Project assignment task description summary:

Through infrastructure automation my organization is using more of the DevOps methodology and it want to facilitate testing, deployment and use its benefits for disaster recovery.

A centralized server for Jenkins will be set up. Terraform will be used to build an EC2 Instance which should have basic infrastructure automation software installed.

***Tools required**:*
Terraform, AWS account with security credentials, Keypair

Expected Deliverables:
Launch an EC2 instance using Terraform
Connect to the instance
Install Jenkins, Java and Python in the instance

Introduction

For this assignment, the given Ubuntu Lab Environment is the basis workspace. Furthermore AWS Webservices are used through given lab credentials.

Terraform and Jenkins Project

Commands to install Terraform

The following commands are entered to install Terraform and check the installed version

1)
fssl: Fail silently on (location and other) server errors
curl: Transfers data to or from a server
The Command adds the private key to trust the HashiCorp package when downloading it

Code:
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -

Terminal:

---

2)
Terraform package is installed

Code:
sudo apt install -y terraform

Terminal:

---

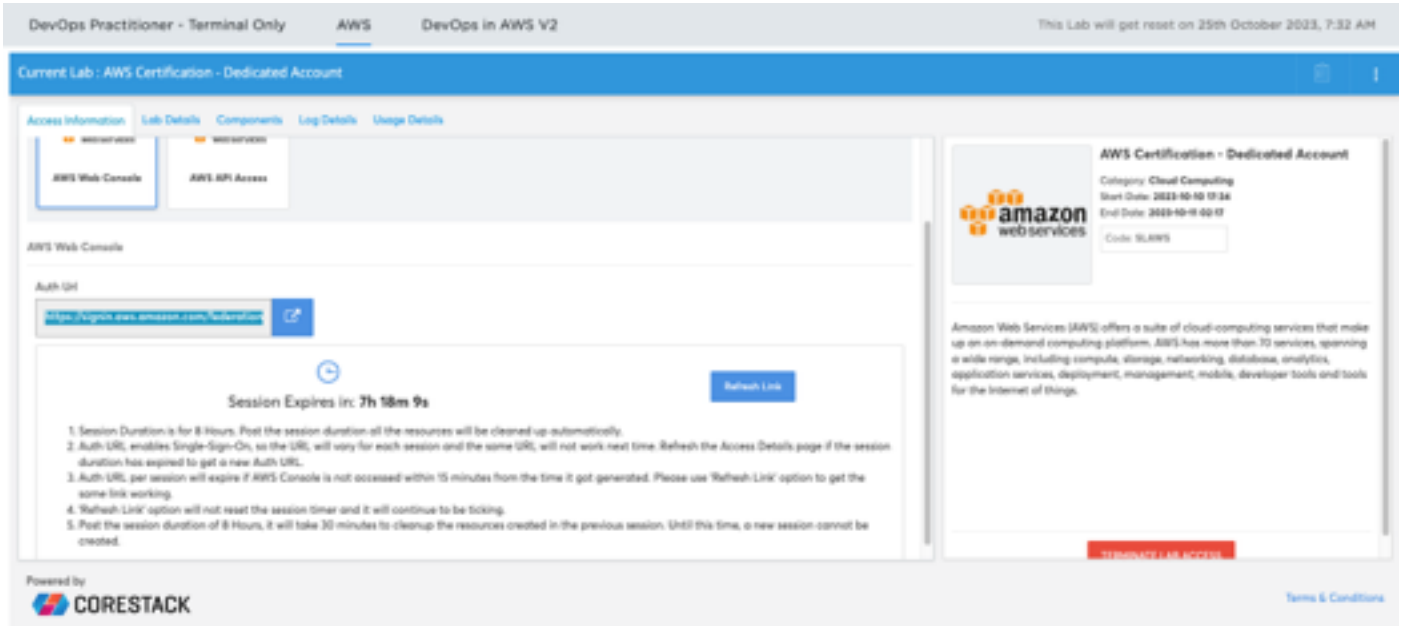
3)
Directory “ProjectTerraform“ for Terraform is created on the desktop

Code:
mkdir ProjectTerraform

Terminal:

