

## **TERRAFORM (IaC)**

### **1) What is IaC and its advantages?**

**A)** Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes. With IaC, configuration files are created that contain your infrastructure specifications, which makes it easier to edit and distribute configurations.

#### **Advantages of IaC**

- Consistency and Reproducibility.
- Version Control.
- Automation and Efficiency.
- Scalability and Flexibility.
- Documentation and Self-Documenting Code.
- Testing and Validation.
- Cross-platform and Cloud Agnostic.

### **2) Types of services can be done by using IaC?**

- A)** 1) Infrastructure management.  
‘2) Configuration Management.

### **3) What is terraform and its alternatives?**

**A)** Terraform is an open-source Infrastructure as Code (IAC) tool developed by HashiCorp. It allows you to define and provision infrastructure resources (such as virtual machines, networks, databases, and more) in a declarative manner using configuration files. It is designed to work with various cloud providers and infrastructure platforms, making it a popular choice for managing infrastructure across different environments.

#### **Key features of Terraform**

- Declarative Configuration.
- Infrastructure Versioning.
- State Management.
- Modularity and Reusability.

Terraform alternative tools are AWS CloudFormation, Google Cloud Deployment Manager, Azure Resource Manager (ARM) Templates, Ansible, Chef and Puppet.

### **4) Terraform terminology?**

**A) i) Terraform Configuration:** It refers to the set of files and directories containing the Terraform code that defines your infrastructure.

**ii) Provider:** It allow Terraform to manage resources in various environments, such as AWS, Azure, Google Cloud, etc.

- iii) **Resource:** It represents a single infrastructure component that Terraform manages, such as virtual machines, networks, databases, etc.
- iv) **Module:** Modules allow you to create custom abstractions and share configurations between different projects.
- v) **Outputs:** Terraform outputs allow you to expose information about your infrastructure to other resources or to the user. You can use outputs to share data between resources, or to display important information to the user.
- vi) **Terraform Cloud:** Terraform Cloud is a hosted service that provides additional features for managing your Terraform infrastructure, such as collaborative workspaces, automated runs, and version control.
- vii) **Remote State:** Terraform's remote state feature allows you to store your infrastructure state in a remote location, such as an S3 bucket or a Terraform Cloud workspace. This can be useful for collaboration and for keeping track of changes to your infrastructure.
- viii) **State File:** The state file is automatically created when `apply` command entered and it is managed by Terraform. By default, it is stored in a local file named `terraform.tfstate`.
- ix) **Variables:** Terraform variables allow you to define values that can be used throughout your infrastructure configuration. You can use variables to make your configuration more flexible and reusable, and to make it easier to manage complex infrastructure.

## 5) Terraform life cycle?

- A) The typical life cycle of Terraform involves the following steps
  - i) **Initialize:** Run '`terraform init`' in the working directory to initialize Terraform for your project. This command downloads the necessary provider plugins and sets up the backend configuration.
  - ii) **Plan:** Run '`terraform plan`' to create an execution plan. It doesn't make any changes during this step. It only shows you what actions thus Terraform will take.
  - iii) **Apply:** Run '`terraform apply`' to execute the changes proposed in the plan. It will create, modify, or delete resources as needed to reach the desired state specified in the configuration files.
  - iv) **Destroy:** Run '`terraform destroy`' to remove all the resources created by Terraform based on the configuration.

## 6) Terraform installation and configuring with AWS cloud for infrastructure provisioning Purpose?

- A) To install and configure terraform in windows machine  
Pre-requisites are
  - 1) Visual Studio Code / VS code (Installation steps shown below)
  - 2) AWS CLI (Installation steps shown below including screenshots)

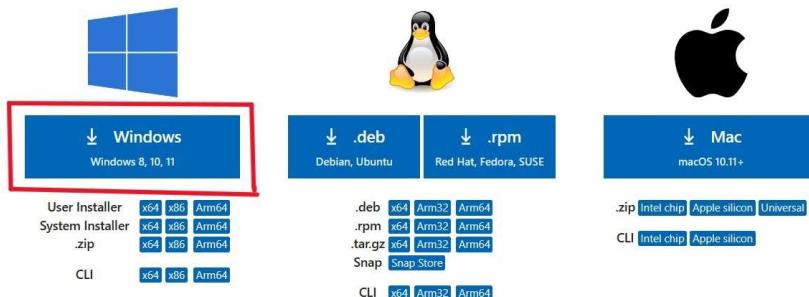
**To download:** <https://code.visualstudio.com/Download>

Steps as follow, select the highlighted box (Windows) shown in below image.

The screenshot shows the Visual Studio Code download page at https://code.visualstudio.com/Download. At the top, there's a navigation bar with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, Learn, and a search bar. A blue button labeled 'Download' is visible. Below the navigation, a message says 'Version 1.80 is now available! Read about the new features and fixes from June.' The main content area is titled 'Download Visual Studio Code' and describes it as 'Free and built on open source. Integrated Git, debugging and extensions.' There are three main download sections: 'Windows' (highlighted with a red box), 'Linux' (Ubuntu, Red Hat, Fedora, SUSE), and 'Mac' (macOS 10.11+). Each section shows download links for different file formats (.deb, .rpm, .tar.gz, Snap, CLI) and architectures (x64, x86, Arm64).

## Download Visual Studio Code

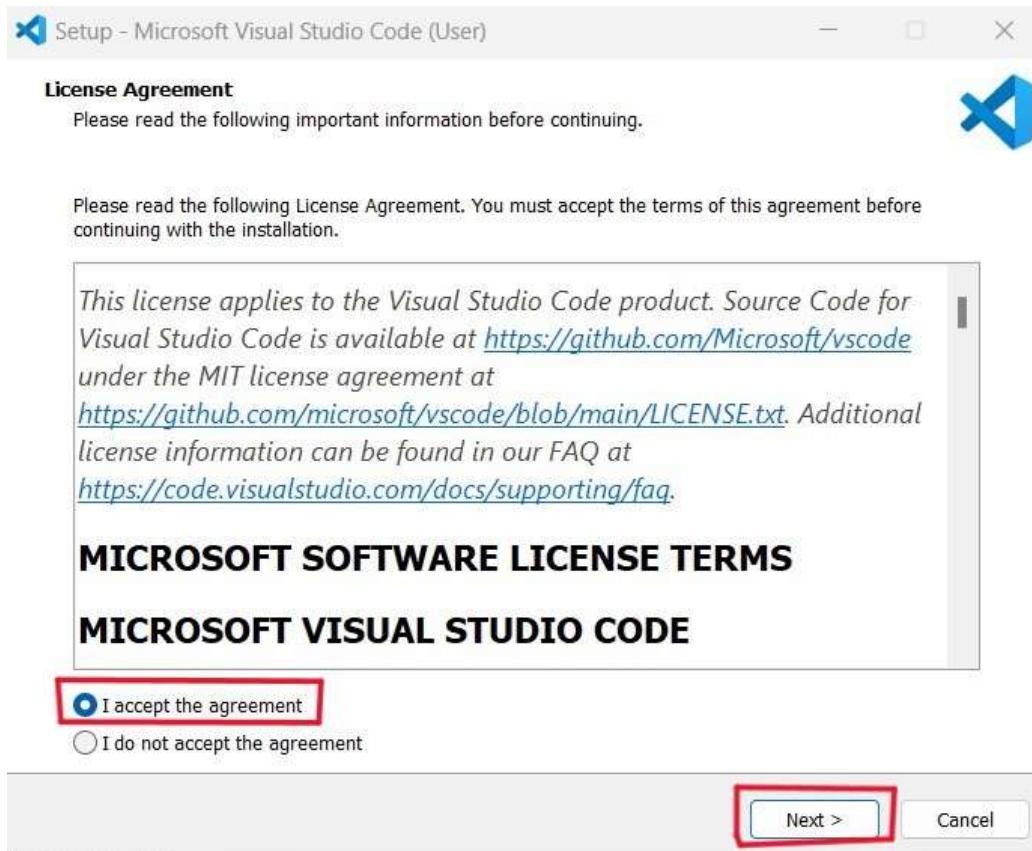
Free and built on open source. Integrated Git, debugging and extensions.



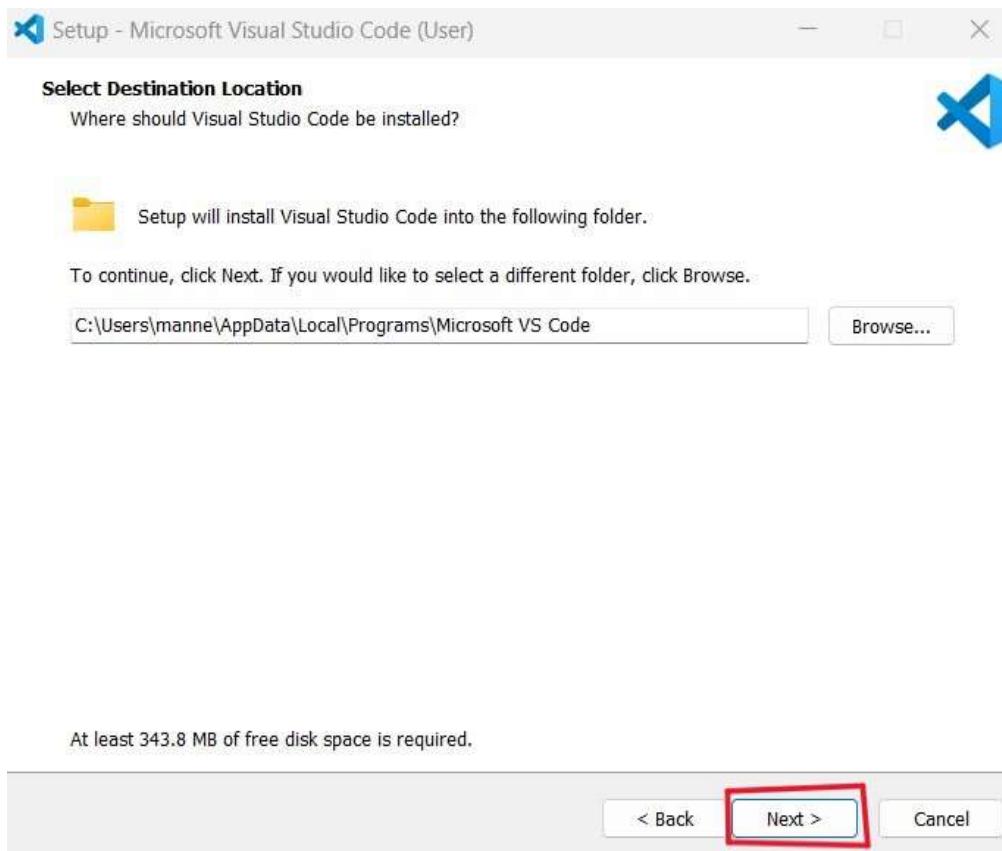
After completion of download, double click on that file

The screenshot shows a Windows File Explorer window with the title bar 'Downloads'. The address bar shows the path 'This PC > OS (C:) > Users > manne > Downloads'. The left sidebar lists 'Home', 'Downloads' (which is selected and highlighted with a red box), 'main folder', and 'New Volume (D:)'. In the main pane, there is a file named 'VSCodeUserSetup-x64-1.80.1'. A red box highlights this file, and a red arrow points from the text 'Double click on this icon' to the file's icon.

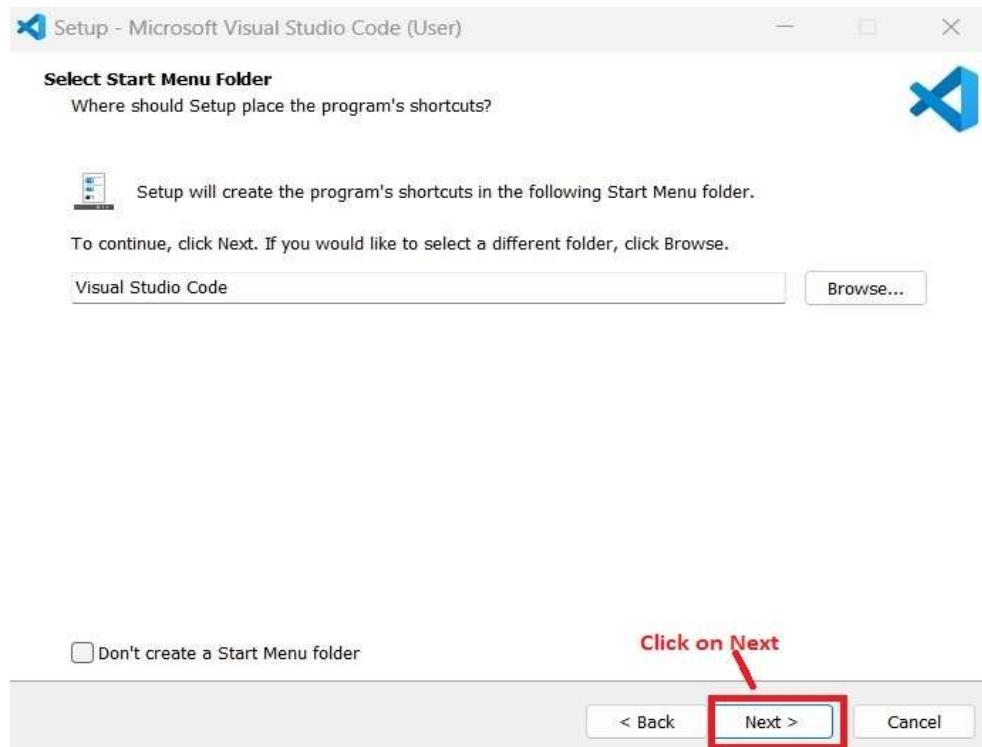
After double clicking on that icon, the following screen will appear.



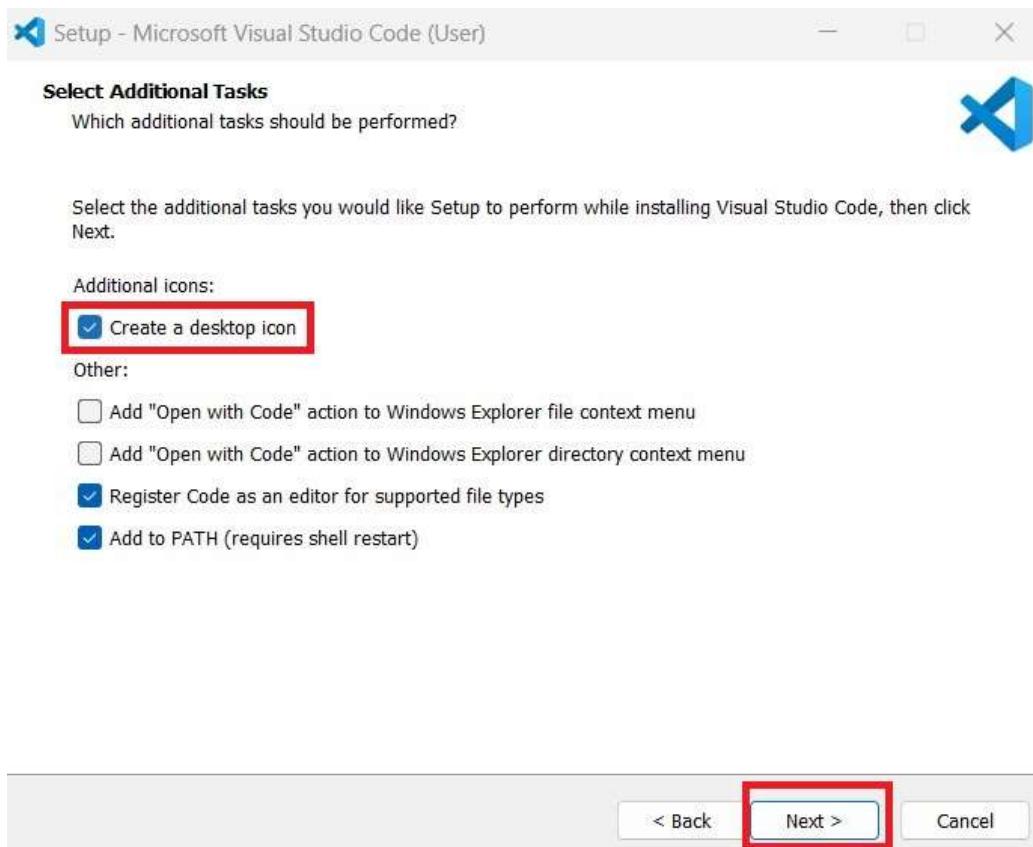
After clicking on Next button, the following screen will appear.



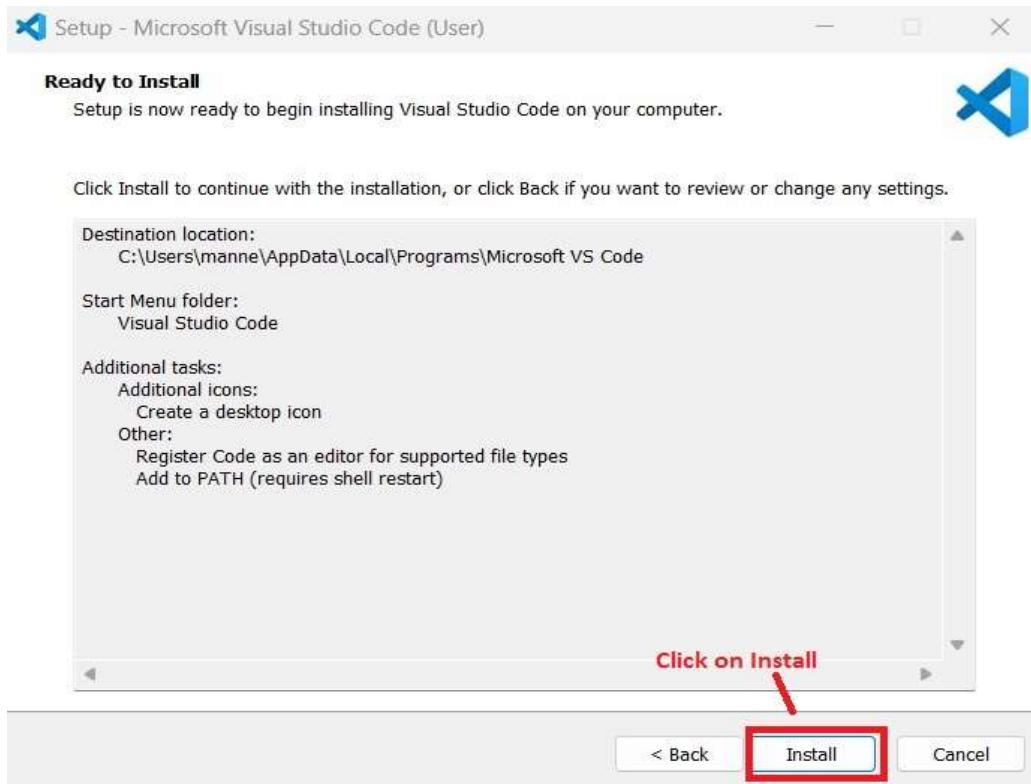
After clicking on Next button, the following screen will appear.



