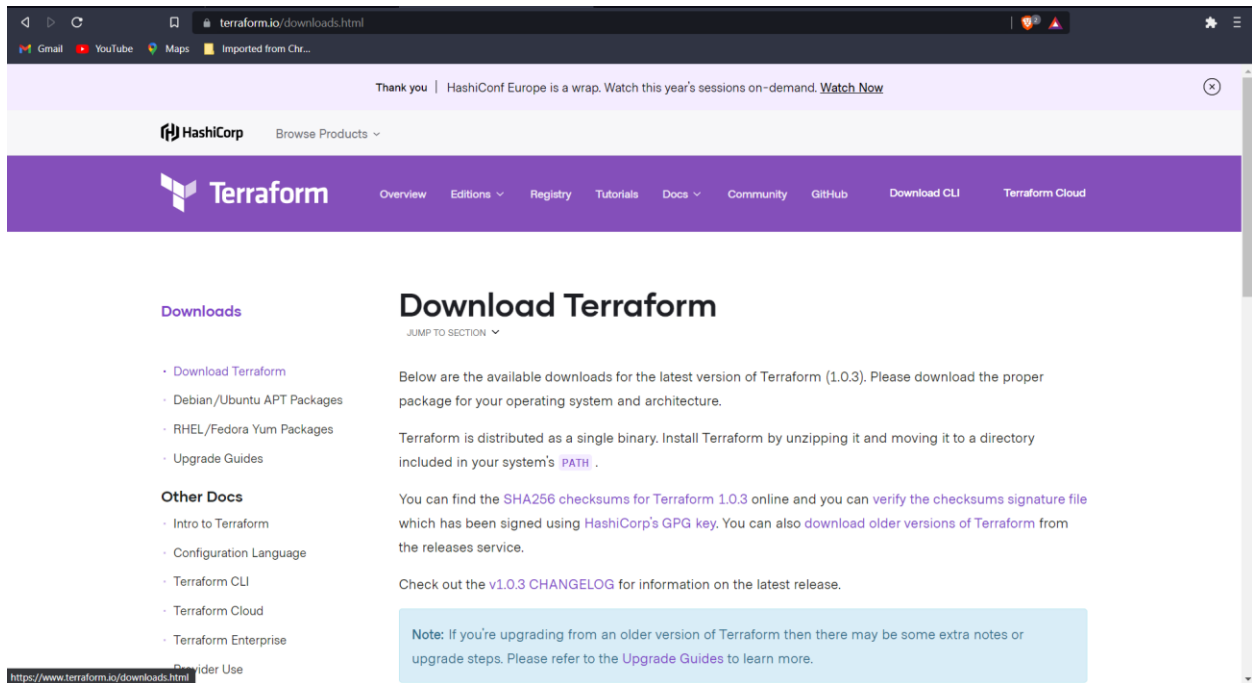


Assignment 5: Terraform installation

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently.

