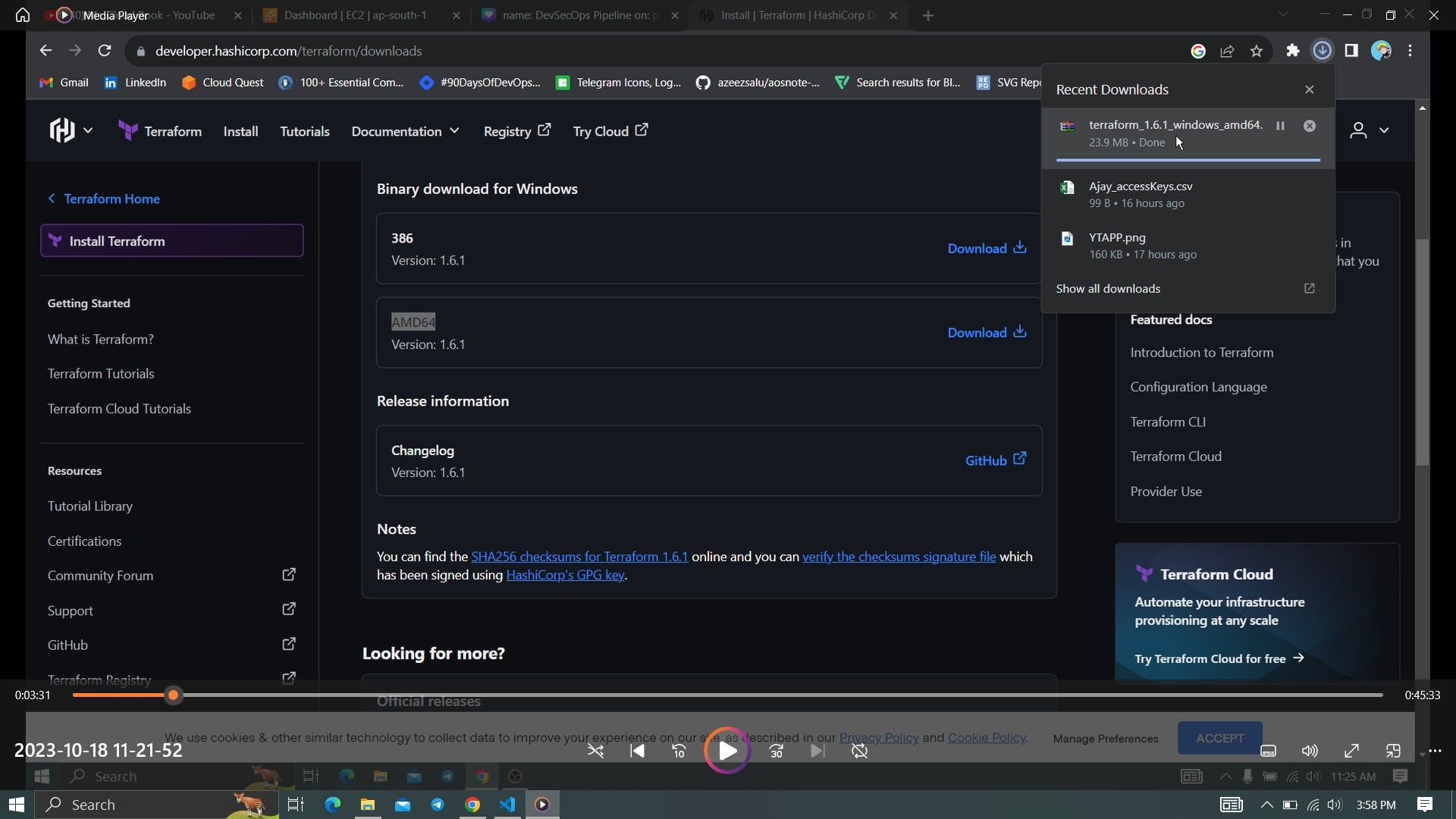
**Terraform: Provisioning an EC2 Instance with Jenkins, Running a SonarQube Container & Trivy**

In today's fast-paced world of software development, achieving automation and scalability in your infrastructure is vital. Combining the power of Infrastructure as Code (IaC) and cloud computing, we can streamline the provisioning of development environments, facilitate Continuous Integration and Continuous Deployment (CI/CD), and enforce high code quality standards. In this blog post, we embark on a journey to demonstrate how to leverage Terraform, an IaC tool, to provision an EC2 instance in the cloud, install Jenkins for continuous integration, and set up a SonarQube container for code quality monitoring. By automating this process, you not only reduce manual configuration but also enhance the reliability and efficiency of your development pipeline. Let's explore how Terraform can transform your infrastructure into a well-oiled machine, ready to take your software projects to new heights.

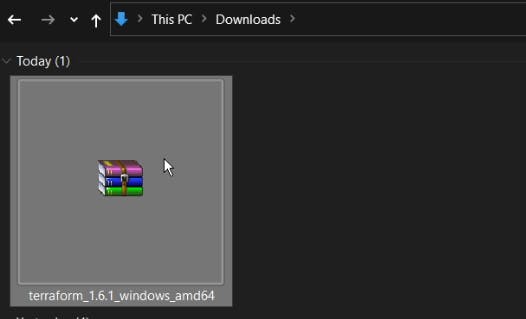
**Step1: How to install and setup Terraform on Windows**

Download Terraform:

Visit the official Terraform website: <https://www.terraform.io/downloads.html>

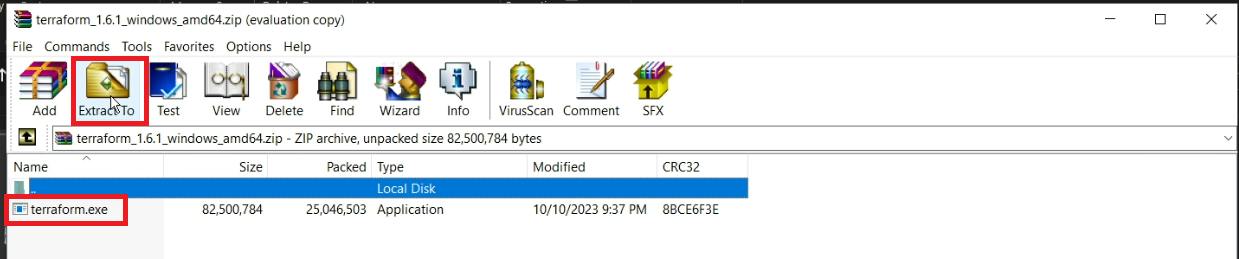


Select AMD64 FOR Windows Version

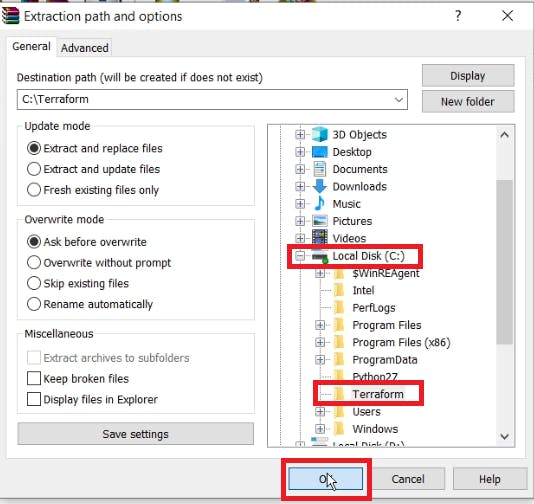


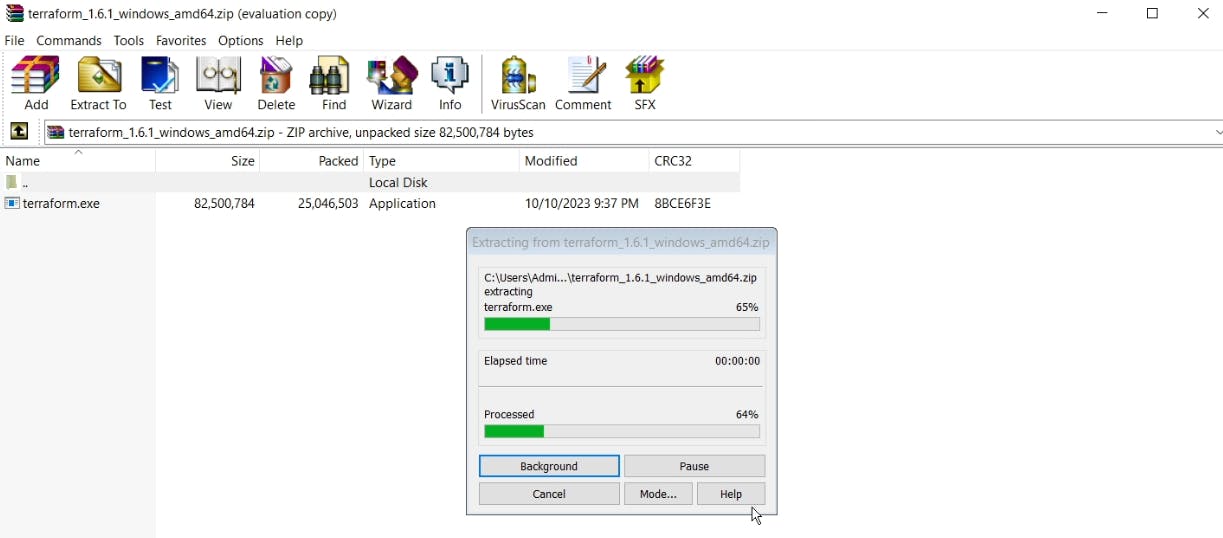
Extract the ZIP Archive:

Once the download is complete, extract the contents of the ZIP archive to a directory on your computer. You can use a tool like 7-Zip or the built-in Windows extraction tool. Ensure that you extract it to a directory that's part of your system's PATH.

