**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.

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

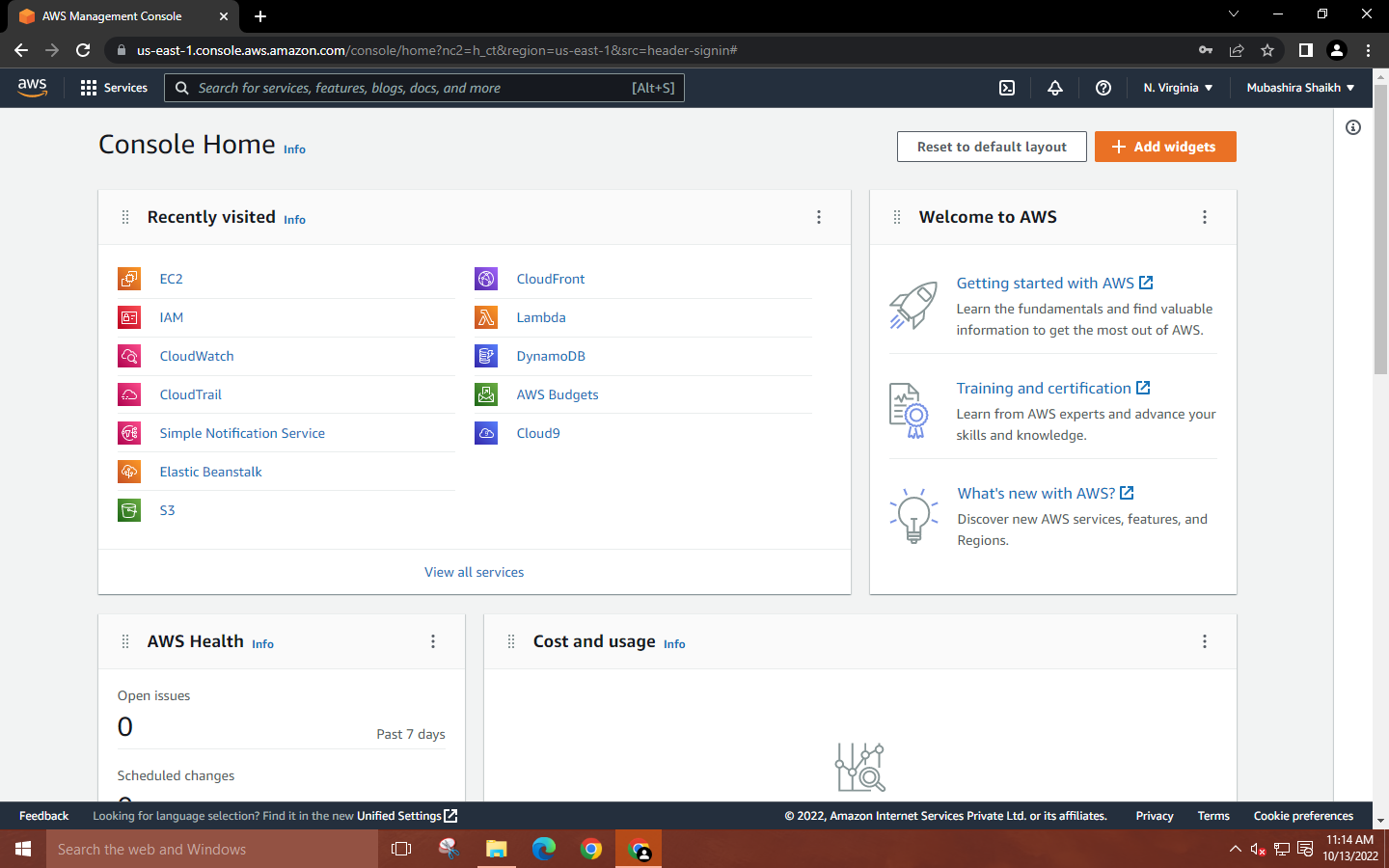
🡺Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.

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.

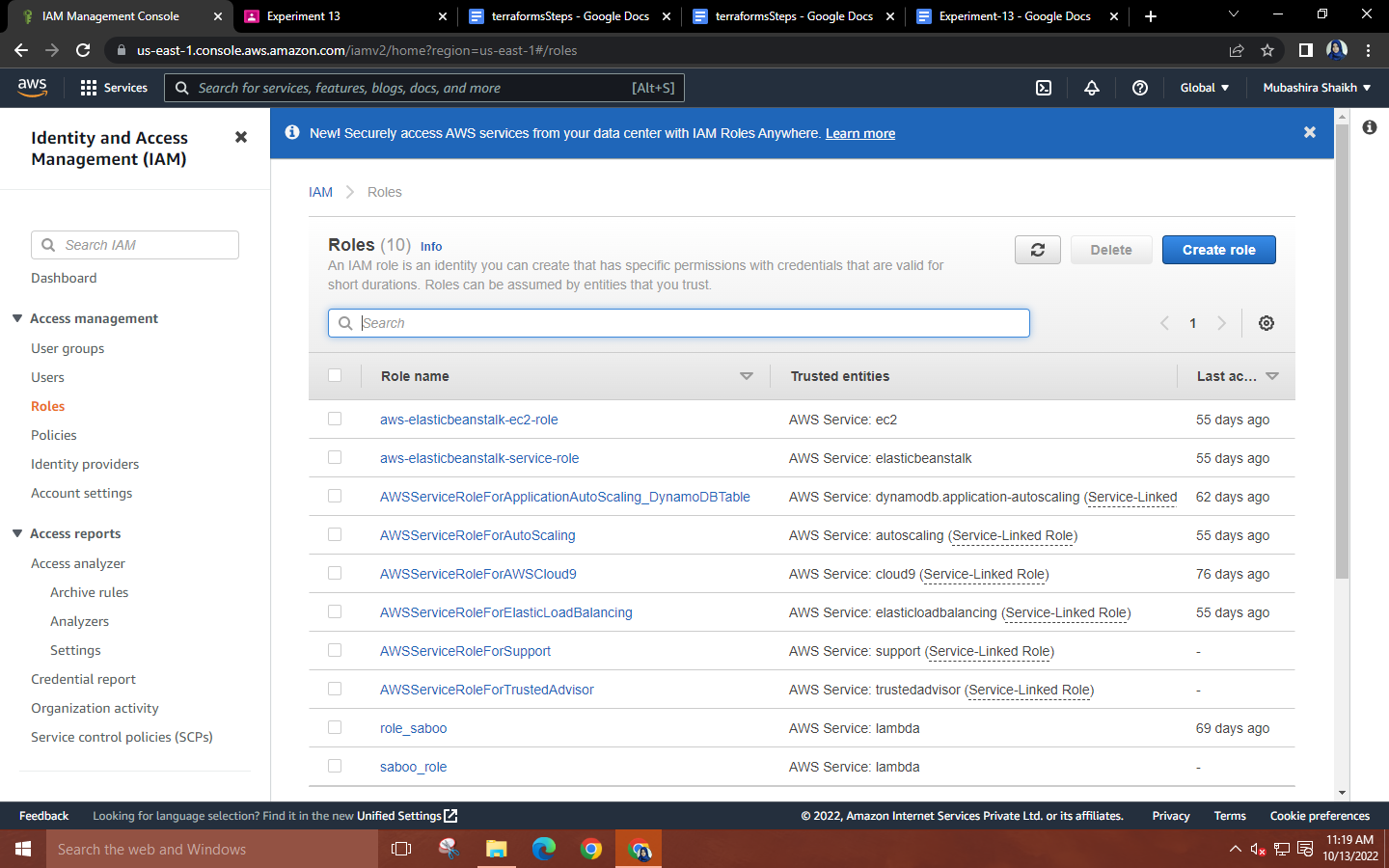
Version control is an important part of IaC, and your configuration files should be under source control just like any other software source code file. Deploying your infrastructure as code also means that you can divide your infrastructure into modular components that can then be combined in different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