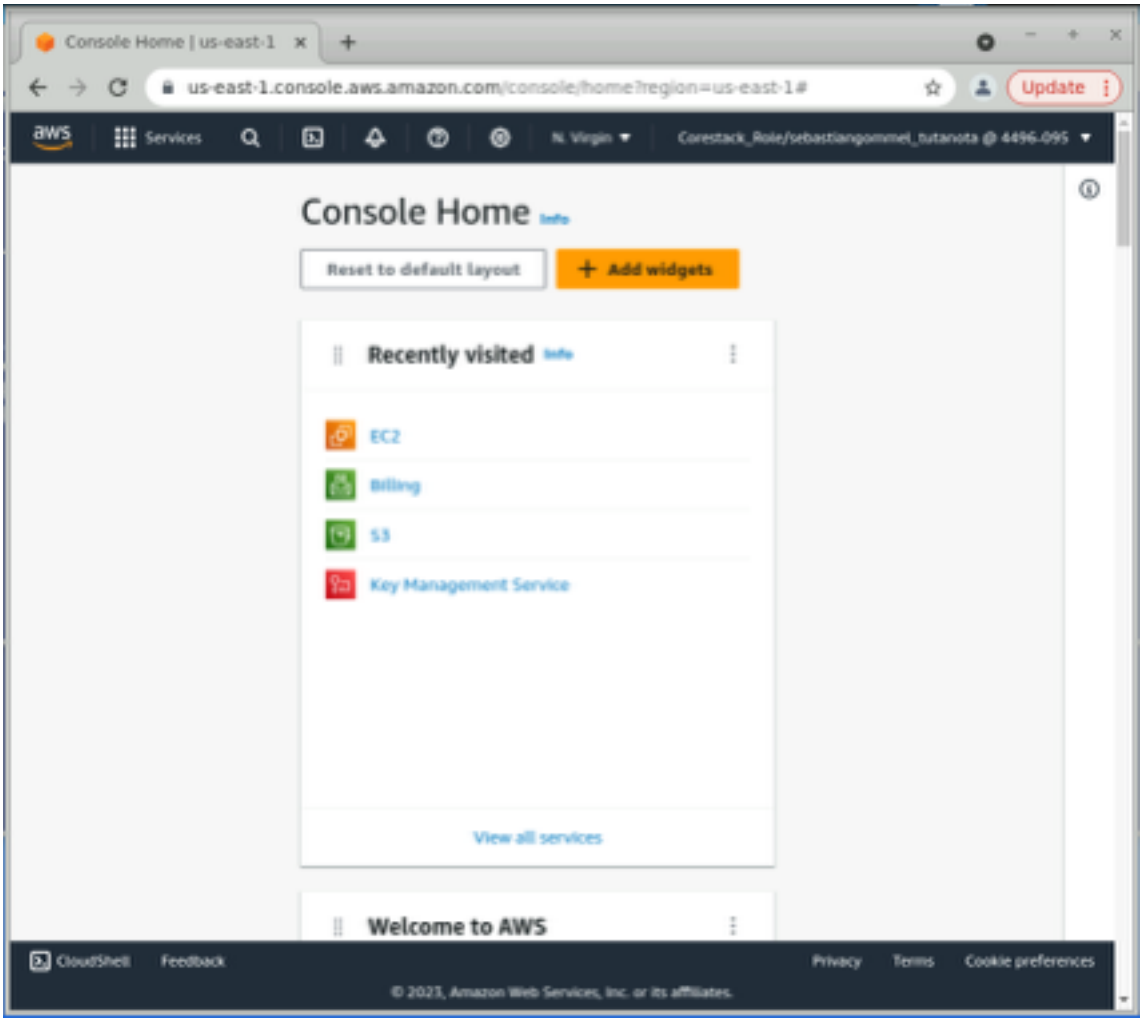
---

4)
Login to AWS EC2 through Single-Sign-On URL



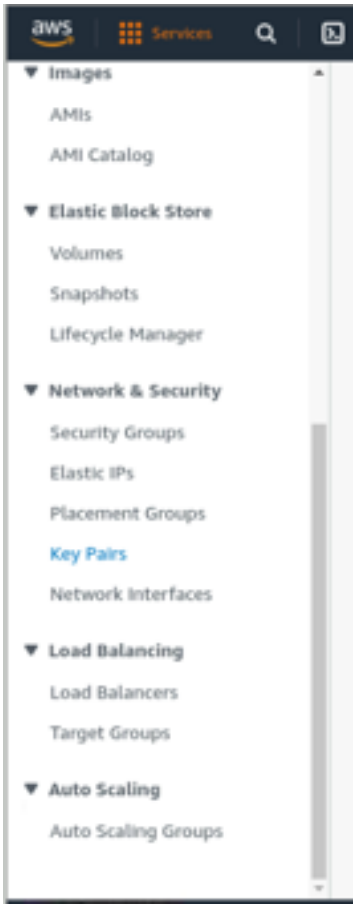
---

5)
AWS EC2 console is opened with the Chrome Browser, EC2 is selected (in Services if not in recent)



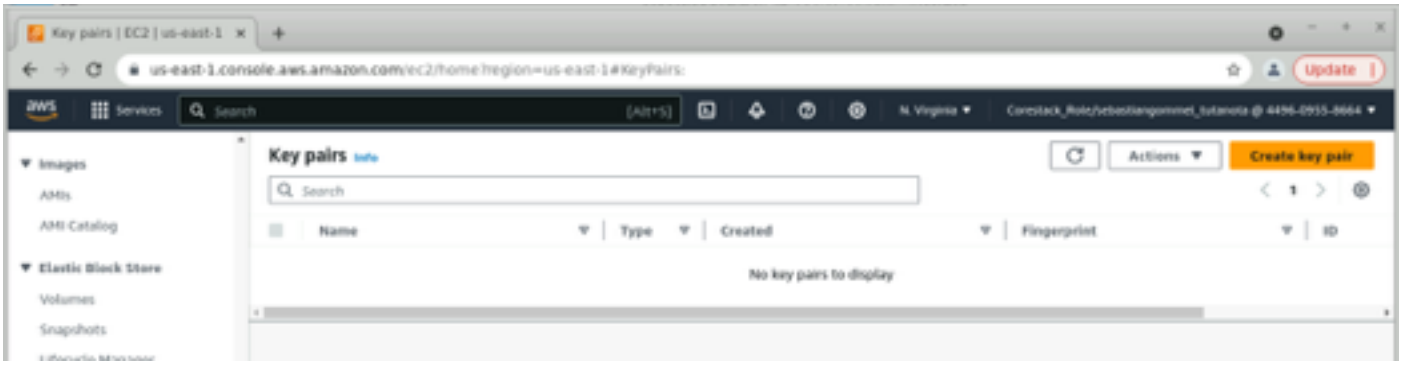
---

6)
Key Pairs is selected here in Network and Security



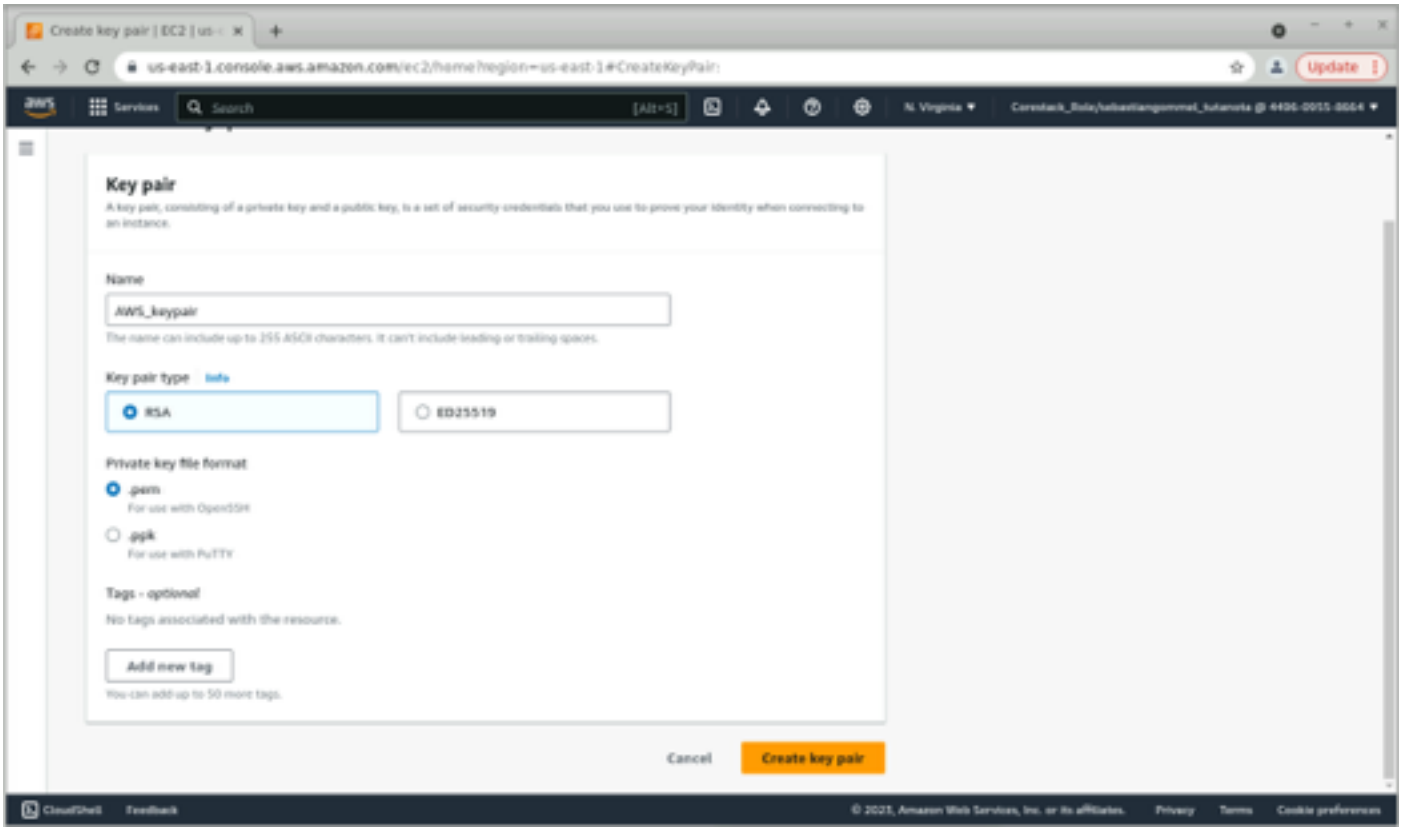
---

7)
Create key pair is selected

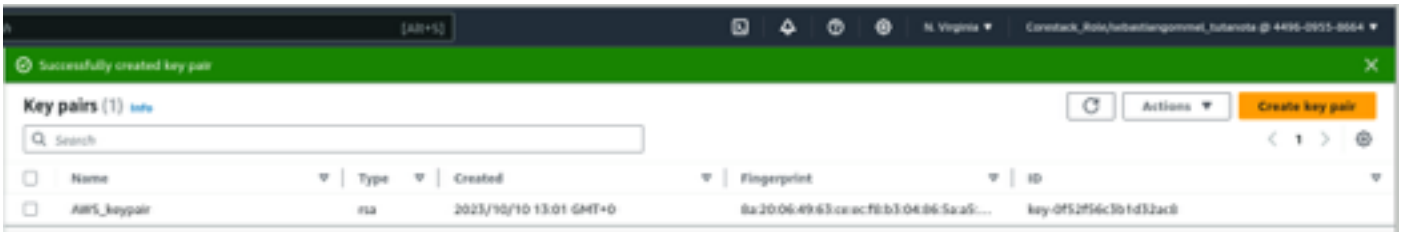


---

8)
Key pair configuration is set as in picture: (name = AWS_keypair..)