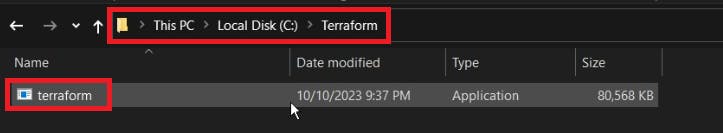


Remember that I created a Terraform Directory in C drive and there I have extracted the zip file of terraform

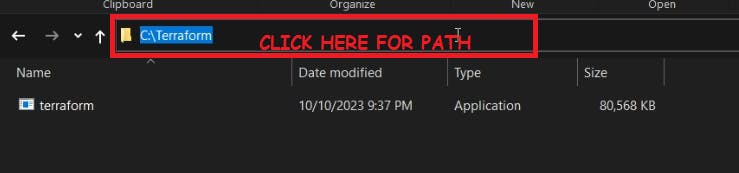




Extracted to C drive



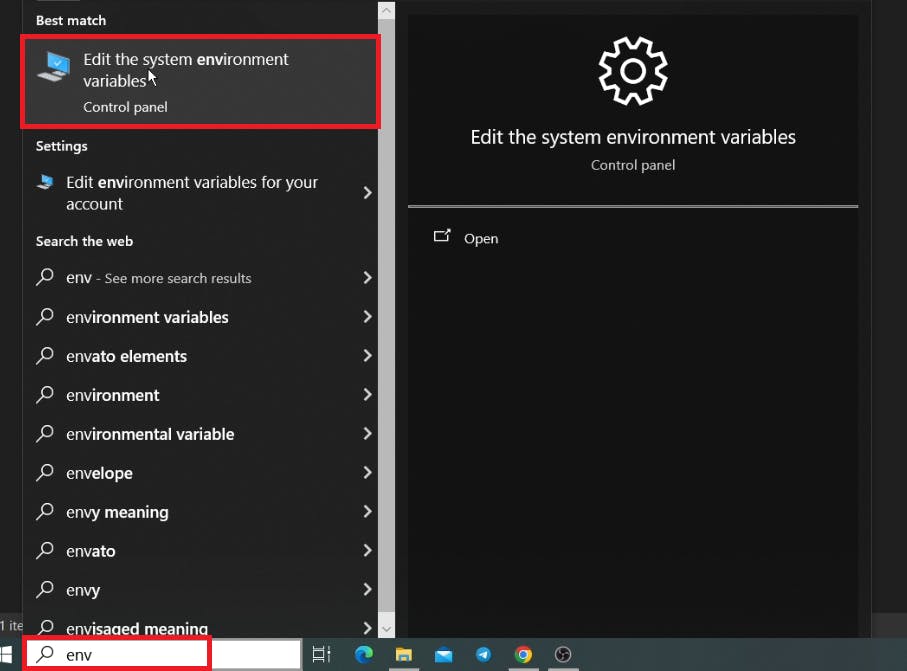
Copy the path



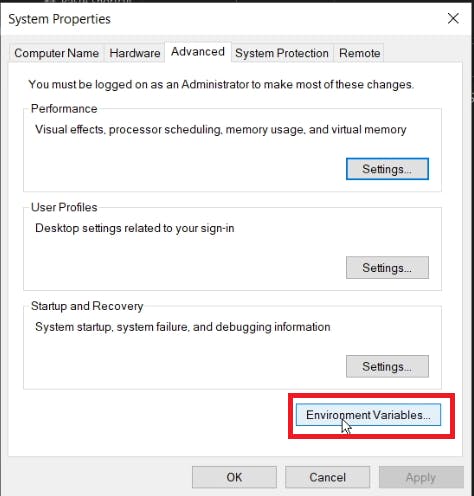
Add Terraform to Your System's PATH:

To make Terraform easily accessible from the command prompt, add the directory where Terraform is extracted to your system's PATH environment variable. Follow these steps:

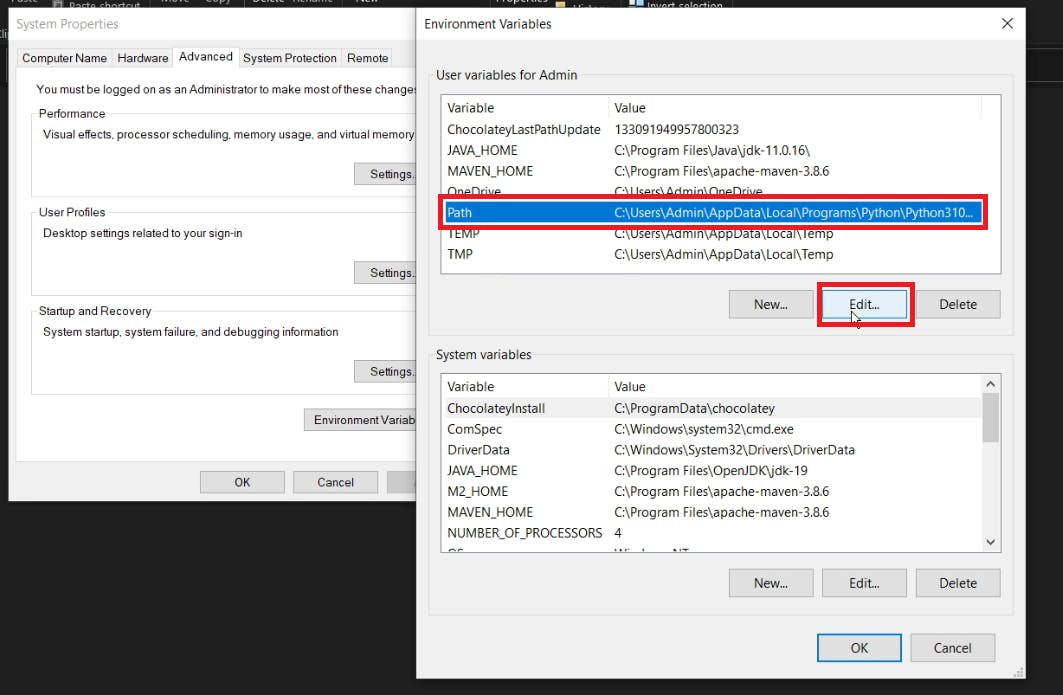
Search for "Environment Variables" in your Windows search bar and click "Edit the system environment variables."



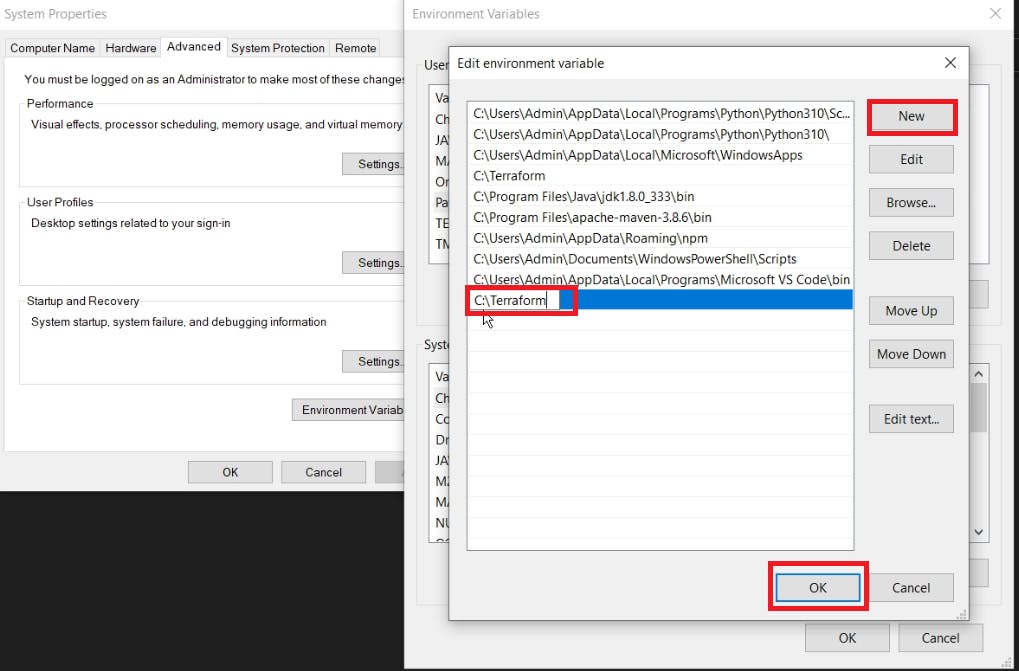
In the "System Properties" window, click the "Environment Variables" button.



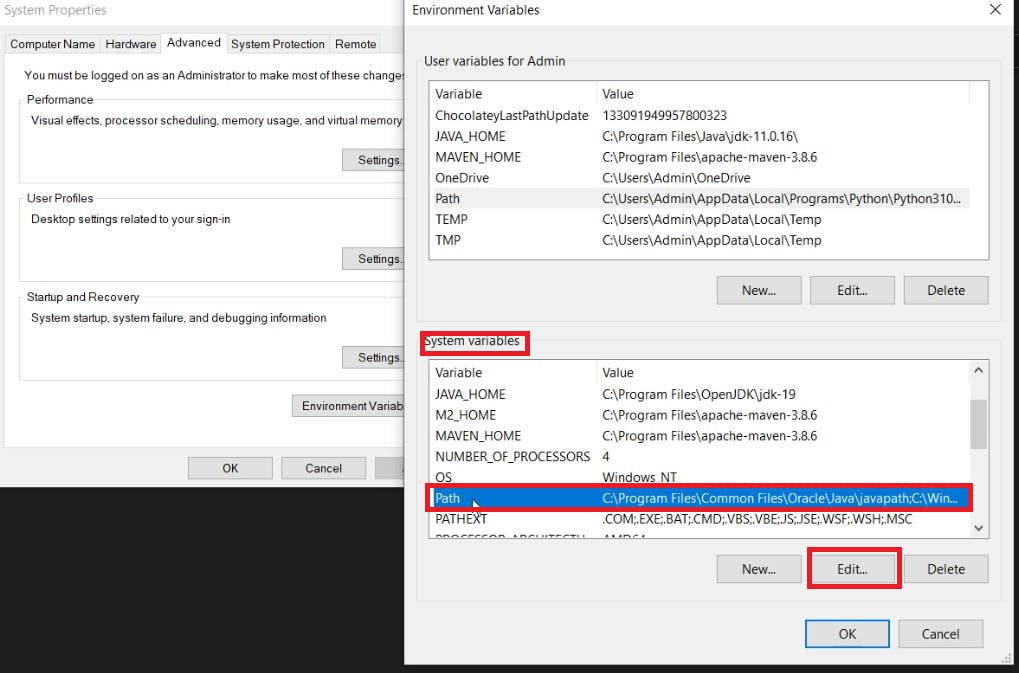
Under "User variables for Admin," find the "Path" variable and click "Edit."