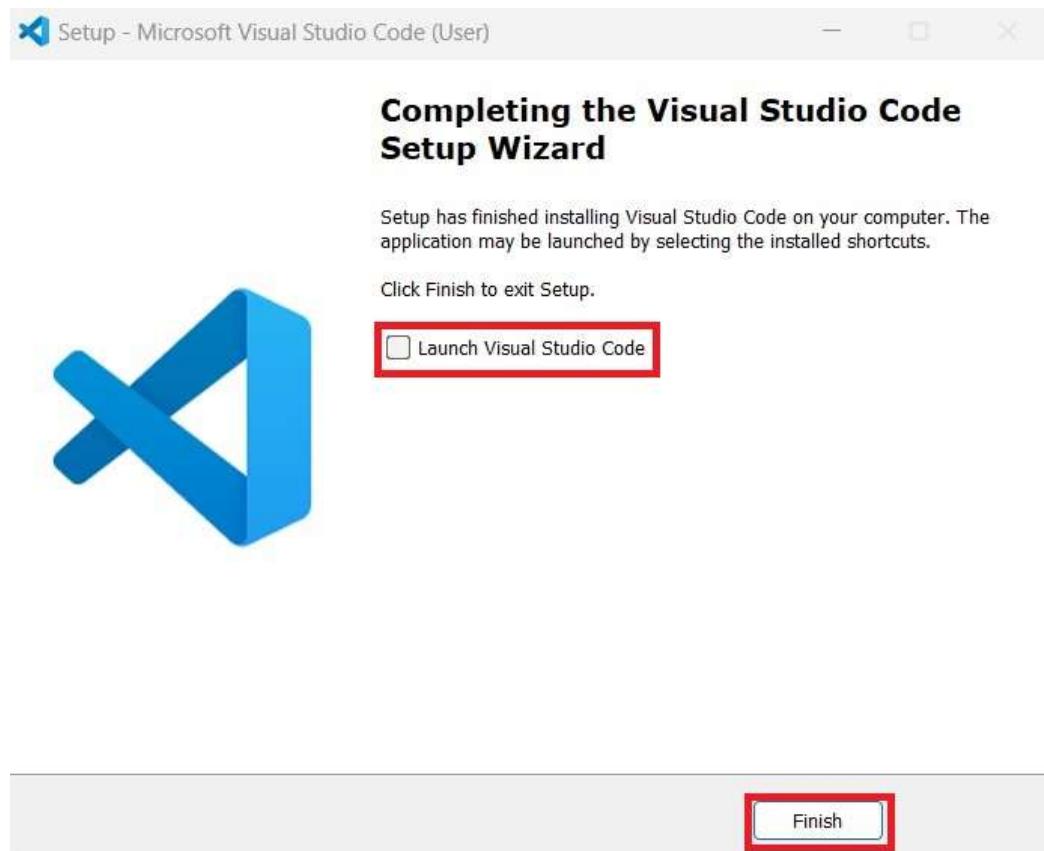
After clicking on Next button, the following screen will appear.



After clicking on Next button, the following screen will appear.



After clicking on Install button, the following screen will appear.

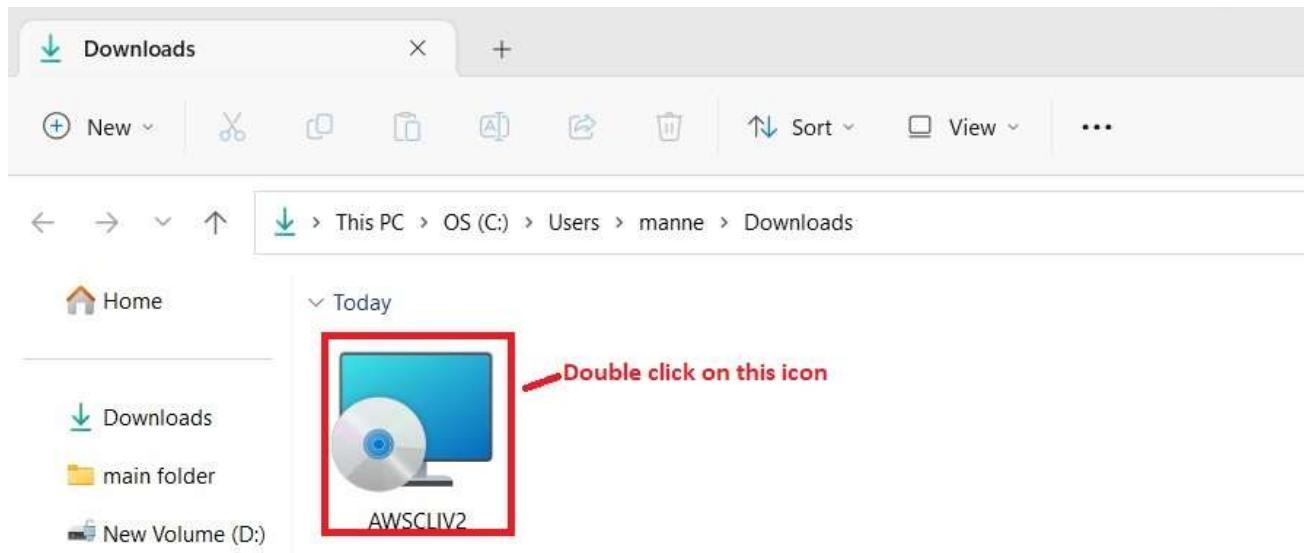


Finally click on that check box and finish icon. Visual studio code setup was finished.

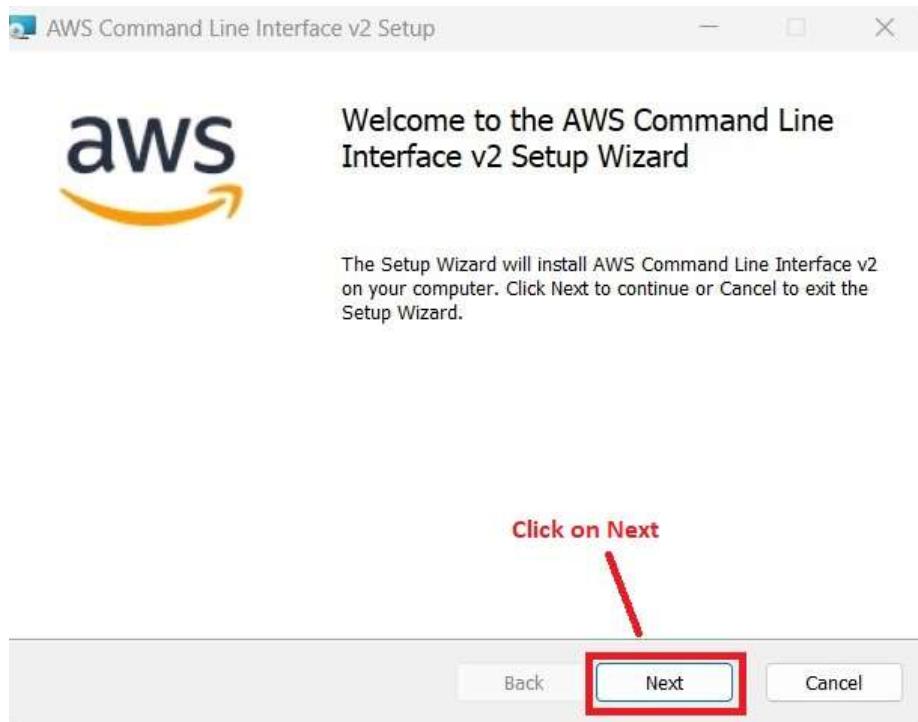
## AWS CLI INSTALLATION

To download: <https://awscli.amazonaws.com/AWSCLIV2.msi>

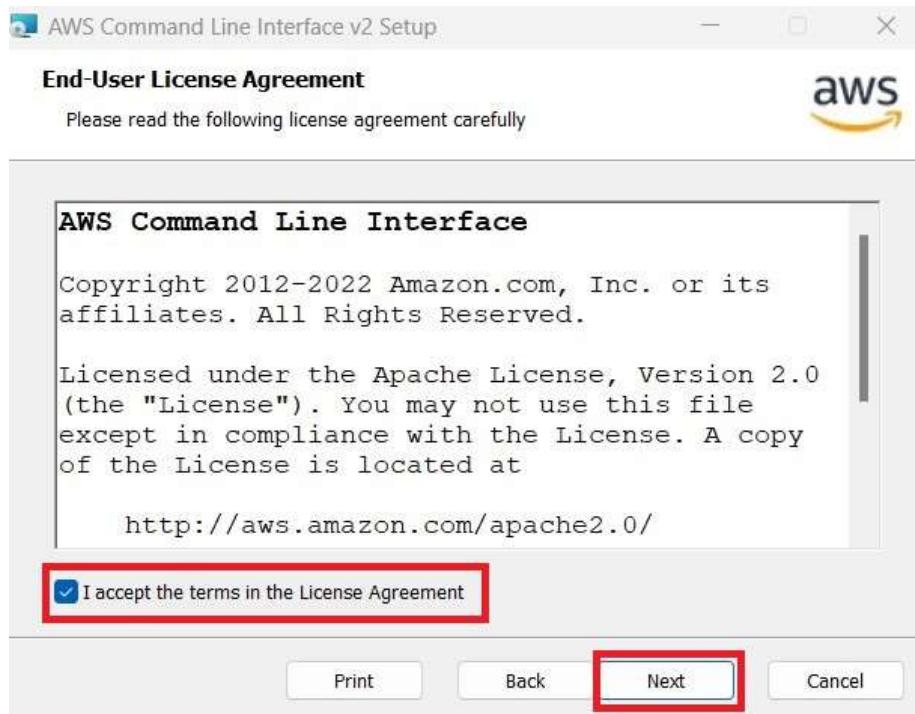
After clicking on the above link, AWS-CLI software will be downloaded.



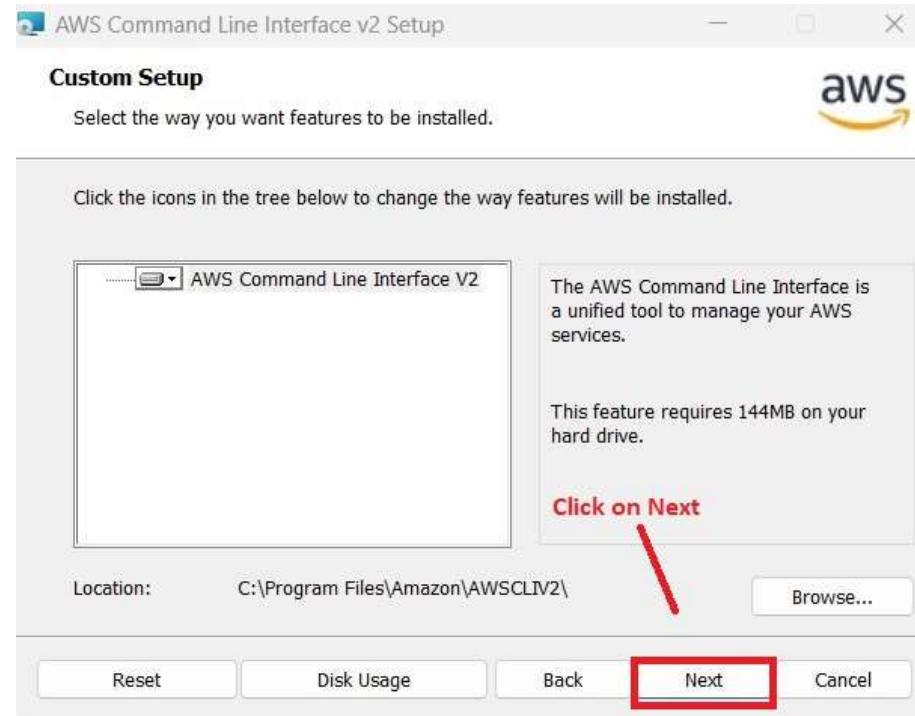
After double-clicking on that icon, the following screen will be displayed



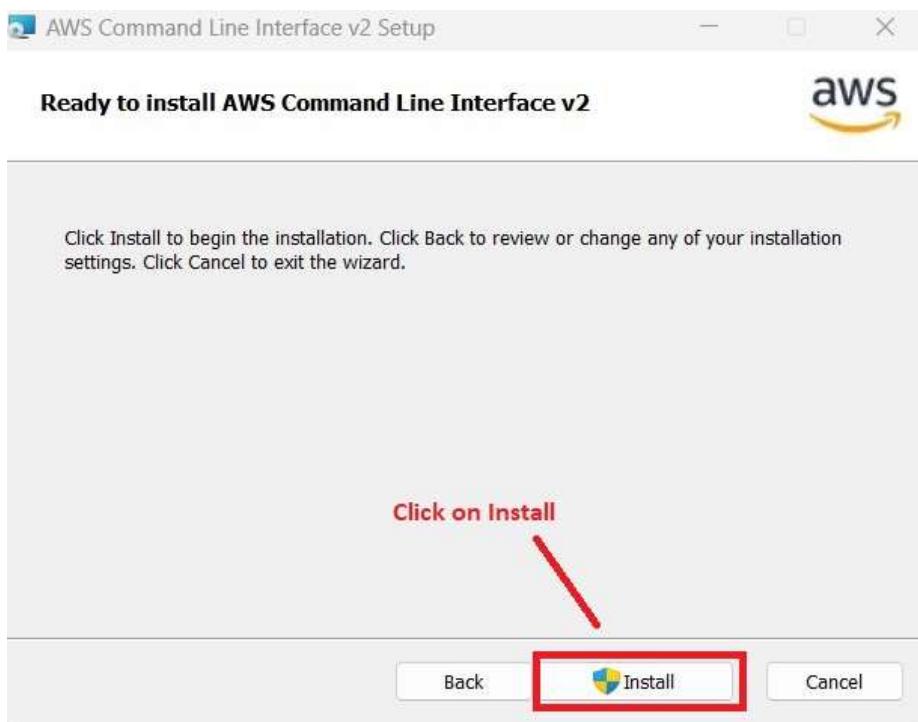
After clicking on next button, the following screen will be displayed.



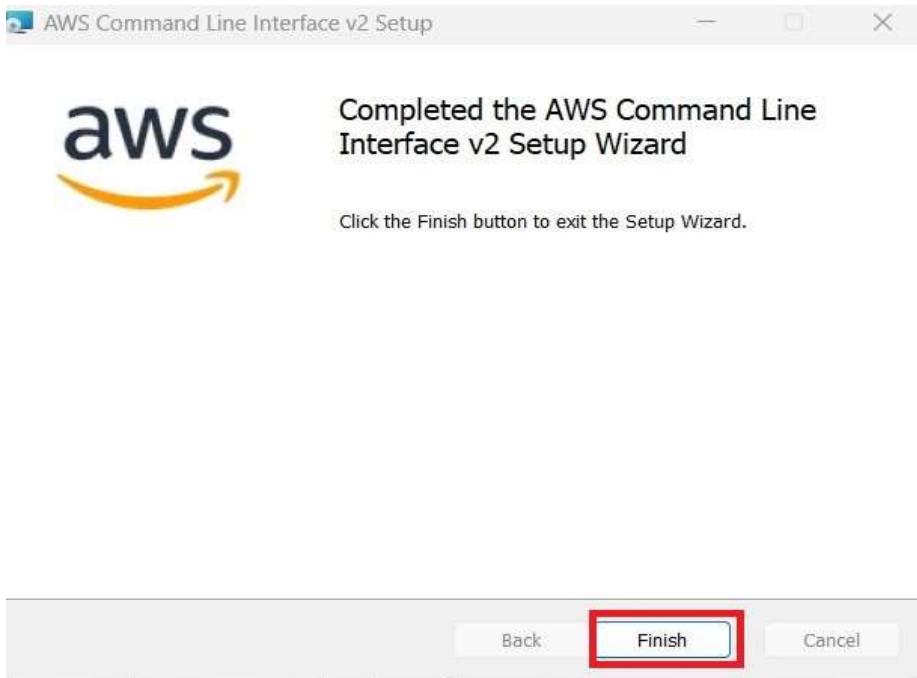
After clicking on next button, the following screen will be displayed.



After clicking on next button, the following screen will be displayed.

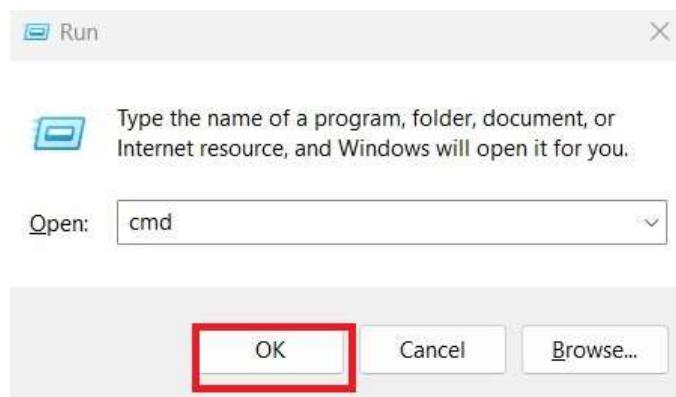


After clicking on ‘Install’ button, Alert will be displayed click on ‘Yes’ the following screen will be displayed.

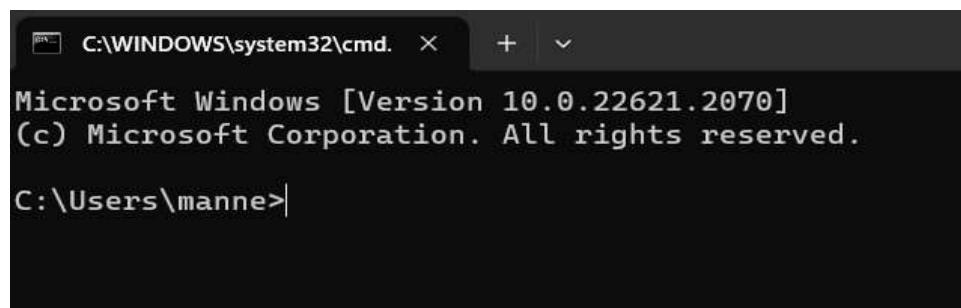


Finally click on finish button, AWS-CLI installation completed.

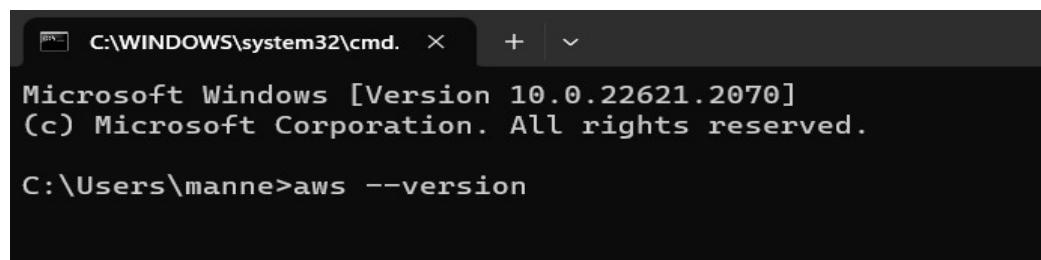
After that, we have to set the environment variables by opening command prompt. Using (Windows Key + R) the following screen will be displayed.



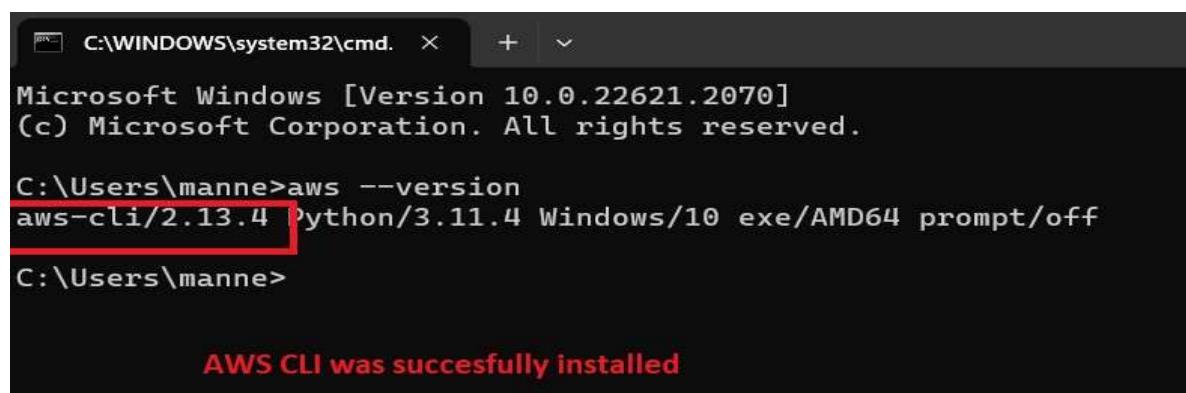
After clicking on that ok button, the following screen will be displayed.



