Course End Project - Automating Infrastructure using Terraform

Problem Statement:

Use Terraform to build a virtual machine and install other required automation tools in it.

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

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

Solution:

Step 1: Terraform Core is installed on the Lab VM:

terraform --version

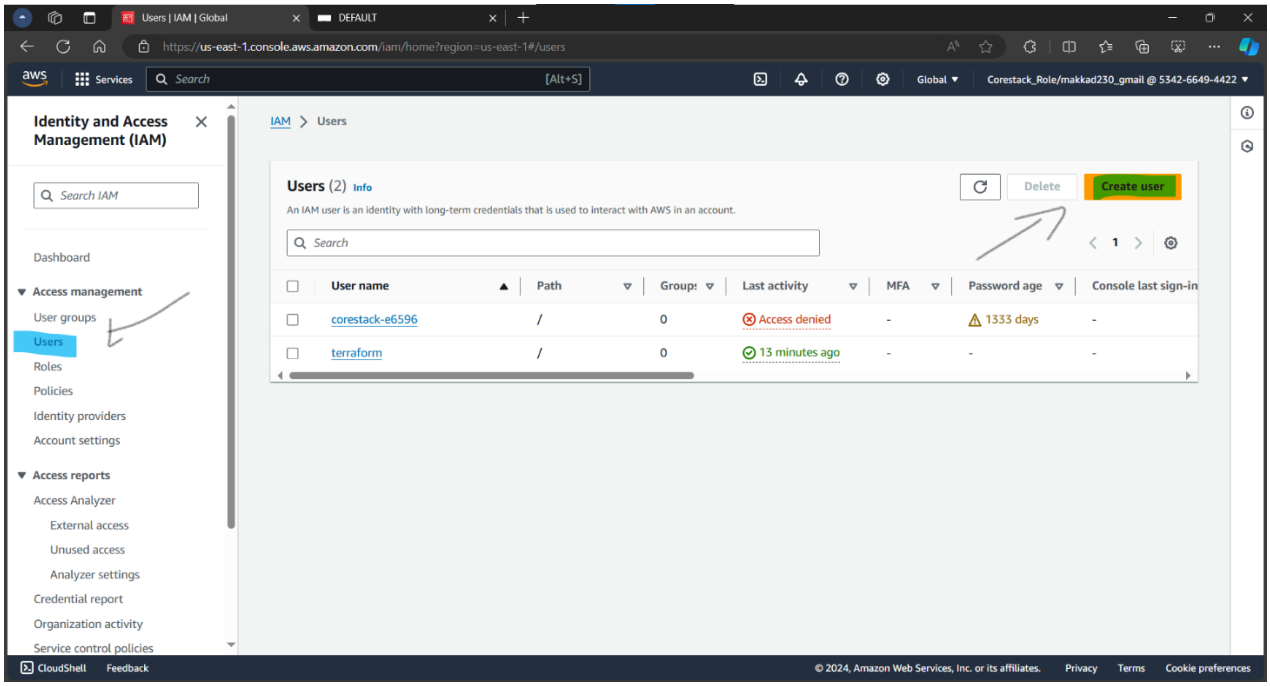
```
File  Edit  View  Search  Terminal  Help
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$ terraform --version
Terraform v1.8.5
on linux_amd64
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$
```

