Name → Siddharth Sutar
ROLL NO → 323054
PRN → 22010517
Subject → Cloud Computing

Assignment No 5

Write IaC using terraform to create EC2 machine on AWS or azure or google cloud. (Compulsory to use Input and output variable files)

AIM

→ Use terraform to create an EC2 instance

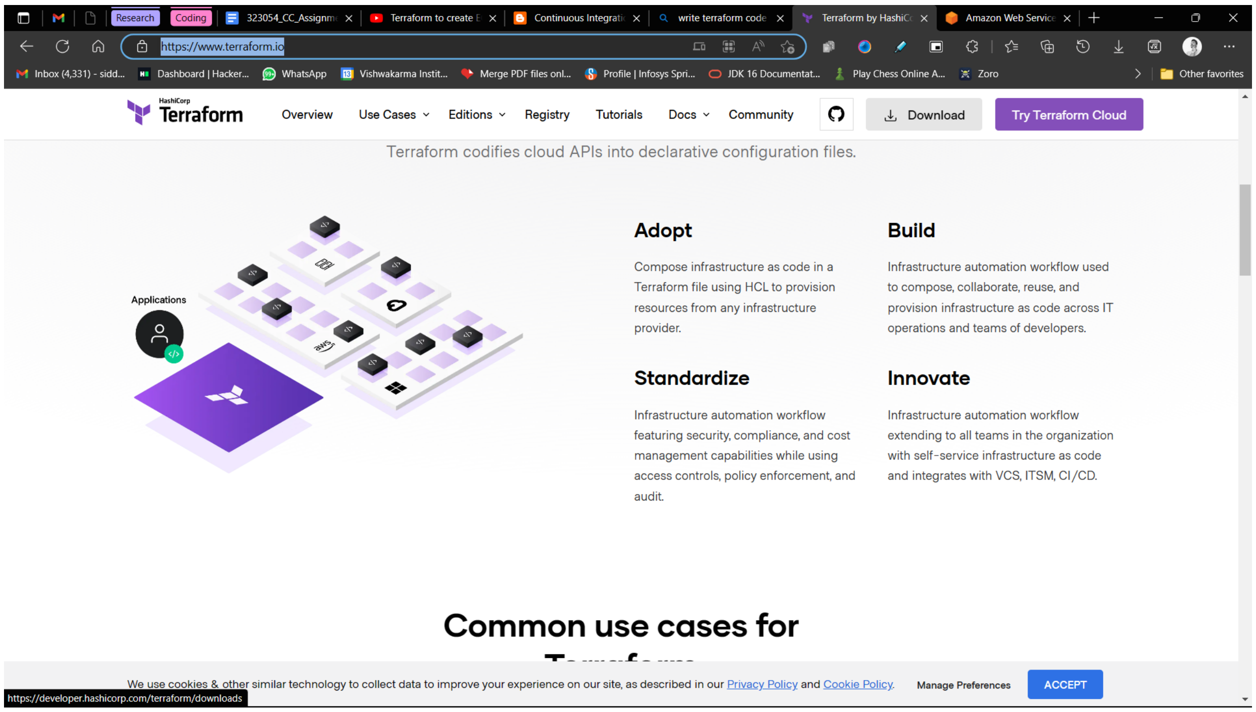
Theory

→ What is terraform?

→ Terraform Cloud enables infrastructure automation for provisioning, compliance, and management of any cloud, datacenter, and service.
→ It is an open-source tool for provisioning and managing cloud infrastructure. Terraform can provision resources on any cloud platform.
→ Terraform allows you to create infrastructure in configuration files(tf files) that describe the topology of cloud resources.
→ These resources include virtual machines, storage accounts, and networking interfaces or virtually any resource you want

Step-by-step screenshot to install and configure Terraform

1. Download terraform from the website



Terraform codifies cloud APIs into declarative configuration files.

Adopt

Compose infrastructure as code in a Terraform file using HCL to provision resources from any infrastructure provider.

Build

Infrastructure automation workflow used to compose, collaborate, reuse, and provision infrastructure as code across IT operations and teams of developers.

Standardize

Infrastructure automation workflow featuring security, compliance, and cost management capabilities while using access controls, policy enforcement, and audit.

Innovate

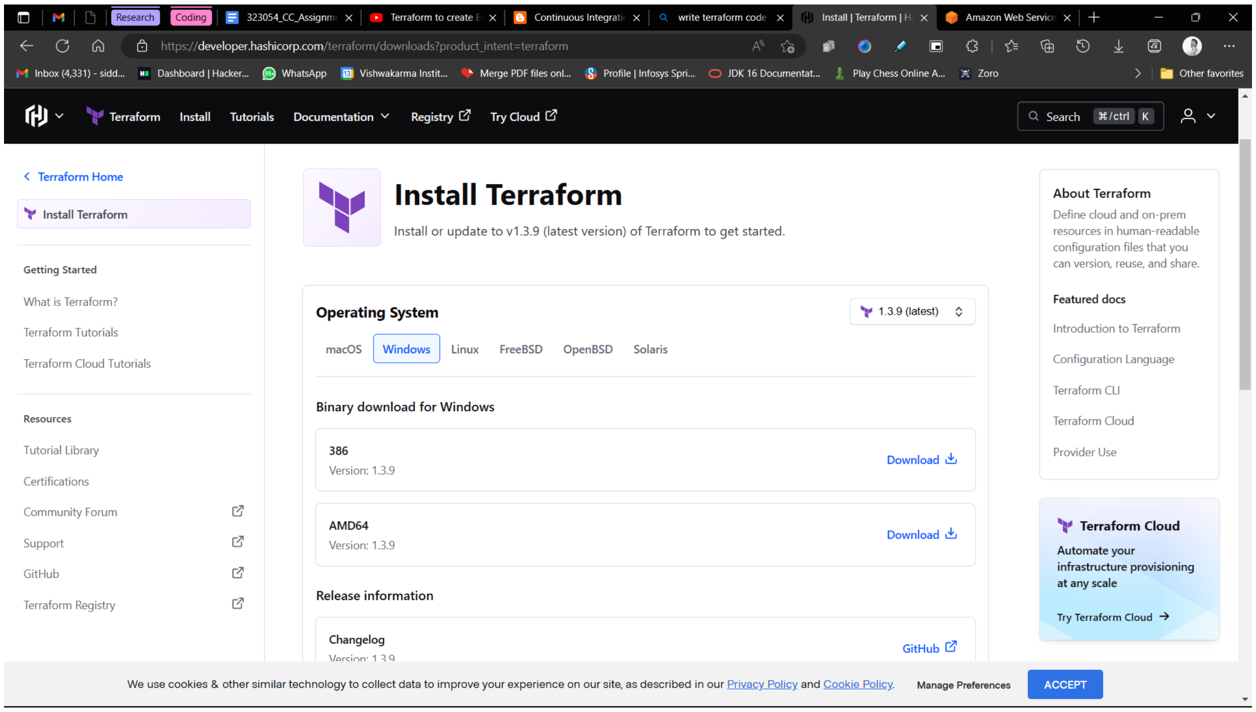
Infrastructure automation workflow extending to all teams in the organization with self-service infrastructure as code and integrates with VCS, ITSM, CI/CD.

Common use cases for

We use cookies & other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie Policy. Manage Preferences ACCEPT

https://developer.hashicorp.com/terraform/downloads

2. Install according to your machine



Install Terraform

Install or update to v1.3.9 (latest version) of Terraform to get started.

Operating System — 1.3.9 (latest)

macOS | Windows | Linux | FreeBSD | OpenBSD | Solaris

Binary download for Windows

386 — Version: 1.3.9 — Download

AMD64 — Version: 1.3.9 — Download

Release information

Changelog — Version: 1.3.9 — GitHub

About Terraform

Define cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.

Featured docs: Introduction to Terraform, Configuration Language, Terraform CLI, Terraform Cloud, Provider Use

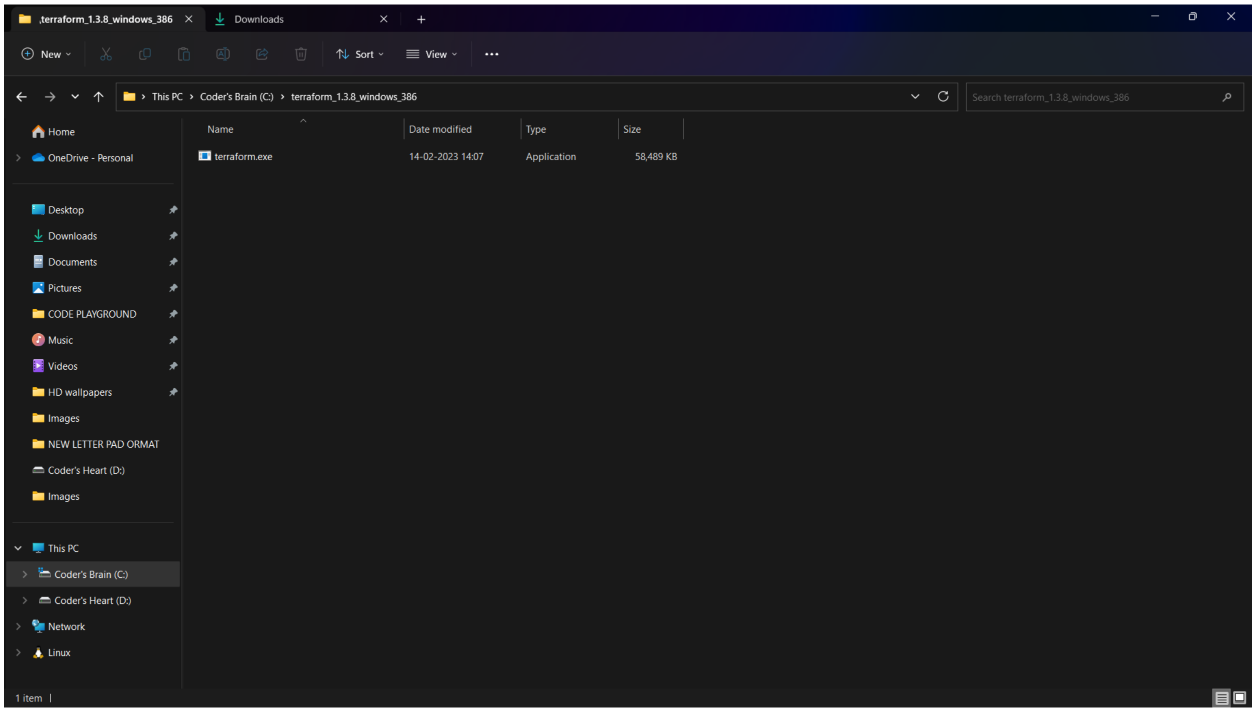
Terraform Cloud

Automate your infrastructure provisioning at any scale

Try Terraform Cloud →

We use cookies & other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie Policy. Manage Preferences ACCEPT