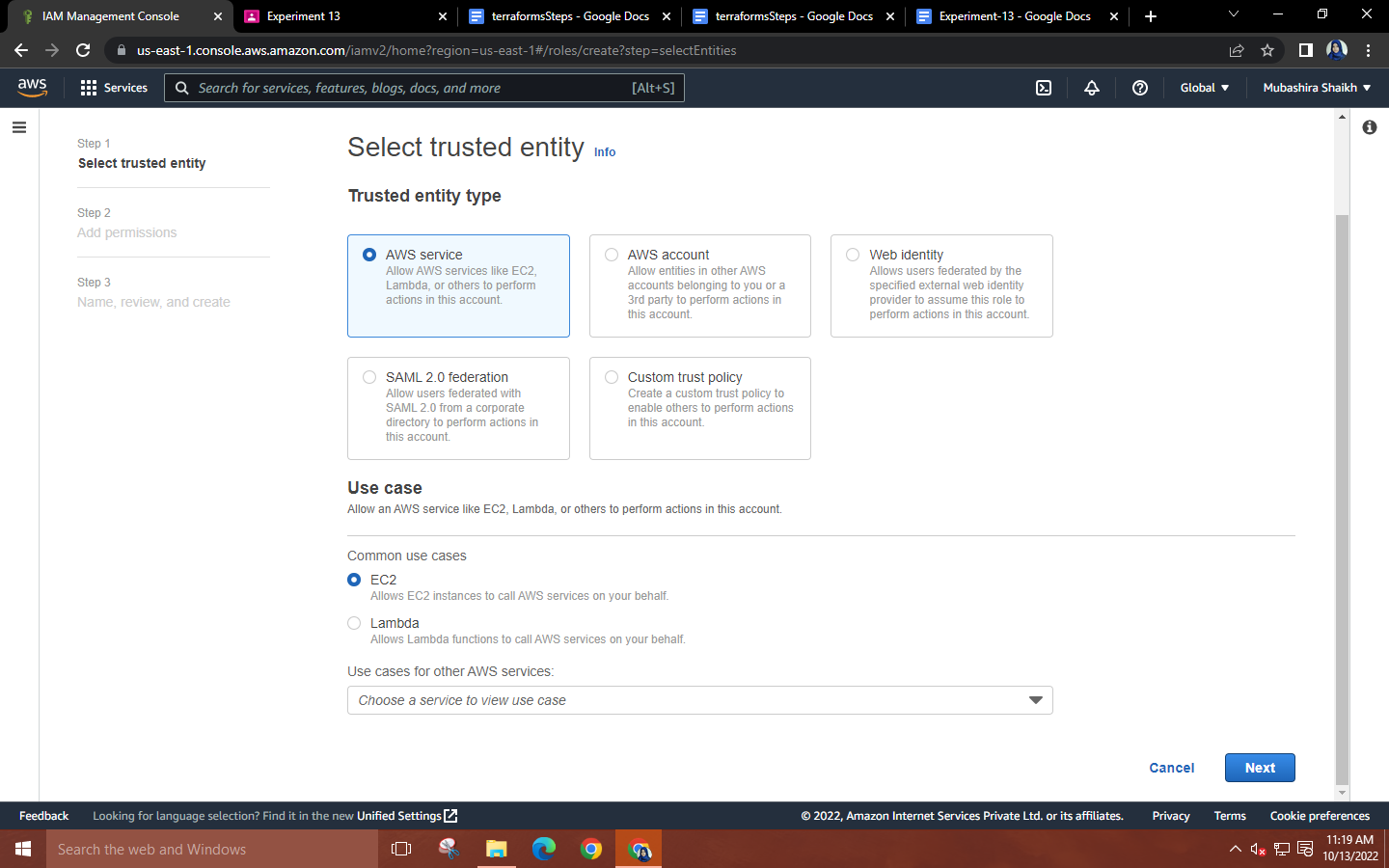
1. **Perform an experiment, to understand Terraform’s lifecycle, and core concepts/terminologies, and install it on a Linux Machine.**

Step 1: AWS Management Console

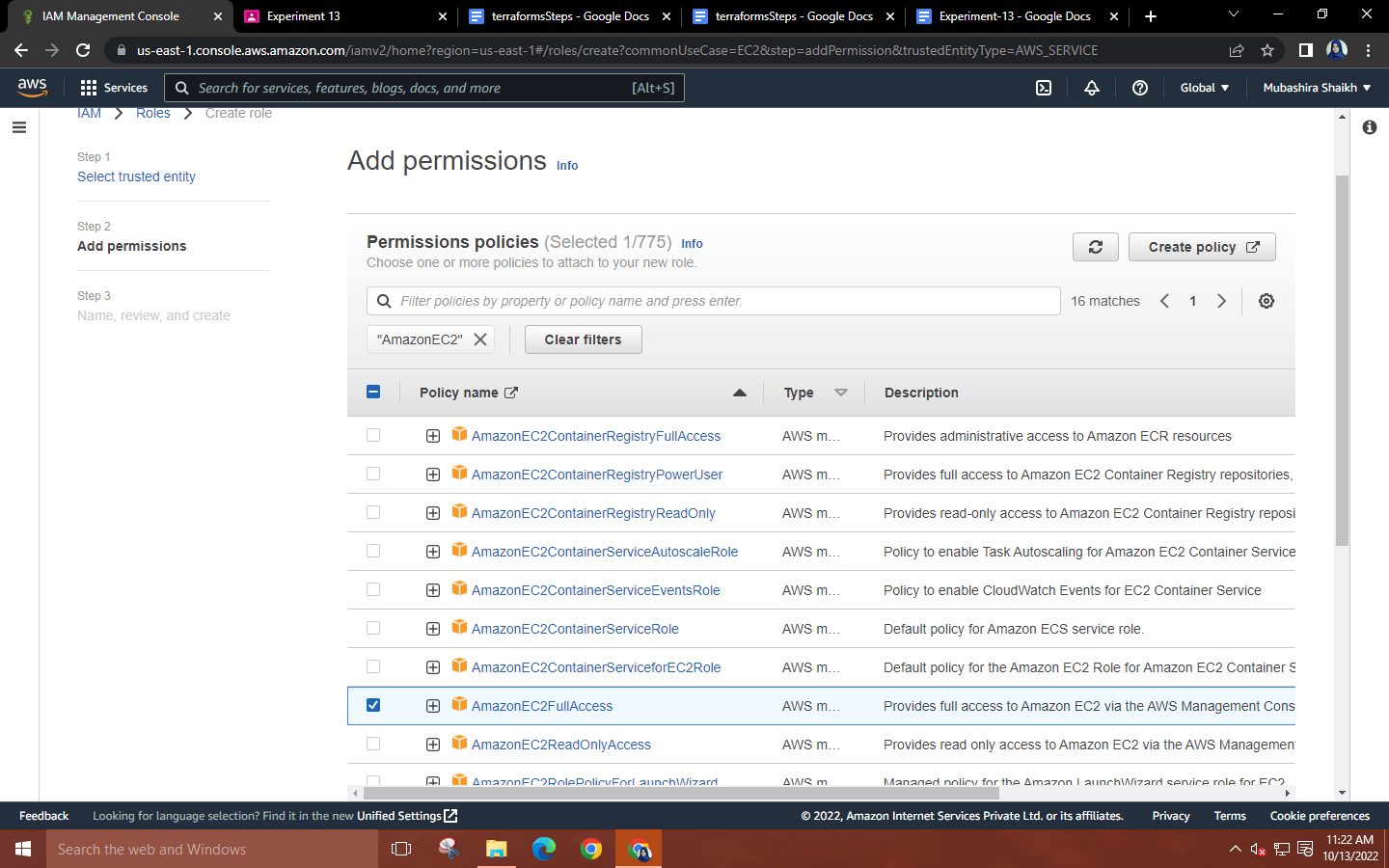


**Step 2: Search for IAM 🡪 Select Roles 🡪 Create role 🡪 Set Trusted entity type to AWS service, select the use case to EC2.**

