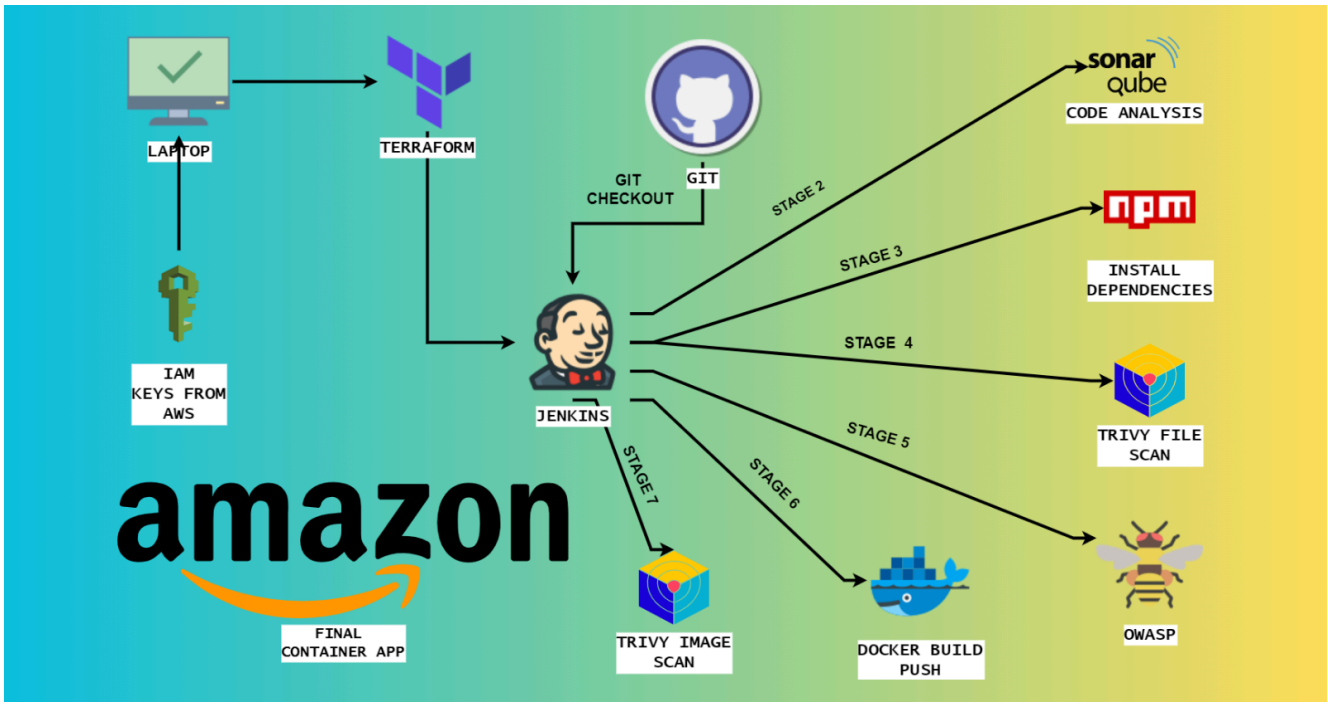
DevOps

Amazon App Deployment: A DevSecOps Approach with Terraform and Jenkins CI/CD

mrcloudbook.com · 8 January 2024



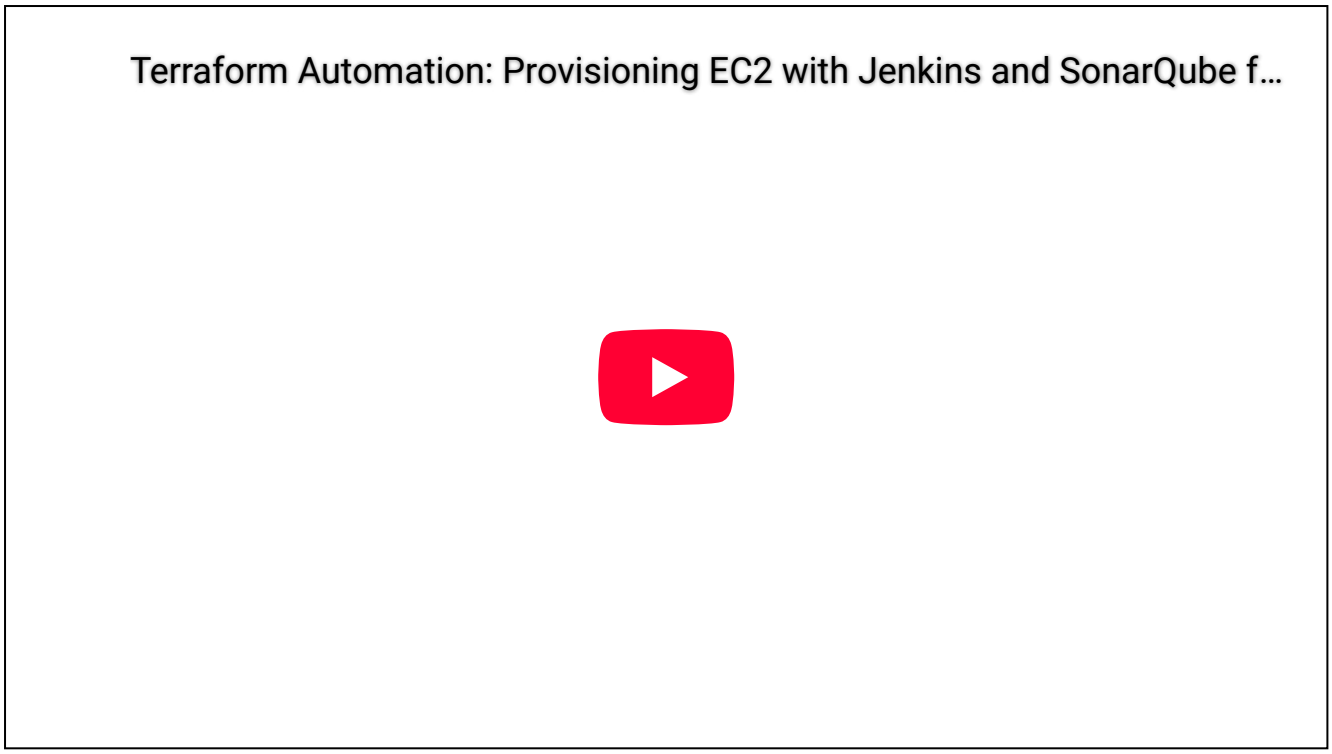
Welcome to our in-depth guide on deploying an Amazon app with a strong focus on security through a DevSecOps approach. In today's fast-paced digital landscape, building and deploying applications not only requires speed but also airtight security. That's where DevSecOps comes into play, blending development, security, and operations into a single, unified process. In this blog post, we will embark on a journey where we'll leverage Terraform, Jenkins CI/CD, SonarQube, and Trivy to create a robust and secure pipeline for deploying applications on Amazon Web

Services (AWS). Whether you're an experienced developer looking to enhance your DevSecOps skills or a newcomer eager to explore this exciting intersection of software development and security, this guide has something valuable to offer. Let's dive in and explore the steps to safeguard your Amazon app while ensuring smooth and efficient deployment.

Contents [hide]

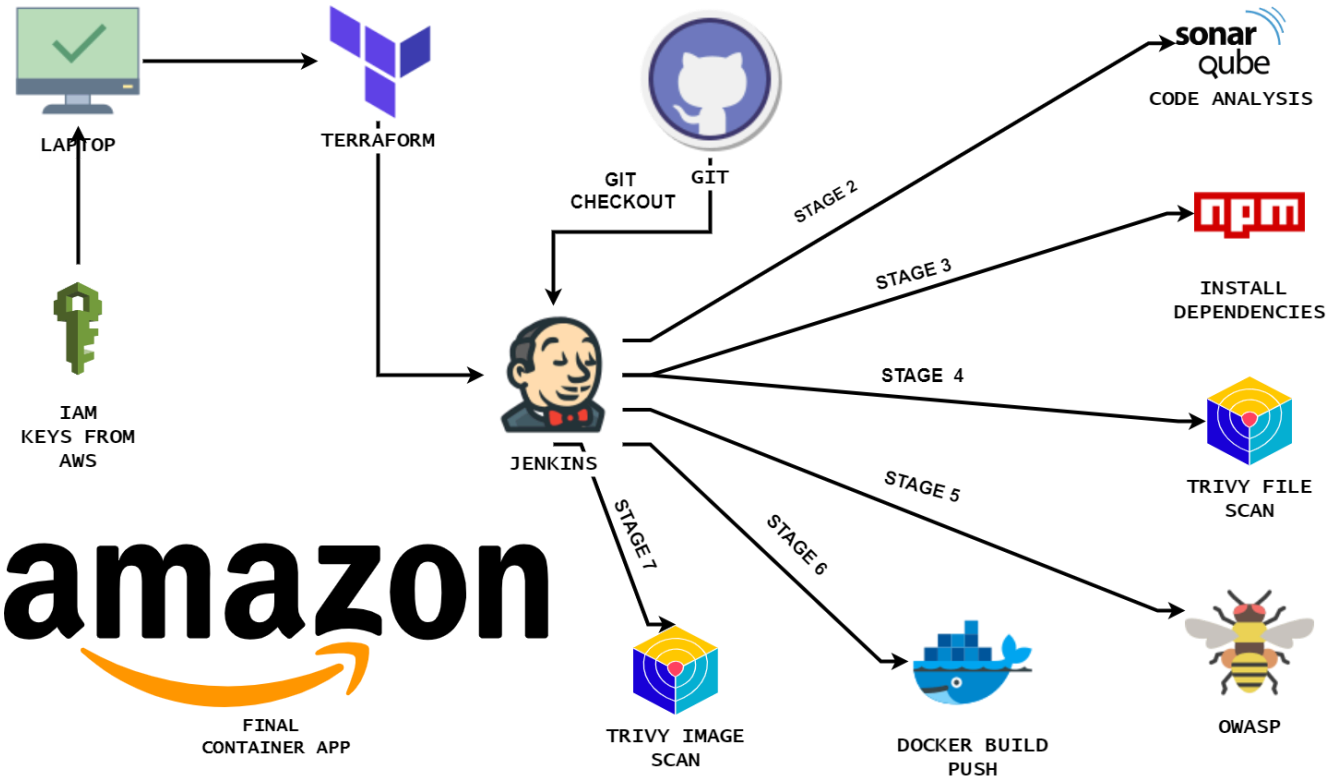
COMPLETE PROJECT VIDEO ON YOUTUBE

If you have doubts watch this video on YouTube



GITHUB LINK

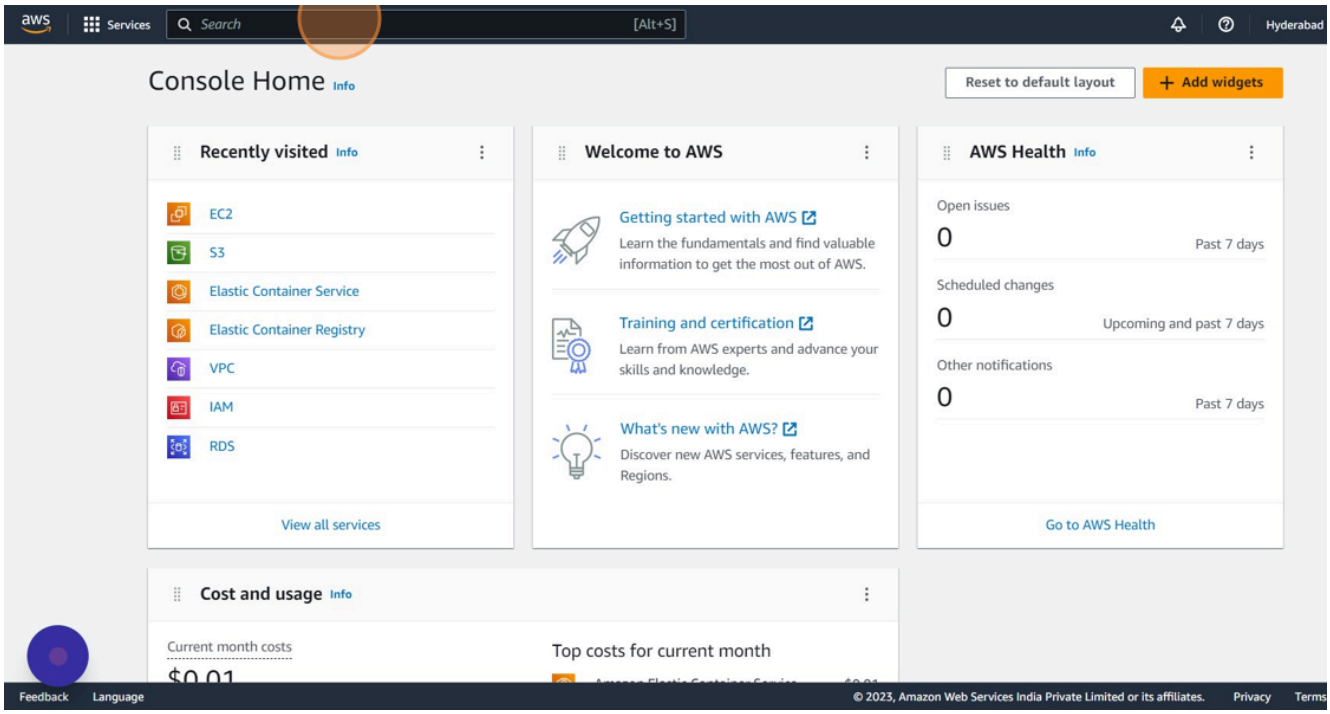
https://github.com/Aj7Ay/Amazon-FE.git



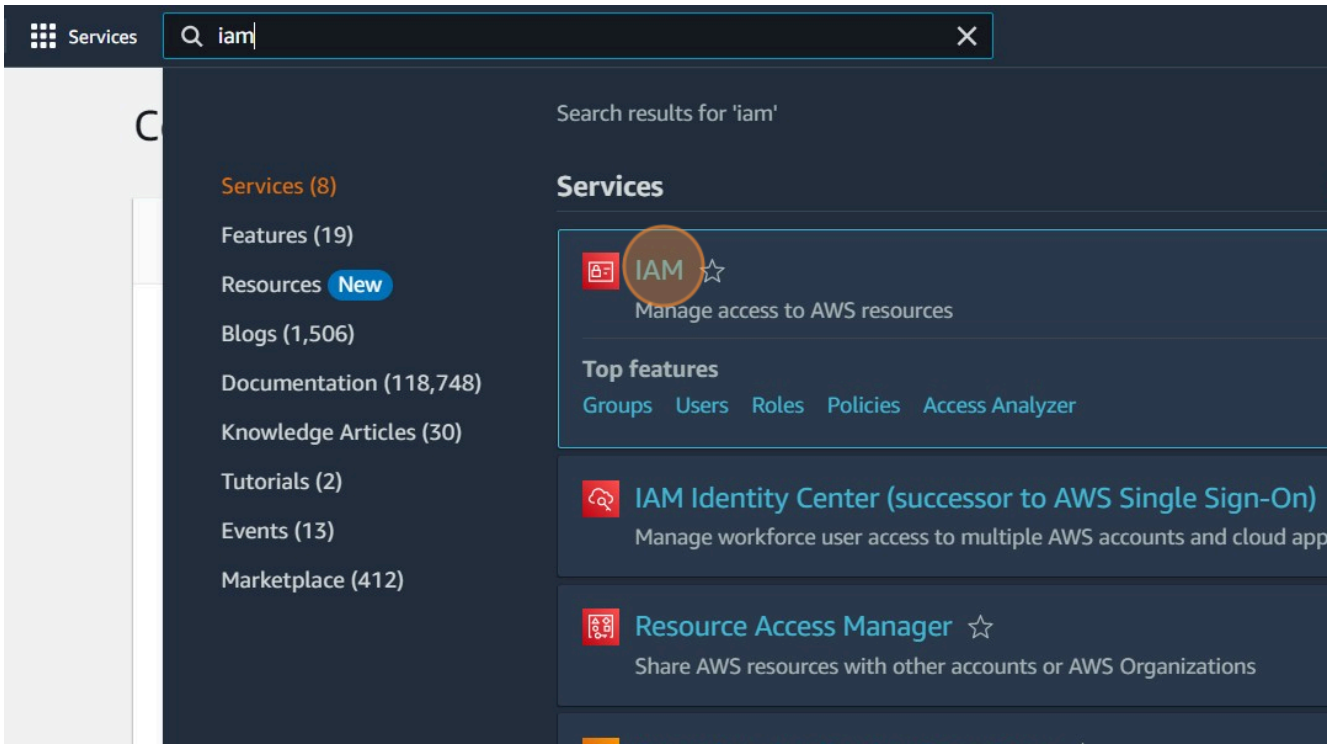
Step1: create an IAM user

Navigate to the **AWS console**

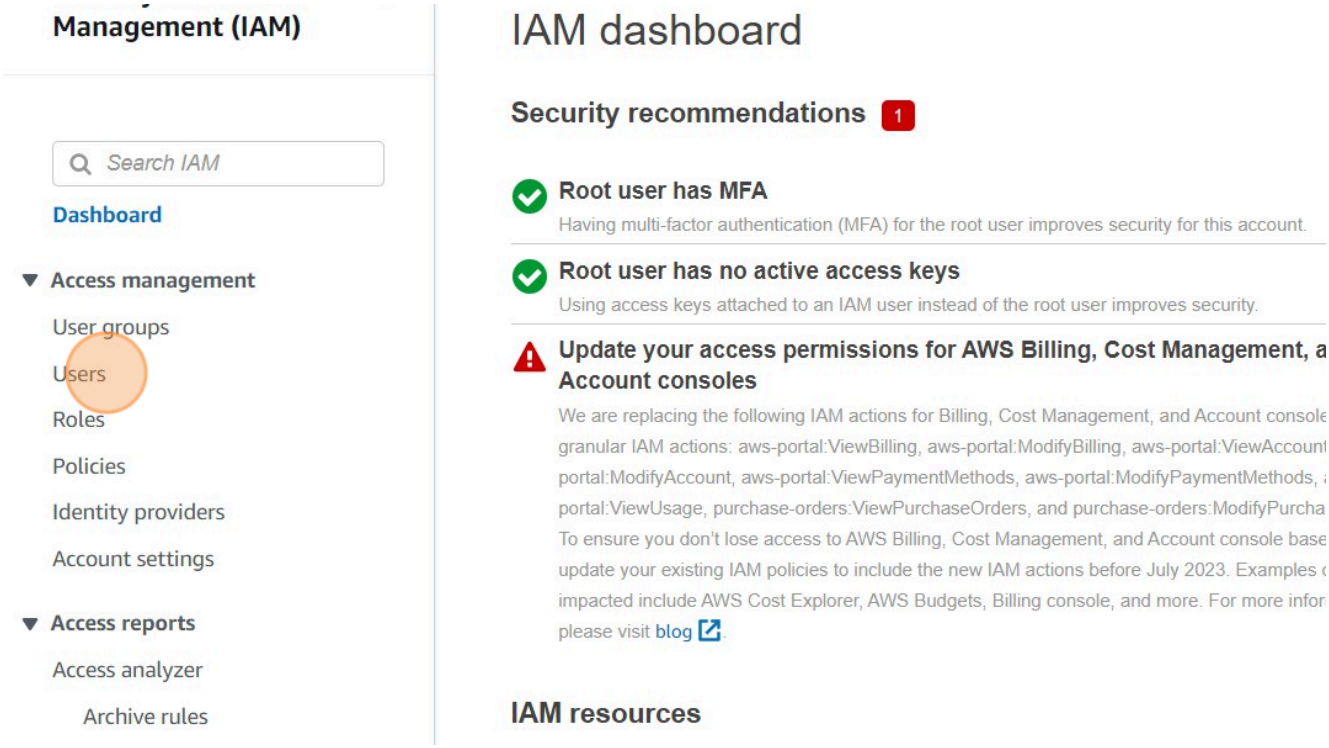
Click the “Search” field.



