Automating Infrastructure using Terraform

Description:

Nowadays, infrastructure automation is critical. We tend to put the most emphasis on software development processes, but infrastructure deployment strategy is just as important. Infrastructure automation not only aids disaster recovery, but it also facilitates testing and development.

Your organization is adopting the DevOps methodology and in order to automate provisioning of infrastructure there's a need to setup a centralised server for Jenkins.

Terraform is a tool that allows you to provision various infrastructure components. Ansible is a platform for managing configurations and deploying applications. It means you'll use Terraform to build a virtual machine, for example, and then use Ansible to instal the necessary applications on that machine.

Considering the Organizational requirement you are asked to automate the infrastructure using Terraform first and install other required automation tools in it.

Tools required:

Terraform, AWS account with security credentials, Keypair

Expected Deliverables:

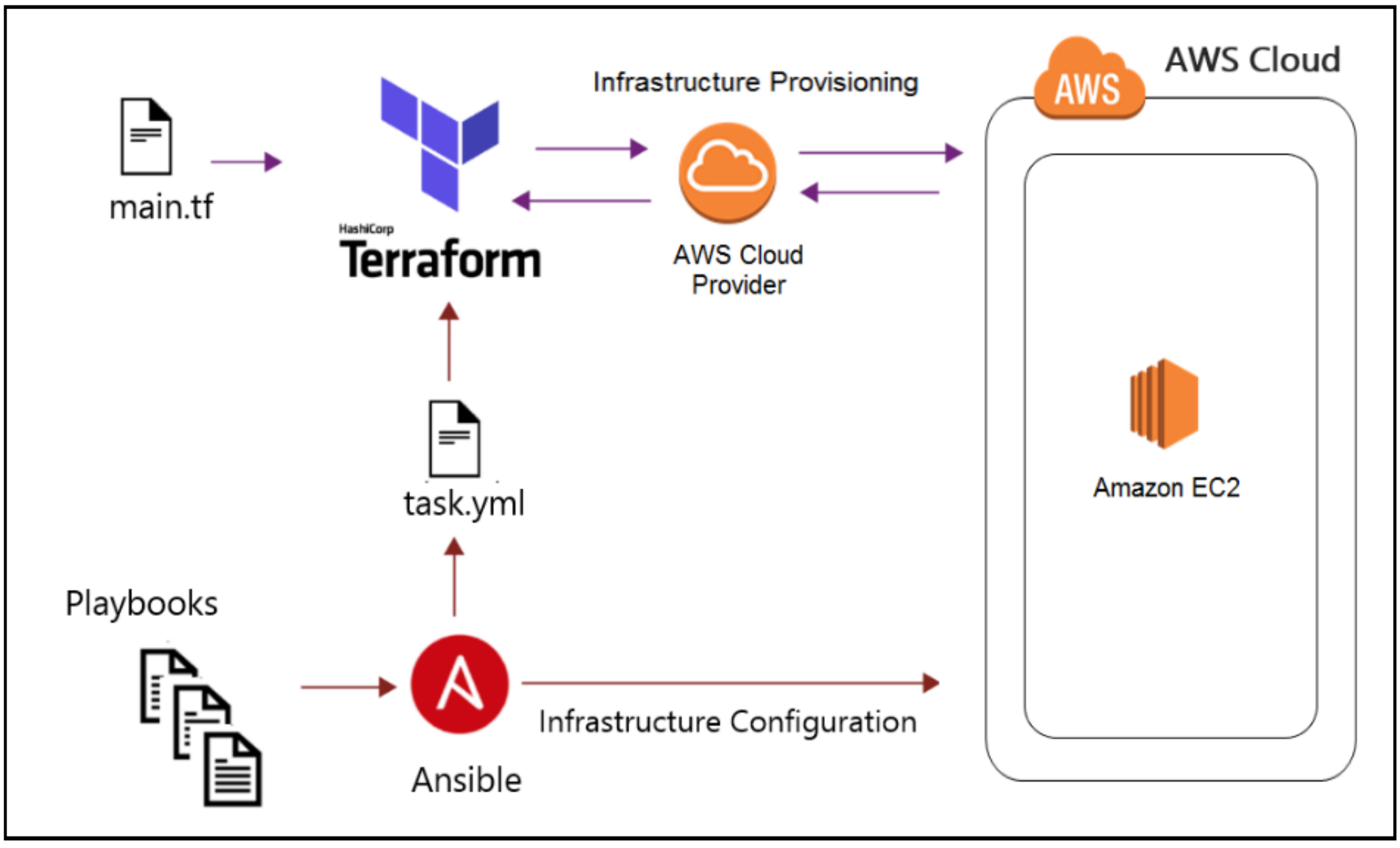
Launch an EC2 instance using Terraform

Connect to the instance

Install Jenkins, Java, and Python in the instance

Project By :- Prerana Maurya

INDEX


main.tf
HashiCorp
Terraform
Infrastructure Provisioning
AWS Cloud
Provider
AWS
AWS Cloud
Amazon EC2
task.yml
Playbooks
Ansible
Infrastructure Configuration

INTRODUCTION

For automating the infrastructure, the required softwares are Web Browser, AWS, IDE, and Terraform. For this project, I am working in an Ubuntu distribution and using a Chrome web browser.

Setting up Prerequisites

INSTALLATION OF TERRAFORM

Terraform is an open-source infrastructure-as-code software tool created by HashiCorp. Users define and provide data centre infrastructure using a declarative configuration language known as HashiCorp Configuration Language, or optionally JSON.

STEP 1 :- **Update the machine**

Update your system with the below command

```
sudo apt update -y
```

```
root@ip-172-31-88-98:/home/ubuntu# sudo apt update -y
```

STEP 2 :- **Open terminal and install Terraform package**

Ensure that your system is up to date, and you have the gnupg, software-properties-common, and curl packages installed. You will use these packages to verify HashiCorp's GPG signature, and install HashiCorp's Debian package repository. [U can prefer to the HashiCorp's official documentation]

Use the below command

```
wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor |
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list
```

Now install the Terraform

Use the below command

```
sudo apt install -y terraform
```

```
ubuntu@ip-172-31-88-98:~$ sudo apt install -y terraform
```

STEP 3 :- **Verify the installation of Terraform**

Use the below command to verify the installation of Terraform.

```
terraform --version
```