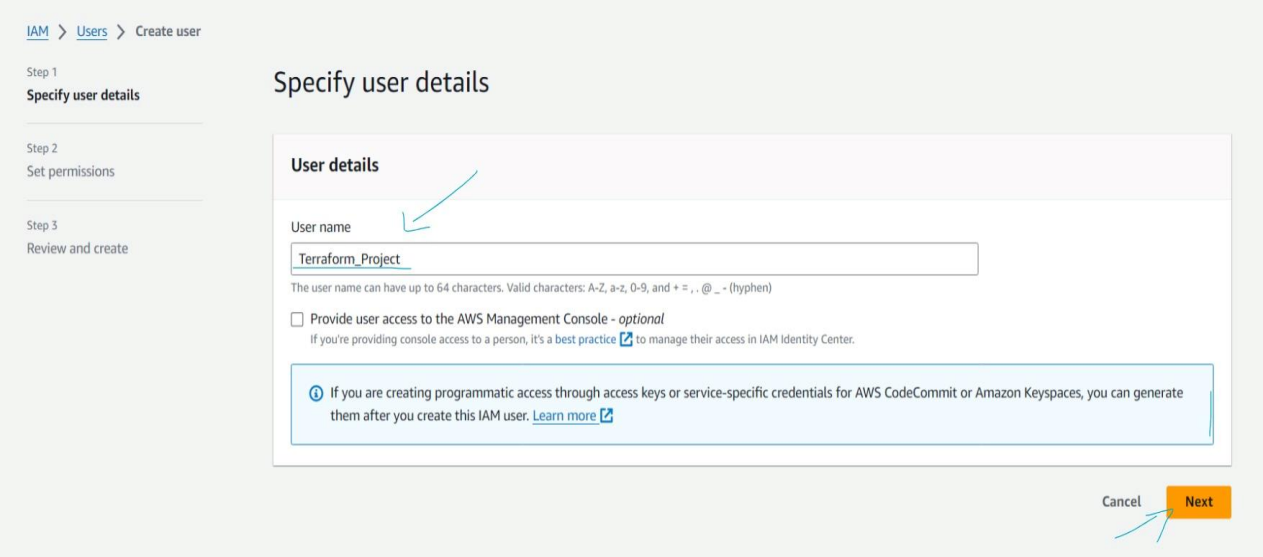
Step 2: Create AWS user and security credentials:

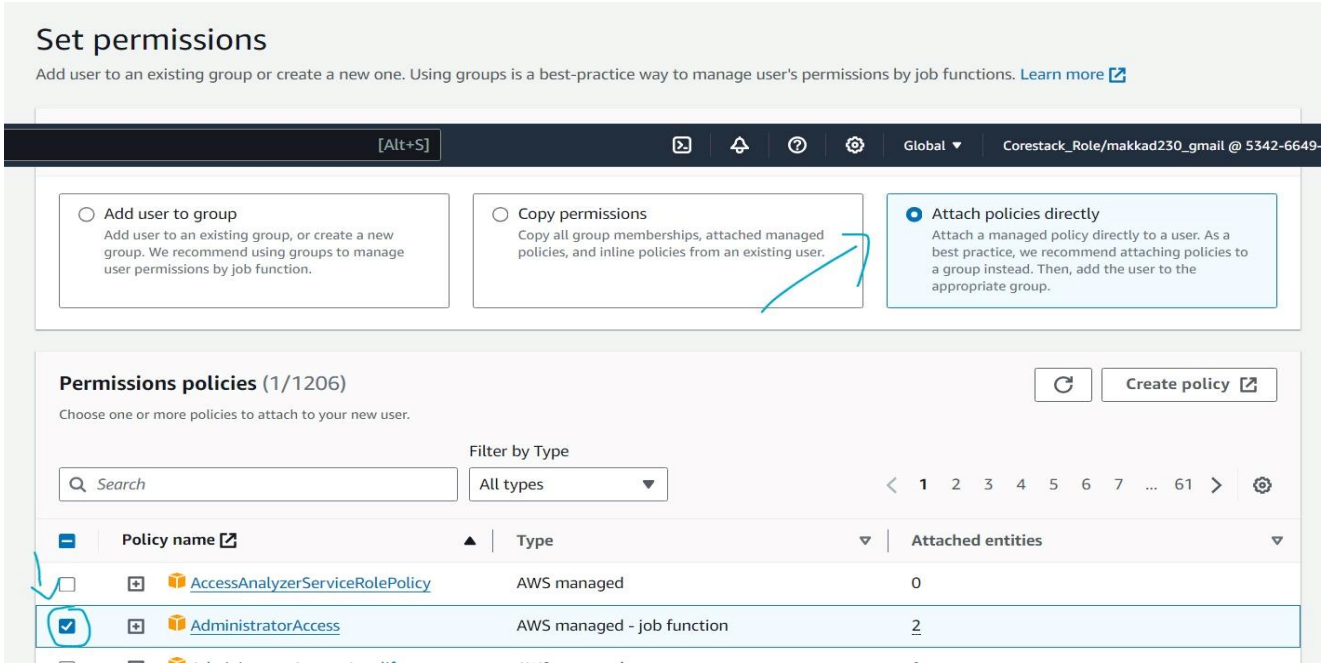
Creating User:

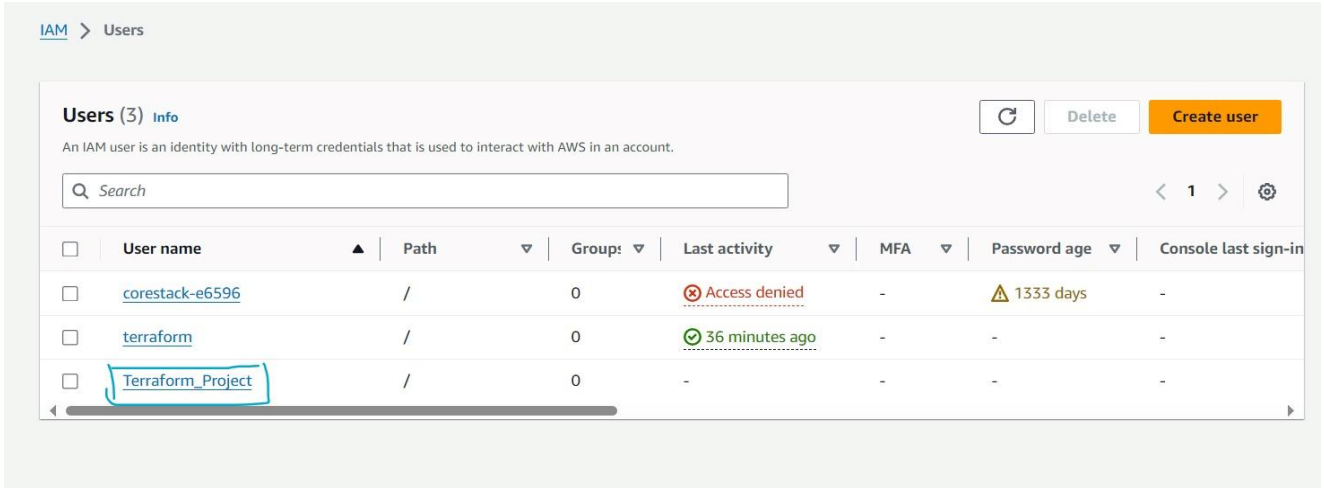
1- In the AWS dashboard, search for IAM and then select it.
2- One the left navigation panel select Users
3- Click on Create user.

4- Provide a Username
5- Click Next
6- Choose Attach policies directly Permission option
7- Choose AdministratorAccess in the Permission Policies
8- Click Next then Click Create user









Create Security Credentials

1- Click on username Terraform_Project
2- Click on Security credentials tab
3- Scroll and create Access Keys
4- Choose command line (CLI)
5- Check the box “I understand the above”
6- Create access Key

Permissions | Groups | Tags | Security credentials | Access Advisor

Console sign-in

Enable console access

Console sign-in link
https://534266494422.signin.aws.amazon.com/console

Console password
Not enabled

Multi-factor authentication (MFA) (0)

Remove | Resync | Assign MFA device

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. Learn more

Type	Identifier	Certifications	Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment

Assign MFA device

Access keys (0)

Create access key

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

IAM > Users > Terraform_Project > Create access key

Step 1
Access key best practices & alternatives

Step 2 - *optional*
Set description tag

Step 3
Retrieve access keys

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

- (●) **Command Line Interface (CLI)**
  You plan to use this access key to enable the AWS CLI to access your AWS account.
- ( ) **Local code**
  You plan to use this access key to enable application code in a local development environment to access your AWS account.
- ( ) **Application running on an AWS compute service**
  You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- ( ) **Third-party service**
  You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- ( ) **Application running outside AWS**
  You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Users > Terraform_Project > Create access key

Step 1
Access key best practices & alternatives

Step 2 - *optional*
Set description tag

Step 3
Retrieve access keys

Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAXYZGDXXLJN2URBCV	********** Show