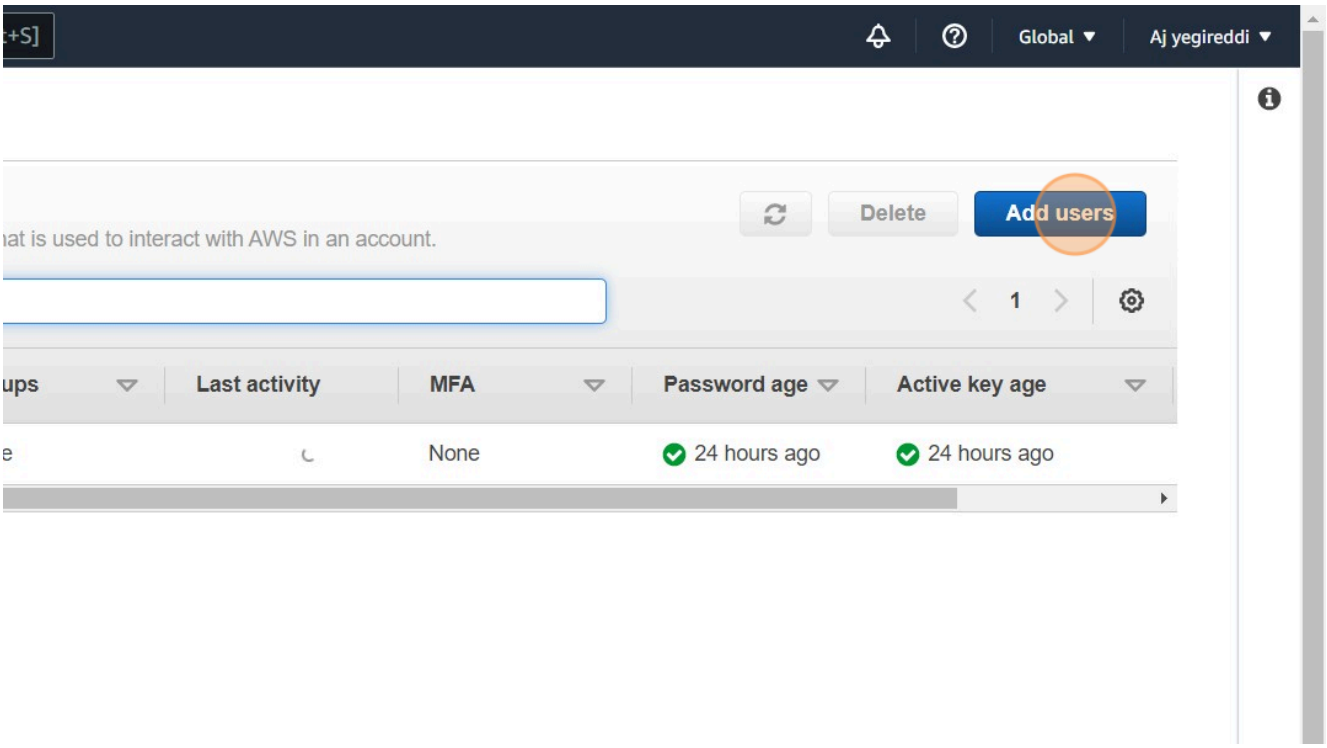
Search for IAM



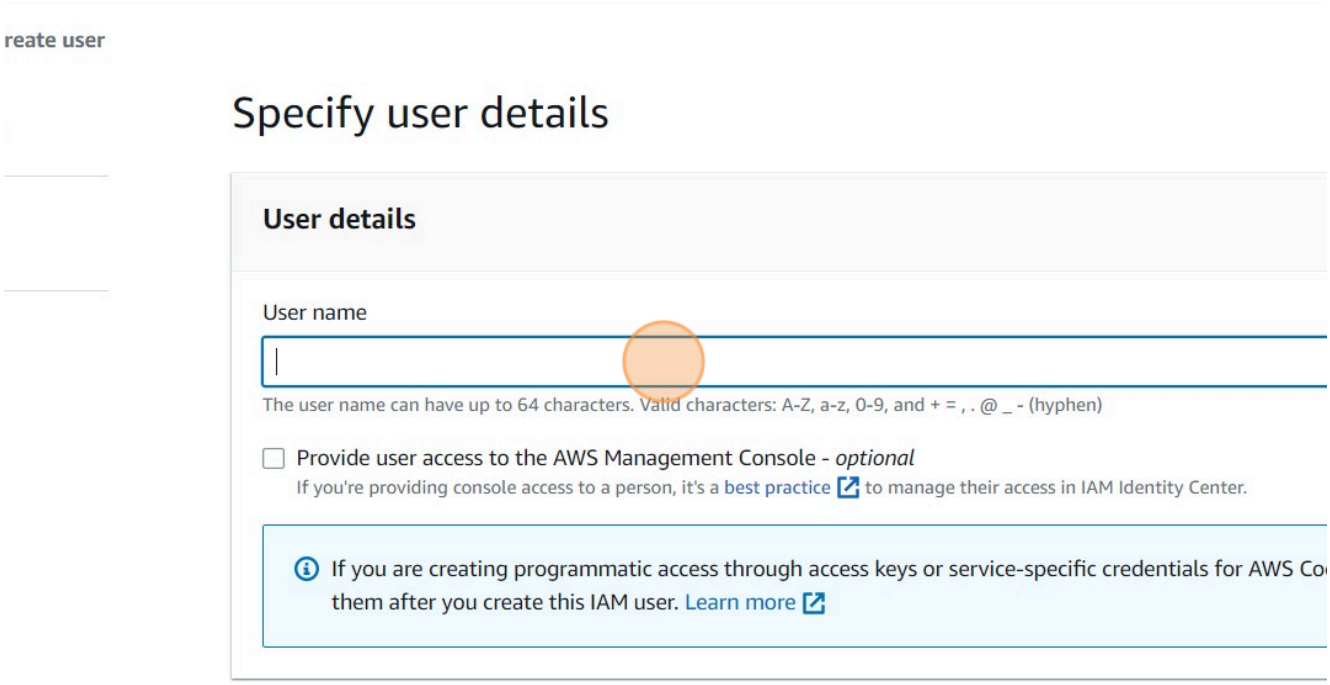
Click “Users”



Click “Add users”

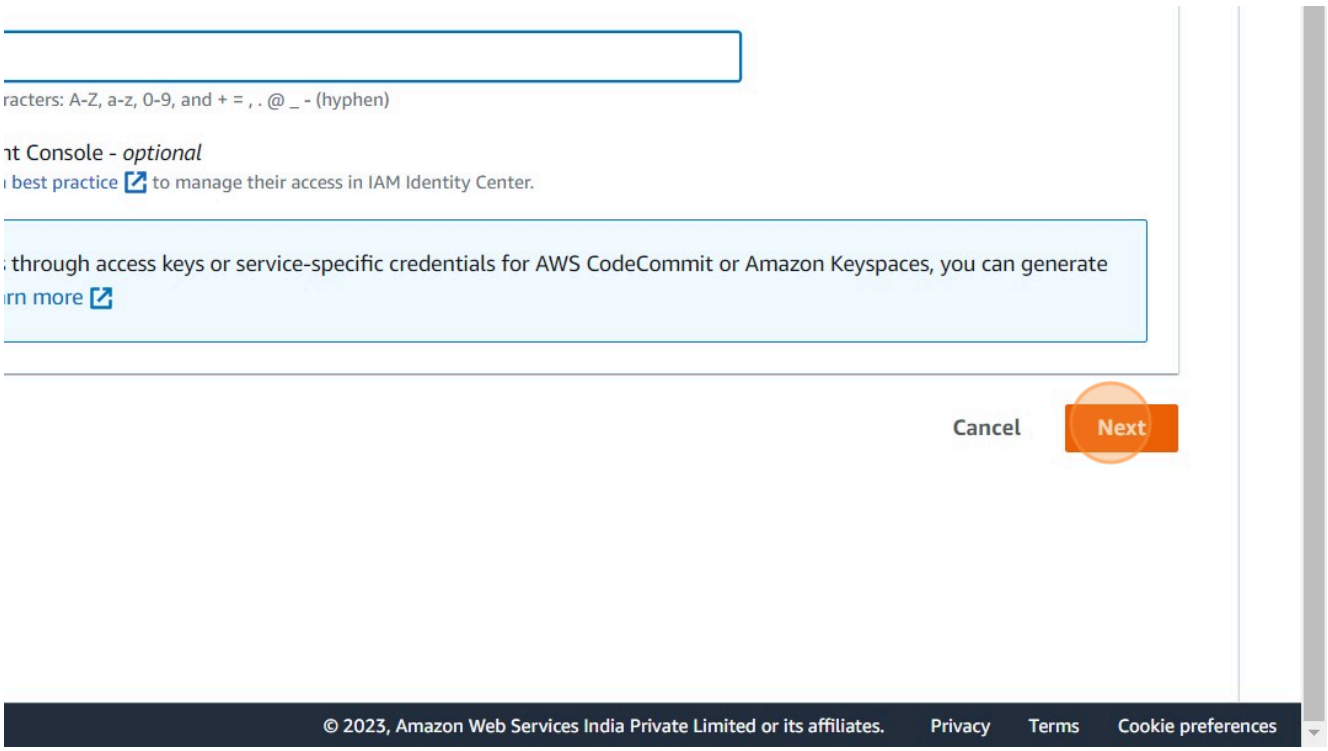


Click the “User name” field.

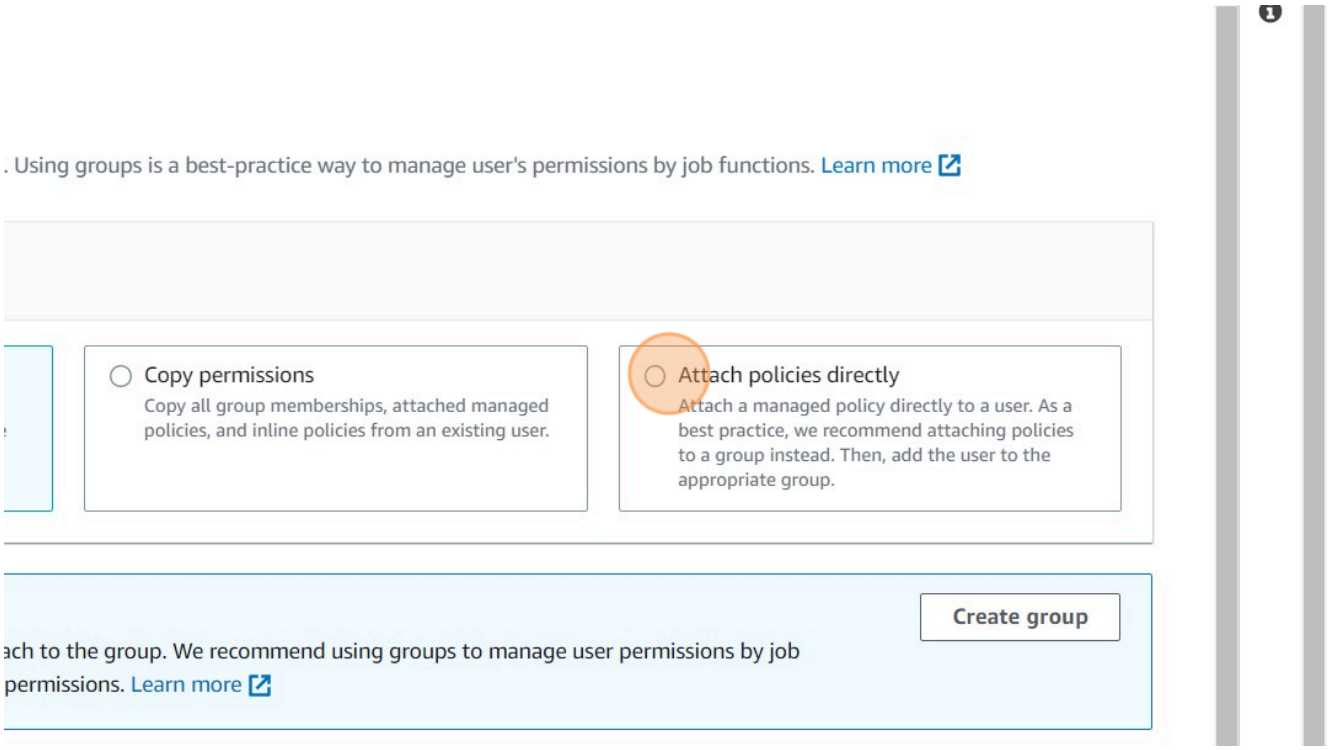


Type “Terraform” or as you wish about the name

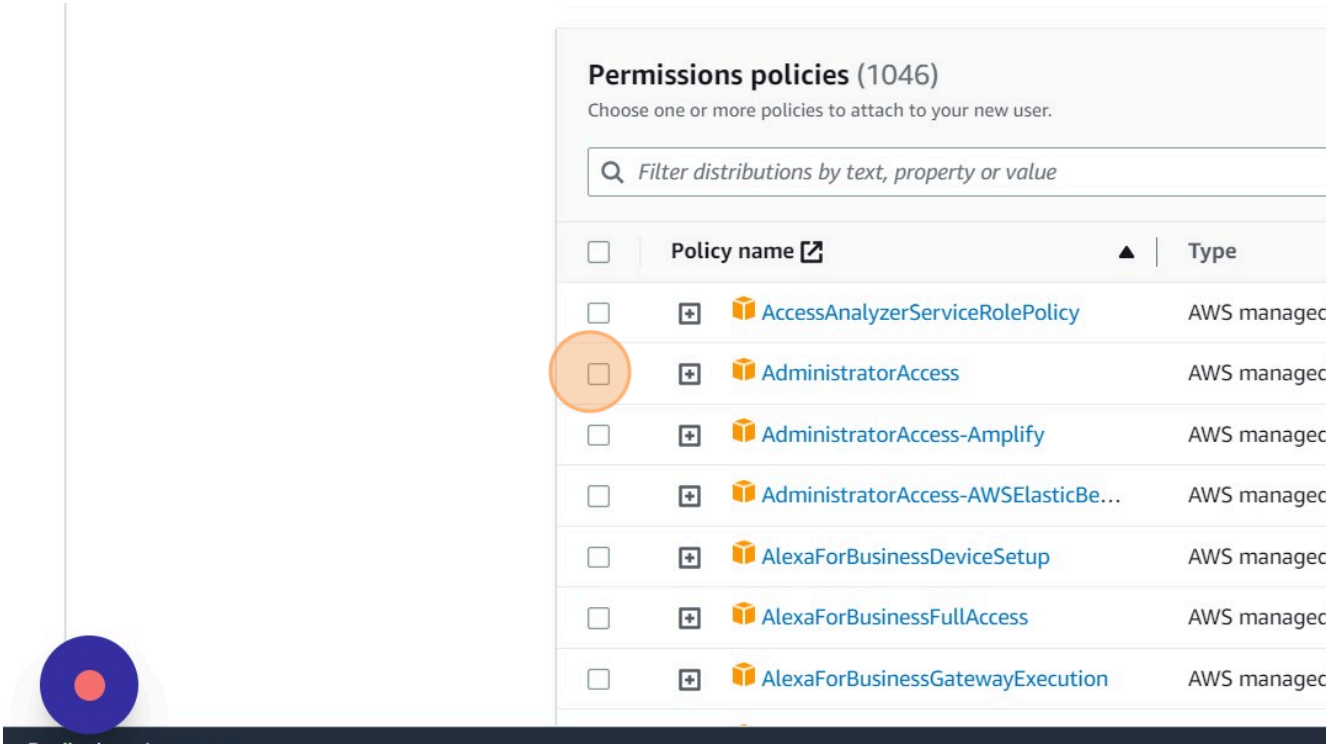
Click Next



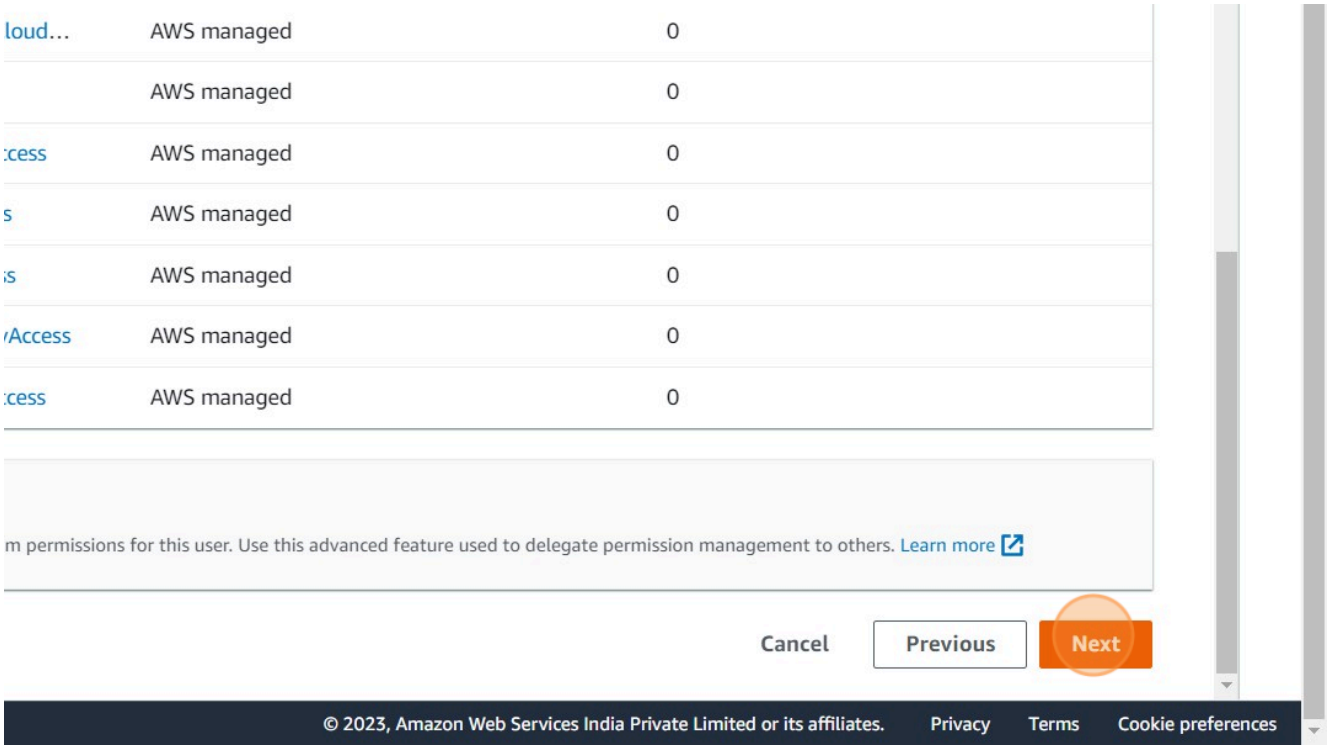
Click “Attach policies directly”