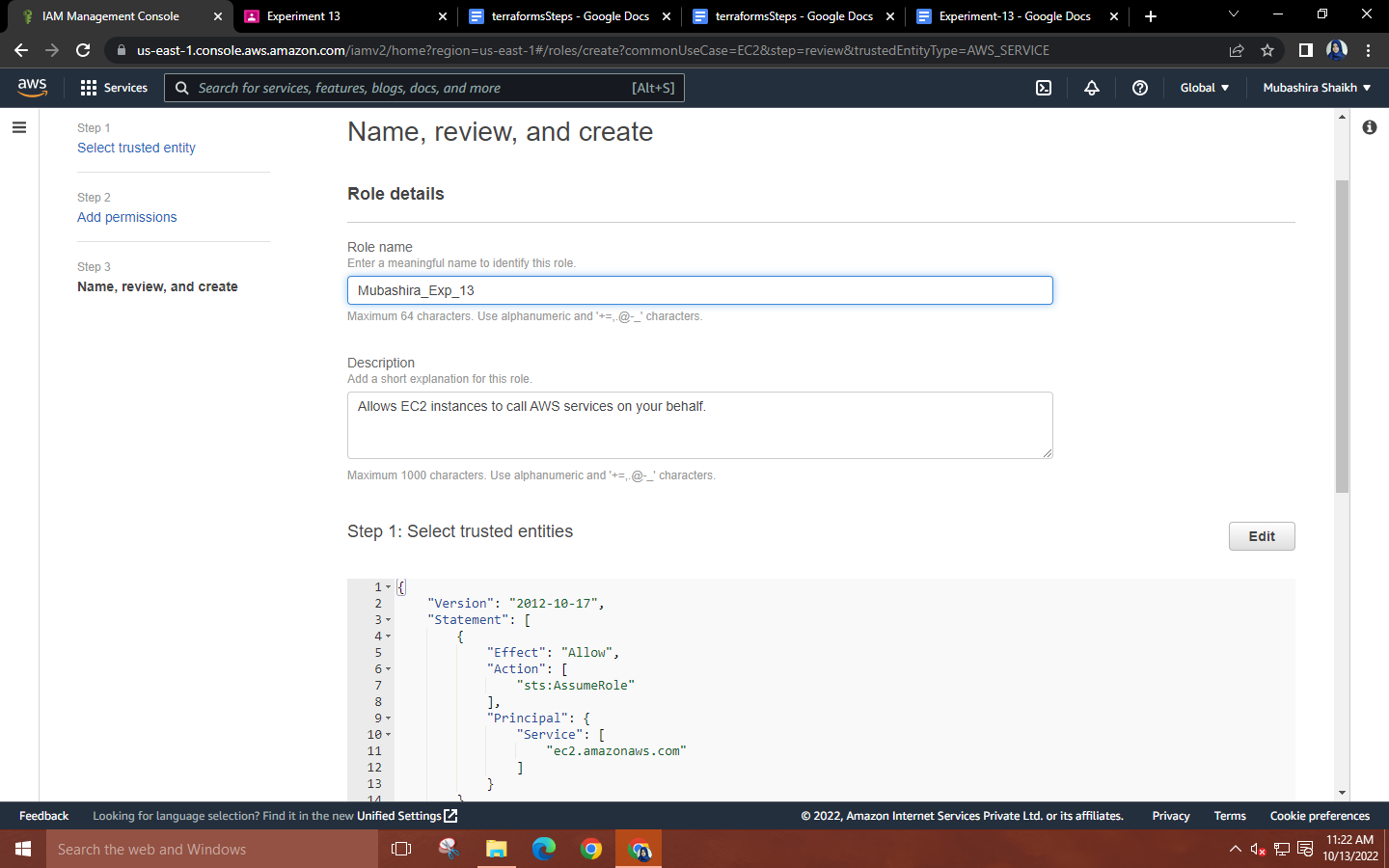
****

****

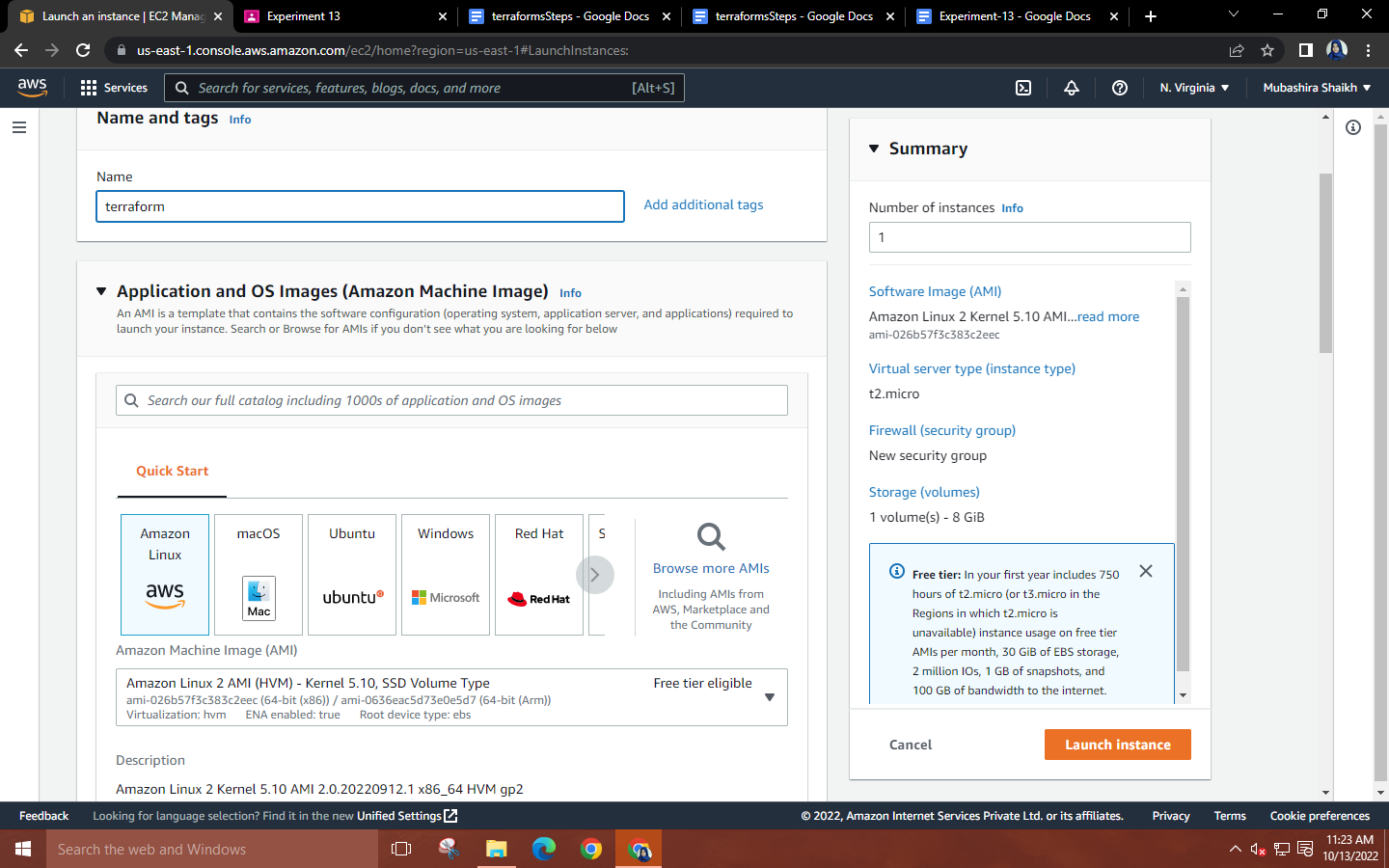
**Step 3: Search for AmazonEC2FullAccess in Permission policies 🡪 Select the policy 🡪 Next**