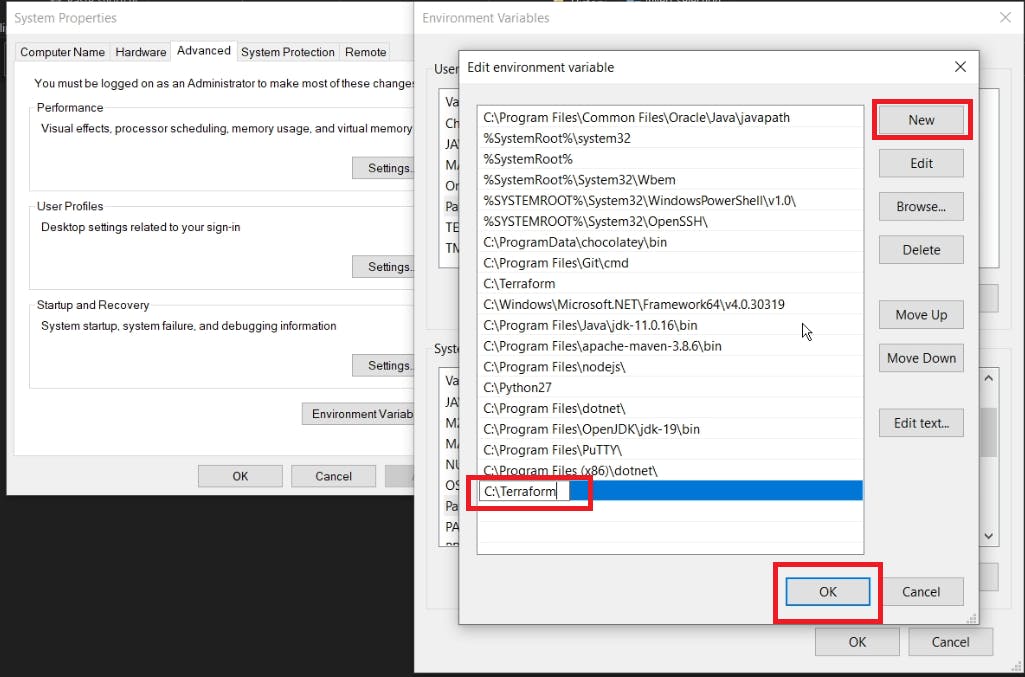
Click on New paste the copied path and click on OK



Under "System variables," find the "Path" variable and click "Edit."



Click "New" and add the path to the directory where you extracted Terraform (e.g., C:\path\to\terraform).



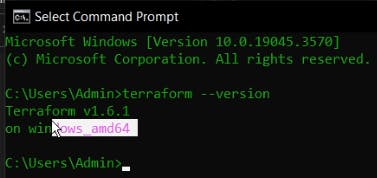
Click "OK" to close the Environment Variables windows.

Click "OK" again to close the System Properties window.

Verify the Installation:

Open a new Command Prompt or PowerShell window.

Type terraform --version and press Enter. This command should display the Terraform version, confirming that Terraform is installed and in your PATH.

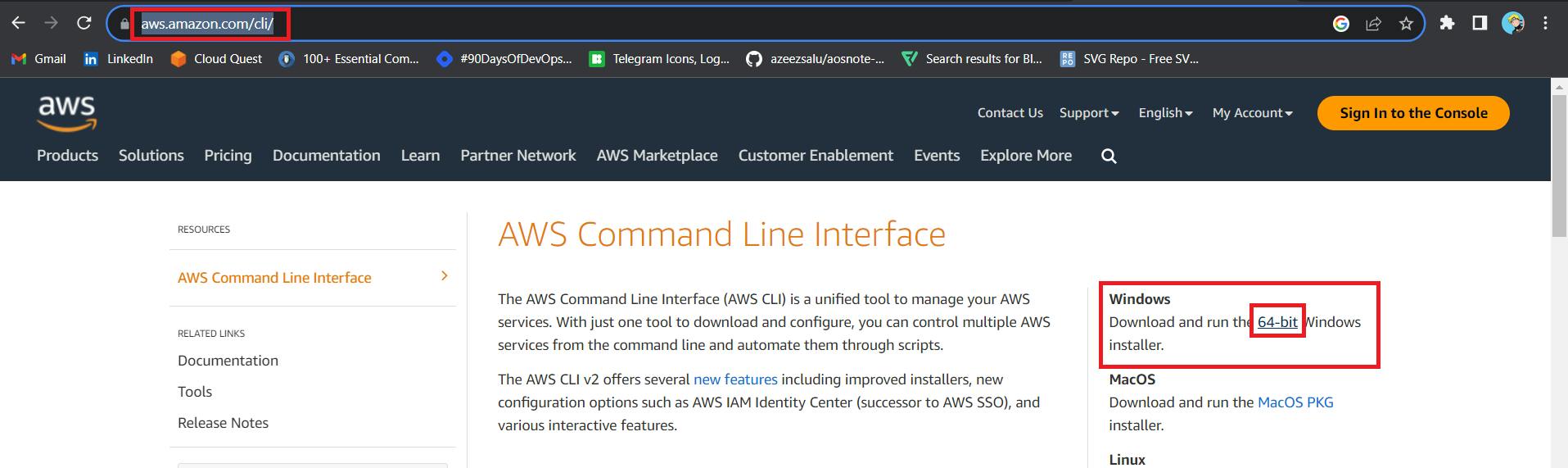


Your Terraform installation is now complete, and you can start using Terraform to manage your infrastructure as code.

Step2: Download the AWS CLI Installer:

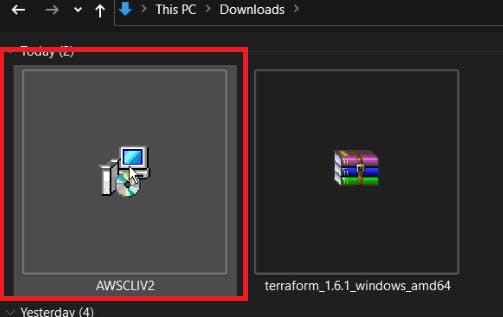
Visit the AWS CLI Downloads page: <https://aws.amazon.com/cli/>

Under "Install the AWS CLI," click on the "64-bit" link to download the AWS CLI installer for Windows.

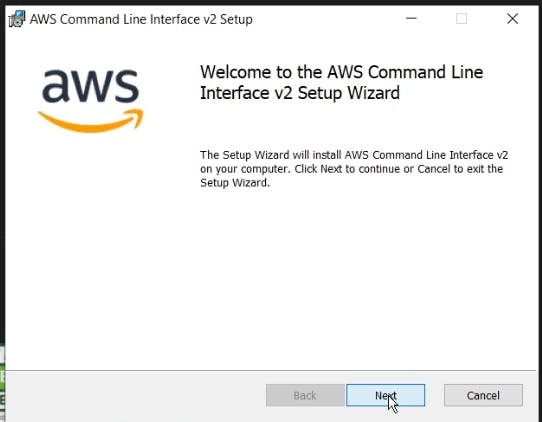


Run the Installer:

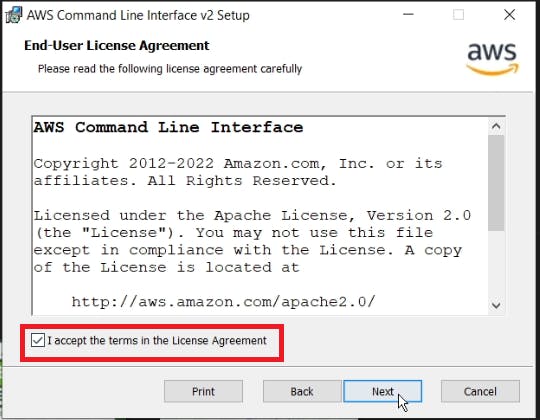
Locate the downloaded installer executable (e.g., AWSCLIV2.exe) and double-click it to run the installer.



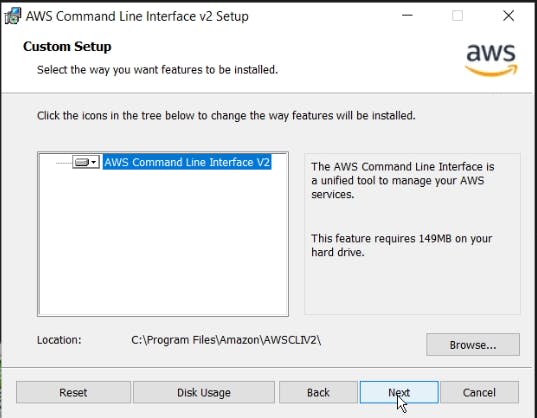
Click on Next



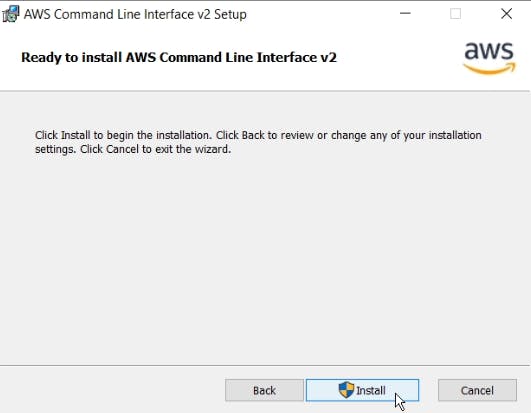
Agree to terms and click on Next



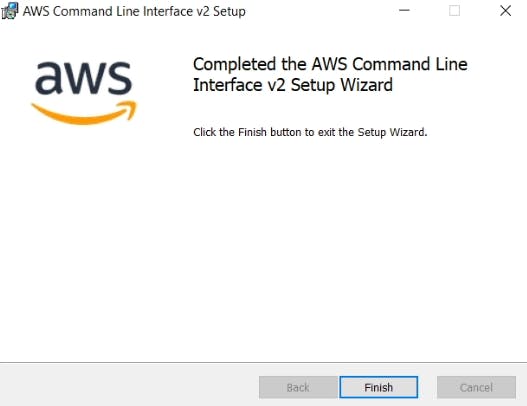
Click Next



Click on install



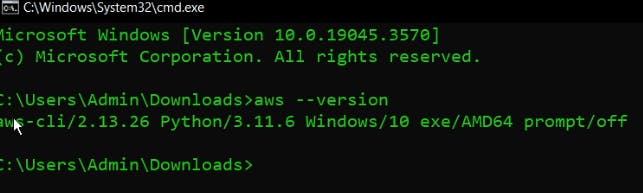
Click Finish Aws cli is installed



Verify the Installation:

Open a Command Prompt or PowerShell window.