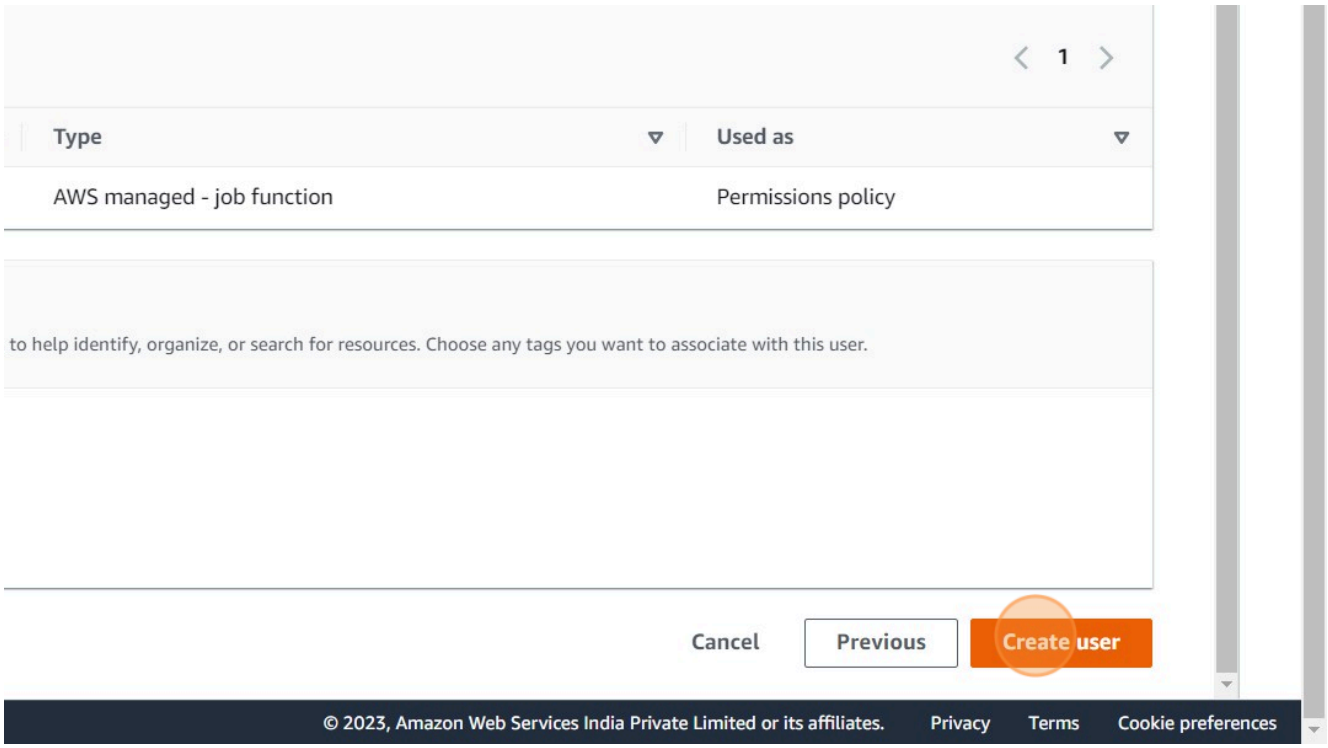
Click this checkbox with Administrator access



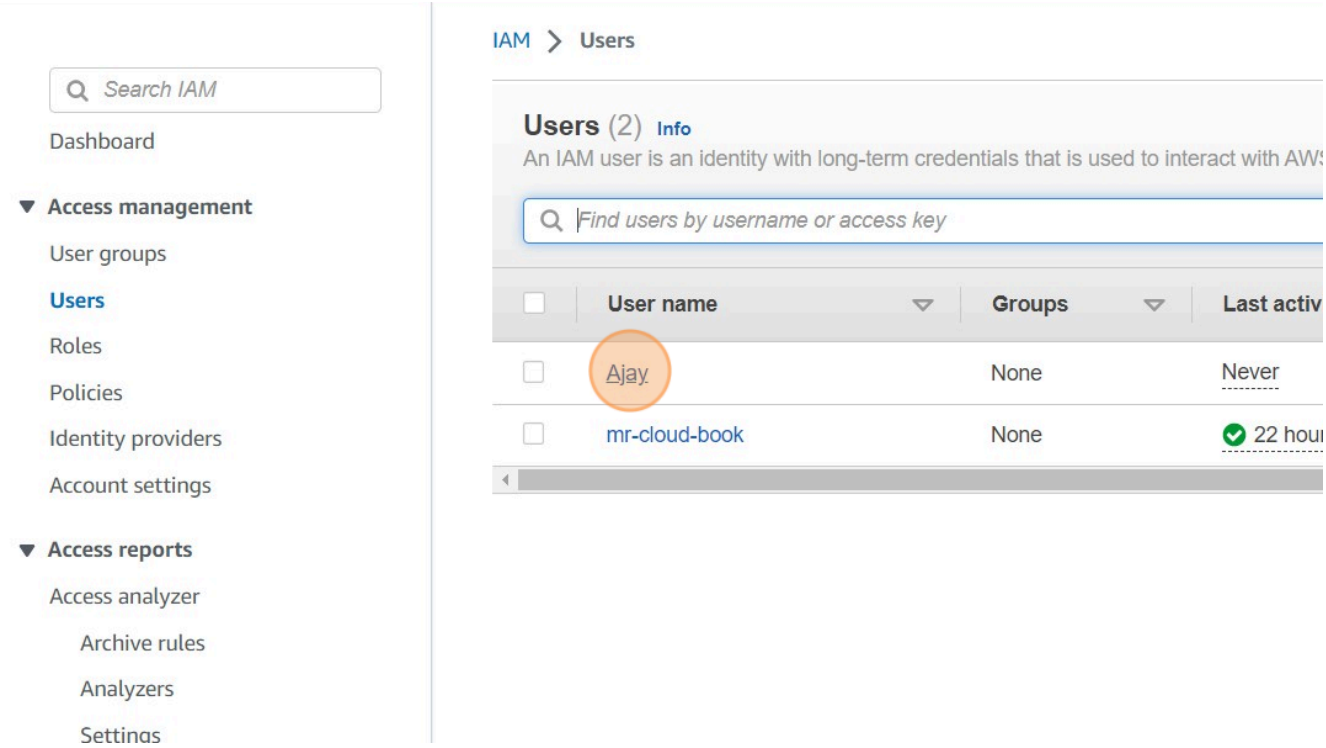
Click “Next”



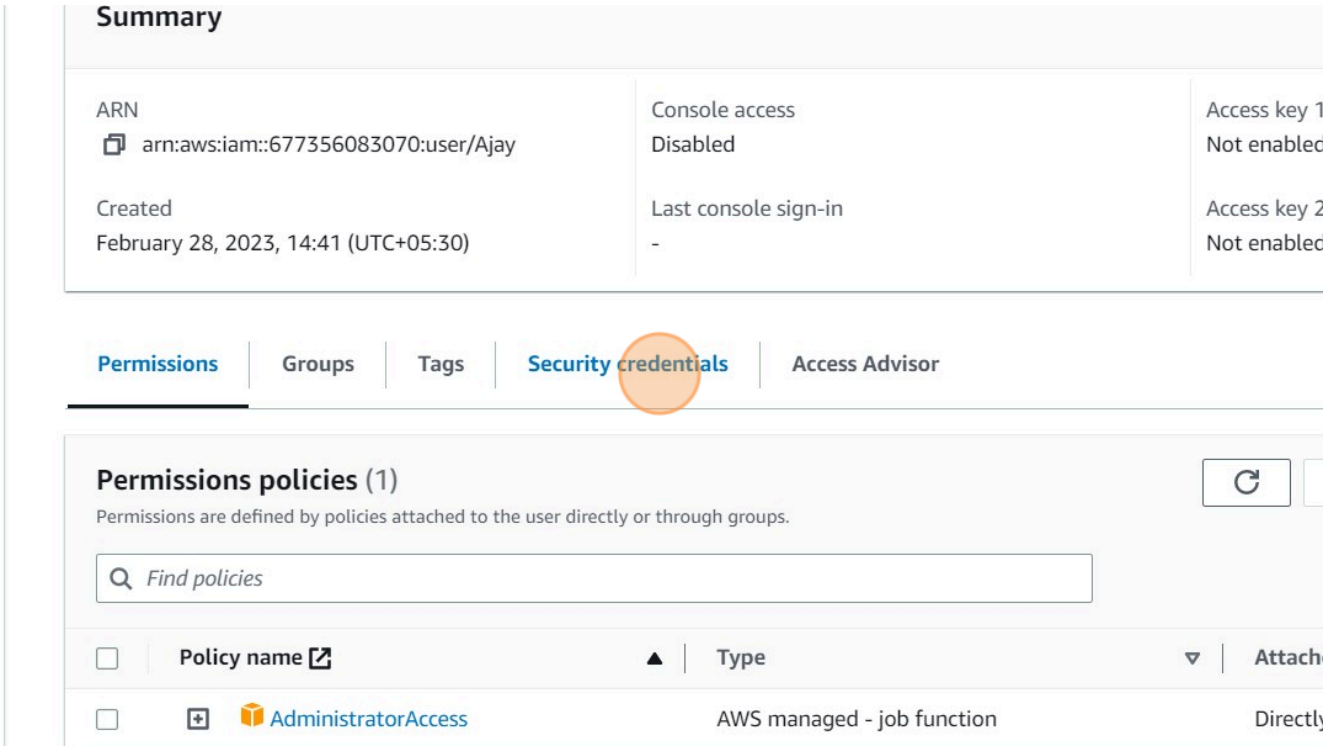
Click “Create user”



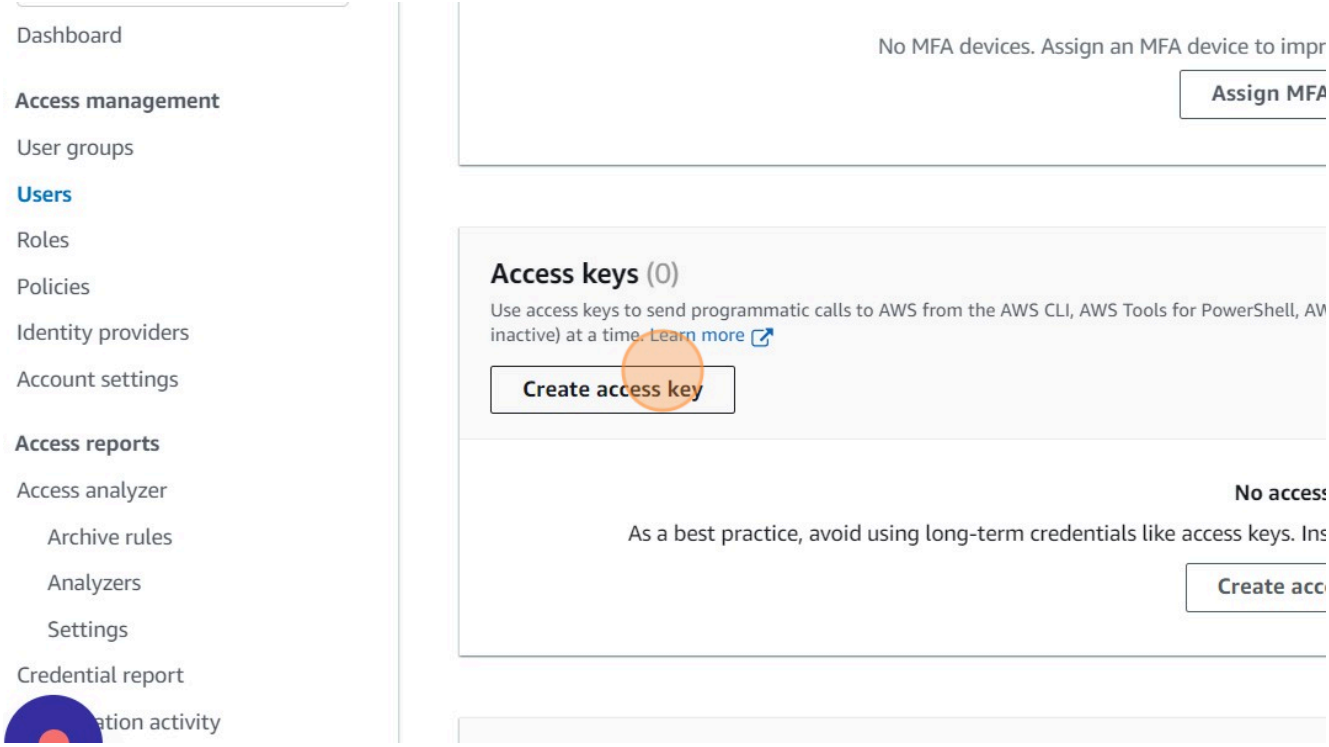
Click newly created user in my case Ajay



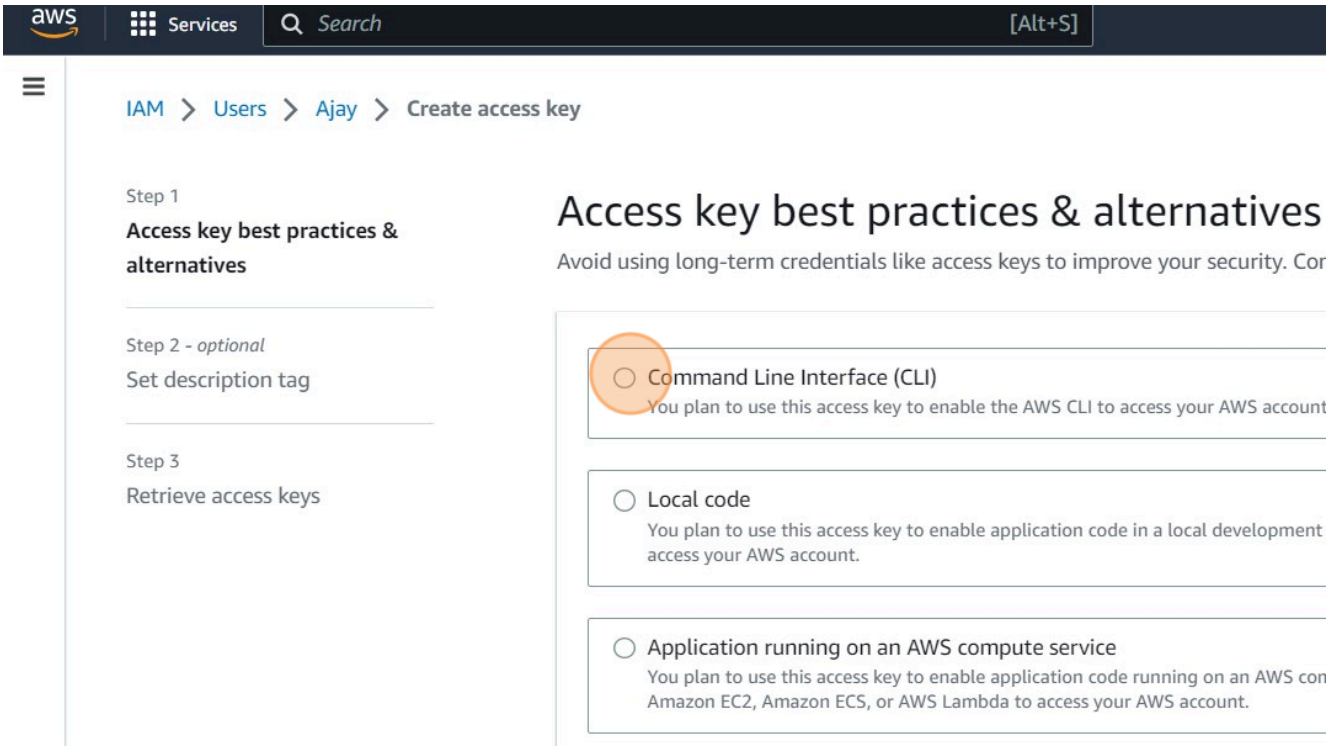
Click “Security credentials”