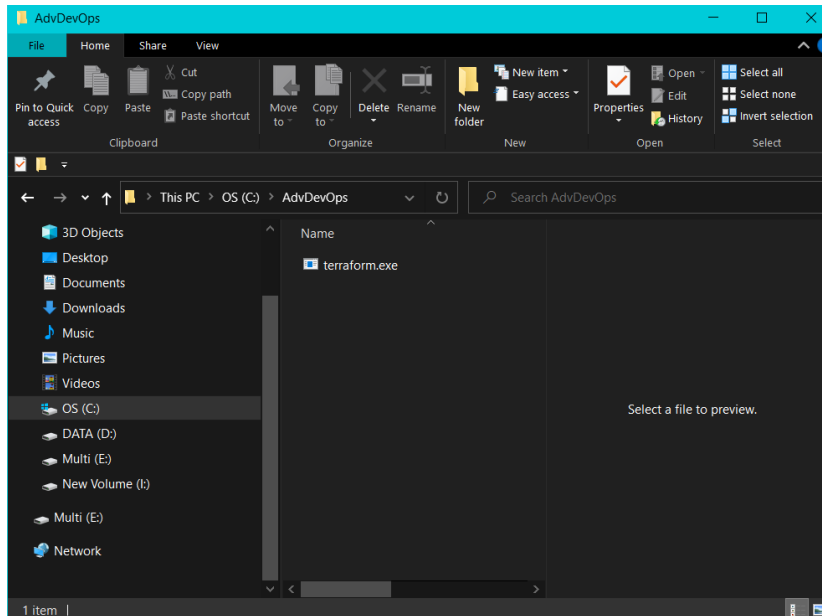
STEPS For Installation-

Google Search – Terraform DOWNLOAD



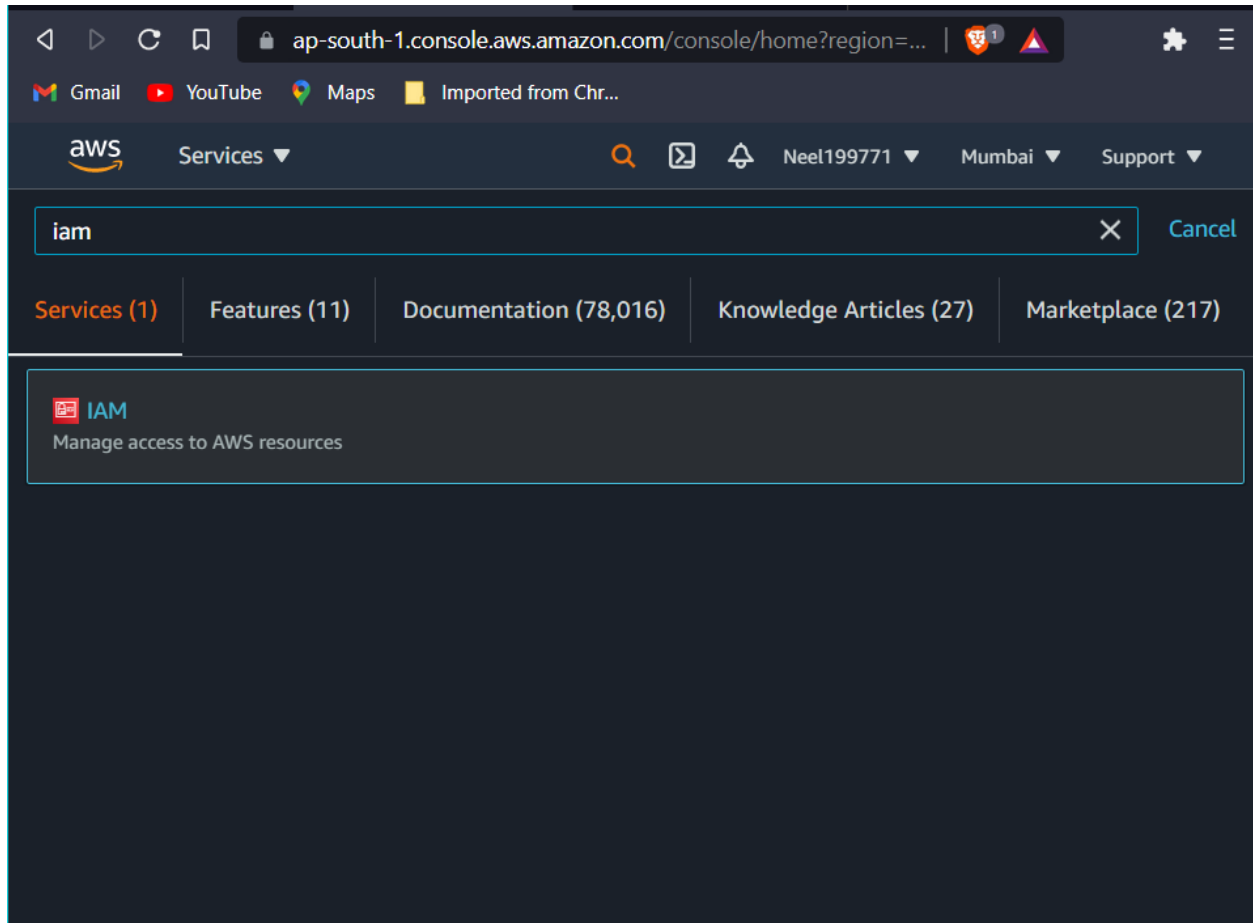
Create a Folder in Your C drive Named AdvDevops

Then Extract The downloaded file in The C drive Folder Named AdvDevops Shown Below