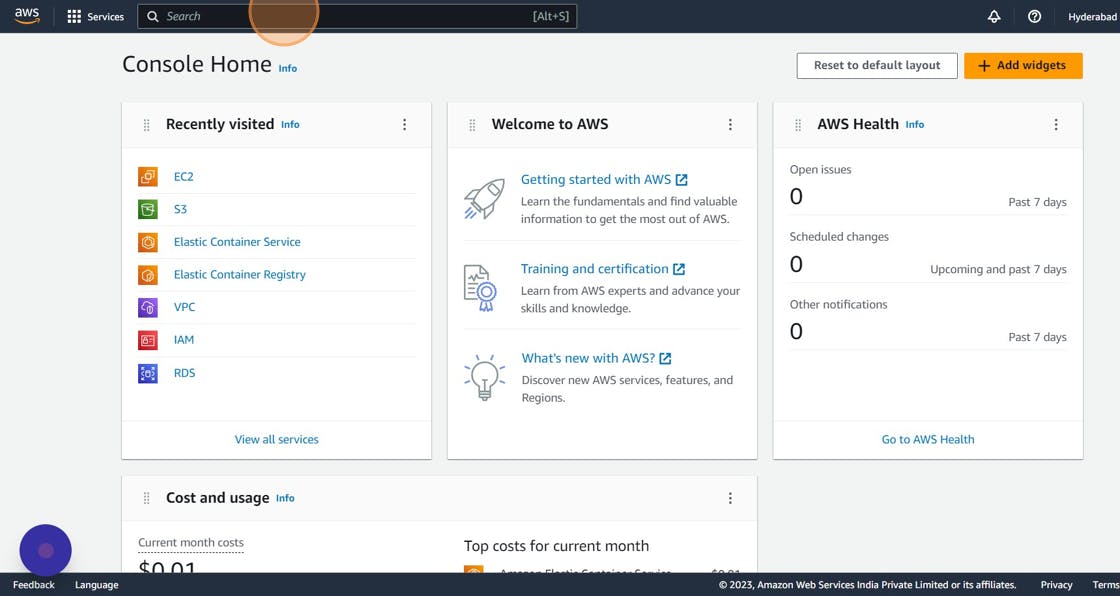
Type aws --version and press Enter. This command should display the AWS CLI version, confirming that the installation was successful.



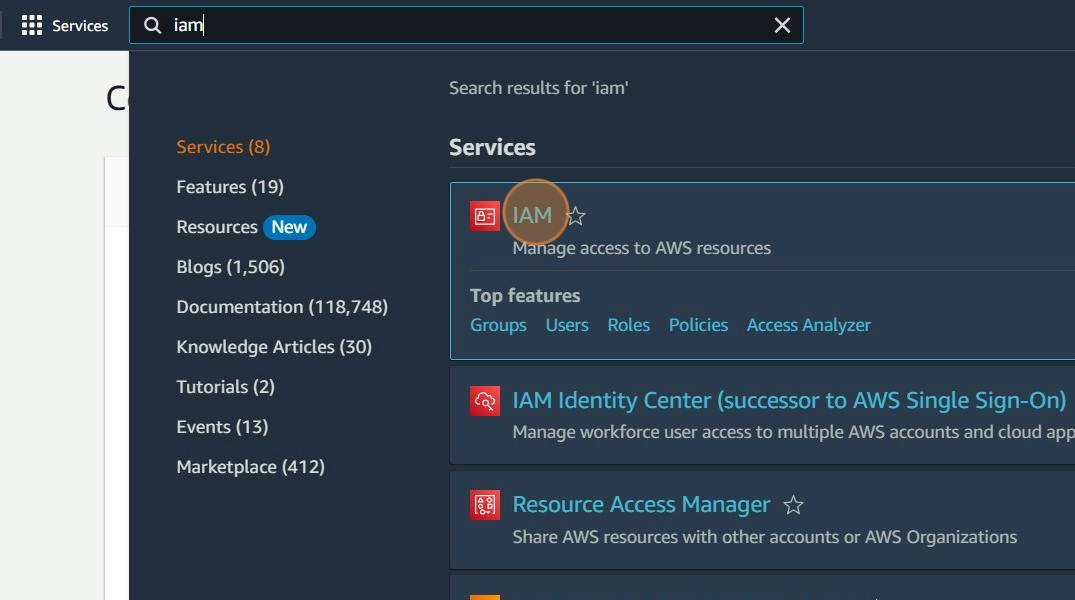
Step3: create an IAM user

Navigate to the AWS console

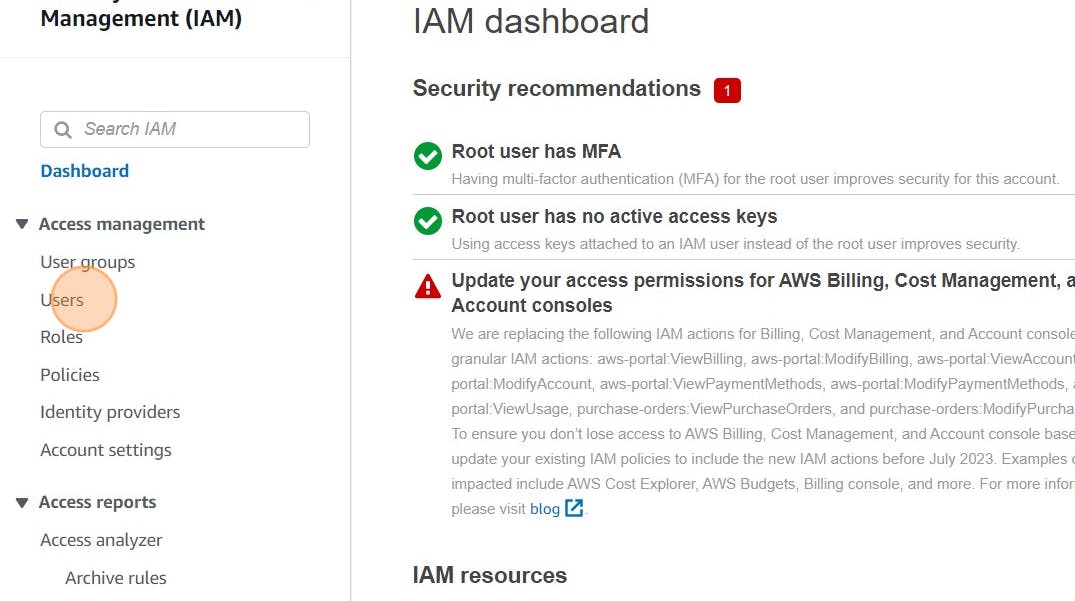
Click the "Search" field.



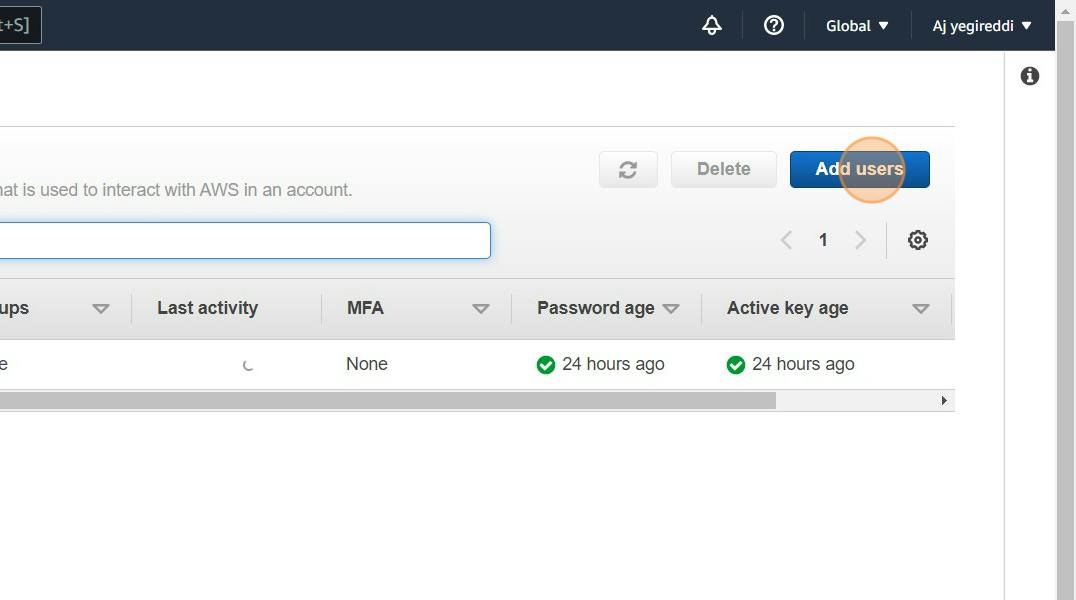
Search for IAM



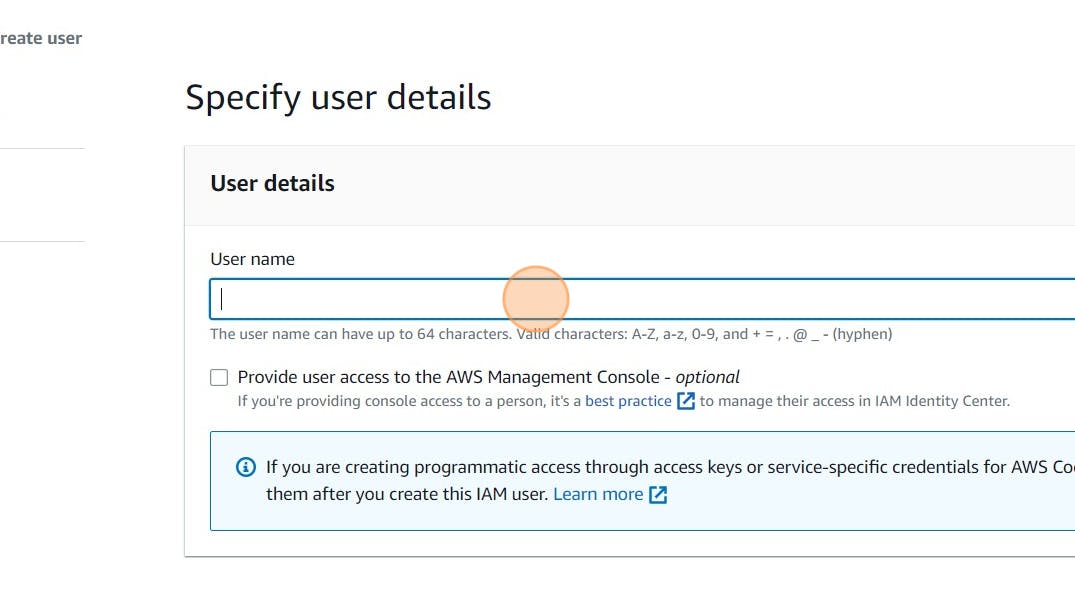
Click "Users"