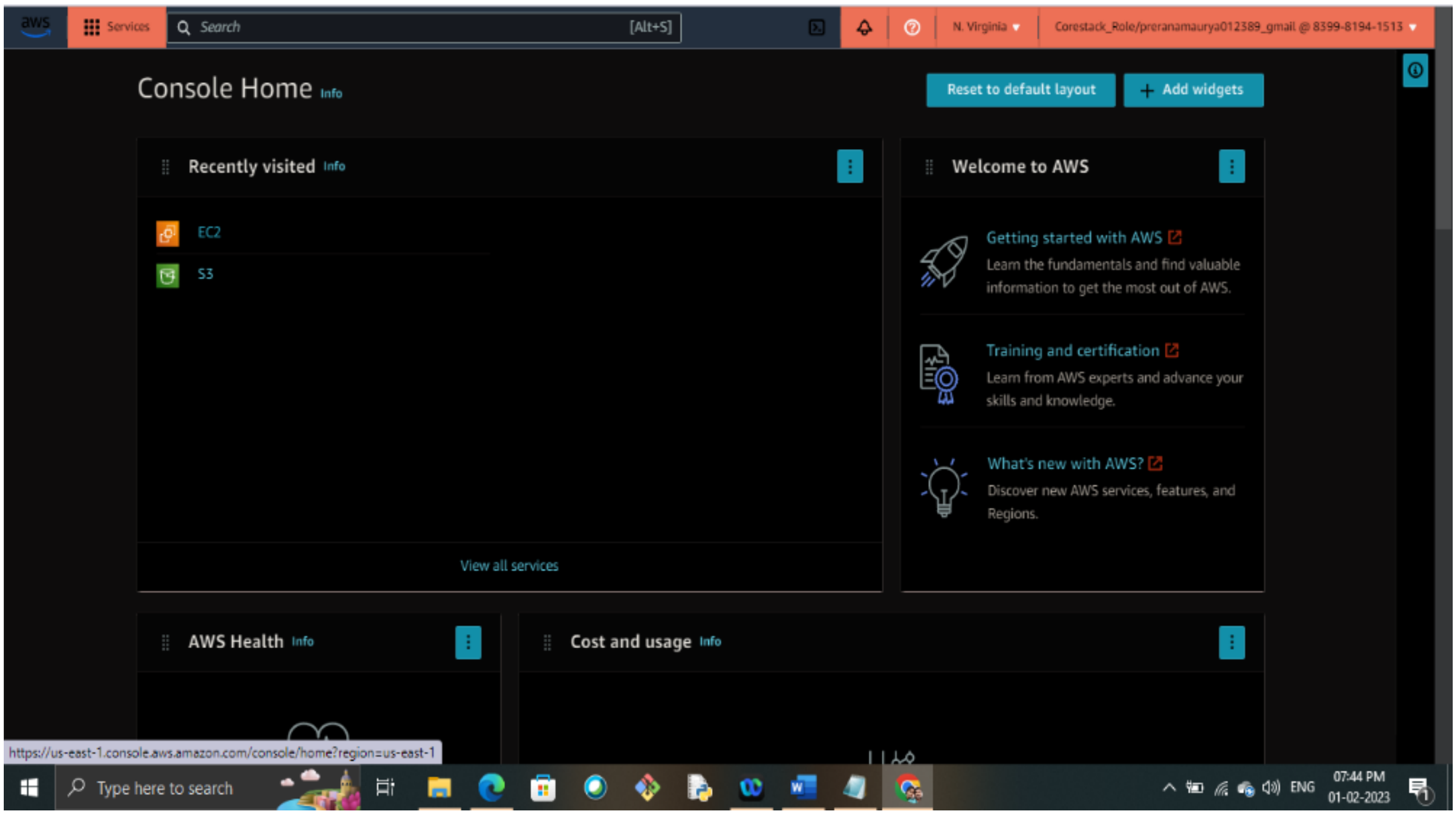
```
which terraform
```

```
ubuntu@ip-172-31-88-98:~$ terraform --version
Terraform v1.3.7
on linux_amd64
ubuntu@ip-172-31-88-98:~$ which terraform
/usr/bin/terraform
```

SETTING UP AWS

STEP 1 :- **Login to the AWS account using credentials**

Choose Sign in to the Console. If Create a new AWS account isn't visible, first choose Sign in to a different account, and then choose Create a new AWS account.

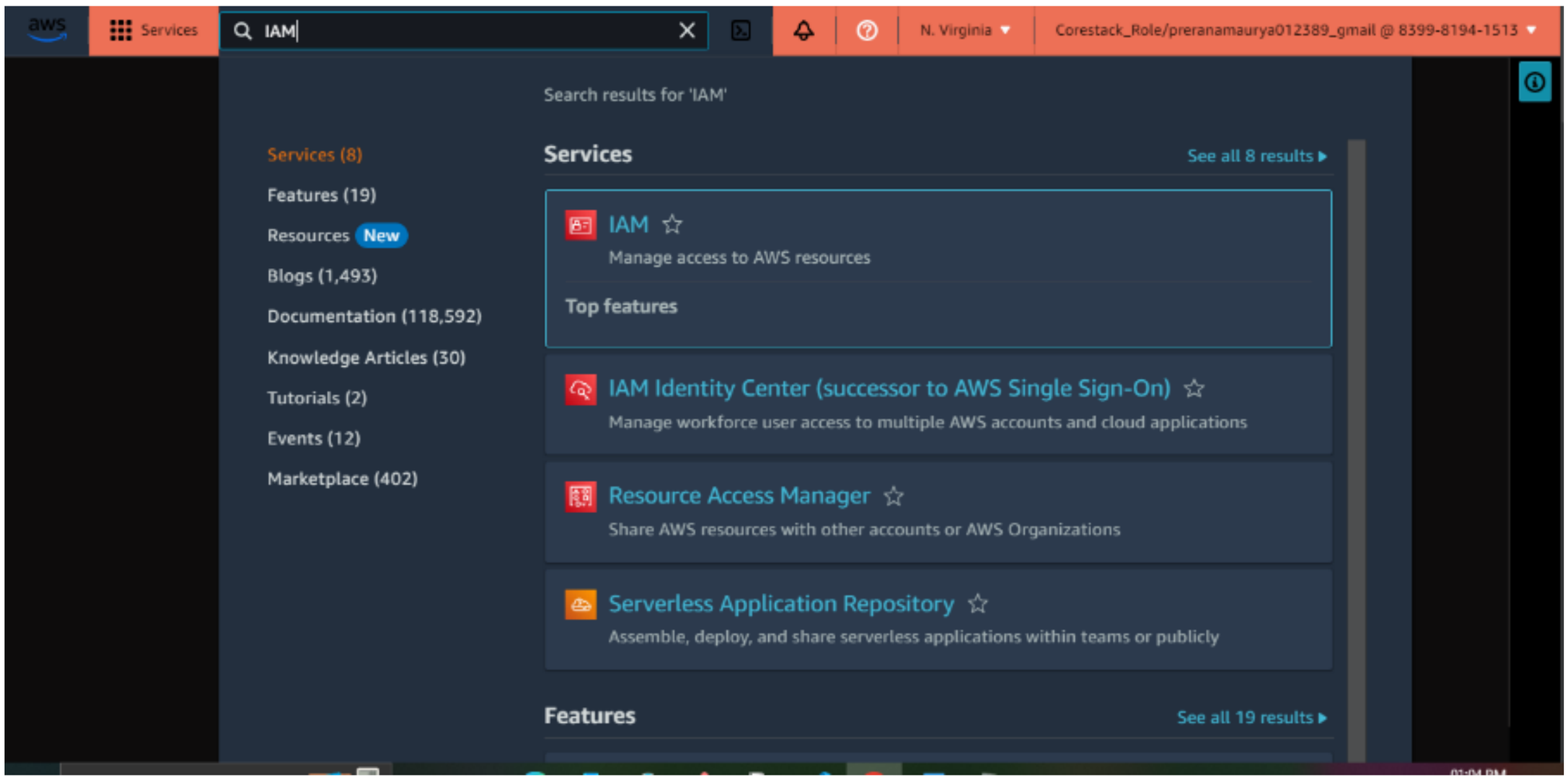


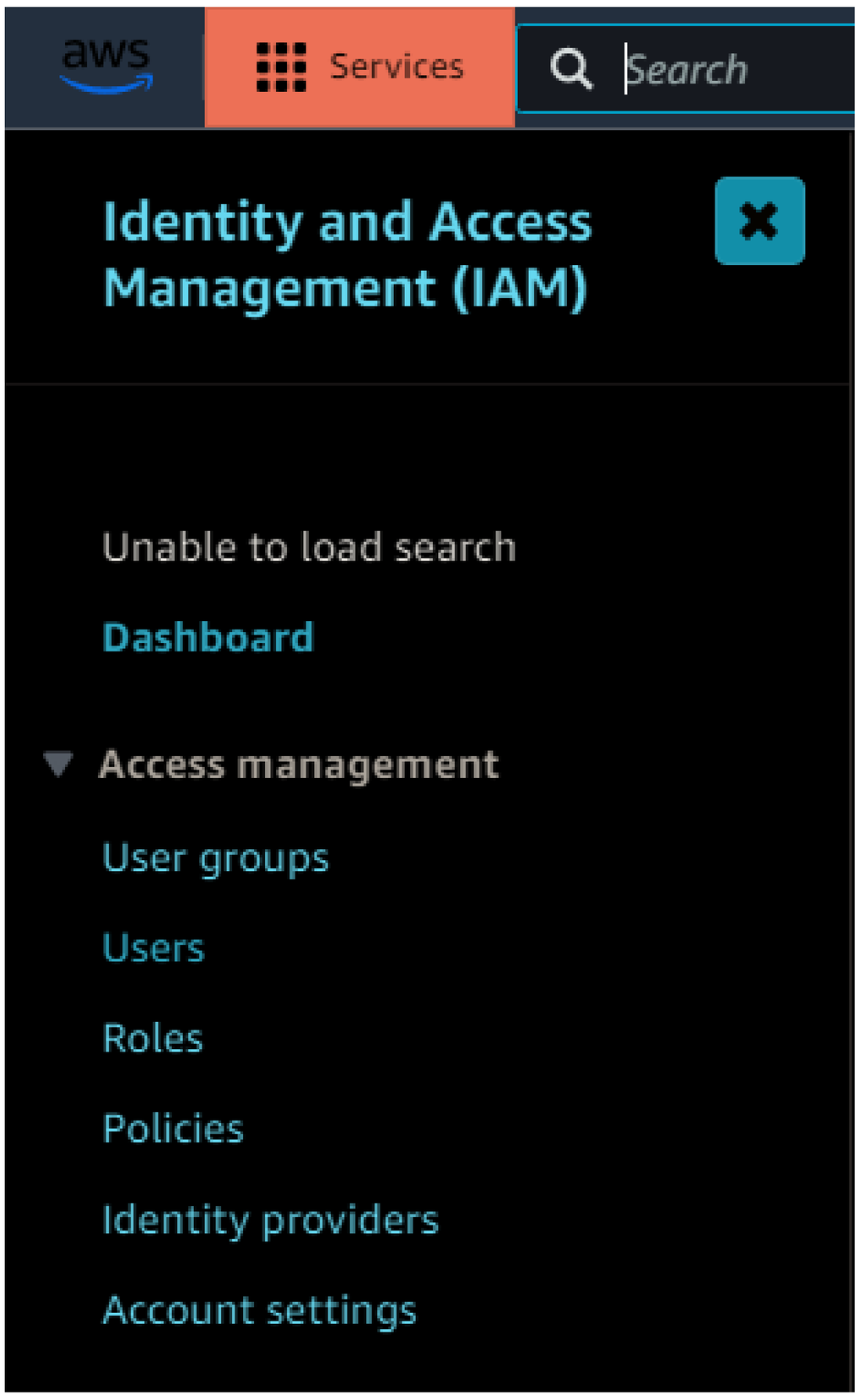
STEP 2 :- **Creating a new user for our project**