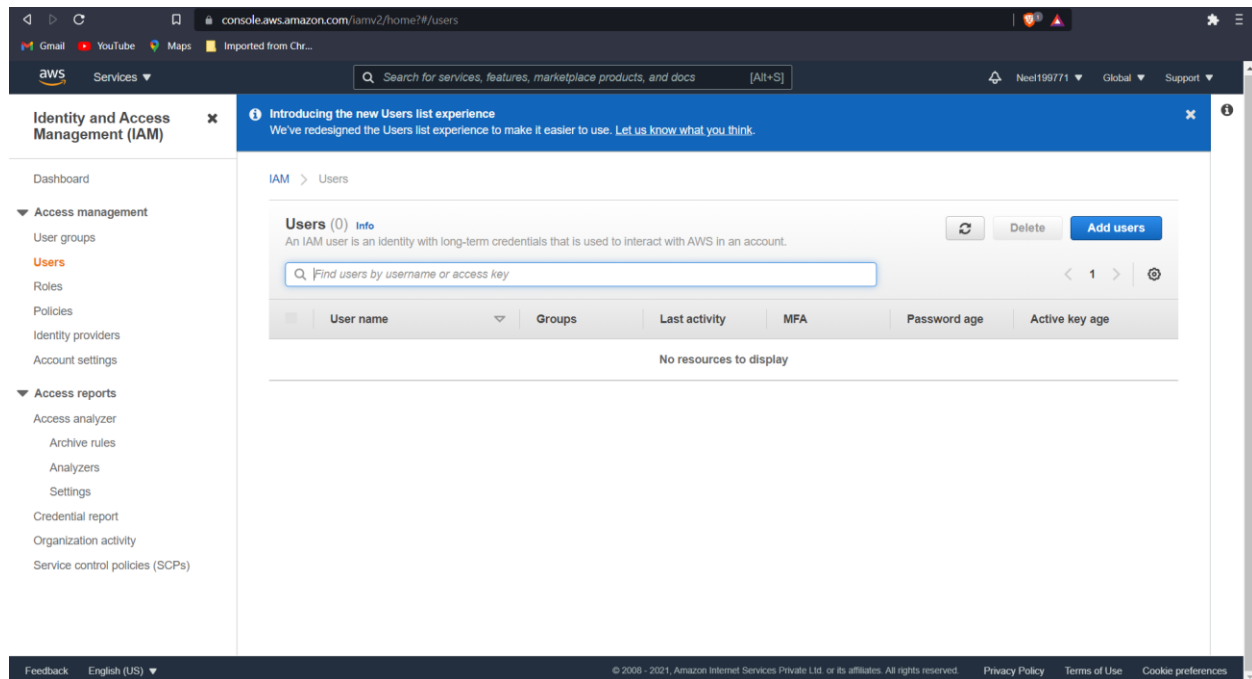


Till the time ,Open And Login to your AWS console-

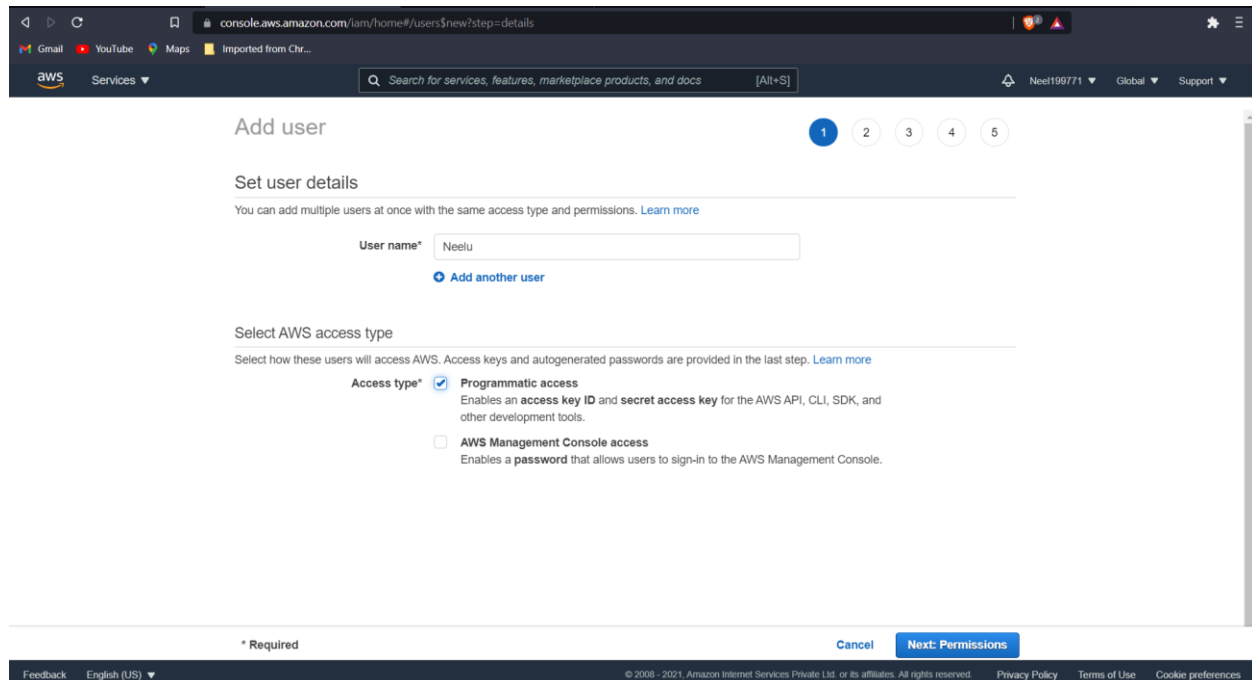
And search IAM and click on it



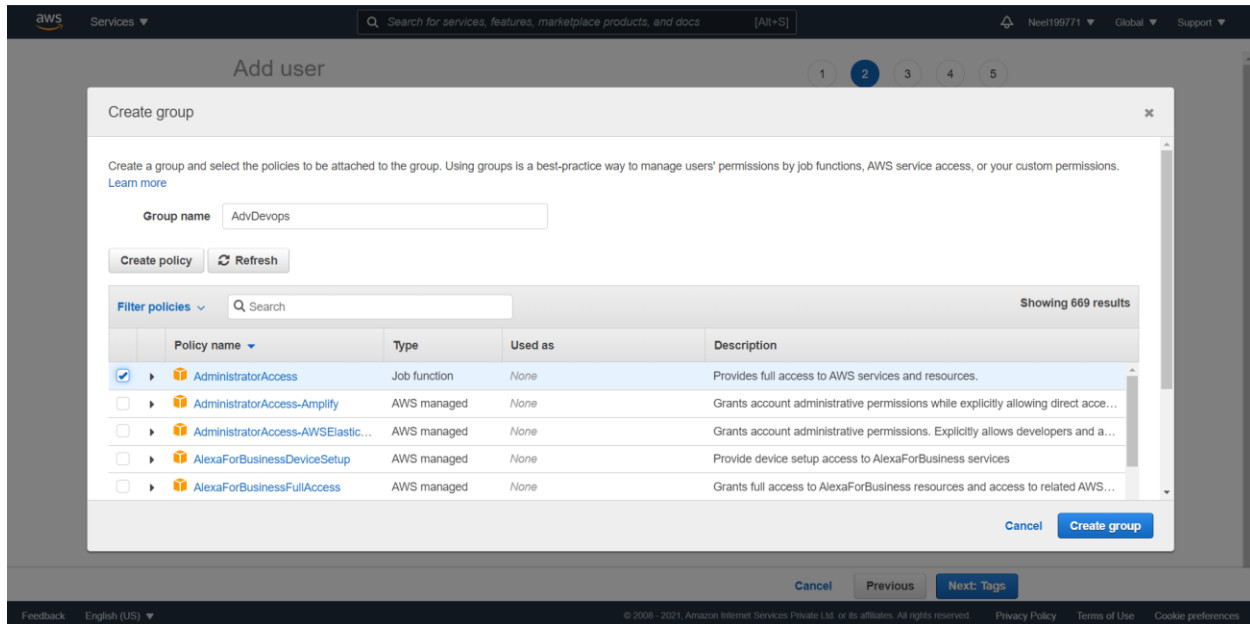