3. Download and extract it somewhere

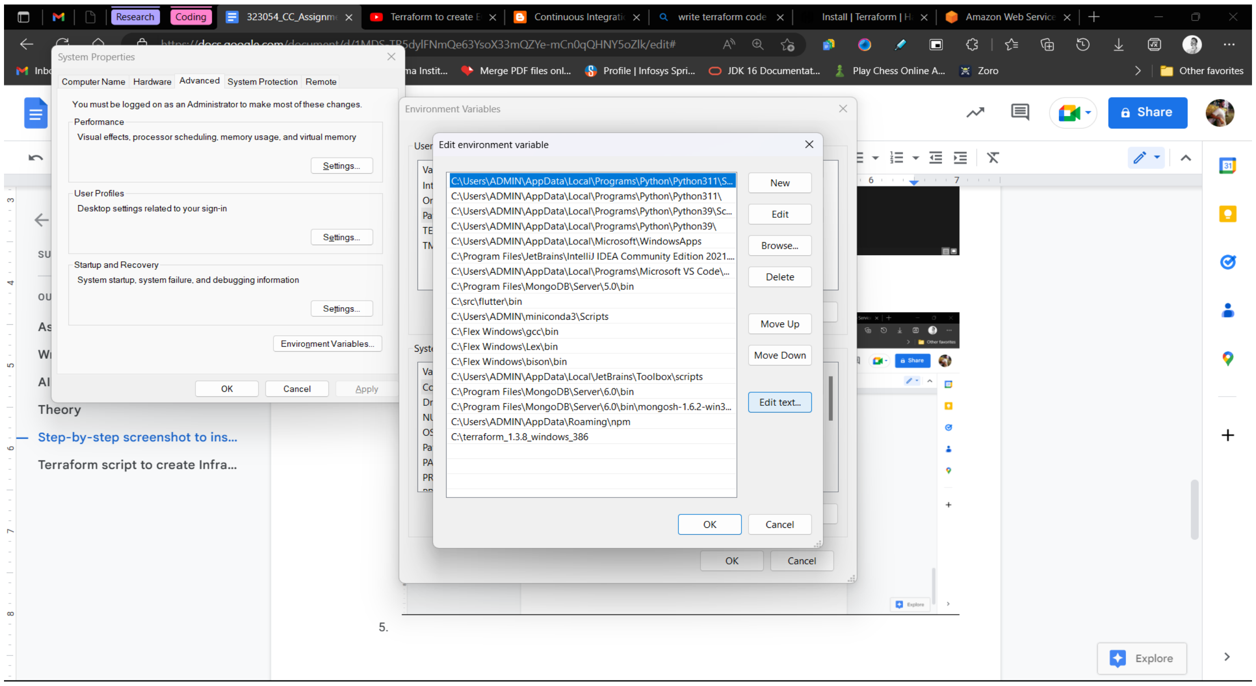


This PC > Coder's Brain (C:) > terraform_1.3.8_windows_386

Name	Date modified	Type	Size
terraform.exe	14-02-2023 14:07	Application	58,489 KB

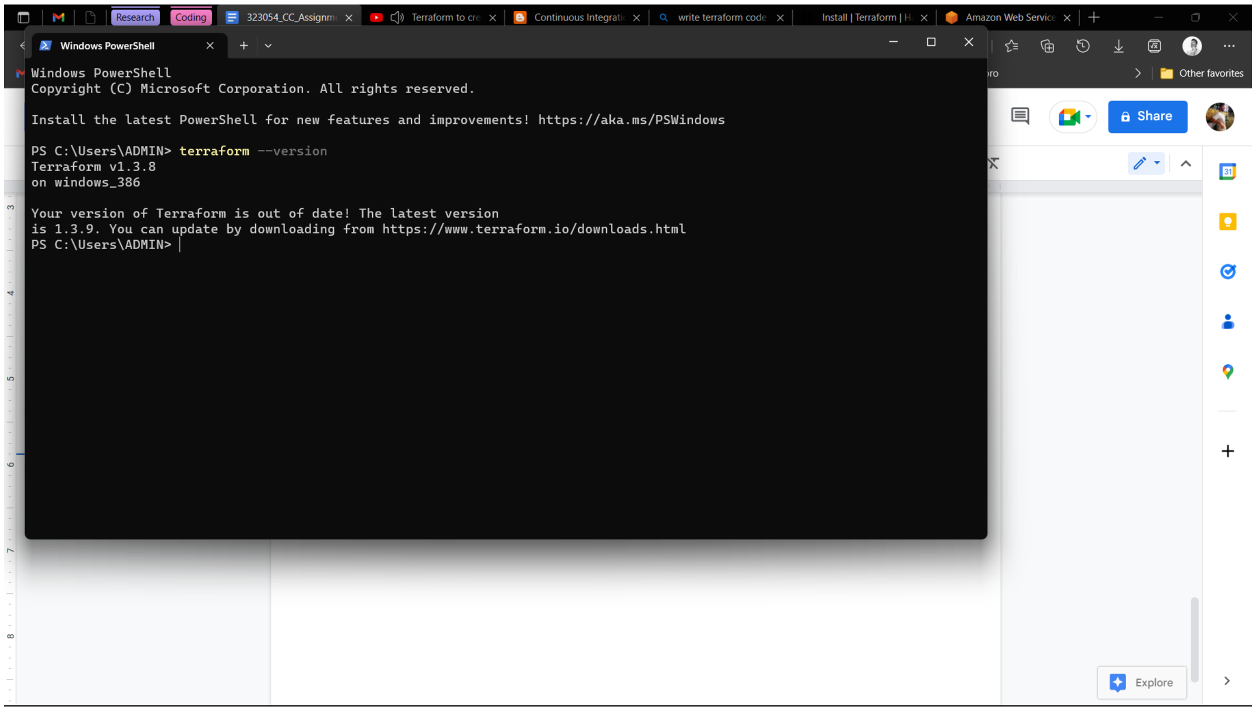
1 item

4. Add this folder to the path & check its version using (terraform --version)



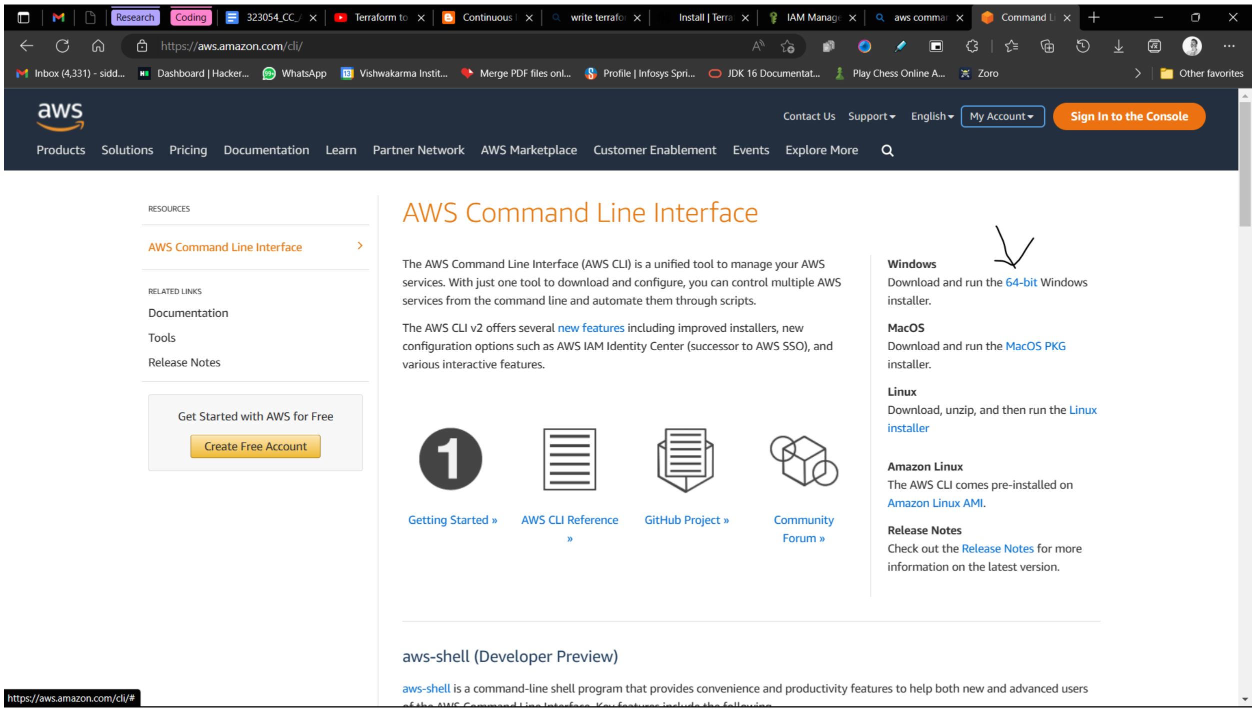
Edit environment variable

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python311\S...
C:\Users\ADMIN\AppData\Local\Programs\Python\Python311\
C:\Users\ADMIN\AppData\Local\Programs\Python\Python39\Sc...
C:\Users\ADMIN\AppData\Local\Programs\Python\Python39\
C:\Users\ADMIN\AppData\Local\Microsoft\WindowsApps
C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021...
C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code\...
C:\Program Files\MongoDB\Server\5.0\bin
C:\src\flutter\bin
C:\Users\ADMIN\miniconda3\Scripts
C:\Flex Windows\gcc\bin
C:\Flex Windows\Lex\bin
C:\Flex Windows\bison\bin
C:\Users\ADMIN\AppData\Local\JetBrains\Toolbox\scripts
C:\Program Files\MongoDB\Server\6.0\bin
C:\Program Files\MongoDB\Server\6.0\bin\mongosh-1.6.2-win3...
C:\Users\ADMIN\AppData\Roaming\npm
C:\terraform_1.3.8_windows_386
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ADMIN> terraform --version
Terraform v1.3.8
on windows_386