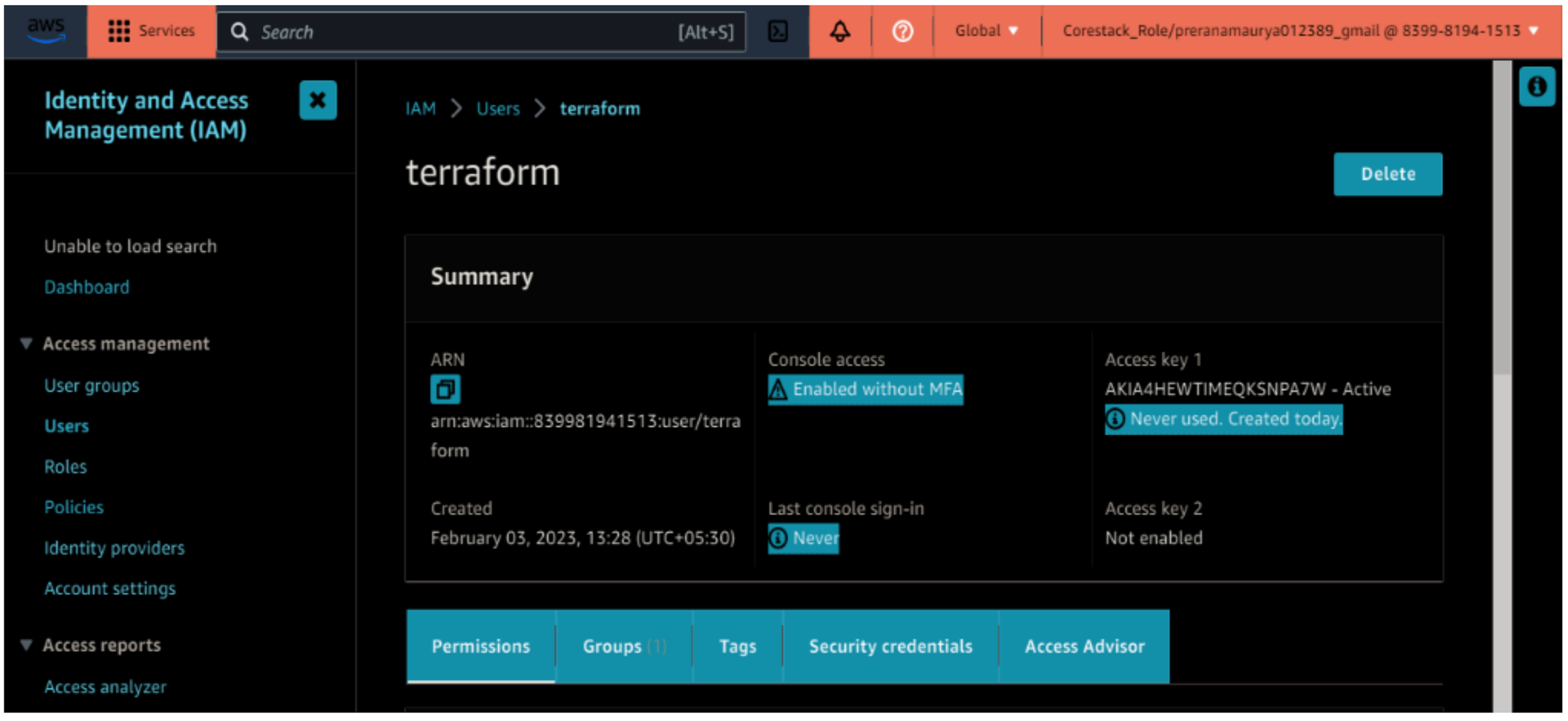
Create a new user for the project so that our credentials would be Secure .(After the completion of the project you can delete the user)

Create a key pair by following the below steps :-

1. Search for " IAM" and click the first result .
2. Click on the " users" .
3. Click on " add user" and name user " terraform" .
4. Attach the required permissions and download the credentials (.csv file).


aws
Services
IAM
N. Virginia
Corestack_Role/preranamaurya012389_gmail @ 8399-8194-1513
Search results for 'IAM'
Services (8)
Features (19)
Resources New
Blogs (1,493)
Documentation (118,592)
Knowledge Articles (30)
Tutorials (2)
Events (12)
Marketplace (402)
Services
See all 8 results
IAM
Manage access to AWS resources
Top features
IAM Identity Center (successor to AWS Single Sign-On)
Manage workforce user access to multiple AWS accounts and cloud applications
Resource Access Manager
Share AWS resources with other accounts or AWS Organizations
Serverless Application Repository
Assemble, deploy, and share serverless applications within teams or publicly
Features
See all 19 results