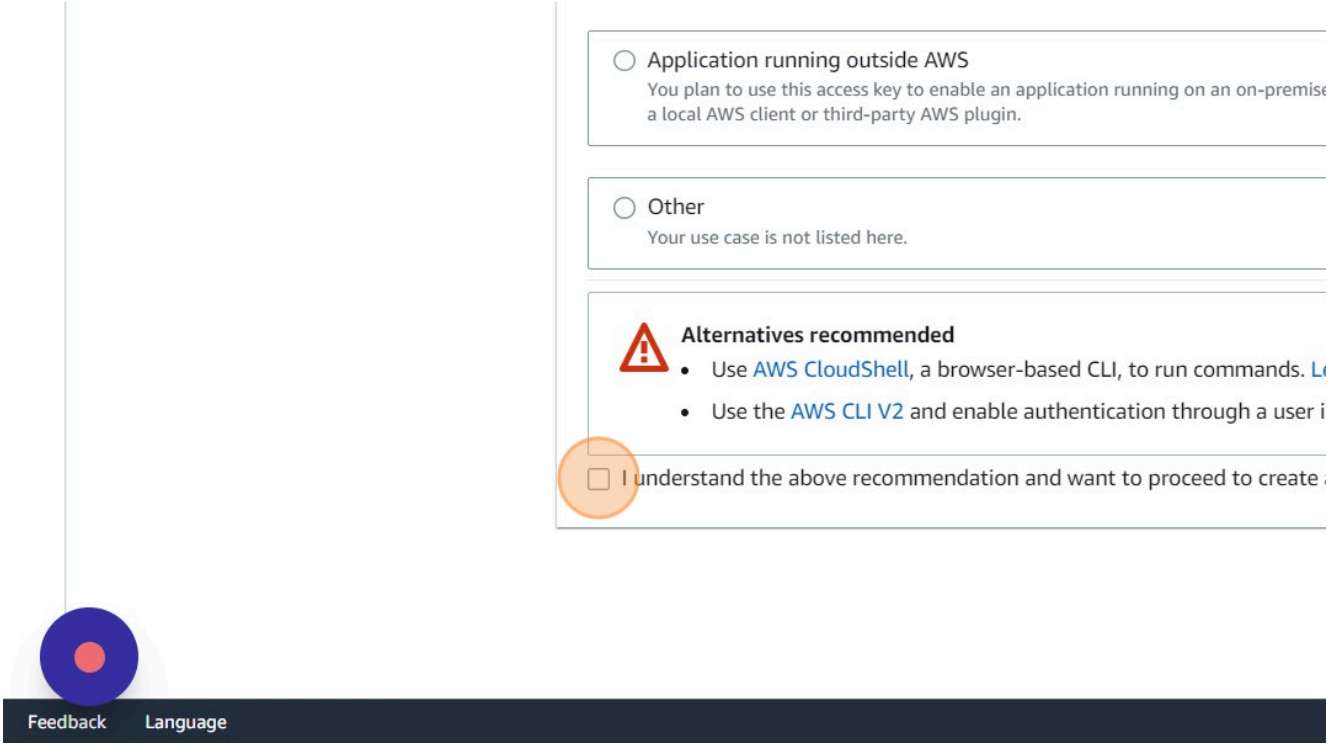
Click “Create access key”



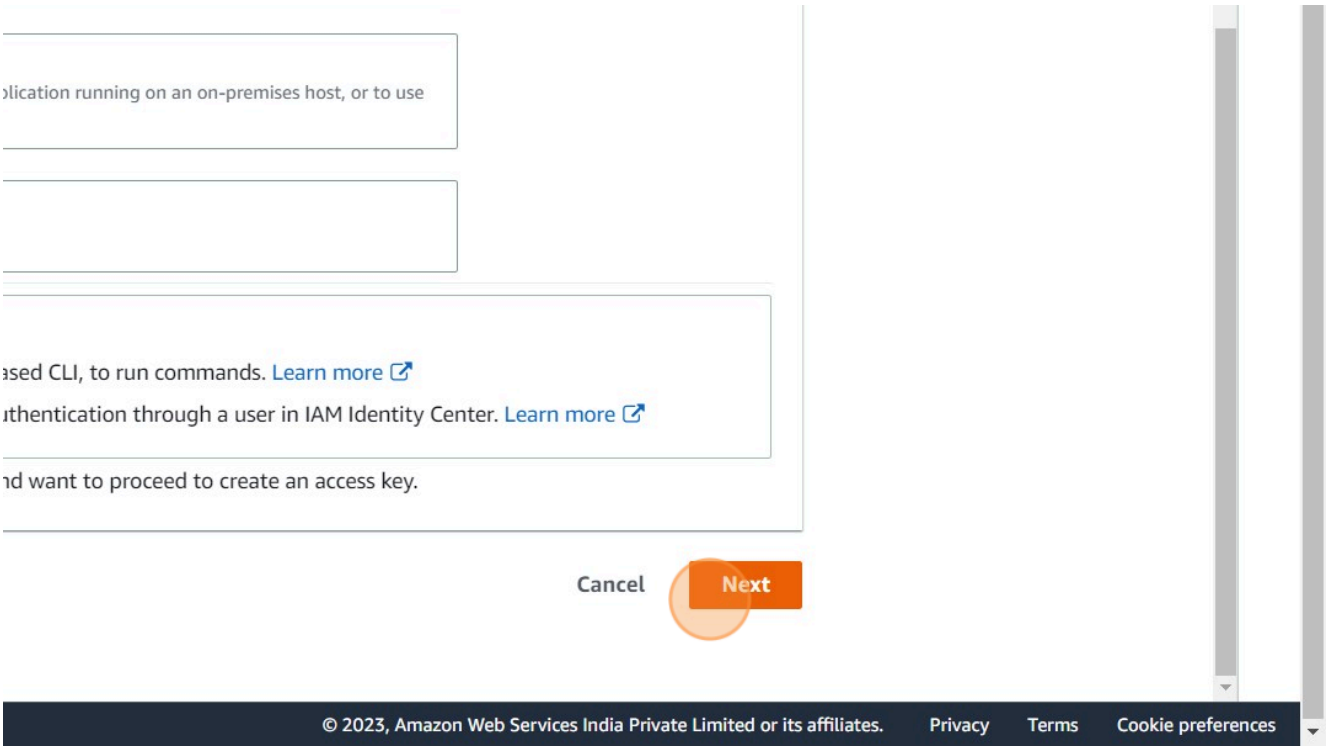
Click this radio button with the CLI



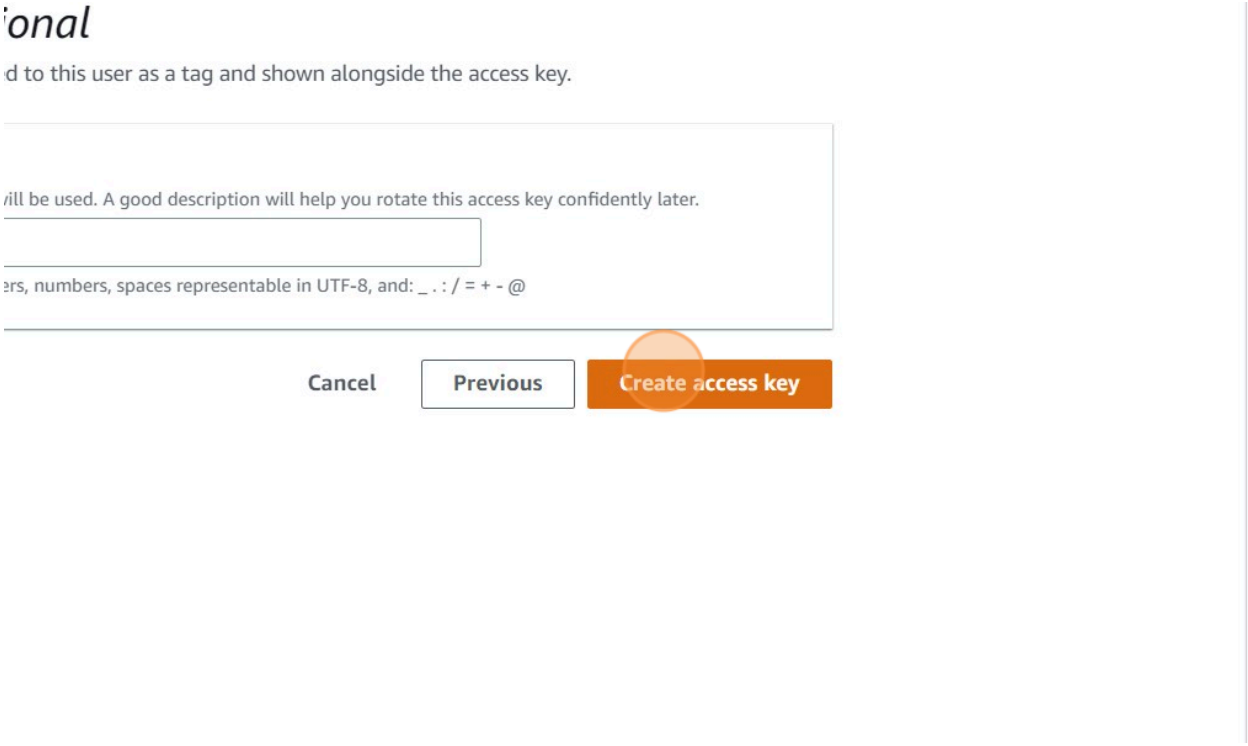
Agree to terms



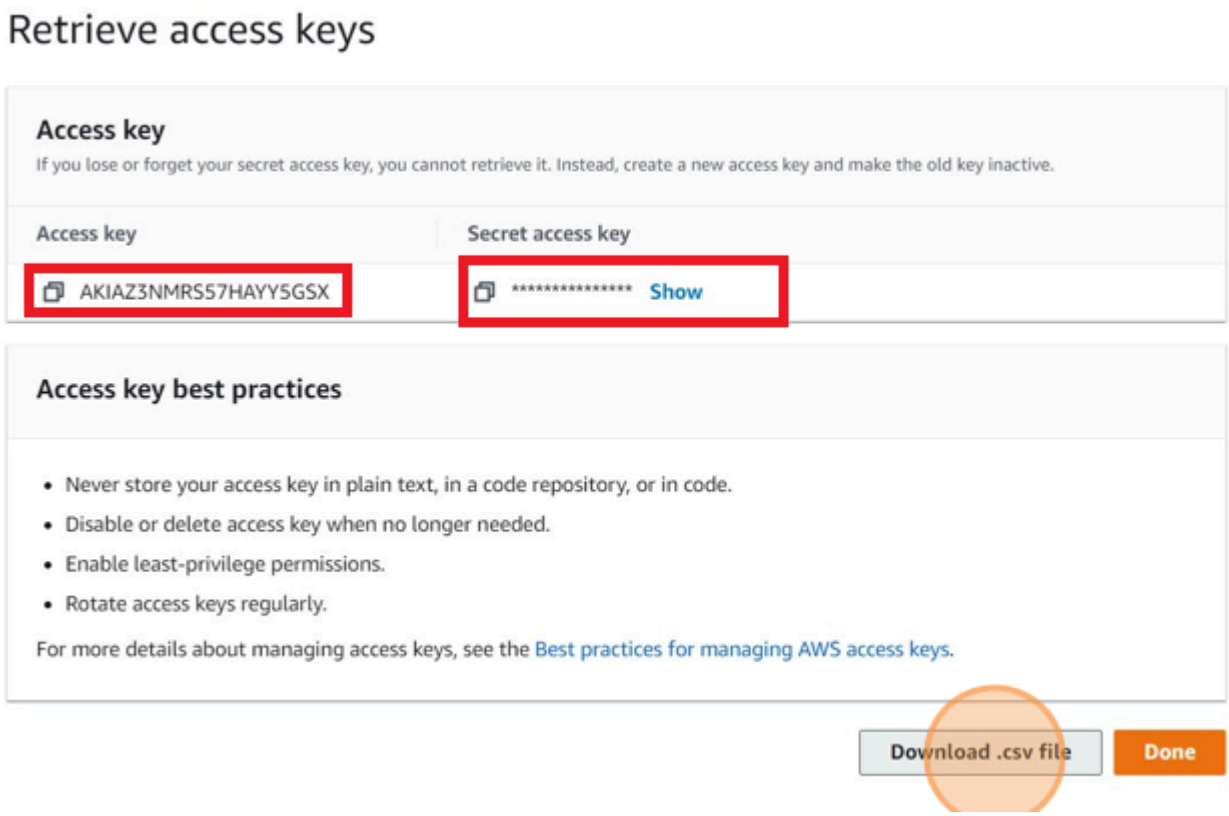
Click Next



Click “Create access key”



Download .csv file

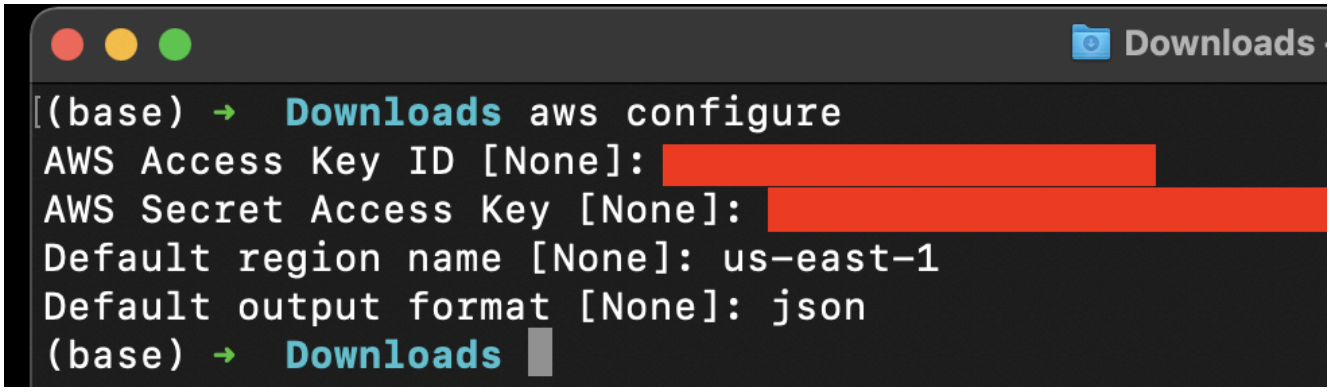


Step2: Aws Configure

Go to vs code or Cmd your wish

```
aws configure
```

Provide your Aws Access key and Secret Access key



Step3: Terraform files and Provision Jenkins,sonar

main.tf

```
resource "aws_instance" "web" {
  ami                    = "ami-0f5ee92e2d63afc18"   #change ami id for
  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, 3000] : {
      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

```
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

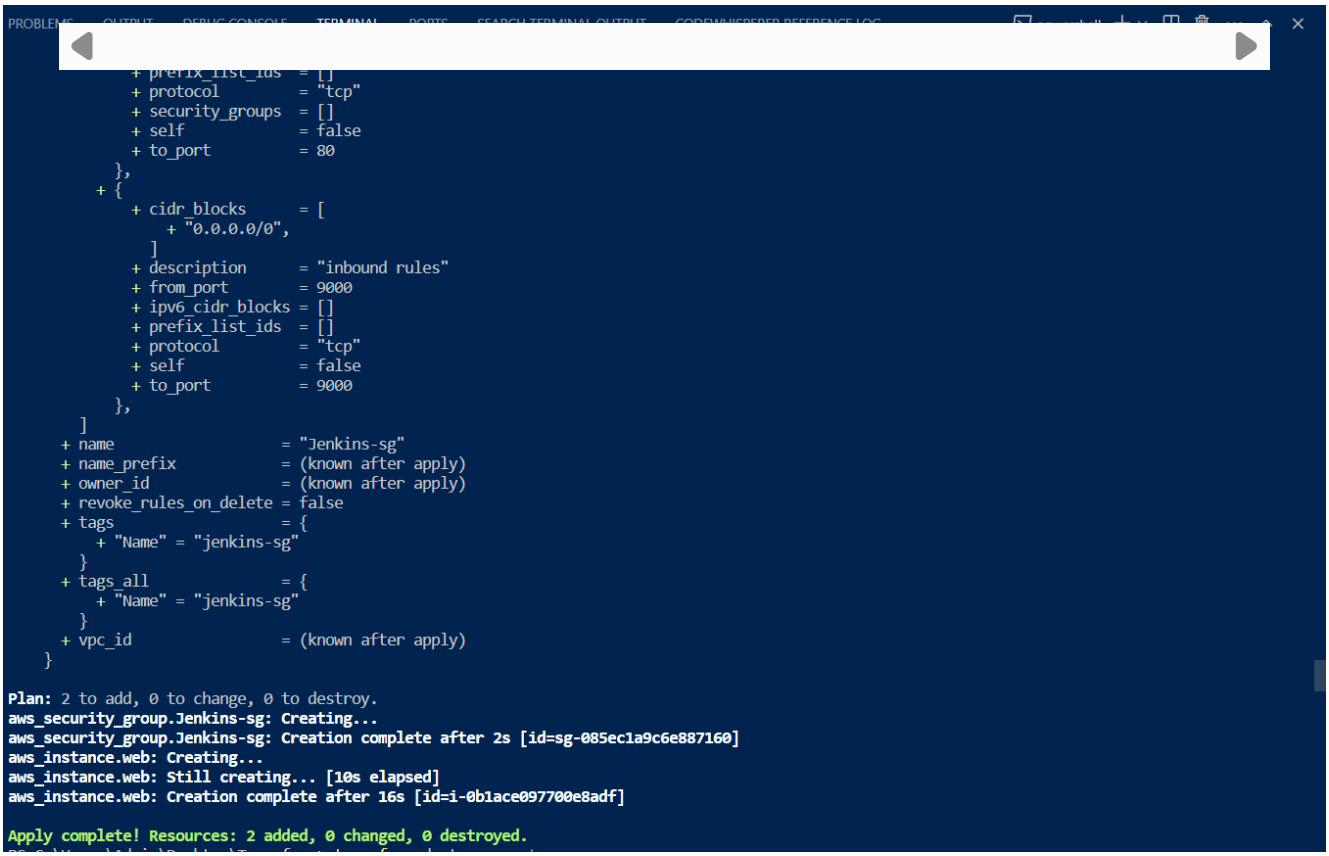
This will install Jenkins and Docker and Sonarqube and trivy

```
#!/bin/bash
sudo apt update -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.ad
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 | su
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg
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 | gp
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity
sudo apt-get update
sudo apt-get install trivy -y
```