Your version of Terraform is out of date! The latest version
is 1.3.9. You can update by downloading from https://www.terraform.io/downloads.html
PS C:\Users\ADMIN>
```

5. Download [AWS command line tool](https://aws.amazon.com/cli/) & install it



AWS Command Line Interface

The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

The AWS CLI v2 offers several new features including improved installers, new configuration options such as AWS IAM Identity Center (successor to AWS SSO), and various interactive features.

Windows
Download and run the 64-bit Windows installer.

MacOS
Download and run the MacOS PKG installer.

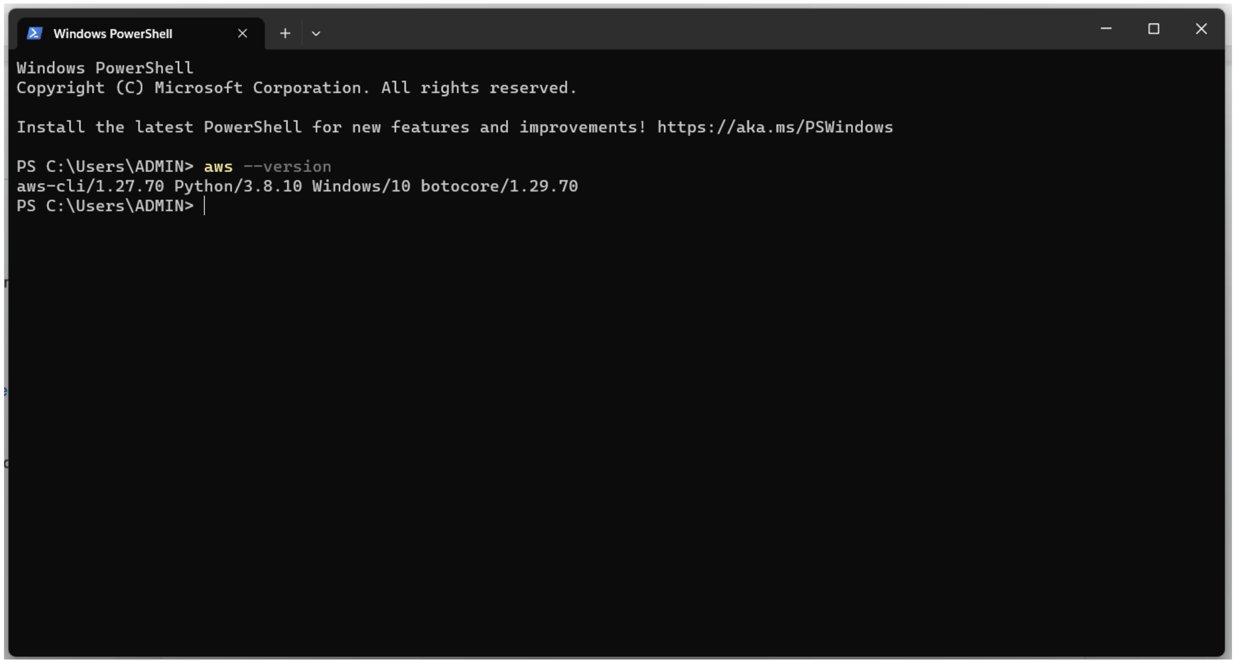
Linux
Download, unzip, and then run the Linux installer

Amazon Linux
The AWS CLI comes pre-installed on Amazon Linux AMI.

Release Notes
Check out the Release Notes for more information on the latest version.

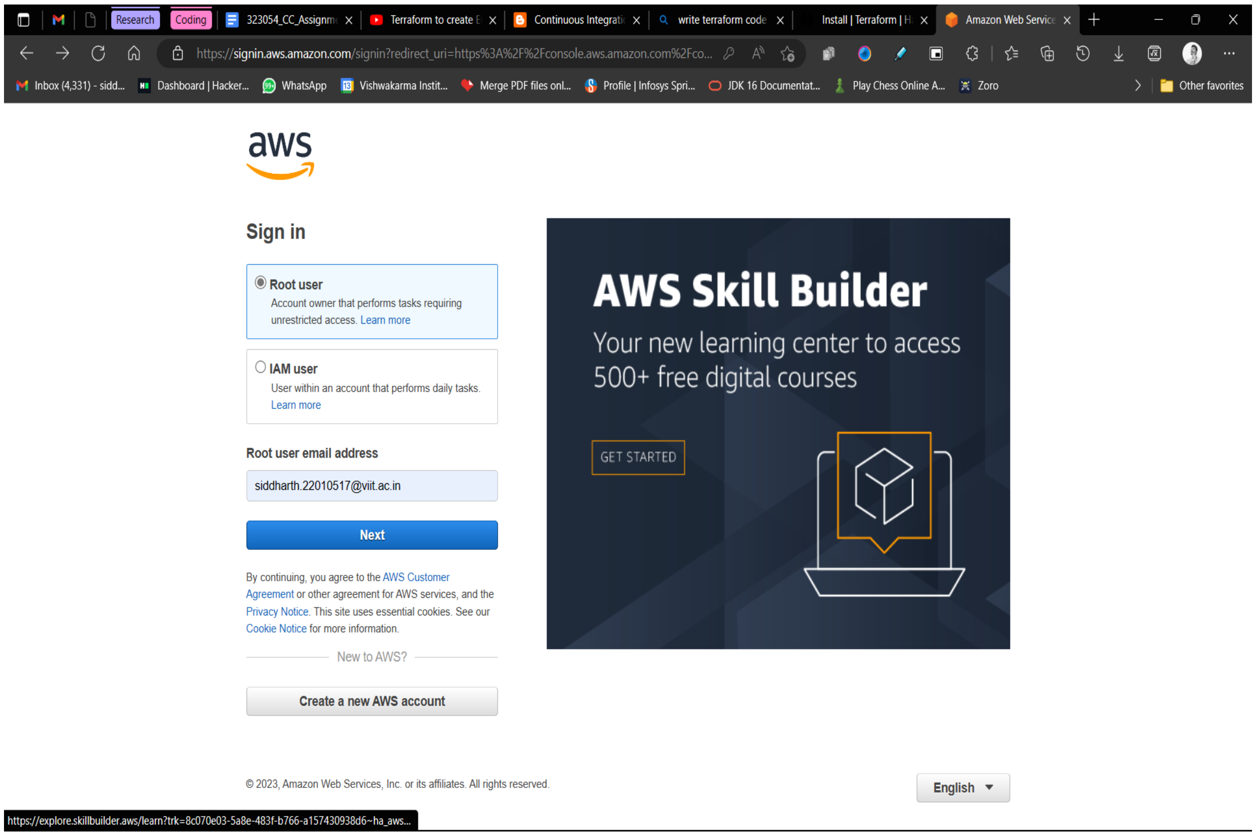
aws-shell (Developer Preview)

aws-shell is a command-line shell program that provides convenience and productivity features to help both new and advanced users



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ADMIN> aws --version
aws-cli/1.27.70 Python/3.8.10 Windows/10 botocore/1.29.70
PS C:\Users\ADMIN>
```

6. Login to aws & find IAM service


https://signin.aws.amazon.com/signin?redirect_uri=https%3A%2F%2Fconsole.aws.amazon.com%2Fco...
aws
AWS Skill Builder
Your new learning center to access 500+ free digital courses
GET STARTED
English

Sign in

Root user
Account owner that performs tasks requiring unrestricted access. Learn more

IAM user
User within an account that performs daily tasks. Learn more

Root user email address

siddharth.22010517@viit.ac.in

Next

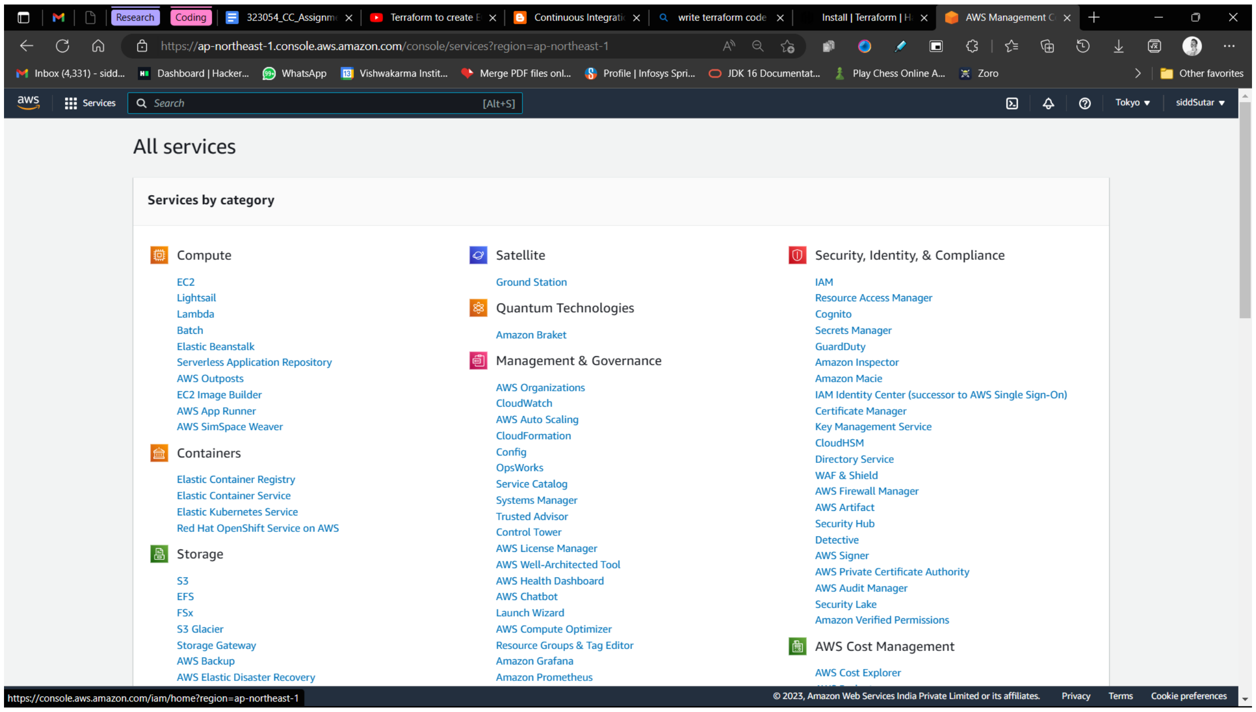
By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information.

New to AWS?