The Key pair has been created and is in downloads



---

9)
Key pair is moved to the project directory “ProjectTerraform“

Code:

```
cd /home/sebastiangommel/Downloads
ls
mv AWS_keypair.pem /home/sebastiangommel/Desktop/ProjectTerraform
```

Terminal:

```
sebastiangommel@ip-172-31-33-148:~/Desktop$ cd /home/sebastiangommel/Downloads
sebastiangommel@ip-172-31-33-148:~/Downloads$ ls
AWS_keypair.pem
sebastiangommel@ip-172-31-33-148:~/Downloads$ mv AWS_keypair.pem /home/sebastiangommel/Desktop/ProjectTerraform
sebastiangommel@ip-172-31-33-148:~/Downloads$
```

Terraform configuration

Creating a .tf file

1)
Change directory to project directory “ProjectTerraform“

Code:
cd /home/sebastiangommel/Desktop/ProjectTerraform/

Terminal:

Main.tf Terraform configuration file

2)
Vim editor is used to create the main.tf file which includes the provider and region as well as the credentials for the EC2 instance.

Code:
vim main.tf

Terminal:

Furthermore the security group settings, as well as the type of resource to be executed and the commands to prepare the automation infrastructure environment for installation of Java, Python and Jenkins are executed.

3)
The following command rewrites the Terraform configuration file to a canonical format and style, which also improves the appearance of the next page with the code.

Code:
terraform fmt

Terminal:

main.tf file

Code:

```
# Configuring the cloud provider with lab provided credentials
provider "aws" {
  region     = "us-east-1"
  access_key = "ASIAWRLWRNKECLZGHYIK"
  secret_key = "t/sHjJDHlrGYPj2VYBJBSf+IAE3unURU3wXWnD5J"
  token      = "FwoGZXIvYXdzEN7//////////wEaDGts0LrZlozkSJPmKSK9AfE6qr612k6bydtmGV/
vqlxBiLeVg1nYnvY02MSwyMbGbldXSDgn8feqwsm5ak+oCFD8yy95CHC9Mw1sK4oR7s4WpefXnapP/
Qu+g+n1KuGWwLXHd80WRWNe9VWCwKAiBi9gO858xIS+LkmYbxsG2F9jOaZuK7fQPfTQr7cwk/
sAWmSmNxIl0HXssqu0s2MzQeKQF1QVRXyWt6jYX+BbBhmXWM6Pb3KL2z3NyIq6KHQ9pcLdC5zo4pgI/
Sxx1yid/JSpBjIt/MfzfQfGX9yEXkxWXj1HTYRCTlrzOJOy+/hT/fErhrWF2tPBErNKVc9H7BF/"
}
# Security group settings
variable "ingress-rules" {
  type    = list(number)
  default = [22, 8080]
}
resource "aws_security_group" "web_traffic" {
  name        = "Allow web traffic"
  description = "SSH/Jenkins inbound, everything outbound"
  dynamic "ingress" {
    iterator = port
    for_each = var.ingress-rules
    content {
      from_port   = port.value
      to_port     = port.value
      protocol    = "TCP"
      cidr_blocks = ["0.0.0.0/0"]
    }
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}



# Type of resource to be executed
resource "aws_instance" "ec2" {
  ami                    = "ami-04505e74c0741db8d"
  instance_type          = "t3.micro"
  key_name               = "AWS_keypair"
  vpc_security_group_ids = [aws_security_group.web_traffic.id]

  # Remotely execute commands to install Java, Python, Jenkins
  provisioner "remote-exec" {
    inline = [
      "sudo apt update && upgrade",
      "sudo apt install -y python3.8",
```

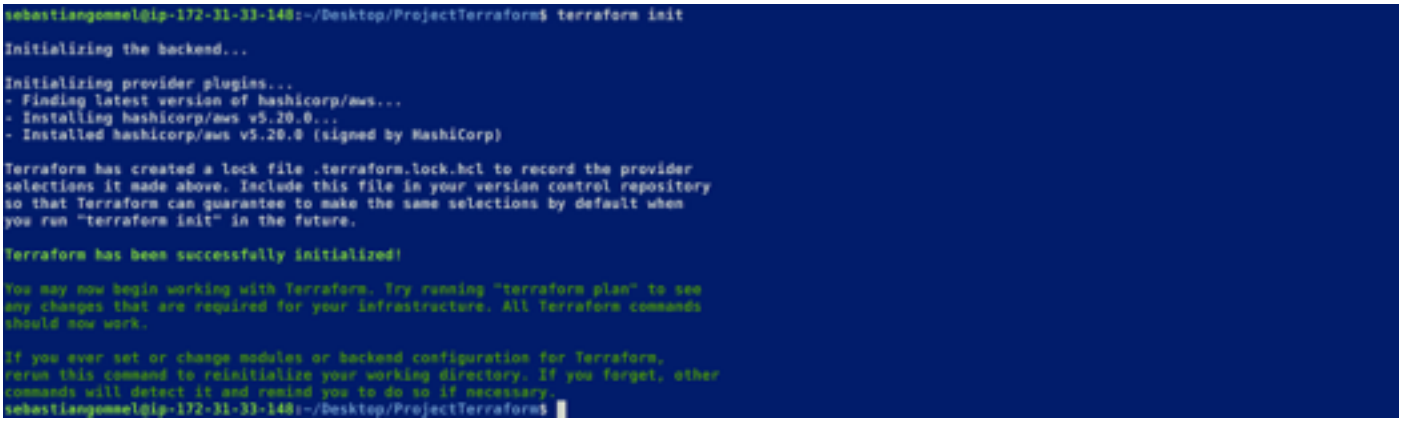
```
    "curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null",
    "echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null",
    "sudo apt-get install fontconfig java-17-openjdk",
    "sudo systemctl daemon-reload",
    "sudo apt-get install jenkins"


  ]
 }

 # Type of connection to be established
 connection {
  type        = "ssh"
  user        = "ubuntu"
  private_key = file("${path.module}/AWS_keypair.pem")
  host        = self.public_ip
 }
}
```

Terraform init

4)
Execute terraform init command

Code:
terraform init

Terminal



Terraform plan

5)
Execute terraform plan command

Code:
terraform plan

Terminal

```
sebastiangommel@ip-172-31-33-148:~/Desktop/ProjectTerraform$ 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_instance.ec2 will be created
  + resource "aws_instance" "ec2" {
      + ami                                  = "ami-04505e74c0741db8d"
      + 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)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t3.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "AWS_keypair"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
```

```
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags_all                             = (known after apply)
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)

      + capacity_reservation_specification {
          + capacity_reservation_preference = (known after apply)

          + capacity_reservation_target {
              + capacity_reservation_id                = (known after apply)
              + capacity_reservation_resource_group_arn = (known after apply)
            }
        }

      + cpu_options {
          + amd_sev_snp      = (known after apply)
          + core_count       = (known after apply)
          + threads_per_core = (known after apply)
        }

      + ebs_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + snapshot_id           = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
```

```
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }

      + enclave_options {
          + enabled = (known after apply)
        }

      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      + instance_market_options {
          + market_type = (known after apply)

          + spot_options {
              + instance_interruption_behavior = (known after apply)
              + max_price                      = (known after apply)
              + spot_instance_type             = (known after apply)
              + valid_until                    = (known after apply)
            }
        }
```

```
      + maintenance_options {
          + auto_recovery = (known after apply)
        }

      + metadata_options {
          + http_endpoint               = (known after apply)
          + http_protocol_ipv6          = (known after apply)
          + http_put_response_hop_limit = (known after apply)
          + http_tokens                 = (known after apply)
          + instance_metadata_tags      = (known after apply)
        }
```

```
      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_card_index    = (known after apply)
          + network_interface_id  = (known after apply)
        }

      + private_dns_name_options {
          + enable_resource_name_dns_a_record    = (known after apply)
          + enable_resource_name_dns_aaaa_record = (known after apply)
          + hostname_type                        = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

  # aws_security_group.web_traffic will be created
  + resource "aws_security_group" "web_traffic" {
      + arn                    = (known after apply)
      + description            = "SSH/Jenkins inbound, everything outbound"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 0
```