****

**Step 4: Assign a name to your role 🡪 Create a role**

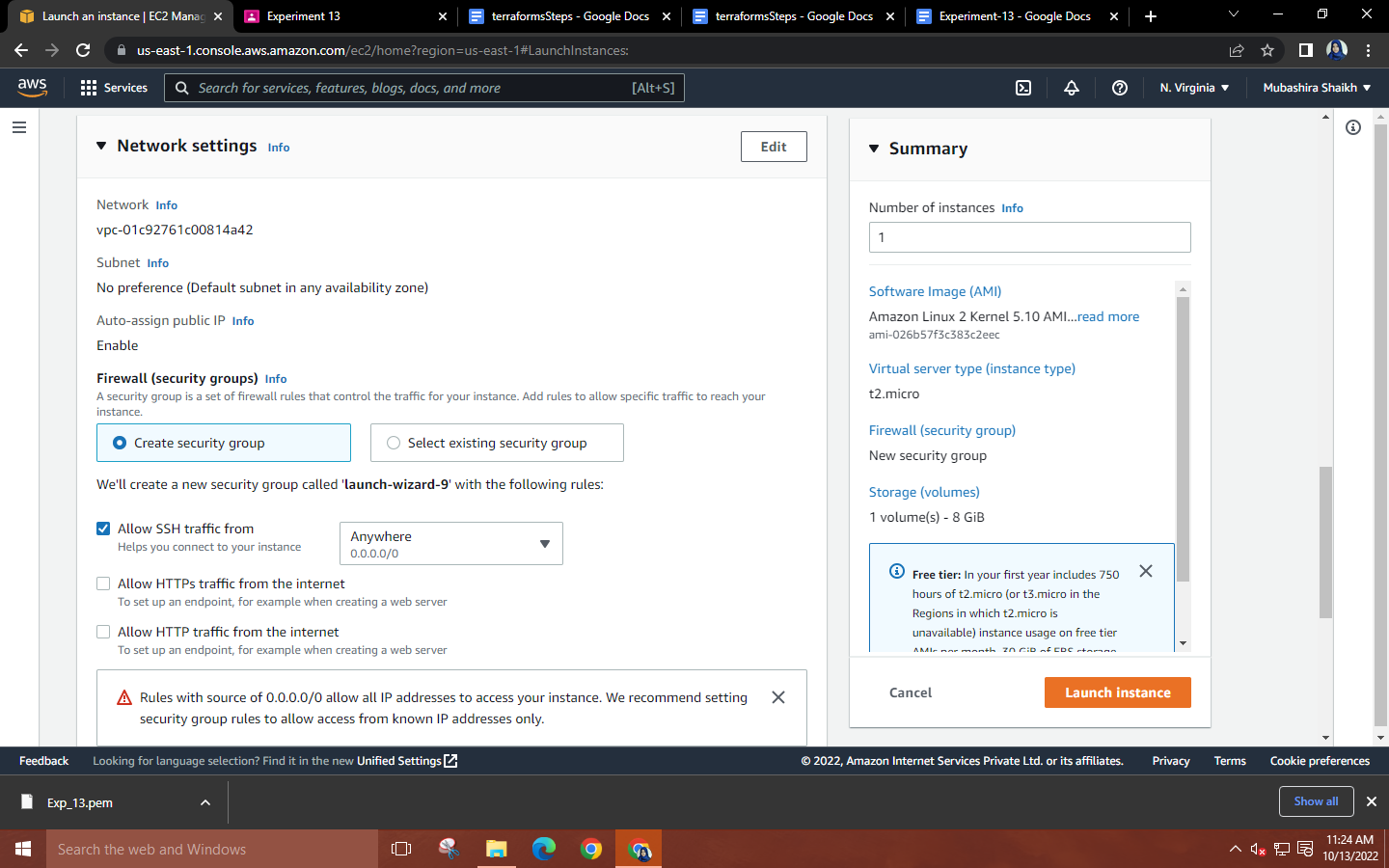
****

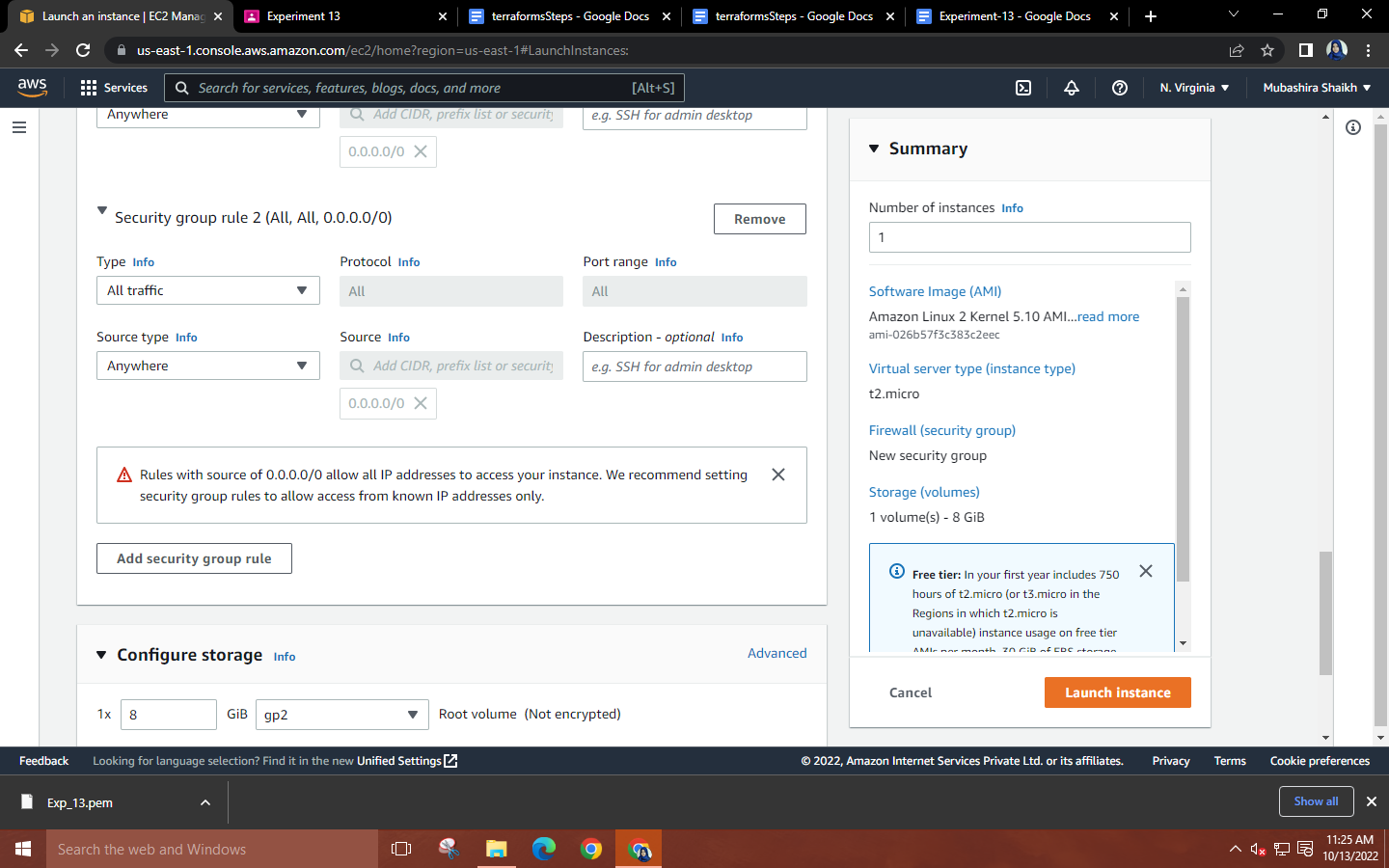
**Step 5: Search for EC2 🡪 Select ‘Launch an instance. Assign a name to your instance 🡪 Choose the ‘Amazon Linux machine 🡪 Select the ‘Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type’ machine image.**

****

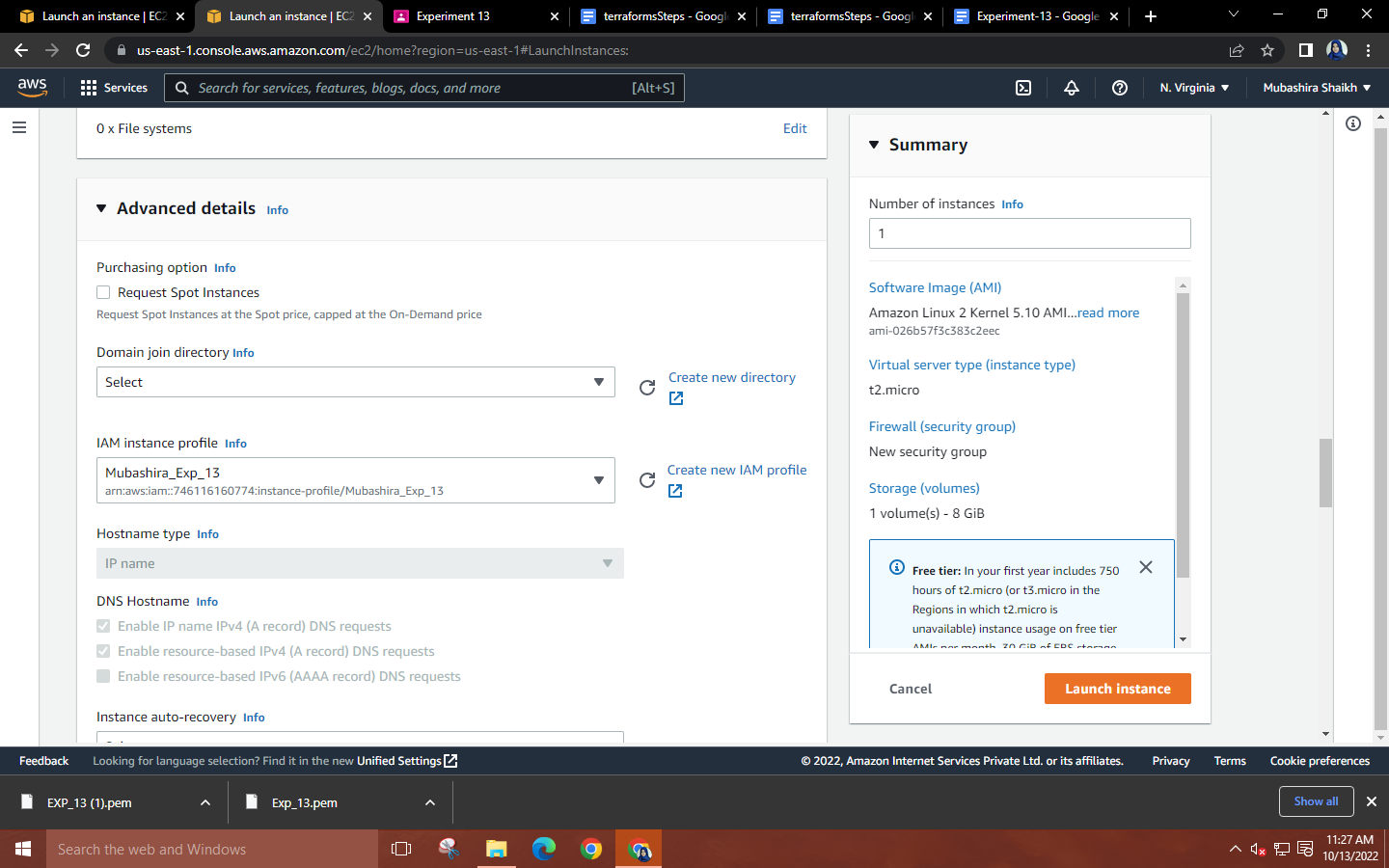
**Step 6: Create a key pair 🡪 In the network settings, allow SSH traffic from anywhere 🡪**