Click "Add users"

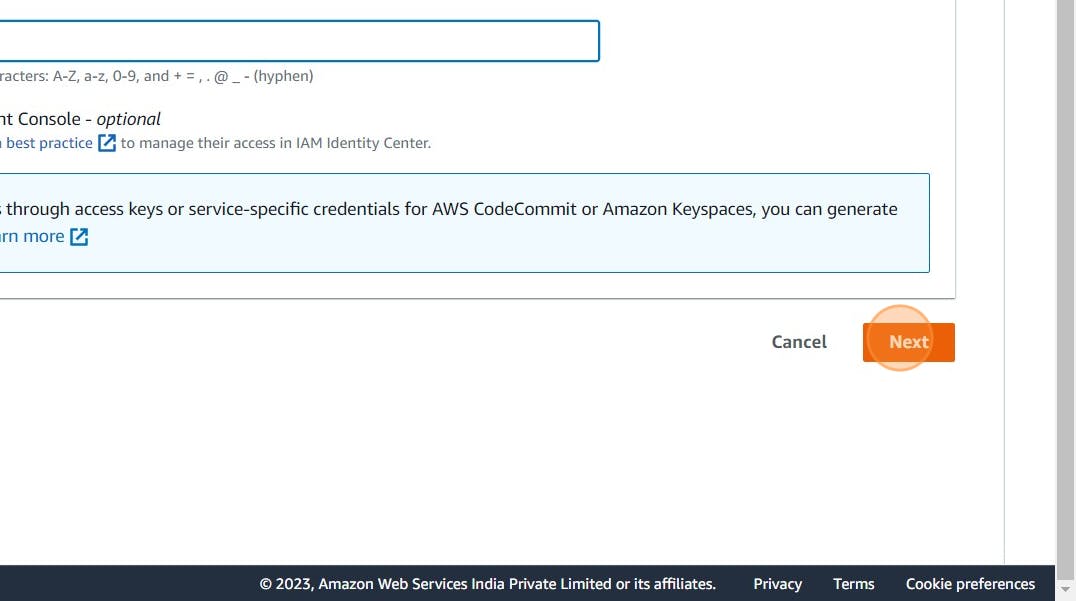


Click the "User name" field.

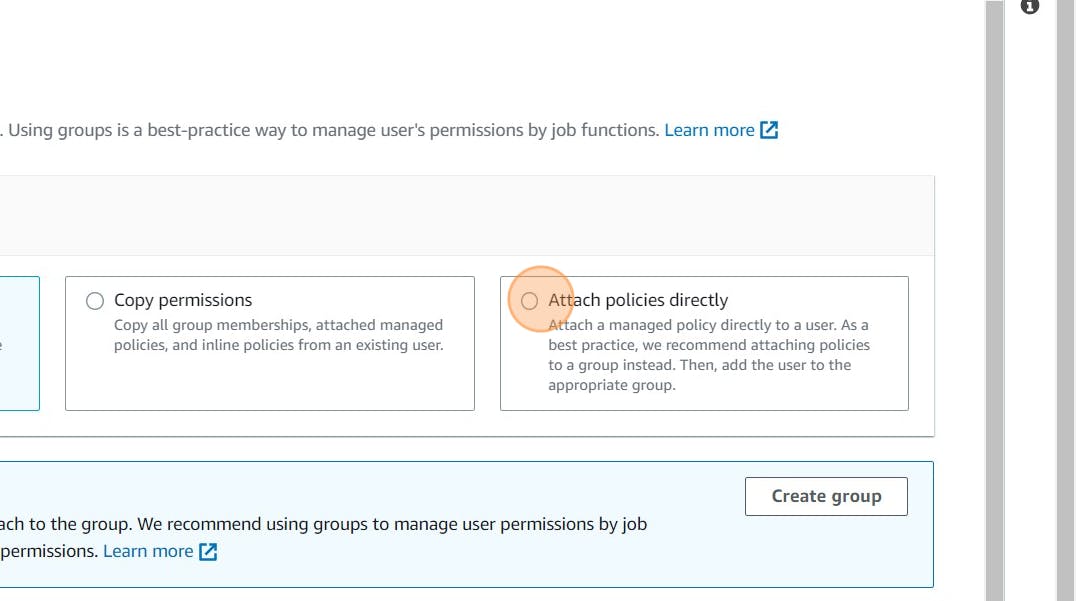


Type "Terraform" or as you wish about the name

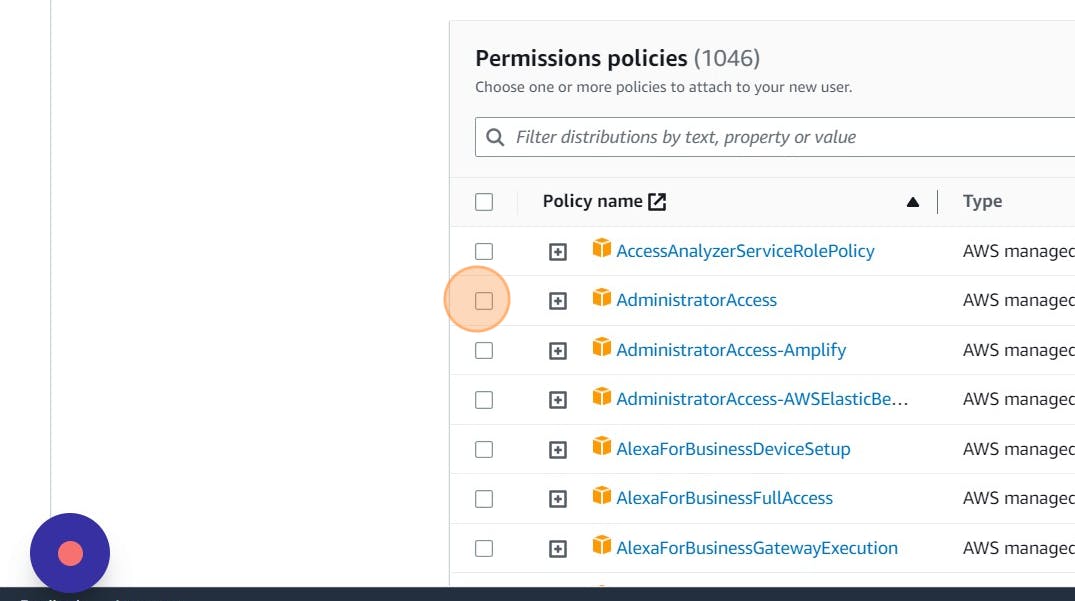
Click Next



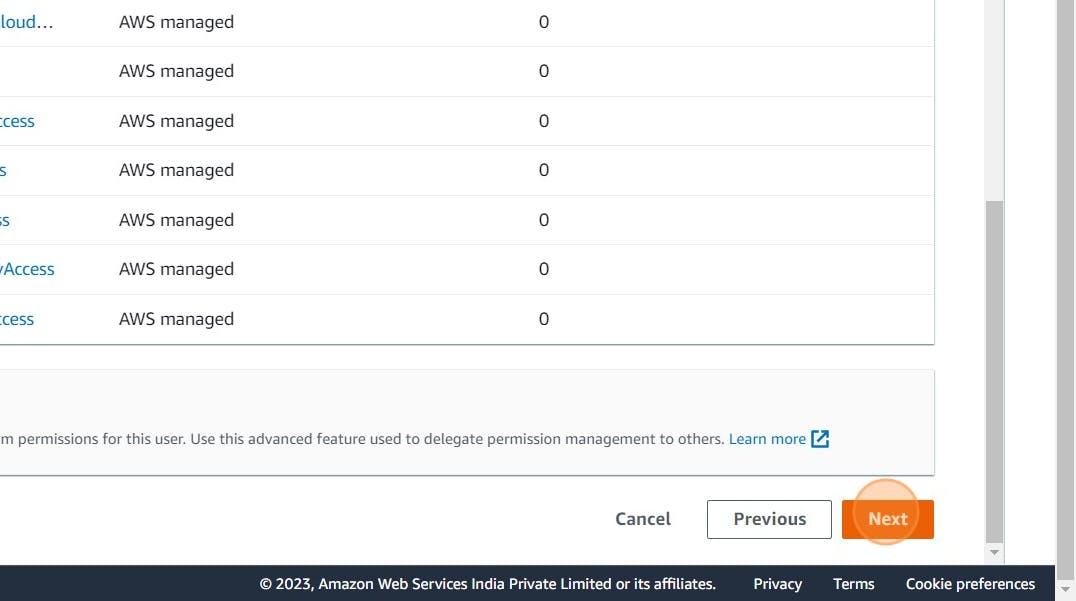
Click "Attach policies directly"



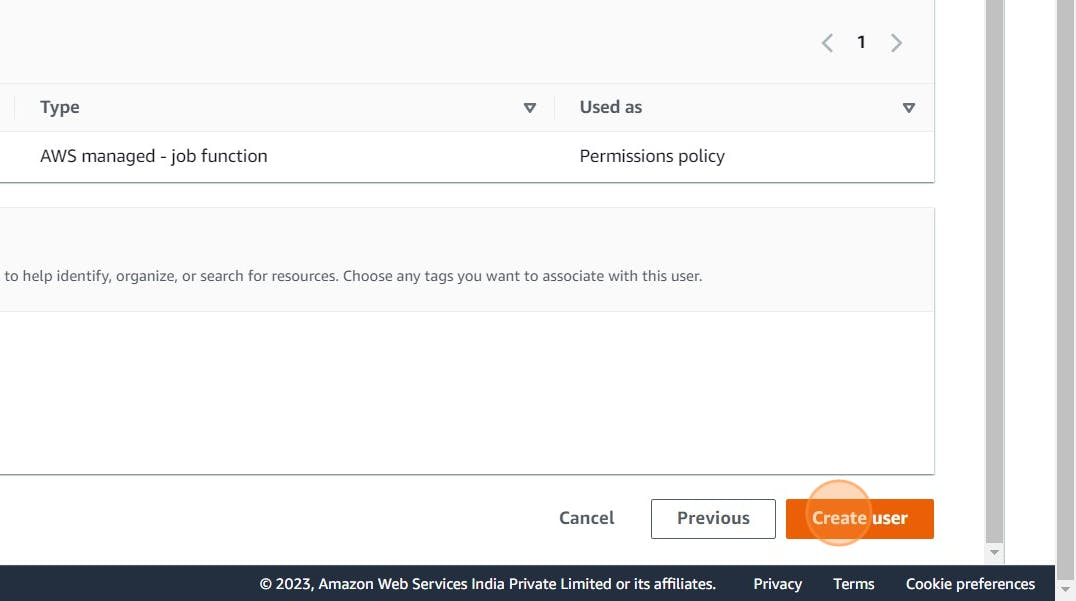
Click this checkbox with Administrator access