Terraform commands to provision

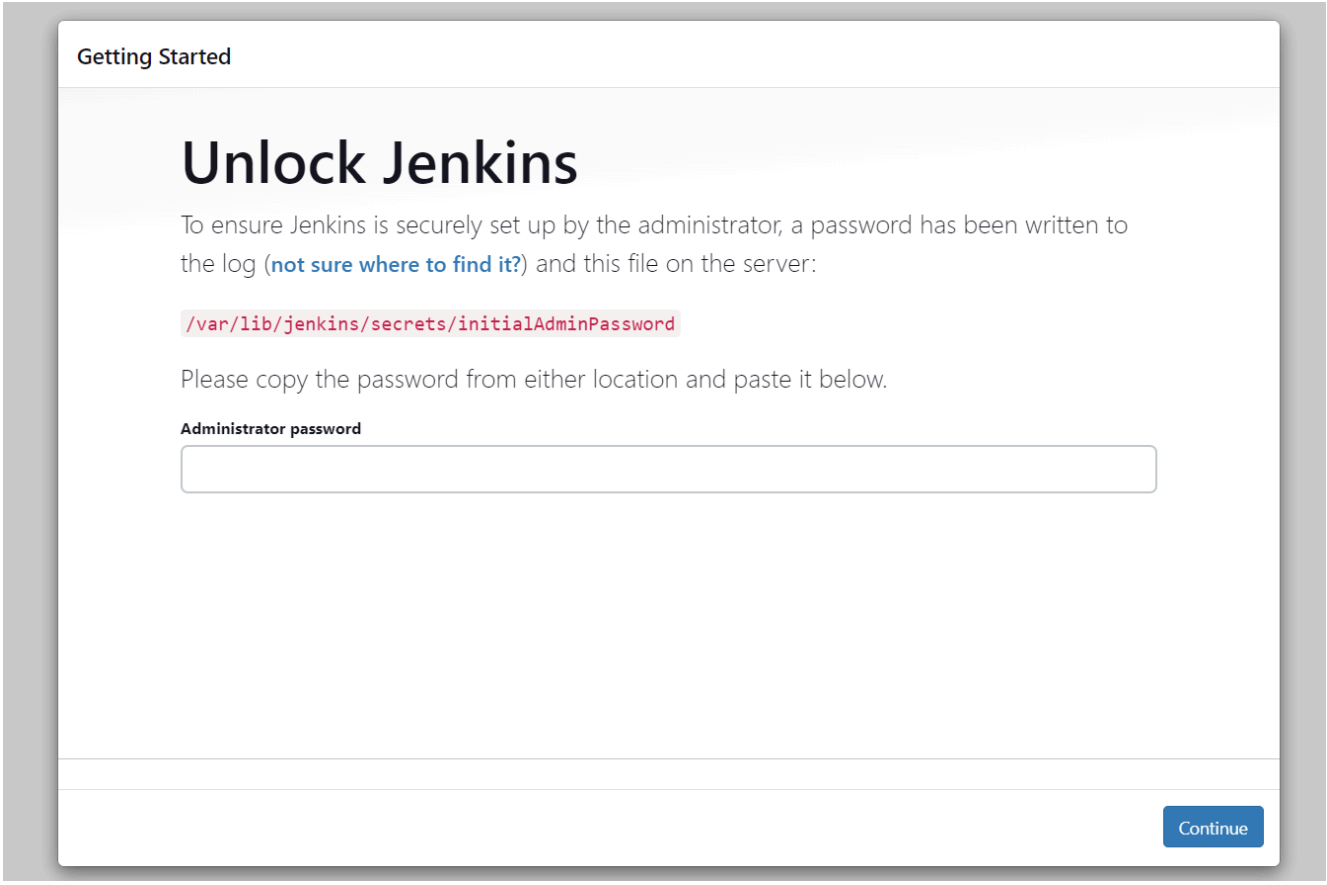
```
terraform init
terraform validate
```

```
terraform plan
terraform apply
```



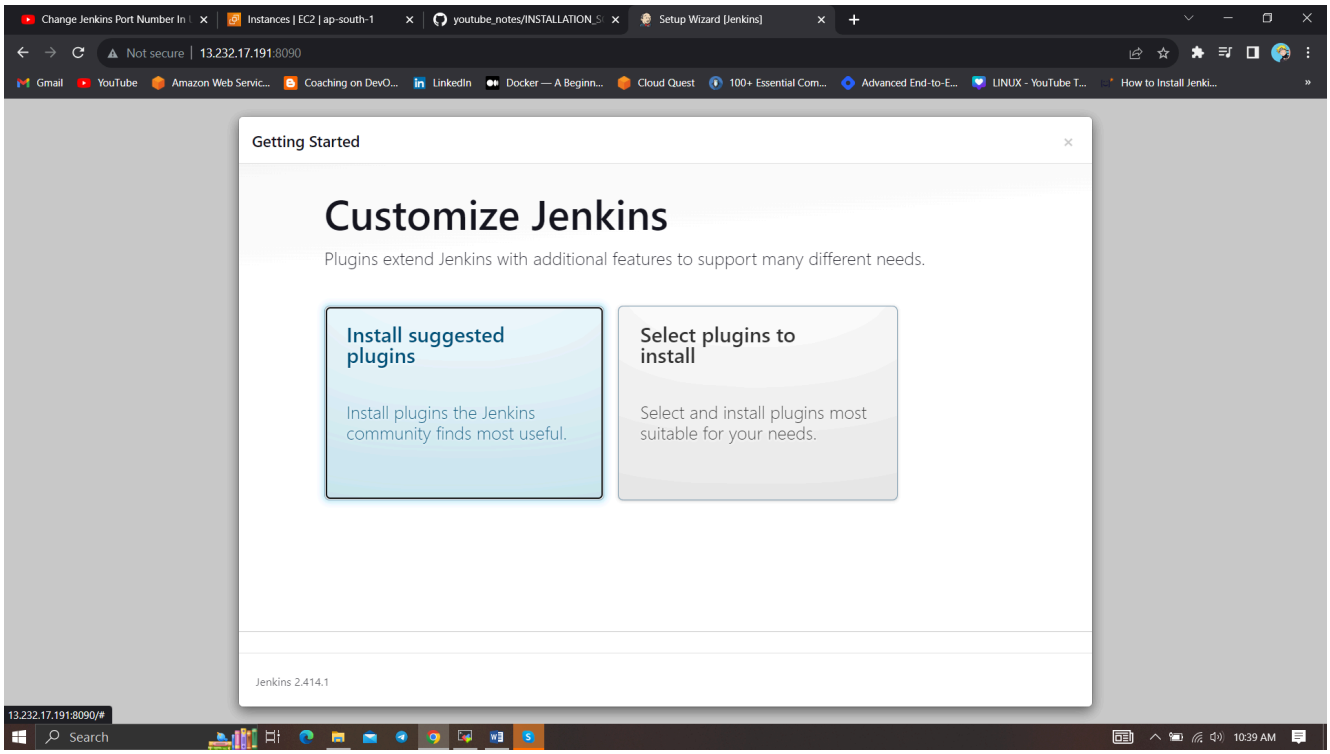
output

```
<instance-ip:8080> #jenkins
```

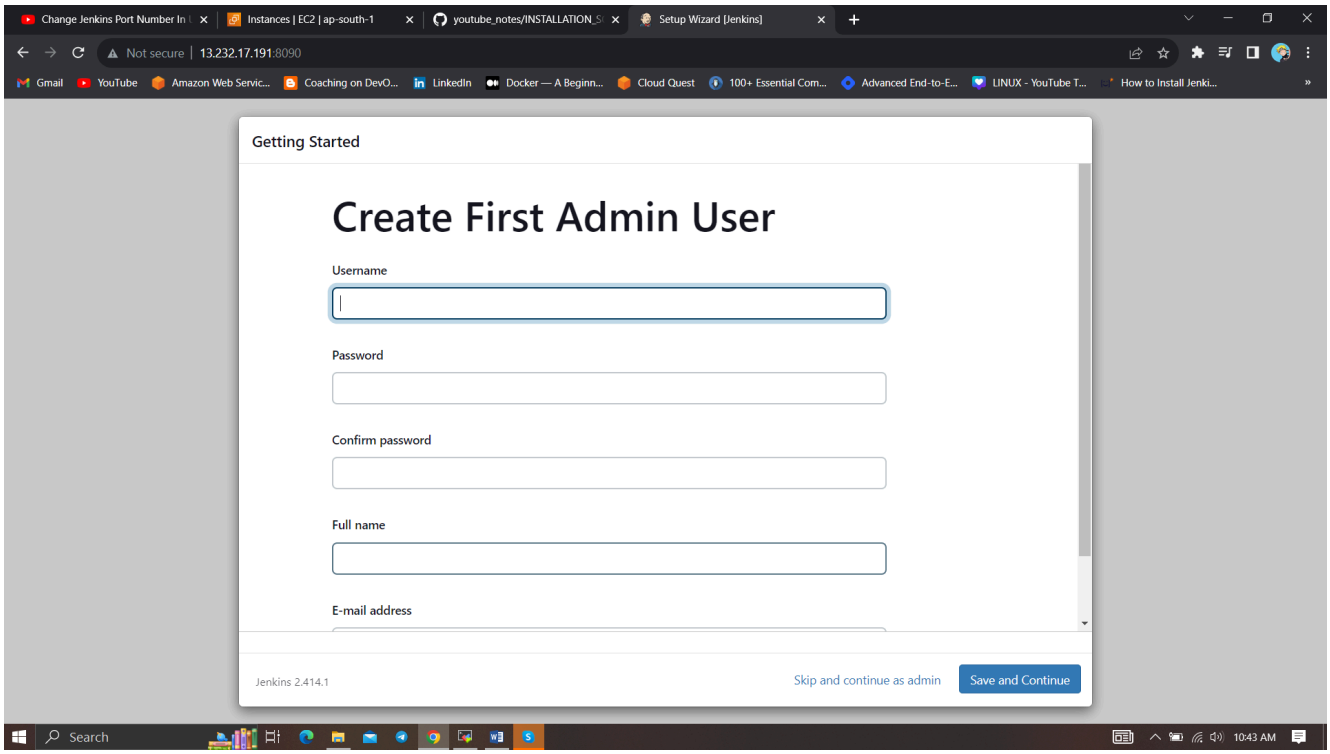



```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Unlock Jenkins using an administrative password and install the suggested plugins.

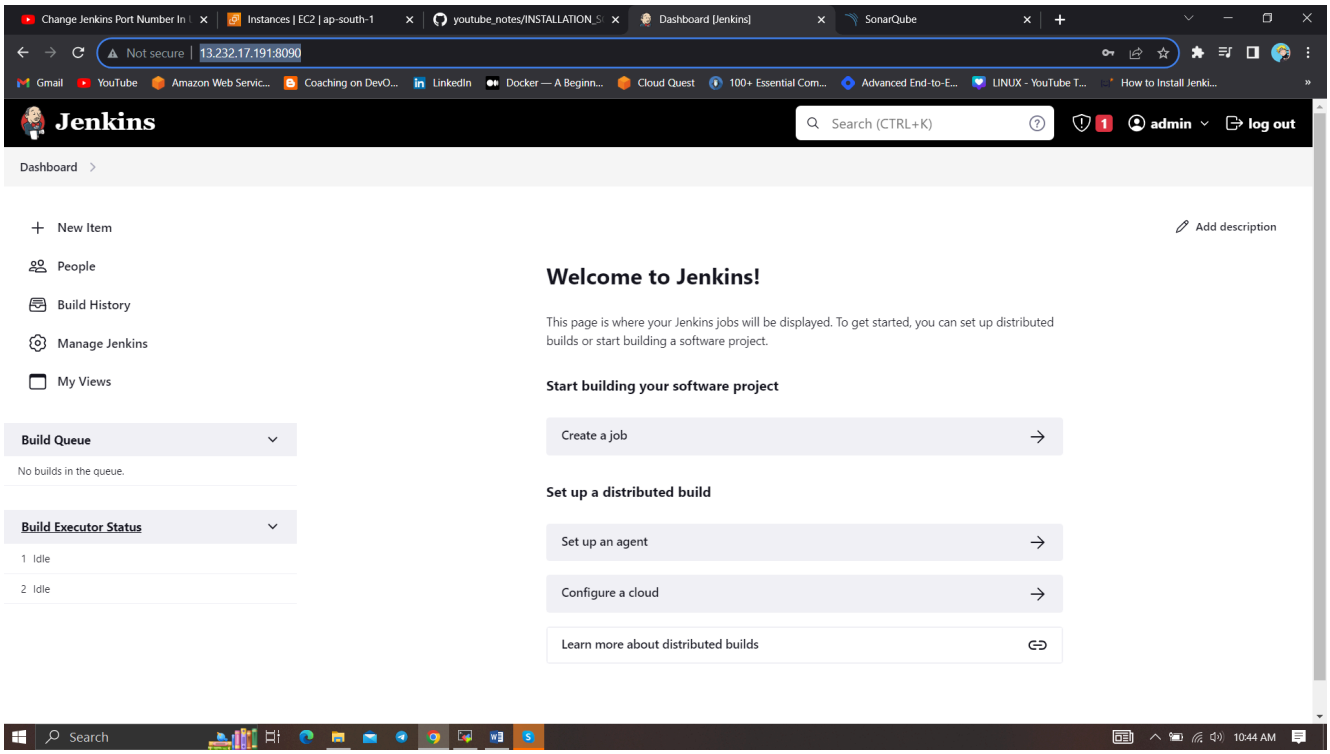


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

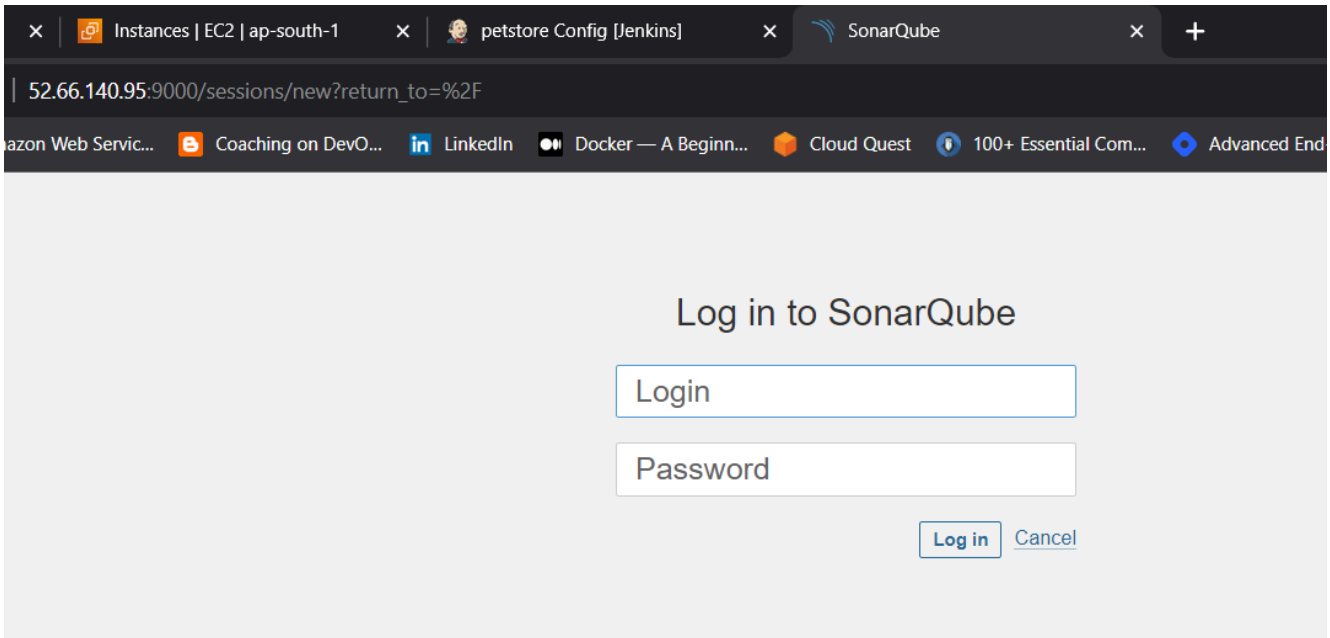
Jenkins Getting Started Screen.



Copy your Public key again and paste it into a new tab

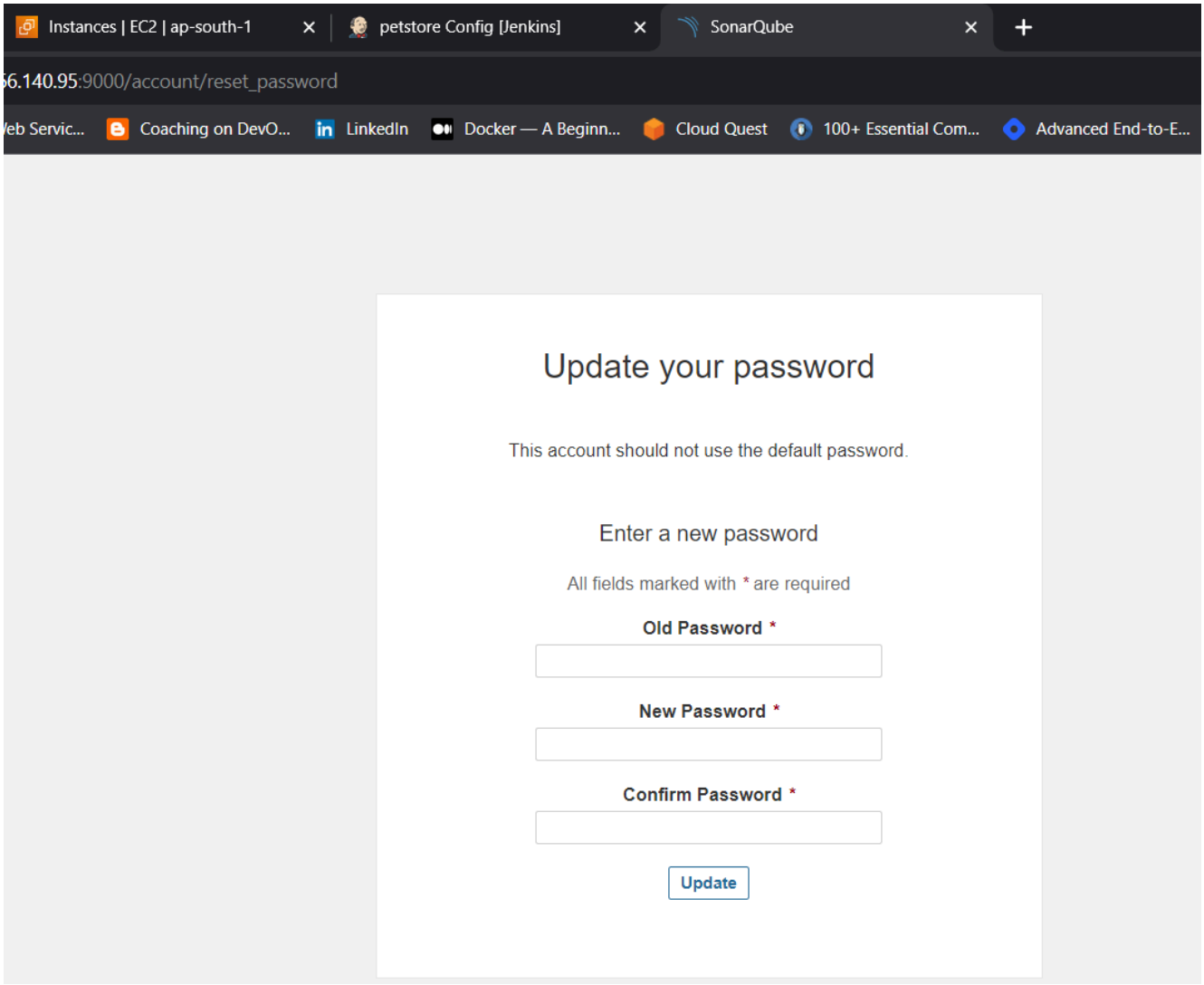
```
<instance-public-ip:9000>
```

Now our sonarqube is up and running

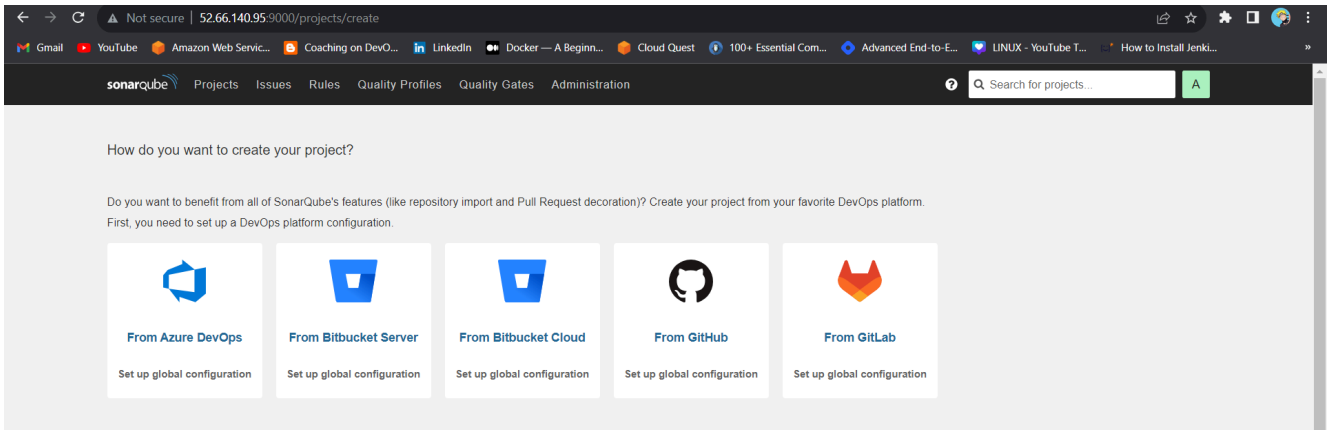


Enter username and password, click on login and change password