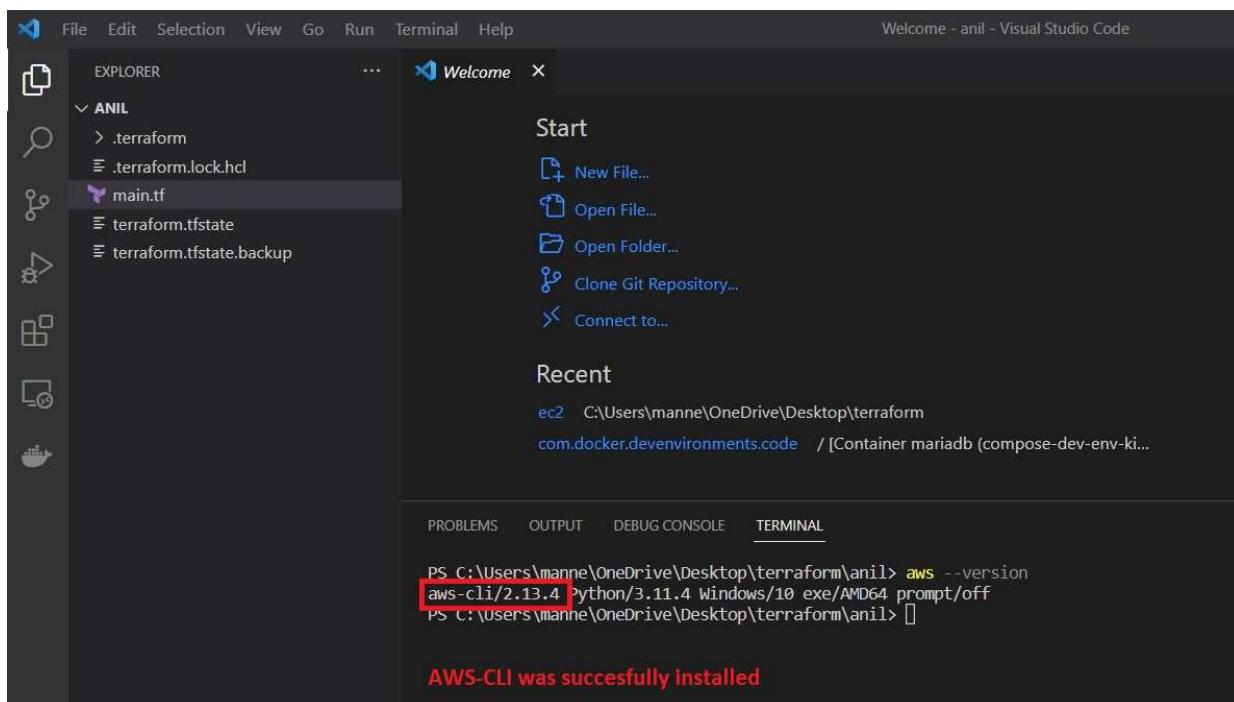
Then to view the AWS-CLI version, in command prompt enter 'aws --version' as shown in screenshot.



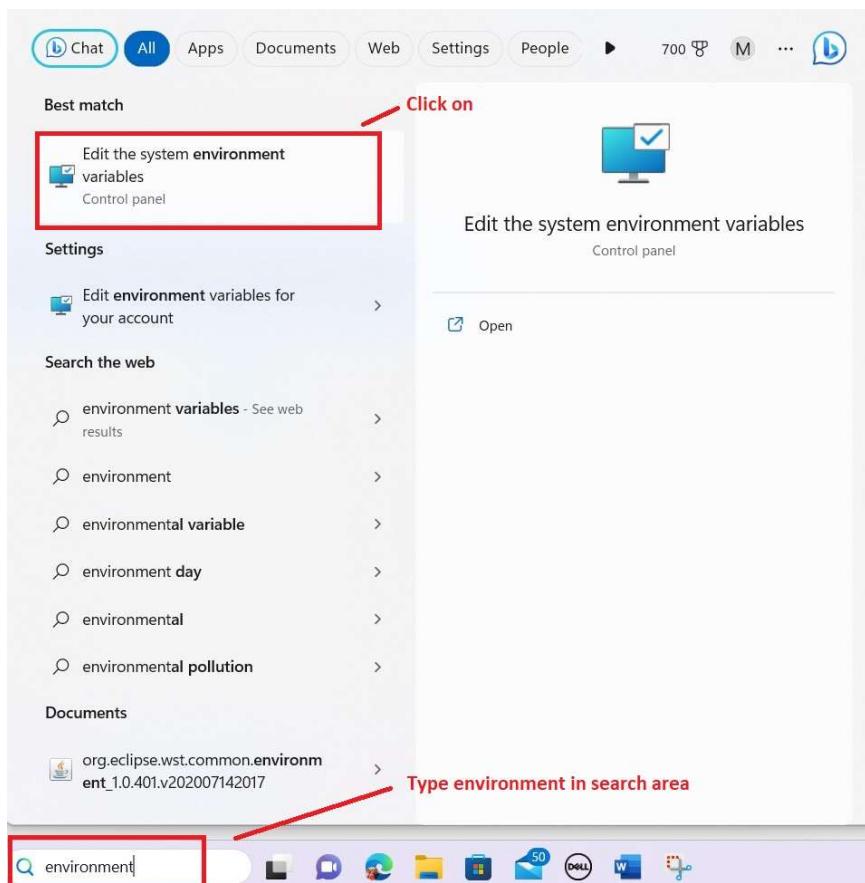
After clicking on Enter button, the following screen will be displayed.



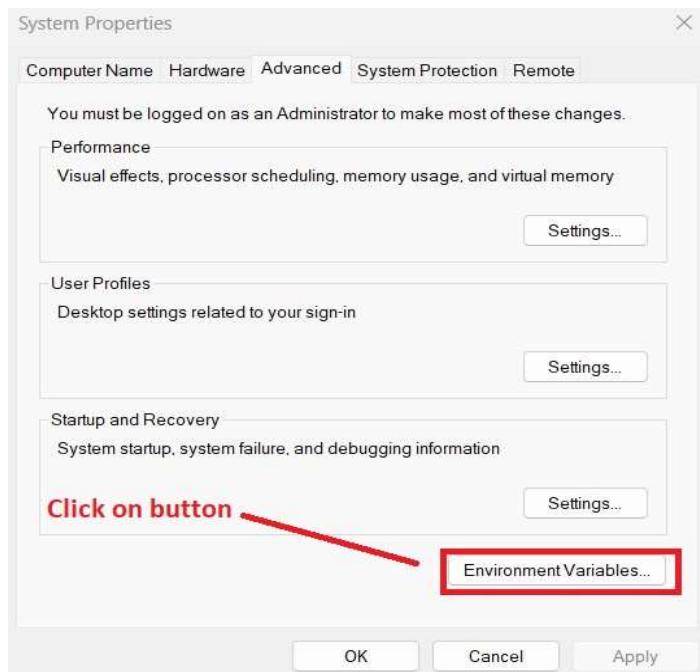
Same step has to be followed, by opening visual studio code and check for AWS-CLI version. scree looks like below.



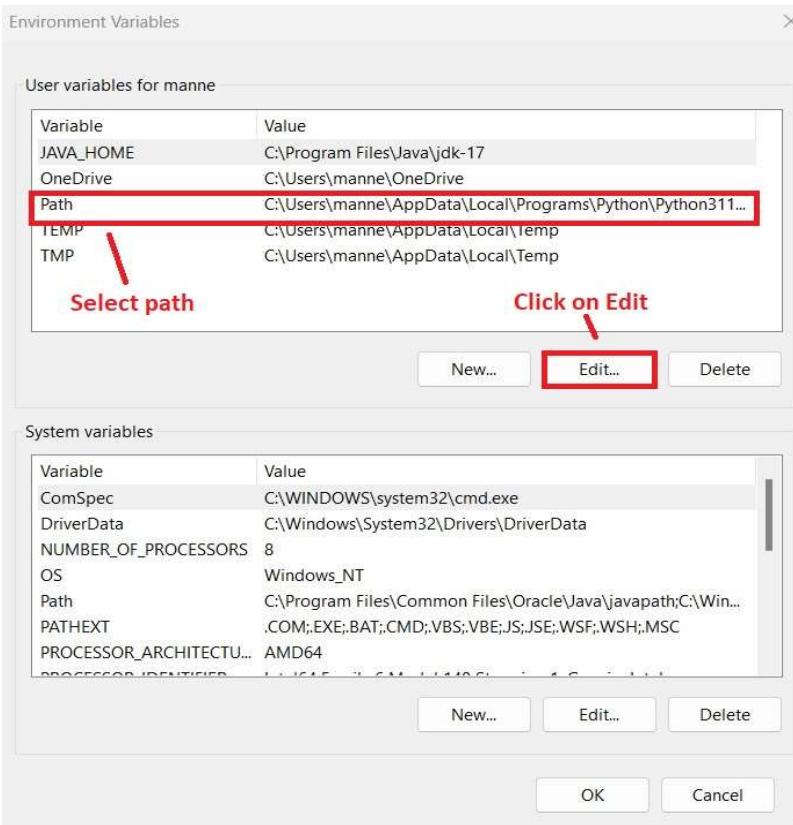
If the content in the above image is not displayed, we have to add AWS-CLI to Environment variables. First search for environment variables settings as follows.



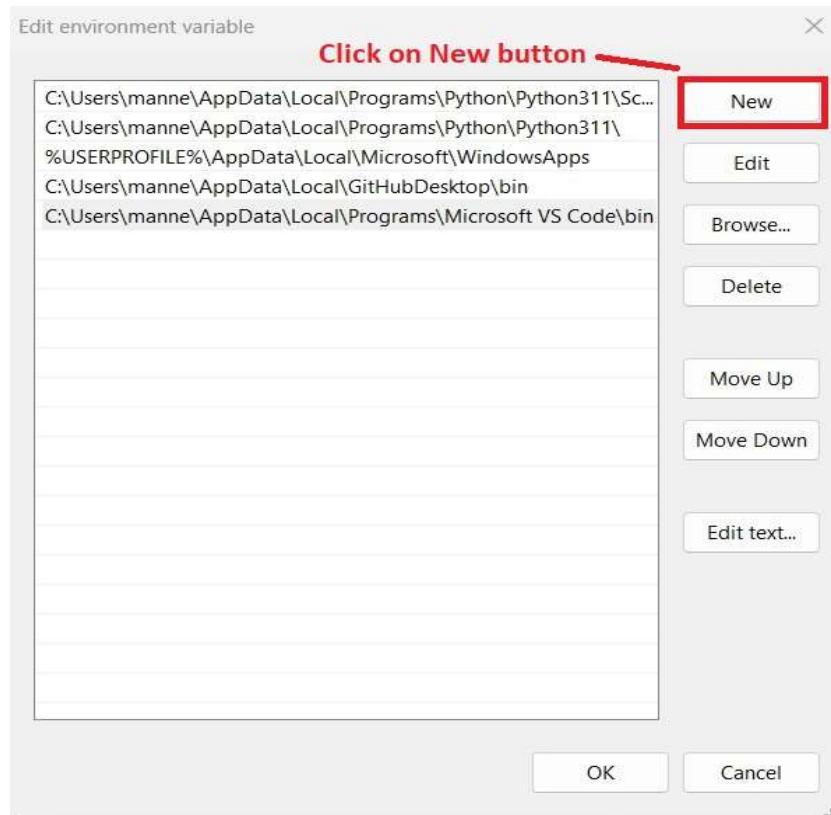
After clicking on the ‘Edit the system environment variables’, the following screen will be displayed.



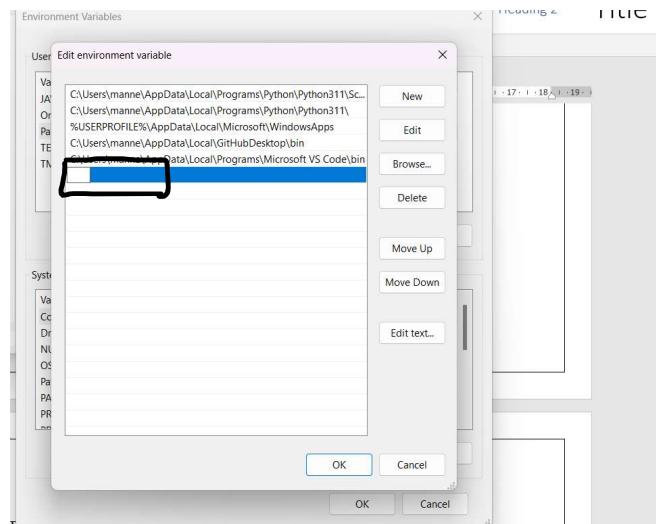
After clicking on that ‘Environment Variables’ button, the following screen will be displayed.



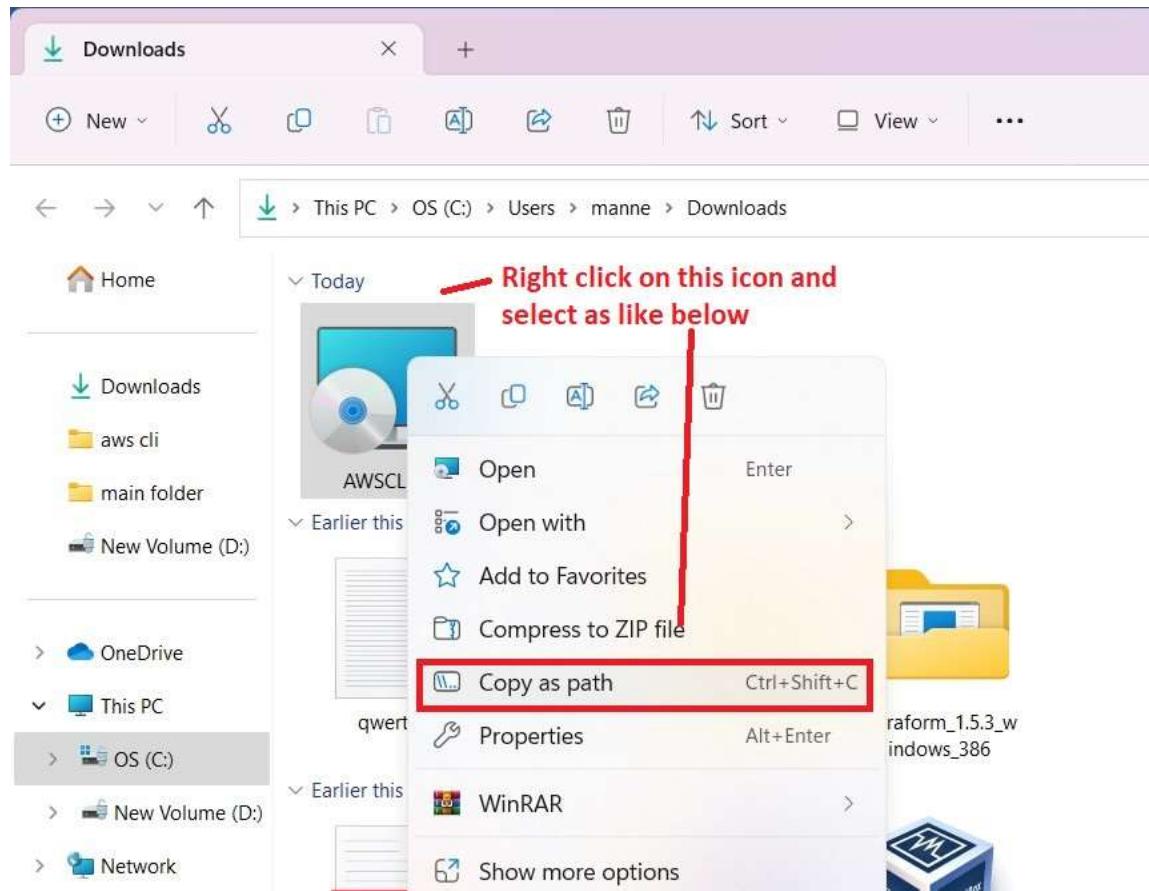
After clicking on that Edit button, the following screen will be displayed.



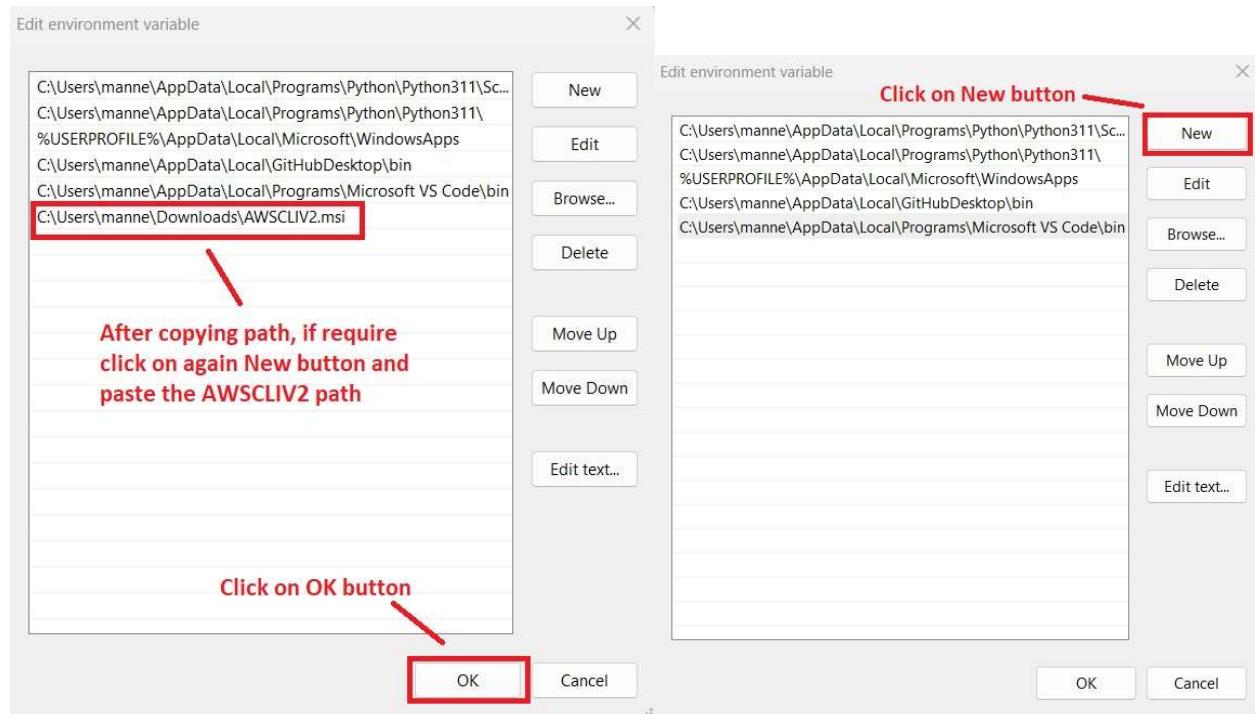
After clicking on new button, the following screen will be displayed.



In the empty field we have to paste the AWS-CLI.MSI path as follows. Goto downloads folder and select AWSCLIV2



After pasting the path, the screen looks like below



After that close all windows related to Environmental variables.

Then again check the aws –version.