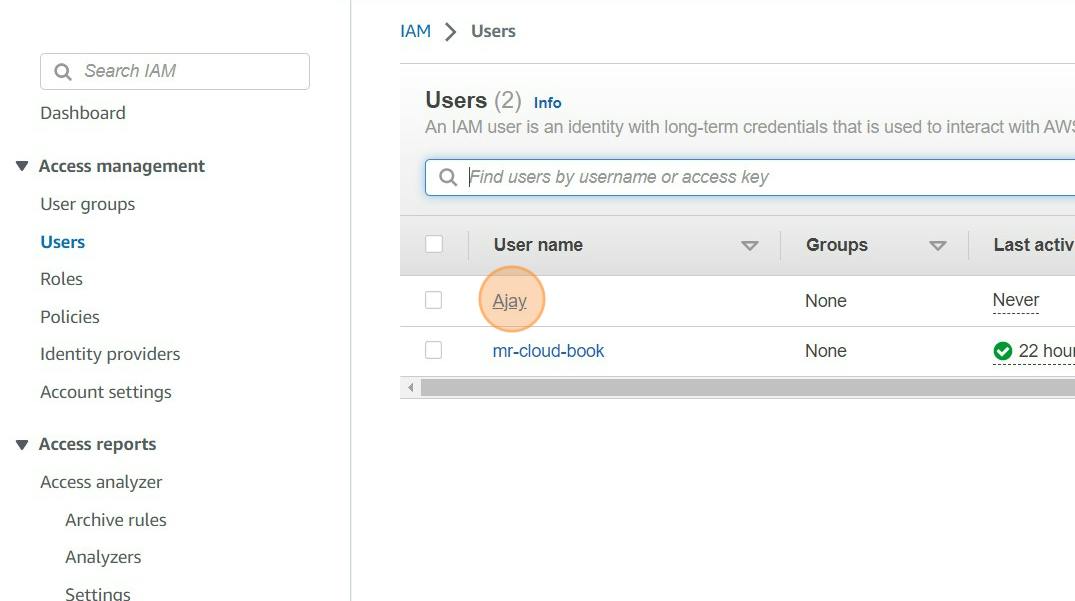
Click "Next"



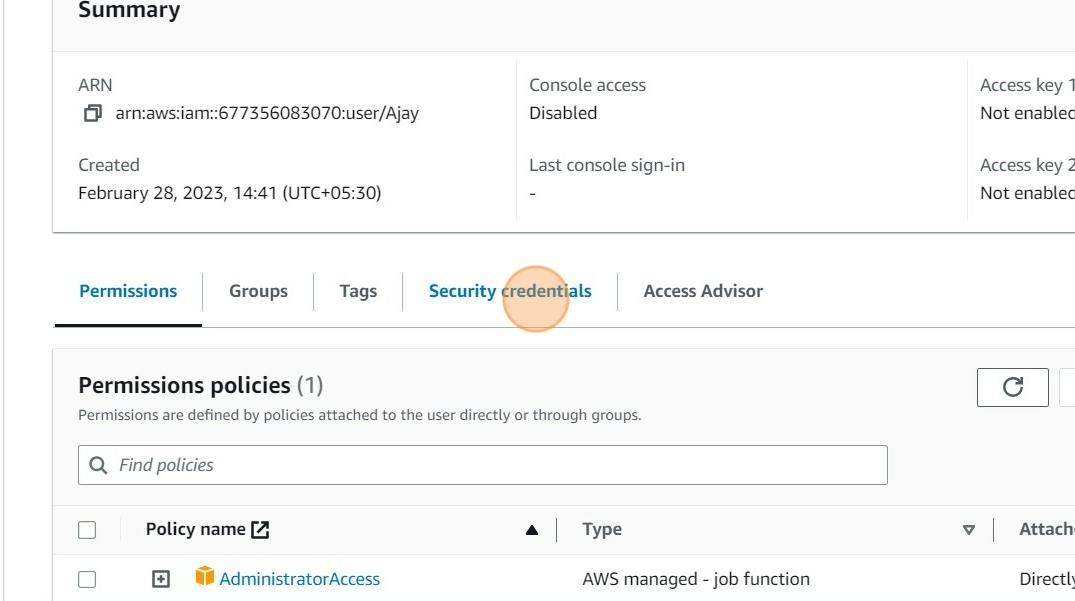
Click "Create user"



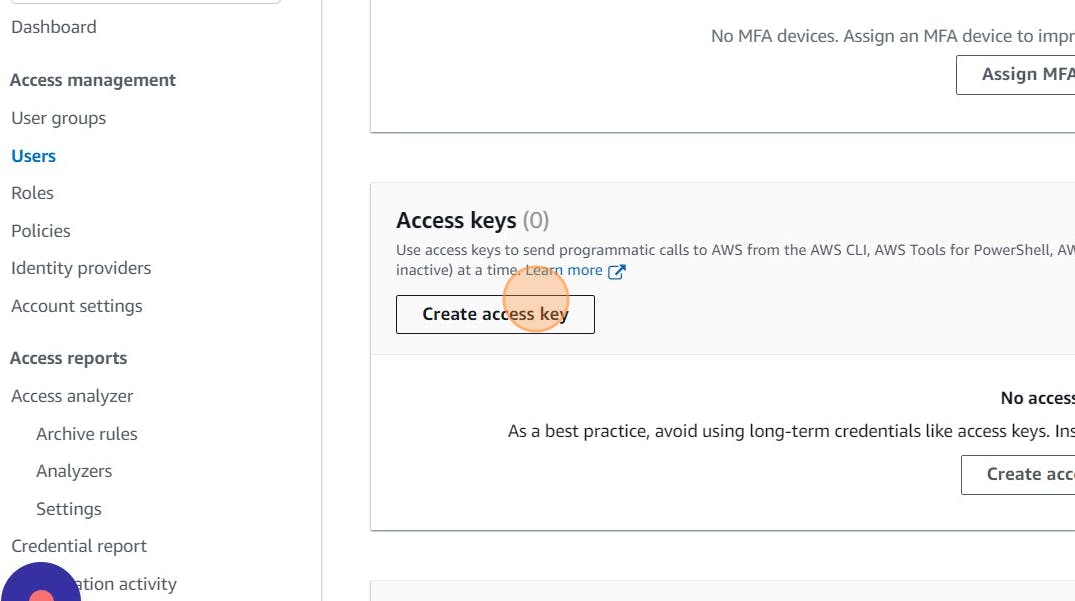
Click newly created user in my case Ajay



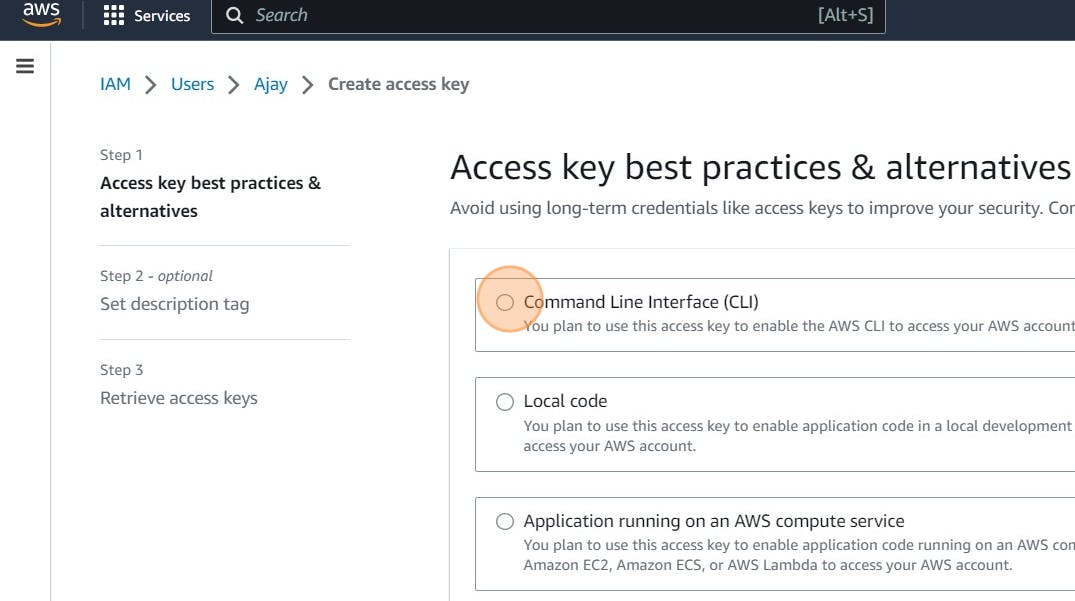
Click "Security credentials"



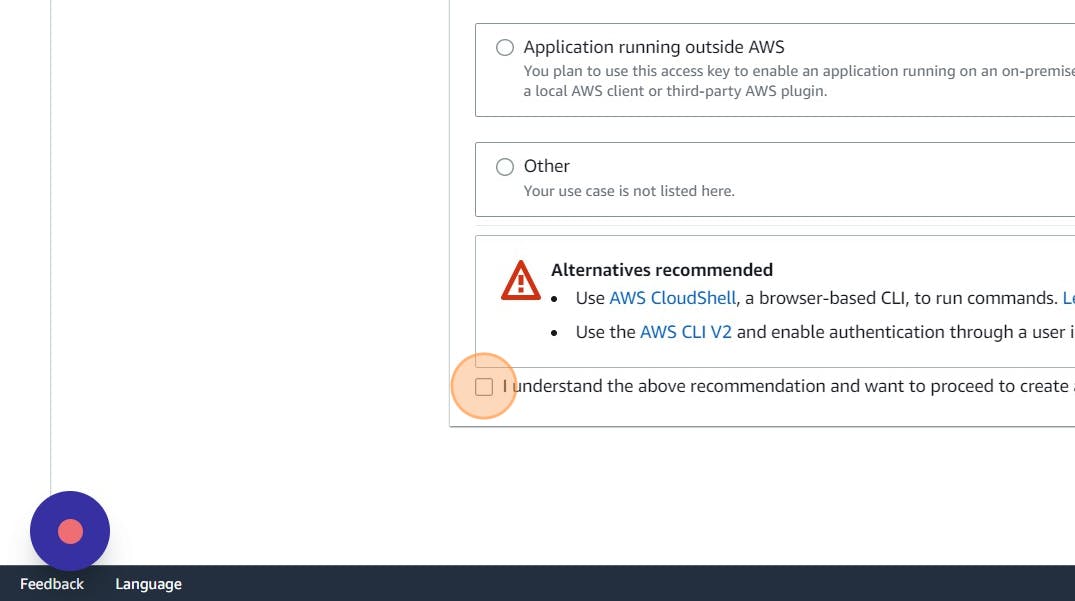
Click "Create access key"