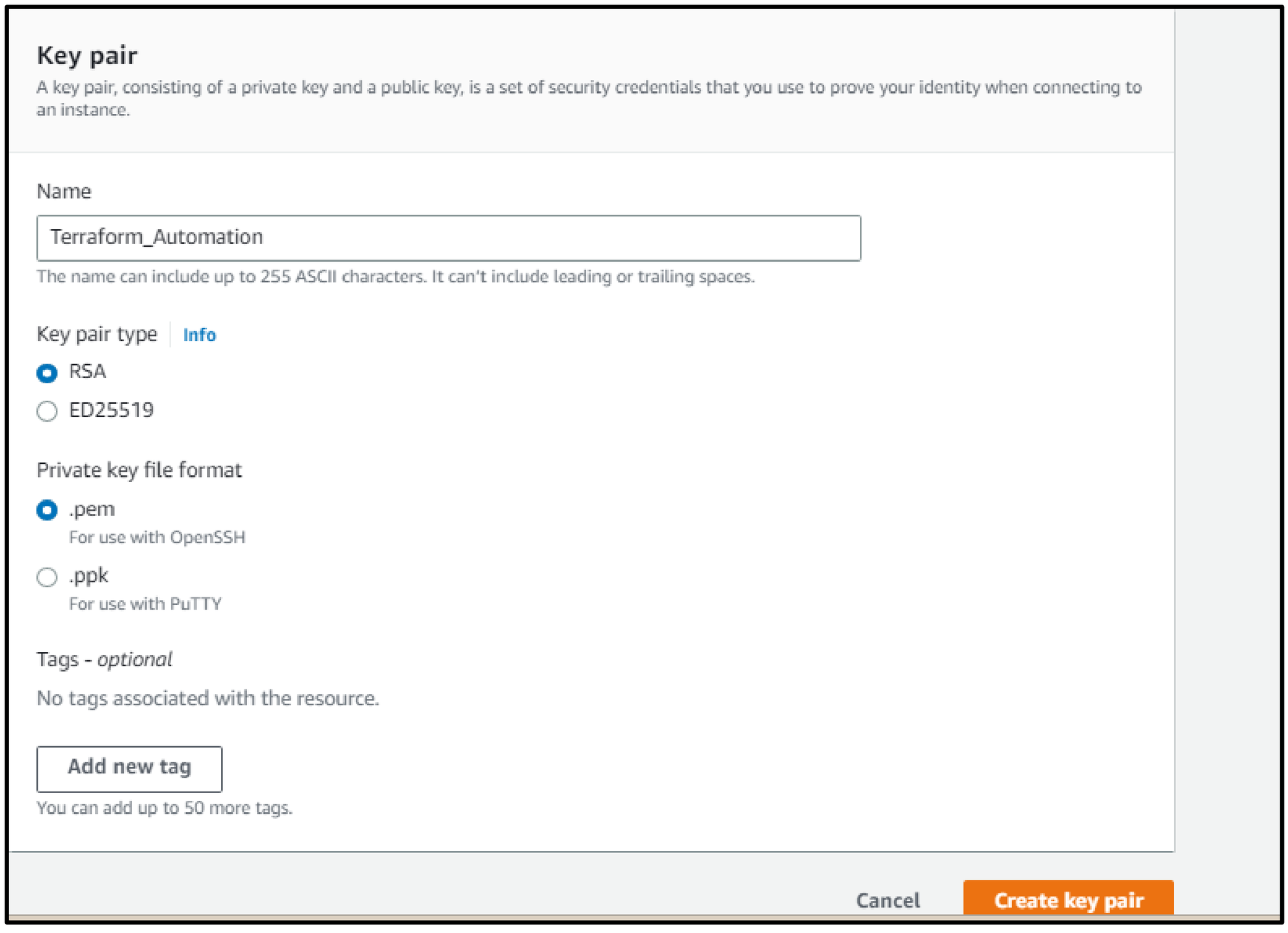

aws
Services
Search
Identity and Access Management (IAM)
Unable to load search
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings



STEP 3 :- **Creating a key pair for our project**

Create a key pair by following the below steps :-

1. Click on “ Create key pair” .
2. Enter the name of key “ Terraform_Automation” .
3. Choose “ RSA” .
4. Choose “ .pem” .
5. Click on “ Create key pair” .



WORKING WITH TERRAFORM

STEP 1 :- **Paste the private key in .ssh directory**

For SSH we will need private key. Steps to setup the private key for making it in use is as follows :-

1. Go to .ssh directory.

   cd .ssh

```
ubuntu@ip-172-31-48-228:~$ cd .ssh
ubuntu@ip-172-31-48-228:~/.ssh$
```

2. Make a .pem file " Terraform_Automation.pem" using vi editor and paste the private key content.

vi Terraform_Automation.pem

```
ubuntu@ip-172-31-48-228:~/.ssh$ vi Terraform_Automation.pem
```

3. The private key file on your local workstation should have permission set to 600 and .ssh directory should have permission set to 700.

sudo chmod 600 Terraform_Automation.pem

```
ubuntu@ip-172-31-48-228:~/.ssh$ sudo chmod 600 Terraform_Automation.pem
ubuntu@ip-172-31-48-228:~/.ssh$ ls -l Terraform_Automation.pem
-rw------- 1 ubuntu ubuntu 1675 Feb 10 14:38 Terraform_Automation.pem
```

sudo chmod 700 .ssh

```
ubuntu@ip-172-31-48-228:~$ sudo chmod 700 .ssh
ubuntu@ip-172-31-48-228:~$ ls -ld .ssh
drwx------ 2 ubuntu ubuntu 4096 Feb 10 14:38 .ssh
```

STEP 2 :- **Create a working directory**

Create a directory for our project by using below command

mkdir terraform_project

```
ubuntu@ip-172-31-88-98:~$ mkdir terraform_project
```

STEP 3 :- **Go to the project directory**

Go to the project directory " **terraform_project**" by using below command

cd terraform_project

```
ubuntu@ip-172-31-48-228:~$ cd terraform_project
ubuntu@ip-172-31-48-228:~/terraform_project$
```

STEP 4 :- **Create a inventory file**

Create a inventory file in project directory that stores the host ip address after terraform apply so that we can use it for ansible configuration.

touch inventory

```
ubuntu@ip-172-31-48-228:~/terraform_project$ touch inventory
```

STEP 5:- **Create a ansible configuration file**

Create a ansible configuration file ansible.cfg that stores configuration related to ansible.

vim ansible.cfg

```
ubuntu@ip-172-31-48-228:~/terraform_project$ vim ansible.cfg
```

In ansible.cfg file

ubuntu@ip-172-31-48-228: ~/terraform_project

```
[defaults]
inventory = ./inventory
private_key_file = /home/ubuntu/.ssh/Terraform_Automation.pem

~
~
~
~
~
~
```

STEP 6:- **Create a ansible playbook " task.yml"**

In the project directory create a file task.yml . This file contains the **YAML** code for Jenkins setup.

vi task.yml

```
ubuntu@ip-172-31-48-228:~/terraform_project$ vi task.yml
```

In task.yml file

```
---
- hosts: web-servers
  become: yes
  tasks:
  - name: Update apt package cache
    apt:
      update_cache: yes
      state: latest
```

```
- name: Install Java
  apt:
    name: openjdk-8-jdk
    state: present

- name: Add Jenkins repository
  apt_repository:
    repo: "deb https://pkg.jenkins.io/debian binary/"
    state: present

- name: Import Jenkins GPG Key
  apt_key:
      url: "https://pkg.jenkins.io/debian/jenkins.io.key"
      state: present

- name: Install Jenkins
  apt:
    name: jenkins
    state: present
```

ubuntu@ip-172-31-48-228: ~/terraform_project

```
---
- hosts: web-servers
  become: yes
  tasks:
  - name: Update apt package cache
    apt:
      update_cache: yes
      state: latest

  - name: Install Java
    apt:
      name: openjdk-8-jdk
      state: present

  - name: Add Jenkins repository
    apt_repository:
      repo: "deb https://pkg.jenkins.io/debian binary/"
      state: present

  - name: Import Jenkins GPG Key
    apt_key:
        url: "https://pkg.jenkins.io/debian/jenkins.io.key"
        state: present

  - name: Install Jenkins
    apt:
      name: jenkins
      state: present

"task.yml" 34L, 644B                                          7,22          Top
```

STEP 7:- **Create a terraform file main.tf**

In the project directory create a file main.tf . This file contains the terraform code to automate the infrastructure.

vi main.tf

```
ubuntu@ip-172-31-48-228:~/terraform_project$ vi main.tf
```

In main.tf file

terraform{

required_providers{

```
    aws={
      source  ="hashicorp/aws"
      version ="~>4.0"
    }
}
}

#login with aws
provider "aws"{
  region     ="us-east-1"
  access_key ="AKIAVVSYG7MPDTCMD5MW"
  secret_key ="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"
}




#variables for inbound rules
variable "ingress-rules"{
  type = list(number)
  default = [22,8080,80,443]

}
```

```
#variables for outbound rules

variable "egress-rules" {

  type = list(number)


#security group

resource "aws_security_group" "webtraffic" {

  name        = "webtraffic"

  description = "Allow inbound and outbound traffic"


  dynamic "ingress" {

    iterator = port

    for_each = var.ingress-rules

    content {

      description      = "Inbound Rules"

      from_port        = port.value

      to_port          = port.value

      protocol         = "TCP"

      cidr_blocks      = ["0.0.0.0/0"]

    }

  }


dynamic "egress" {

    iterator = port
```

```
for_each = var.egress-rules
content{
    description      ="outboundRules"
    from_port        =port.value
    to_port          =port.value
        protocol         ="TCP"
        cidr_blocks      =["0.0.0.0/0"]
    }
  }
resource"aws_instance""ec2"{
  ami ="ami-00874d747dde814fa"
  instance_type ="t2.micro"
key_name ="Terraform_Automation"
vpc_security_group_ids =[aws_security_group.webtraffic.id]
tags ={
  Name ="web server"
}

#configuring the machine
provisioner"remote-exec"{
inline =[
  "sudo apt update && upgrade",
  "sudo apt install software-properties-common -y",
  "sudo add-apt-repository --yes  ppa:deadsnakes/ppa",
```

```
    "sudo apt update -y",
    "sudo apt install python2 -y",
    "sudo apt install default-jdk -y",
    "sudo wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -",
     "sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'"
    ]
    connection {
      type = "ssh"
      user = "ubuntu"
      private_key = file("/home/ubuntu/.ssh/Terraform_Automation.pem")
      host = aws_instance.ec2.public_ip
    }
  #this will store the ip address for later ansible configuration
   provisioner "local-exec" {
     command = "echo '[web-servers]' > inventory"
   }
   provisioner "local-exec" {
     command = "echo '${aws_instance.ec2.public_ip}' >> inventory"
   }
  #we will setup jenkins using ansible playbook
    provisioner "local-exec" {
```

```
command="ansible-playbook task.yml -i
/home/ubuntu/terraform_project/inventory
--private-key=/home/ubuntu/.ssh/Terraform_Automation.pem"
  }
}
```