**Click on edit 🡪 Add a security group for ‘All traffic’ and select the source type as ‘anywhere’.**

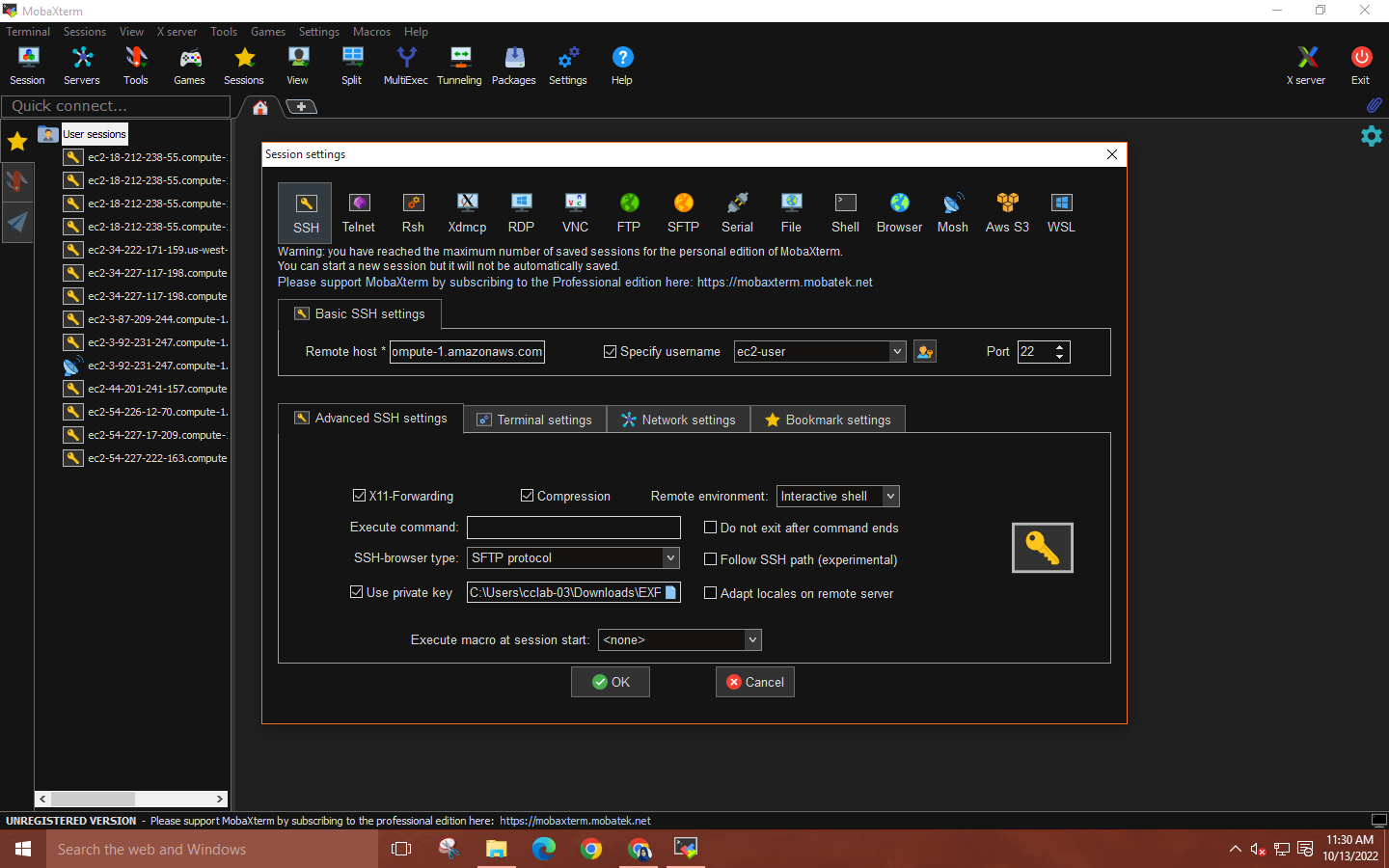
****

****

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

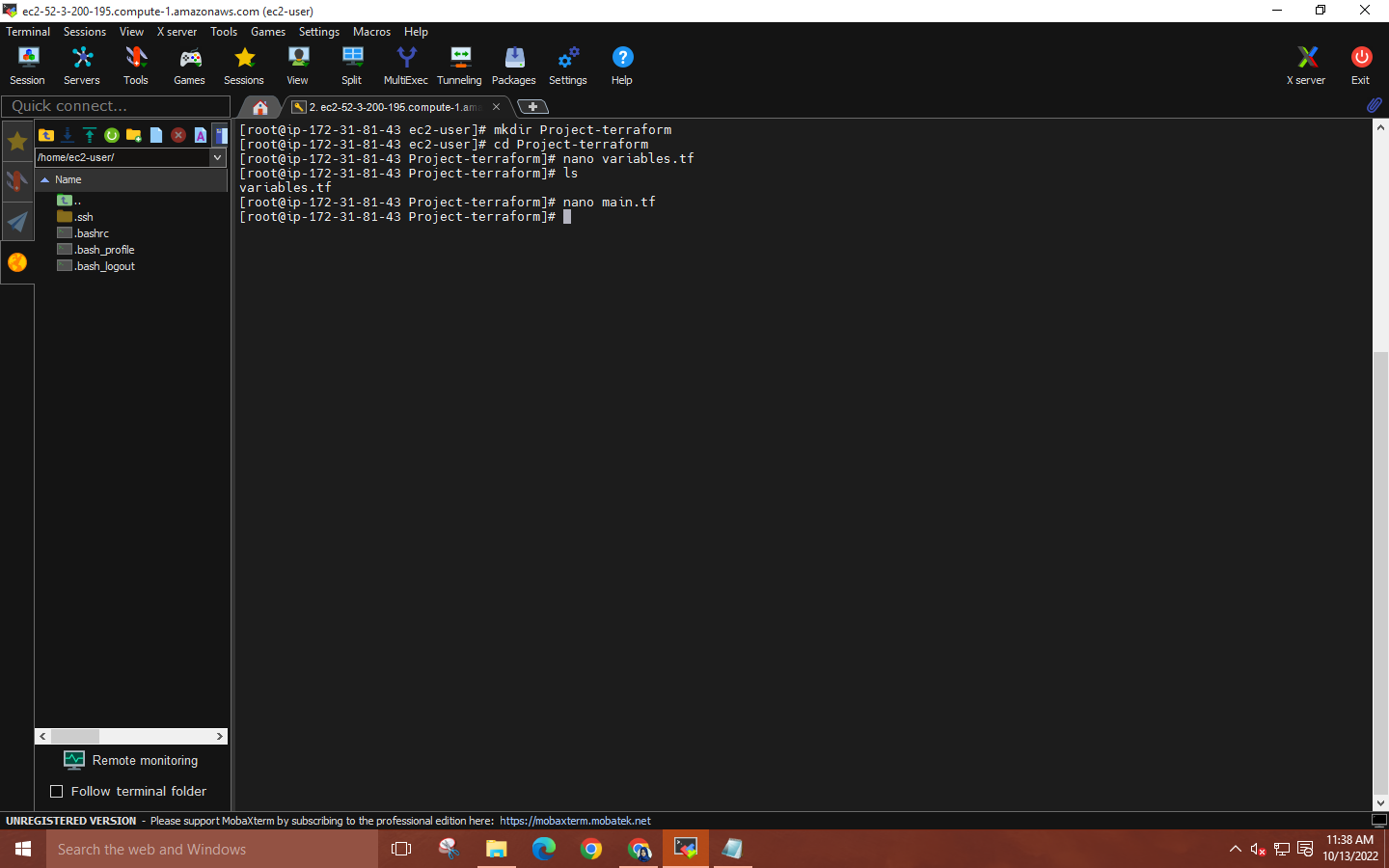