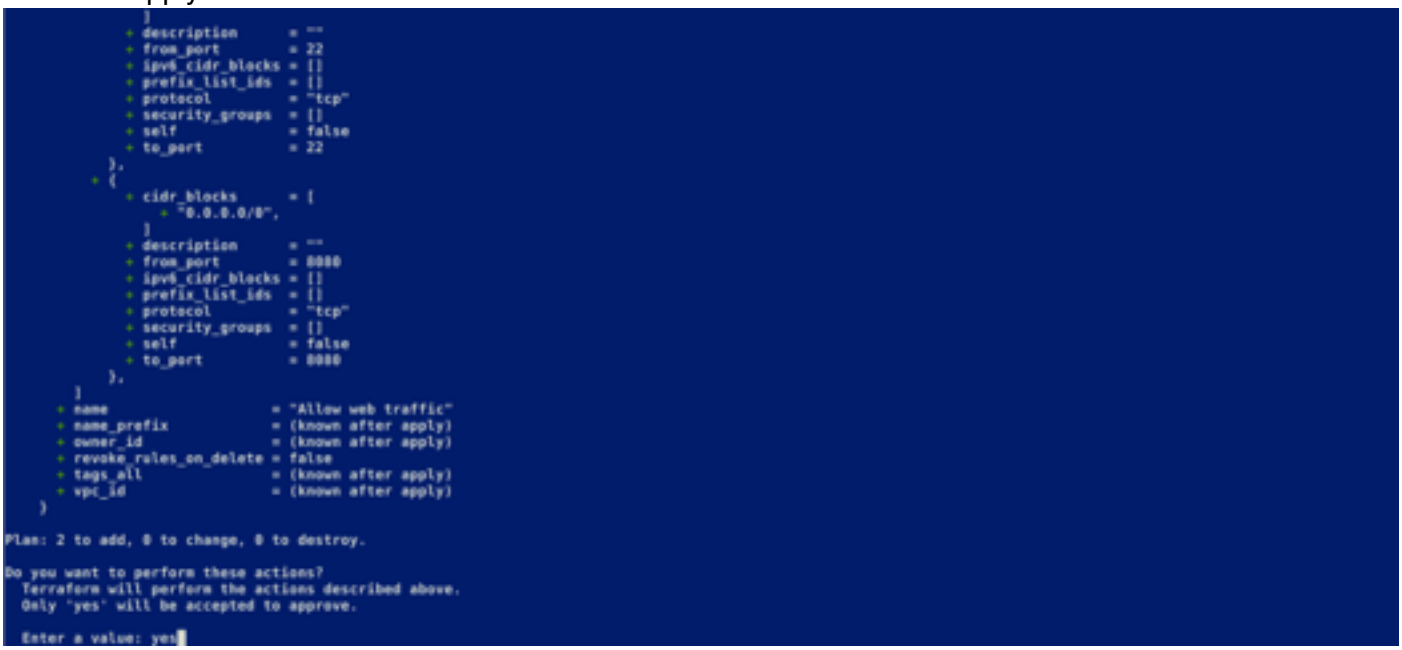
```
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 22
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 22
            },
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 8080
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 8080
            },
        ]
      + name                   = "Allow web traffic"
      + name_prefix            = (known after apply)
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags_all               = (known after apply)
      + vpc_id                 = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
sebastiangommel@ip-172-31-33-148:~/Desktop/ProjectTerraform$
```

Terraform apply

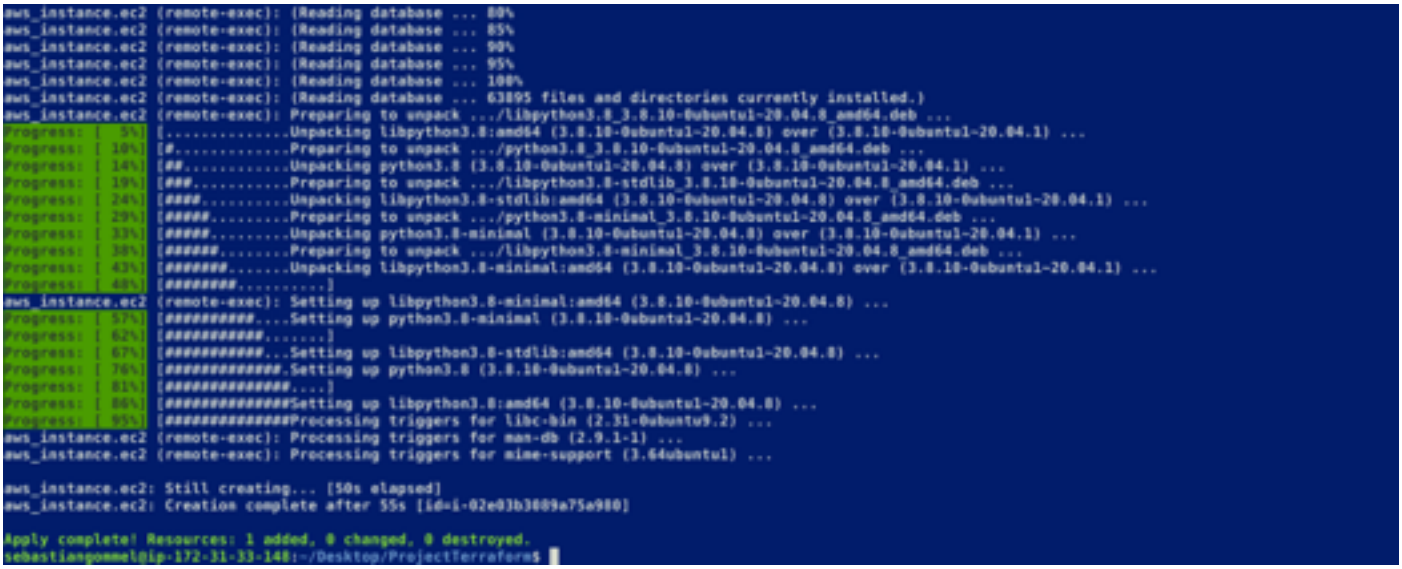
6)
Execute terraform apply command

Yes has to typed in after it is prompted to confirm that the actions will be performed.