Now click on Add Users in The User Section as shown in the image



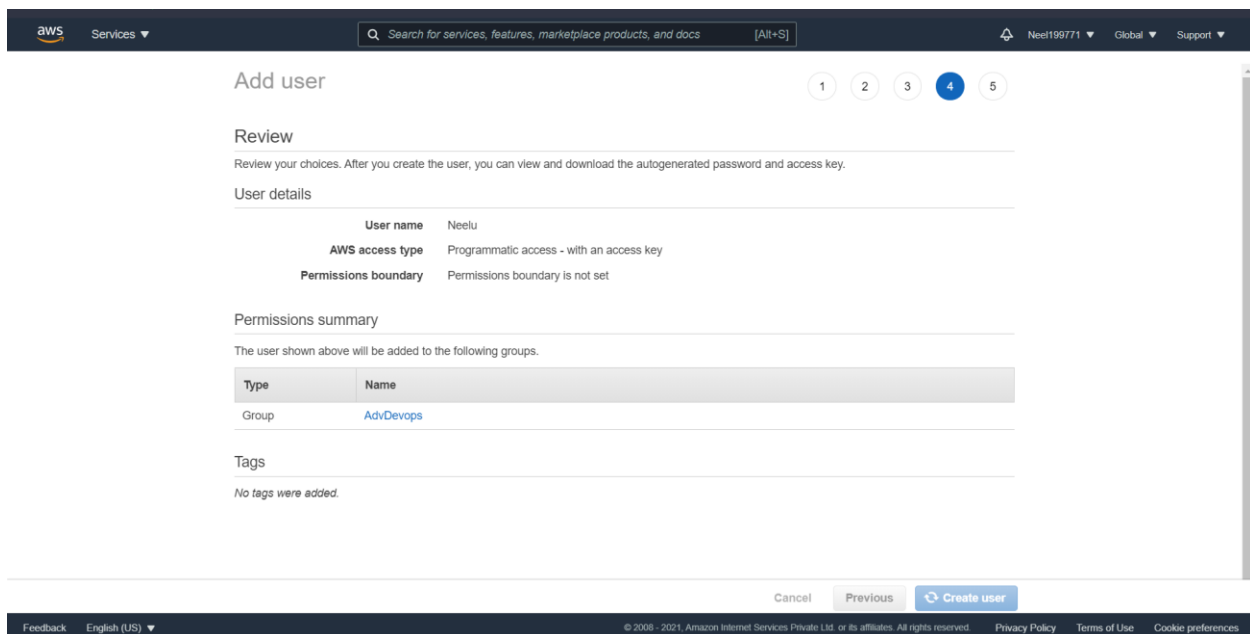
Now give any name For username And Check The Programmatic Access field shown below