## **Now, terraform installation will start.**

To download: <https://developer.hashicorp.com/terraform/downloads>

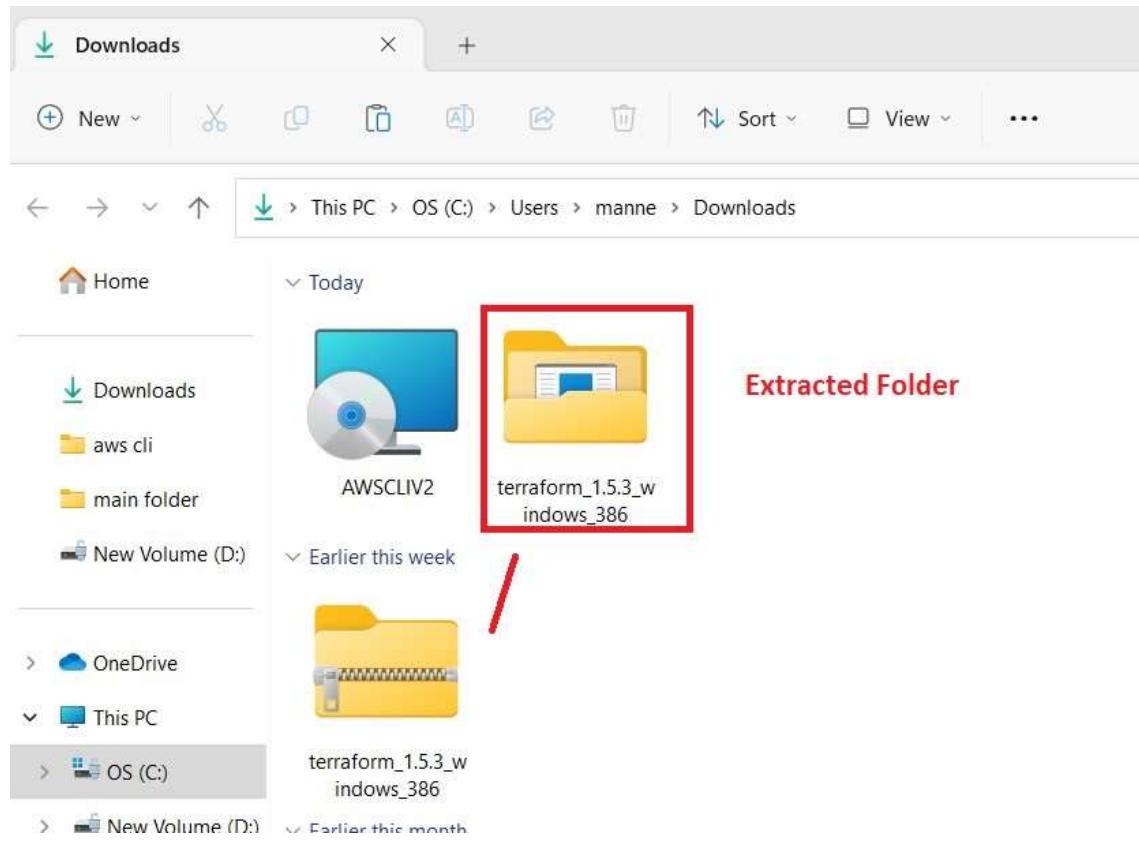
After clicking that link the following page will be displayed.

The screenshot shows the Terraform download page at https://developer.hashicorp.com/terraform/downloads. The main heading is "Install Terraform". Below it, a sub-section titled "Operating System" has "Windows" selected. Under "Binary download for Windows", there are two options: "386" and "AMD64". Each option has a "Download" button. A red arrow points to the "Windows" tab with the text "Click on windows". Another red arrow points to the "Download" button for the AMD64 version with the text "Click on download".

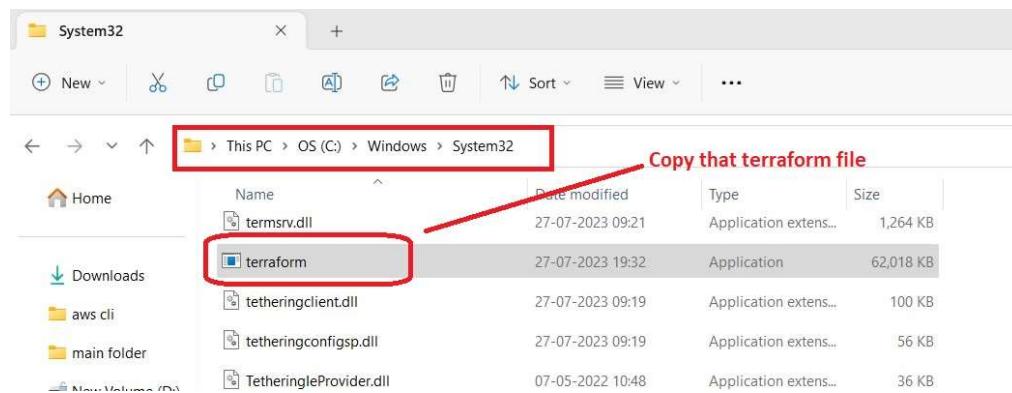
After downloading the terraform, extract the software as shown below.

The screenshot shows a Windows File Explorer window with the path "This PC > OS (C:) > Users > manne > Downloads". Inside the Downloads folder, there is a "terraform\_windows.zip" file. A context menu is open over this file, with the text "Right click on downloaded terraform folder for extracting" overlaid. The menu options include "Open", "Open with", "Open in new tab", "Open in new window", "Extract All...", "Pin to Quick access", and "Pin to Start". A red arrow points to the "Extract All..." option.

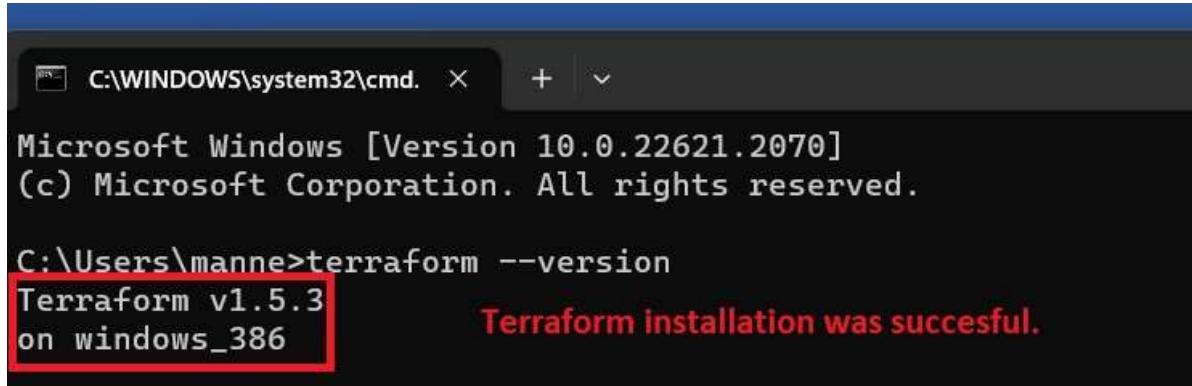
After extracting the folder, it seems like as shown in following screenshot.



After extracting that folder, open that folder and copy the terraform.exe file and paste in the path as shown in following screen. (C:\Windows\System32)



After pasting the terraform.exe file in the above said folder, open command prompt and check for terraform –version as below



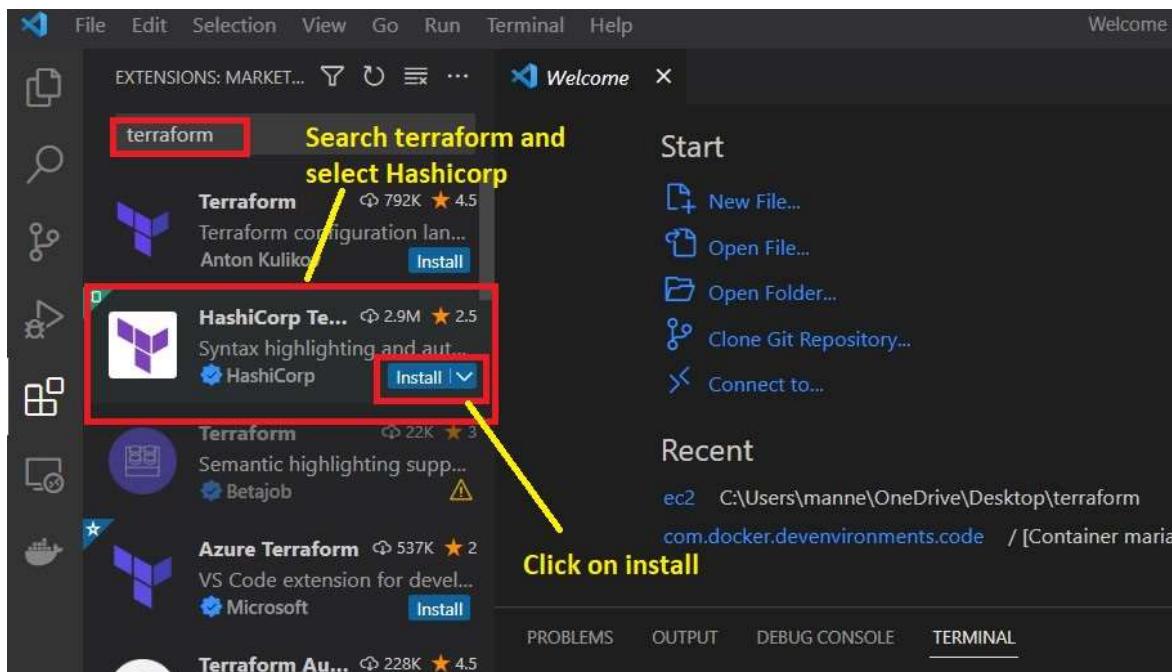
```
C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [Version 10.0.22621.2070]
(c) Microsoft Corporation. All rights reserved.

C:\Users\manne>terraform --version
Terraform v1.5.3
on windows_386
Terraform installation was successful.
```

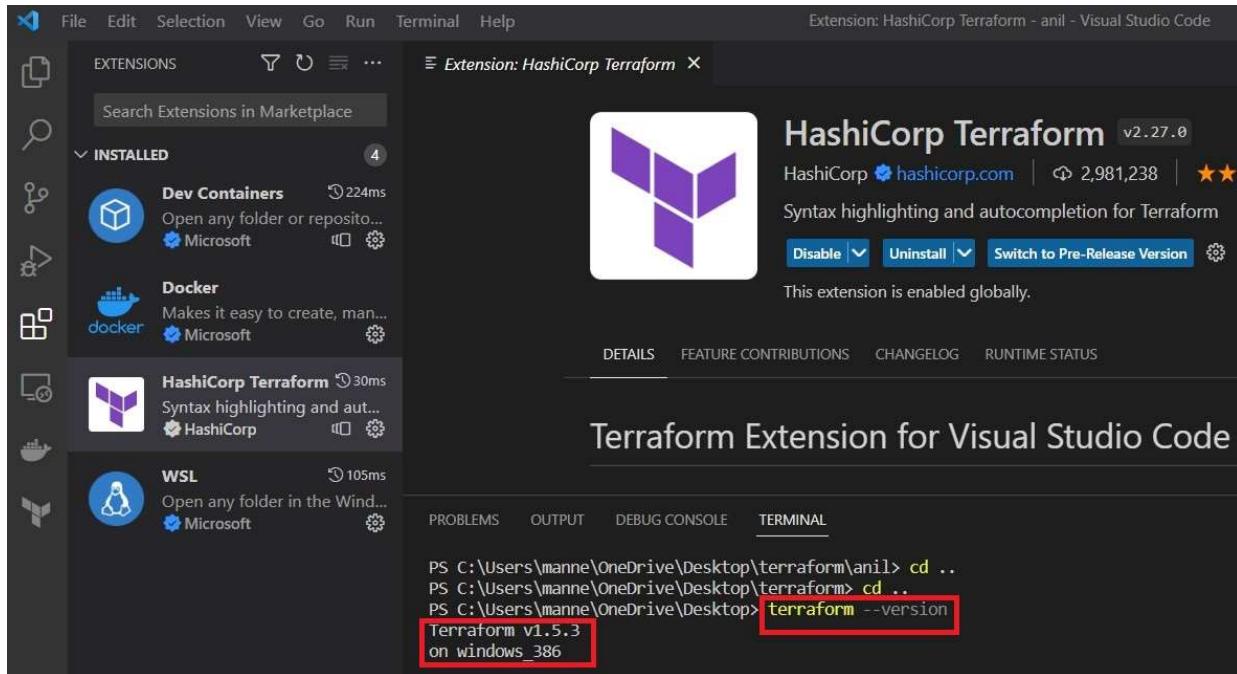
If terraform version found, the installation is successful.

After that, again open Visual studio code and install terraform plugin as shown below.

(Plugin is required to get terraform script suggestions)



After that again check for terraform –version in visual studio and gets validate about successful installation. Screen looks like in next page.

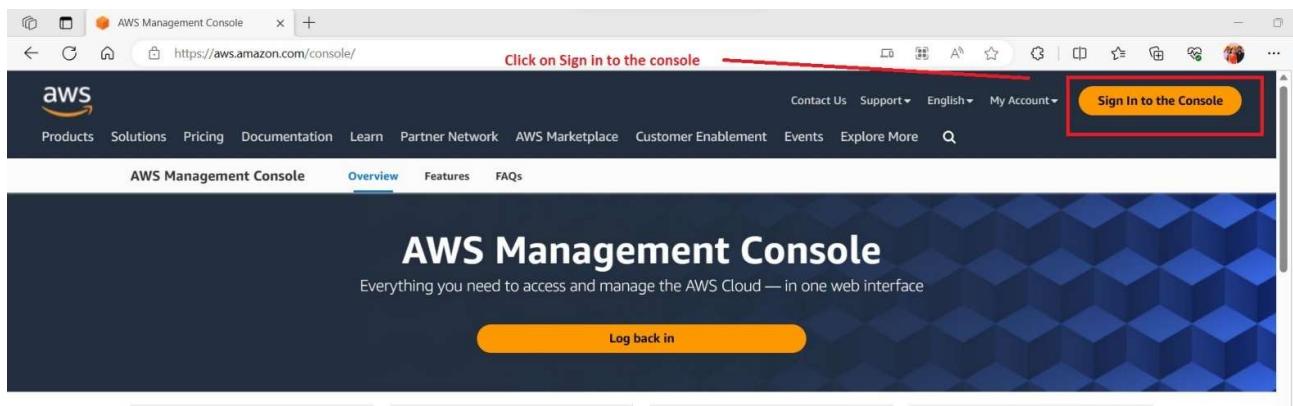


After that we have to integrate AWS (Using Access key and Secret access key) and terraform.

So, to get access key and secret access key from AWS console. (Follow the steps)

URL: <https://aws.amazon.com/console/>

After opening that link the following page will be displayed.



After clicking on ‘sign in to the console’ button, the following page will be displayed.

The screenshot shows the AWS sign-in interface. On the left, there's a 'Sign in' form with two radio button options: 'Root user' (selected) and 'IAM user'. Below these are fields for 'Root user email address' (containing 'username@example.com') and a 'Next' button. On the right, there's a promotional banner for 'Amazon Lightsail' with the text 'Lightsail is the easiest way to get started on AWS' and a 'Learn more »' button.

After clicking on ‘Next’ button, the following page will be displayed

The screenshot shows the 'Root user sign in' page. It has fields for 'Email' (with placeholder '@gmail.com') and 'Password' (with placeholder '\*\*\*\*\*'). There are 'Forgot password?' and 'Sign in' buttons. Below the form are links for 'Sign in to a different account' and 'Create a new AWS account'. To the right, there's a promotional banner for 'SageMaker Fridays' with the text 'Join SageMaker Fridays for live coding, demos, and more' and a 'Register now' button.

After clicking on ‘Sign In’ button, the following page will be displayed.

The screenshot shows the AWS Management Console home page. At the top, there's a search bar with the placeholder 'Enter IAM in search area'. Below the search bar are service icons for EC2, S3, VPC, and Elastic Kubernetes Service. The main area displays 'Console Home' with sections for 'Welcome to AWS' (with a 'Getting started with AWS' link) and 'AWS Health' (showing 0 open issues). There are also 'Reset to default layout' and 'Add widgets' buttons.

After entering IAM in search area, the following page will be displayed.

The screenshot shows the AWS Management Console search results for 'IAM'. The search bar at the top has 'IAM' entered. Below it, a list of results is shown under the heading 'Services'. The first result, 'IAM' with the sub-description 'Manage access to AWS resources', is highlighted with a red box and has a white arrow pointing to it from the text 'Click on IAM'.

After clicking on IAM, the following page will be displayed.

The screenshot shows the IAM Management Console dashboard. On the left, there is a navigation sidebar with sections like 'Dashboard', 'Access management', 'Access reports', etc. The main area is titled 'IAM dashboard' and contains sections for 'Security recommendations', 'IAM resources', and 'What's new'. In the 'AWS Account' sidebar on the right, there is a 'Quick Links' section with a link to 'My security credentials', which is also highlighted with a red box and has a white arrow pointing to it from the text 'Click on My security credentials'.

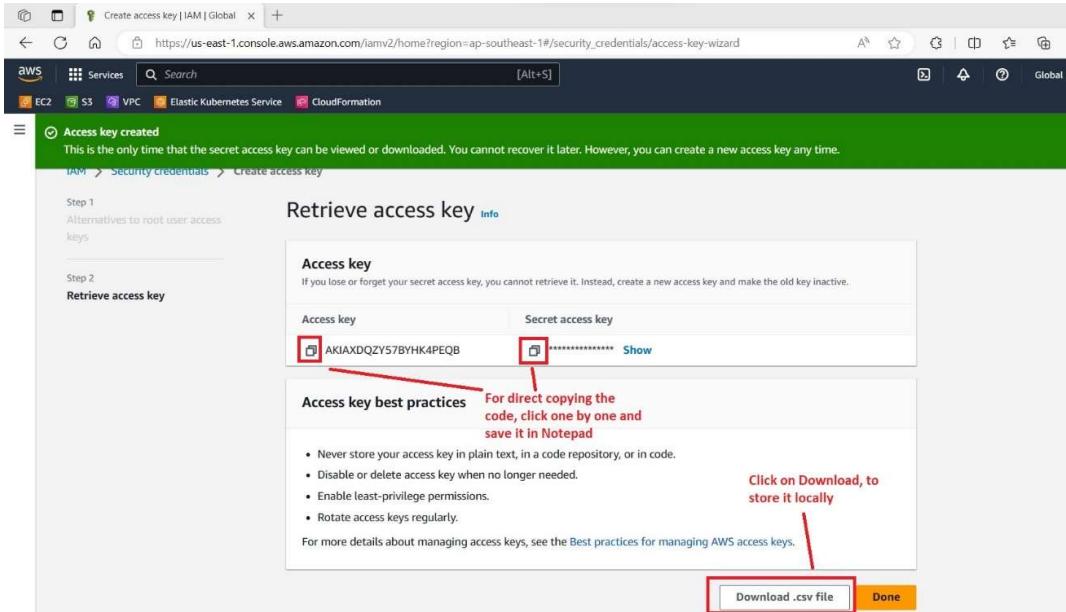
After clicking on ‘My security credentials’ button, page displayed as shown below.

The screenshot shows the AWS IAM Management Console with the URL [https://us-east-1.console.aws.amazon.com/iamv2/home?region=ap-southeast-1#/security\\_credentials](https://us-east-1.console.aws.amazon.com/iamv2/home?region=ap-southeast-1#/security_credentials). The left sidebar shows 'Identity and Access Management (IAM)'. The main content area is titled 'My security credentials (root user)'. It displays a warning: 'You don't have MFA assigned' with a note to assign MFA. Below this is a 'Account details' section with a 'Edit account name, email, and password' button. A red arrow points from the 'Create access key' button in the 'Access keys (0)' section to the 'Create access key' button at the bottom of the page.

After clicking on ‘Create access key’ button the following page will be displayed.