****

**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.**

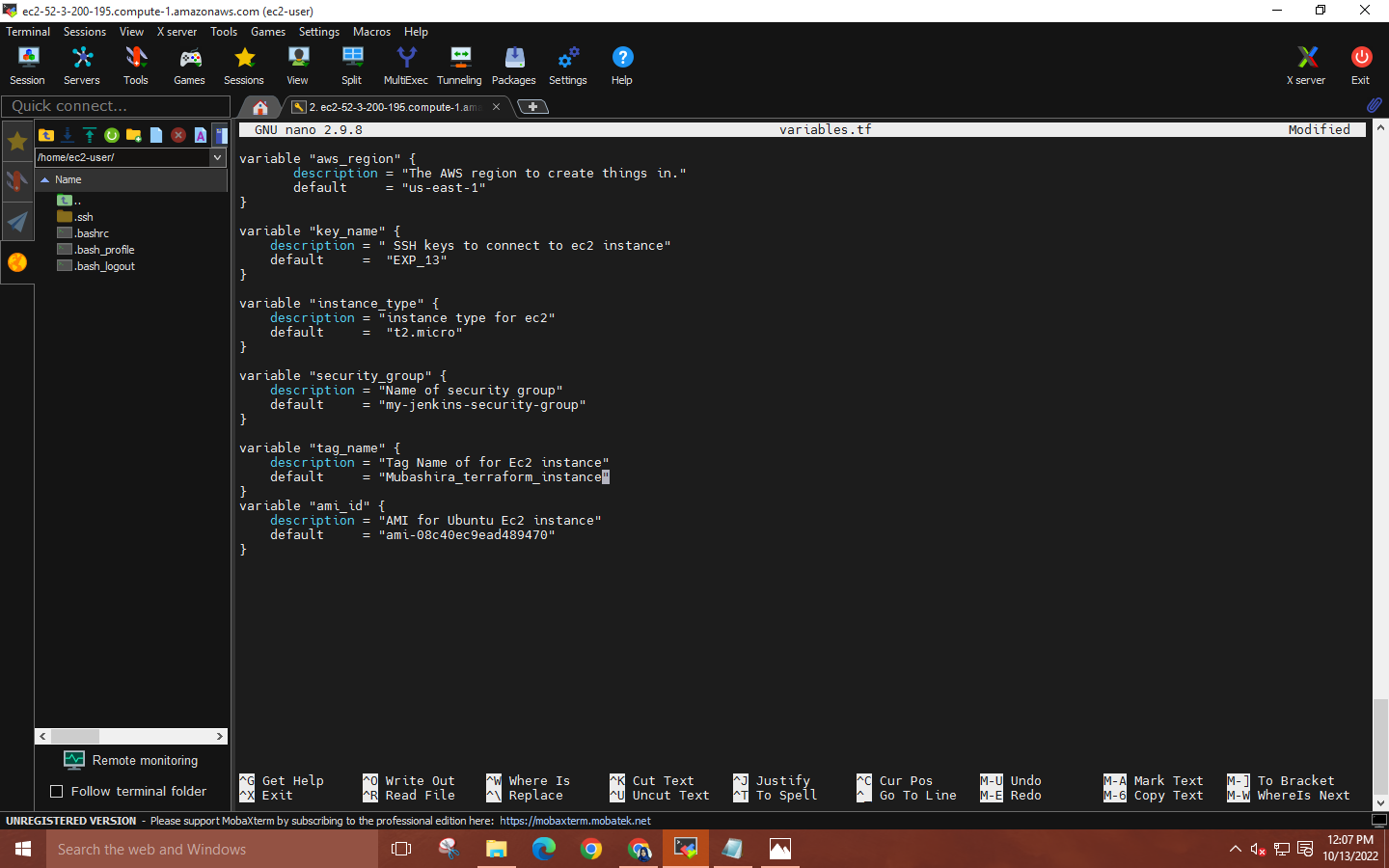
****

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

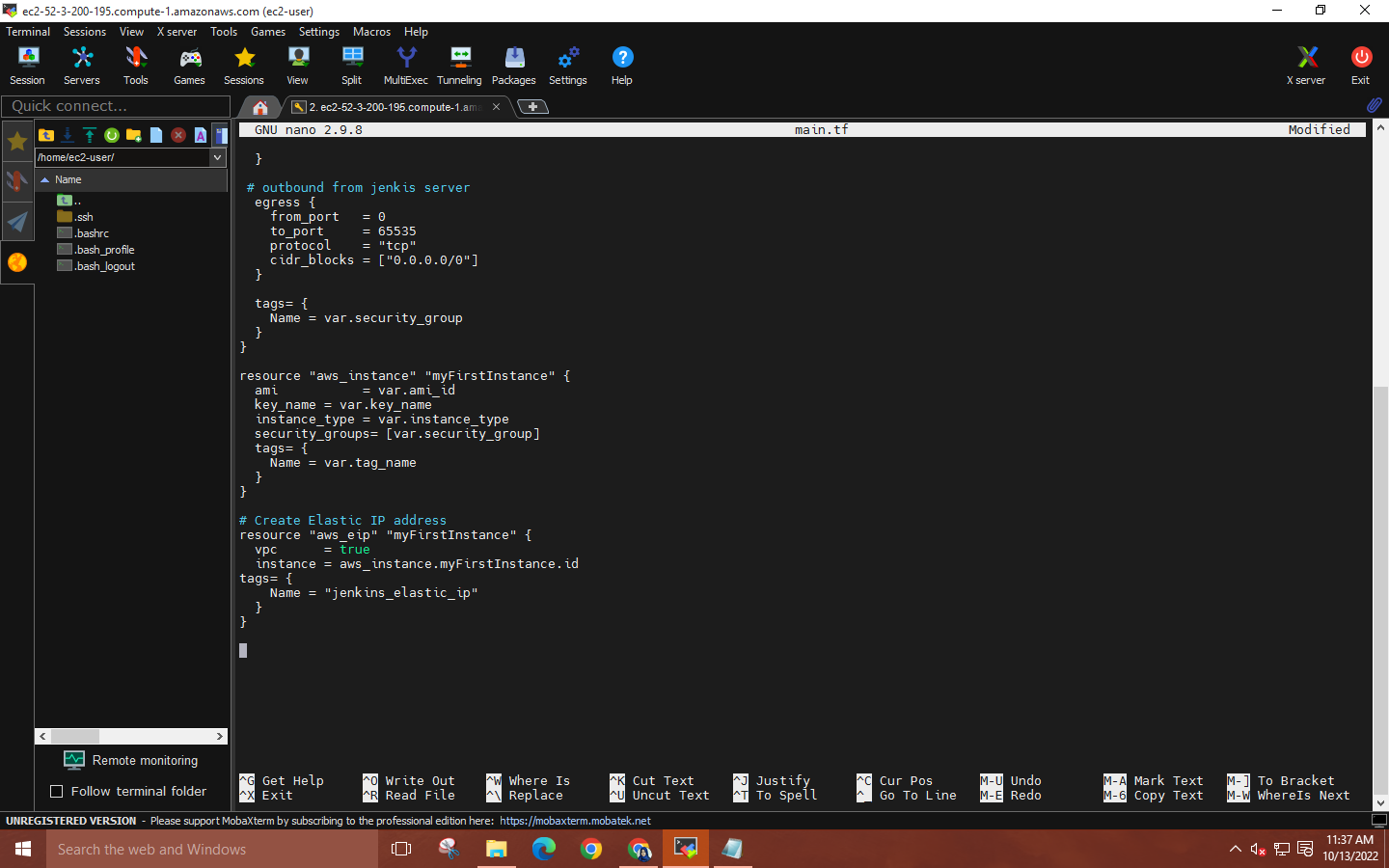
* **sudo su**
* **mkdir Project-terraform**
* **cd Project-terraform**
* **nano variables.tf**
* **nano main.tf**

