

PROJECT 4: INFRASTRUCTURE AS CODE USING TERRAFORM AND AWS EC2

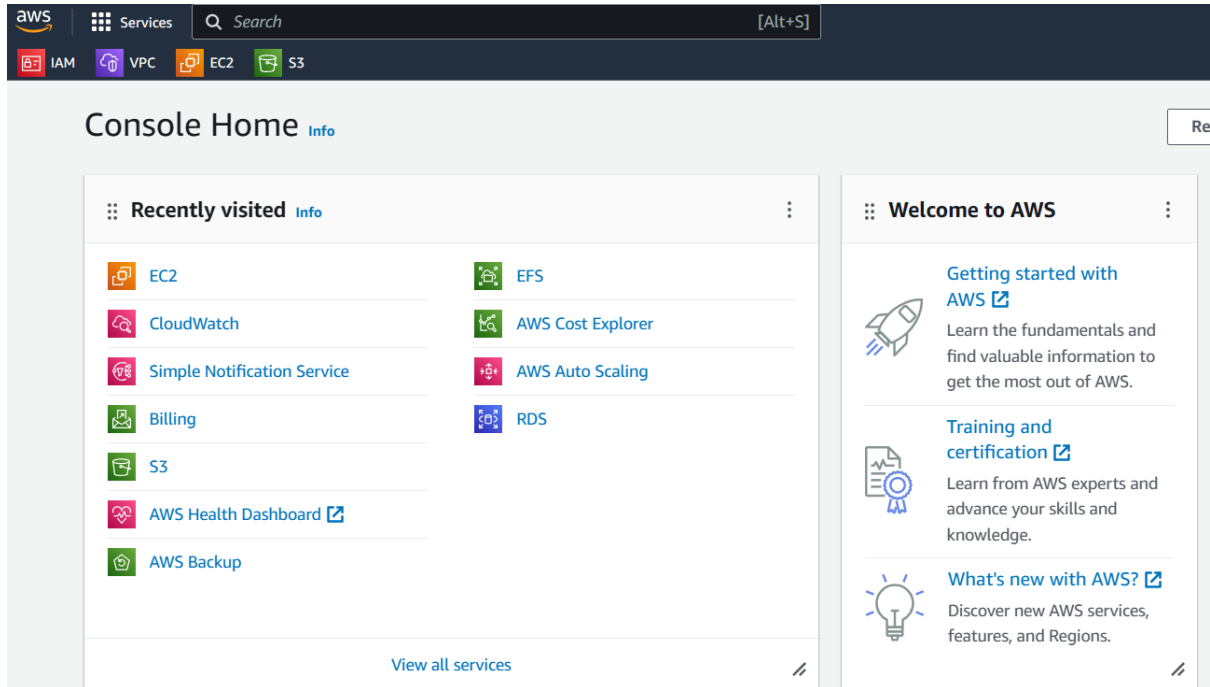
Question:

- **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.

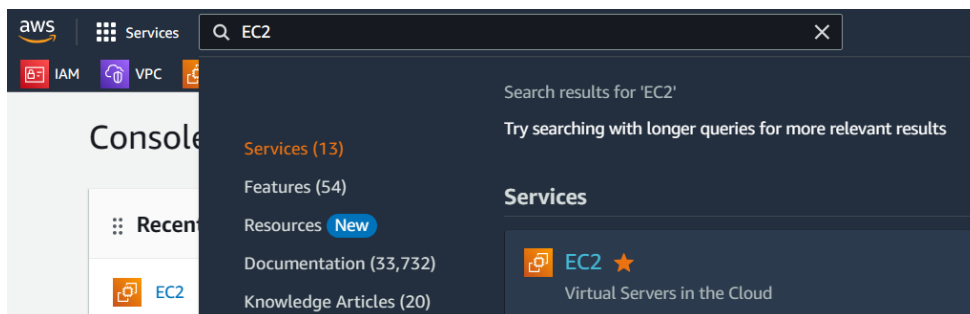
Solution:

Step:1 – Creating an EC2 instance for Terraform:

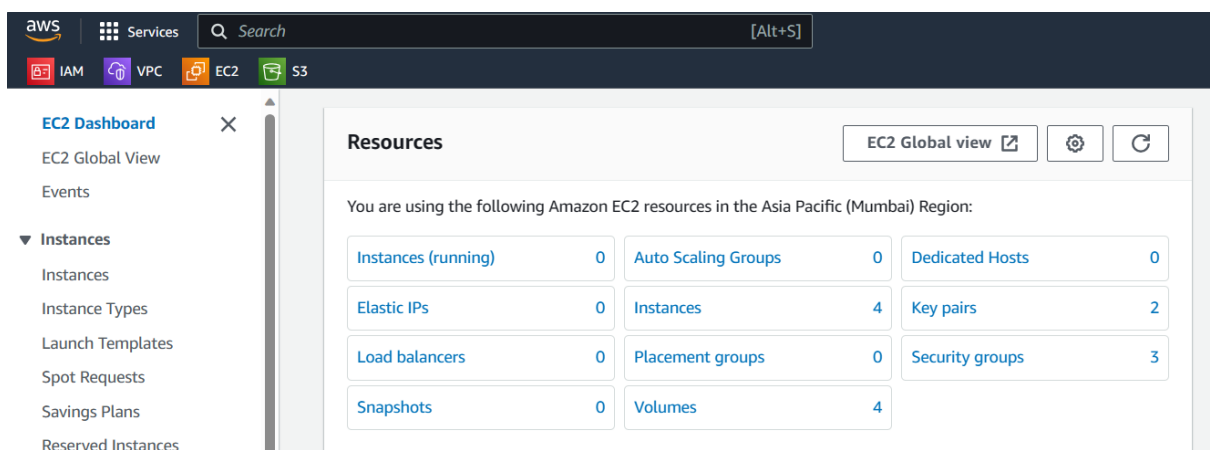
- First login into your AWS Management Console:



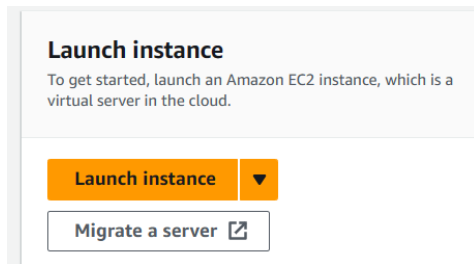
- Then search EC2 on service panel: click that one



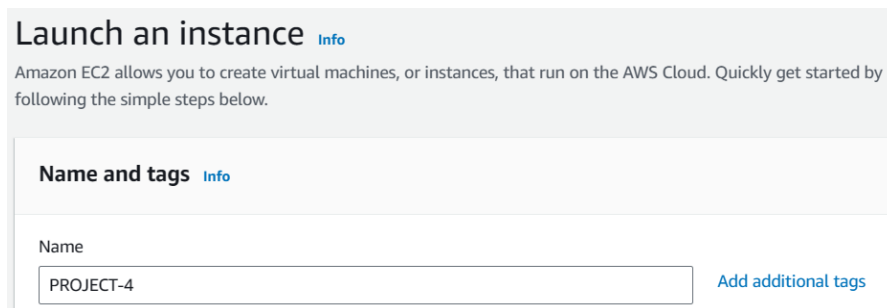
- EC2 Management console will appear:



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

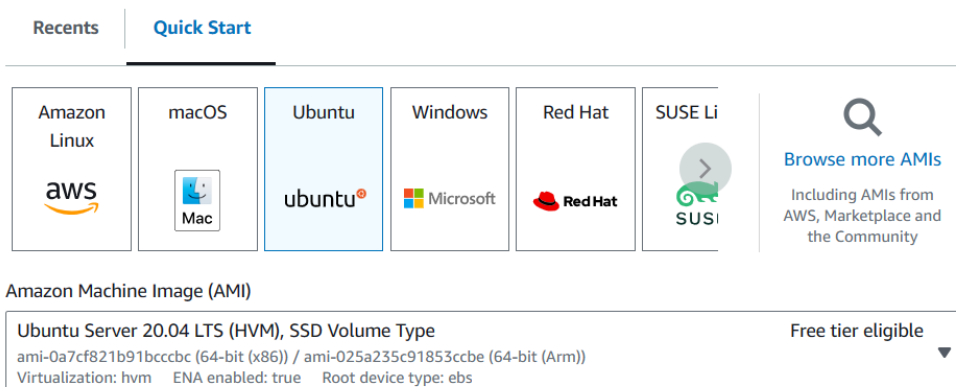


- Then name the instance according to your preferences,

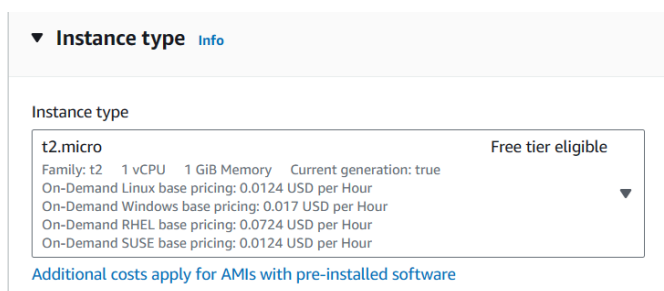


- Then we need to select an AMI Image for the instance, choose AMI image according to your preferences:

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:

▼ **Key pair (login)** [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

mumbai_key [Create new key pair](#)

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

▼ **Network settings** [Info](#)

VPC - required Info

vpc-04dc687e3ffd22a68 (default) ▼

Subnet Info

No preference [Create new subnet](#)

Auto-assign public IP [Info](#)

Enable 

- Then creating a security group for terraform:

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific instance.

☒ Create security group

☐ Select existing security group

Security group name - *required*

TERRAFORM-SC

This security group will be added to all network interfaces. The name can't be edited after the security group is created. The name can contain up to 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&:{}!\$*

Description - *required* [Info](#)

```
for terraform execution
```

- Adding the rules, ssh & http:

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) Remove

Type [Info](#) Protocol [Info](#) Port range [Info](#)

ssh TCP 22

Source type [Info](#) Source [Info](#) Description - optional [Info](#)

Anywhere Add CIDR, prefix list or security 0.0.0.0/0 X e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0) Remove

Type [Info](#) Protocol [Info](#) Port range [Info](#)

HTTP TCP 80

Source type [Info](#) Source [Info](#) Description - optional [Info](#)

Anywhere Add CIDR, prefix list or security 0.0.0.0/0 X e.g. SSH for admin desktop

- Then keeping the default storage and click launch instances:

▼ **Configure storage** [Info](#) Advanced

1x 8 GiB gp2 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage X

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems Edit

Virtual server type (instance type) t2.micro

Firewall (security group) New security group

Storage (volumes) 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which X

Cancel Launch instance [Review commands](#)

The instance has been launched:

[EC2](#) > [Instances](#) > Launch an instance

Success
Successfully initiated launch of instance (i-09af4a63d3f83a318)

► Launch log

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

Instances (1) [Info](#) Refresh Connect

Find Instance by attribute or tag (case-sensitive)

Instance state = running X Clear filters

<input type="checkbox"/>	Name ✎	Instance ID	Instance state	Instance type
<input type="checkbox"/>	PROJECT-4	i-09af4a63d3f83a318	Running 🔍	t2.micro

