PROJECT 4: INFRASTRUCTURE AS CODE USING TERRAFORM AND AWS EC2

Question:

- **Create a Terraform configuration file that defines the desired infrastructure:**

  ➔ Create a new directory for the Terraform configuration files.
  ➔ Create a new file called main.tf and specify the desired infrastructure (e.g., EC2 instance, security group, key pair).

- **Use Terraform to apply the configuration file and provision the infrastructure in AWS:**

  ➔ Install Terraform on your local machine or a separate server.
  ➔ Authenticate Terraform with your AWS account and configure the necessary AWS permissions.
  ➔ Run the "terraform init" command to initialize the Terraform configuration.
  ➔ Run the "terraform apply" command to apply the Terraform configuration and provision the infrastructure in AWS.

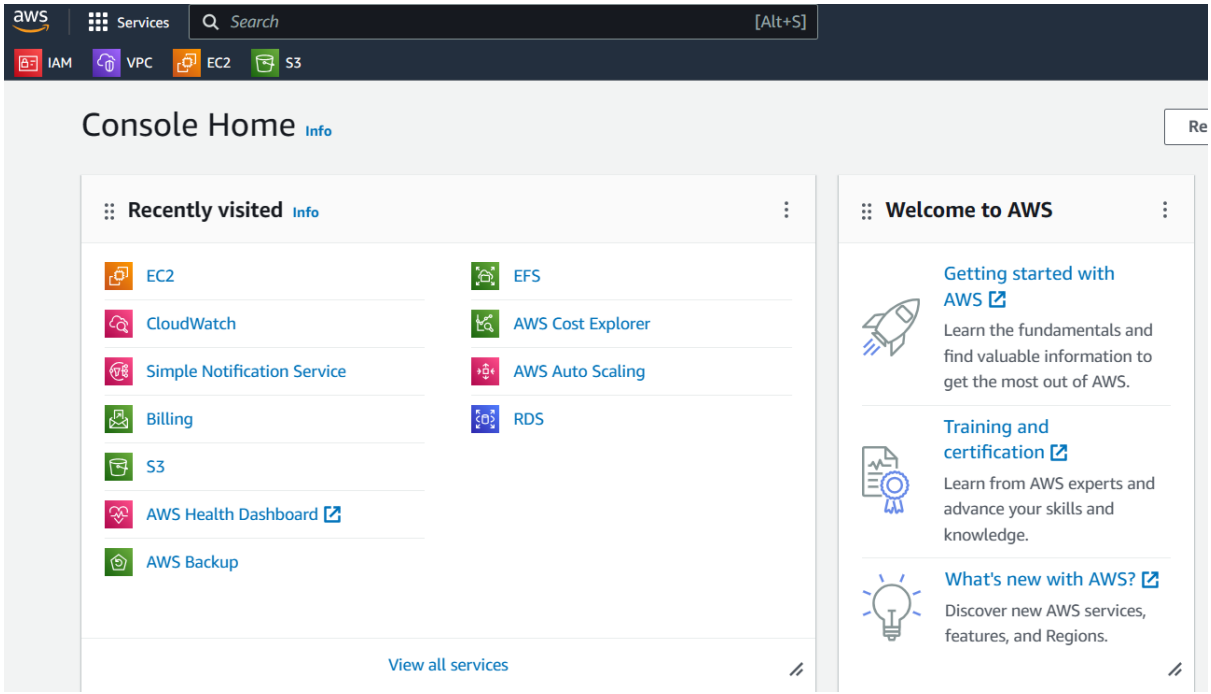
- **Use Terraform to update the infrastructure as necessary:**

  ➔ Modify the main.tf file to update the infrastructure configuration as necessary (e.g., change the instance type, add new security rules).
  ➔ Run the "terraform apply" command again to update the infrastructure in AWS.

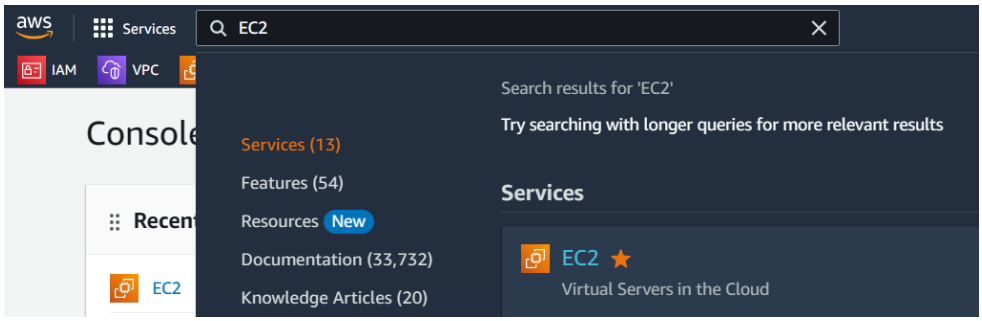
Solution:

Step:1 – Creating an EC2 instance for Terraform:

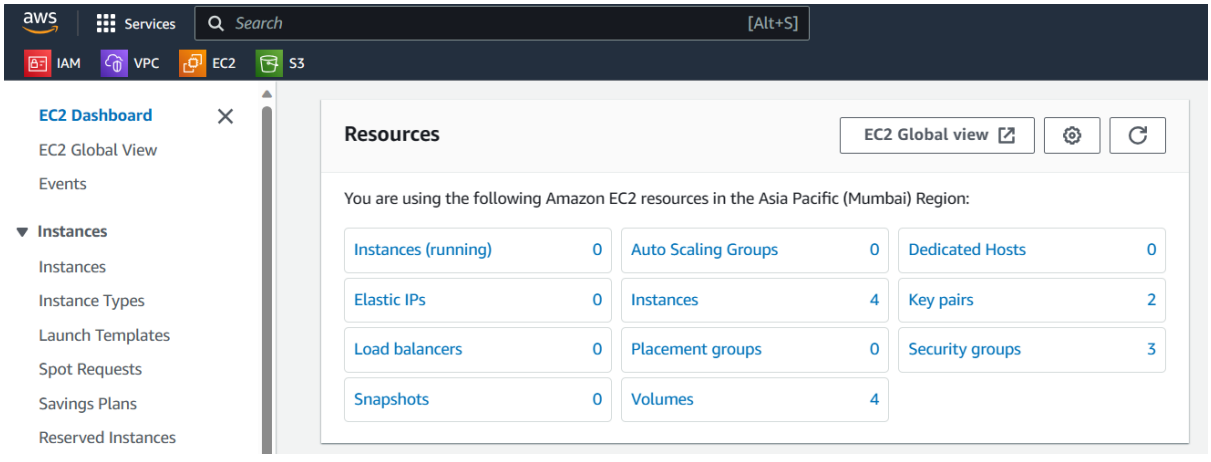
- First login into your AWS Management Console:



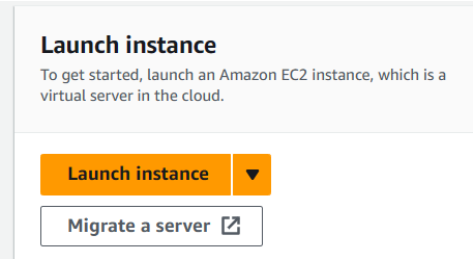
- Then search EC2 on service panel: click that one



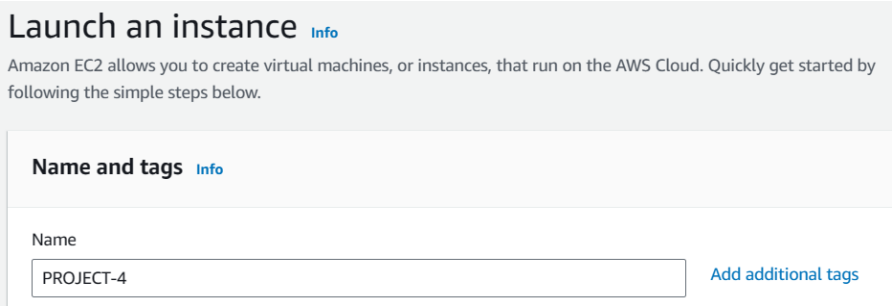
- EC2 Management console will appear:



- Then under the EC2 management console we could able to find the Launch instance option, click that one for creating an instance:

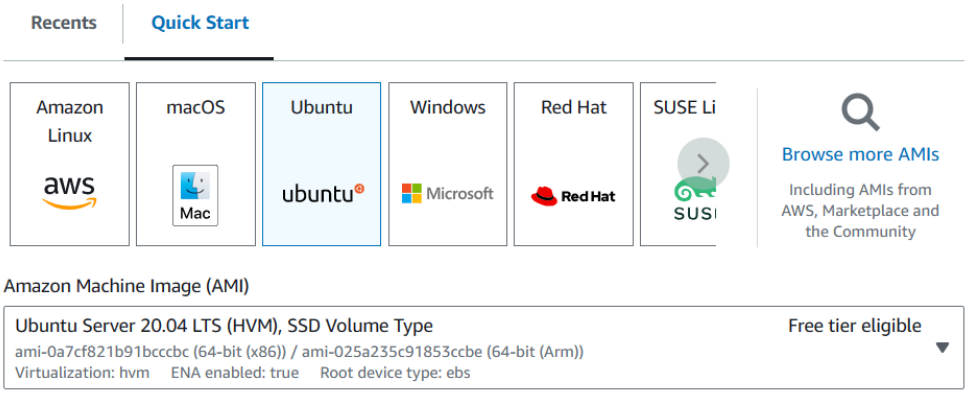


- Then name the instance according to your preferences,

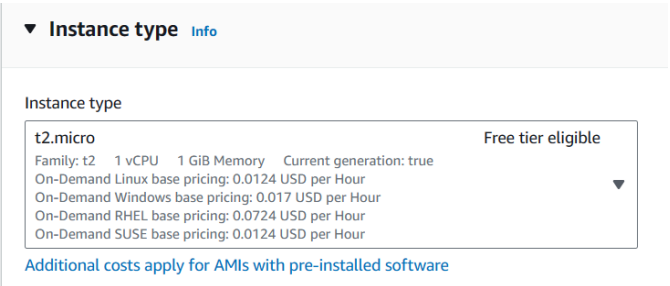


- Then we need to select an AMI Image for the instance, choose AMI image according to your preferences:

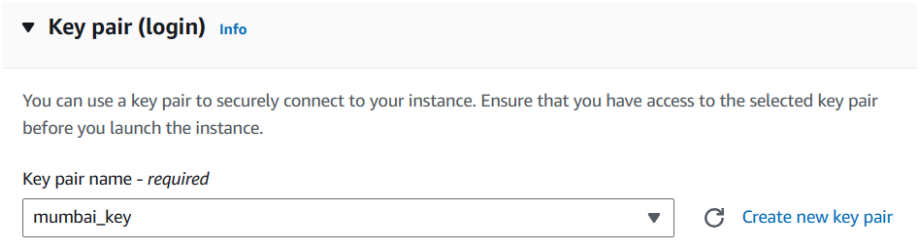
  Here I am selecting Ubuntu 20.04 AMI Image.



- Then we need to select the instance type, here I am selecting t2.micro which is free tier eligible provided by AWS.

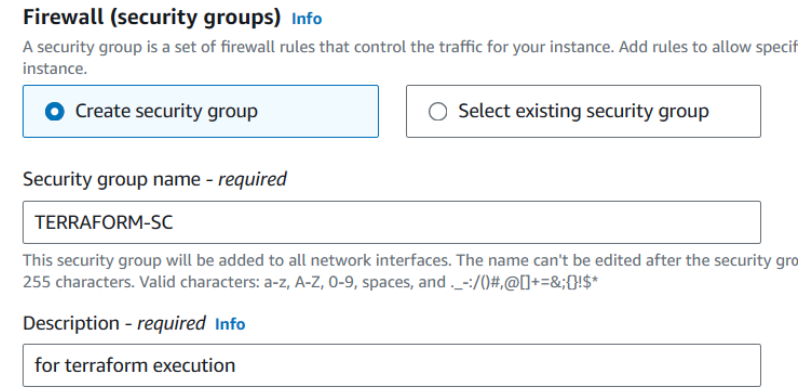


- Then we need to select the keypair for security authentication purpose of the instance:

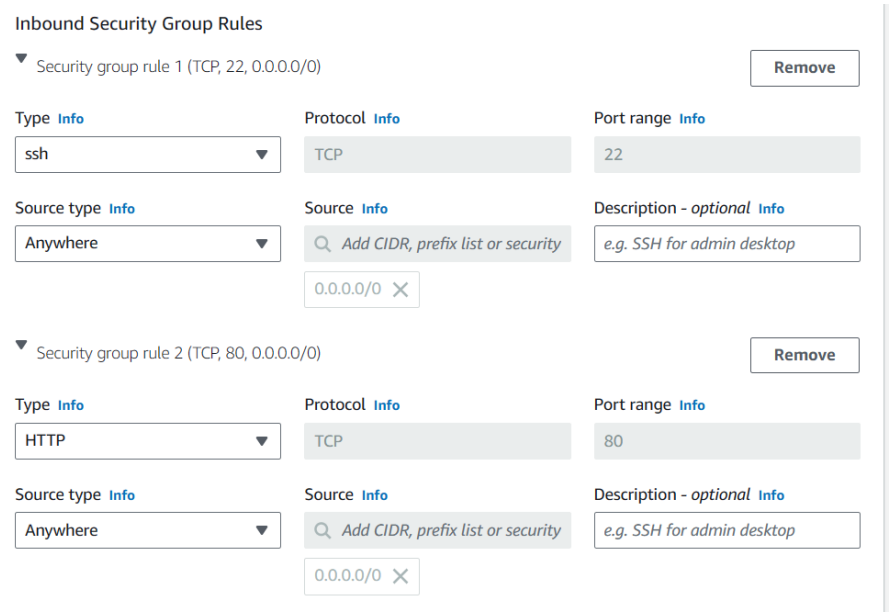


- Then under network settings, I am selecting default VPC, subnet no preference option, auto-assign public is enabled by default for default VPC & Subnet.

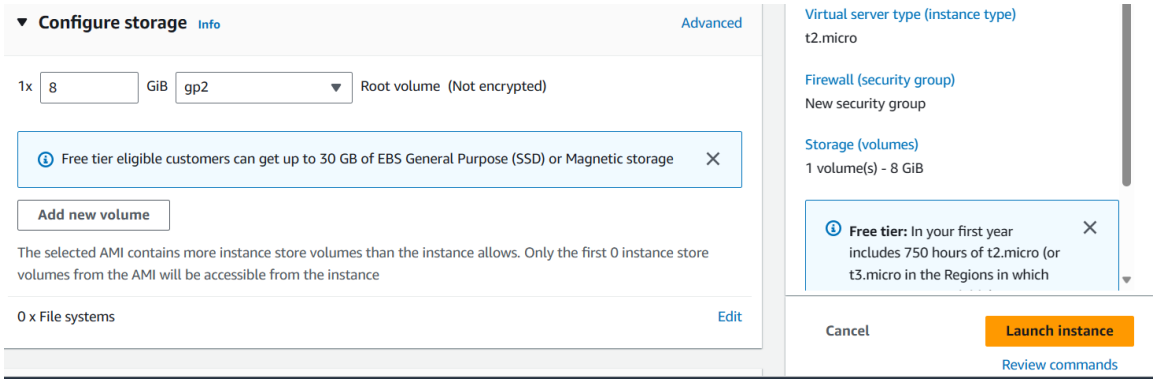
- Then creating a security group for terraform:



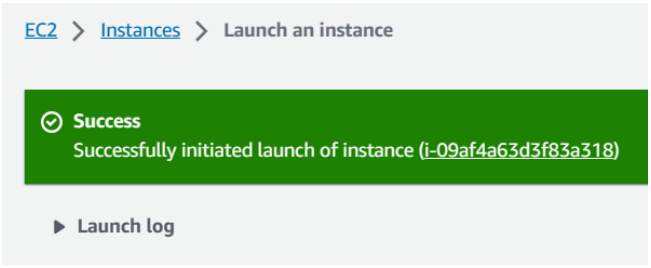
- Adding the rules, ssh & http:



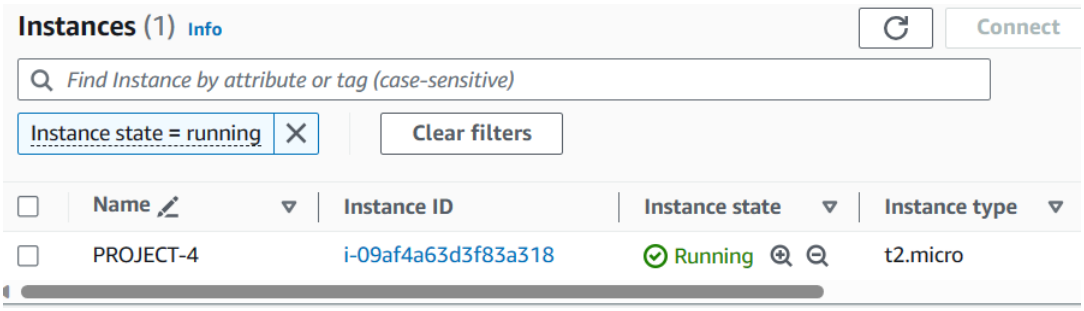
- Then keeping the default storage and click launch instances:



The instance has been launched:



- Just click the instances we could able to see the instance that we created:



Step:2 – Installing terraform on the created EC2 instance:

The commands to install terraform:

sudo apt-get update && sudo apt-get install -y gnupg software-properties-common

wget -O- https://apt.releases.hashicorp.com/gpg | \

gpg --dearmor | \

sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

gpg --no-default-keyring \

--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \

--fingerprint

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

sudo apt-get update

sudo apt-get install -y terraform

- to check the terraform is installed or not by using the command: **terraform --version** command:

```
root@ip-172-31-45-119:/home/ubuntu# terraform --version
Terraform v1.6.3
on linux_amd64
root@ip-172-31-45-119:/home/ubuntu#
```

****

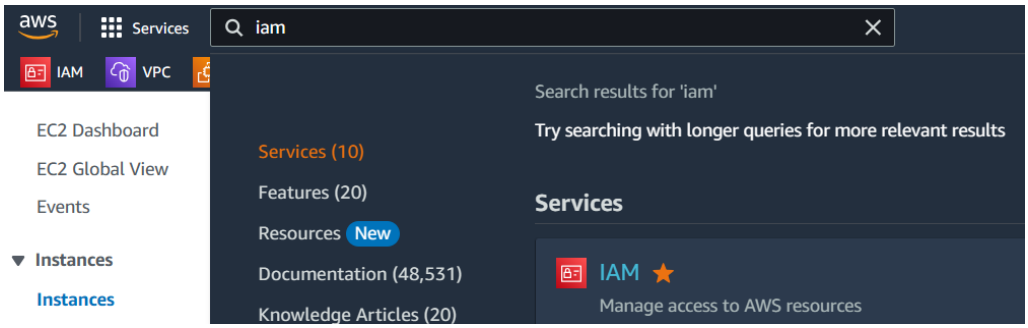
Step:3 – Creating IAM User:

- The IAM user should have full credentials or permissions to

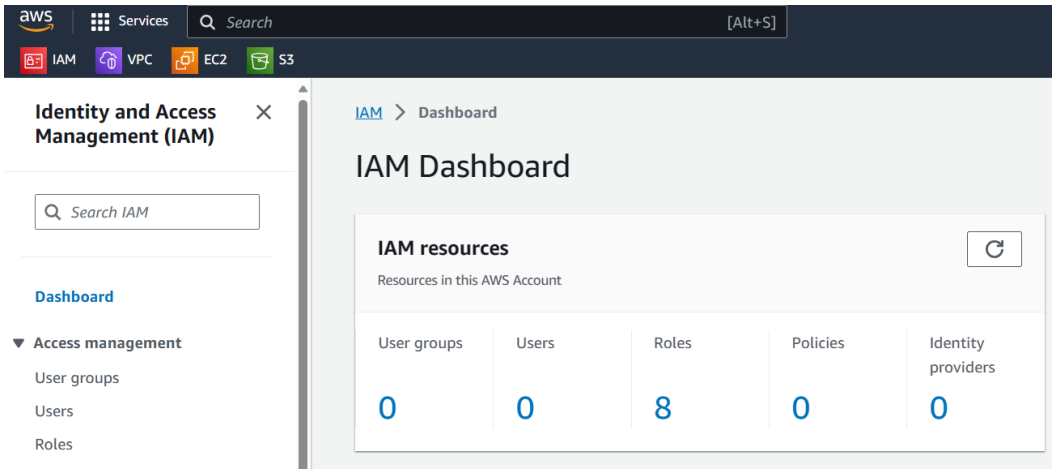
  ➔ **EC2 full access**

  ➔ **VPC full access**

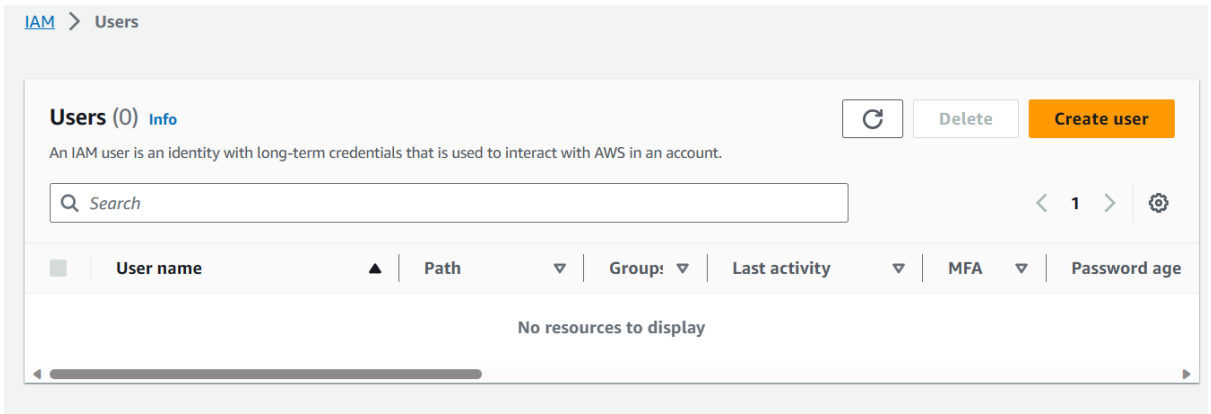
- On service panel search **IAM**, click that one:



- Then IAM dashboard will appear, on that **click users**:

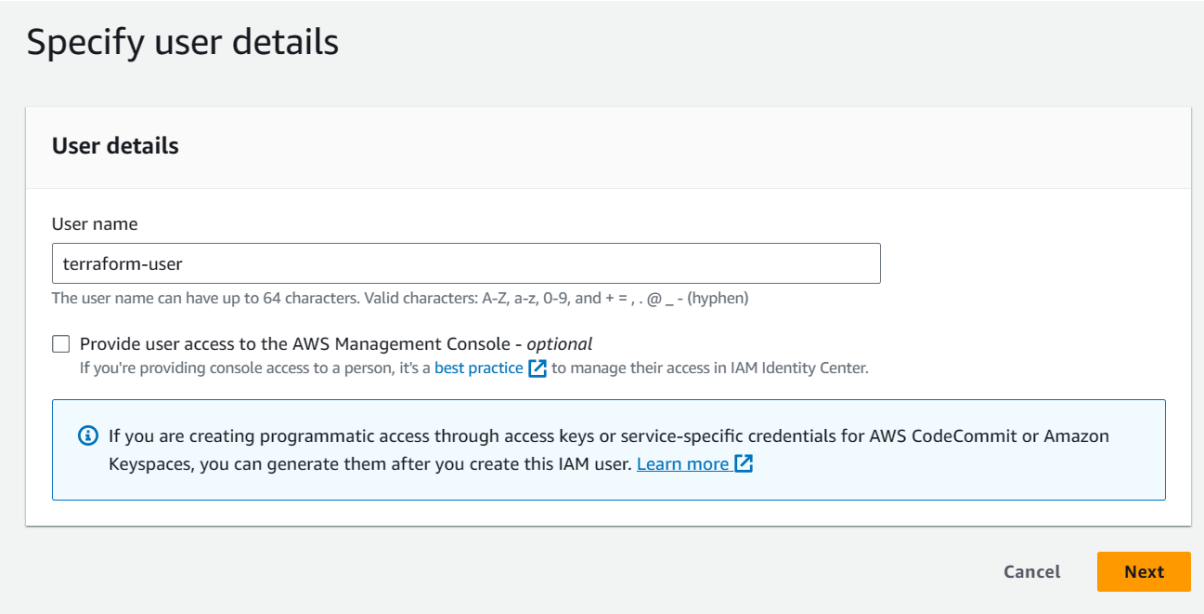


- Then click **create user:**

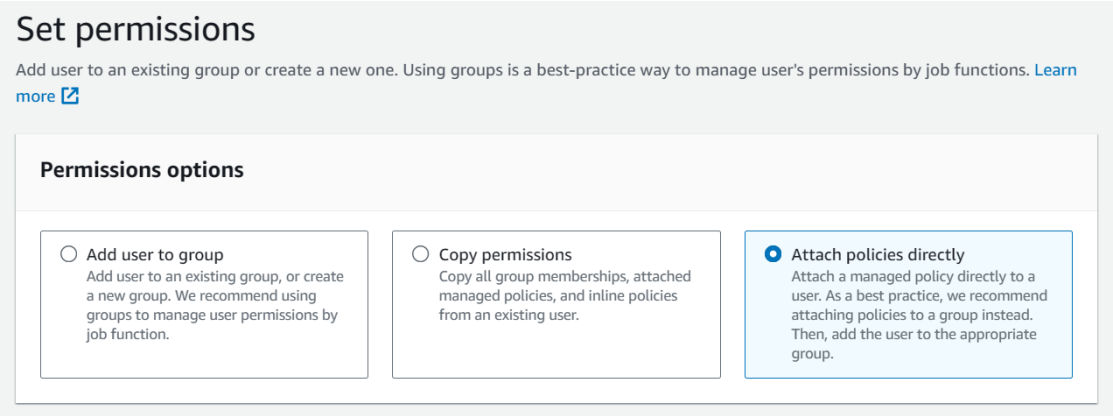


- Then name the IAM user, **if you provide this user access to AWS Management console, click that checkbox**. But here I am not selecting that option:

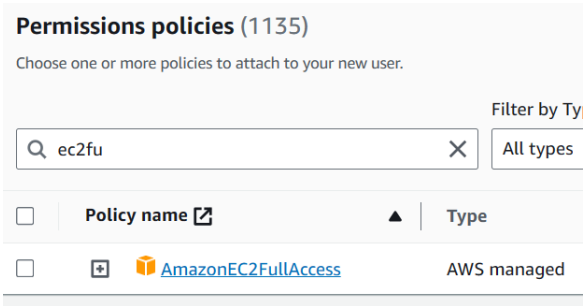
  Click next

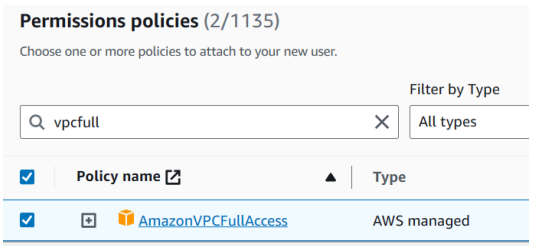


- Then under set permissions, select **attach policies directly:**

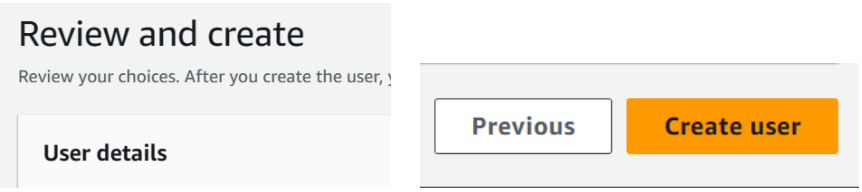


- Then attach the policies: **EC2 Full access, VPC full access:**

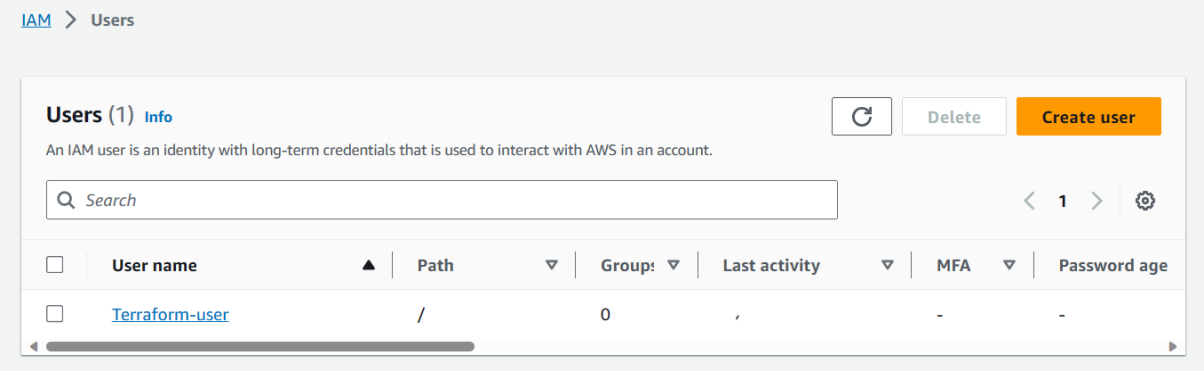




- Then click next, after that review and create page will appear, just review the configurations click **create user:**



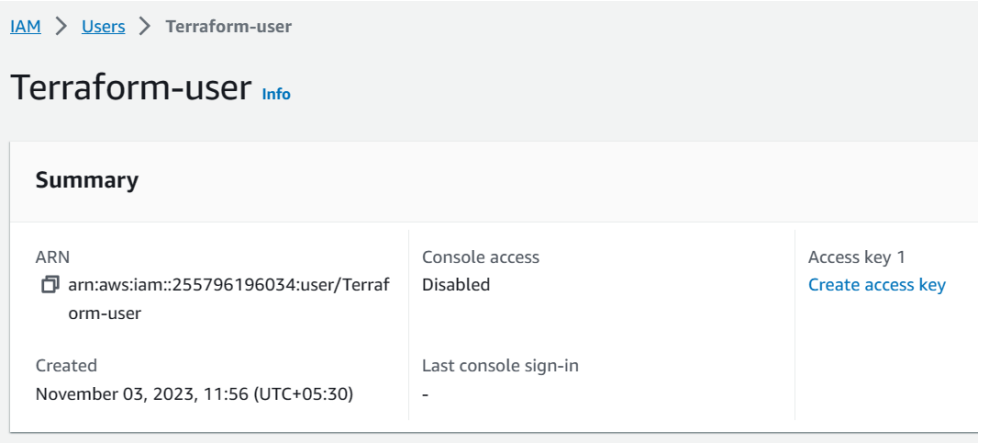
- The IAM user has been created successfully:



****

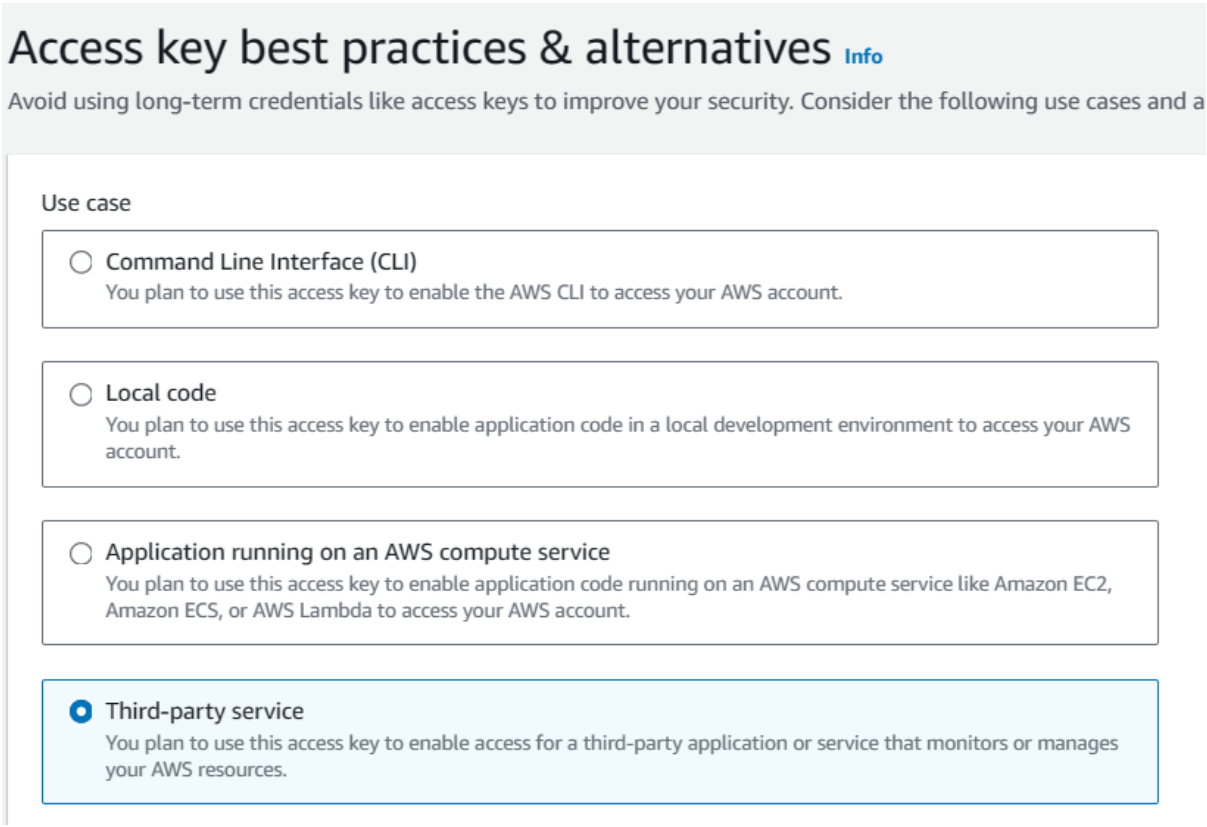
Step:4 – Creating access key & secret key for the created IAM user:

- Just click the created IAM user:

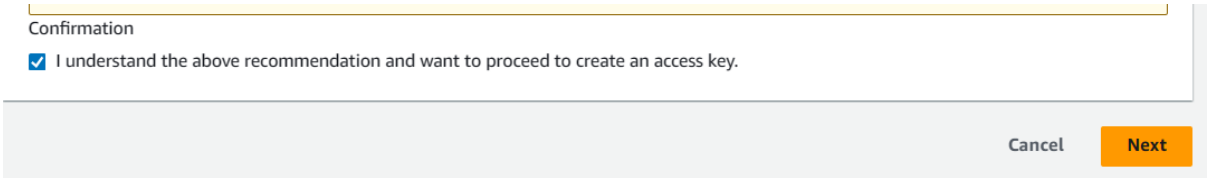


- Now click **create access key option:**
- after that we need to select the use case for creating this access key & secret key,

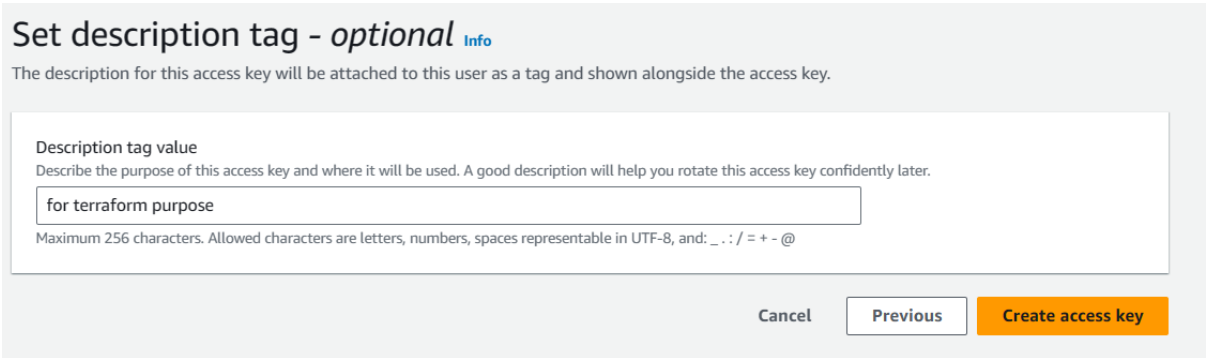
Note: here I am selecting third-party service because terraform is third party service:



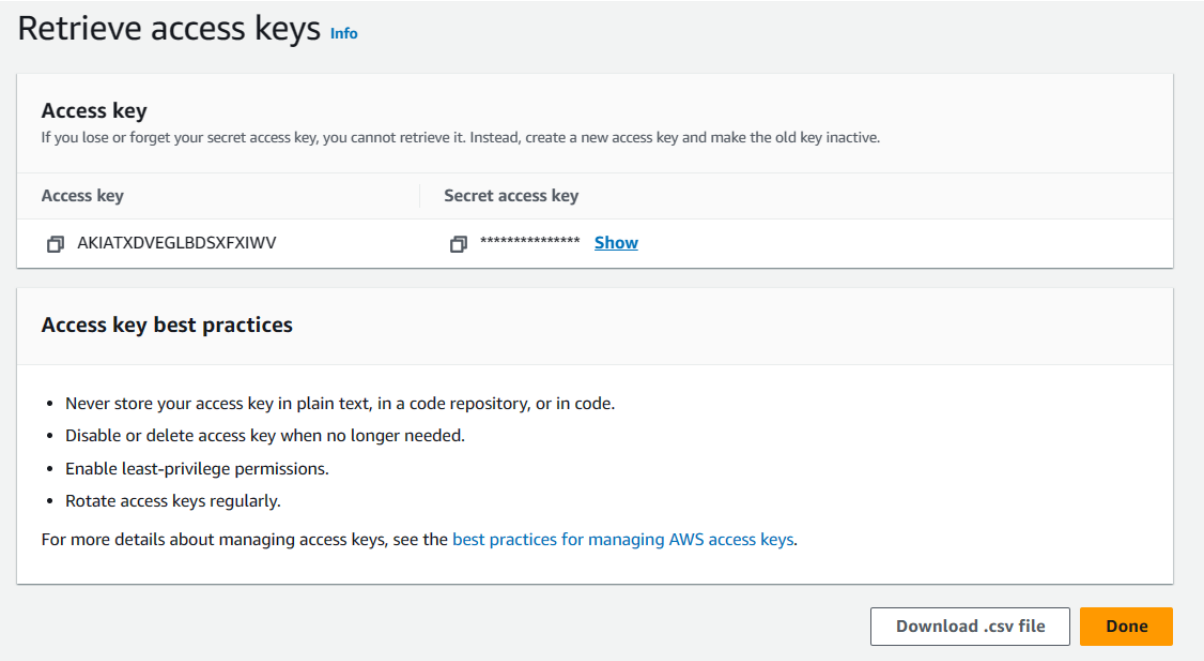
- now just assure the confirmation, click next:



- Then give the description tag, click create access key:



- After that we could able to see the access key and secret access key has been created:



****

Step:5 – Creating a terraform script for provisioning infrastructure:

- Creating a main.tf file:

```
root@ip-172-31-45-119:/home/ubuntu# mkdir project-4
root@ip-172-31-45-119:/home/ubuntu# ls
project-4
root@ip-172-31-45-119:/home/ubuntu# cd project-4/
root@ip-172-31-45-119:/home/ubuntu/project-4# touch main.tf
root@ip-172-31-45-119:/home/ubuntu/project-4# ls
main.tf
root@ip-172-31-45-119:/home/ubuntu/project-4#
```

Terraform script details:

- Coming to the script, under the script first we need to specify the credentials details of IAM user by installing awscli and configuring it:

  aws configure command:

```
root@ip-172-31-45-119:/home/ubuntu/project-4# aws configure
AWS Access Key ID [None]: AKIATXDVEGLBDSXFXIWV
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-a
Default output format [None]: json
root@ip-172-31-45-119:/home/ubuntu/project-4# aws configure
AWS Access Key ID [****************XIWV]:
AWS Secret Access Key [****************p6xZ]:
Default region name [ap-south-a]: ap-south-1
Default output format [json]:
```

- Then we need to setup profile under **~/.aws/credentials path:**

```
root@ip-172-31-45-119:/home/ubuntu/project-4# cd
root@ip-172-31-45-119:~# cd .aws/
root@ip-172-31-45-119:~/.aws# ls
config  credentials
```

```
  GNU nano 4.8                                                    cr
[terraform-user]
aws_access_key_id = AKIATXDVEGLBDSXFXIWV
aws_secret_access_key = 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
```

And saving it:

- Then we need to specify the provider, here I am using AWS Cloud, so under provider details, I will use AWS as the provider:

  Selecting the Mumbai region:

```
provider "aws" {
    profile = "terraform-user"
  region     = "ap-south-1"

}
```

- Then I am creating VPC with required components, this script will create VPC first with the CIDR block = 10.0.0.0/16.

```
# VPC Creation
resource "aws_vpc" "project-4-vpc" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"
```

```
  tags = {
    Name = "project-4-vpc"
  }
}
```

- Then I am creating internet gateway and attaching it to the newly created VPC:

```
# Internet Gateway Creation and Attaching it to the VPC
resource "aws_internet_gateway" "project-4-igw" {
  vpc_id = aws_vpc.project-4-vpc.id

  tags = {
    Name = "project-4-igw"
  }
}
```

- Then I am creating public subnet, with
  subnet block = 10.0.1.0/24
  availability zone = ap-south-1b

  mapping it to the newly created vpc:

```
# Subnet Creation
resource "aws_subnet" "project-4-public-subnet" {
  vpc_id            = aws_vpc.project-4-vpc.id
  cidr_block        = "10.0.1.0/24"
  availability_zone = "ap-south-1b"
  tags = {
    Name = "project-4-public-subnet"
  }
}
```

- Then creating a route table and mapping it with the newly created vpc with this script

```
# Route Table Creation
resource "aws_route_table" "project-4-route-table" {
```

```
  vpc_id = aws_vpc.project-4-vpc.id
  tags = {
    Name = "project-4-route-table"
  }
}
```

- Then in the newly created route table inserting the internet gateway route, so that subnet associated with this route table will have internet access:

```
# Inserting the Route to Internet Gateway
resource "aws_route" "project-4-routing" {
  route_table_id         = aws_route_table.project-4-route-table.id
  destination_cidr_block = "0.0.0.0/0" # Route all traffic to the
Internet Gateway
  gateway_id             = aws_internet_gateway.project-4-igw.id
}
```

- Then attaching the public subnet which we created with the newly created route table:

```
# Subnet Association with the Route Table
resource "aws_route_table_association" "subnet_asscio" {
  subnet_id      = aws_subnet.project-4-public-subnet.id
  route_table_id = aws_route_table.project-4-route-table.id
}
```