```
username admin
password admin
```



Update New password, This is Sonar Dashboard.



Check trivy version

```
trivy --version
```

Step 4 – Install Plugins like JDK, Sonarqube Scanner, NodeJs, OWASP Dependency Check

4A – Install Plugin

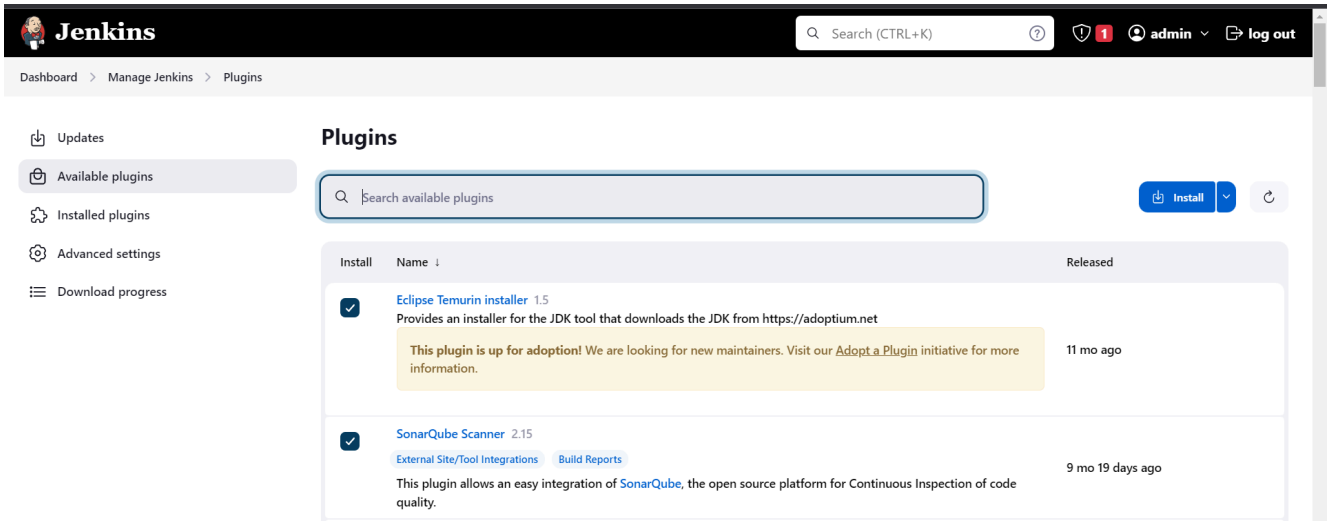
Goto Manage Jenkins →Plugins → Available Plugins →

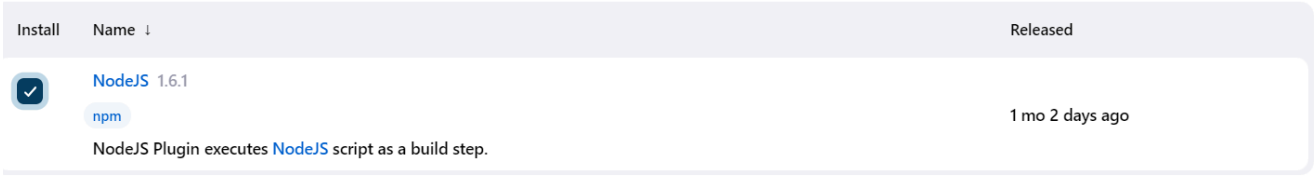
Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)

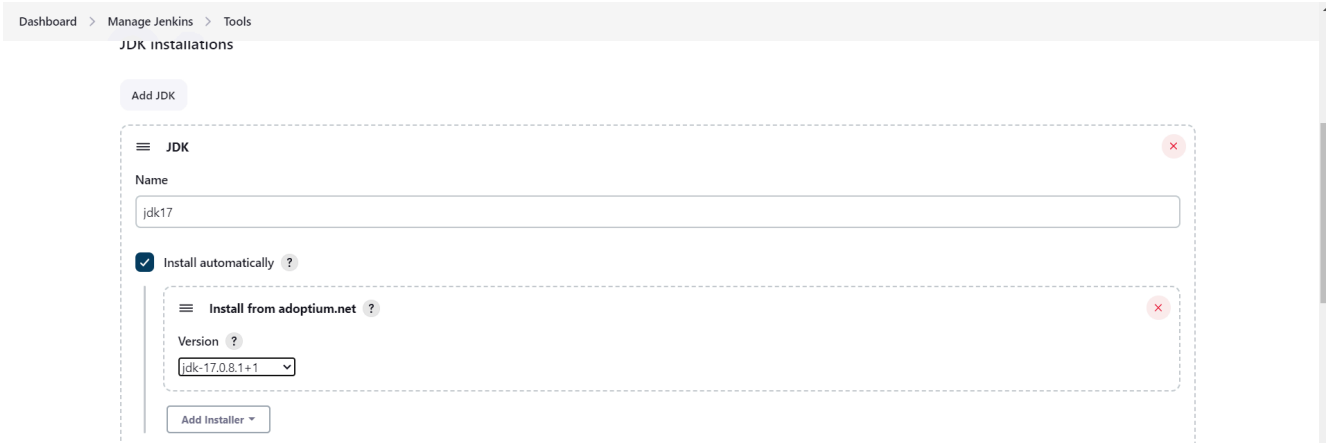
3 → NodeJs Plugin (Install Without restart)





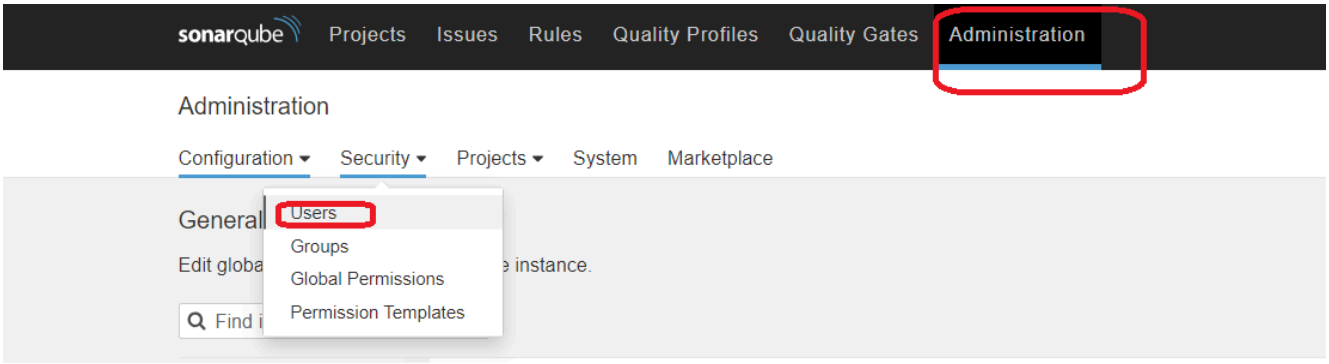
4B – Configure Java and Nodejs in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save

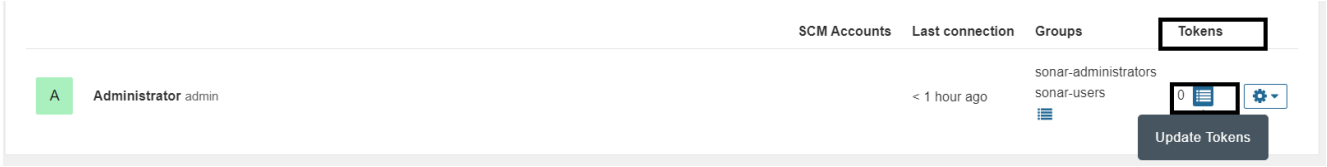


Step 5 – Configure Sonar Server in Manage Jenkins

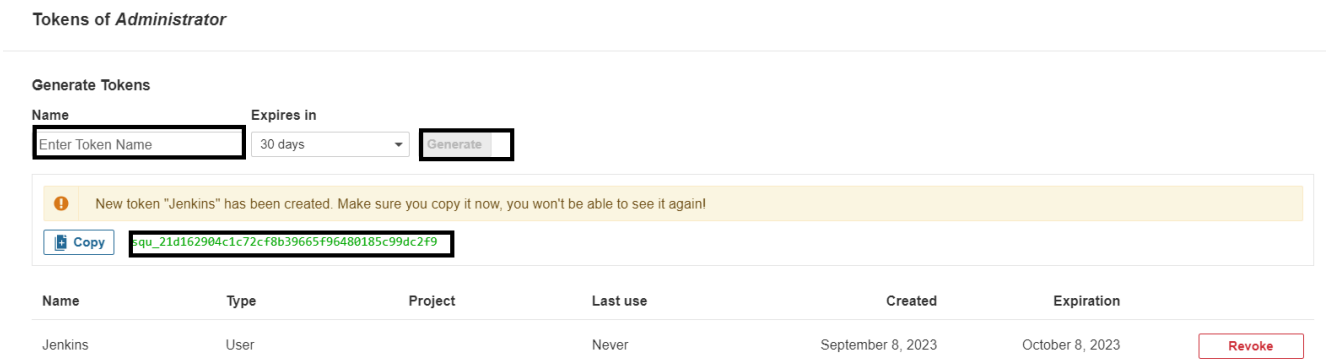
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

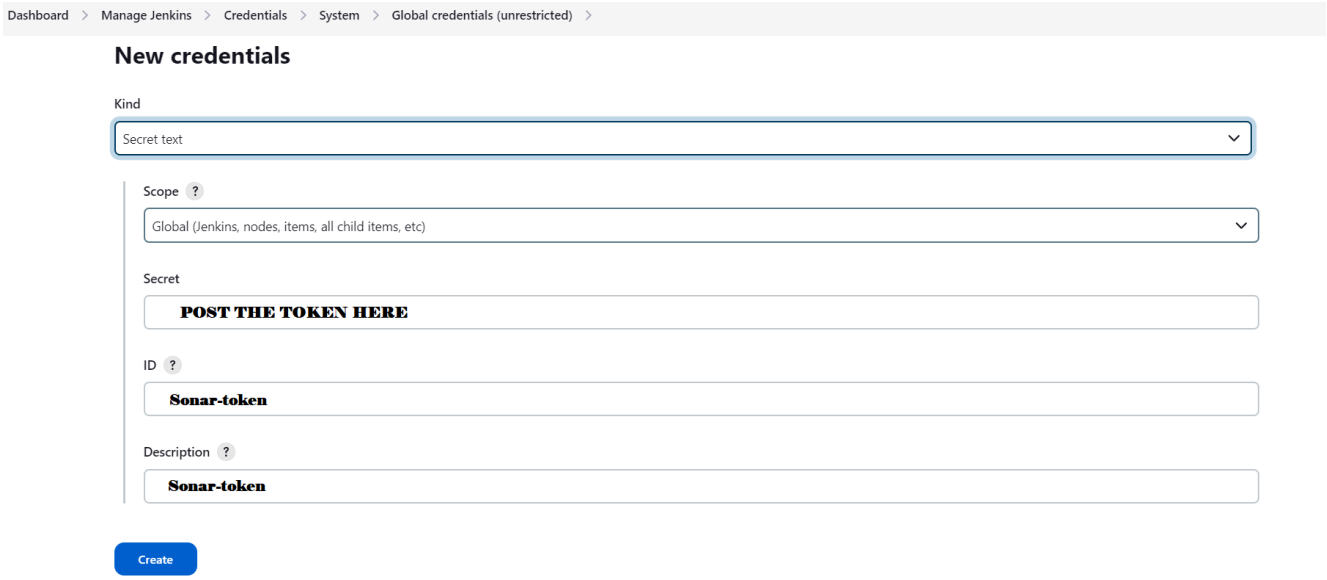


Create a token with a name and generate

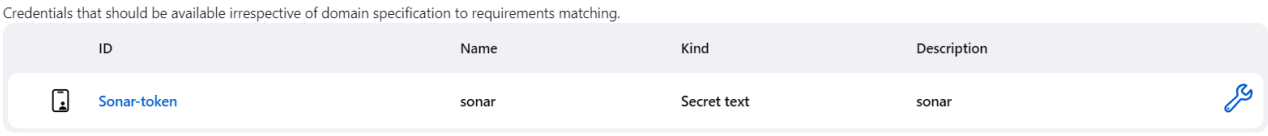


copy Token

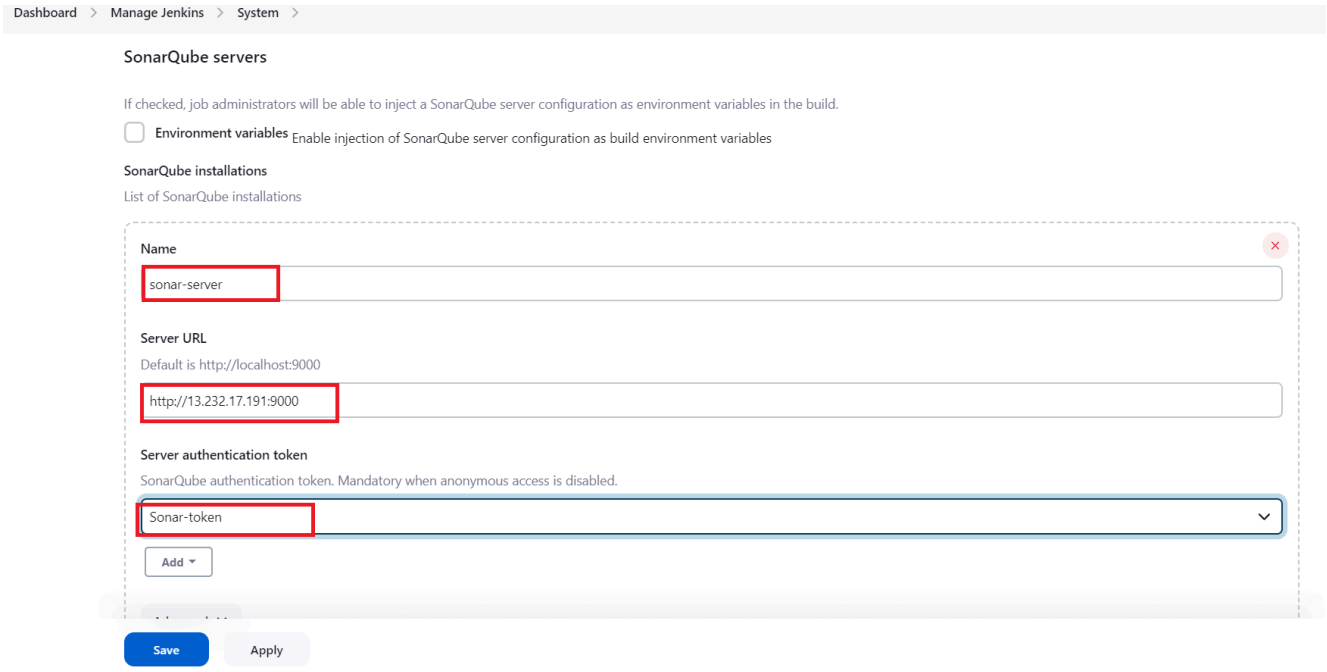
Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this



You will this page once you click on create



Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

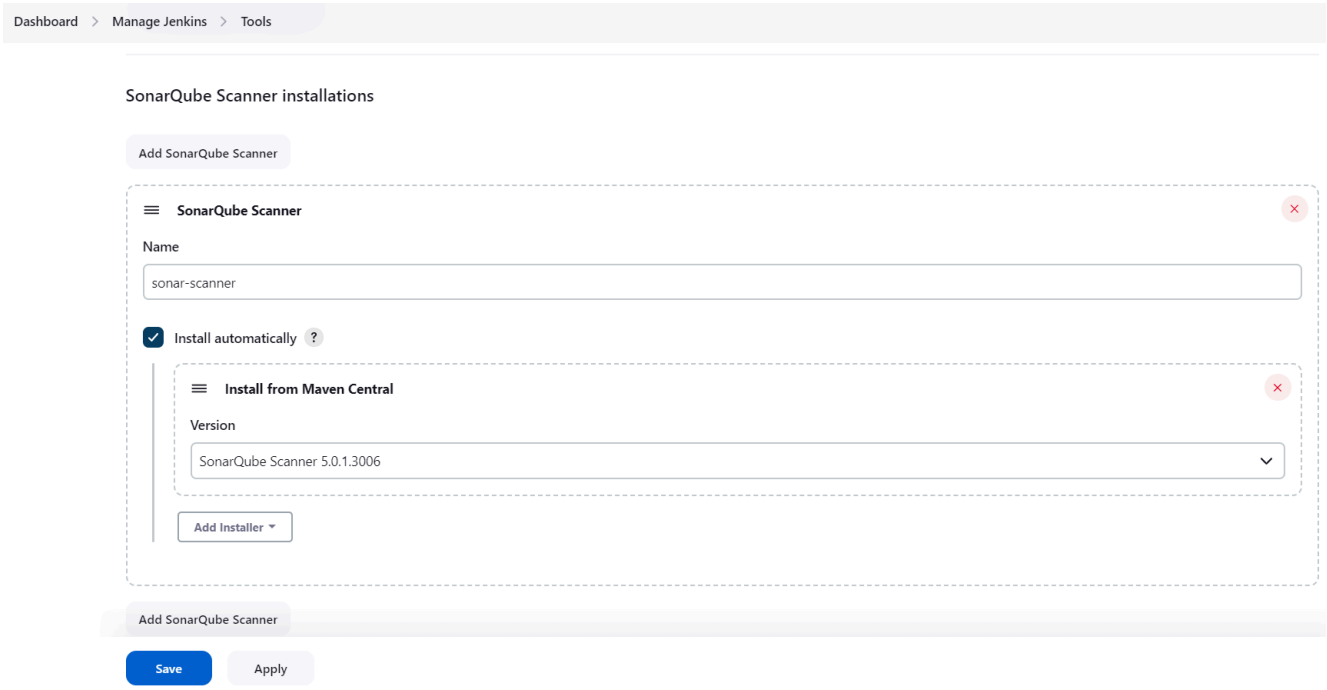


Click on Apply and Save

The Configure System option is used in Jenkins to configure different server

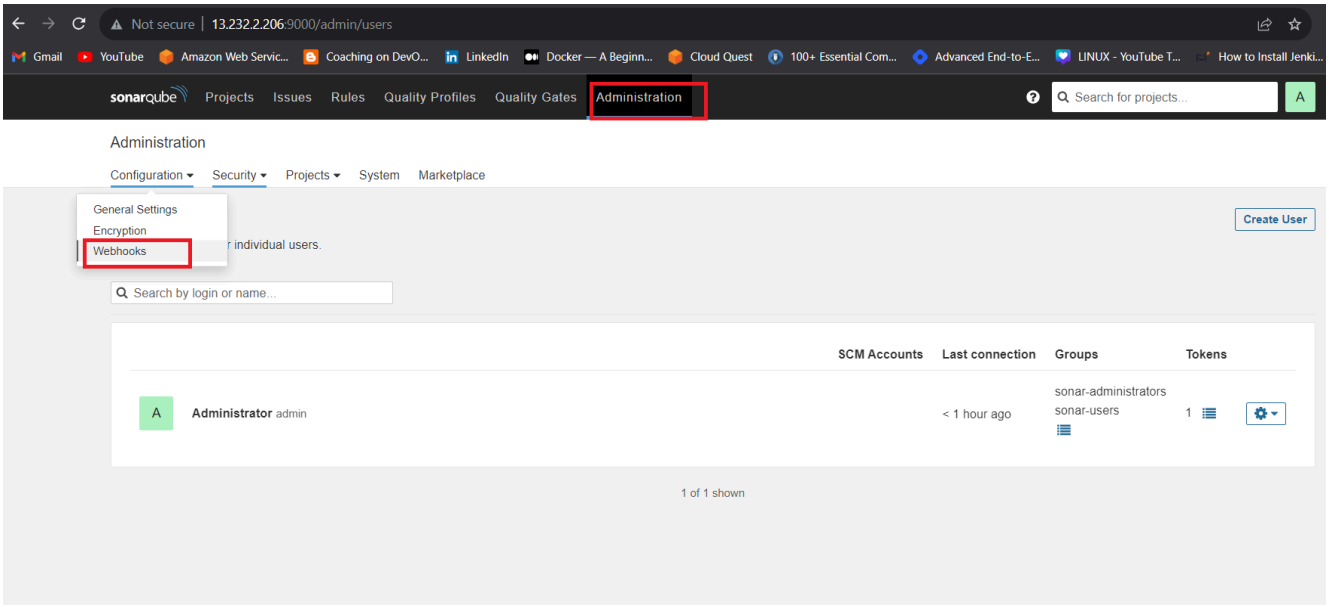
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

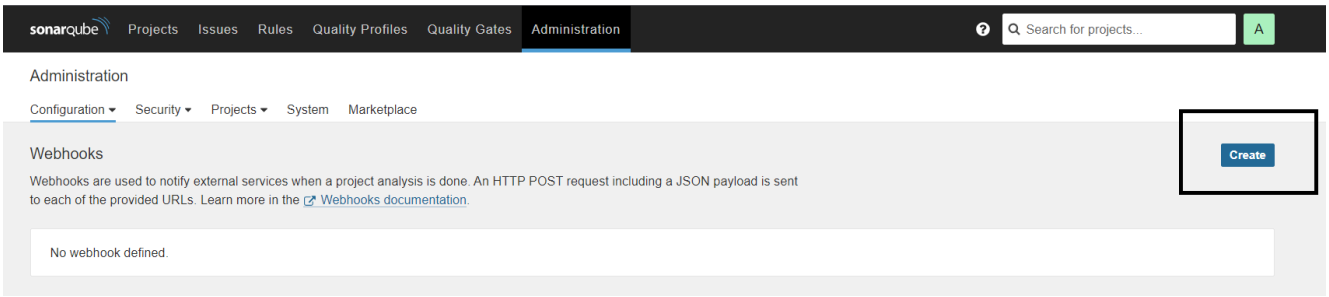


In the Sonarqube Dashboard add a quality gate also

Administration–> Configuration–>Webhooks

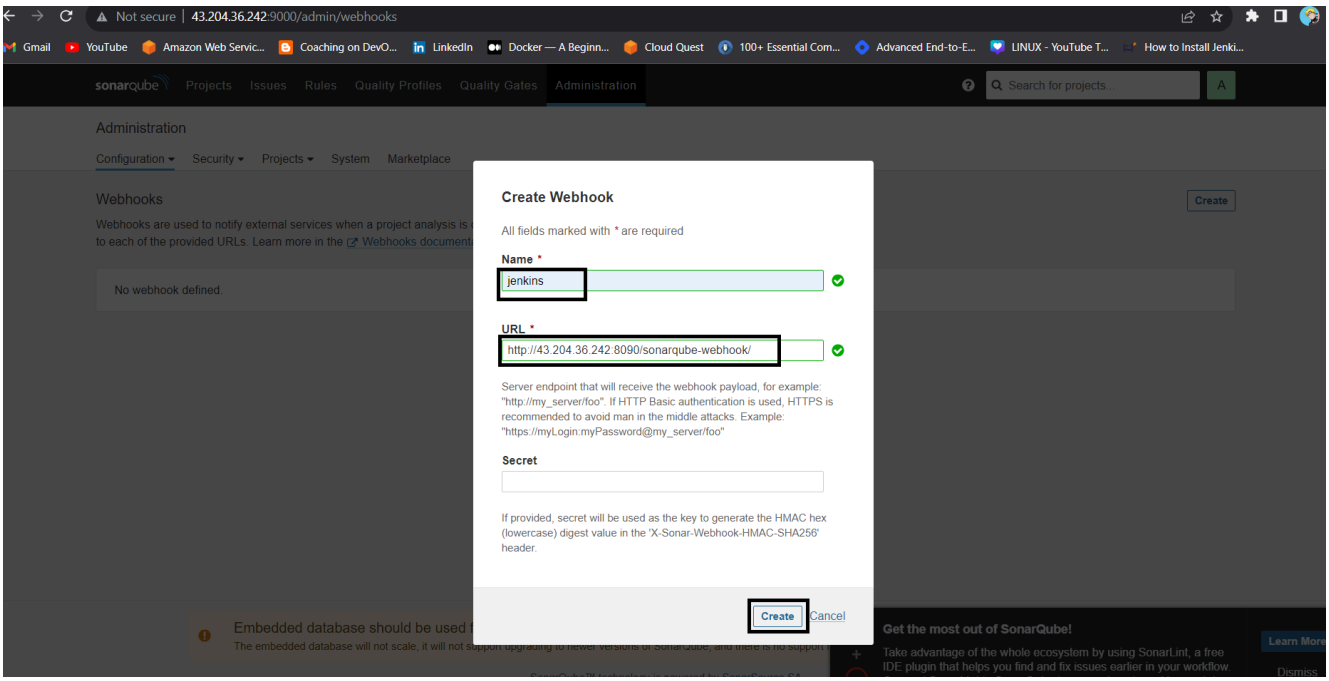


Click on Create



Add details