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

#login with aws
provider "aws" {
  region     = "us-east-1"
  access_key = "AKIAVVSYG7MPDTCMD5MW"
  secret_key = "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"
}



# variables for inbound rules
variable "ingress-rules" {
  type = list(number)
  default = [ 22,8080,80,443 ]

}

# variables for outbound rules
variable "egress-rules" {
  type = list(number)
                                                                1,
```

```
#security group

resource "aws_security_group" "webtraffic" {
  name        = "webtraffic"
  description = "Allow inbound and outbound traffic"

  dynamic "ingress" {
    iterator = port
    for_each = var.ingress-rules
    content {
        description      = "Inbound Rules"
        from_port        = port.value
        to_port          = port.value
        protocol         = "TCP"
        cidr_blocks      = ["0.0.0.0/0"]
    }
  }

dynamic "egress" {
   iterator = port
   for_each = var.egress-rules
   content {
        description      = "outbound Rules"
        from_port        = port.value
        to_port          = port.value
        protocol         = "TCP"
        cidr_blocks      = ["0.0.0.0/0"]
   }
  }
                                                                  57,38
```

```
  }
}



resource "aws_instance" "ec2" {
  ami = "ami-00874d747dde814fa"
  instance_type = "t2.micro"
  key_name = "Terraform_Automation"
  vpc_security_group_ids = [aws_security_group.webtraffic.id]
  tags = {
    Name = "web server"
  }

  # configuring the machine
  provisioner "remote-exec" {
    inline = [
      "sudo apt update && upgrade",
      "sudo apt install software-properties-common -y",
      "sudo add-apt-repository --yes  ppa:deadsnakes/ppa",
      "sudo apt update -y",
      "sudo apt install python2 -y",
      "sudo apt install default-jdk -y",
      "sudo wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -",
      "sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenki
ns.list'"
    ]
    connection {
      type = "ssh"
                                                          85,99         71%
```

```
  ]
  connection {
    type = "ssh"
    user = "ubuntu"
    private_key = file("/home/ubuntu/.ssh/Terraform_Automation.pem")
    host = aws_instance.ec2.public_ip
  }

  }
 #this will store the ip address for later ansible configuration
  provisioner "local-exec" {
    command =" echo '[web-servers]' > inventory"
  }

  provisioner "local-exec" {
    command = "echo '${aws_instance.ec2.public_ip}' >> inventory"
  }
  #we will setup jenkins using ansible playbook
    provisioner "local-exec" {
    command = "ansible-playbook task.yml -i /home/ubuntu/terraform_project/inventory --private-key=/hom
e/ubuntu/.ssh/Terraform_Automation.pem"
  }

}
```

RUNNING TERRAFORM CODE

STEP 1 :- **terraform init**

Run terraform init command to initialize a existing terraform working directory.

```
terraform init
```

```
ubuntu@ip-172-31-48-228:~/terraform_project$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.53.0

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.
ubuntu@ip-172-31-48-228:~/terraform_project$
```

STEP 2:- **terraform validate**

Run terraform validate command to check the configuration is valid or not.

```
terraform validate
```

```
ubuntu@ip-172-31-48-228:~/terraform_project$ terraform validate
Success! The configuration is valid.
```

STEP 3 :- **terraform plan**

Run terraform plan command to create an execution plan.

```
terraform plan
```

```
ubuntu@ip-172-31-48-228:~/terraform_project$ 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-00874d747dde814fa"
      + 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_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "Terraform_Automation"
      + 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
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "web server"
        }
      + tags_all                             = {
          + "Name" = "web server"
        }
      + 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_target {
          + capacity_reservation_id                 = (known after apply)
          + capacity_reservation_resource_group_arn = (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)
    }
```

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

  + metadata_options {
      + http_endpoint               = (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)
```

```
      + 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.webtraffic will be created
+ resource "aws_security_group" "webtraffic" {
    + arn                         = (known after apply)
    + description                 = "Allow inbound and outbound traffic"
    + egress                      = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "outbound Rules"
            + 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      = "outbound Rules"
            + from_port        = 25
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 25
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "outbound Rules"
            + from_port        = 443
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 443
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "outbound Rules"
            + from_port        = 8080
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
```

```
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 8080
    },
  + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = "outbound Rules"
      + from_port        = 80
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 80
    },
  ]
+ id                     = (known after apply)
+ ingress                = [
    + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = "Inbound Rules"
      + 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      = "Inbound Rules"
      + from_port        = 443
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 443
    },
  + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = "Inbound Rules"
      + from_port        = 8080
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 8080
    },
```

```
    },
  + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = "Inbound Rules"
      + from_port        = 80
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 80
    },
  ]
+ name                   = "webtraffic"
+ 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.
```

STEP 4:- **terraform apply**

Run terraform apply command to apply the changes specified in the terraform configuration to the infrastructure.

```
terraform apply
```

```
ubuntu@ip-172-31-48-228:~/terraform_project$ terraform apply
```

Output that show terraform configuration of infrastructure is successfull.

```
ubuntu@ip-172-31-48-228: ~/terraform_project
aws_instance.ec2 (remote-exec): Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d i
nstead (see apt-key(8)).
aws_instance.ec2 (remote-exec): OK
aws_instance.ec2: Provisioning with 'local-exec'...
aws_instance.ec2 (local-exec): Executing: ["/bin/sh" "-c" " echo '[web-servers]' > inventory"]
aws_instance.ec2: Provisioning with 'local-exec'...
aws_instance.ec2 (local-exec): Executing: ["/bin/sh" "-c" "echo '18.210.10.7' >> inventory"]
aws_instance.ec2: Provisioning with 'local-exec'...
aws_instance.ec2 (local-exec): Executing: ["/bin/sh" "-c" "ansible-playbook task.yml -i /home/ubuntu/te
rraform_project/inventory --private-key=/home/ubuntu/.ssh/Terraform_Automation.pem"]
aws_instance.ec2 (local-exec): [WARNING]: Invalid characters were found in group names but not replaced
, use
aws_instance.ec2 (local-exec): -vvvv to see details

aws_instance.ec2 (local-exec): PLAY [web-servers] ***************************************************
********

aws_instance.ec2 (local-exec): TASK [Gathering Facts] ***********************************************
********
The authenticity of host '18.210.10.7 (18.210.10.7)' can't be established.
ED25519 key fingerprint is SHA256:MY9qnWVjYTXaqqZYol7MQo894elME0MusVkDyC9laFQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? aws_instance.ec2: Still creating..
. [2m10s elapsed]
aws_instance.ec2: Still creating... [2m20s elapsed]
yes
aws_instance.ec2: Still creating... [2m30s elapsed]
aws_instance.ec2 (local-exec): ok: [18.210.10.7]

aws_instance.ec2 (local-exec): TASK [Update apt package cache] ***************************************
```

ubuntu@ip-172-31-48-228: ~/terraform_project

```
********
aws_instance.ec2: Still creating... [3m0s elapsed]
aws_instance.ec2 (local-exec): changed: [18.210.10.7]

aws_instance.ec2 (local-exec): TASK [Import Jenkins GPG Key] ************************************************
********
aws_instance.ec2 (local-exec): ok: [18.210.10.7]

aws_instance.ec2 (local-exec): TASK [Install Jenkins] ***********************************************************
********
aws_instance.ec2: Still creating... [3m10s elapsed]
aws_instance.ec2: Still creating... [3m20s elapsed]
aws_instance.ec2: Still creating... [3m30s elapsed]
aws_instance.ec2: Still creating... [3m40s elapsed]
aws_instance.ec2: Still creating... [3m50s elapsed]
aws_instance.ec2 (local-exec): changed: [18.210.10.7]

aws_instance.ec2 (local-exec): TASK [Start Jenkins Service] *************************************************
********
aws_instance.ec2 (local-exec): ok: [18.210.10.7]

aws_instance.ec2 (local-exec): PLAY RECAP *******************************************************************
********
aws_instance.ec2 (local-exec): 18.210.10.7               : ok=7    changed=4    unreachable=0    faile
d=0    skipped=0    rescued=0    ignored=0

aws_instance.ec2: Creation complete after 4m0s [id=i-0788d9e150745375e]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-48-228:~/terraform_project$
```