Create a new AWS account

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Click on all services & find + click IAM in security, Identity & compliance



All services

Services by category

Compute
EC2
Lightsail
Lambda
Batch
Elastic Beanstalk
Serverless Application Repository
AWS Outposts
EC2 Image Builder
AWS App Runner
AWS SimSpace Weaver

Containers
Elastic Container Registry
Elastic Container Service
Elastic Kubernetes Service
Red Hat OpenShift Service on AWS

Storage
S3
EFS
FSx
S3 Glacier
Storage Gateway
AWS Backup
AWS Elastic Disaster Recovery

Satellite
Ground Station

Quantum Technologies
Amazon Braket

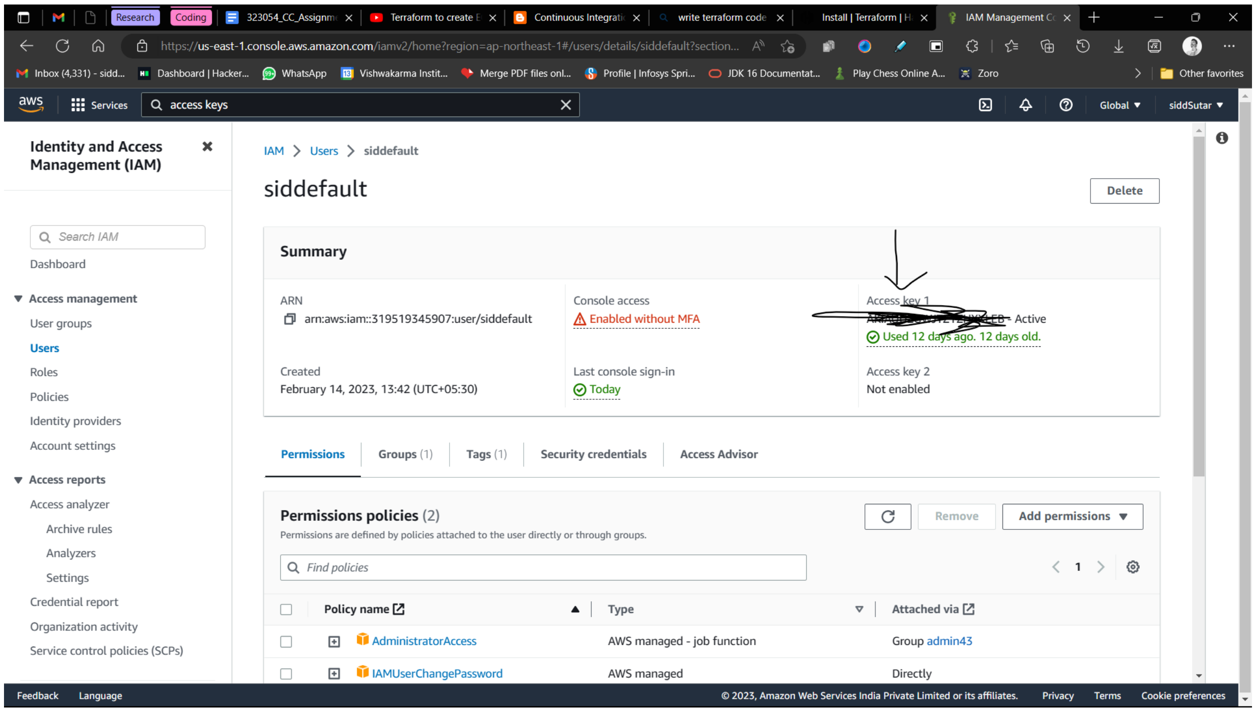
Management & Governance
AWS Organizations
CloudWatch
AWS Auto Scaling
CloudFormation
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Control Tower
AWS License Manager
AWS Well-Architected Tool
AWS Health Dashboard
AWS Chatbot
Launch Wizard
AWS Compute Optimizer
Resource Groups & Tag Editor
Amazon Grafana
Amazon Prometheus

Security, Identity, & Compliance
IAM
Resource Access Manager
Cognito
Secrets Manager
GuardDuty
Amazon Inspector
Amazon Macie
IAM Identity Center (successor to AWS Single Sign-On)
Certificate Manager
Key Management Service
CloudHSM
Directory Service
WAF & Shield
AWS Firewall Manager
AWS Artifact
Security Hub
Detective
AWS Signer
AWS Private Certificate Authority
AWS Audit Manager
Security Lake
Amazon Verified Permissions

AWS Cost Management
AWS Cost Explorer

https://console.aws.amazon.com/iam/home?region=ap-northeast-1

7. Create a user if you don't have one. In my case I have a user so I will be copying the access keys for later use (Note you will also need secret key so make sure you download the access keys .csv file when access keys are created)



IAM > Users > siddefault

siddefault

Summary

ARN
arn:aws:iam::319519345907:user/siddefault

Created
February 14, 2023, 13:42 (UTC+05:30)

Console access
Enabled without MFA

Last console sign-in
Today

Access key 1
[redacted] - Active
Used 12 days ago. 12 days old.

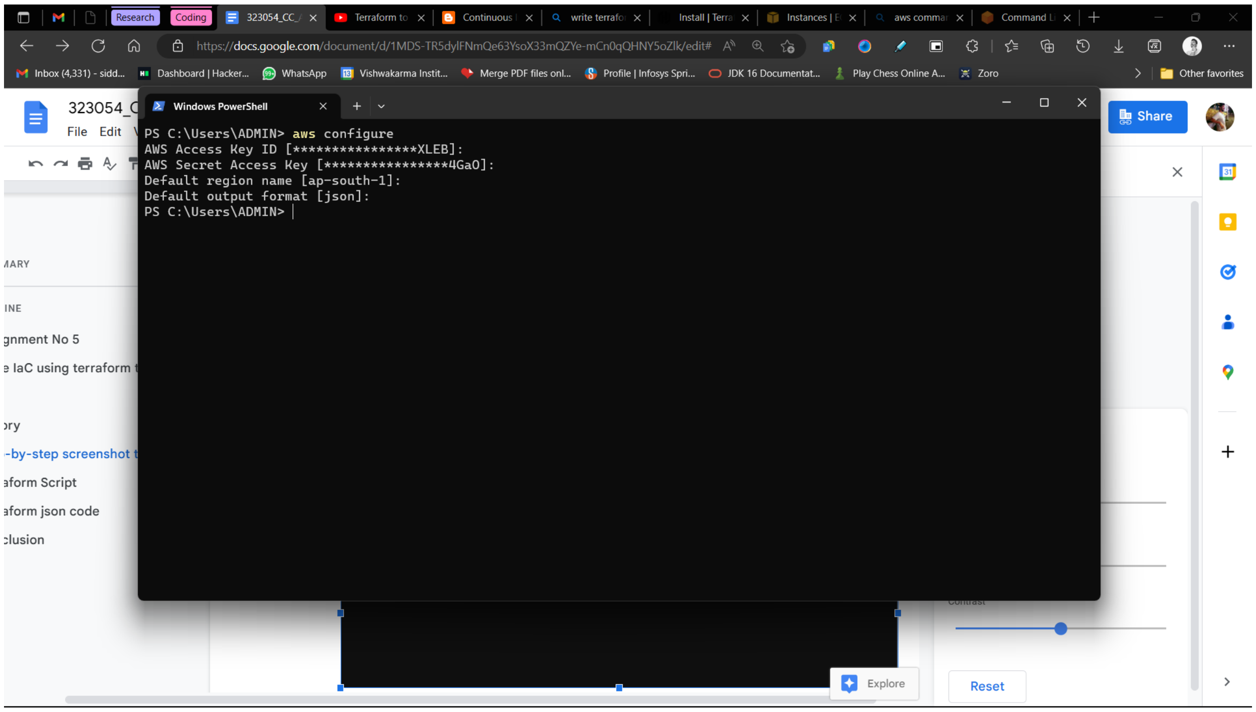
Access key 2
Not enabled

Permissions | Groups (1) | Tags (1) | Security credentials | Access Advisor

Permissions policies (2)
Permissions are defined by policies attached to the user directly or through groups.

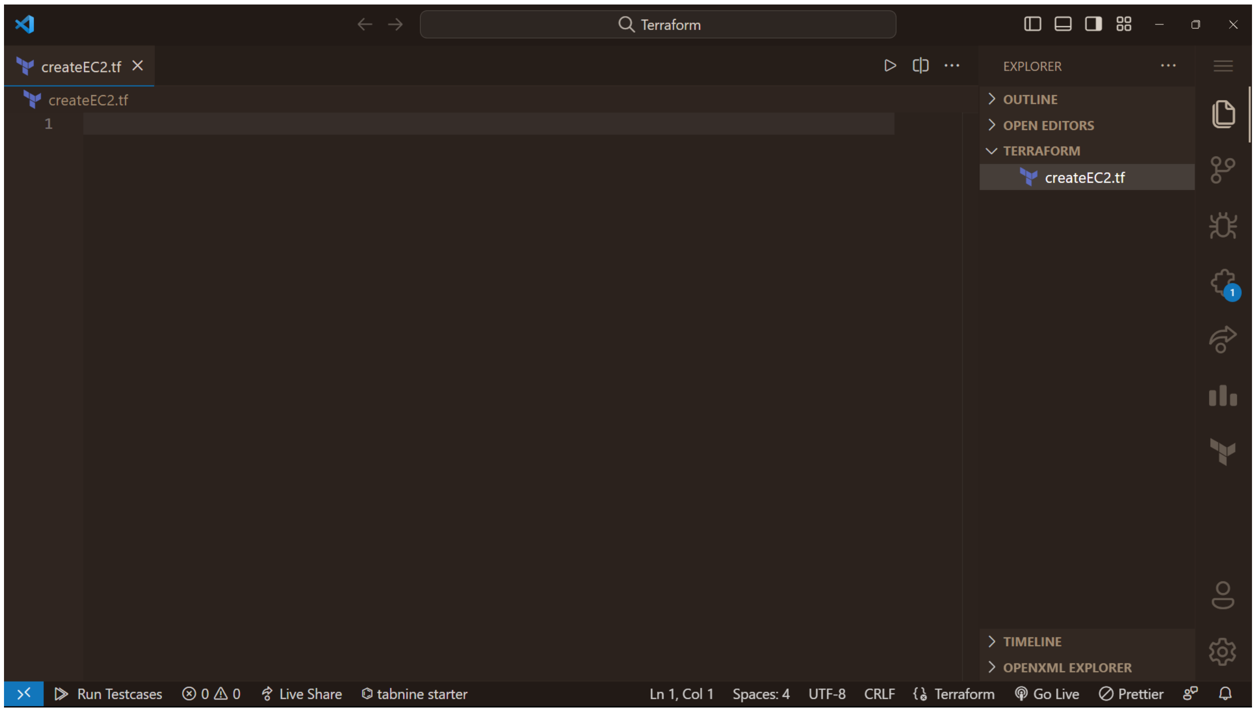
Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Group admin43
IAMUserChangePassword	AWS managed	Directly