```
#in url section of quality gate
<http://jenkins-public-ip:8080>/sonarqube-webhook/
```

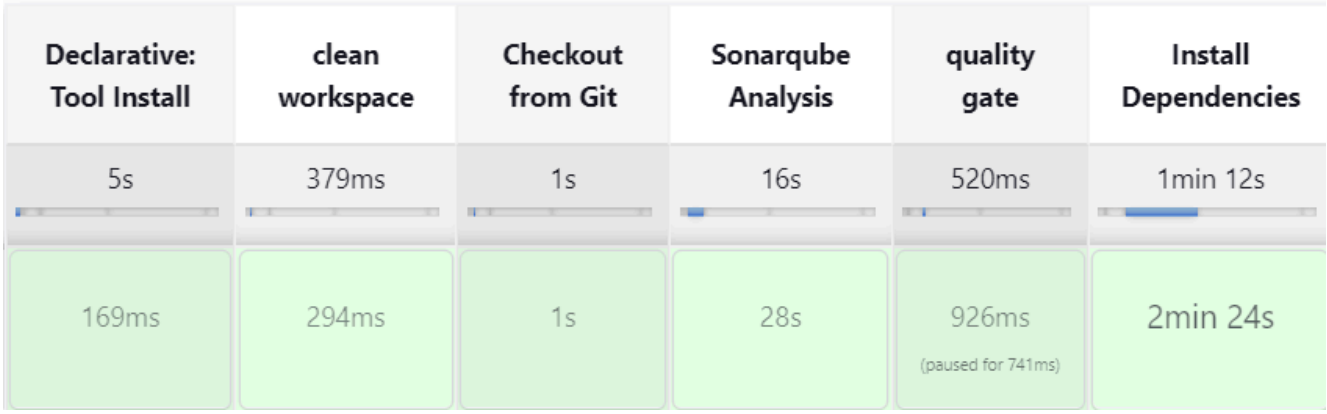


Let's go to our Pipeline and add the script in our Pipeline Script.

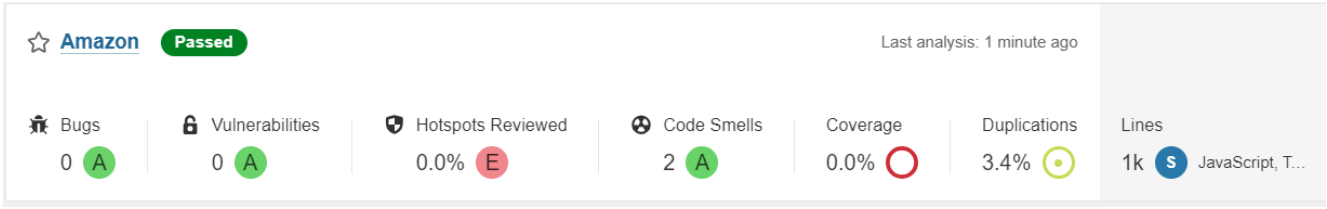
```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
```

```
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
            }
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Aj7Ay/Amazo
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.proje
                    -Dsonar.projectKey=Amazon '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentials
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
            }
        }
    }
}
```

Click on Build now, you will see the stage view like this



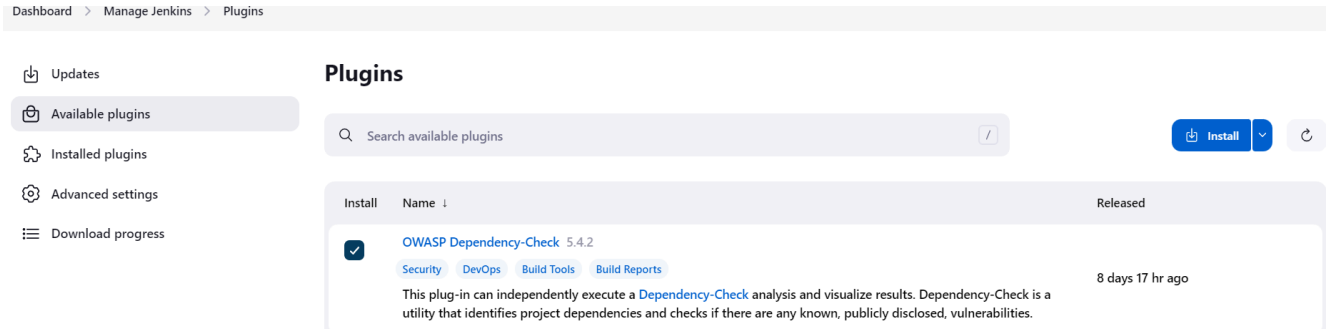
To see the report, you can go to Sonarqube Server and go to Projects.



You can see the report has been generated and the status shows as passed. You can see that there are 1k lines it scanned. To see a detailed report, you can go to issues.

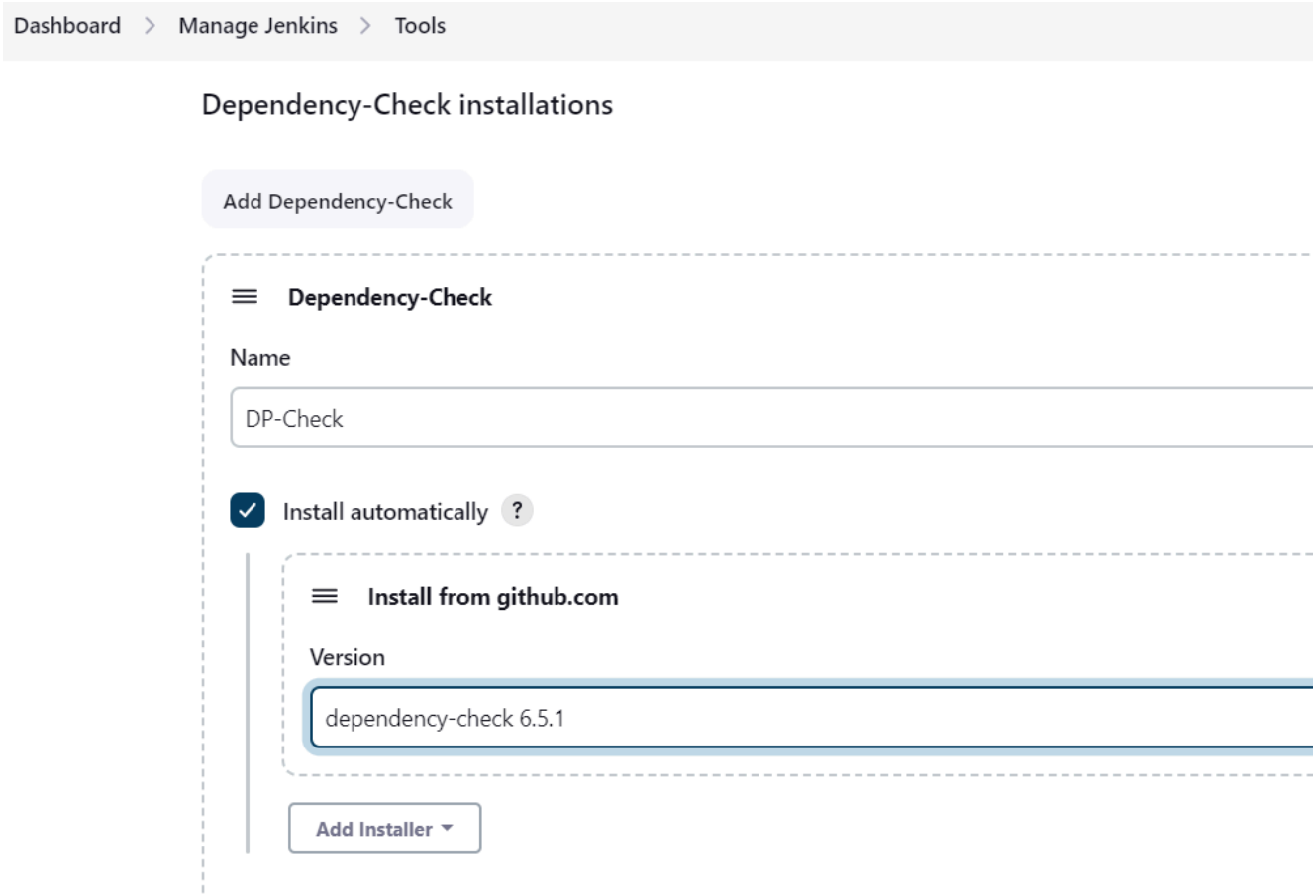
Step 6 – Install OWASP Dependency Check Plugins

GotoDashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install it without restart.



First, we configured the Plugin and next, we had to configure the Tool

Goto Dashboard → Manage Jenkins → Tools →



Click on Apply and Save here.

Now go configure → Pipeline and add this stage to your pipeline and build.