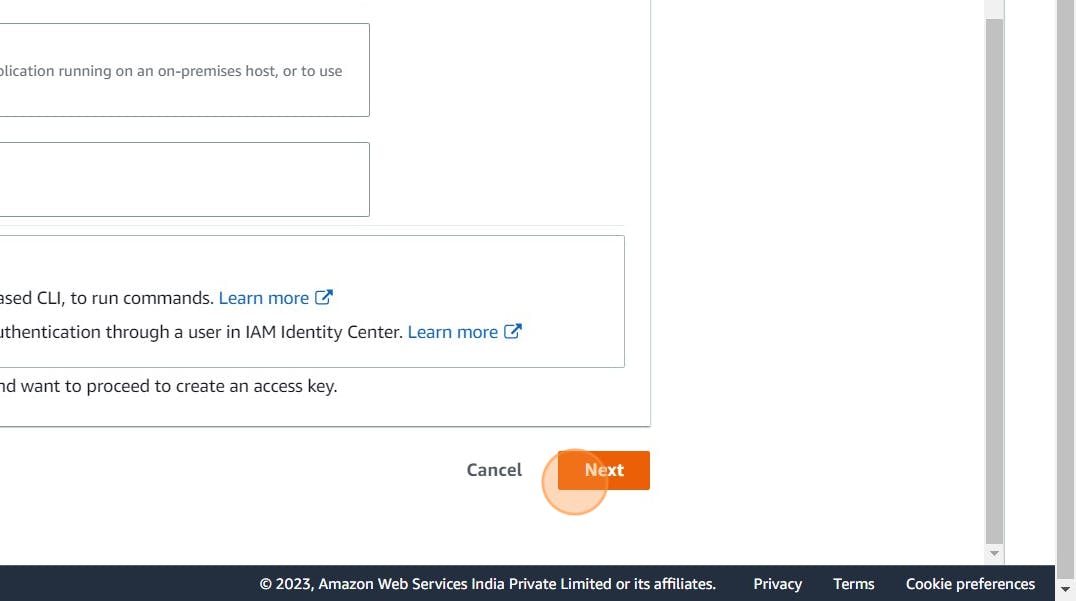
Click this radio button with the CLI



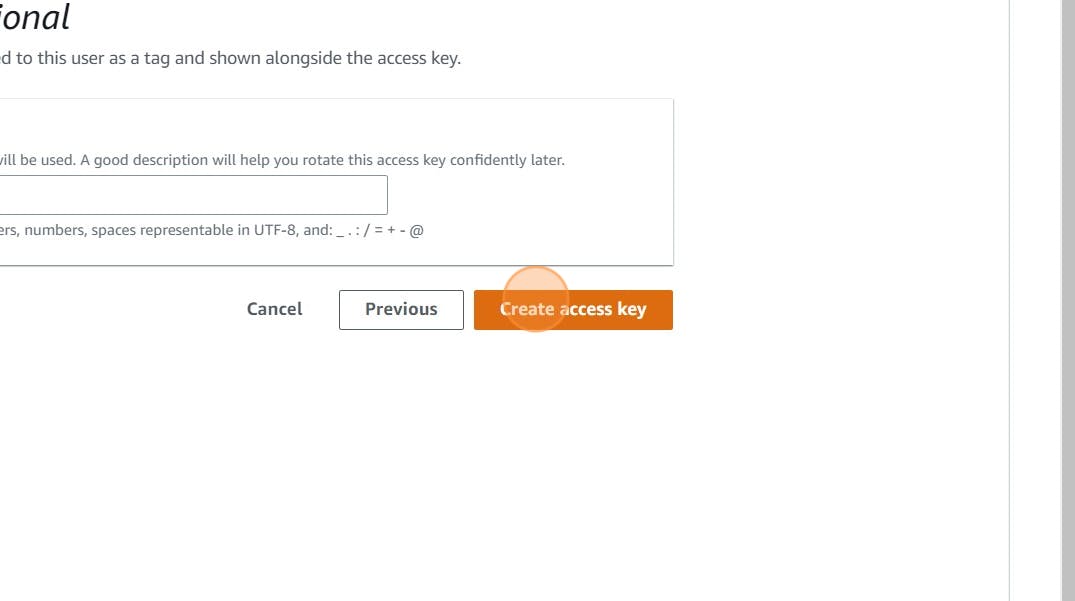
Agree to terms



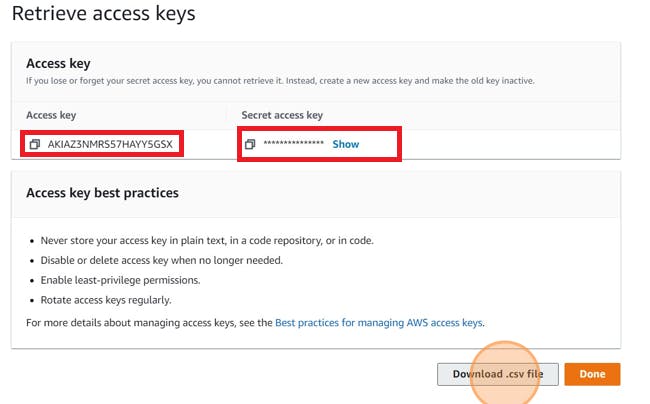
Click Next



Click "Create access key"



Download .csv file



Step4: Aws Configure

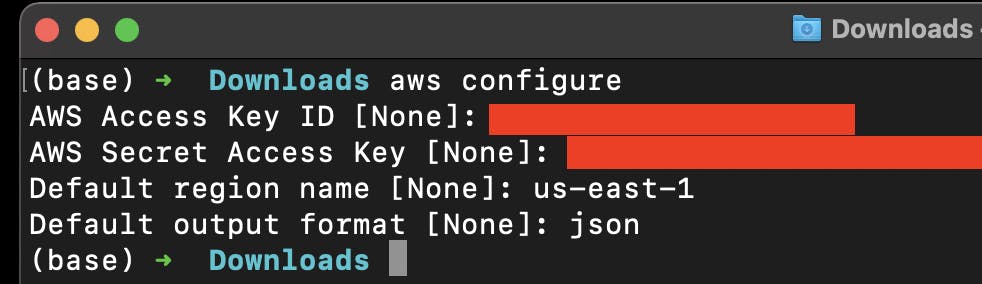
Go to vs code or Cmd your wish

COPY

COPY

aws configure

Provide your Aws Access key and Secret Access key



Step5: Terraform files and Provision

[**main.tf**](http://main.tf/)

resource "aws\_instance" "web" {

ami = "ami-0f5ee92e2d63afc18" #change ami id for different region

instance\_type = "t2.large"

key\_name = "Mumbai"

vpc\_security\_group\_ids = [aws\_security\_group.Jenkins-sg.id]

user\_data = templatefile("./install.sh", {})

tags = {

Name = "Jenkins-sonarqube"

}

root\_block\_device {

volume\_size = 30

}

}

resource "aws\_security\_group" "Jenkins-sg" {

name = "Jenkins-sg"

description = "Allow TLS inbound traffic"

ingress = [

for port in [22, 80, 443, 8080, 9000] : {

description = "inbound rules"

from\_port = port

to\_port = port

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = []

prefix\_list\_ids = []

security\_groups = []

self = false

}

]

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "jenkins-sg"

}

}

[**provider.tf**](http://provider.tf/)

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

}

}

# Configure the AWS Provider

provider "aws" {

region = "ap-south-1" #change your region

}

[**install.sh**](http://install.sh/)

This will install Jenkins and Docker and Sonarqube and trivy

#!/bin/bash

sudo apt update -y

wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc

echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION\_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list