****

**variables.tf:**

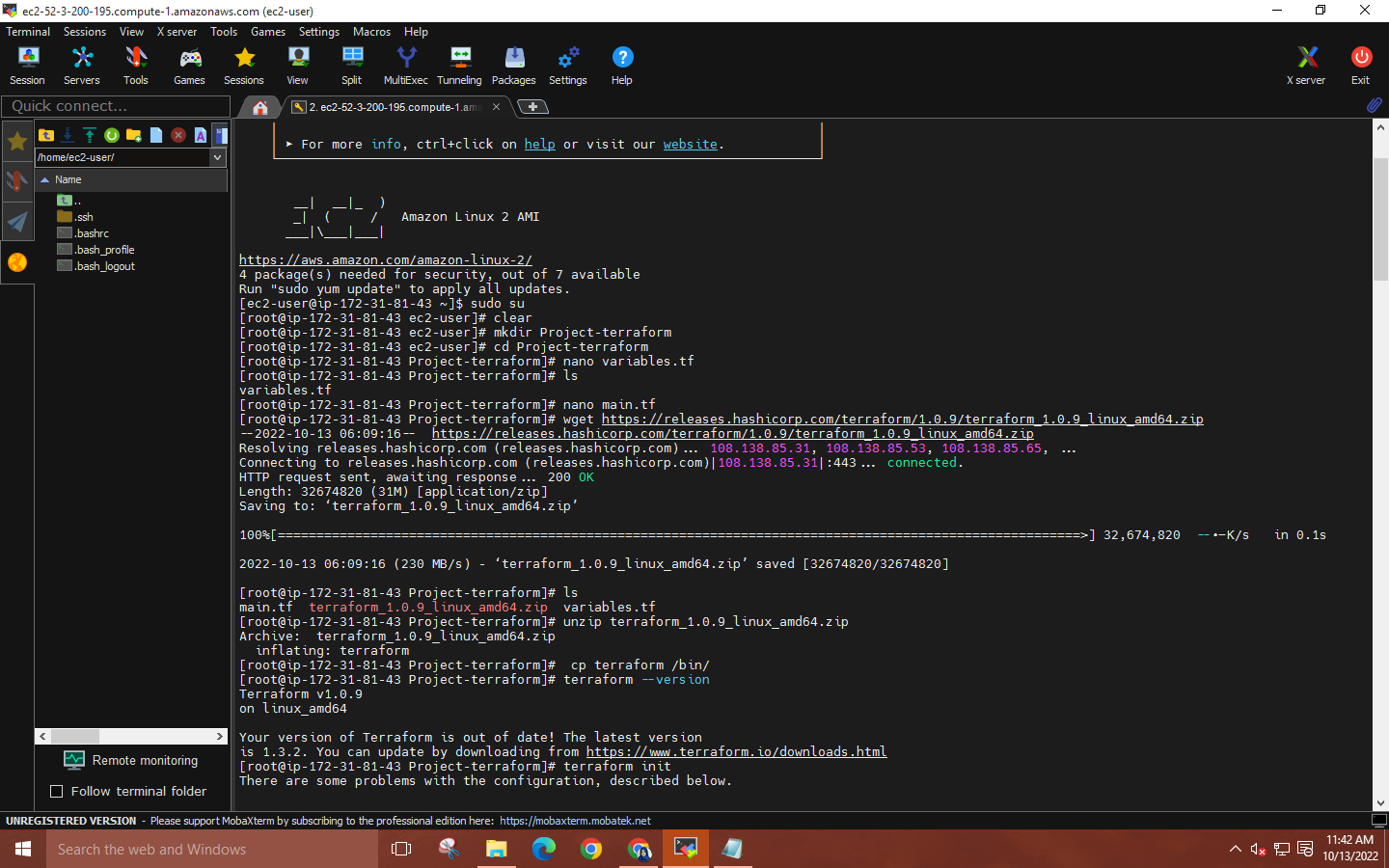
****

**main.tf:**

****

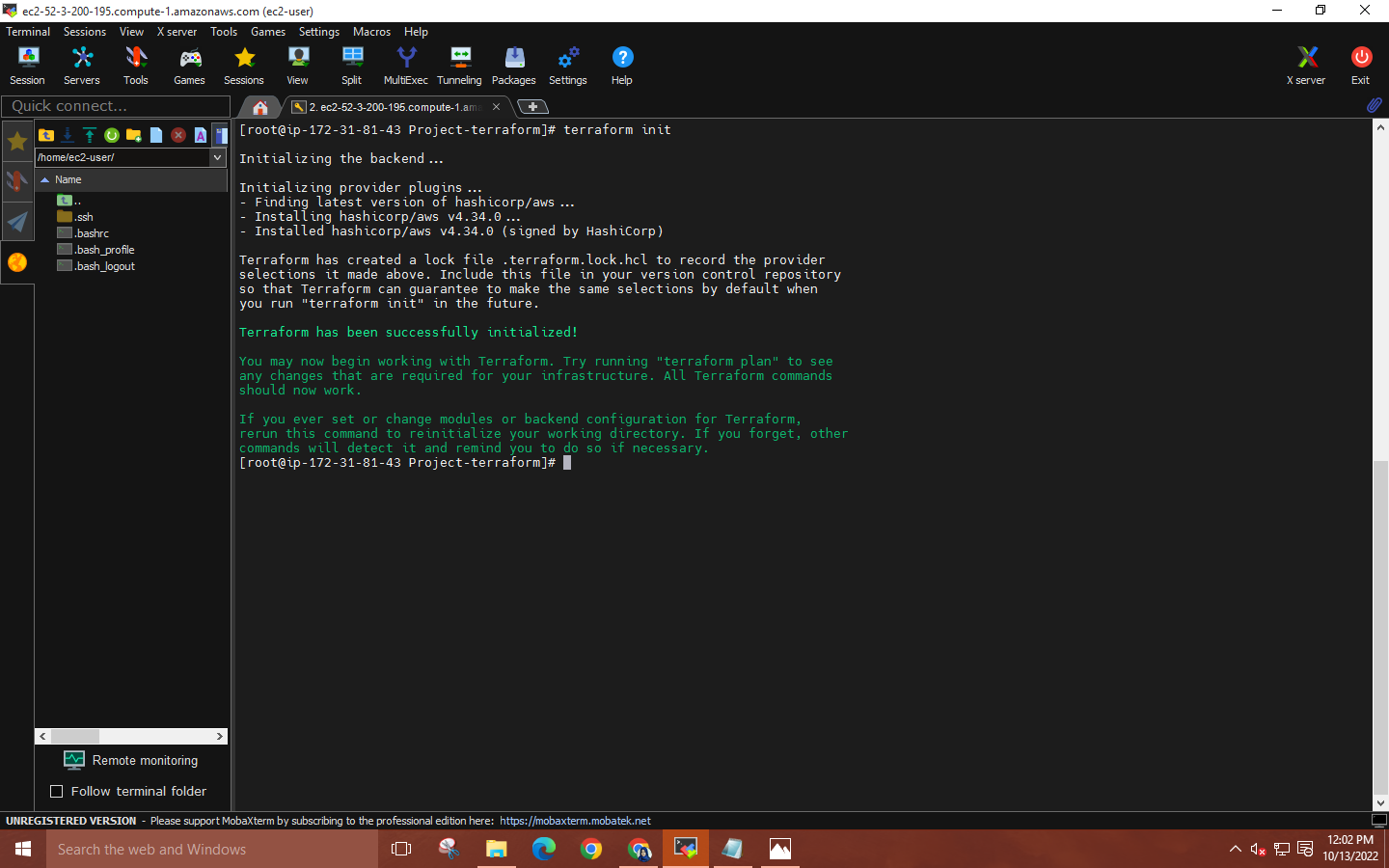
**Step 10: Install Terraform by executing the commands given below –**

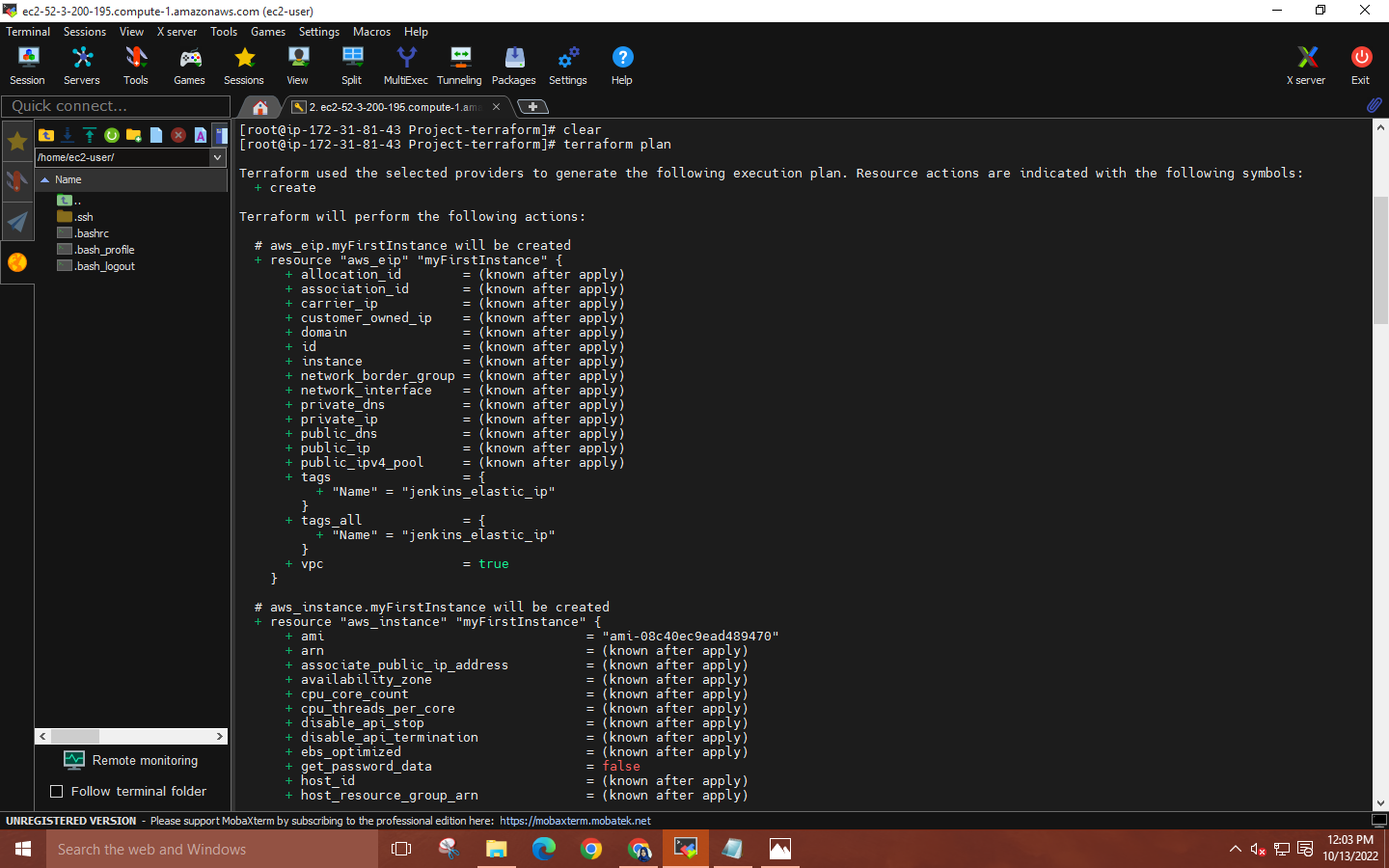
* **wget** [**https://releases.hashicorp.com/terraform/1.0.9/terraform 1.0.9 linux +amd64.zip**](https://releases.hashicorp.com/terraform/1.0.9/terraform%201.0.9%20linux%20+amd64.zip)
* **ls**
* **unzip terraform\_1.0.9\_linux\_amd64.zip**
* **cp terraform /bin/**
* **Now, check the Terraform version installed by using 🡪 terraform –version**