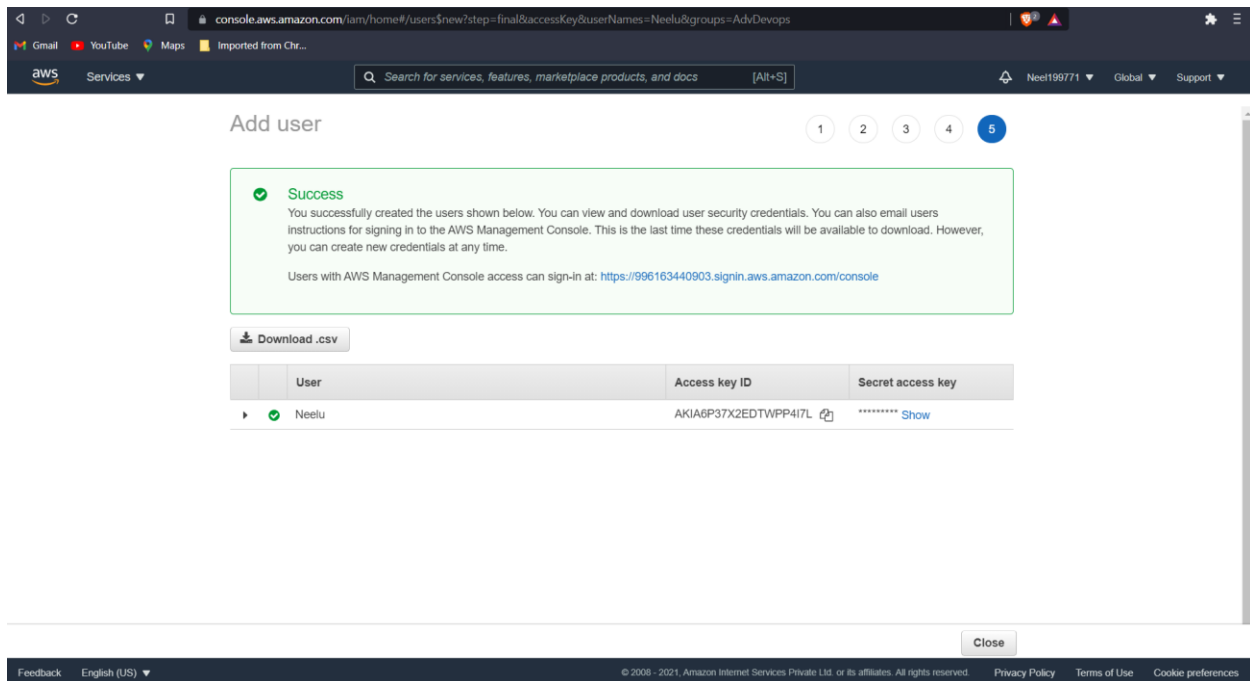
Add Group name and Check the first Policy Name



Don't add tags



Now Download .csv file

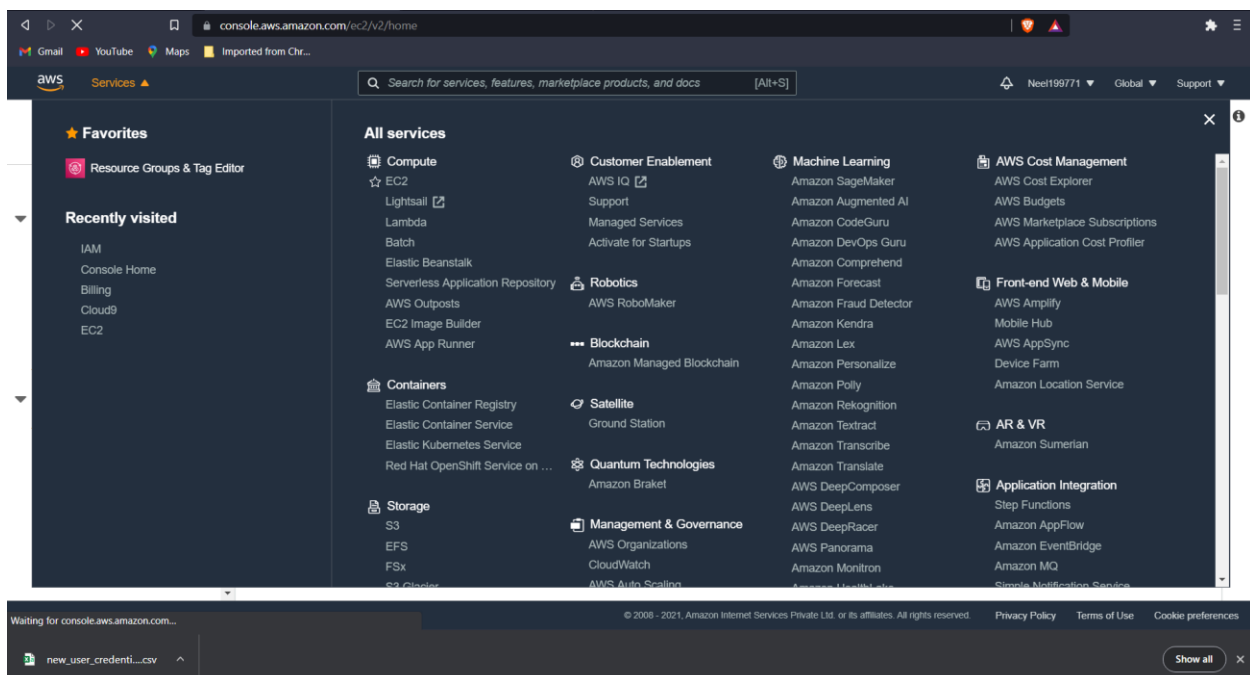


The screenshot shows the AWS IAM console 'Add user' page. A success message indicates that users were created successfully. Below the message is a 'Download .csv' button and a table of the created users.