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
```

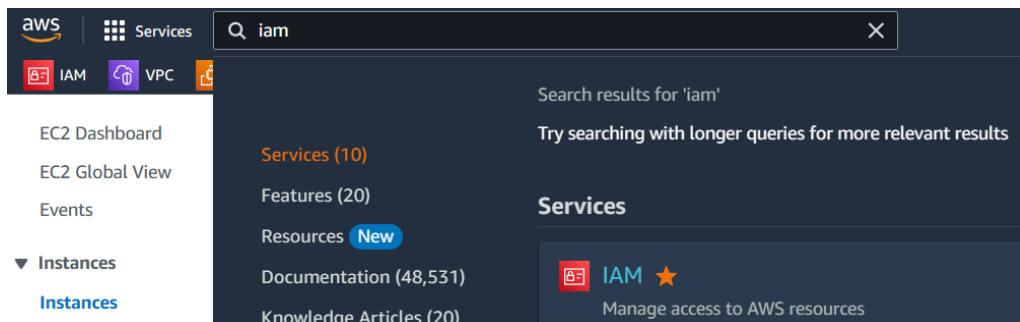
```
sudo apt-get install -y terraform
```

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

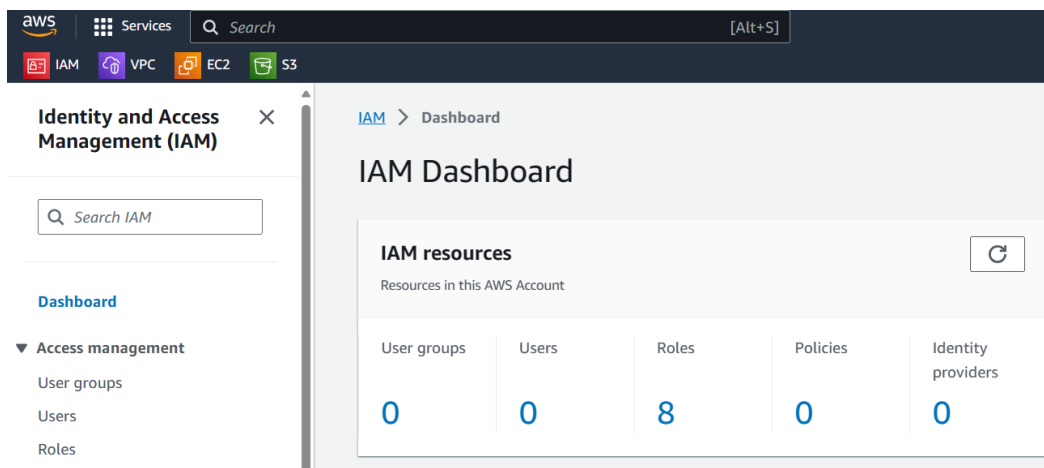
```
root@ip-172-31-45-119:/home/ubuntu# terraform --version
Terraform v1.6.3
on linux_amd64
root@ip-172-31-45-119:/home/ubuntu#
```

Step:3 – Creating IAM User:

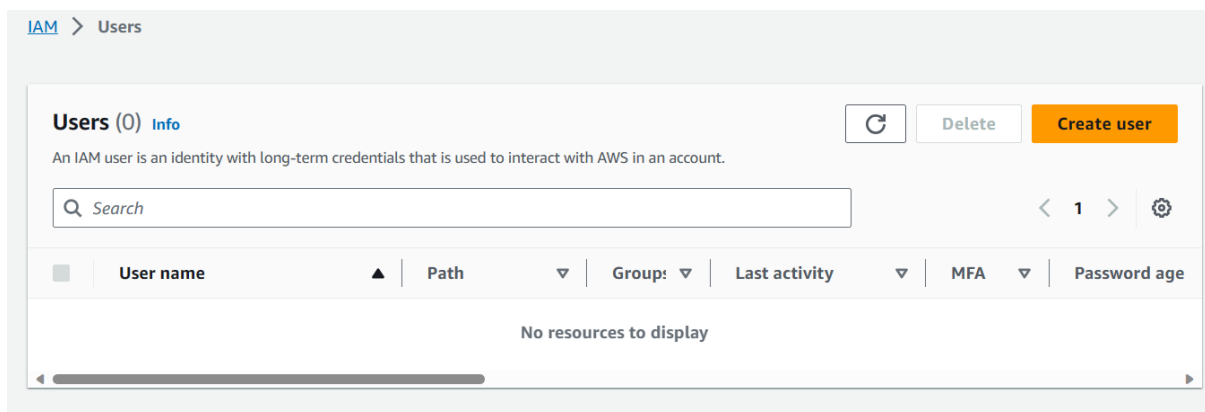
- The IAM user should have full credentials or permissions to
 - ➔ **EC2 full access**
 - ➔ **VPC full access**
- On service panel search **IAM**, click that one:



- 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

Specify user details

User details

User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

i If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel
Next

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

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ **Add user to group**
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ **Copy permissions**
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ **Attach policies directly**
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

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

Permissions policies (1/135)

Choose one or more policies to attach to your new user.

Filter by Type
All types

<input type="checkbox"/>	Policy name	Type
<input type="checkbox"/>	AmazonEC2FullAccess	AWS managed

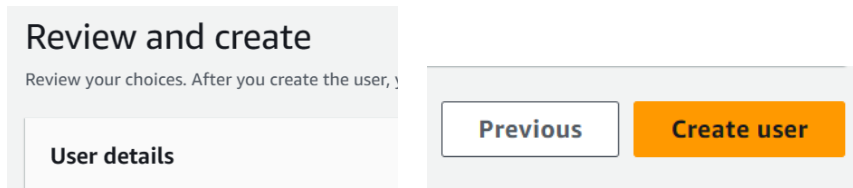
Permissions policies (2/135)