VERIFYING THE CONFIGURATION

STEP 1:- **connect the instance using SSH**

Connect the ec2 instance using SSH connection in the local machine.

```
ssh -i "/home/ubuntu/.ssh/Terraform_Automation.pem"
ubuntu@ec2-18-210-10-7.compute-1.amazonaws.com
```

```
ubuntu@ip-172-31-48-228:~$ ssh -i "/home/ubuntu/.ssh/Terraform_Automation.pem" ubuntu@ec2-18-210-10-7.compute-1.amazonaws.
com
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1028-aws x86_64)

 * Documentation:  https://help.ubuntu.com


 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Feb 18 07:21:28 UTC 2023

  System load:  0.080078125       Processes:             99
  Usage of /:   41.6% of 7.57GB   Users logged in:       1
  Memory usage: 51%               IPv4 address for eth0: 172.31.60.247
  Swap usage:   0%


8 updates can be applied immediately.
To see these additional updates run: apt list --upgradable


*** System restart required ***
Last login: Sat Feb 18 07:12:55 2023 from 172.31.48.228
ubuntu@ip-172-31-60-247:~$
```

STEP 2:- **verifying of packages that are installed**

1. **JAVA**

java --version

which java

```
ubuntu@ip-172-31-60-247:~$ java --version
openjdk 11.0.17 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu222.04, mixed mode, sharing)
ubuntu@ip-172-31-60-247:~$ which java
/usr/bin/java
ubuntu@ip-172-31-60-247:~$
```

2. **PYTHON**

python2 --version

which python2

```
ubuntu@ip-172-31-60-247:~$ python2 --version
Python 2.7.18
ubuntu@ip-172-31-60-247:~$ which python2
/usr/bin/python2
ubuntu@ip-172-31-60-247:~$
```

3. **JENKINS**

jenkins --version

which jenkins

```
ubuntu@ip-172-31-60-247:~$ jenkins --version
2.391
ubuntu@ip-172-31-60-247:~$ which jenkins
/usr/bin/jenkins
ubuntu@ip-172-31-60-247:~$
```

STEP 2:- checking the status of jenkins

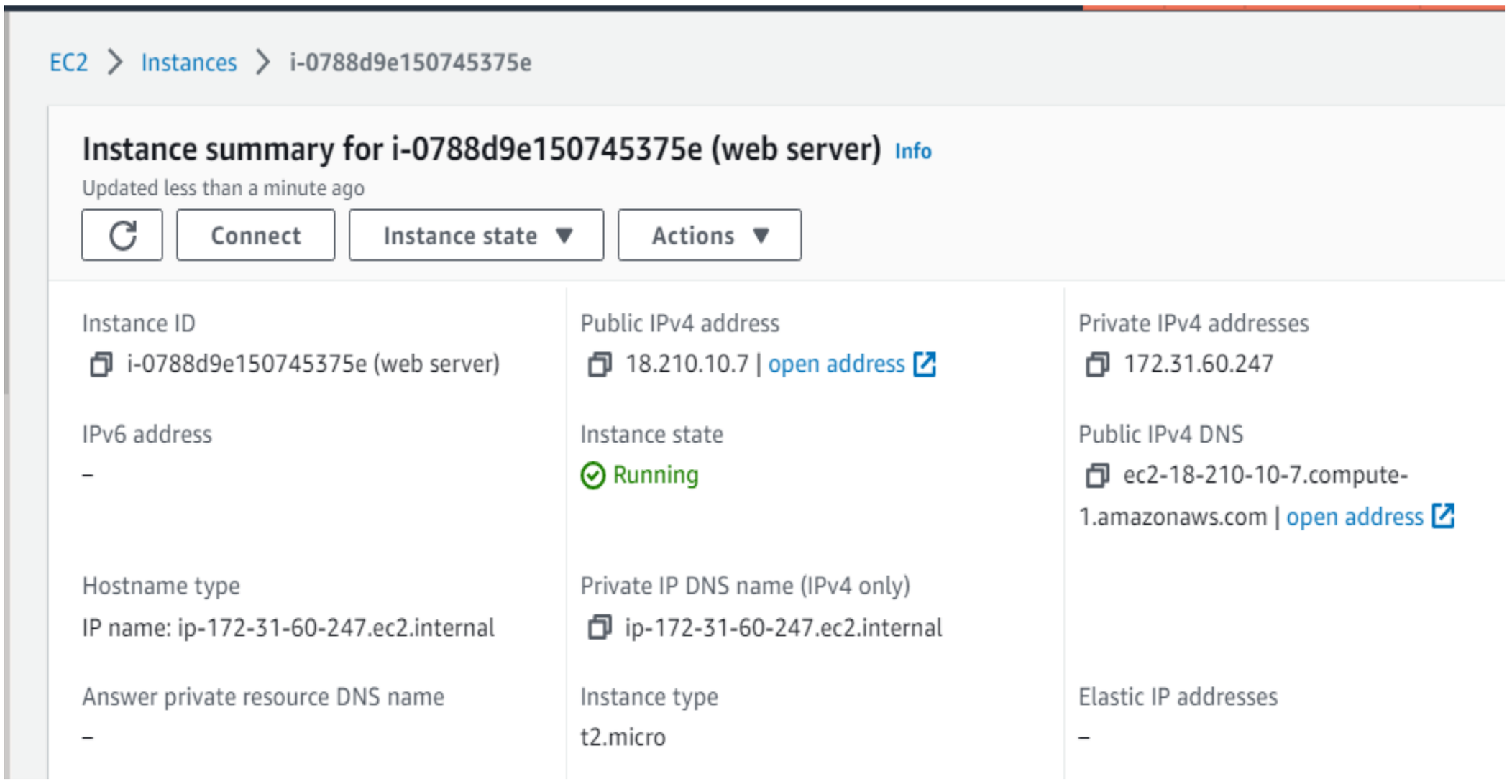
sudo systemctl status Jenkins

```
ubuntu@ip-172-31-60-247:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2023-02-18 05:40:05 UTC; 2h 1min ago
   Main PID: 7567 (java)
      Tasks: 40 (limit: 1143)
     Memory: 279.8M
        CPU: 53.765s
     CGroup: /system.slice/jenkins.service
             └─7567 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=>

Feb 18 05:39:33 ip-172-31-60-247 jenkins[7567]: b86cf8184c2144f2b3258191ba6d8ad0
Feb 18 05:39:33 ip-172-31-60-247 jenkins[7567]: This may also be found at: /var/lib/jenkins/secrets/in>
Feb 18 05:39:33 ip-172-31-60-247 jenkins[7567]: ************************************************************>
Feb 18 05:39:33 ip-172-31-60-247 jenkins[7567]: ************************************************************>
Feb 18 05:39:33 ip-172-31-60-247 jenkins[7567]: ************************************************************>
Feb 18 05:40:05 ip-172-31-60-247 jenkins[7567]: 2023-02-18 05:40:05.194+0000 [id=29]        INFO      >
Feb 18 05:40:05 ip-172-31-60-247 jenkins[7567]: 2023-02-18 05:40:05.228+0000 [id=22]        INFO      >
Feb 18 05:40:05 ip-172-31-60-247 systemd[1]: Started Jenkins Continuous Integration Server.
Feb 18 05:40:05 ip-172-31-60-247 jenkins[7567]: 2023-02-18 05:40:05.361+0000 [id=44]        INFO      >
Feb 18 05:40:05 ip-172-31-60-247 jenkins[7567]: 2023-02-18 05:40:05.362+0000 [id=44]        INFO      >
lines 1-20/20 (END)
```