The screenshot shows the 'Create access key' wizard with the URL [https://us-east-1.console.aws.amazon.com/iamv2/home?region=ap-southeast-1#/security\\_credentials/access-key-wizard](https://us-east-1.console.aws.amazon.com/iamv2/home?region=ap-southeast-1#/security_credentials/access-key-wizard). The left sidebar shows 'Step 1 Alternatives to root user access keys'. The main content area is titled 'Alternatives to root user access keys'. It contains a warning about root user access keys and suggests using IAM roles or IAM Identity Center instead. Below this is a 'Continue to create access key?' section with a checkbox labeled 'I understand creating a root access key is not a best practice, but I still want to create one.' and a 'Create access key' button. Red arrows point from the 'Enable the check box' to the checkbox and from the 'Click on button' to the 'Create access key' button.

After enabling check box and clicking on ‘Create access key’, the following page will be displayed.



Finally, we had generated the Access key and secret access key. Alternatively, we had downloaded the file and it will be stored in local.

## Terraform and AWS CLI integration

Open visual studio code and in terminal, by using the following commands

aws configure -- Click on enter button

```
PS C:\Users\manne\Desktop\terraform\anil> aws configure
AWS Access Key ID [None]: AKIAJDQZV57BYHK4PEQB
AWS Secret Access Key [None]: Mz...Tz@HWZLYTsJXayyDnoTFz5De1kDCA50kb
Default region name [None]: ap-south-1
Default output format [None]: json
PS C:\Users\manne\Desktop\terraform\anil>
```

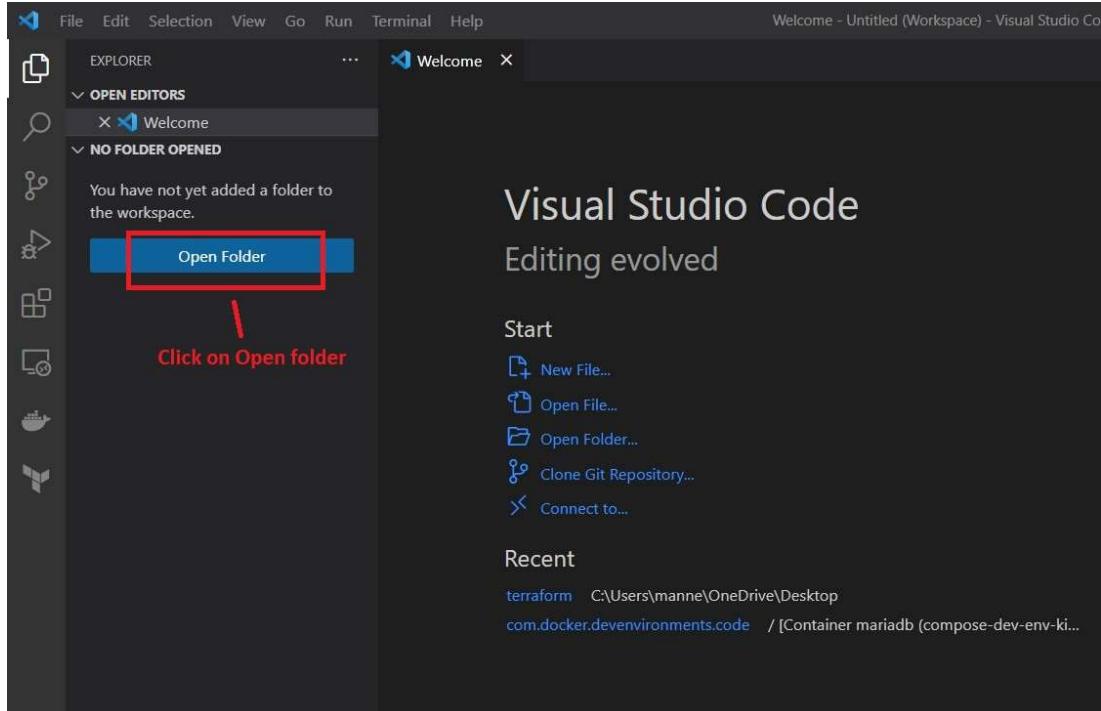
Now you are ready to launch a service in AWS through terraform script.

To select region in AWS console.

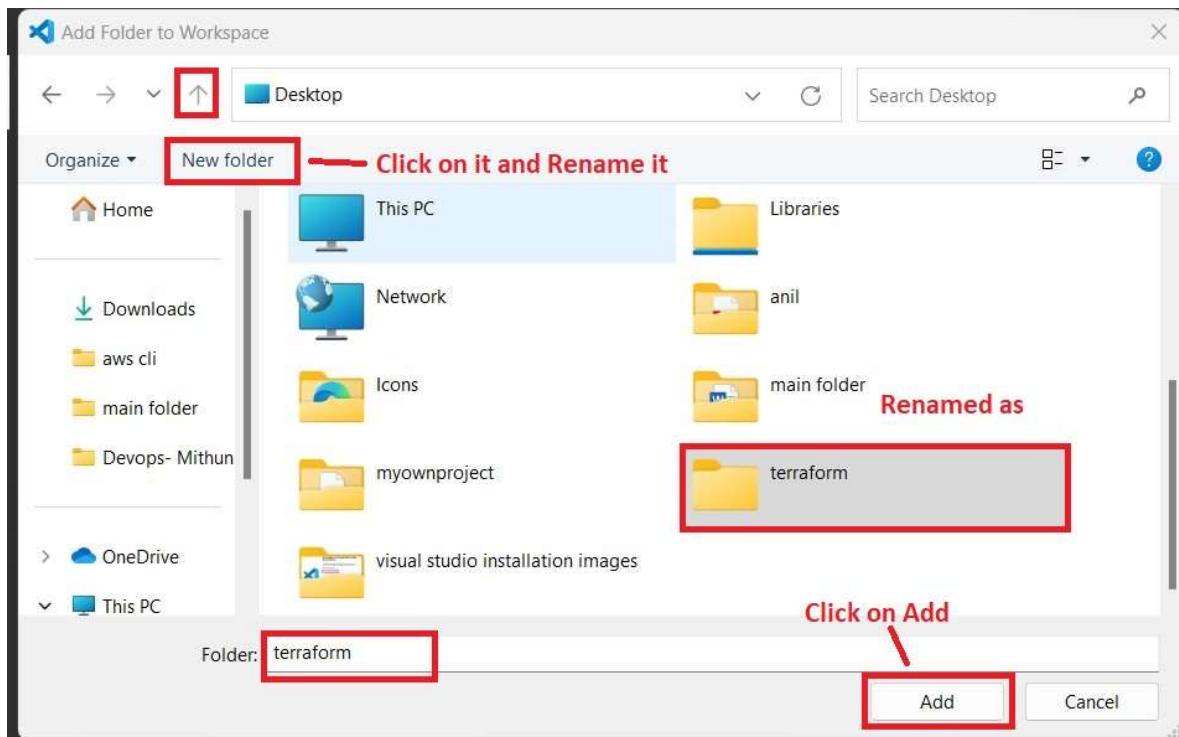
The screenshot shows the AWS Management Console homepage. On the right side, there is a sidebar titled "Reset to default" which lists various AWS regions. The "Asia Pacific (Mumbai)" region, which is highlighted with a red box, is selected. Other regions listed include US East (N. Virginia), US East (Ohio), US West (N. California), US West (Oregon), US West (Oregon) (another entry), Asia Pacific (Osaka), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), and us-east-1, us-east-2, us-west-1, us-west-2, ap-northeast-3, ap-northeast-2, ap-southeast-1, ap-southeast-2, and ap-northeast-1.

## 7) Basic services like EC2, VPC and S3 bucket creation with terraform?

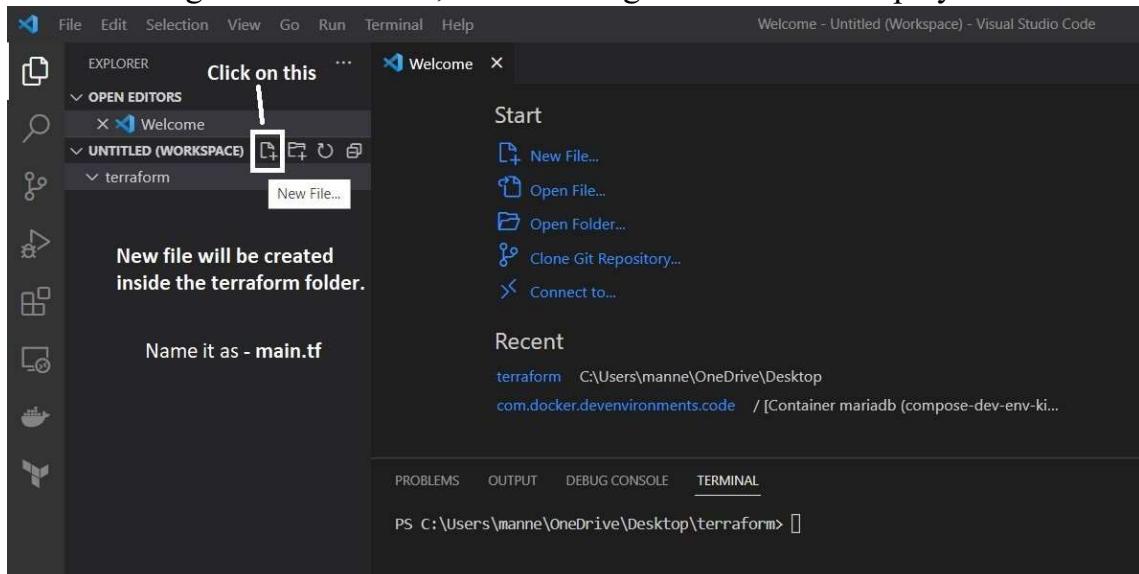
- A) To create any service using terraform, first we need to create an empty folder (in any location) and in that main.tf file have to be created in visual studio code (for script suggestions) Follow step by step. Individually each service will be shown. For each service we have to create new folder and new file i.e., main.tf.



After clicking on open folder, you have to select your own folder location. Shown below.



After clicking on 'Add' button, the following screen will be displayed.



New file is renamed as main.tf and script have to be written in that file.

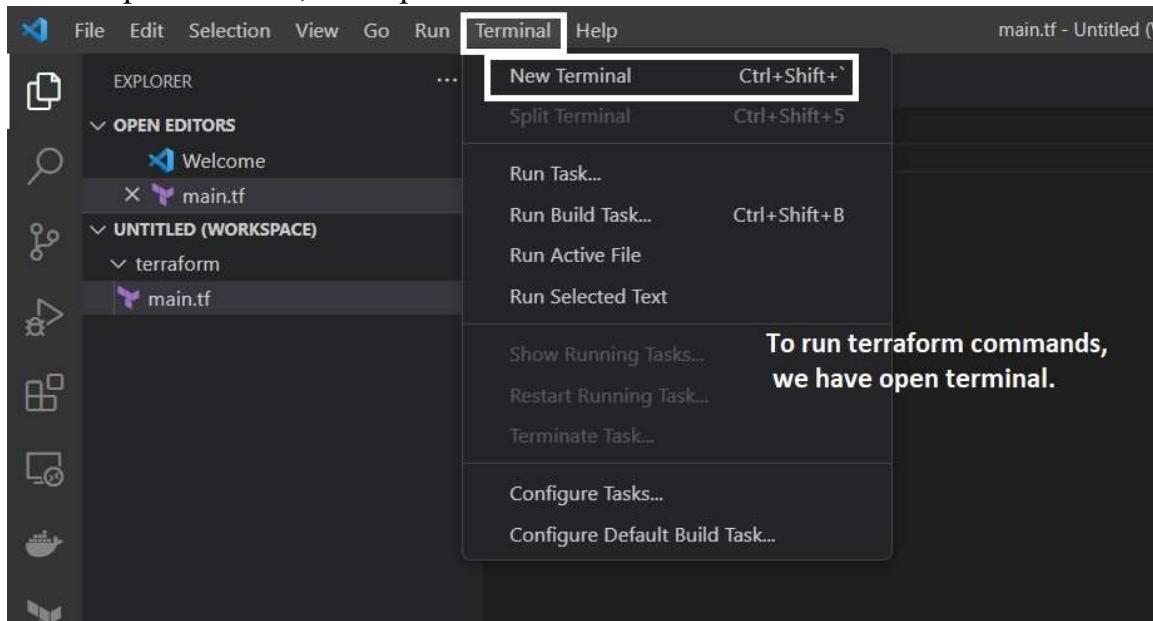
### Main.tf(Script)

```
provider "aws" {
    region = "us-west-2"
}
resource "aws_instance" "web" {
    ami      = "ami-08541bb85074a743a"
    instance_type = "t2.micro"

    tags = {
        Name = "My_First_Instance"
    }
}
```

The screenshot shows the Visual Studio Code interface with the main.tf file open in the editor. The file content is displayed as "Our Script have to written here". The terminal tab is visible at the top.

To open terminal, do steps as follows.



Sample script for Ec2 creation, based on the OS requirement. AMI id will be selected.

The screenshot shows the Visual Studio Code interface with a Terraform configuration script (main.tf) open. The script defines an AWS provider and an AWS instance resource. A tooltip says "After writing script, save the file." Below the editor, a terminal window shows the command "terraform init" being typed. Another tooltip at the bottom says "Run the command 'terraform init' to initialize the supported plugins."

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "web" {
  ami     = "ami-08541bb85074a743a"
  instance_type = "t2.micro"
  tags = {
    Name = "My_First_Instance"
}
```

After saving the run the command, '**terraform init**', screen looks like shown in next page.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\manne\OneDrive\Desktop\terraform> terraform init

Initializing the backend...

Initializing provider plugins...

- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.10.0

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.

PS C:\Users\manne\OneDrive\Desktop\terraform> █

After successful initialization of terraform, run '**terraform plan**' in terminal.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
+ secondary_private_ips          = (known after apply)
+ security_groups                = (known after apply)
+ source_dest_check              = true
+ spot_instance_request_id       = (known after apply)
+ subnet_id                      = (known after apply)
+ tags                           = {
    + "Name" = "My_First_Instance"
}
+ tags_all                       = {
    + "Name" = "My_First_Instance"
}
+ tenancy                         = (known after apply)
+ user_data                       = (known after apply)
+ user_data_base64                = (known after apply)
+ user_data_replace_on_change     = false
+ vpc_security_group_ids          = (known after apply)
```

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

It is just a dry run, Changes will be shown here

After that, you have to run the command '**terraform apply --auto-approve=true**' in terminal for infrastructure creation in AWS.

```
+ secondary_private_ips          = (known after apply)
+ security_groups                = (known after apply)
+ source_dest_check              = true
+ spot_instance_request_id       = (known after apply)
+ subnet_id                      = (known after apply)
+ tags                           = {
    + "Name" = "My_First_Instance"
}
+ tags_all                       = {
    + "Name" = "My_First_Instance"
}
+ tenancy                         = (known after apply)
+ user_data                       = (known after apply)
+ user_data_base64                = (known after apply)
+ user_data_replace_on_change     = false
+ vpc_security_group_ids          = (known after apply)
```

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

```
aws_instance.web: Creating...
aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Still creating... [20s elapsed]
```

Instance creation under progress

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

+ security_groups = (known after apply)
+ source_dest_check = true
+ spot_instance_request_id = (known after apply)
+ subnet_id = (known after apply)
+ tags =
  + "Name" = "My_First_Instance"
}
+ tags_all =
  + "Name" = "My_First_Instance"
}
+ tenancy = (known after apply)
+ user_data = (known after apply)
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.
aws_instance.web: Creating...
aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Still creating... [20s elapsed]
aws_instance.web: Still creating... [30s elapsed]
aws_instance.web: Creation complete after 36s [id=i-0731740d23b42c6bd] Instance created successfully in AWS

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\manineeth\Desktop\terraform>

```

View the instance in AWS console.

To terminate the instance, run the command ‘**terraform destroy --auto-approve=true**’ in terminal for infrastructure creation in AWS.

```

- tags = {} -> null
- throughput = 125 -> null
- volume_id = "vol-02e5b775af393fad1" -> null
- volume_size = 8 -> null
- volume_type = "gp3" -> null
}

Plan: 0 to add, 0 to change, 1 to destroy.
aws_instance.web: Destroying... [id=i-01b3de39aa141b819]
aws_instance.web: Still destroying... [id=i-01b3de39aa141b819, 10s elapsed]
aws_instance.web: Still destroying... [id=i-01b3de39aa141b819, 20s elapsed]
aws_instance.web: Still destroying... [id=i-01b3de39aa141b819, 30s elapsed]
aws_instance.web: Destruction complete after 32s Instance terminated

Destroy complete! Resources: 1 destroyed.
PS C:\Users\manineeth\Desktop\terraform>

```

View the termination status in AWS.

The screenshot shows the AWS EC2 Management Console with the URL [https://us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#Instances:instanceState=terminated;v=3;\\$case=tags:true%5Cclient:](https://us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#Instances:instanceState=terminated;v=3;$case=tags:true%5Cclient:). The search bar contains "[Alt+S]". The main pane displays "Instances (3) Info" with a filter set to "Instance state = terminated". A table lists one instance: "My\_First\_Instance" with Instance ID "i-01b3de39aa141b819", Instance state "Terminated", Instance type "t2.micro", and Status "-".

To create VPC, again we have to create a new folder inside terraform folder and named it as VPC. Again, create a file with name ‘main.tf’,

The screenshot shows Visual Studio Code with the title "Welcome - Untitled (Workspace) - Visual Studio Code". The Explorer sidebar shows a workspace structure with "UNTITLED (WORKSPACE)" containing "terraform" and "VPC". A "New folder" option is highlighted with a callout "To create folder inside, terraform.". The terminal at the bottom shows the command "PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2>".

The screenshot shows Visual Studio Code with the title "main.tf - Untitled (Workspace) - Visual Studio Code". The Explorer sidebar shows the workspace structure with "terraform" and "VPC" selected. Inside "VPC", a new file "main.tf" is being created, indicated by a callout "New file created main.tf file named as main.tf". A second click on the "main.tf" file is shown with a callout "Second click on". The terminal at the bottom shows the command "1 main.tf". A callout "Script to be written here." points to the main.tf file in the editor.

## Main.tf script

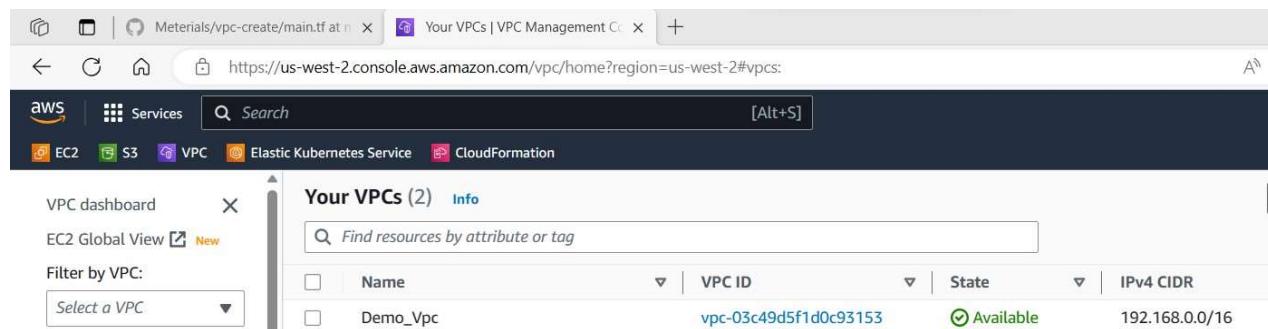
```
provider "aws" {
  region = var.aws_region
}
resource "aws_vpc" "main" {
  cidr_block = var.cidr_block[0]
  tags = {
    Name = var.vpc_name
  }
}
resource "aws_subnet" "subnet1" {
  vpc_id = aws_vpc.main.id
  cidr_block = var.cidr_block[1]
  tags = {
    Name = var.subnet_name[0]
  }
}
resource "aws_subnet" "subnet2" {
  vpc_id = aws_vpc.main.id
  cidr_block = var.cidr_block[2]
  tags = {
    Name = var.subnet_name[1]
  }
}
```

After saving the file, we have to run the command '**terraform init**' in terminal.

After that, successful initialization run the command '**terraform plan**' in terminal.

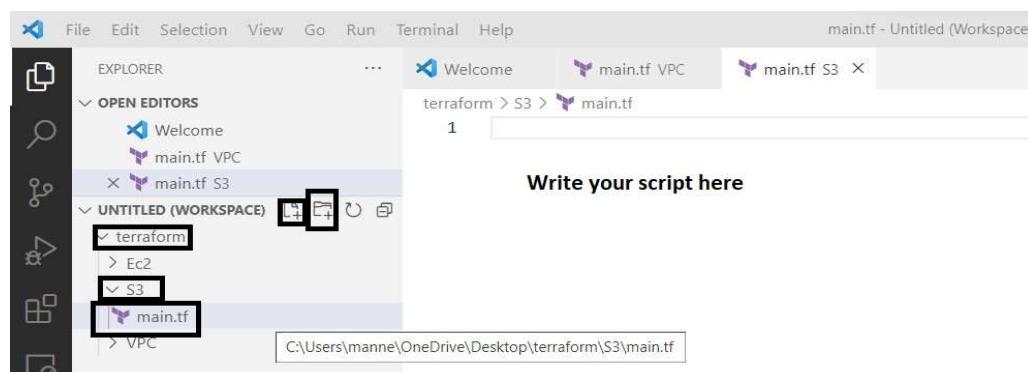
After that, run the command '**terraform apply --auto-approve=true**' in terminal.

After that, run the command '**terraform destroy --auto-approve=true**' in terminal.



The screenshot shows the AWS VPC Management console. At the top, there's a navigation bar with tabs like 'Materials/vpc-create/main.tf at n' and 'Your VPCs | VPC Management'. Below the navigation bar, there's a search bar and a breadcrumb trail 'https://us-west-2.console.aws.amazon.com/vpc/home?region=us-west-2#vpes:'. The main content area is titled 'Your VPCs (2)' and contains a table with two rows. The table has columns for 'Name', 'VPC ID', 'State', and 'IPv4 CIDR'. The first row shows 'Demo\_Vpc' with 'vpc-03c49d5f1d0c93153' and 'Available' status. The second row is partially visible. A sidebar on the left shows 'EC2 Global View' and a dropdown menu 'Select a VPC'.

To create S3 bucket, again we have to create a new folder inside terraform folder and named it as S3. Again, create a file with name 'main.tf'.



The screenshot shows the Visual Studio Code interface. The left sidebar is the 'EXPLORER' view, which shows a tree structure with 'OPEN EDITORS' containing 'Welcome' and 'main.tf VPC', and an 'UNTITLED (WORKSPACE)' folder containing 'terraform', 'Ec2', 'S3', and 'main.tf'. The 'main.tf' file is currently selected. The right side of the screen is a code editor with the title 'main.tf - Untitled (Workspace)'. The code editor has a placeholder 'Write your script here' and shows the contents of the 'main.tf' file. The status bar at the bottom indicates the file path 'C:\Users\manne\OneDrive\Desktop\terraform\S3\main.tf'.

## S3 Script

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_s3_bucket" "testbkt" {
  bucket = "myftestbkt-kelly999"

  tags = {
    Name      = "My bucket"
  }
}
```