User	Access key ID	Secret access key
Neelu	AKIA6P37X2EDTWP4I7L	***** Show

Close

Go to services and Ec2



The screenshot shows the AWS Management Console 'All services' page. The left sidebar contains 'Favorites' and 'Recently visited' sections. The main area displays a grid of service categories and their respective services.

Favorites

- Resource Groups & Tag Editor

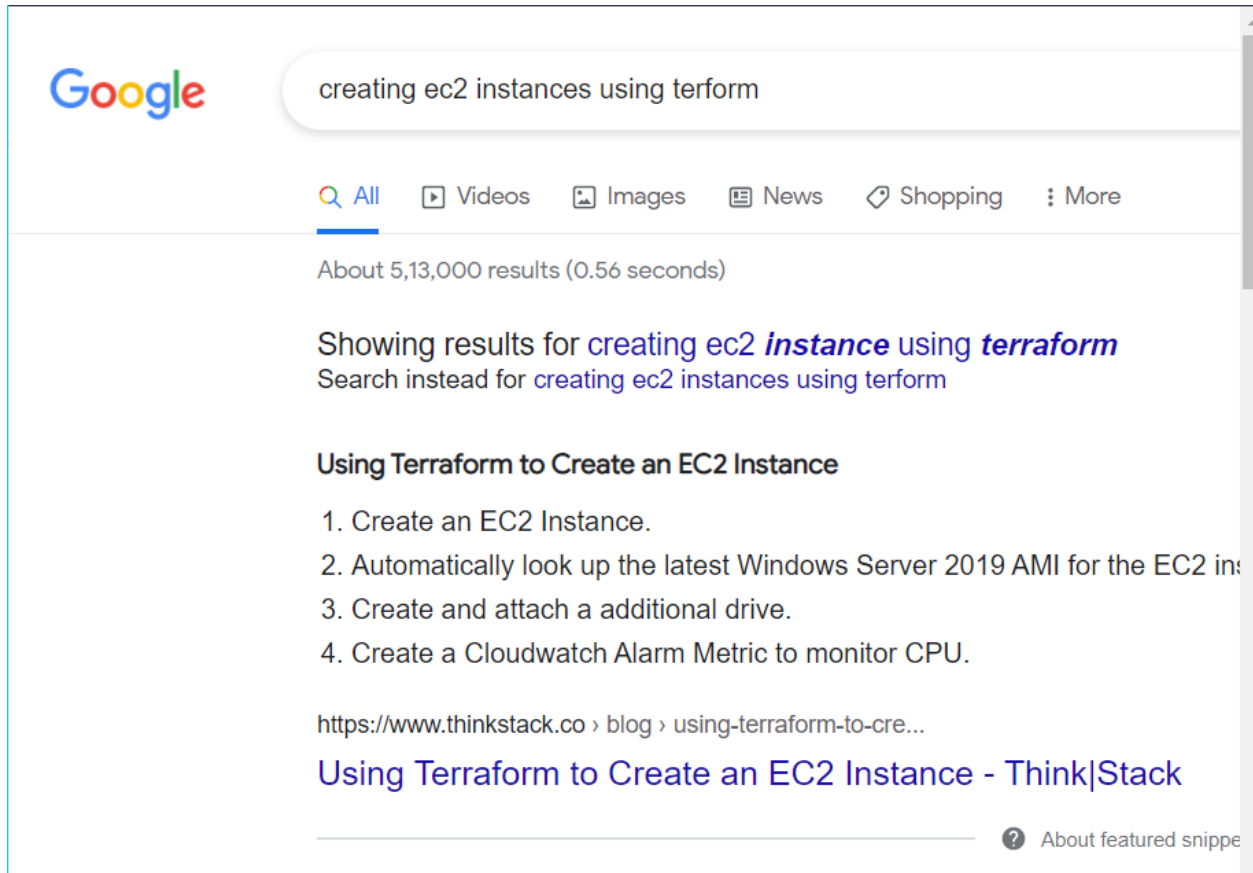
Recently visited

- IAM
- Console Home
- Billing
- Cloud9
- EC2

All services

- Compute**
 - EC2
 - Lightsail
 - Lambda
 - Batch
 - Elastic Beanstalk
 - Serverless Application Repository
 - AWS Outposts
 - EC2 Image Builder
 - AWS App Runner
- Containers**
 - Elastic Container Registry
 - Elastic Container Service
 - Elastic Kubernetes Service
 - Red Hat OpenShift Service on AWS
- Storage**
 - S3
 - EFS
 - FSx
- Customer Enablement**
 - AWS IQ
 - Support
 - Managed Services
 - Activate for Startups
- Robotics**
 - AWS RoboMaker
- Blockchain**
 - Amazon Managed Blockchain
- Satellite**
 - Ground Station
- Quantum Technologies**
 - Amazon Braket
- Machine Learning**
 - Amazon SageMaker
 - Amazon Augmented AI
 - Amazon CodeGuru
 - Amazon DevOps Guru
 - Amazon Comprehend
 - Amazon Forecast
 - Amazon Fraud Detector
 - Amazon Kendra
 - Amazon Lex
 - Amazon Personalize
 - Amazon Polly
 - Amazon Rekognition
 - Amazon Textract
 - Amazon Transcribe
 - Amazon Translate
 - AWS DeepComposer
 - AWS DeepLens
 - AWS DeepRacer
 - AWS Panorama
 - Amazon Monitron
- AWS Cost Management**
 - AWS Cost Explorer
 - AWS Budgets
 - AWS Marketplace Subscriptions
 - AWS Application Cost Profiler
- Front-end Web & Mobile**
 - AWS Amplify
 - Mobile Hub
 - AWS AppSync
 - Device Farm
 - Amazon Location Service
- AR & VR**
 - Amazon Sumerian
- Application Integration**
 - Step Functions
 - Amazon AppFlow
 - Amazon EventBridge
 - Amazon MQ
 - Simple Notification Service
- Management & Governance**
 - AWS Organizations
 - CloudWatch
 - AWS Auto Scaling