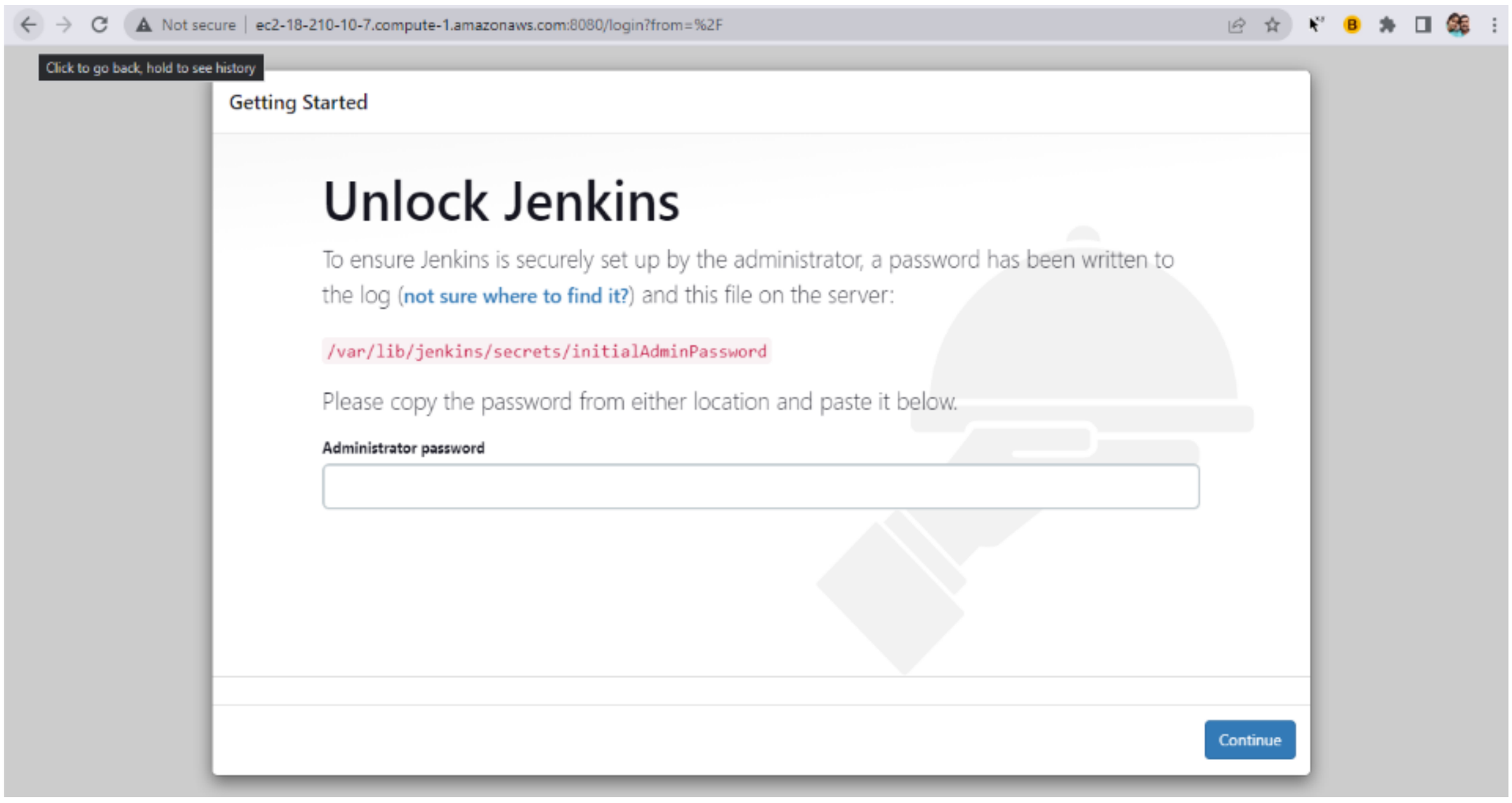
STEP 2:- Access Jenkins and continue with the installation

Find the public IP from AWS EC2 instance



Go to chrome and enter the below command

http://ec2-18-210-10-7.compute-1.amazonaws.com:8080/



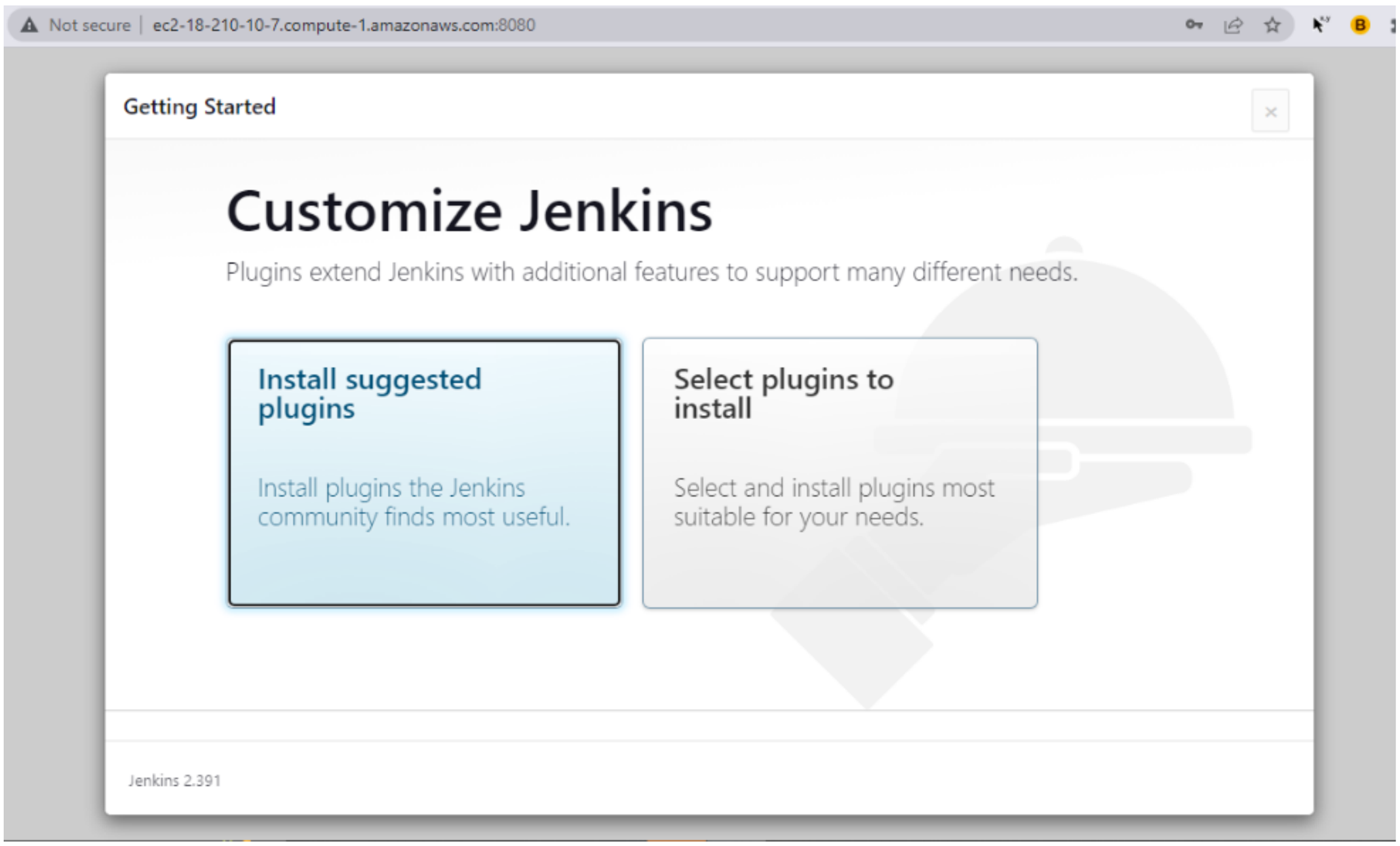
Go back to the terminal and enter the following command