After saving the file, we have to run the command ‘**terraform init**’ in terminal.

After that, successful initialization run the command ‘**terraform plan**’ in terminal.

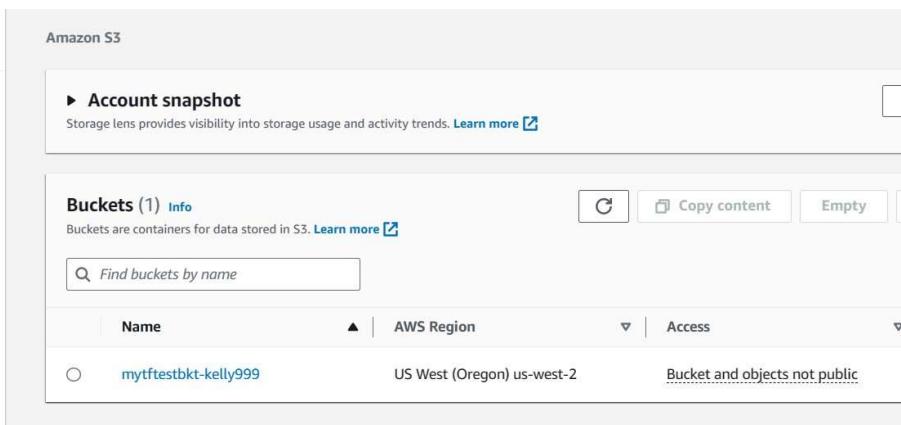
After that, run the command ‘**terraform apply –auto-approve=true**’ in terminal.

After that, run the command ‘**terraform destroy –auto-approve=true**’ in terminal.

```
}
```

+ tags\_all = {  
+ "Name" = "My bucket"  
}  
+ website\_domain = (known after apply)  
+ website\_endpoint = (known after apply)  
}  
  
Plan: 1 to add, 0 to change, 0 to destroy.  
aws\_s3\_bucket.testbkt: Creating...  
aws\_s3\_bucket.testbkt: Still creating... [10s elapsed]  
aws\_s3\_bucket.testbkt: Still creating... [20s elapsed]  
aws\_s3\_bucket.testbkt: Creation complete after 23s [id=myftestbkt-kelly999]  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
DE C:\Users\mmano\OneDrive\Desktop\terraform\ex\

After creating S3 bucket, view it in the AWS console.



The screenshot shows the AWS S3 console interface. On the left, there is a navigation sidebar with links like 'Buckets', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'IAM Access Analyzer for S3', 'Block Public Access settings for this account', 'Storage Lens', and 'Dashboards'. The main content area has a header 'Amazon S3' and 'Amazon S3'. Below the header, there's a 'Account snapshot' section with a note about storage usage and activity trends. The 'Buckets' section shows one bucket named 'myftestbkt-kelly999' located in 'US West (Oregon) us-west-2' with 'Bucket and objects not public' access. There are buttons for 'Copy content' and 'Empty'.

## 8) Using terraform script instance creation, connecting and installation of software's?

A) After writing script in main.tf file, save it and execute the following commands in terminal. For Ec2 instance creation and connecting to it we require key-pair. So, initially we have to create a key-pair. As follows

First create a directory to store the key-pair and after that enter into that directory and execute the command '**ssh-keygen -t rsa -f demo**' (key-pair name is demo)

```
25 mkdir key_pair  
26 cd .\key_pair\  
  
PS C:\Users\manne\OneDrive\Desktop\terraform\key_pair> ssh-keygen -t rsa -f demo  
Generating public/private rsa key pair.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in demo  
Your public key has been saved in demo.pub  
The key fingerprint is:  
SHA256:Vsgg18tsx9XaBqkgsaTFKgf03gd8jju7a/DU2/roCw manne@Anil  
The key's randomart image is:  
+---[RSA 3072]---+  
|o* .  
|B++ .  
|=+* .  
.B.= = .  
.+= + B S  
.o . o *  
o + + +.  
| B E,* .o  
. o ooo oo.  
+---[SHA256]----+  
PS C:\Users\manne\OneDrive\Desktop\terraform\key_pair> ls  
  
Directory: C:\Users\manne\OneDrive\Desktop\terraform\key_pair  
  
Mode LastWriteTime Length Name  
---- ----- ----- ----  
-a--- 30-07-2023 14:44 2590 demo  
-a--- 30-07-2023 14:44 565 demo.pub
```

### Main.tf (script)

```
provider "aws" {  
    region = "us-west-2" # Replace with your preferred AWS region  
}  
resource "aws_key_pair" "test" {  
    key_name    = "demo"  
    public_key = file("C:/Users/manne/OneDrive/Desktop/terraform/demo.pub")  
}  
resource "aws_instance" "web" {  
    ami          = "ami-0c65adc9a5c1b5d7c" # Replace with your desired AMI ID  
    instance_type = "t2.micro"           # Replace with your desired instance type  
    key_name    = aws_key_pair.test.key_name    # Replace with your key pair name  
    vpc_security_group_ids = ["sg-026d6806f077f66a5"] # your security group id to be replaced.  
    tags = {  
        Name = "Demo_Instance"  
    }  
    connection {  
        type    = "ssh"  
        user    = "ubuntu"  
        private_key = file("C:/Users/manne/OneDrive/Desktop/terraform/demo") # Replace with your private key path  
        host    = self.public_ip  
        timeout = "1m"  
        agent   = false  
    }  
    provisioner "remote-exec" {  
        inline = [  
            "sudo apt-get update",  
            "sudo apt-get install tomcat9 tomcat9-admin -y", #Installing tomcat server.  
        ]  
    }  
}
```

## We have to run the command ‘terraform init’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\instance connect with key> terraform init
```

Initializing the backend...

Initializing provider plugins...

- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.10.0

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.

## After that, successful initialization run the command ‘terraform plan’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\instance connect with key> 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.web will be created
+ resource "aws_instance" "web" {
    + ami                                = "ami-0c65adc9a5c1b5d7c"
    + arn                                = (known after apply)
    + associate_public_ip_address          = (known after apply)
    + key_name                            = "demo"
    + key_name_prefix                     = (known after apply)
    + key_pair_id                         = (known after apply)
    + key_type                            = (known after apply)
    + public_key                          = "ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQC40y5A380J0I3XgbxZj9eSIHxHg1oF2HurGkRu2zGkb1+4z0vXMy1Ce1viEjQMGyCbk7DK4Qsu9VR3k01r
lskffmHoACEg4ymET9UFopVkj+96a1jd1LKRMSok+jyPVnk1qwlrevCj0MjPt+B7T3rp1cw/eniblw0zjX6vgVA5mnQ9d53SBHqn4XIdhlygpl1KK90W34ZXLPwGcNTEm0lii60l1FxLgy+e
06hRicVzL9hkklYc0uwLq0otD17qMEDt/1VWa9w4MfcKgbZachjN6jC6roqujTFFRF1Lqb244m66+Yij5oarsYwYphdZzscedi+b+8ts40QiXxKi1luAfUiWeYDBwPxhzBypGzphaBQb1+YLVZE
qYN2DF4nnGCUKh316FkefimhTxvKmN50+CyQLzd1h0B8Xw+J2eptIawlkWn6H93z4RLz+eiCws30lf/202HfmzXsPAZosmlne3P/K4h6odILiiziEDHh2q5oYCsj+wM7IU= manne@Anil"
    + tags_all                           = (known after apply)
}
```

Plan: 2 to add, 0 to change, 0 to destroy.

## After that, for creation run the command ‘terraform apply –auto-approve=true’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\instance connect with key> terraform apply --auto-approve=true
```

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.web will be created
+ resource "aws_instance" "web" {
    + ami                                = "ami-0c65adc9a5c1b5d7c"
    + 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)
    ...
aws_instance.web (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wants/tomcat9.service → /lib/systemd/system/tomcat9.service.
aws_instance.web (remote-exec): Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
aws_instance.web (remote-exec): Processing triggers for rsyslog (8.2001.0-1ubuntu1.3) ...
aws_instance.web (remote-exec): Processing triggers for man-db (2.9.1-1) ...
aws_instance.web (remote-exec): Processing triggers for ca-certificates (2021016ubuntu0.20.04.1) ...
aws_instance.web (remote-exec): Updating certificates in /etc/ssl/certs...
aws_instance.web (remote-exec): 0 added, 0 removed; done.
aws_instance.web (remote-exec): Running hooks in /etc/ca-certificates/update.d...

aws_instance.web: Still creating... [1m20s elapsed]
aws_instance.web (remote-exec): done.
aws_instance.web (remote-exec): done.
aws_instance.web: Creation complete after 1m28s [id=i-084149dbb98921116]
```

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

## After that, for termination run the command ‘terraform destroy –auto-approve=true’

```
PS C:\Users\manne\OneDrive\Desktop\terraform\tomcat instally> terraform destroy --auto-approve=true
aws_key_pair.test: Refreshing state... [id=demo]
aws_instance.web: Refreshing state... [id=i-084149dbb98921116]
```

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.web will be destroyed
- resource "aws_instance" "web" {
    - ami                                = "ami-0c65adc9a5c1b5d7c" -> null
    - arn                                = "arn:aws:ec2:us-west-2:488606592963:instance/i-084149dbb98921116" -> null
    - associate_public_ip_address          = true -> null
```

## 9) What is Terraform modules and Example script?

- A) Terraform module is an encapsulated and reusable collection of Terraform configurations that represent a specific piece of infrastructure or a set of related resources. Modules help you organize your Terraform code and promote best practices like modularity, reusability, and maintainability.

### Main.tf script

```
provider "aws" {
  region = "us-west-2"
}

module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  name = "my-vpc"
  cidr = "10.0.0.0/16"
  azs      = ["us-west-2a", "us-west-2b", "us-west-2c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24"]
  public_subnets = ["10.0.101.0/24", "10.0.102.0/24"]
  enable_nat_gateway = false
  enable_vpn_gateway = false
  tags = {
    Terraform = "true"
    Environment = "dev"
  }
}