Choose one or more policies to attach to your new user.

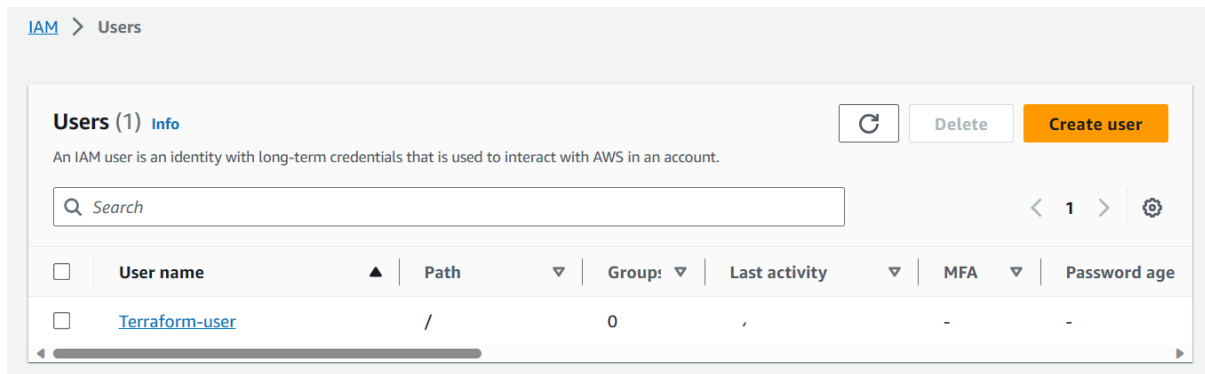
Filter by Type
All types

<input checked="" type="checkbox"/>	Policy name	Type
<input checked="" type="checkbox"/>	AmazonVPCFullAccess	AWS managed

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

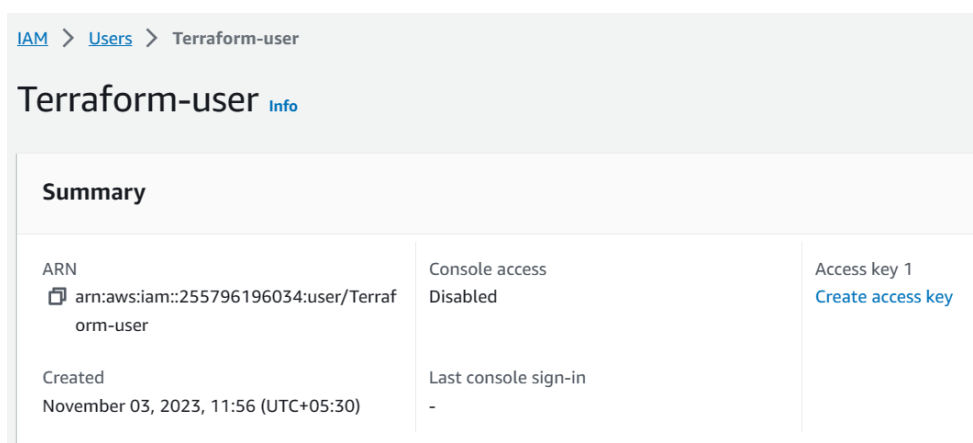


- The IAM user has been created successfully:



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:

Access key best practices & alternatives [Info](#)

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

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.

- now just assure the confirmation, click next:

Confirmation

☒ I understand the above recommendation and want to proceed to create an access key.

Cancel

Next

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

Set description tag - *optional* [Info](#)

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

for terraform purpose

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

Cancel

Previous

Create access key

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

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
AKIATXDVEGLBDSXFIWV	***** 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: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
AAAAB3NzaC1yc2EAAAADAQABAAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2c
EikKDEY0aIj41qgxMCP/iteneqXSIFZBp5vizPvaoIR3Um9xK7PGow8giupG
n+EPuxIA4cDM4vz0q0kiMPHz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7