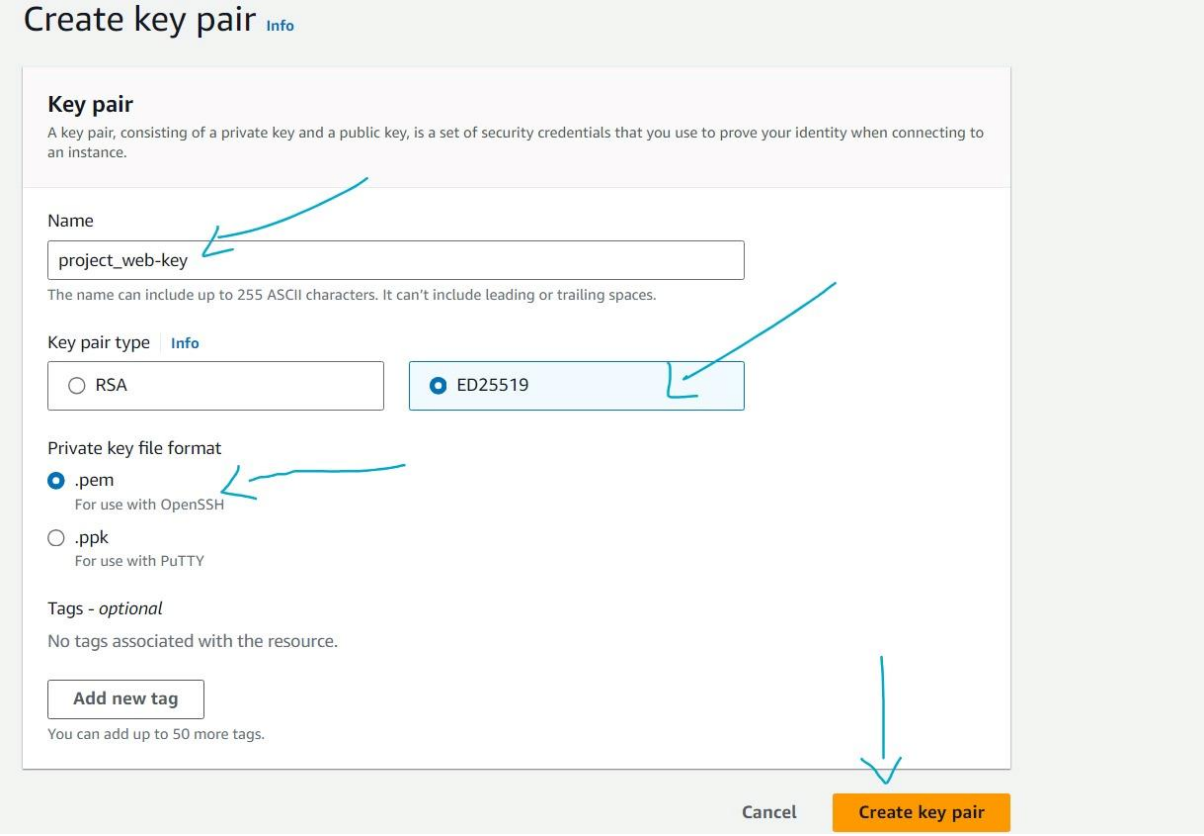
Access key best practices

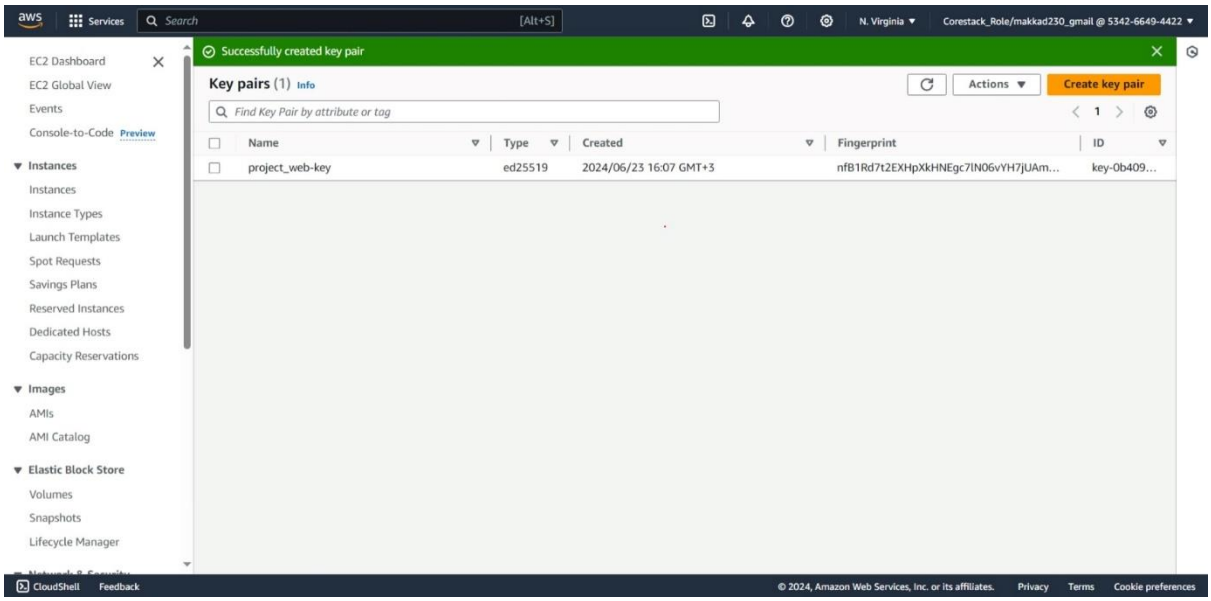
- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