Again google search the following terms

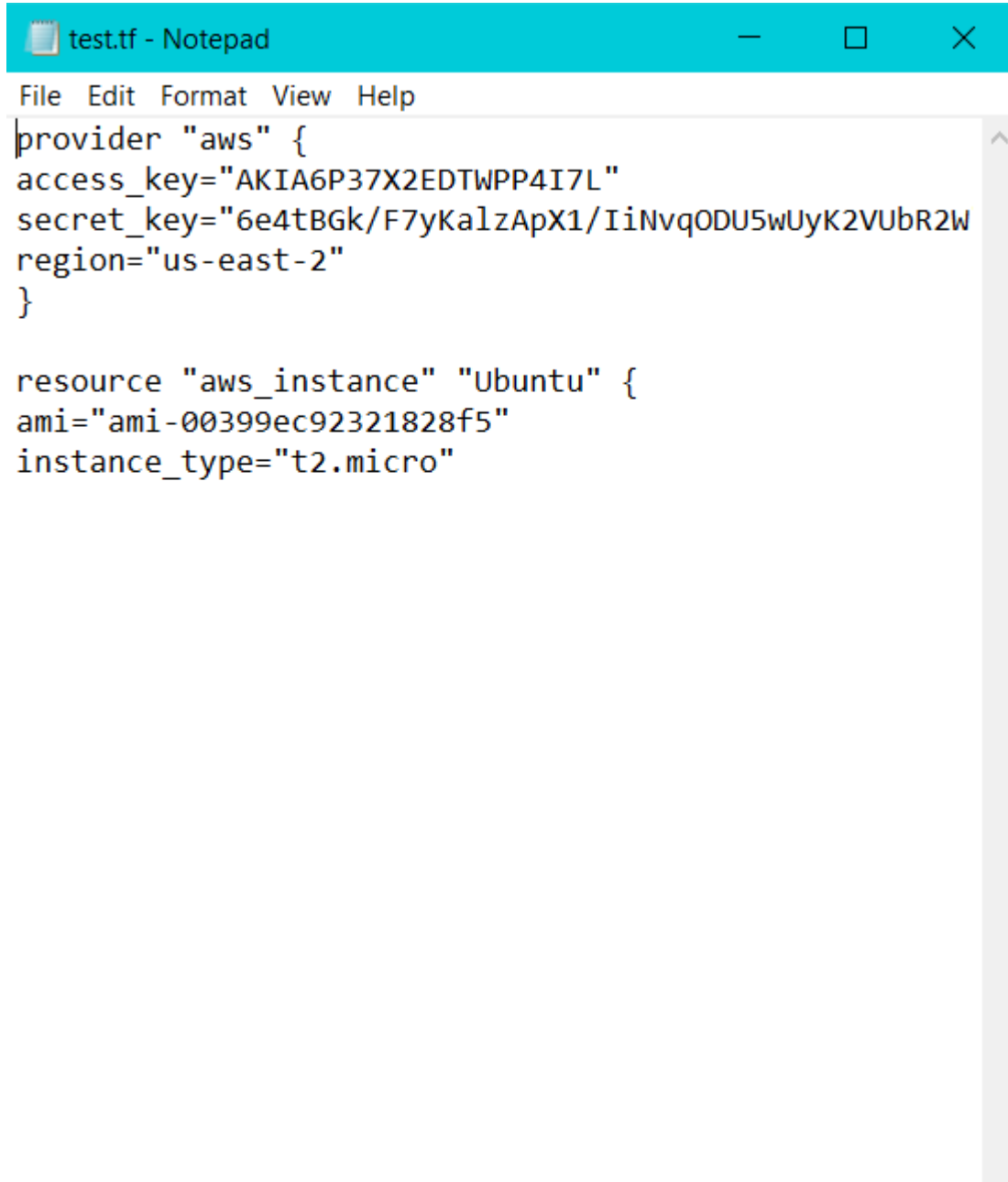


Created a folder Name Terraform Scripts in the C drive where the AdvDevops folder was created



Now Go to note pad And Type the below Details properly But before It Just Change the ACCESS KEY AND SECRET KEY TO THE ONE IN YOUR .csv File .

Set region same as below if you want MUMBAI as your region.