```

```
tgBNL0cYlWSYVvkz4G/fs1NfRPW5mYAM49f4fhtxPb5ok4Q2Lg9dPKVHO/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:

Instances (1) Info			
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>			
<div> <div>Instance state = running ✕</div> <div>Clear filters</div> </div>			
<input type="checkbox"/>	Name ✎	Instance ID	Instance state
<input type="checkbox"/>	PROJECT-4	i-09af4a63d3f83a318	Running

Key pairs:

Key pairs (2) Info	
<input type="text" value="Search"/>	
<input type="checkbox"/>	Name
<input type="checkbox"/>	keypair
<input type="checkbox"/>	mumbai_key

Security groups:

Security Groups (4) Info			
<input type="text" value="Filter security groups"/>			
<input type="checkbox"/>	Name	Security group ID	Security group name
<input type="checkbox"/>	-	sg-0f000c02505e076a5	DOCKER-SC
<input type="checkbox"/>	-	sg-007c2254605a3bc92	TERRAFORM-SC
<input type="checkbox"/>	-	sg-034f4386289538f29	KUBERNETES_SC
<input type="checkbox"/>	-	sg-0b93a0c536749bf63	default

VPC:

Your VPCs (1) Info			
<input type="text" value="Search"/>			
<input type="checkbox"/>	Name	VPC ID	State
<input type="checkbox"/>	-	vpc-04dc687e3ffd22a68	Available

Subnets:

Subnets (3) Info					
<input type="text" value="Find resources by attribute or tag"/>					
<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR
<input type="checkbox"/>	-	subnet-081475cc3e7371bd3	Available	vpc-04dc687e3ffd22a68	172.31.16.0/20
<input type="checkbox"/>	-	subnet-070aa370b8f53b227	Available	vpc-04dc687e3ffd22a68	172.31.32.0/20
<input type="checkbox"/>	-	subnet-0712d94e0638aca75	Available	vpc-04dc687e3ffd22a68	172.31.0.0/20

Route tables:

Route tables (1) Info	
<input type="text" value="Find resources by attribute or tag"/>	
<input type="checkbox"/>	Name
<input type="checkbox"/>	-
Route table ID	
rtb-0aa8cc37f12bcccc3	

Internet Gateway:

Internet gateways (1) Info			Actions
<input type="text" value="Search"/>			
<input type="checkbox"/>	Name	Internet gateway ID	State
<input type="checkbox"/>	-	igw-0fa214a20c9ce490e	Attached
		VPC ID	
		vpc-04dc687e3ffd22a68	

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_assoc: Creating...
aws_route_table_association.subnet_assoc: 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:


Instances (2) Info					
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>					
<div> <div>Instance state = running ✕</div> <div>Clear filters</div> </div>					
<input type="checkbox"/>	Name ✎	Instance ID	Instance state ▼	Instance type ▼	Status check
<input type="checkbox"/>	project-4	i-03be04e5e05321858	✔ Running 🔍 🔍	t2.nano	🕒 Initializing
<input type="checkbox"/>	PROJECT-4	i-09af4a63d3f83a318	✔ Running 🔍 🔍	t2.micro	✔ 2/2 checks passed

- Instance details:

Vpc & subnet are associated with the instance:

Instance: i-03be04e5e05321858 (project-4)	
Auto-assigned IP address  13.126.161.211 [Public IP]	VPC ID  vpc-0302d315cdb91aff0 (project-4-vpc) 🔗
IAM Role -	Subnet ID  subnet-0fe0a8f5e96a5a610 (project-4-public-subnet) 🔗

Key pair associated with the instance:

Instance: i-03be04e5e05321858 (project-4)	
Default	normal
AMI Launch index	Key pair assigned at launch
0	 project-4-key

Security group of the instance:

Instance: i-03be04e5e05321858 (project-4)

sg-02ff8b40dadff19a1 (project-4-sc)

Inbound rules

Filter rules

<1>

Name	Security group rule ID	Port range	Protocol	Source	Security group
-	sgr-06b7ed78c3a36ccf5	80	TCP	0.0.0.0/0	project-4-sc
-	sgr-02b0b0230d47f8084	22	TCP	0.0.0.0/0	project-4-sc