8. Add secret key & access key to aws cli (I have already added it so I will just press enter here)

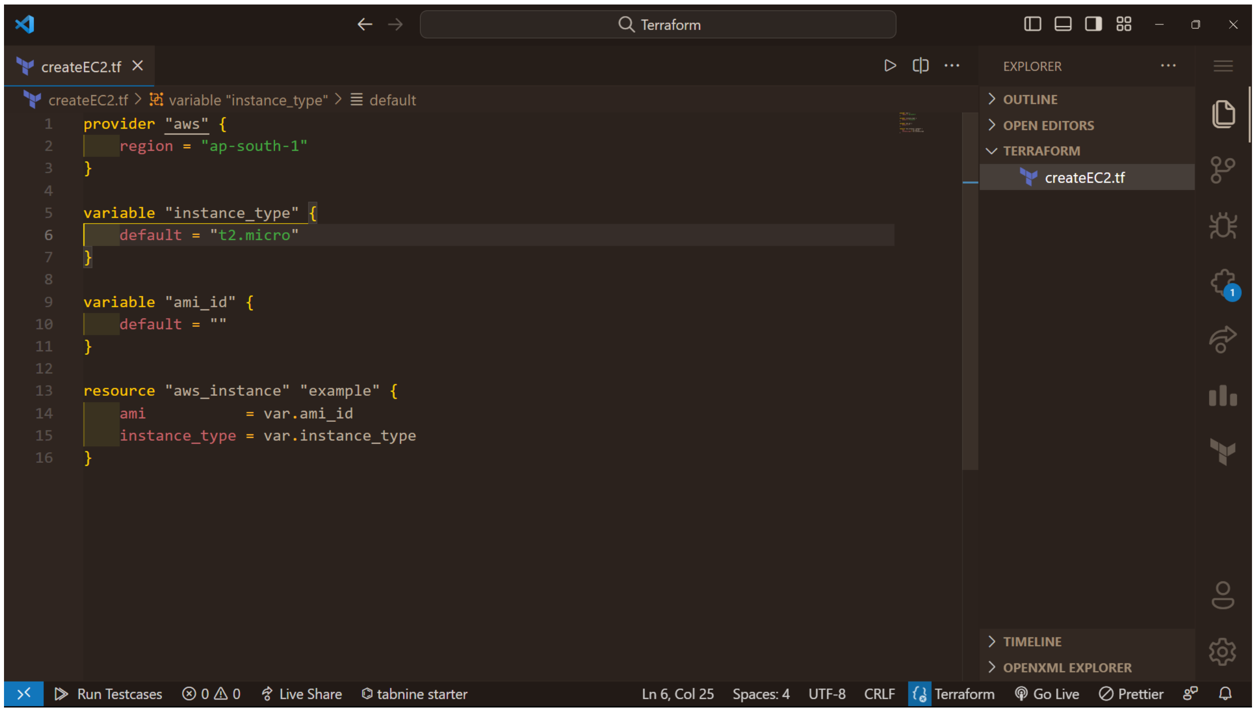


```
PS C:\Users\ADMIN> aws configure
AWS Access Key ID [****************XLEB]:
AWS Secret Access Key [****************4GaO]:
Default region name [ap-south-1]:
Default output format [json]:
PS C:\Users\ADMIN>
```

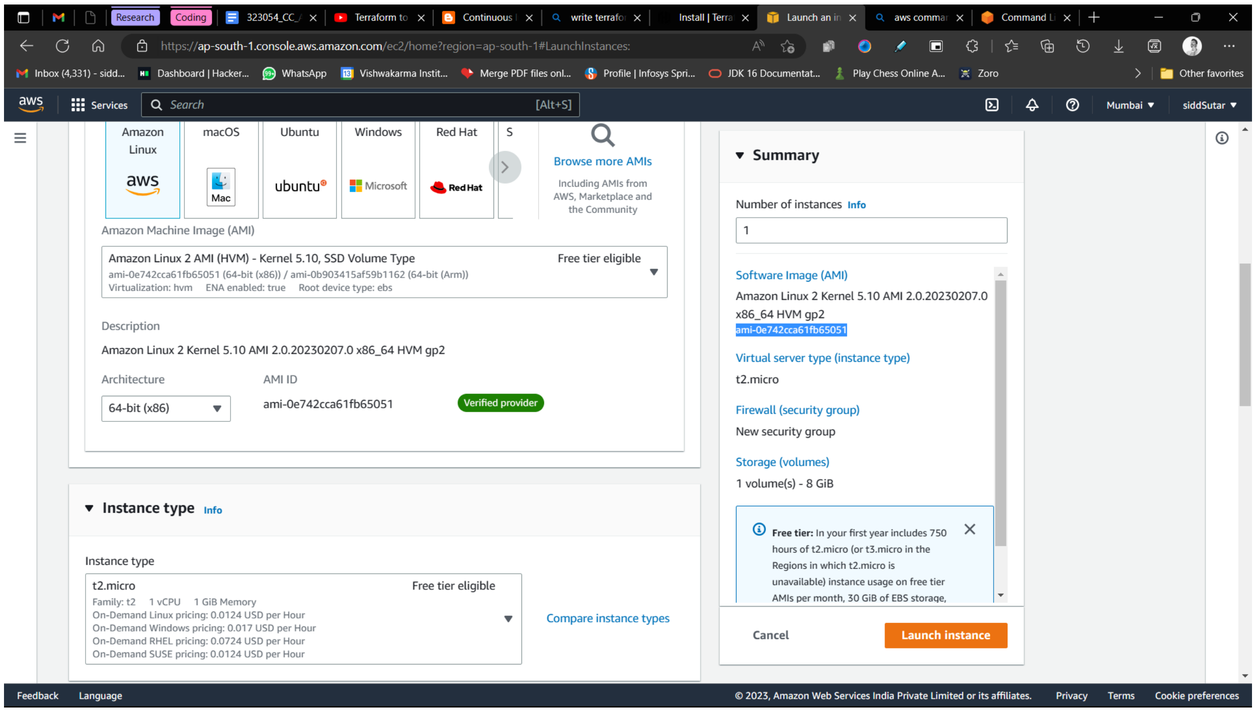
9. Create a folder named anything & create a terraform .tf file



10. Write json code to create an EC2 instance & select the AMI ID for the machine



```
provider "aws" {
    region = "ap-south-1"
}

variable "instance_type" {
    default = "t2.micro"
}

variable "ami_id" {
    default = ""
}

resource "aws_instance" "example" {
    ami           = var.ami_id
    instance_type = var.instance_type
}
```



Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-0e742cca61fb65051 (64-bit (x86)) / ami-0b903415af59b1162 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Amazon Linux 2 Kernel 5.10 AMI 2.0.20230207.0 x86_64 HVM gp2

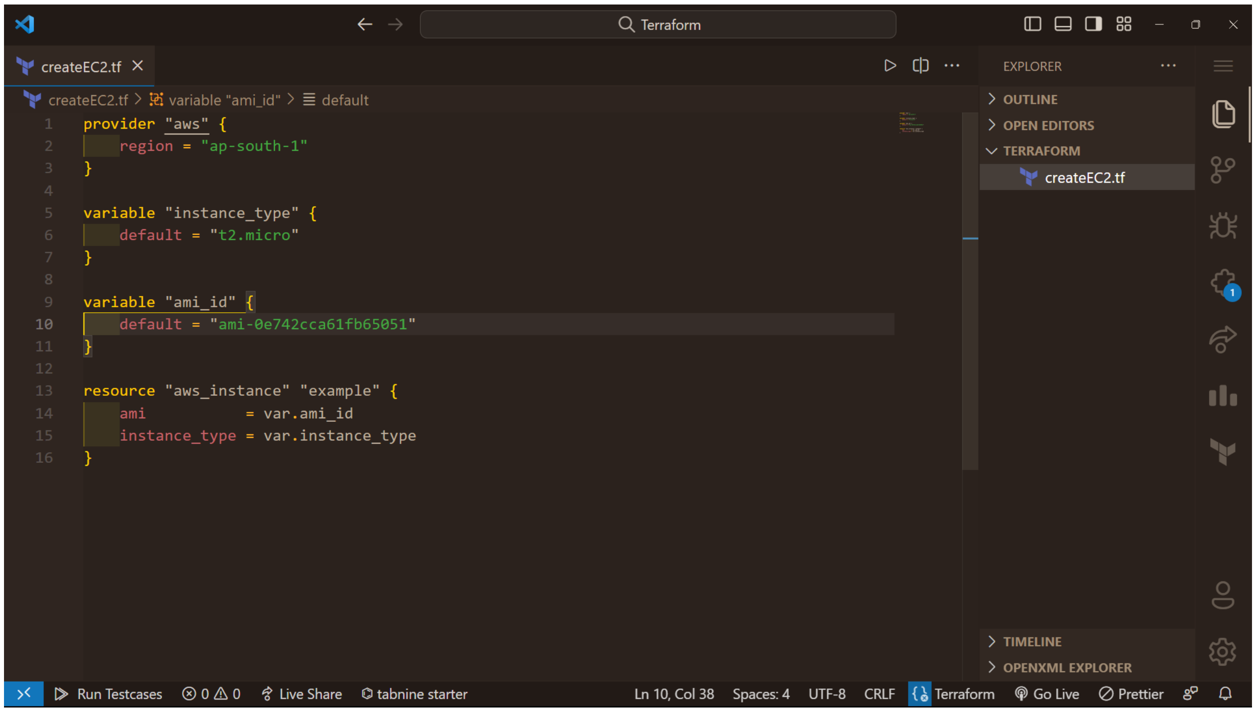
Architecture: 64-bit (x86)
AMI ID: ami-0e742cca61fb65051 — Verified provider

Instance type
t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory
On-Demand Linux pricing: 0.0124 USD per Hour
On-Demand Windows pricing: 0.017 USD per Hour
On-Demand RHEL pricing: 0.0724 USD per Hour
On-Demand SUSE pricing: 0.0124 USD per Hour

Summary
Number of instances: 1
Software Image (AMI): Amazon Linux 2 Kernel 5.10 AMI 2.0.20230207.0 x86_64 HVM gp2 ami-0e742cca61fb65051
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 t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage,