- Then creating a security group with the inbound roles and outbound rules:

  Inbound rules:
  ➔ Ssh
  ➔ http

  Outbound rules:
  ➔ anywhere

```
resource "aws_security_group" "project-4-sc" {
  name        = "project-4-sc"
  description = "security group for AWS EC2 instances"
```

```
  vpc_id      = aws_vpc.project-4-vpc.id
  # Ingress rules (inbound traffic)
  # Allow SSH (port 22) from anywhere
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow HTTP (port 80) from anywhere
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # Egress rules (outbound traffic)
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "project-4-sc"
  }
}
```

- Then creating a keypair under given region:

```
resource "aws_key_pair" "project-4-key" {
  key_name   = "project-4-key"
  public_key = "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2c
EikKDEY0aIj41qgxMCP/iteneqXSIFZBp5vizPvaoIR3Um9xK7PGoW8giupG
n+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7
```

```
tgBNL0cYlWSYVkz4G/fslNfRPW5mYAM49f4fhtxPb5ok4Q2Lg9dPKVHO/Bge
u5woMc7RY0p1ej6D4CKFE6lymSDJpW0YHX/wqE9+cfEauh7xZcG0q9t2ta6F
6fmX0agvpFyZo8aFbXeUBr7osSCJNgvavWbM/06niWrOvYX2xwWdhXmXSrbX
8ZbabVohBK41 email@example.com"
}
```

- Now writing a script to create an EC2 instance under the VPC environment which we created by script:

```
# EC2 Instance Creation
resource "aws_instance" "project-4" {
  ami                         = "ami-0a7cf821b91bcccbc"
  instance_type               = "t2.nano"
  subnet_id                   = aws_subnet.project-4-public-
subnet.id
  key_name                    = aws_key_pair.project-4-
key.id
  vpc_security_group_ids      =
["${aws_security_group.project-4-sc.id}"]
  user_data                   = file("apache.sh")
  associate_public_ip_address = true
  tags = {
    Name = "project-4"
  }
}
```

- Now combining all these scripts into 1 single file as main.tf

```
root@ip-172-31-45-119:/home/ubuntu/project-4# ls
apache.sh  main.tf
root@ip-172-31-45-119:/home/ubuntu/project-4#
```

****

Step:6 – Executing the terraform script:

- The terraform script will create
  - ➔ VPC
  - ➔ Internet gateway
  - ➔ Subnet
  - ➔ Route table
  - ➔ Security group
  - ➔ Key pair
  - ➔ EC2 instance

- First command we need to perform is **terraform init:**

```
root@ip-172-31-45-119:/home/ubuntu/project-4# terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.24.0

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.
root@ip-172-31-45-119:/home/ubuntu/project-4#
```

- The second command we need to perform here is **terraform plan:** this is plan to create infrastructure according to your script and display a blueprint of it:

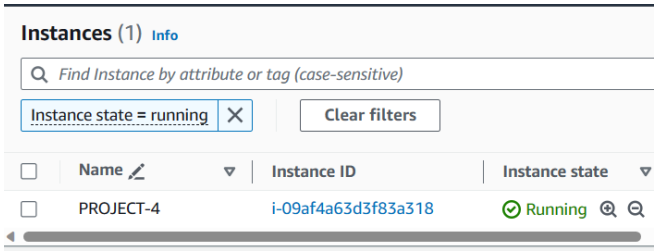
```
root@ip-172-31-45-119:/home/ubuntu/project-4# terraform plan

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:
```

```
      + instance_tenancy                    = "default"
      + ipv6_association_id                 = (known after apply)
      + ipv6_cidr_block                     = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                 = (known after apply)
      + owner_id                            = (known after apply)
      + tags                                = {
          + "Name" = "project-4-vpc"
        }
      + tags_all                            = {
          + "Name" = "project-4-vpc"
        }
    }

Plan: 9 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.
root@ip-172-31-45-119:/home/ubuntu/project-4#
```

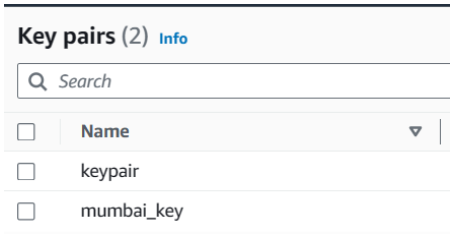
- Then the next command is **terraform apply -auto-approve** command for terraform init the creating,

Before applying **terraform apply** command:

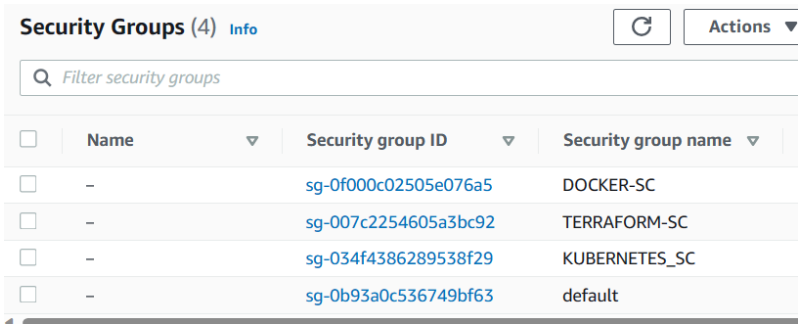
Instance:



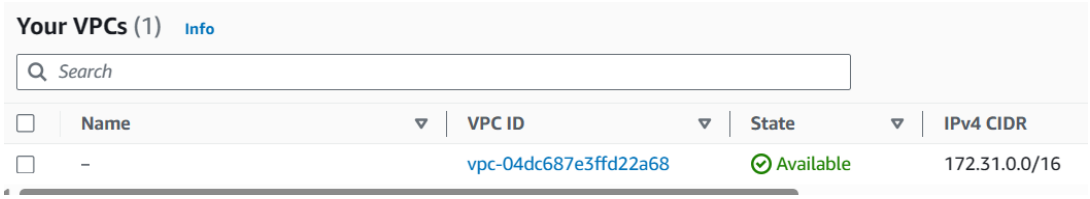
Key pairs:



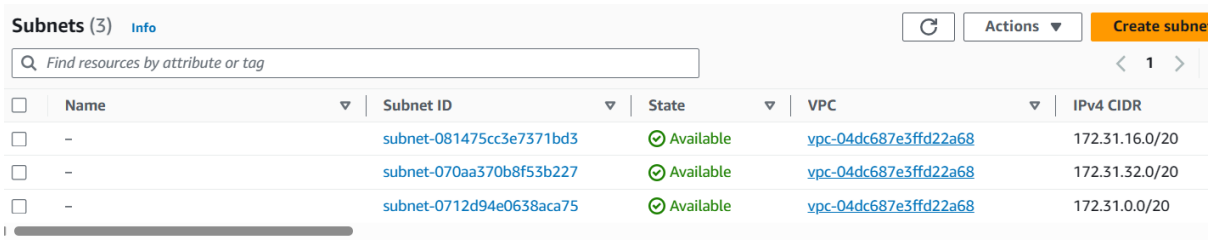
Security groups:



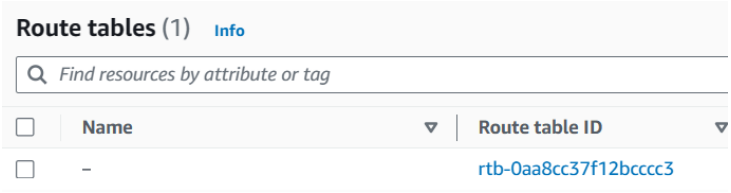
VPC:



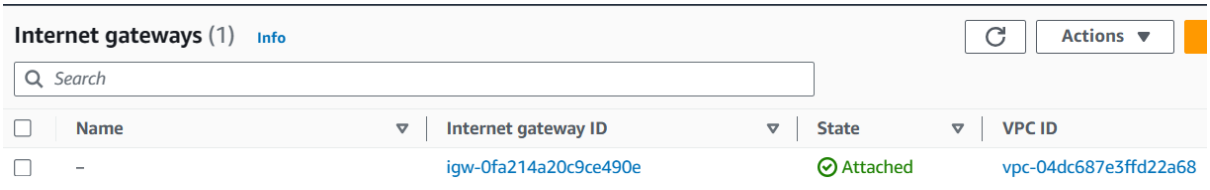
Subnets:



Route tables:



Internet Gateway:



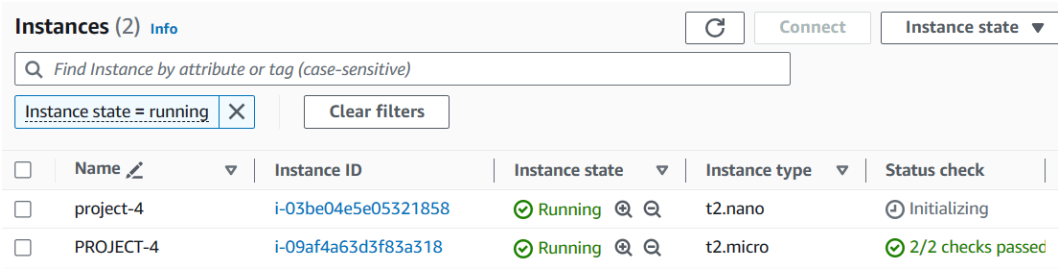
Applying terraform apply command:

```
root@ip-172-31-45-119:/home/ubuntu/project-4# 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:
```

```
Plan: 9 to add, 0 to change, 0 to destroy.
aws_key_pair.project-4-key: Creating...
aws_vpc.project-4-vpc: Creating...
aws_key_pair.project-4-key: Creation complete after 0s [id=project-4-key]
aws_vpc.project-4-vpc: Creation complete after 0s [id=vpc-0302d315cdb91aff0]
aws_route_table.project-4-route-table: Creating...
aws_security_group.project-4-sc: Creating...
aws_internet_gateway.project-4-igw: Creating...
aws_subnet.project-4-public-subnet: Creating...
aws_internet_gateway.project-4-igw: Creation complete after 1s [id=igw-089e7a7943db56511]
aws_subnet.project-4-public-subnet: Creation complete after 1s [id=subnet-0fe0a8f5e96a5a610]
aws_route_table.project-4-route-table: Creation complete after 1s [id=rtb-08bff820e8f7b2bde]
aws_route.project-4-routing: Creating...
aws_route_table_association.subnet_asscio: Creating...
aws_route_table_association.subnet_asscio: Creation complete after 0s [id=rtbassoc-0afe5ab7abc78598d]
aws_route.project-4-routing: Creation complete after 0s [id=r-rtb-08bff820e8f7b2bde1080289494]
aws_security_group.project-4-sc: Creation complete after 2s [id=sg-02ff8b40dadff19a1]
aws_instance.project-4: Creating...
aws_instance.project-4: Still creating... [10s elapsed]
aws_instance.project-4: Still creating... [20s elapsed]
aws_instance.project-4: Creation complete after 21s [id=i-03be04e5e05321858]

Apply complete! Resources: 9 added, 0 changed, 0 destroyed.
root@ip-172-31-45-119:/home/ubuntu/project-4#
```

Output:

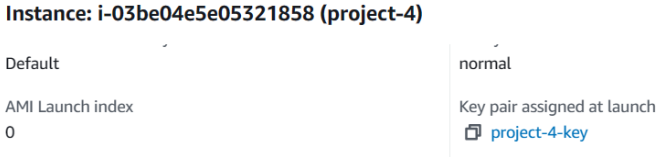
- **Instances:**



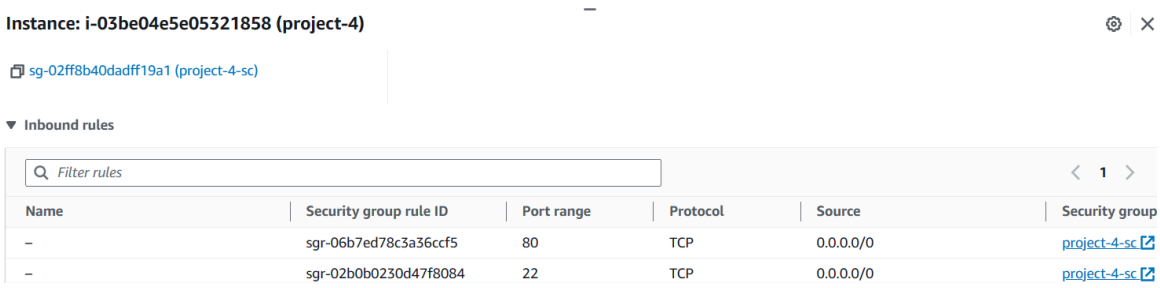
- **Instance details:**

Vpc & subnet are associated with the instance:

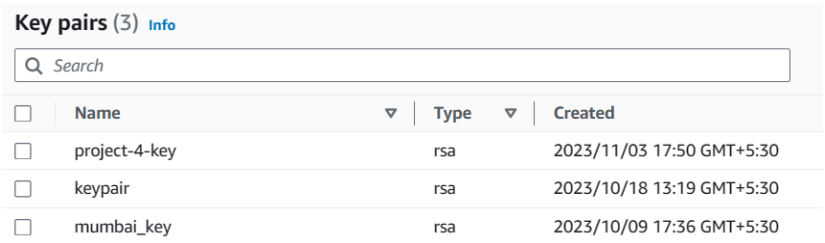
Key pair associated with the instance:



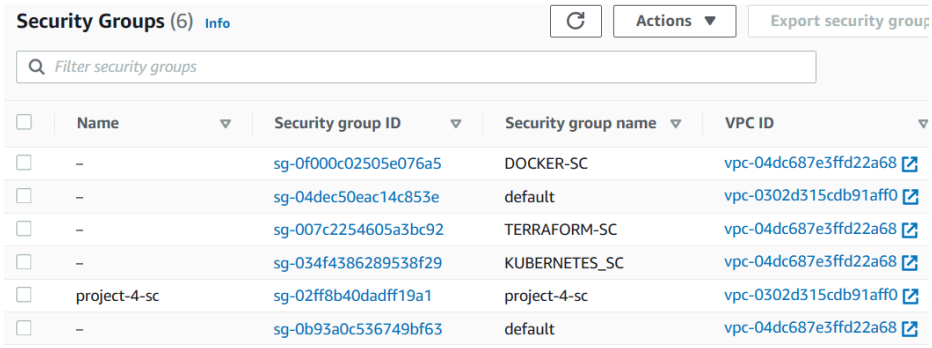
Security group of the instance:



- **Key pairs:**



- **Security groups:**



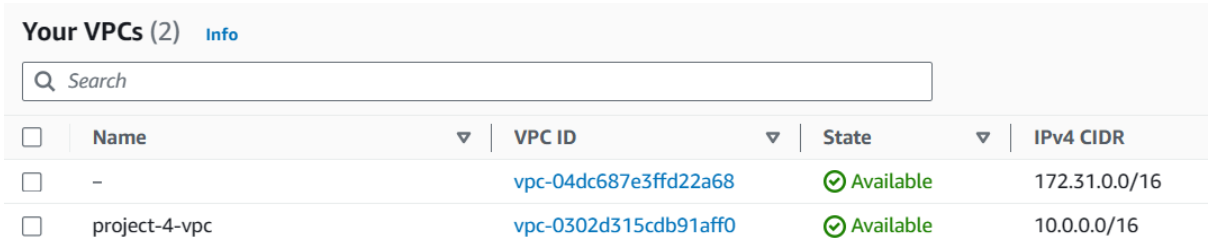
Security Groups (6) Info

Actions | Export security group

Filter security groups

Name	Security group ID	Security group name	VPC ID
–	sg-0f000c02505e076a5	DOCKER-SC	vpc-04dc687e3ffd22a68
–	sg-04dec50eac14c853e	default	vpc-0302d315cdb91aff0
–	sg-007c2254605a3bc92	TERRAFORM-SC	vpc-04dc687e3ffd22a68
–	sg-034f4386289538f29	KUBERNETES_SC	vpc-04dc687e3ffd22a68
project-4-sc	sg-02ff8b40dadff19a1	project-4-sc	vpc-0302d315cdb91aff0
–	sg-0b93a0c536749bf63	default	vpc-04dc687e3ffd22a68

- **VPC:**

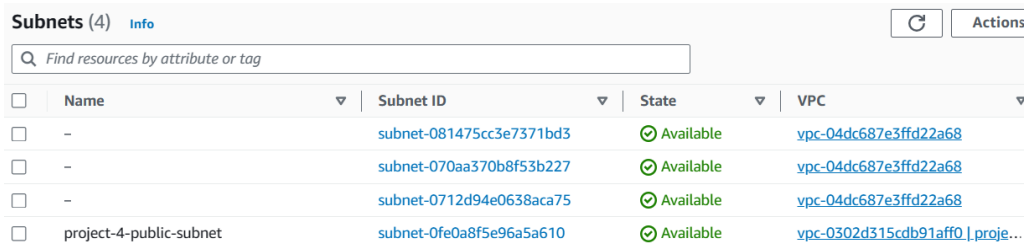


Your VPCs (2) Info

Search

Name	VPC ID	State	IPv4 CIDR
–	vpc-04dc687e3ffd22a68	Available	172.31.0.0/16
project-4-vpc	vpc-0302d315cdb91aff0	Available	10.0.0.0/16