Download .csv file | Done

Step 4: Create a key pair in AWS, we will use this keypair to connect to the ec2 instance





Step 5: Prepare the terraform configuration file with provider and resource blocks

In the configuration file , we will use:

- **Provider : aws**
- **Resource: security_group**
- **Resource: aws_instance**
- **Resource: aws_network_interface_sg_attachment**

Final code:

```
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.55.0...
- Installed hashicorp/aws v5.55.0 (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.
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$
```

```
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$ cat awsInfra.tf
provider "aws" {

   region = "us-east-1"

   access_key = "AKIAXYZGDXXLJN2URBCV"

   secret_key = "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"

 }


resource "aws_security_group" "mysg" {

  name        = "mysg"

  description = "Allow inbound SSH"


  ingress {

    from_port        = 22

    to_port          = 22

    protocol         = "tcp"

    cidr_blocks      = ["0.0.0.0/0"]

    ipv6_cidr_blocks = ["::/0"]

  }
ingress {

     description = "HTTP"

     from_port   = 8080

     to_port     = 8080

     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"]

   }

}

resource "aws_instance" "myec2" {

ami = "ami-0eaf7c3456e7b5b68"

instance_type = "t2.micro"

key_name = "project_web-key"
```

```
tags = {

    Name = "instance-1"

  }

 user_data = <<-EOF

      #!/bin/bash

        sudo yum install git -y
        sudo amazon-linux-extras install java-openjdk11 -y
        sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
        sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
        sudo yum install jenkins -y
        sudo systemctl start jenkins
        sudo yum install python3 -y

EOF

}


resource "aws_network_interface_sg_attachment" "sg_attachment1" {

security_group_id = aws_security_group.mysg.id

network_interface_id = aws_instance.myec2.primary_network_interface_id


}
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$
```