Copy the ami id to our json file

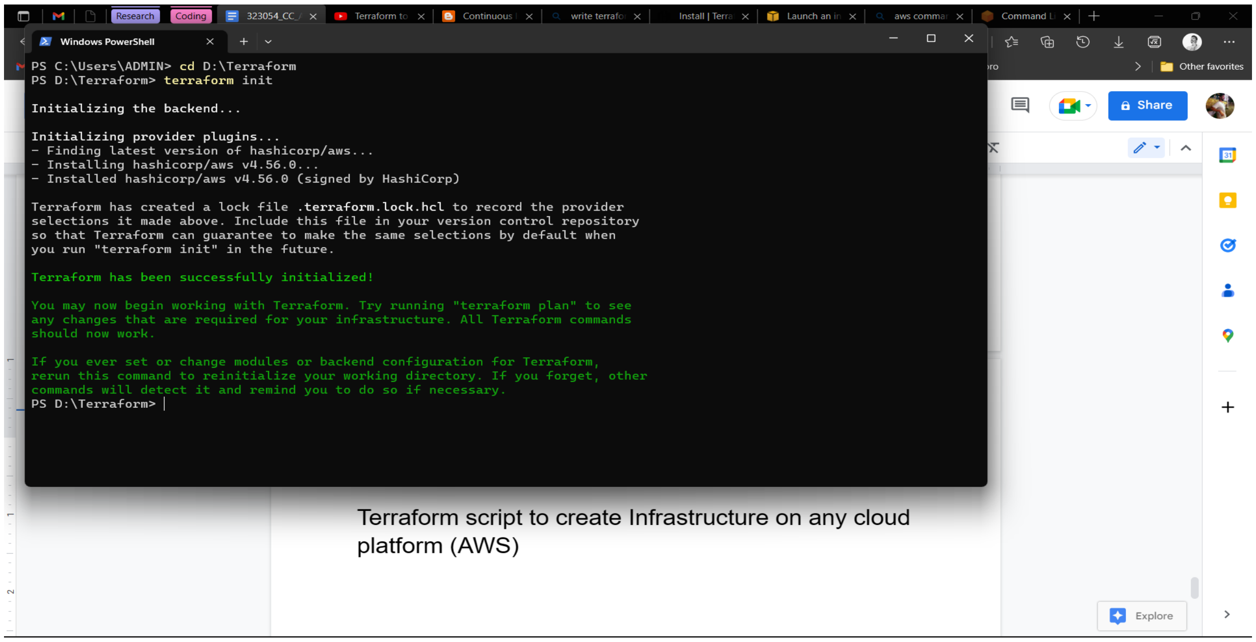


```hcl
provider "aws" {
    region = "ap-south-1"
}

variable "instance_type" {
    default = "t2.micro"
}

variable "ami_id" {
    default = "ami-0e742cca61fb65051"
}

resource "aws_instance" "example" {
    ami           = var.ami_id
    instance_type = var.instance_type
}
```

Now as we done with the setup we will move to terraform script

Terraform commands

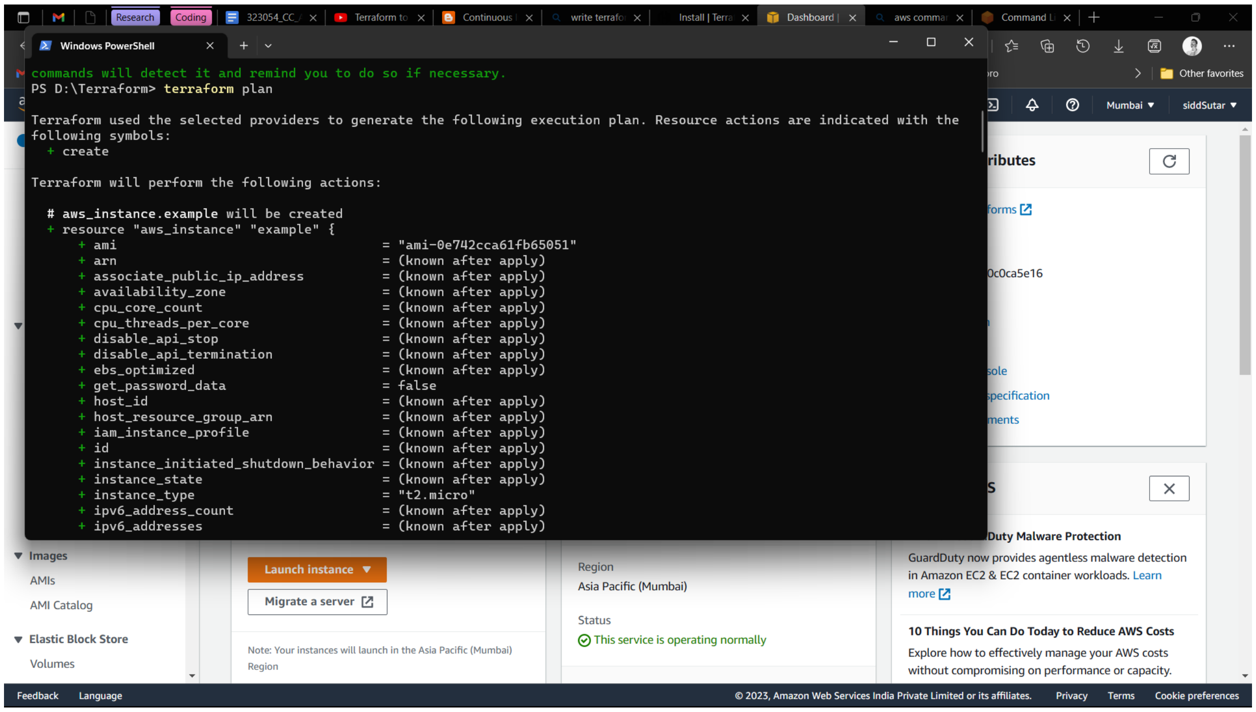
1. Change the directory & enter command terraform init



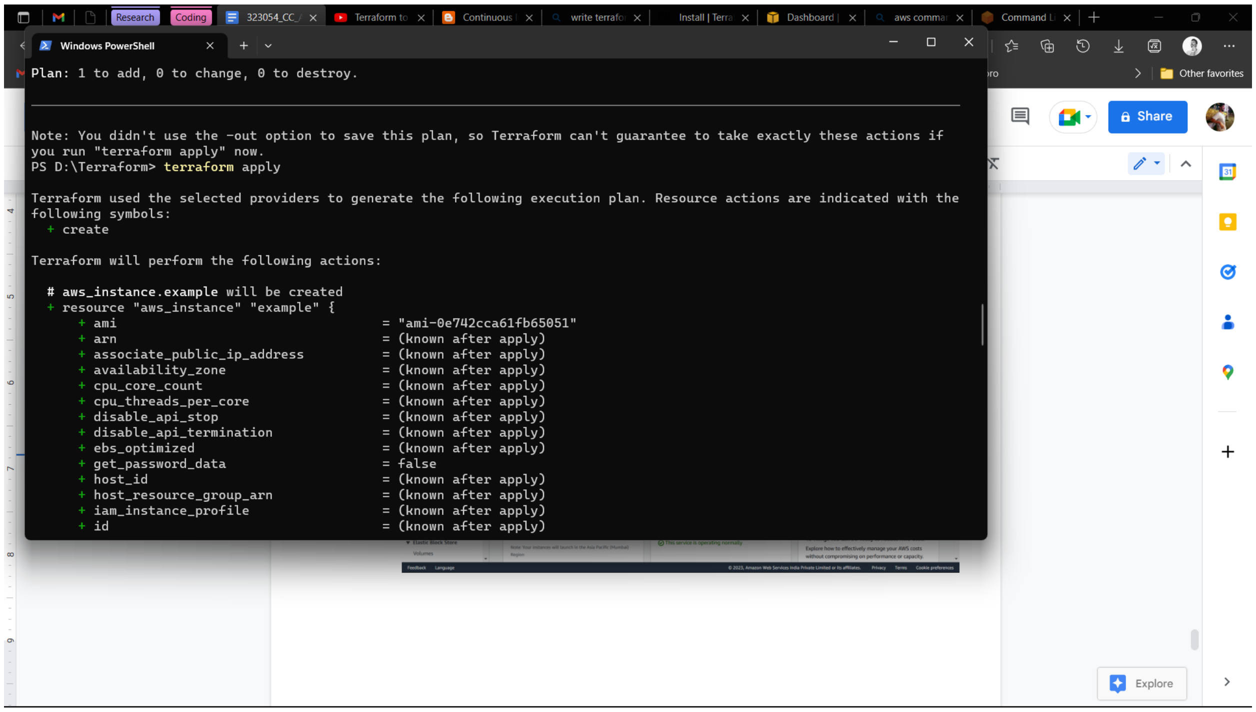
```
PS C:\Users\ADMIN> cd D:\Terraform
PS D:\Terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.56.0...
- Installed hashicorp/aws v4.56.0 (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.
PS D:\Terraform> |
```

Terraform script to create Infrastructure on any cloud platform (AWS)

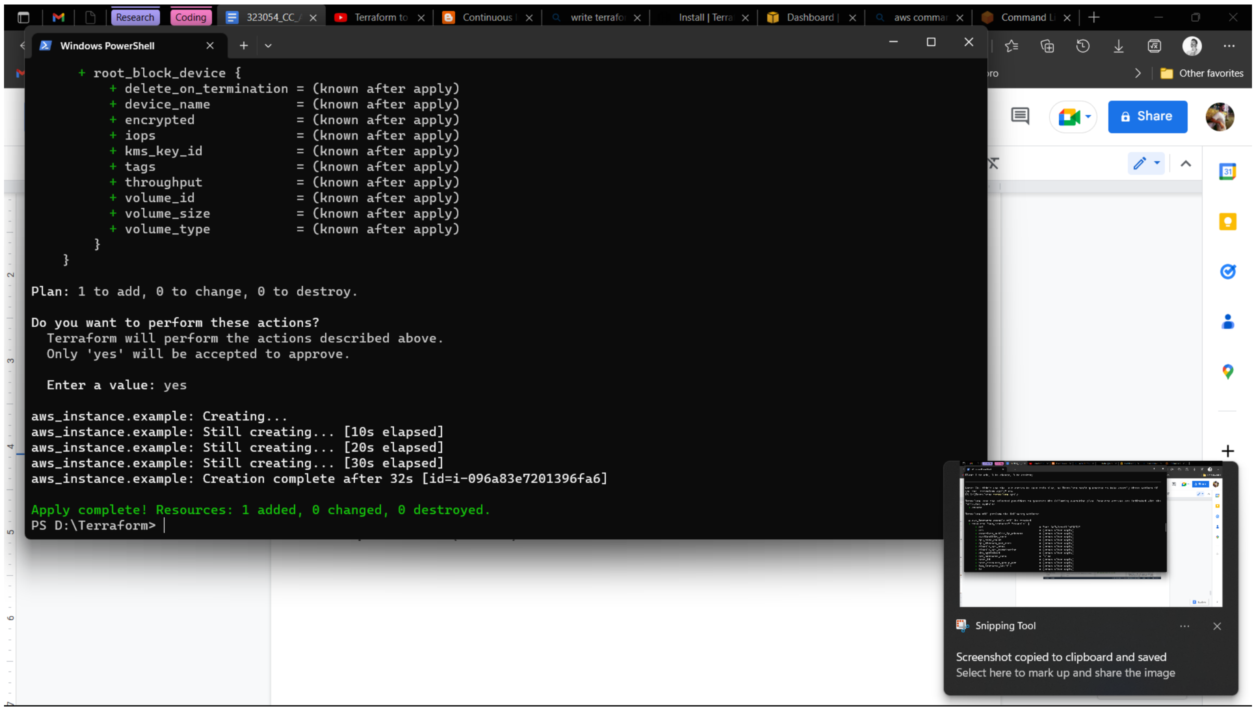
2. Put command terraform plan & terraform apply to create EC2 instance



```
commands will detect it and remind you to do so if necessary.
PS D:\Terraform> 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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-0e742cca61fb65051"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
```