Code:
terraform apply



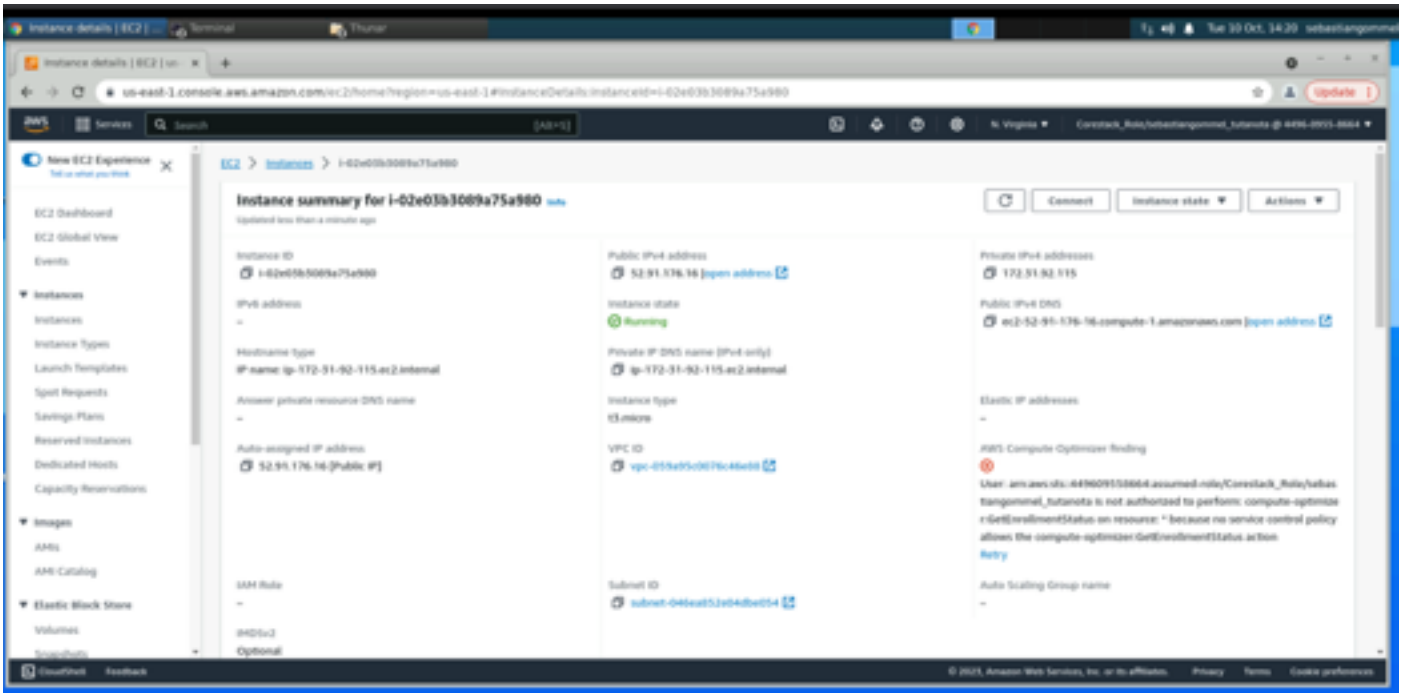
Last Output of Apply Command:

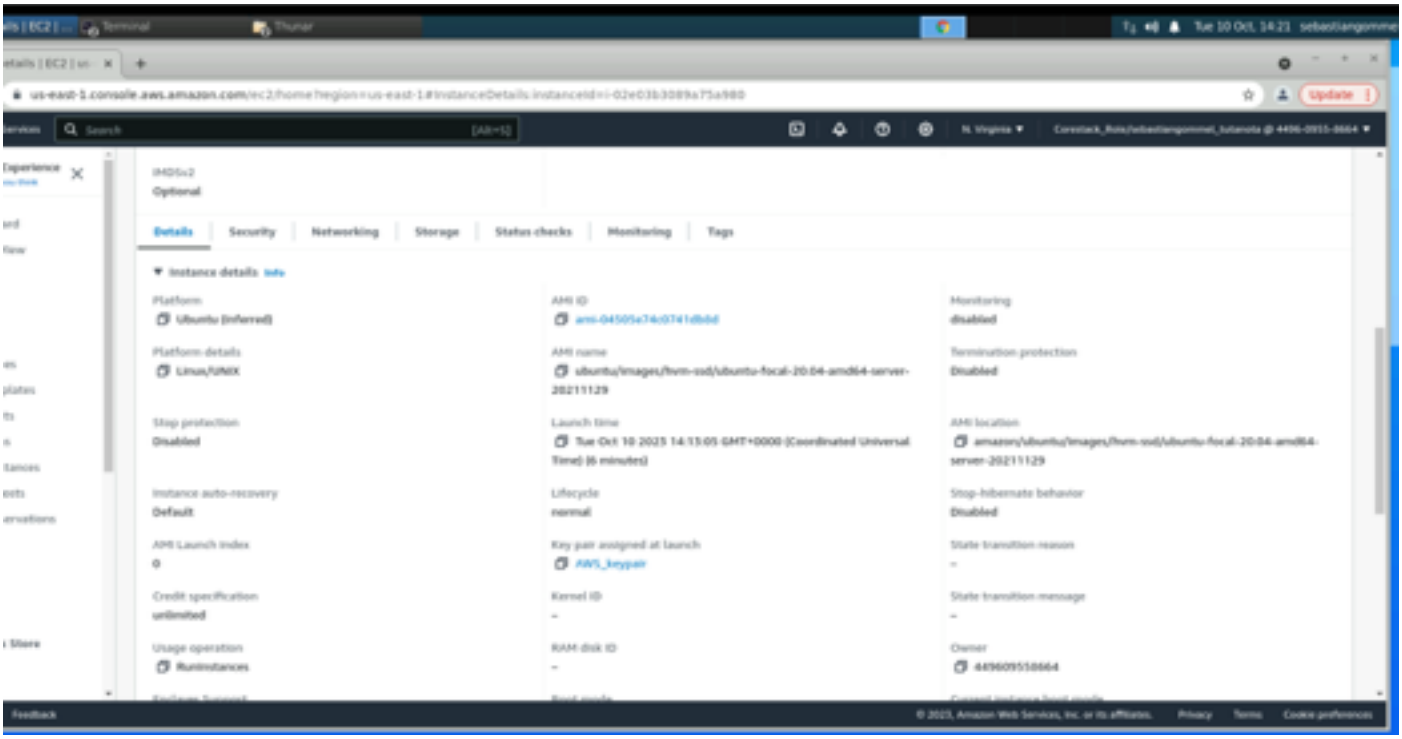


Connect to EC2 instance

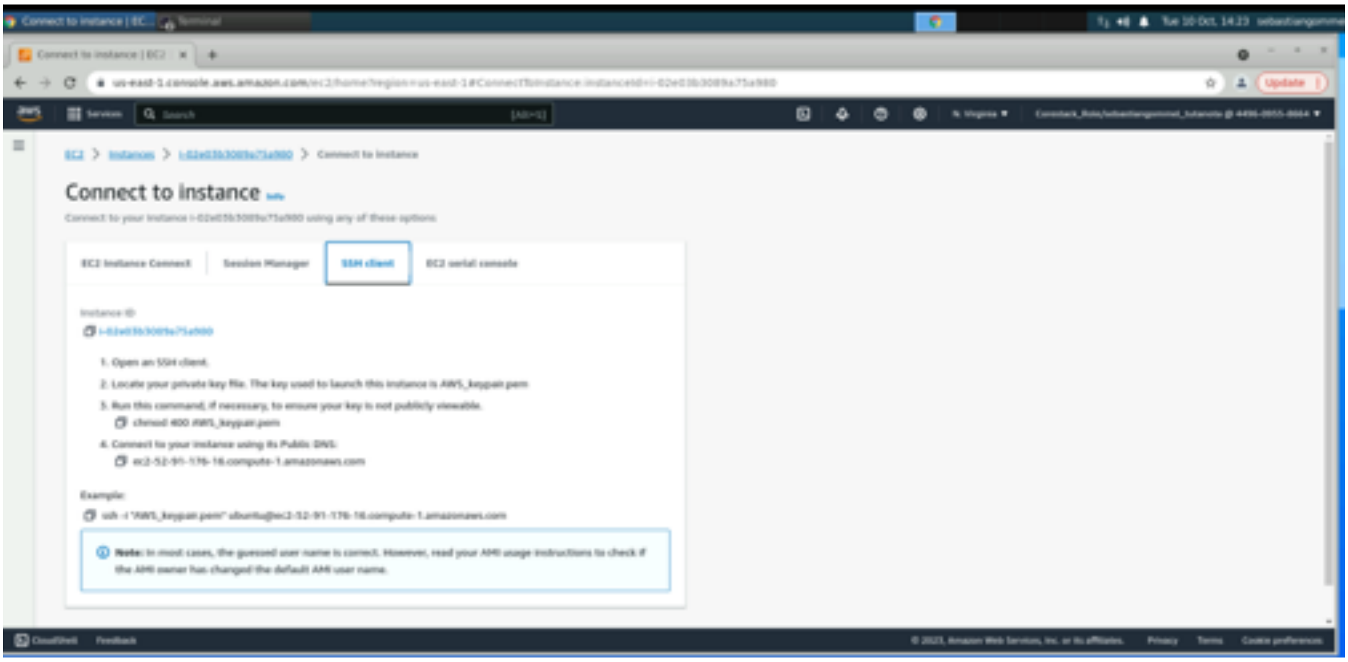
7)
Connecting to EC2 instance

Get information form AWS, by going to AWS GUI to instances and clicking on the instance ID.
Here Connect is clicked on the top right of the AWS GUI





8)
Click on SSH client to copy the relevant SSH information (after having clicked connect):



---

9)
Back to the terminal, the permission for the pem file is changed here first before running the ssh command, as indicated in the step 3 of the AWS console

Code:
chmod 400 AWS_keypair.pem

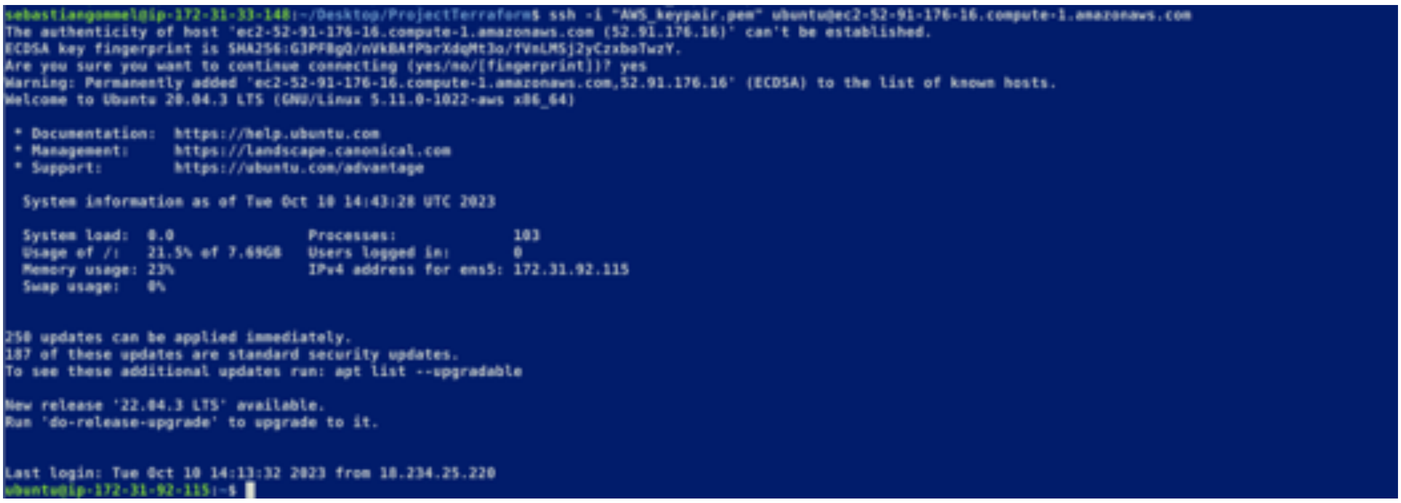
Terminal:

---

10)
The next step is to connect to the EC2 instance with the ssh information copied from AWS GUI

Code:
ssh -i "AWS_keypair.pem" ubuntu@ec2-52-91-176-16.compute-1.amazonaws.com

Terminal:



Check installed versions

11)
Code:
python3 -- version

Terminal:

---

12)
Code:
java -showversion

Terminal:

---

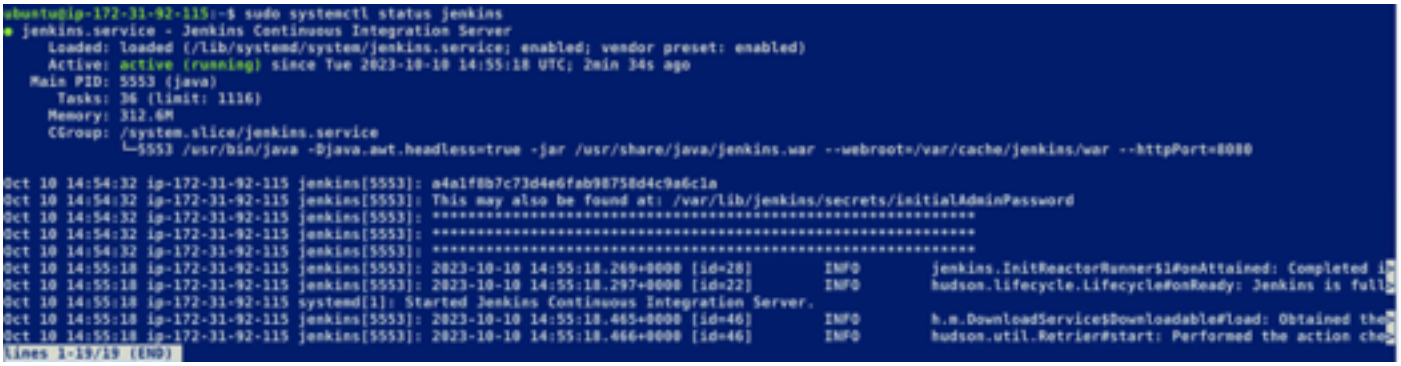
13)
Code:
jenkins - - version

Terminal:

Specifically check Jenkins status

14)
Code:
sudo systemctl status jenkins

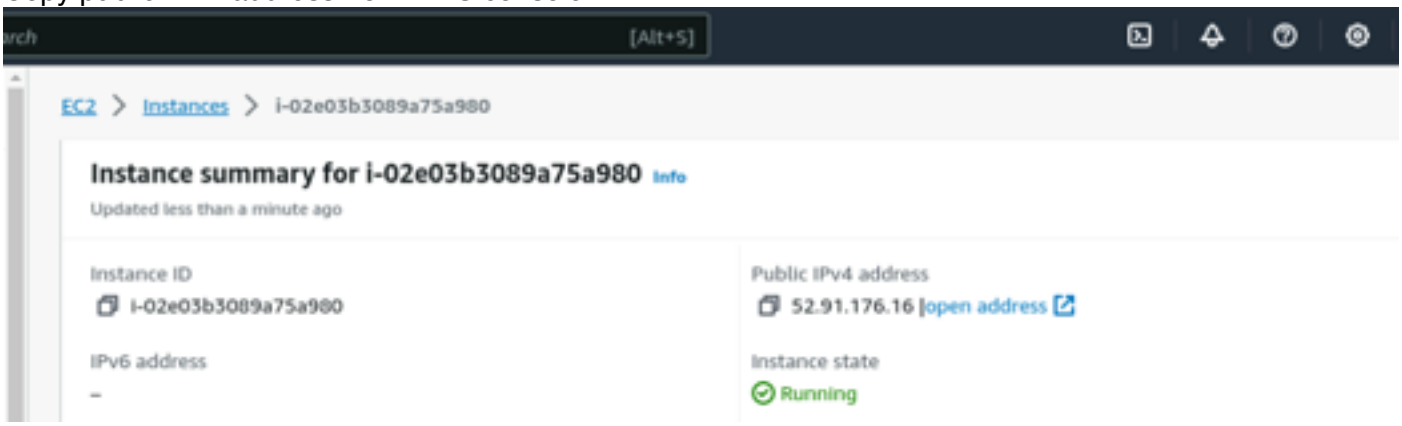
Terminal:



Accessing Jenkins

Getting Public IP

1)
Copy public IPv4 address from AWS console



—> 52.91.176.16:8080

---

2)
Ip is entered in chrome web browser with jenkins standard port 8080
52.91.176.16:8080

---

3)
At the terminal the password is found out with the following command

Code:
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

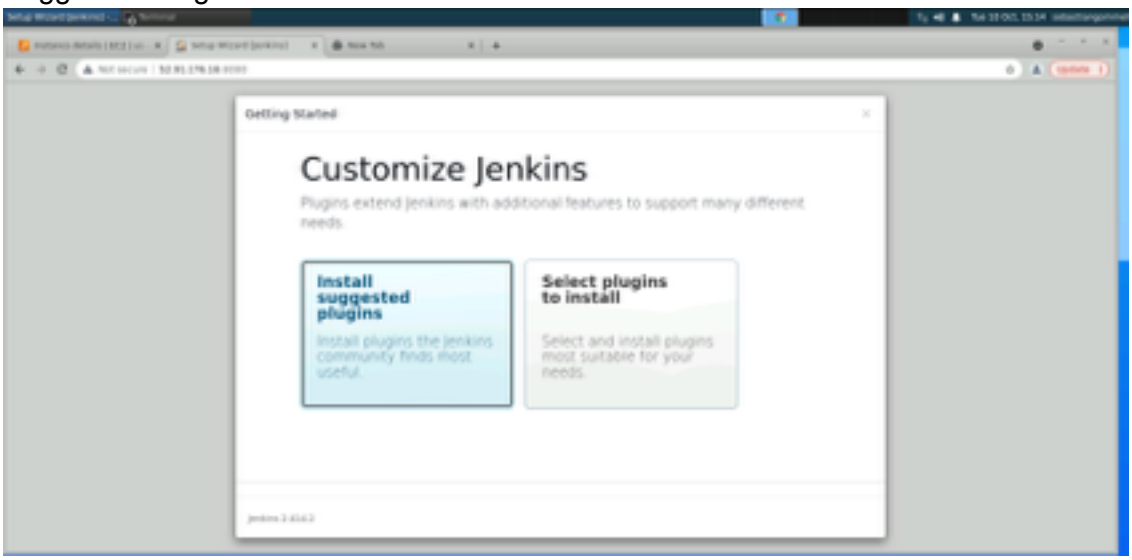
Terminal:

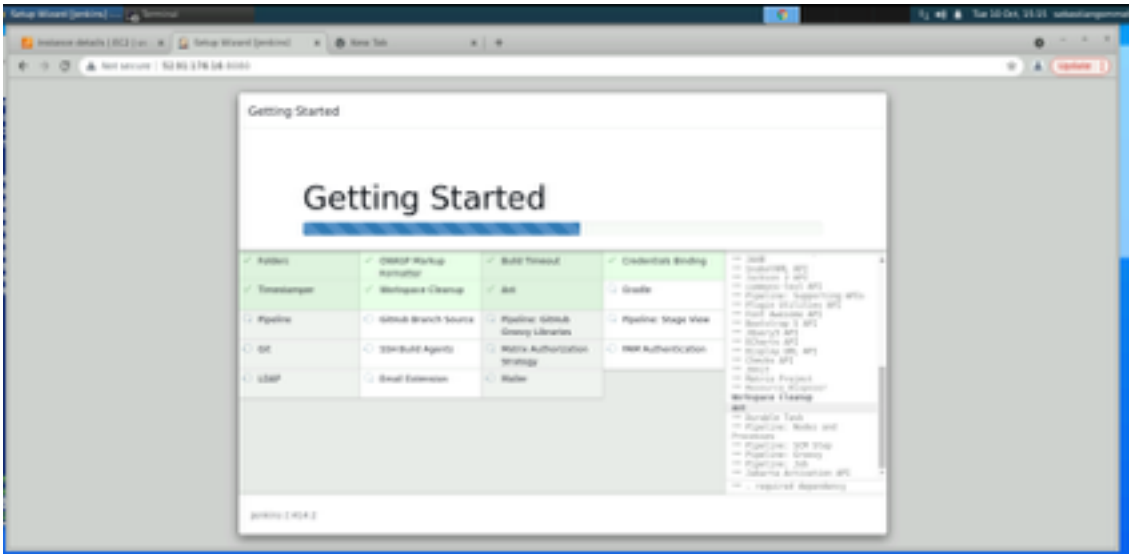
```
ubuntu@ip-172-31-92-115:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
a4a1f8b7c73d4e6fab98758d4c9a6c1a
ubuntu@ip-172-31-92-115:~$
```

a4a1f8b7c73d4e6fab98758d4c9a6c1a
This is the inital password to log in in Jenkins

Jenkins, GUI first steps

4)
After Entering the password on the unlock Jenkins Video the following page is shown.
Suggested Plugins