```
stage('OWASP FS SCAN') {
            steps {
                dependencyCheck additionalArguments: '--scan ./ --disab
                dependencyCheckPublisher pattern: '**/dependency-check-
            }
        }
        stage('TRIVY FS SCAN') {
            steps {
                sh "trivy fs . > trivyfs.txt"
            }
        }
```

The stage view would look like this,

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN
5s	379ms	1s	16s	520ms	1min 12s	1min 45s	13s
169ms	294ms	1s	28s	926ms (paused for 741ms)	2min 24s	3min 31s	27s

You will see that in status, a graph will also be generated and Vulnerabilities.

Dependency-Check Results

SEVERITY DISTRIBUTION

5 6 6

Search

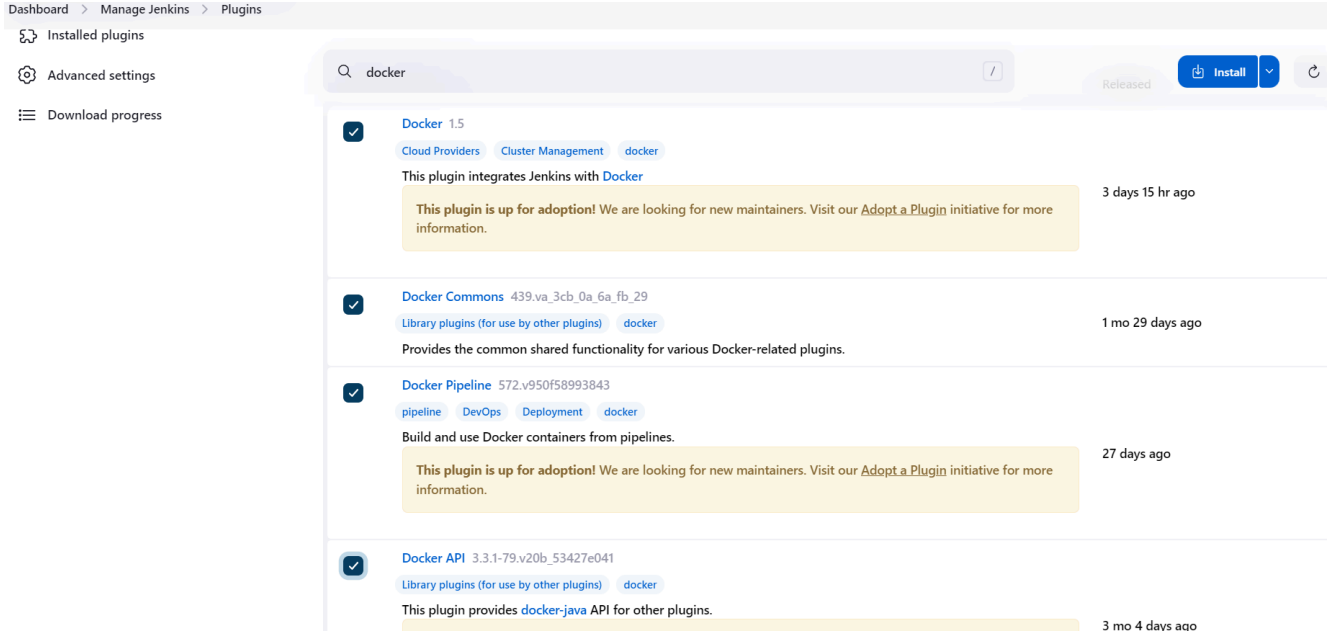
File Name	Vulnerability	Severity	Weakness
axios:1.3.4	OSSINDEX CVE-2023-45857	Medium	CWE-352
css-what:3.4.2	OSSINDEX CVE-2022-21222	High	CWE-1333
ejs:3.1.9	NVD CVE-2023-29827	Critical	CWE-74
jsonpointer:5.0.1	NVD CVE-2022-4742	Critical	CWE-1321
nth-check:1.0.2	NVD CVE-2021-3803	High	CWE-1333
parseurl:1.3.3	NVD CVE-2022-0722	High	CWE-200
parseurl:1.3.3	NVD CVE-2022-2216	Critical	CWE-918
parseurl:1.3.3	NVD CVE-2022-2217	Medium	CWE-79
parseurl:1.3.3	NVD CVE-2022-2218	Medium	CWE-79
parseurl:1.3.3	NVD CVE-2022-2900	Critical	CWE-918

Step 7 – Docker Image Build and Push

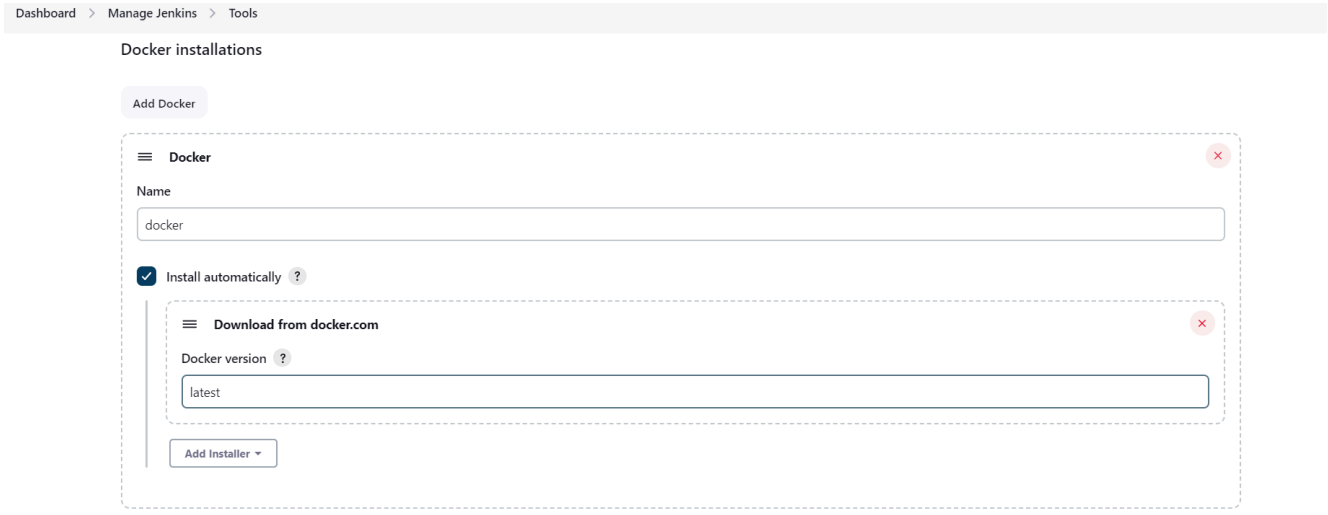
We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

```
Docker

Docker Commons

Docker Pipeline

Docker API

docker-build-step
```

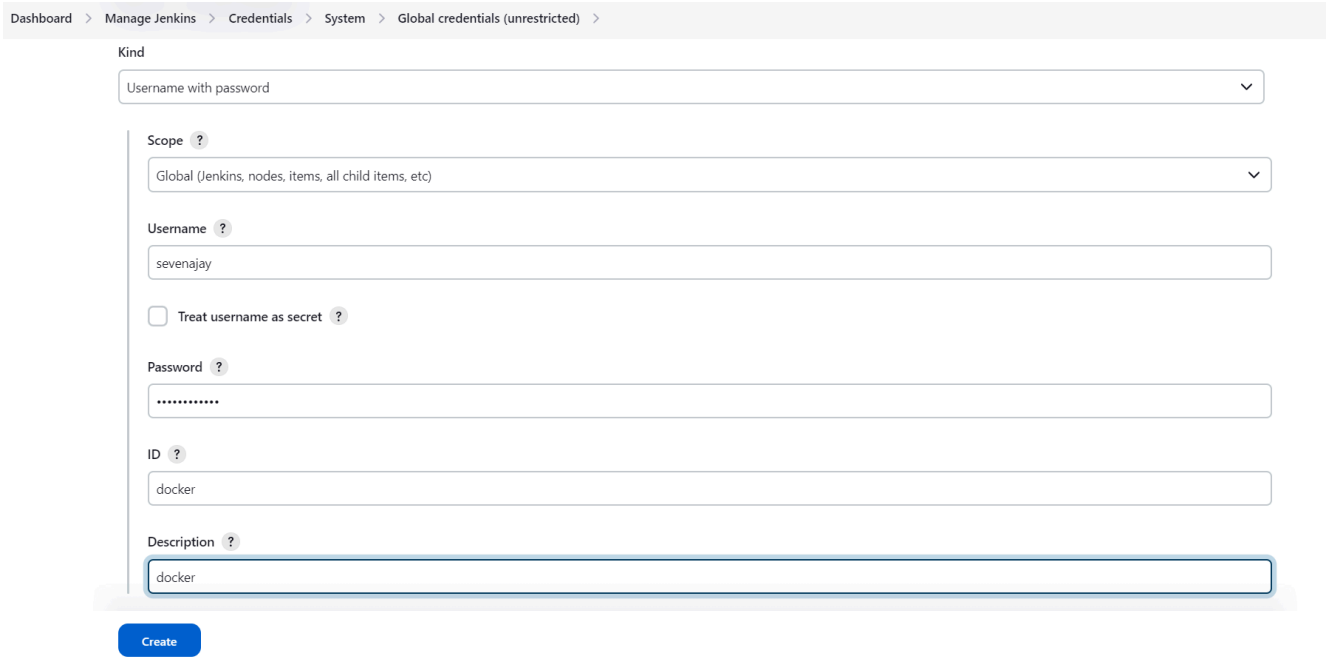
and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



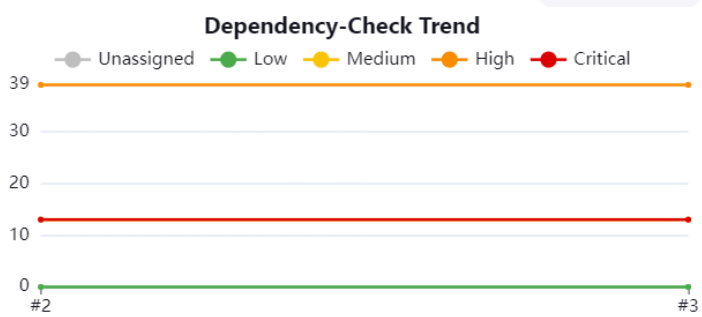
Add DockerHub Username and Password under Global Credentials



Add this stage to Pipeline Script

```
stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName
                       sh "docker build -t amazon ."
                       sh "docker tag amazon sevenajay/amazon:latest "
                       sh "docker push sevenajay/amazon:latest "
                    }
                }
            }
        }
        stage("TRIVY"){
            steps{
                sh "trivy image sevenajay/amazon:latest > trivyimage.tx
            }
        }
```

You will see the output below, with a dependency trend.



Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY
3s	366ms	1s	19s	451ms	1min 20s	2min 1s	16s	3min 9s	4s
154ms	341ms	1s	25s	315ms	1min 36s	2min 31s	23s	3min 9s	4s

When you log in to Dockerhub, you will see a new image is created

Now Run the container to see if the game coming up or not by adding the below stage

```
stage('Deploy to container'){
            steps{
                sh 'docker run -d --name amazon -p 3000:3000 sevenajay/a
            }
        }
```

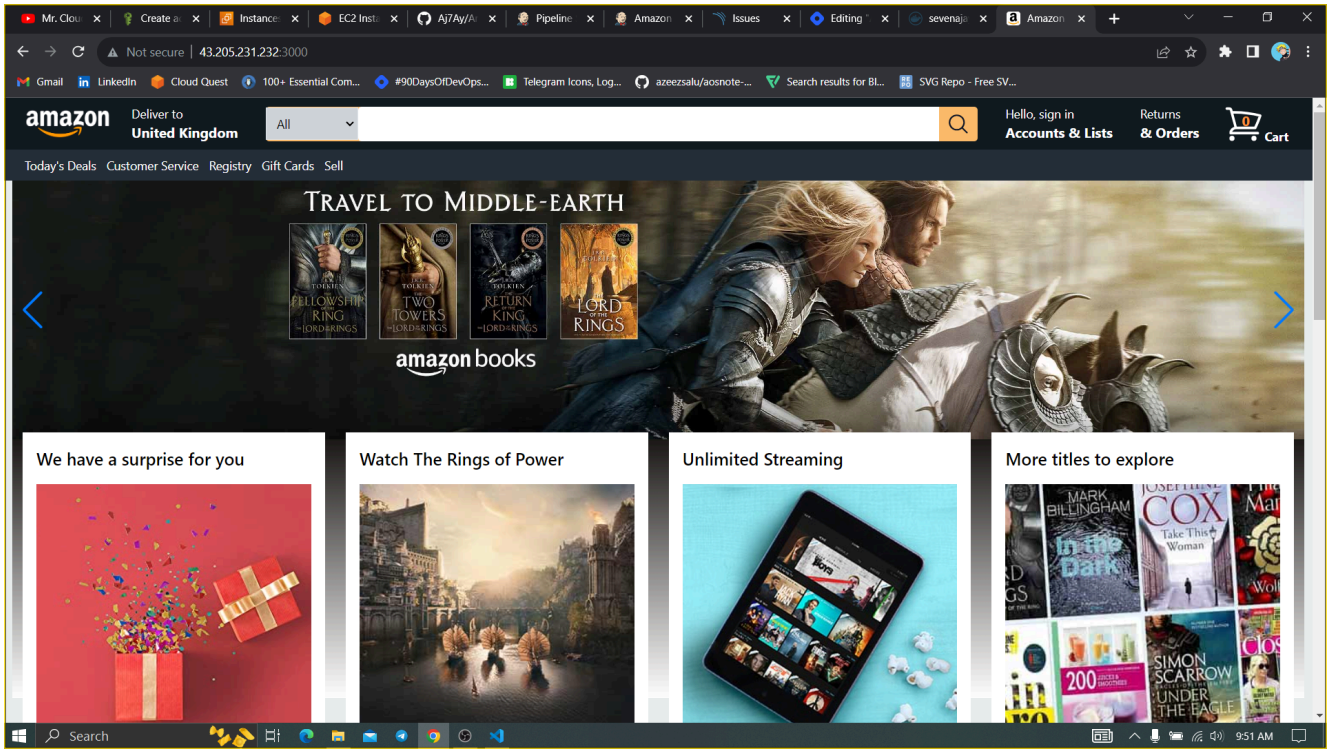
stage view

Amazon - Stage View

	Declarative: Tool Install	CW	GIT CO	SONAR CA	QG	NPM	OWASP FS SCAN	TRIVY FS	DB & Dp	TRIVY IMAGE	AMAZON APP
Average stage times: (Average full run time: ~6min 8s)	133ms	274ms	1s	25s	466ms	24s	4min 40s	24s	3min 26s	2min 7s	1s
#9 Oct 25 09:40 No Changes	135ms	279ms	1s	23s	297ms	17s	2min 51s	23s	3min 26s	2min 7s	1s
#8 Oct 25 09:27 No Changes	136ms	277ms	1s	24s	379ms	17s	6min 29s	26s			
#7 Oct 25 09:22 No Changes	128ms	266ms	1s	28s	723ms (paused for 1s)	37s					

```
<Jenkins-public-ip:3000>
```

You will get this output



Let's destroy everything

Go to VS Code and provide the below command (or) Go to the path where you provisioned the Ec2 instance

```
terraform destroy --auto-approve
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH TERMINAL OUTPUT  CODEWHISPERER REFERENCE LOG

          - {
              - cidr_blocks      = [
                  - "0.0.0.0/0",
                ]
              - description      = "TLS from VPC"
              - from_port        = 9000
              - ipv6_cidr_blocks = []
              - prefix_list_ids  = []
              - protocol         = "tcp"
              - security_groups  = []
              - self             = false
              - to_port          = 9000
            },
        ] -> null
      - name                   = "Jenkins-Security Group" -> null
      - owner_id               = "672618677785" -> null
      - revoke_rules_on_delete = false -> null
      - tags                   = {
          - "Name" = "Jenkins-sg"
        } -> null
      - tags_all               = {
          - "Name" = "Jenkins-sg"
        } -> null
      - vpc_id                 = "vpc-0f6bdd74ced5c07c0" -> null
    }

Plan: 0 to add, 0 to change, 2 to destroy.
aws_instance.web: Destroying... [id=i-03cbca10897fae32d]
aws_instance.web: Still destroying... [id=i-03cbca10897fae32d, 10s elapsed]
aws_instance.web: Still destroying... [id=i-03cbca10897fae32d, 20s elapsed]
aws_instance.web: Destruction complete after 30s
aws_security_group.Jenkins-sg: Destroying... [id=sg-0a1273b4ae042b673]
aws_security_group.Jenkins-sg: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
```

In this age of digital transformation, safeguarding your applications is no longer an option but a necessity. The synergy of Terraform, Jenkins, SonarQube, and Trivy empowers us to not only deploy our applications with speed and efficiency but also to do so with an unwavering focus on security. We hope this guide has been a valuable resource in your journey towards embracing DevSecOps principles and securing your Amazon app deployments on AWS. Remember, the world of technology is ever-evolving, and so are the threats that come with it. Stay vigilant, stay informed, and continue to adapt your DevSecOps practices to stay ahead in the realm of secure and efficient application development. Thank you for joining us, and we wish you every success in your DevSecOps endeavours.

Ajay Kumar Yegireddi is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

Comments

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot

reCAPTCHA
Privacy - Terms

Post Comment

Uncategorized

How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point

22 July 2025

AI

How to Run Docker Model Runner on Ubuntu 24.04

11 July 2025

AI, DevOps

How to Install docker-ai on Ubuntu 24.04

15 June 2025

Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.

Important Links

Privacy Policy

Terms & Conditions

Contact

Resources

Blog

YouTube Channel

 Follow Us on YouTube