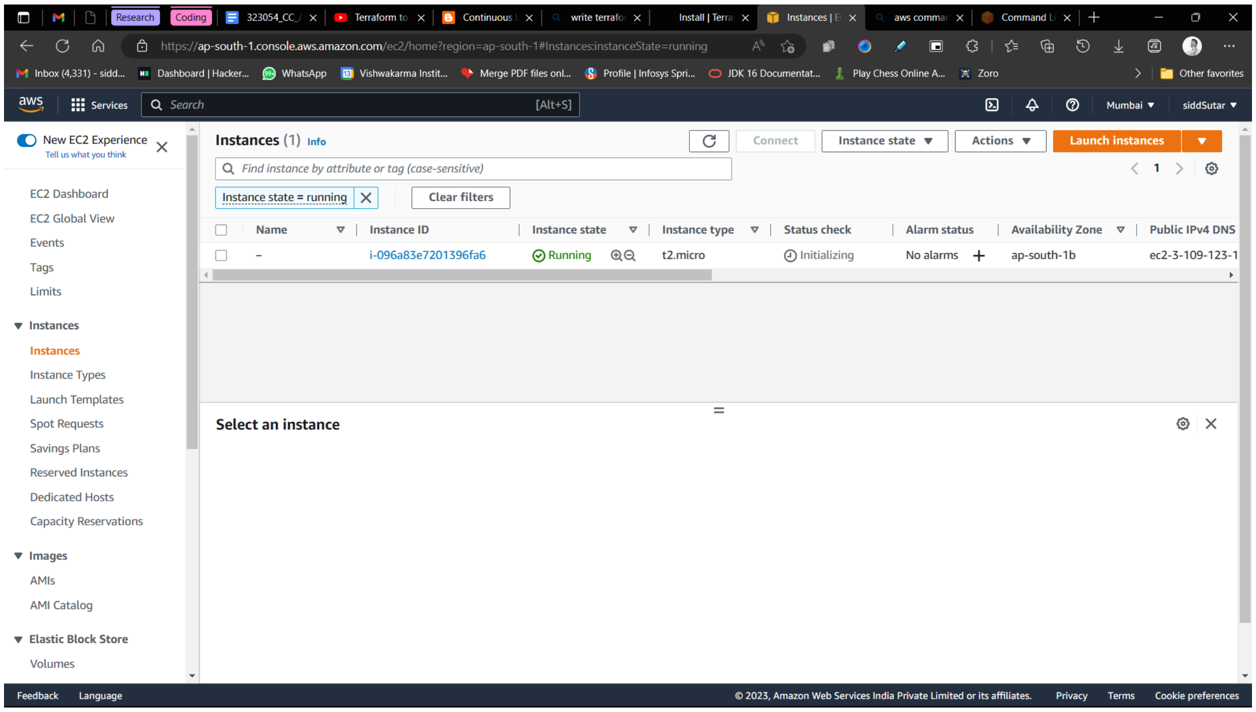
```
Plan: 1 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────

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.
PS D:\Terraform> 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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-0e742cca61fb65051"
      + 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)
```



```
      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (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.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Still creating... [30s elapsed]
aws_instance.example: Creation complete after 32s [id=i-096a83e7201396fa6]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS D:\Terraform>
```

Windows PowerShell
Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share the image

See the create AWS “example” instance



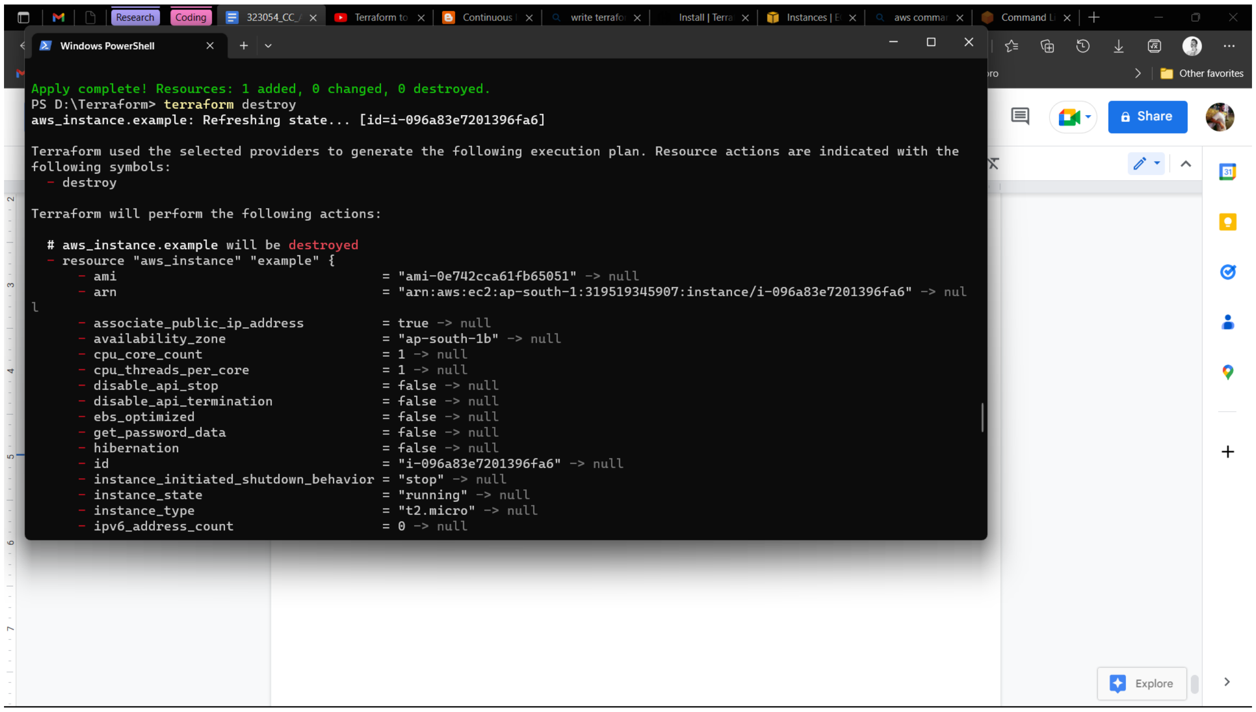
Instances (1)

Instance state = running

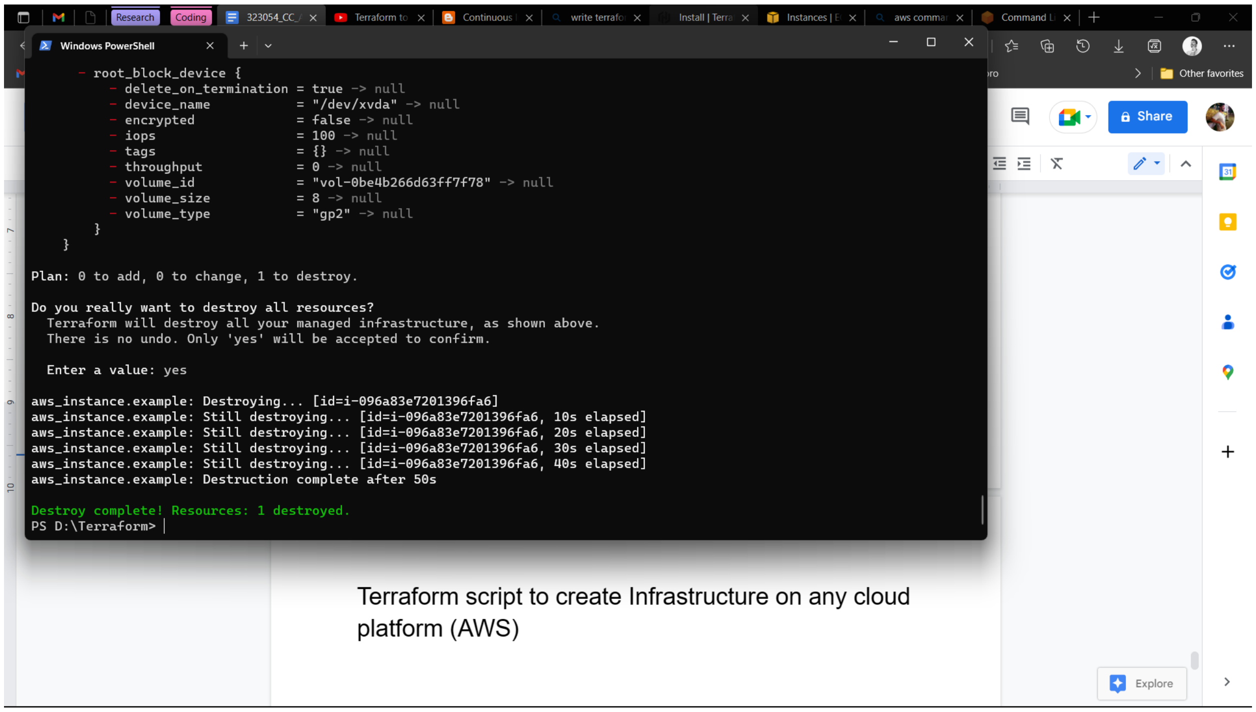
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
–	i-096a83e7201396fa6	Running	t2.micro	Initializing	No alarms	ap-south-1b	ec2-3-109-123-1

© 2023, Amazon Web Services India Private Limited or its affiliates.

