" "myFirstInstance" {
  vpc      = true
  instance = aws_instance.myFirstInstance.id
tags= {
    Name = "jenkins_elastic_ip"
  }
}
```

Step 10: Install Terraform by executing the commands given below –

- ⇒ **wget https://releases.hashicorp.com/terraform/1.0.9/terraform 1.0.9 linux +amd64.zip**
- ⇒ **ls**
- ⇒ **unzip terraform_1.0.9_linux_amd64.zip**
- ⇒ **cp terraform /bin/**
- ⇒ **Now, check the Terraform version installed by using → terraform –version**

```
[root@ip-172-31-81-43 Project-terraform]# wget https://releases.hashicorp.com/terraform/1.0.9/terraform_1.0.9_linux_amd64.zip
--2022-10-13 06:09:16--  https://releases.hashicorp.com/terraform/1.0.9/terraform_1.0.9_linux_amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 108.138.85.31, 108.138.85.53, 108.138.85.65, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|108.138.85.31|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 32674820 (31M) [application/zip]
Saving to: 'terraform_1.0.9_linux_amd64.zip'

100%[=============================================================================================================>] 32,674,820  --.-K/s   in 0.1s

2022-10-13 06:09:16 (230 MB/s) - 'terraform_1.0.9_linux_amd64.zip' saved [32674820/32674820]

[root@ip-172-31-81-43 Project-terraform]# ls
main.tf  terraform_1.0.9_linux_amd64.zip  variables.tf
[root@ip-172-31-81-43 Project-terraform]# unzip terraform_1.0.9_linux_amd64.zip
Archive:  terraform_1.0.9_linux_amd64.zip
  inflating: terraform
[root@ip-172-31-81-43 Project-terraform]#  cp terraform /bin/
[root@ip-172-31-81-43 Project-terraform]# terraform --version
Terraform v1.0.9
on linux_amd64

Your version of Terraform is out of date! The latest version
is 1.3.2. You can update by downloading from https://www.terraform.io/downloads.html
[root@ip-172-31-81-43 Project-terraform]# terraform init
There are some problems with the configuration, described below.
```

Step 11: Execute the following Terraform commands to create an EC2 instance using Terraform –

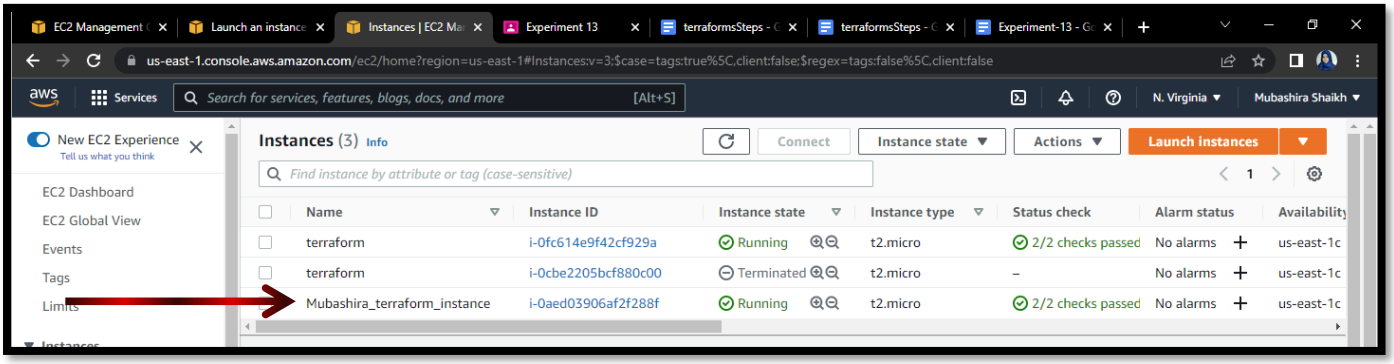
- ⇒ **terraform init**
- ⇒ **terraform plan**
- ⇒ **terraform apply**
  - ♦ **Enter 'yes' to complete the creation of EC2 instance using Terraform**

```
[root@ip-172-31-81-43 Project-terraform]# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.34.0...
- Installed hashicorp/aws v4.34.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[root@ip-172-31-81-43 Project-terraform]#
```

```
[root@ip-172-31-81-43 Project-terraform]# clear
[root@ip-172-31-81-43 Project-terraform]# terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_eip.myFirstInstance will be created
  + resource "aws_eip" "myFirstInstance" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = (known after apply)
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
      + private_dns          = (known after apply)
      + private_ip           = (known after apply)
      + public_dns           = (known after apply)
      + public_ip            = (known after apply)
      + public_ipv4_pool     = (known after apply)
      + tags                 = {
          + "Name" = "jenkins_elastic_ip"
        }
      + tags_all             = {
          + "Name" = "jenkins_elastic_ip"
        }
      + vpc                  = true
    }
```

```
[root@ip-172-31-81-43 Project-terraform]# clear
[root@ip-172-31-81-43 Project-terraform]# terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_eip.myFirstInstance will be created
  + resource "aws_eip" "myFirstInstance" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = (known after apply)
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
      + private_dns          = (known after apply)
      + private_ip           = (known after apply)
      + public_dns           = (known after apply)
      + public_ip            = (known after apply)
      + public_ipv4_pool     = (known after apply)
      + tags                 = {
          + "Name" = "jenkins_elastic_ip"
        }
      + tags_all             = {
          + "Name" = "jenkins_elastic_ip"
        }
      + vpc                  = true
    }

  # aws_instance.myFirstInstance will be created
  + resource "aws_instance" "myFirstInstance" {
      + ami                                  = "ami-08c40ec9ead489470"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
```

Step 12: Now go to EC2 console, to see the new instances up and running.



Finally, delete the IAM role and terminate both the instances.

5. Explain following Terraform commands in one line

(i) terraform init

The terraform init command initializes a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

(ii) 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.

(iii) terraform plan

The terraform plan command creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure.

(iv) terraform apply

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

(v) terraform destroy

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