- Key pairs:

Key pairs (3) Info			
<input type="text" value="Search"/>			
<input type="checkbox"/>	Name ▼	Type ▼	Created
<input type="checkbox"/>	project-4-key	rsa	2023/11/03 17:50 GMT+5:30
<input type="checkbox"/>	keypair	rsa	2023/10/18 13:19 GMT+5:30
<input type="checkbox"/>	mumbai_key	rsa	2023/10/09 17:36 GMT+5:30

- **Security groups:**

Security Groups (6) Info

⌂

Actions ▾

Export security group

🔍 Filter security groups

<input type="checkbox"/>	Name ▾	Security group ID ▾	Security group name ▾	VPC ID ▾
<input type="checkbox"/>	-	sg-0f000c02505e076a5	DOCKER-SC	vpc-04dc687e3ffd22a68 🔗
<input type="checkbox"/>	-	sg-04dec50eac14c853e	default	vpc-0302d315cdb91aff0 🔗
<input type="checkbox"/>	-	sg-007c2254605a3bc92	TERRAFORM-SC	vpc-04dc687e3ffd22a68 🔗
<input type="checkbox"/>	-	sg-034f4386289538f29	KUBERNETES_SC	vpc-04dc687e3ffd22a68 🔗
<input type="checkbox"/>	project-4-sc	sg-02ff8b40dadff19a1	project-4-sc	vpc-0302d315cdb91aff0 🔗
<input type="checkbox"/>	-	sg-0b93a0c536749bf63	default	vpc-04dc687e3ffd22a68 🔗

- **VPC:**

Your VPCs (2) Info

🔍 Search

<input type="checkbox"/>	Name ▾	VPC ID ▾	State ▾	IPv4 CIDR
<input type="checkbox"/>	-	vpc-04dc687e3ffd22a68	✔ Available	172.31.0.0/16
<input type="checkbox"/>	project-4-vpc	vpc-0302d315cdb91aff0	✔ Available	10.0.0.0/16

- **Subnets:**

Subnets (4) Info

⌂

Actions

🔍 Find resources by attribute or tag

<input type="checkbox"/>	Name ▾	Subnet ID ▾	State ▾	VPC ▾
<input type="checkbox"/>	-	subnet-081475cc3e7371bd3	✔ Available	vpc-04dc687e3ffd22a68
<input type="checkbox"/>	-	subnet-070aa370b8f53b227	✔ Available	vpc-04dc687e3ffd22a68
<input type="checkbox"/>	-	subnet-0712d94e0638aca75	✔ Available	vpc-04dc687e3ffd22a68
<input type="checkbox"/>	project-4-public-subnet	subnet-0fe0a8f5e96a5a610	✔ Available	vpc-0302d315cdb91aff0 proje...

- **Route table:**

Route tables (3) Info

🔍 Find resources by attribute or tag

<input type="checkbox"/>	Name ▾	Route table ID ▾	Explicit subnet associati... ▾	Ec ▾
<input type="checkbox"/>	-	rtb-0aa8cc37f12bcccc3	-	-
<input type="checkbox"/>	project-4-route-table	rtb-08bff820e8f7b2bde	subnet-0fe0a8f5e96a5a6...	-
<input type="checkbox"/>	-	rtb-07bfab98574917dbb	-	-

- **Internet gateway:**

Internet gateways (2) Info

⌂

Actions ▾

Create intern

🔍 Search

<input type="checkbox"/>	Name ▾	Internet gateway ID ▾	State ▾	VPC ID
<input type="checkbox"/>	-	igw-0fa214a20c9ce490e	✔ Attached	vpc-04dc687e3ffd22a68
<input type="checkbox"/>	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-sg: Modifying... [id=sg-02ff8b40dadff19a1]
aws_security_group.project-4-sg: 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:

Instances (1/2) Info						Refresh	Connect	Instance status check
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>								
Instance state = running X			Clear filters					
<input checked="" type="checkbox"/>	Name ✎	Instance ID	Instance state ▼	Instance type ▼	Status check			
<input checked="" type="checkbox"/>	project-4	i-03be04e5e05321858	Running 🔍 🔍	t2.micro	🔍 Initializing			

➔ Checking the security group:

Instance: i-03be04e5e05321858 (project-4)						Settings X
<input type="text" value="Filter rules"/>						< 1 >
Name	Security group rule ID	Port range	Protocol	Source	Security group	
-	sgr-03479896991959967	443	TCP	0.0.0.0/0	project-4-sc 🔗	
-	sgr-06b7ed78c3a36ccf5	80	TCP	0.0.0.0/0	project-4-sc 🔗	
-	sgr-02b0b0230d47f8084	22	TCP	0.0.0.0/0	project-4-sc 🔗	

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>