module "ec2_instance" {
  source = "terraform-aws-modules/ec2-instance/aws"
  name = "single-instance"
  instance_type      = "t2.micro"
  key_name           = "demo_key"
  monitoring         = false
  vpc_security_group_ids = ["sg-07ba58c4c9914f12a"]
  subnet_id          = "subnet-0c72c059eb405c8b3"
  tags = {
    Terraform = "true"
    Environment = "dev"
  }
}
```

After saving the file, we have to run the command '**terraform init**' in terminal.

After that, successful initialization run the command '**terraform plan**' in terminal.

After that, run the command '**terraform apply --auto-approve=true**' in terminal.

After that, run the command '**terraform destroy --auto-approve=true**' in terminal.

## 10) Terraform workspaces and cloud configuration?

- A) To create terraform registry account and configuring work spaces, follow the steps as mentioned below.

To register Link : <https://registry.terraform.io>

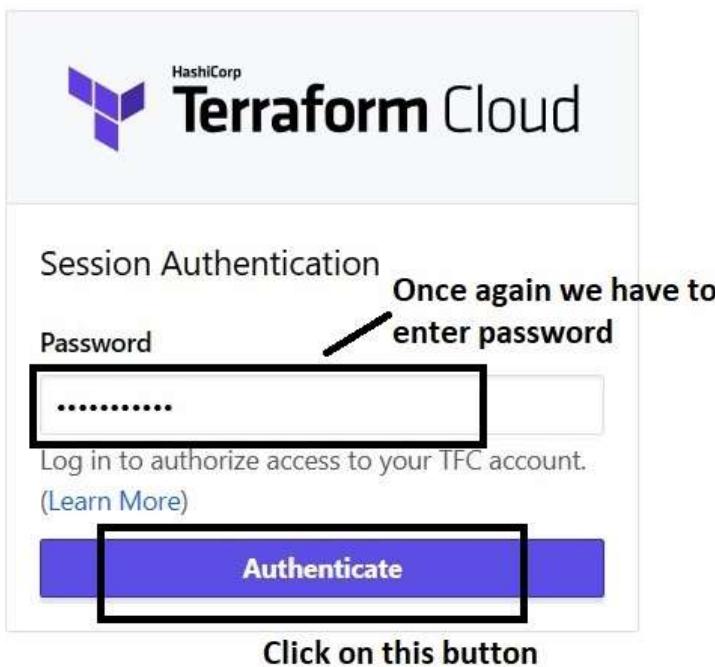
After opening the link, the following page will appear.

The screenshot shows the Terraform Registry homepage. At the top, there is a navigation bar with links for 'Materials/ec2-create/main.tf at r' (closed), 'Terraform Registry' (active tab), 'Browse', 'Publish', 'Sign-in', and a button labeled 'Use Terraform Cloud for free'. Below the navigation bar is a search bar with the placeholder 'Search all resources'. The main content area has a purple header with the text 'Terraform Registry' and a sub-header: 'Discover Terraform providers that power all of Terraform's resource types, or find modules for quickly deploying common infrastructure configurations.' Below this are four buttons: 'Browse Providers', 'Browse Modules', 'Browse Policy Libraries', and 'Browse Run Tasks'. A callout bubble with an arrow points to the 'Use Terraform Cloud for free' button, with the text 'Click on this button'.

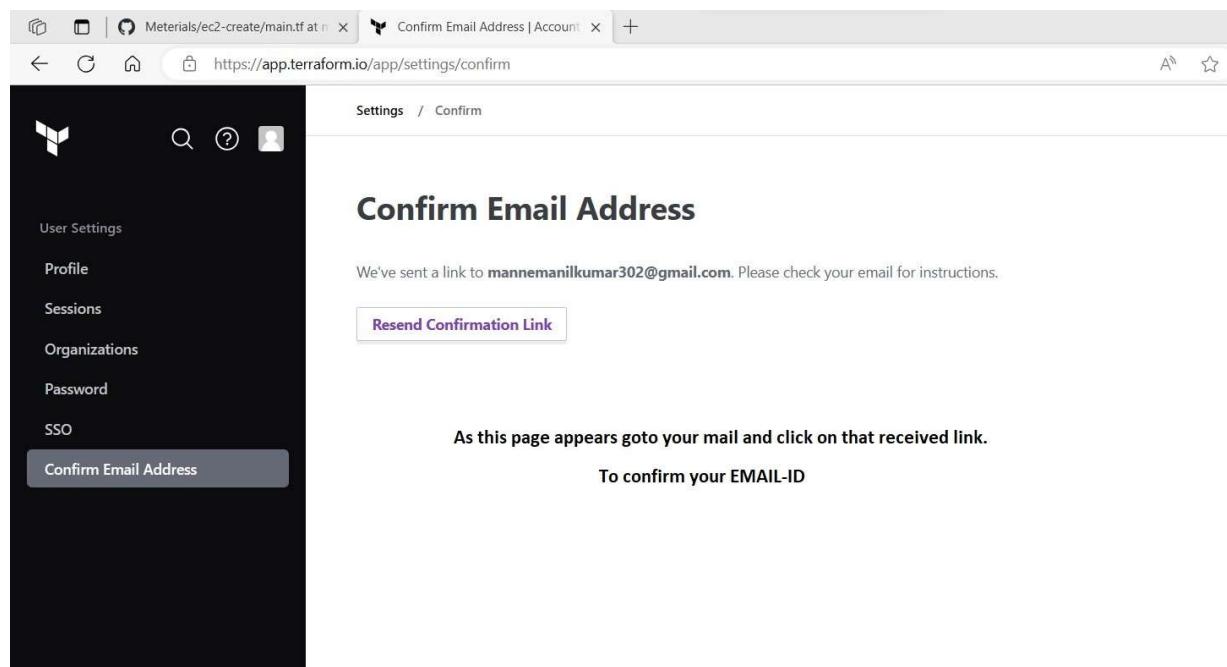
After clicking on that button, the following page will be displayed.

The screenshot shows the 'Create an account' page for Terraform Cloud. The URL in the address bar is 'https://app.terraform.io/public/signup/account'. The page features a large 'Continue with HCP account' button with a blue icon. Below it is a horizontal line with the word 'OR' in the center. There are three input fields: 'Username' containing 'Sample', 'Email' containing 'sample@gmail.com', and 'Password' containing '\*\*\*\*\*'. To the right of these fields is a sidebar with the Terraform logo and the text 'You're minutes away from launching your first infrastructure project!'. It also lists three checked checkboxes: 'Single workspace', 'Write infrastructure', and 'Re-use code across multiple projects'. A 'Learn more' button is at the bottom of the sidebar. At the bottom left, there are two checkboxes: 'I agree to the Terms of Use.' and 'I acknowledge the Privacy Policy.'. A callout bubble with an arrow points to the 'Create account' button at the bottom left, with the text 'Finally click on this button'.

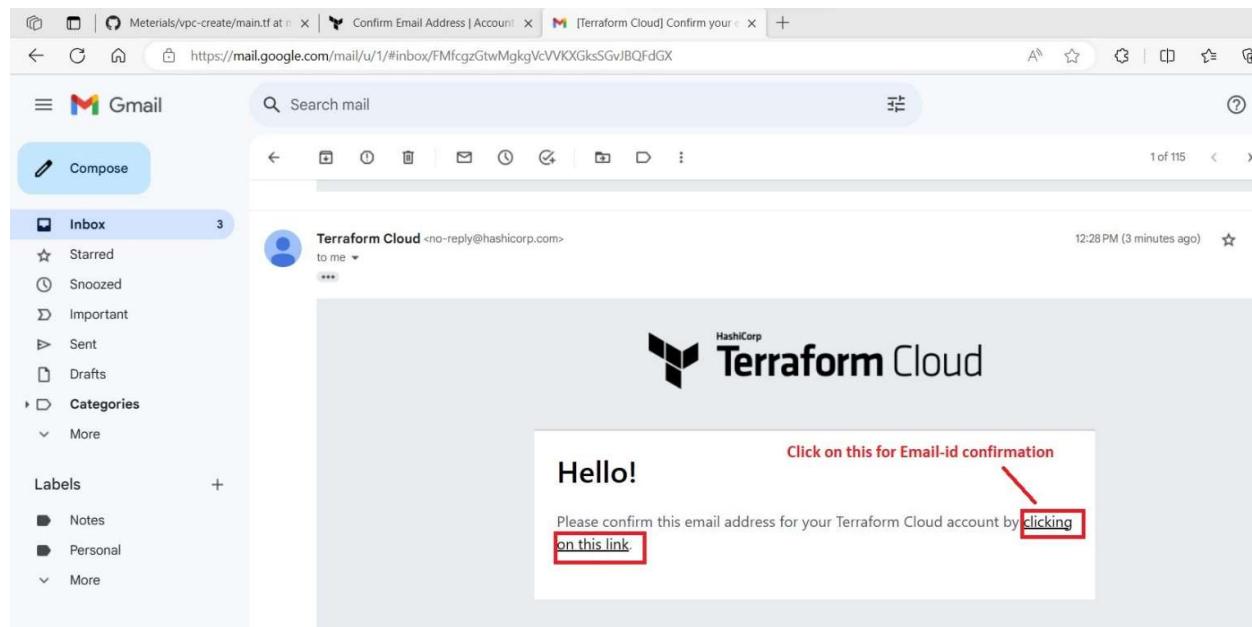
After clicking on button ‘create account’, the following page will appear. Again, we have to enter our password. Which was set by us at the time of account creation.



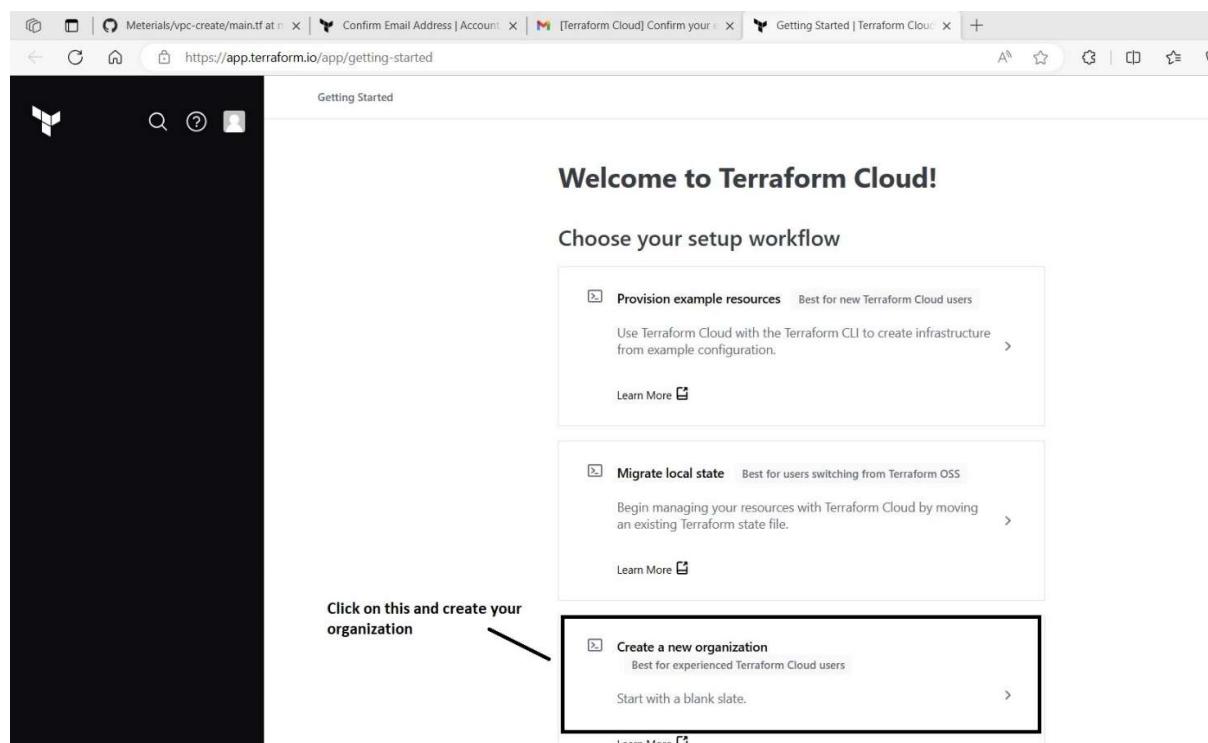
After clicking on the button ‘Authenticate’, the following page will appear.



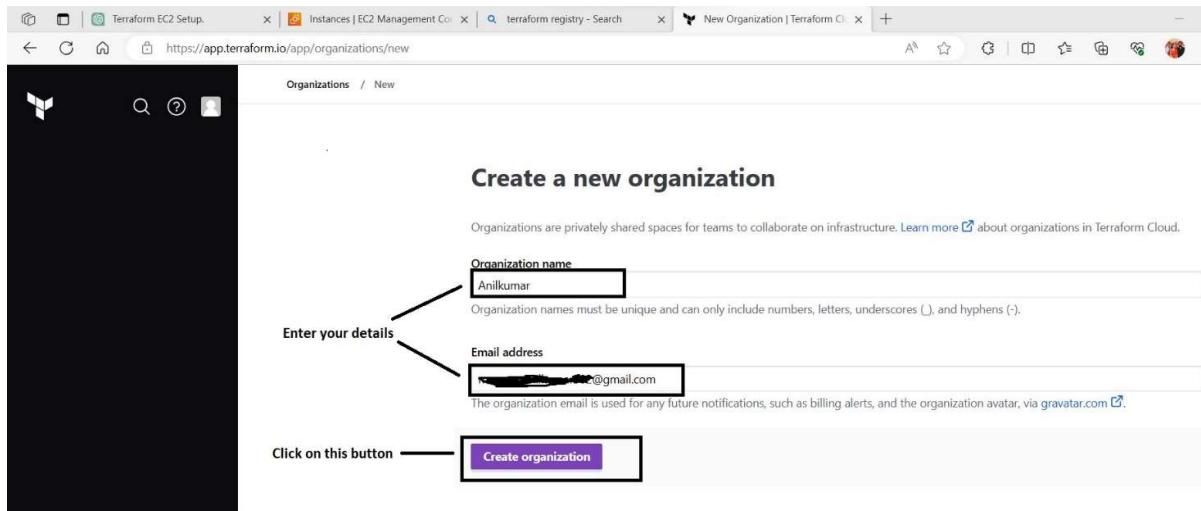
As this page appears, you have to login into your mail account through system as shown.



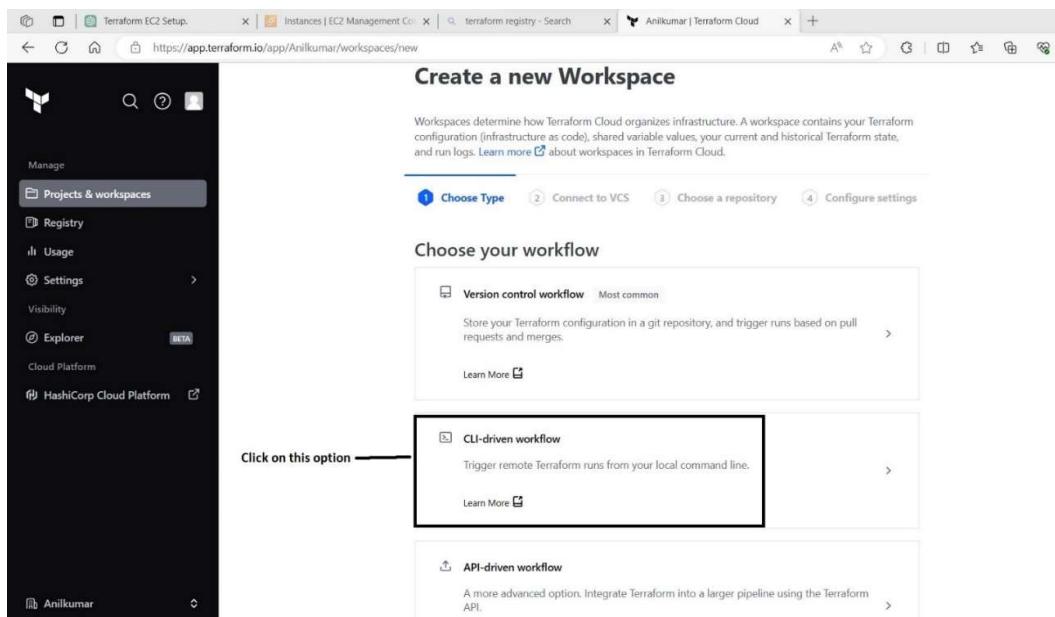
After clicking on the confirmation link, the following page will appear.



After clicking on the option ‘Create a new organization’ the following page will appear.



After clicking on that button ‘Create organization’ the following page will be appeared.



After selecting that option ‘CLI-driven workflow’, the following page will be displayed.

The screenshot shows a browser window with the URL <https://app.terraform.io/app/Anilkumar/workspaces/mannem>. The page title is "my workspace" and the status is "Unlocked". The main heading is "Waiting for configuration" with a progress bar indicating "Checking for configuration". A note states: "This workspace currently has no Terraform configuration files associated with it. Terraform Cloud is waiting for the configuration to be uploaded." Below this, under "CLI-driven runs", there are four numbered steps:

1. Ensure you are properly authenticated into Terraform Cloud by running `terraform login` on the command line or by using a credentials block [🔗](#).
2. Add a code block to your Terraform configuration files to set up the cloud integration. You can add this configuration block to any `.tf` file in the directory where you run Terraform.
3. Run `terraform init` to initialize the workspace.
4. Run `terraform apply` to start the first run for this workspace.

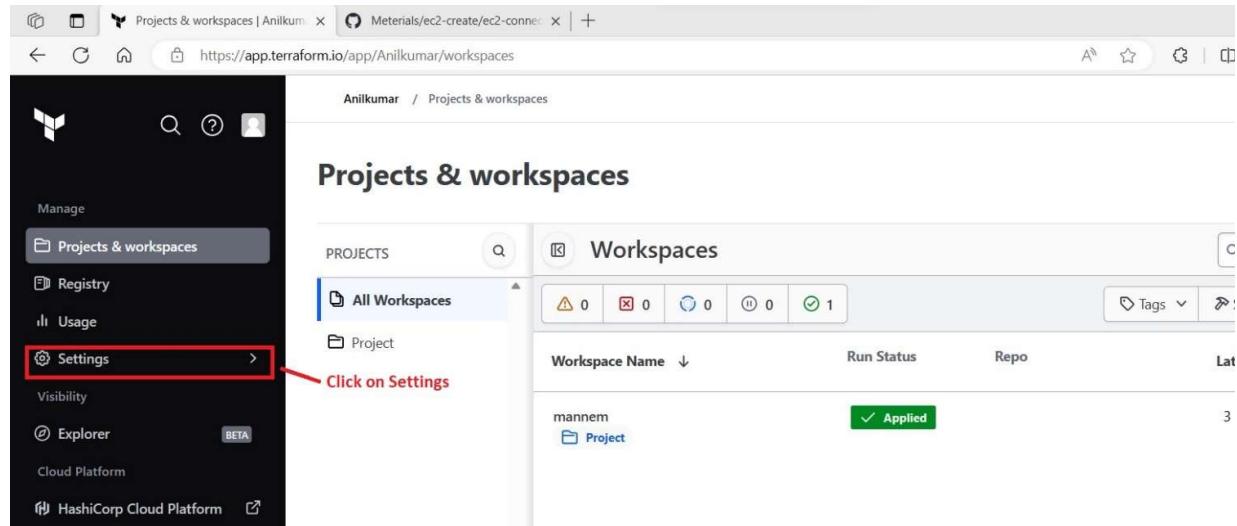
A callout box highlights a copy icon next to the configuration code example, with the text: "Click on this button and copy the code and paste it to main.tf file".

```
provider "aws" {  
    region = "us-west-2"  
}  
  
terraform {  
    cloud {  
        organization = "Anilkumar" # Code copied from terraform registry.  
        workspaces {  
            name = "mannem"  
        }  
    }  
}  
  
resource "aws_instance" "web" {  
    ami   = "ami-08541bb85074a743a"  
    instance_type = "t2.micro"  
    tags = {  
        Name = "My_First_Instance"  
    }  
}
```

## Main.tf (script)

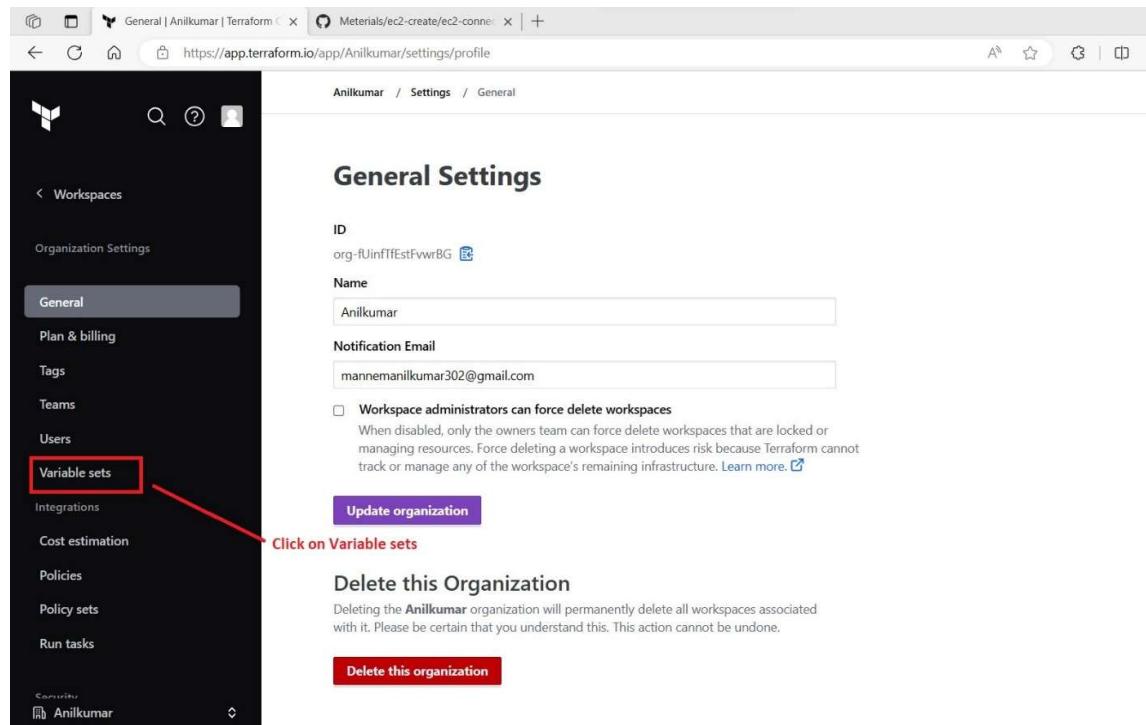
```
provider "aws" {  
    region = "us-west-2"  
}  
  
terraform {  
    cloud {  
        organization = "Anilkumar" # Code copied from terraform registry.  
        workspaces {  
            name = "mannem"  
        }  
    }  
}  
  