Step 7: Execute the terraform configuration

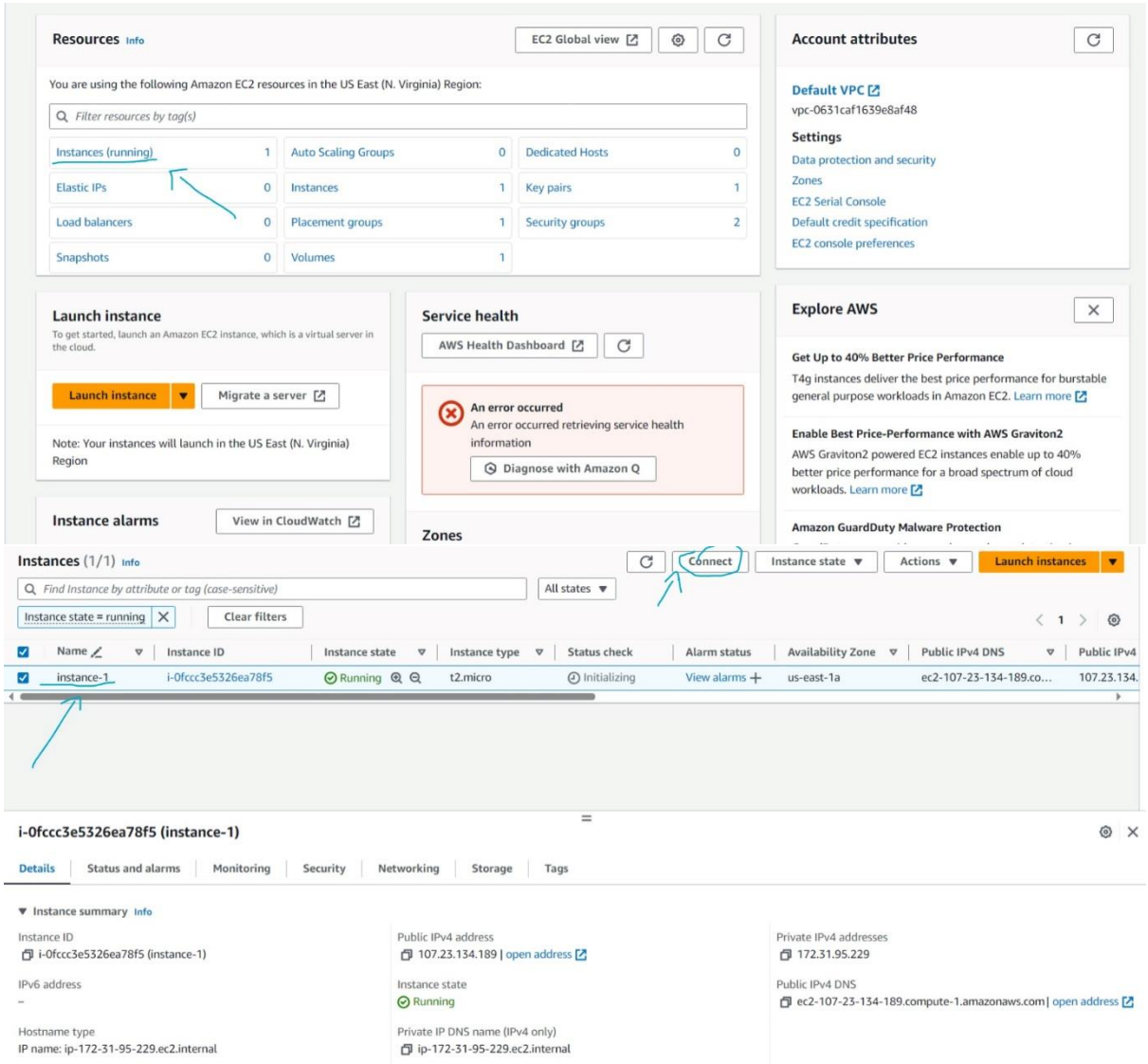
terraform apply

```
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$ terraform apply --auto-approve

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.myec2 will be created
  + resource "aws_instance" "myec2" {
      + ami                                  = "ami-0eaf7c3456e7b5b68"
      + 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                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "project_web-key"
      + 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_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                               = {
        + "Name" = "instance-1"
      }
    + tags_all                           = {
        + "Name" = "instance-1"
      }
    + tenancy                            = (known after apply)
    + user_data                          = "4baa13d20d79a9e74027be9aa8939bf2603d7790"
    + user_data_base64                   = (known after apply)
    + user_data_replace_on_change        = false
    + vpc_security_group_ids             = (known after apply)
  }

# aws_network_interface_sg_attachment.sg_attachment1 will be created
+ resource "aws_network_interface_sg_attachment" "sg_attachment1" {
    + id                 = (known after apply)
    + network_interface_id = (known after apply)
    + security_group_id    = (known after apply)
  }

# aws_security_group.mysg will be created
+ resource "aws_security_group" "mysg" {
    + arn                    = (known after apply)
    + description            = "Allow inbound SSH"
    + egress                 = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + from_port        = 0
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "-1"
            + security_groups  = []
            + self             = false
            + to_port          = 0
              # (1 unchanged attribute hidden)
          },
      ]
    + id                     = (known after apply)
    + ingress                = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + from_port        = 22
            + ipv6_cidr_blocks = [
                + "::/0",
              ]
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 22
              # (1 unchanged attribute hidden)
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "HTTP"
            + from_port        = 8080
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
```

```
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 8080
          },
        ]
      + name                   = "mysg"
      + 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: 3 to add, 0 to change, 0 to destroy.
aws_security_group.mysg: Creating...
aws_instance.myec2: Creating...
aws_security_group.mysg: Creation complete after 2s [id=sg-0eb9cf3f43d7eebd6]
aws_instance.myec2: Still creating... [10s elapsed]
aws_instance.myec2: Still creating... [20s elapsed]
aws_instance.myec2: Still creating... [30s elapsed]
aws_instance.myec2: Creation complete after 31s [id=i-002626e6f6361a61b]
aws_network_interface_sg_attachment.sg_attachment1: Creating...
aws_network_interface_sg_attachment.sg_attachment1: Creation complete after 0s [id=sg-0eb9cf3f43d7eebd6_eni-0d6ddb38aabe939c8]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
makkad230gmail@ip-172-31-22-116:~/Desktop/TerraformPorject$
```