sudo apt update -y

sudo apt install temurin-17-jdk -y

/usr/bin/java --version

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update -y

sudo apt-get install jenkins -y

sudo systemctl start jenkins

sudo systemctl status jenkins

#install docker

sudo apt-get update

sudo apt-get install docker.io -y

sudo usermod -aG docker ubuntu

newgrp docker

sudo chmod 777 /var/run/docker.sock

docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

#install trivy

sudo apt-get install wget apt-transport-https gnupg lsb-release -y

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy –y

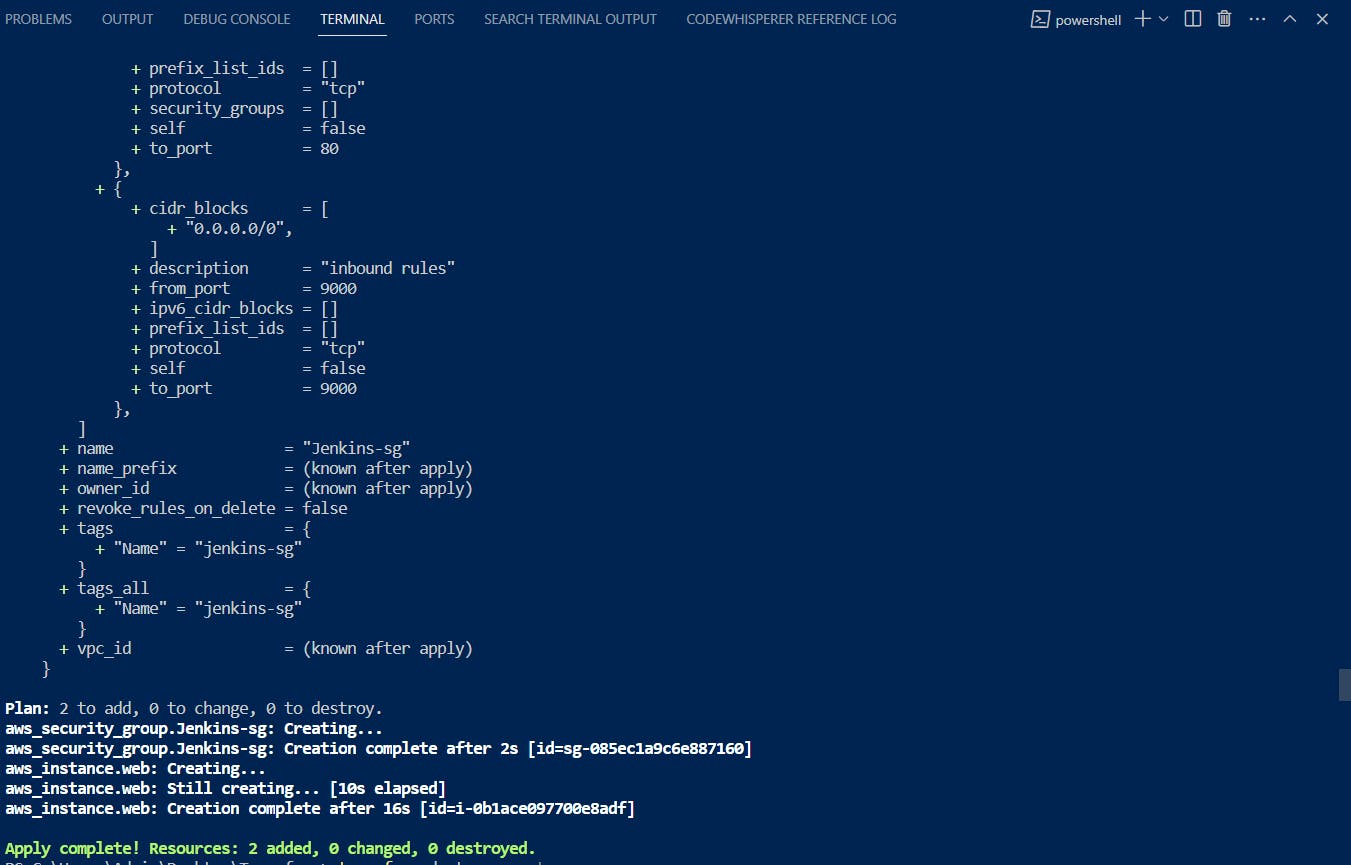
**Terraform commands to provision**

terraform init

terraform validate

terraform plan

terraform apply

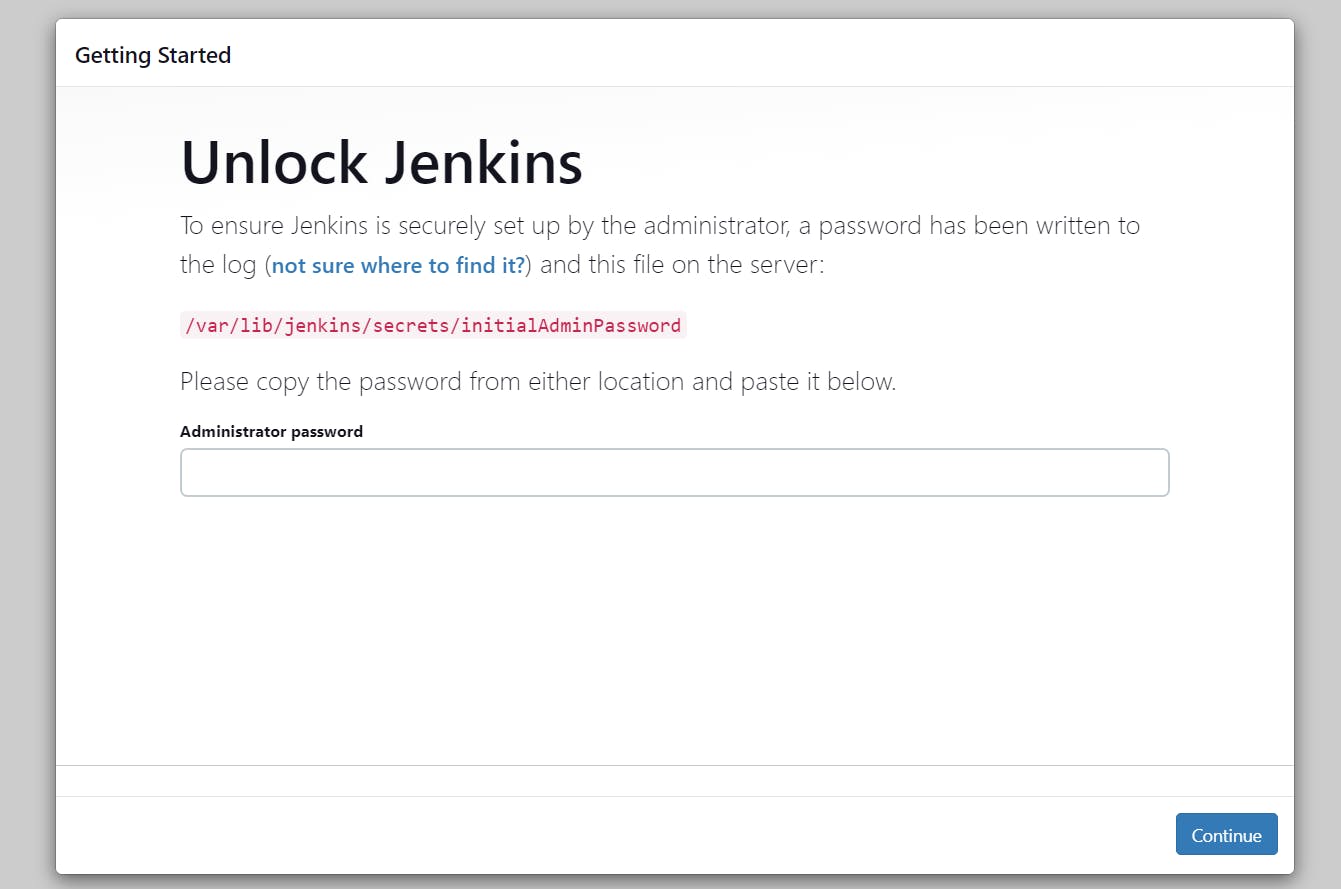


**output**

<instance-ip:8080> #jenkins

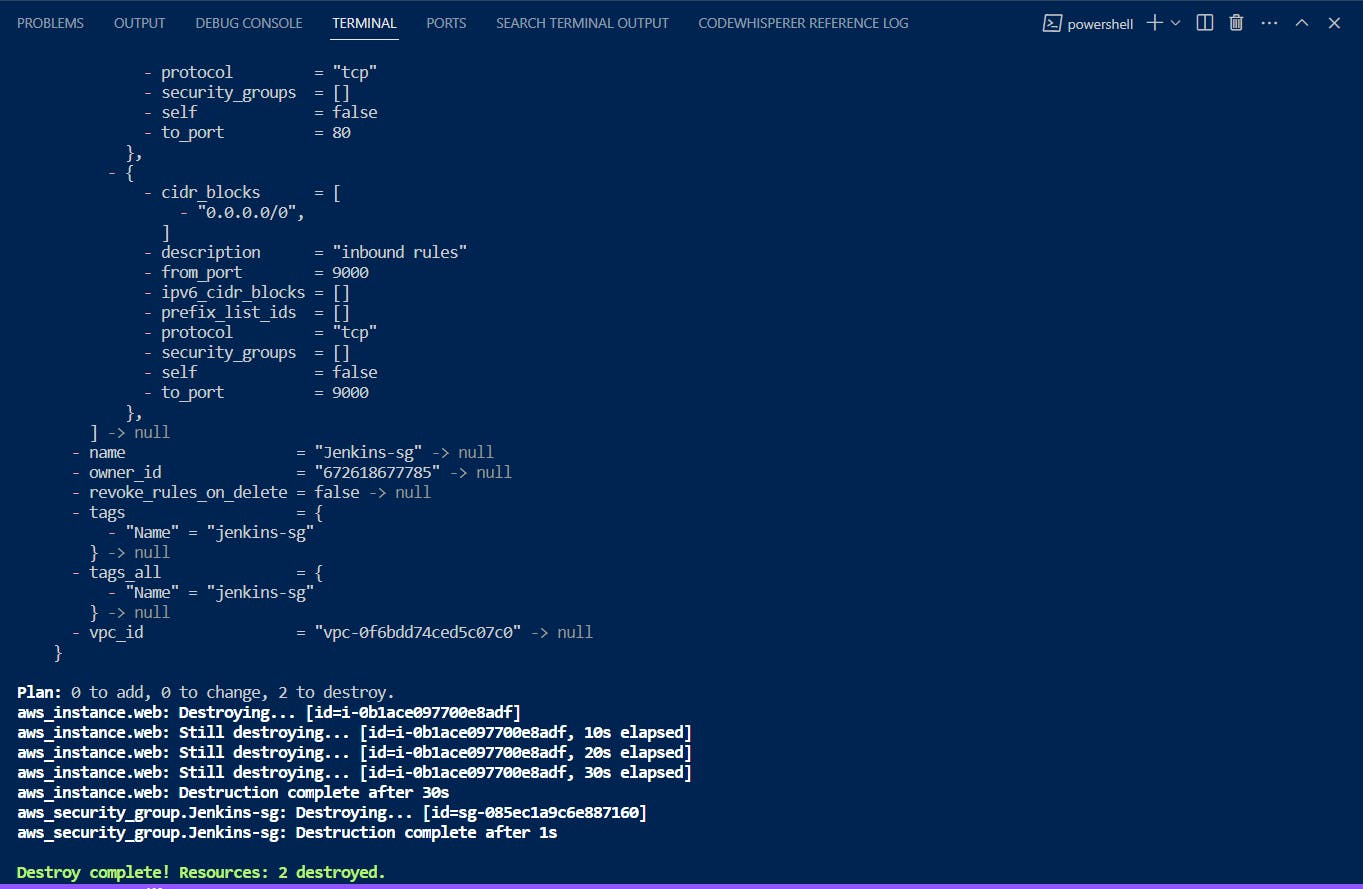
<instance-ip:9000> sonarqube

trivy --version #connect to putty and see



**Destroy**

terraform destroy



As we conclude our journey into the world of Infrastructure as Code (IaC) and automation, we've witnessed the power of Terraform in provisioning resources, Jenkins in automating our continuous integration, and SonarQube in maintaining code quality. The ability to create and manage infrastructure as code simplifies deployment, scaling, and maintenance processes, making it an invaluable asset for modern development teams. By employing these tools, we've taken significant strides towards a more efficient and effective DevOps workflow. Embrace the world of IaC, explore the vast possibilities it offers, and empower your team to build, deploy, and maintain exceptional software with confidence. Now, you're equipped to embark on your automation journey and elevate your projects to new heights. Happy coding!