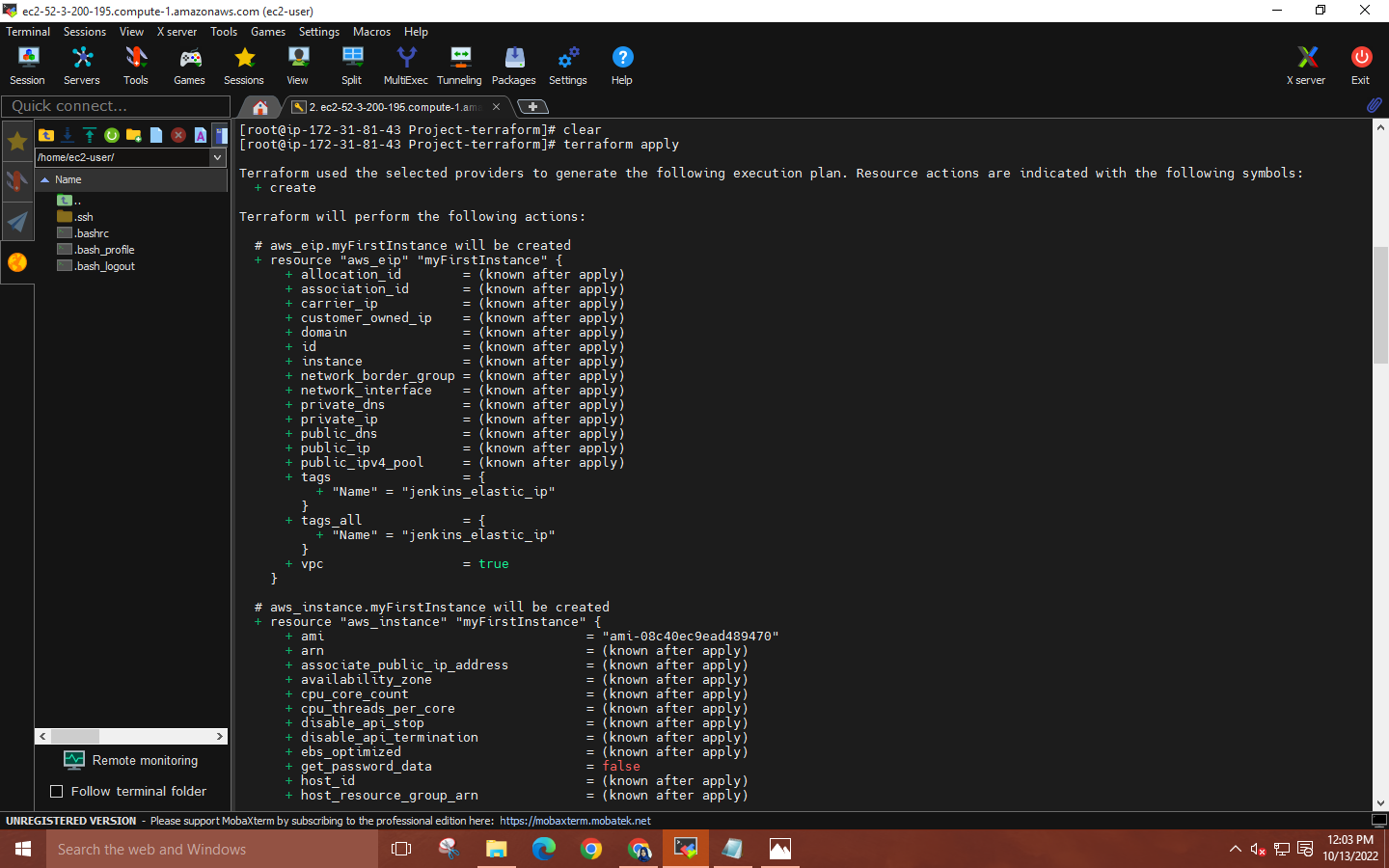
****

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

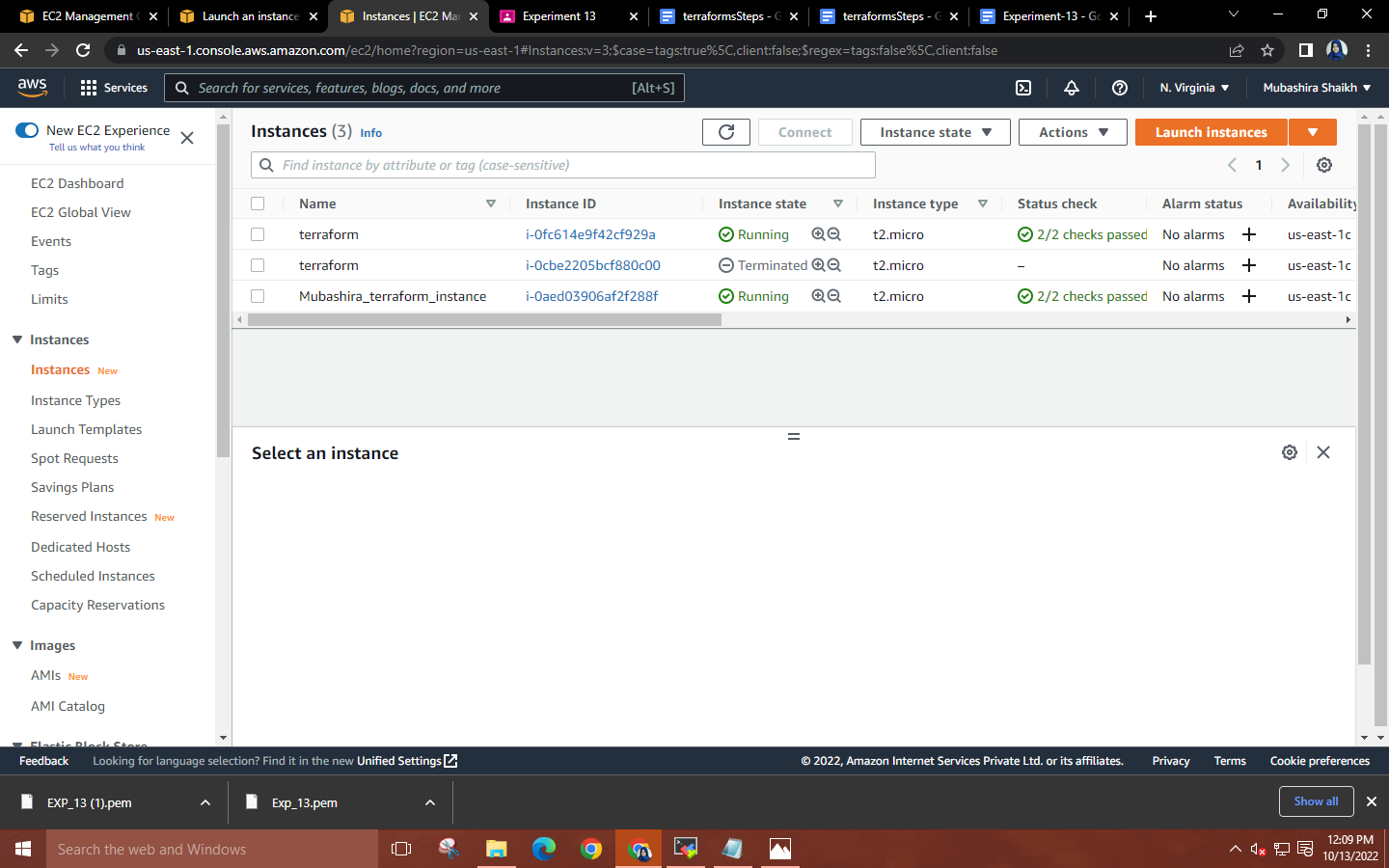
* **terraform init**
* **terraform plan**
* **terraform apply**
* **Enter ‘yes’ to complete the creation of EC2 instance using Terraform**

****

****

****

**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.