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

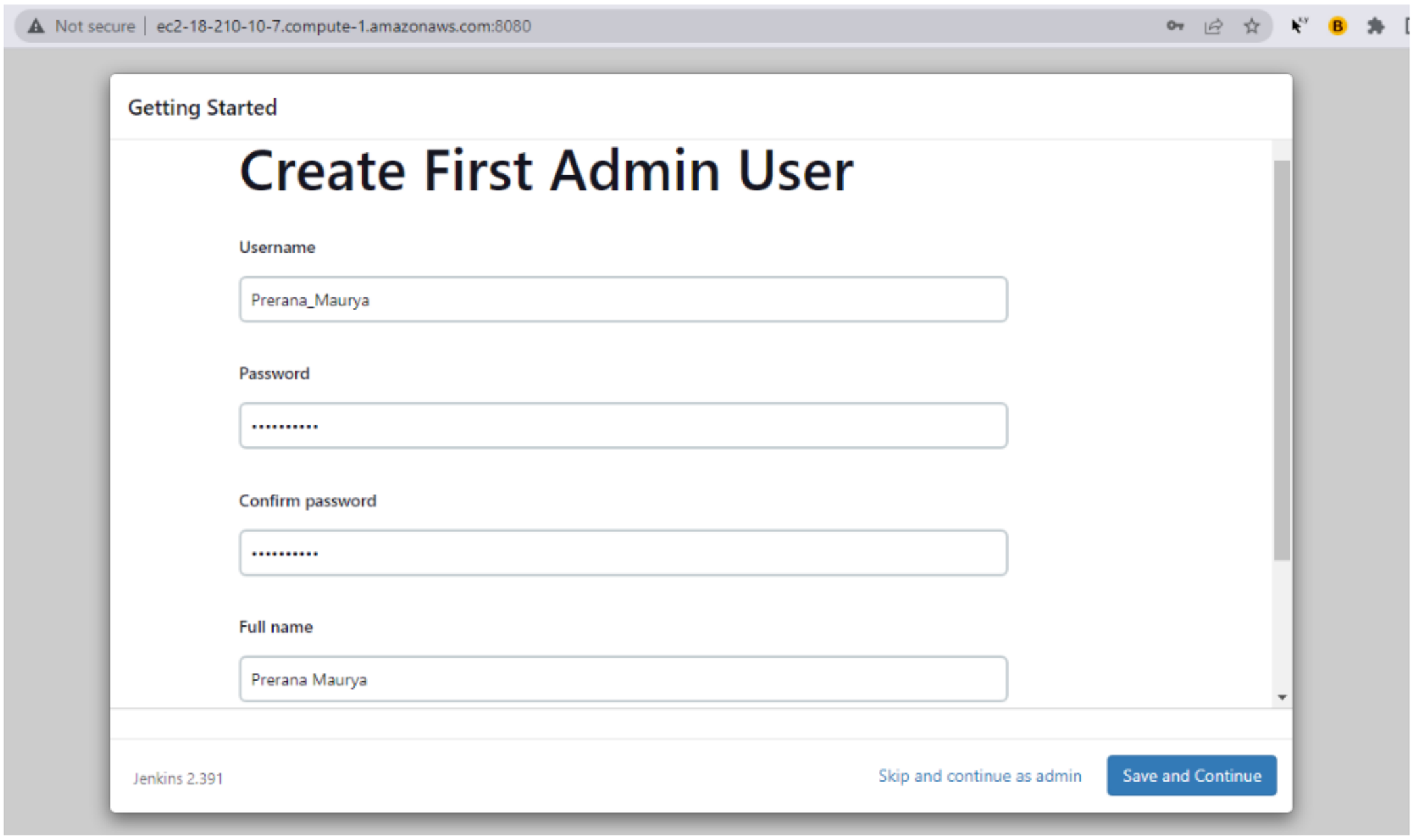
```
ubuntu@ip-172-31-60-247:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
b86cf8184c2144f2b3258191ba6d8ad0
ubuntu@ip-172-31-60-247:~$
```

Copy the above password and enter it in Jenkins to continue

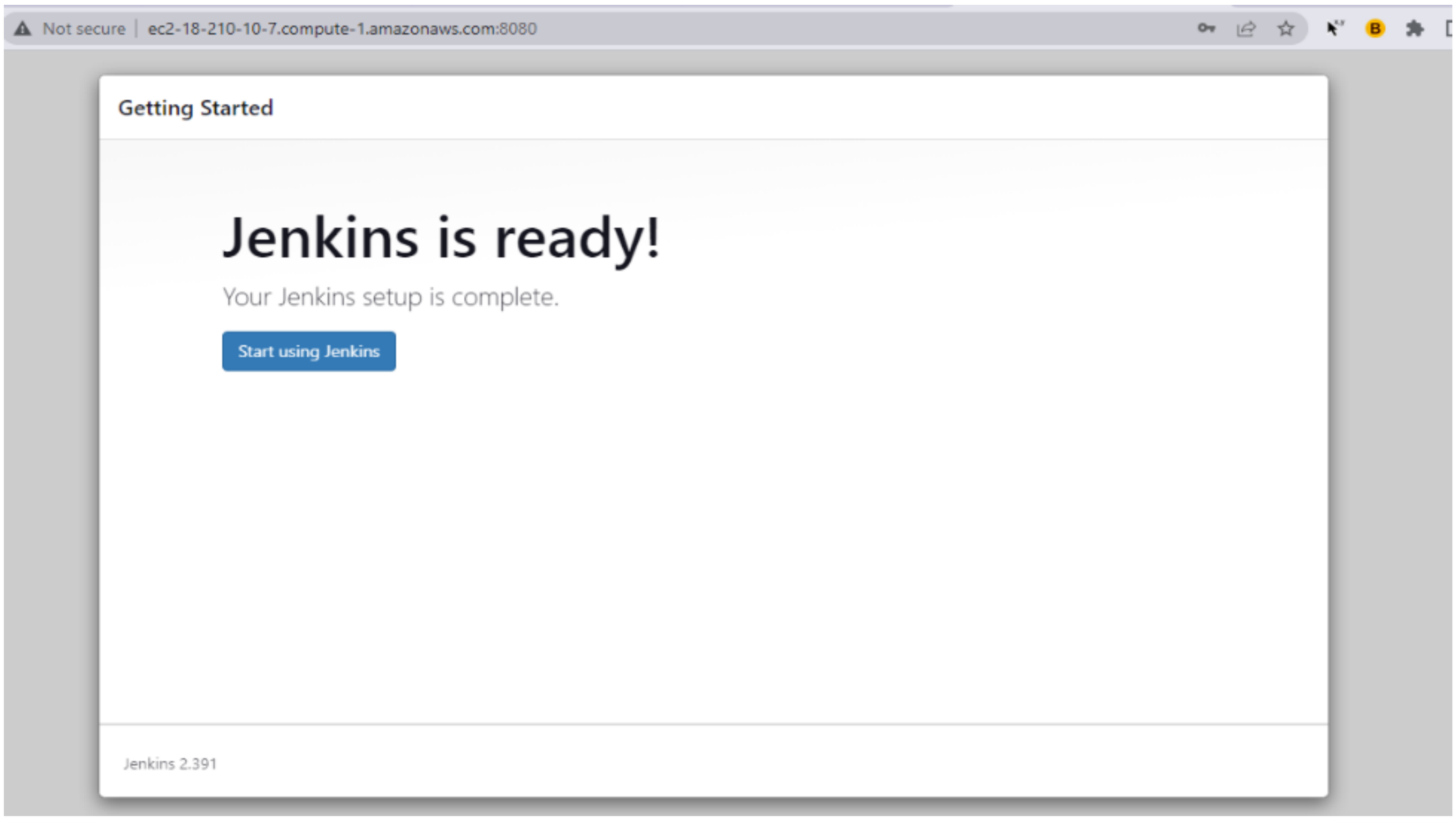
Now select “install suggested plugins”



Enter your information in the next screen to continue.



Click Save and Finish.



You are all set! Congratulations!