- **Subnets:**



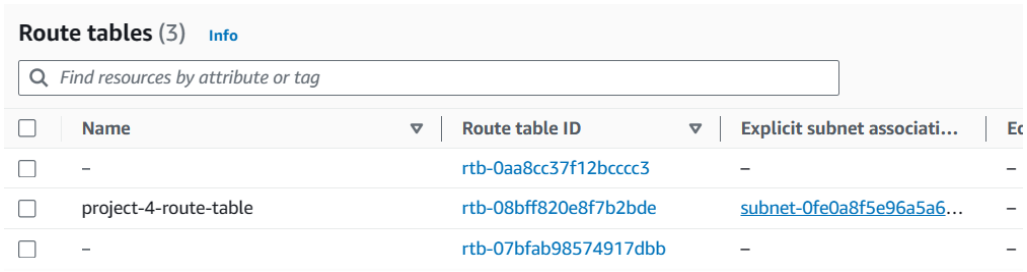
Subnets (4) Info

Actions

Find resources by attribute or tag

Name	Subnet ID	State	VPC
–	subnet-081475cc3e7371bd3	Available	vpc-04dc687e3ffd22a68
–	subnet-070aa370b8f53b227	Available	vpc-04dc687e3ffd22a68
–	subnet-0712d94e0638aca75	Available	vpc-04dc687e3ffd22a68
project-4-public-subnet	subnet-0fe0a8f5e96a5a610	Available	vpc-0302d315cdb91aff0 | proje...

- **Route table:**

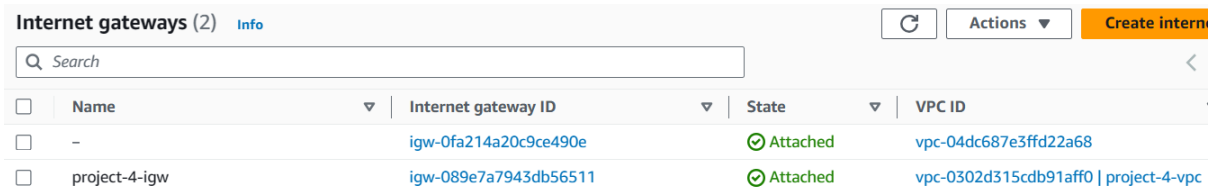


Route tables (3) Info

Find resources by attribute or tag

Name	Route table ID	Explicit subnet associati...	Ec
–	rtb-0aa8cc37f12bcccc3	–	–
project-4-route-table	rtb-08bff820e8f7b2bde	subnet-0fe0a8f5e96a5a6...	–
–	rtb-07bfab98574917dbb	–	–

- **Internet gateway:**



Internet gateways (2) Info

Actions | Create intern

Search

Name	Internet gateway ID	State	VPC ID
–	igw-0fa214a20c9ce490e	Attached	vpc-04dc687e3ffd22a68
project-4-igw	igw-089e7a7943db56511	Attached	vpc-0302d315cdb91aff0 | project-4-vpc

****

Step:7 - Modifying the terraform script:

- After creating the instance with the help of the terraform, now we need to make a small change by changing

➔ **The instance type from t2.nano to t2.micro**

```
# EC2 Instance Creation
resource "aws_instance" "project-4" {
  ami                         = "ami-0a7cf821b91bcccbc"
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.project-4-public-subnet.id
  key_name                    = aws_key_pair.project-4-key.id
  vpc_security_group_ids      = ["${aws_security_group.project-4-sc.id}"]
  user_data                   = file("apache.sh")
  associate_public_ip_address = true
  tags = {
    Name = "project-4"
  }
}
```

➔ **Adding https port under inbound rules of the security group:**

```
# Allow HTTPS (port 443) from anywhere
ingress {
  from_port   = 443
  to_port     = 443
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
```

- Now perform **terraform plan** command:

```
root@ip-172-31-45-119:/home/ubuntu/project-4# vi main.tf
root@ip-172-31-45-119:/home/ubuntu/project-4# terraform plan
aws_key_pair.project-4-key: Refreshing state... [id=project-4-key]
aws_vpc.project-4-vpc: Refreshing state... [id=vpc-0302d315cdb91aff0]
```

- Terraform has notified the changes from the script:

```
Terraform will perform the following actions:

  # aws_instance.project-4 will be updated in-place
  ~ resource "aws_instance" "project-4" {
        id                                   = "i-03be04e5e05321858"
      ~ instance_type                        = "t2.nano" -> "t2.micro"
        tags                                 = {
            "Name" = "project-4"
        }
        # (31 unchanged attributes hidden)

        # (8 unchanged blocks hidden)
    }
```

```
  + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = ""
      + from_port        = 443
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 443
    },
```

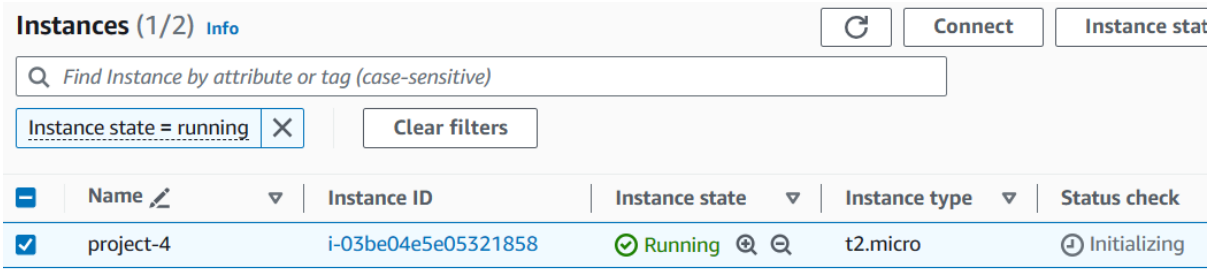
- Now we need to perform **terraform apply** command:

```
root@ip-172-31-45-119:/home/ubuntu/project-4# terraform apply -auto-approve
aws_vpc.project-4-vpc: Refreshing state... [id=vpc-0302d315cdb91aff0]
aws_key_pair.project-4-key: Refreshing state... [id=project-4-key]
```

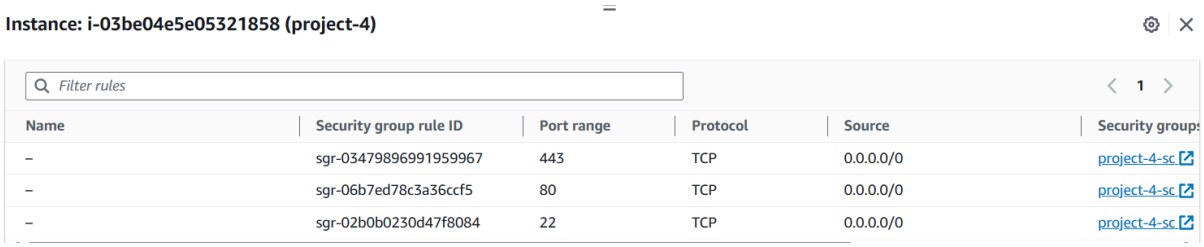
```
Plan: 0 to add, 2 to change, 0 to destroy.
aws_security_group.project-4-sc: Modifying... [id=sg-02ff8b40dadff19a1]
aws_security_group.project-4-sc: Modifications complete after 1s [id=sg-02ff8b40dadff19a1]
aws_instance.project-4: Modifying... [id=i-03be04e5e05321858]
aws_instance.project-4: Still modifying... [id=i-03be04e5e05321858, 10s elapsed]
aws_instance.project-4: Still modifying... [id=i-03be04e5e05321858, 20s elapsed]
aws_instance.project-4: Still modifying... [id=i-03be04e5e05321858, 30s elapsed]
aws_instance.project-4: Still modifying... [id=i-03be04e5e05321858, 40s elapsed]
aws_instance.project-4: Still modifying... [id=i-03be04e5e05321858, 50s elapsed]
aws_instance.project-4: Still modifying... [id=i-03be04e5e05321858, 1m0s elapsed]
aws_instance.project-4: Modifications complete after 1m0s [id=i-03be04e5e05321858]

Apply complete! Resources: 0 added, 2 changed, 0 destroyed.
root@ip-172-31-45-119:/home/ubuntu/project-4#
```

Output:

➔ Checking the instance type:



➔ Checking the security group:



Note:

Once everything is done do **terraform destroy -auto-approve** command to destroy everything we created by terraform:

****

- **The entire terraform script has been uploaded under the**

GitHub repository: https://github.com/Ravivarman16/Use-case-project-4.git

- **All the terraform scripts has been taken from official Terraform registry and edited according to my preferences.**

Terraform registry URL:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs

****