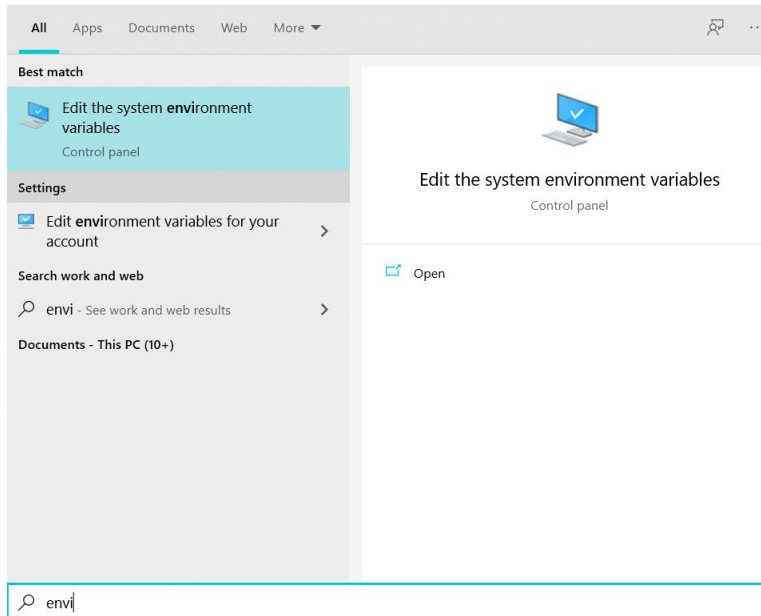


```
test.tf - Notepad
File Edit Format View Help
provider "aws" {
  access_key="AKIA6P37X2EDTWPP4I7L"
  secret_key="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"
  region="us-east-2"
}

resource "aws_instance" "Ubuntu" {
  ami="ami-00399ec92321828f5"
  instance_type="t2.micro"
```

Now search EDIT THE SYSTEM ENVIRONMENT VARIABLES in your windows search.

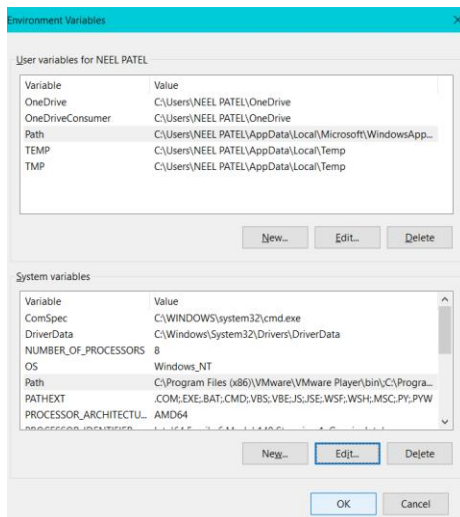
Open it



Now click on PATH OF USER VARIABLES, then click on Edit option

Now go to edit and then add new path C:\AdvDevOps

Repeat same procedure for system variables.



Now Open Command Prompt and then paste the path of Terraform script

Eg. CD C:\Terraform Script as shown below

Now type Terraform Init command

```
C:\Teraform Script>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.52.0...
- Installed hashicorp/aws v3.52.0 (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.
```

Then if there are no errors type Terraform Plan as shown below (type YES when command prompt ask)

```
C:\Teraform Script>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.Ubuntu will be created
+ resource "aws_instance" "Ubuntu" {
  + ami              = "ami-0c1a7f89451184c8b"
  + 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_termination = (known after apply)
  + ebs_optimized     = (known after apply)
  + get_password_data  = false
  + host_id            = (known after apply)
```

Now Finally Type Terraform Apply

```
C:\Terraform Script>terraform apply
```

```
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.Ubuntu will be created
```

```
+ resource "aws_instance" "Ubuntu" {  
  + ami              = "ami-0c1a7f89451184c8b"  
  + arn              = (known after apply)  
  + associate_public_ip_address = (known after apply)  
  + availability_zone = (known after apply)  
  + cpu_core_count    = (known after apply)
```

```
Plan: 1 to add, 0 to change, 0 to destroy.
```

```
Do you want to perform these actions?
```

```
Terraform will perform the actions described above.
```

```
Only 'yes' will be accepted to approve.
```

```
Enter a value: yes
```

```
aws_instance.Ubuntu: Creating...
```

```
aws_instance.Ubuntu: Still creating... [10s elapsed]
```

```
aws_instance.Ubuntu: Still creating... [20s elapsed]
```

```
aws_instance.Ubuntu: Still creating... [30s elapsed]
```

```
aws_instance.Ubuntu: Creation complete after 31s [id=i-0eac948a456860494]
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Now go to EC2 and check that is an instance created by the name of UBUNTU and is it in running status or not If it is in Running Status then Come back to Command prompt And Terminate the Instance by

Typing - Terraform destroy

```
Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_instance.Ubuntu: Destroying... [id=i-0eac948a456860494]
aws_instance.Ubuntu: Still destroying... [id=i-0eac948a456860494, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0eac948a456860494, 20s elapsed]
aws_instance.Ubuntu: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.

C:\Terraform Script>_
```

Now go back to EC2 if the instance is terminated, if yes then logout of the Aws Console.

And close the command prompt!