Step 8: Validate and check if the tools have been installed on the VM or not.



EC2 > Instances > i-0fccc3e5326ea78f5 > Connect to instance

Connect to instance Info

Connect to your instance i-0fccc3e5326ea78f5 (instance-1) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID

i-0fccc3e5326ea78f5 (instance-1)

Connection Type

- (selected) Connect using EC2 Instance Connect
  Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.
- Connect using EC2 Instance Connect Endpoint
  Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address

107.23.134.189

Username

Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

ec2-user

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

You have insufficient IAM permissions to connect to an instance using EC2 Instance Connect due to an organization policy

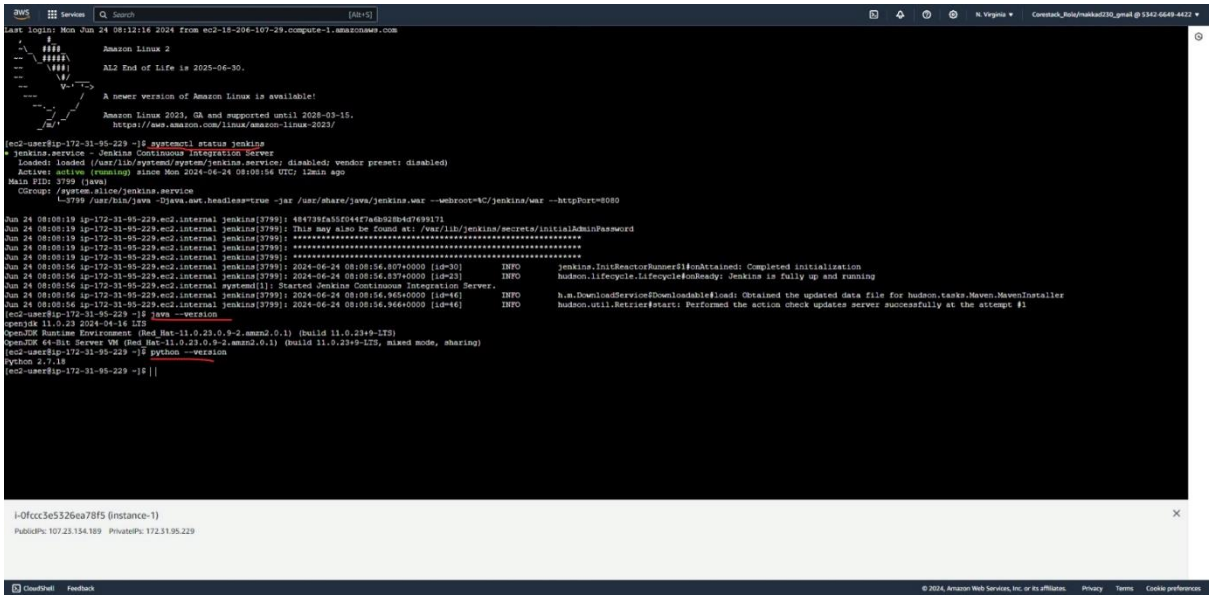
Your account is associated with an organization that has Service Control Policies (SCPs) enforcing limited permissions. To connect to an instance via EC2 Instance Connect, both SCPs and your IAM policy must grant the following permissions:

- `ec2-instance-connect:SendSSHPublicKey`
- `ec2:DescribeInstances`

Please contact your organization administrator to update SCPs in order to provide necessary permissions.

For more information about IAM policy examples, see Grant IAM permissions for EC2 Instance Connect.

Cancel | Connect



Completed the project