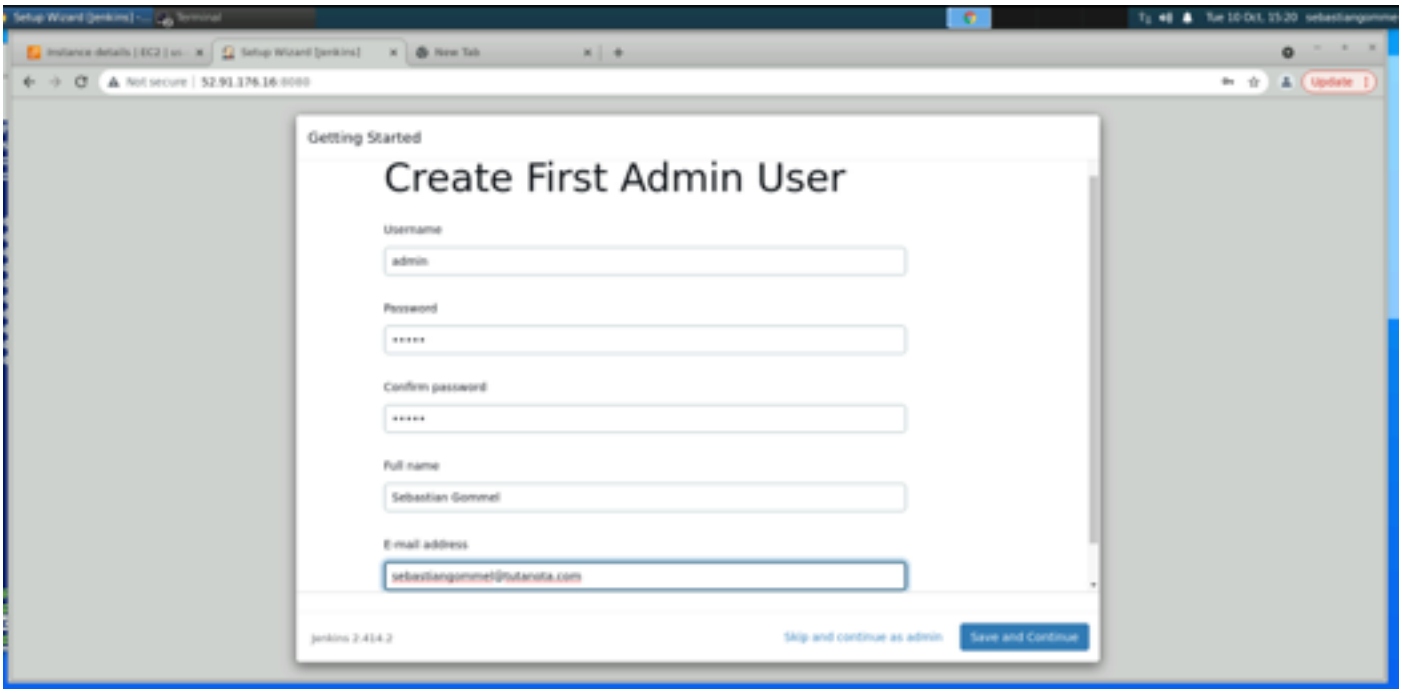




Initial default configuration is set up

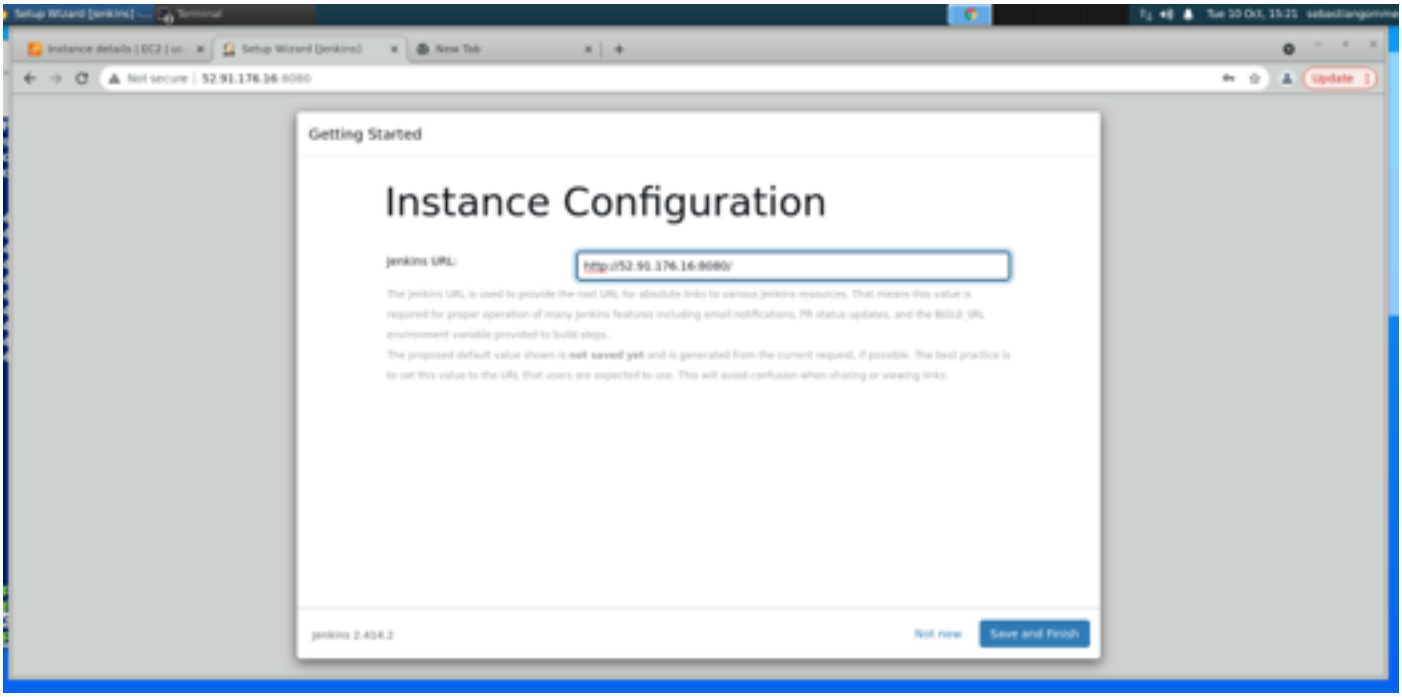
---

5)
The first User is created

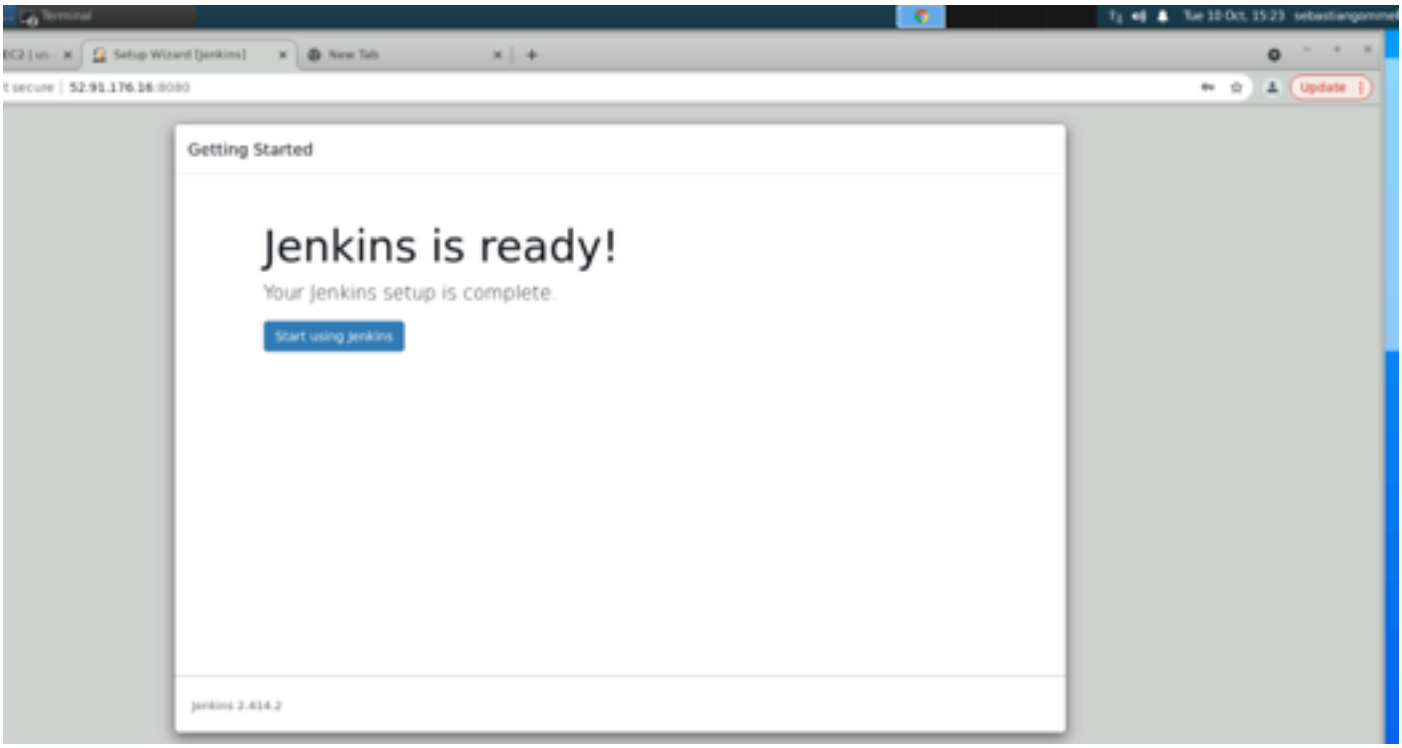


---

6)
Click Save and Finish



Jenkins setup has been completed



Jenkins is now ready to use through the EC2 instance