resource "aws_instance" "web" {  
    ami   = "ami-08541bb85074a743a"  
    instance_type = "t2.micro"  
    tags = {  
        Name = "My_First_Instance"  
    }  
}
```

After copying that code in main.tf file, we have to create the variable sets in terraform registry as follows.



The screenshot shows the Terraform Cloud interface. On the left, there's a sidebar with various options like 'Projects & workspaces', 'Registry', 'Usage', 'Settings' (which is highlighted with a red box and has a red arrow pointing to it with the label 'Click on Settings'), 'Visibility', 'Explorer' (BETA), 'Cloud Platform', and 'HashiCorp Cloud Platform'. The main area is titled 'Projects & workspaces' and shows a table for 'Workspaces'. One workspace named 'mannem' is listed with a status of 'Applied' and a green checkmark icon.

After clicking on settings, the following page will appear.



The screenshot shows the 'General Settings' page. The sidebar on the left has a 'General' tab selected, which is highlighted with a red box and has a red arrow pointing to it with the label 'Click on Variable sets'. Other tabs include 'Plan & billing', 'Tags', 'Teams', 'Users', 'Integrations', 'Cost estimation', 'Policies', 'Policy sets', 'Run tasks', and 'SonarQube'. The main content area is titled 'General Settings' and contains fields for 'ID' (org-fUinfTfEstFvwrBG), 'Name' (Anilkumar), 'Notification Email' (mannemanilkumar302@gmail.com), and a checkbox for 'Workspace administrators can force delete workspaces'. Below these is a purple 'Update organization' button. At the bottom, there's a section titled 'Delete this Organization' with a red 'Delete this organization' button.

After clicking on variable sets, the following page will appear.

The screenshot shows the Terraform Cloud interface. On the left, a dark sidebar menu includes 'Workspaces', 'Organization Settings', 'General', 'Plan & billing', 'Tags', 'Teams', 'Users', 'Variable sets' (which is selected and highlighted in grey), 'Integrations', 'Cost estimation', 'Policies', 'Policy sets', and 'Run tasks'. At the bottom of the sidebar is a 'Security' section with a lock icon and the name 'Anilkumar'. The main content area is titled 'Variable sets'. It contains a brief description of what variable sets are, a note about creating a variable set for variables used in more than one workspace, and a section on 'Variable conflicts and precedence'. A search bar labeled 'Search by variable set name' is present. A large central message states 'There are no variable sets in this organization.' Below this message is a note explaining that variable sets allow defining variables once across multiple workspaces. A prominent blue button labeled 'Create variable set' is located at the top right of the main content area. A callout arrow points from the text 'Click on this' to the 'Create variable set' button.

After clicking on create variable set, the following page will appear.

The screenshot shows the 'Create a new variable set' page. The left sidebar is identical to the previous screenshot. The main title is 'Create a new variable set'. A descriptive note says 'Variable sets allow you to define and apply variables one time across multiple workspaces within an organization.' Below this is a 'Configure settings' section with fields for 'Name' (containing 'anikumar') and 'Description (Optional)'. The next section is 'Variable set scope' with two options: 'Apply globally' (selected, with a note that all current and future workspaces in the organization will access this variable set) and 'Apply to specific projects and workspaces'. The final section is 'Variables', which contains a note about adding Terraform and Environment variables. A scroll bar on the right side of the page is highlighted with a callout arrow labeled 'Scroll down the page'.

After scrolling down the page, the following options will be visible.

## Variables

You can add any number of Terraform [variables](#) and Environment [variables](#). Terraform will use these variables for all plan and apply operations in the specified workspaces.

Key	Value	Category
There are no variables added.		
<div style="border: 2px solid black; padding: 5px; width: fit-content;">+ Add variable</div> <span style="margin-left: 10px;">Click on this button</span>		
<a href="#">Create variable set</a>		

After click on that button ‘add variable’, the following page will appear.

## Variables

You can add any number of Terraform [variables](#) and Environment [variables](#). Terraform will use these variables for all plan and apply operations in the specified workspaces.

Key	Value	Category
There are no variables added.		
<p>Select variable category</p> <p><input type="radio"/> Terraform variable These variables should match the declarations in your configuration. Click the HCL box to use interpolation or set a non-string value.</p> <p><input checked="" type="radio"/> Environment variable These variables are available in the Terraform runtime environment.</p>		
Key	Value	<input checked="" type="checkbox"/> Sensitive ⓘ
AWS_ACCESS_KEY	AKIAJDQZY57BV <del>XXXXXXXXXX</del>	
<p>Description (Optional)</p> <p>description (optional)</p> <p>Paste your ACCESS KEY</p>		
<div style="border: 2px solid black; padding: 5px; width: fit-content;">Add variable</div> <span style="margin-left: 10px;">Finally click on Add variable</span>		
<a href="#">Create variable set</a>		

After clicking on ‘Add variable’, the following page will appear.

## Variables

You can add any number of Terraform  and Environment  variables. Terraform will use these variables for all plan and apply operations in the specified workspaces.

Key	Value	Category	...
AWS_ACCESS_KEY SENSITIVE	Sensitive - write only	env	...

Select variable category

Terraform variable  
These variables should match the declarations in your configuration. Click the HCL box to use interpolation or set a non-string value.

Environment variable  
These variables are available in the Terraform runtime environment.

Key Value Category

AWS\_SECRET\_KEY TzMVkkJs3covFgegpkhsQMRGN6qaTRb7UmBKUK0f env Sensitive 

Description (Optional)  
description (optional)

Add variable Cancel

Again, we have click on ‘Add variable’, the following page will appear.

## Variable set scope

### Apply globally

All current and future workspaces in this organization will access this variable set.

### Apply to specific projects and workspaces

## Variables

You can add any number of Terraform  and Environment  variables. Terraform will use these variables for all plan and apply operations in the specified workspaces.

Key	Value	Category	...
AWS_ACCESS_KEY SENSITIVE	Sensitive - write only	env	...
AWS_SECRET_KEY SENSITIVE	Sensitive - write only	env	...

+ Add variable

We had added the keys

Create variable set

After clicking on ‘Create variable set’ button, the following page will be displayed.

## Variable sets

Create variable set

Terraform uses [variables](#) for all plans and applies within a workspace. [Variable sets](#) are a group of commonly used variables that you can apply to multiple workspaces in an organization.

We recommend creating a variable set for variables used in more than one workspace.

### Variable conflicts and precedence

Conflicts occur when one or more variables applied to a workspace have the same type and the same key. Workspace-specific variables always overwrite conflicting variables from variable sets. When different variable sets contain conflicts, Terraform Cloud prioritizes them first based on the variable set scope and then by the lexical precedence of the variable set name. Learn more about variable precedence [↗](#).

The screenshot shows the Terraform Cloud interface for managing variable sets. A search bar at the top contains the placeholder 'Search by variable set name'. Below it, a list of variable sets is shown, with 'anilkumar' highlighted in a black box. To the right of 'anilkumar', the text 'Finally we had created the variable set' is displayed. Below the list, it says 'All workspaces · 2 variables' and 'Last updated July 31st 2023, 2:42:33 pm'. At the bottom, there are navigation controls for pages 1-1 of 1, an 'Items per page' dropdown set to 50, and a refresh icon.

Then after, First run the command ‘**terraform login**’ in VS code terminal.

The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal shows the command 'PS C:\Users\manne\Desktop> terraform login' being run. Below the command, a message says 'Terraform will request an API token for app.terraform.io using your browser.' A prompt 'Do you want to proceed?' with the note 'Only 'yes'' will be accepted to confirm.' is followed by an input field containing 'Enter a value: yes'. A callout box highlights this input field with the text 'Yes to be entered'.

After entering yes, click enter. The following screen will appear and in parallel automatically web browser will be opened. Copy that generated token and paste it.

The screenshot shows the terminal output of the 'terraform login' command. It includes instructions to open a browser to tokens.app.terraform.io if it doesn't open automatically. It also shows the generation of a token and its storage location: 'C:\Users\manne\AppData\Roaming\terraform.d\credentials.tfrc.json'. A callout box highlights the input field 'Enter a value: [ ]' with the text 'After pasting the value also its seems empty, just press Enter key.'

After entering yes, for token generation the following will appear.

A screenshot of a web browser showing the Terraform Cloud interface. The URL is https://app.terraform.io/app/settings/tokens?source=terraform-login. On the left, there's a sidebar with options like User Settings, Profile, Sessions, Organizations, Password, Two Factor Authentication, SSO, and Tokens (which is selected). The main area is titled 'Tokens' and contains a sub-section 'Creating a user token'. It has fields for 'Description' (set to 'terraform login') and 'Expiration' (set to '30 days'). A button labeled 'Generate token' is highlighted with a red box. Below the dialog, the tokens list shows one entry: 'terraform login' created 30 days ago.

After clicking the button ‘Generate token’, the following page will appear.

A screenshot of the same Terraform Cloud interface after generating a token. The 'Tokens' list now shows two entries: 'terraform login' and a new token named 'C16aPplm0dzyKA.atlasv1.V9wm36m7LdnxJyj81ngzRY5XdtYiNXPBzyDENxnQzhnWmkmiWQtG1yeBUTlmx9e58'. The second token has a 'Copy' icon and a note: 'Click on this to copy token'.

After pasting the token, click enter. The following screen will appear.

A screenshot of the Terraform Cloud terminal. The URL is https://app.terraform.io/. The terminal shows the command 'Token for app.terraform.io:' followed by 'Enter a value:'. The user has entered 'mannem302'. The output shows 'Retrieved token for user mannam302' and then displays the retrieved token as a long string of characters. Below the terminal, a message says 'Welcome to Terraform Cloud!' and 'Documentation: terraform.io/docs/cloud'.

After successful login, run the command ‘**terraform init**’ in the terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2> terraform init
Initializing Terraform Cloud...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.10.0

Terraform Cloud has been successfully initialized!

You may now begin working with Terraform Cloud. Try running "terraform plan" to
see any changes that are required for your infrastructure.

If you ever set or change modules or Terraform Settings, run "terraform init"
again to reinitialize your working directory.
```

After successful initialization, run the command ‘**terraform plan**’ in the terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2> terraform plan
Running plan in Terraform Cloud. Output will stream here. Pressing Ctrl-C
will stop streaming the logs, but will not stop the plan running remotely.

Preparing the remote plan...

To view this run in a browser, visit:
https://app.terraform.io/app/Anilkumar/mannem/runs/run-R6Zna4Dedppo6Q3t

Waiting for the plan to start...

Terraform v1.5.4
on linux_amd64
Initializing plugins and modules...

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.web will be created
+ resource "aws_instance" "web" {
    + ami                               = "ami-08541bb85074a743a"
    + arn                             = (known after apply)
    + associate_public_ip_address      = (known after apply)

    }
    + tenancy                         = (known after apply)
    + user_data                       = (known after apply)
    + user_data_base64                = (known after apply)
    + user_data_replace_on_change     = false
    + vpc_security_group_ids          = (known after apply)
}

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

After that, you have to run the command ‘**terraform apply –auto-approve=true**’.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2> terraform apply --auto-approve=true
Running apply in Terraform Cloud. Output will stream here. Pressing Ctrl-C
will cancel the remote apply if it's still pending. If the apply started it
will stop streaming the logs, but will not stop the apply running remotely.

Preparing the remote apply...

To view this run in a browser, visit:
https://app.terraform.io/app/Anilkumar/mannem/runs/run-vAkzoGaPXMkdZIUL

Waiting for the manually locked workspace to be unlocked...

Terraform v1.5.4
on linux_amd64
Initializing plugins and modules...

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

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

aws_instance.web: Creating...
aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Still creating... [20s elapsed]
aws_instance.web: Still creating... [30s elapsed]
aws_instance.web: Creation complete after 32s [id=i-07f06d10cb947255f]
```

Instance created and tfstate file was  
uploaded to terraform registry.

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

To terminate the instance, run the command ‘**terraform destroy –auto-approve=true**’.

```
aws_instance.web: Destroying... [id=i-0eed04333b1ba3c88]
aws_instance.web: Still destroying... [10s elapsed]
aws_instance.web: Still destroying... [20s elapsed]
aws_instance.web: Still destroying... [30s elapsed]
aws_instance.web: Still destroying... [40s elapsed]
aws_instance.web: Destruction complete after 40s
```

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

For confirmation purpose, we can view in terraform registry. As shown below.

The screenshot shows a browser window with multiple tabs open, including 'Terraform EC2 Setup', 'Build Infrastructure', 'Overview | manne', 'hashicorp/tfc-getting-started', 'run-j7EKxURXG2vh9ej8', and 'Docs overview | hashicorp/terraform'. The main content is a workspace named 'mannem'. The sidebar on the left has options like 'Workspaces', 'mannem', 'Overview', 'Runs' (which is selected), 'States', 'Variables', and 'Settings'. The 'Runs' section shows a 'Triggered via CLI' run. The run details show it was triggered by 'mannem302' from the CLI 5 minutes ago. It includes fields for Run ID (run-j7EKxURXG2vh9ej8), Configuration (From CLI), Trigger (Run manually triggered), and Execution Mode (Remote). Below the run details, there are two status cards: 'Plan finished' 5 minutes ago (Resources: 0 to add, 0 to change, 1 to destroy) and 'Apply finished' 5 minutes ago (Resources: 0 added, 0 changed, 1 destroyed).

## 11) Terraform backend configuration, to store terraform.tfstate file?

A) To store the terraform.tfstate file in S3 bucket, use the following script.

### Main.tf (Script)

```
provider "aws" {
  region = "us-west-2"
}

terraform {
  backend "s3" {
    bucket      = "anilkumar9999" # In S3 already, this bucket was available.
    key         = "C:/Users/manne/OneDrive/Desktop/terraform/Ec2/terraform.tfstate"
    region     = "us-west-2"
  }
}

resource "aws_instance" "web" {
  ami      = "ami-08541bb85074a743a"
  instance_type = "t2.micro"
  tags = {
    Name = "My_First_Instance"
  }
}
```

After writing the main.tf script, save the file and execute the below commands.

First run the command, ‘**terraform init**’ in the terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2 s3 state file> terraform init
Initializing the backend...
Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
```

After successful initialization, run the command, ‘**terraform plan**’ in the terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2 s3 state file> 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.web will be created
+ resource "aws_instance" "web" {
    + ami                               = "ami-08541bb85074a743a"
    + arn                               = (known after apply)
    + associate_public_ip_address       = (known after apply)

    + tenancy                           = (known after apply)
    + user_data                         = (known after apply)
    + user_data_base64                  = (known after apply)
    + user_data_replace_on_change      = false
    + vpc_security_group_ids           = (known after apply)
}

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

After that run the command, ‘**terraform apply --auto-approve=true**’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2 s3 state file> terraform apply --auto-approve=true
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.web will be created
+ resource "aws_instance" "web" {
    + ami                               = "ami-08541bb85074a743a"
    + arn                               = (known after apply)
    + associate_public_ip_address       = (known after apply)
    + availability_zone                 = (known after apply)

    + tags_all                          = {
        + "Name" = "My_First_Instance"
    }
    + tenancy                           = (known after apply)
    + user_data                         = (known after apply)
    + user_data_base64                  = (known after apply)
    + user_data_replace_on_change      = false
    + vpc_security_group_ids           = (known after apply)
}

aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Still creating... [20s elapsed]
aws_instance.web: Creation complete after 26s [id=i-080f4204420f50c54]

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

After that run the command, ‘**terraform destroy --auto-approve=true**’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2 s3 state file> terraform destroy --auto-approve=true
aws_instance.web: Refreshing state... [id=i-080f4204420f50c54]
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.web will be destroyed
- resource "aws_instance" "web" {
    - ami                               = "ami-08541bb85074a743a" -> null
    - arn                               = "arn:aws:ec2:us-west-2:488606592963:instance/i-080f4204420f50c54" -> null
```

```

        - throughput
        - volume_id
        - volume_size
        - volume_type
    }
}

Plan: 0 to add, 0 to change, 1 to destroy.
aws_instance.web: Destroying... [id=i-080f4204420f50c54]
aws_instance.web: Still destroying... [id=i-080f4204420f50c54, 10s elapsed]
aws_instance.web: Still destroying... [id=i-080f4204420f50c54, 20s elapsed]
aws_instance.web: Still destroying... [id=i-080f4204420f50c54, 30s elapsed]
aws_instance.web: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.

```

For confirmation, view the S3 bucket in AWS console.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with 'Amazon S3' selected. The main area shows a breadcrumb navigation path: Amazon S3 > Buckets > anilkumar9999 > Ec2/ > C/ > Users/ > manne/ > OneDrive/ > Desktop/ > terraform/ > Ec2/. Below the path, there are tabs for 'Objects' and 'Properties'. Under the 'Objects' tab, it says 'Objects (1)'. There is one object listed: 'terraform.tfstate'. The table below shows the details for this object:

Name	Type	Last modified	Size	Storage class
terraform.tfstate	tfstate	July 31, 2023, 00:06:07 (UTC+05:30)	4.7 KB	Standard

**12)** In terraform, what is terraform.tfstate file and its importance?

A) In Terraform, the state file (often referred to as terraform.tfstate) is a crucial component of the infrastructure management process. It is an automatically generated file that keeps track of the resources created and managed by Terraform. The state file records the current state of your infrastructure as defined in your Terraform configuration files. The terraform.tfstate file is a JSON-formatted file that contains detailed information about the resources Terraform manages, such as resource IDs, metadata, attributes, dependencies, and more. It acts as the source of truth for Terraform to understand the current state of your infrastructure. This file can be saved in terraform registry and as well as in S3 bucket for common state maintaining purpose in an organization.

**13)** Terraform variables creation (in different types)?

A) To declare variables in terraform, we need to create main.tf, variables.tf, values.tfvars and outputs.tf

## Values.tfvars (script)

```
aws_region = "us-west-2"
cidr_block = ["192.168.0.0/16","192.168.0.0/24", "192.168.1.0/24"]
vpc_name = "demovpc"
subnet_name = ["sn1","sn2"]
```

## Outputs.tf (script)

```
output "vpc_name" {
  value = var.vpc_name
}
output "subnets" {
  value = var.subnet_name
}
```

Main.tf(Script)	Variables.tf(script)
<pre>provider "aws" {   region = var.aws_region }  terraform {   cloud {     organization = "kellydevopsdemo"     workspaces {       name = "test"     }   } }  resource "aws_vpc" "main" {   cidr_block = var.cidr_block[0]   tags = {     Name = var.vpc_name   } }  resource "aws_subnet" "subnet1" {   vpc_id = aws_vpc.main.id   cidr_block = var.cidr_block[1]   tags = {     Name = var.subnet_name[0]   } }  resource "aws_subnet" "subnet2" {   vpc_id = aws_vpc.main.id   cidr_block = var.cidr_block[2]   tags = {     Name = var.subnet_name[1]   } }</pre>	<pre>variable "aws_region" {   type = string   #default = "us-west-2" }  variable "vpc_name" {   type = string   #default = "testvpc" }  variable "cidr_block" {   type = list   #default =   ["192.168.0.0/16","192.168.0.0/24", "192.168.1.0/24"] }  variable "subnet_name" {   type = list   #default = ["sn1","sn2"] }</pre>

After writing the scripts and saving the files, we have to execute the following commands '**terraform init**' in terminal. (screen shows in next page will be displayed)

```
PS C:\Users\manne\OneDrive\Desktop\terraform\variables execution> terraform init
Initializing Terraform Cloud...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.10.0

Terraform Cloud has been successfully initialized!

You may now begin working with Terraform Cloud. Try running "terraform plan" to
see any changes that are required for your infrastructure.

If you ever set or change modules or Terraform Settings, run "terraform init"
again to reinitialize your working directory.
```

After that run the command, ‘**terraform plan**’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\variables execution> terraform plan
Running plan in Terraform Cloud. Output will stream here. Pressing Ctrl-C
will stop streaming the logs, but will not stop the plan running remotely.

Preparing the remote plan...

To view this run in a browser, visit:
https://app.terraform.io/app/Anilkumar/mannem/runs/run-NNBswBf61hx9sCJk

Waiting for the plan to start...

Terraform v1.5.4
on linux_amd64
Initializing plugins and modules...

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_subnet.subnet1 will be created
+ resource "aws_subnet" "subnet1" {
    + arn
    = (known after apply)
```

After that run the command, ‘**terraform plan --auto-approve=true**’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\Ec2 s3 state file> terraform apply --auto-approve=true

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.web will be created
+ resource "aws_instance" "web" {
    + ami
    = "ami-08541bb85074a743a"
    + arn
    = (known after apply)
    + associate_public_ip_address
    = (known after apply)
    + availability_zone
    = (known after apply)
    + tags.all
        + "Name" = "My_First_Instance"
    }
    + tenancy
    = (known after apply)
    + user_data
    = (known after apply)
    + user_data_base64
    = (known after apply)
    + user_data_replace_on_change
    = false
    + vpc_security_group_ids
    = (known after apply)
}

aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Still creating... [20s elapsed]
aws_instance.web: Creation complete after 26s [id=i-080f4204420f50c54]

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

After that run the command, ‘**terraform destroy --auto-approve=true**’ in terminal.

```
PS C:\Users\manne\OneDrive\Desktop\terraform\variables execution> terraform destroy --auto-approve=true
Running apply in Terraform Cloud. Output will stream here. Pressing Ctrl-C
will cancel the remote apply if it's still pending. If the apply started it
will stop streaming the logs, but will not stop the apply running remotely.

Preparing the remote apply...

To view this run in a browser, visit:
https://app.terraform.io/app/Anilkumar/mannem/runs/run-r46NCCfPCPzYtuoD

Waiting for the plan to start...

Terraform v1.5.4
on linux_amd64
Initializing plugins and modules...
aws_vpc.main: Refreshing state... [id=vpc-00e9e458b5577b124]
aws_subnet.subnet1: Refreshing state... [id=subnet-0587e5bc96e2afee7]
aws_subnet.subnet2: Refreshing state... [id=subnet-06be1fbead7f1e96f]

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

```
Changes to Outputs:  
- subnets = [  
  - "sn1",  
  - "sn2",  
] -> null  
- vpc_name = "testvpc" -> null  
  
aws_subnet.subnet2: Destroying... [id=subnet-06be1fbead7f1e96f]  
aws_subnet.subnet1: Destroying... [id=subnet-0587e5bc96e2afee7]  
aws_subnet.subnet1: Destruction complete after 1s  
aws_subnet.subnet2: Destruction complete after 1s  
aws_vpc.main: Destroying... [id=vpc-00e9e458b5577b124]  
aws_vpc.main: Destruction complete after 1s  
  
Apply complete! Resources: 0 added, 0 changed, 3 destroyed.
```

Variables, we can declare directly or we can pass it as file or we can pass it as a command line argument.

Variable passing through file command is ‘**terraform apply --auto-approve=true -var-file="values.tfvars"** ‘

**Note:-** In variables.tf we have to comment the default values.

Variables passing through command line is ‘**terraform apply -var= ‘cidr\_block=[“IP”,”IP”]**  
**-var= ‘subnet\_name=[“sn1”,”sn2”]**’ -var= “aws\_region=us-west-2”