3. Put command terraform destroy to delete/stop the instance



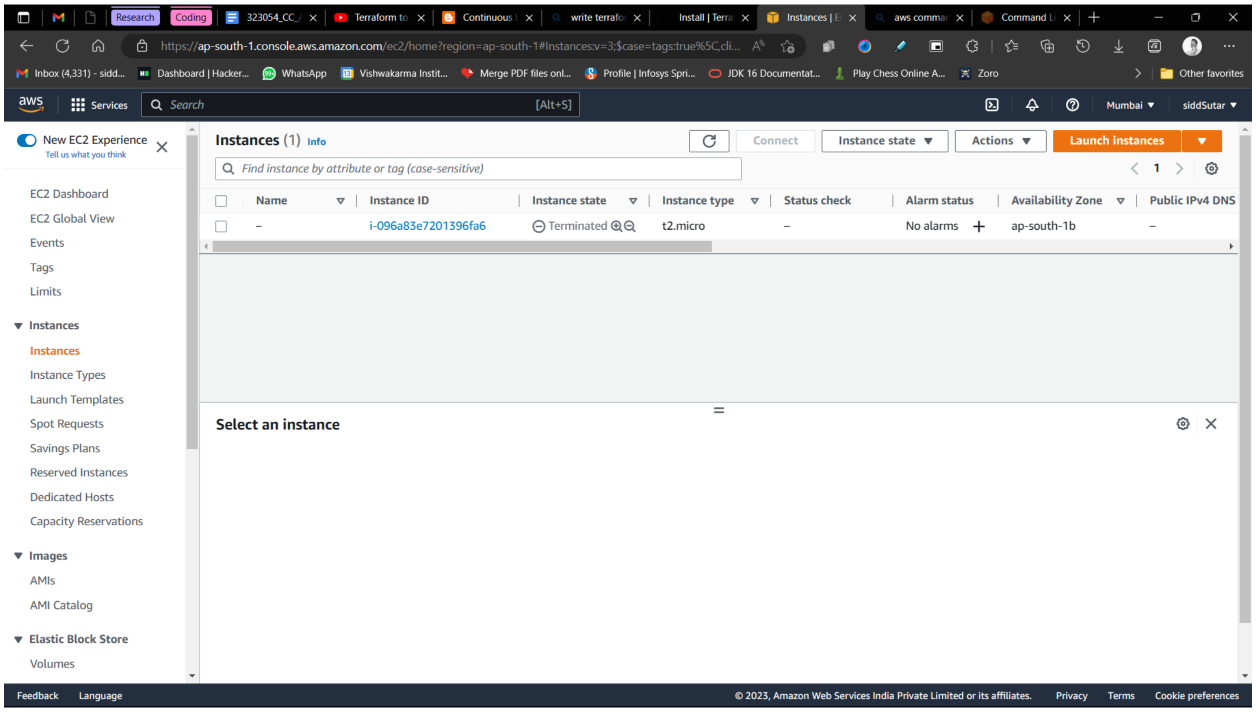
```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS D:\Terraform> terraform destroy
aws_instance.example: Refreshing state... [id=i-096a83e7201396fa6]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.example will be destroyed
  - resource "aws_instance" "example" {
      - ami                                  = "ami-0e742cca61fb65051" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:319519345907:instance/i-096a83e7201396fa6" -> nul
l
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-096a83e7201396fa6" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
```



```
      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/xvda" -> null
          - encrypted             = false -> null
          - iops                  = 100 -> null
          - tags                  = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-0be4b266d63ff7f78" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
        }
    }

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.example: Destroying... [id=i-096a83e7201396fa6]
aws_instance.example: Still destroying... [id=i-096a83e7201396fa6, 10s elapsed]
aws_instance.example: Still destroying... [id=i-096a83e7201396fa6, 20s elapsed]
aws_instance.example: Still destroying... [id=i-096a83e7201396fa6, 30s elapsed]
aws_instance.example: Still destroying... [id=i-096a83e7201396fa6, 40s elapsed]
aws_instance.example: Destruction complete after 50s

Destroy complete! Resources: 1 destroyed.
PS D:\Terraform>
```

Terraform script to create Infrastructure on any cloud platform (AWS)

Let's check whether the instance is properly terminated or not

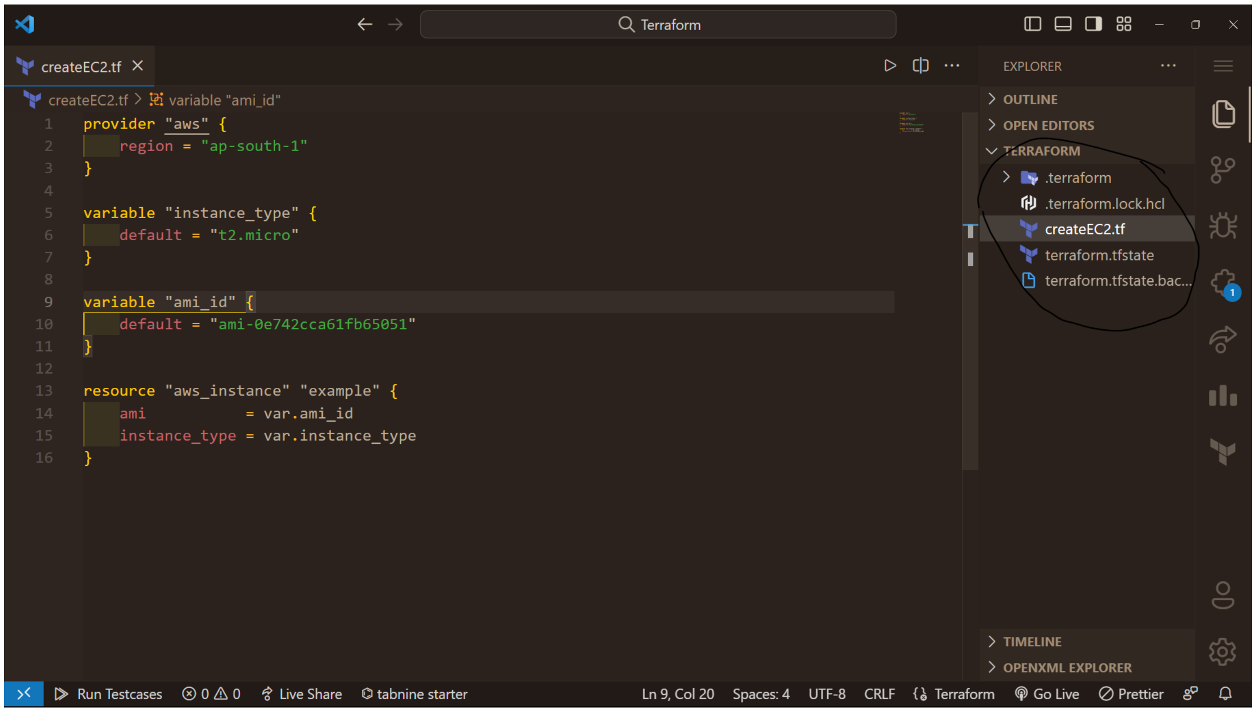


Instances (1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
–	i-096a83e7201396fa6	Terminated	t2.micro	–	No alarms	ap-south-1b	–

Select an instance

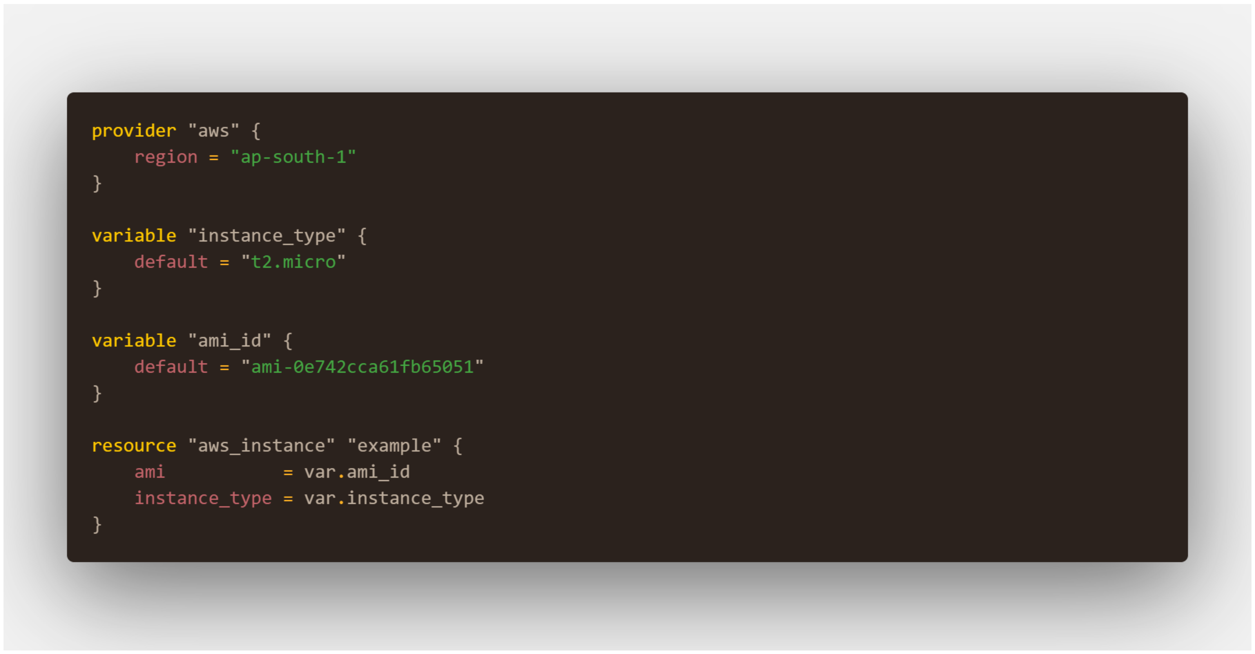
4. The final file should look like this



createEC2.tf

```
provider "aws" {
    region = "ap-south-1"
}

variable "instance_type" {
    default = "t2.micro"
}

variable "ami_id" {
    default = "ami-0e742cca61fb65051"
}

resource "aws_instance" "example" {
    ami           = var.ami_id
    instance_type = var.instance_type
}
```

Terraform json code



```
provider "aws" {
    region = "ap-south-1"
}

variable "instance_type" {
    default = "t2.micro"
}

variable "ami_id" {
    default = "ami-0e742cca61fb65051"
}

resource "aws_instance" "example" {
    ami           = var.ami_id
    instance_type = var.instance_type
}
```

Conclusion

→ Terraform is understood alongside its basic commands.