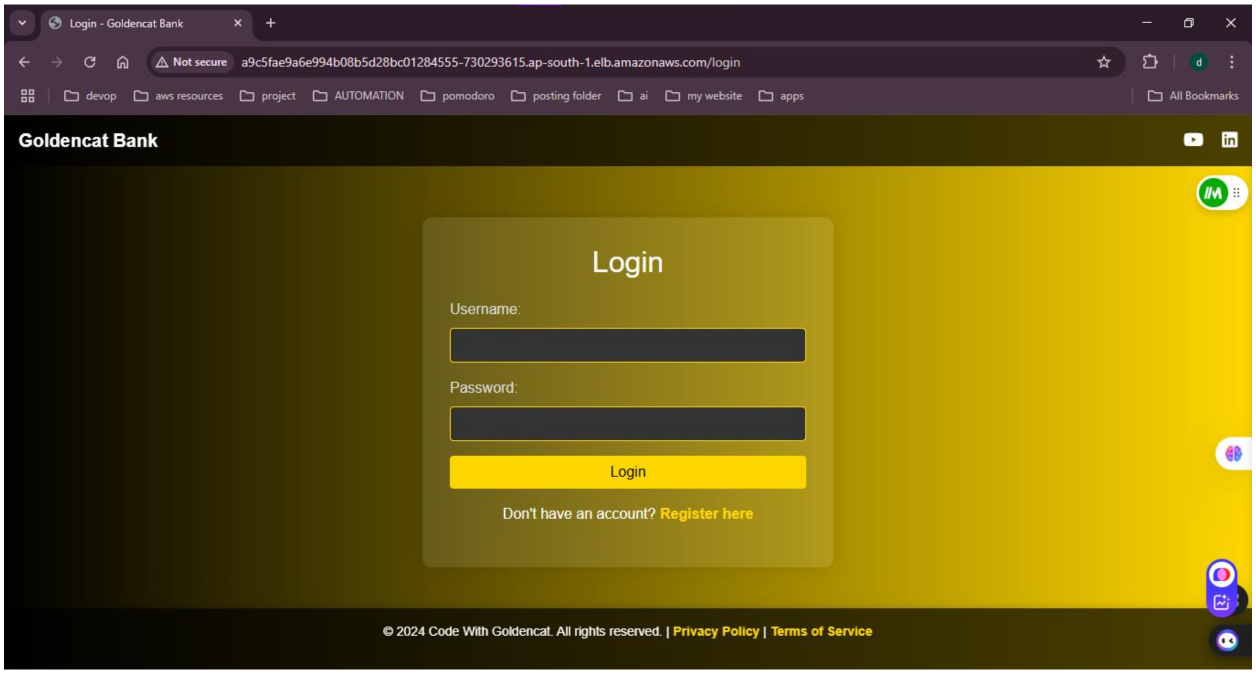
Project-6(Production-level Blue –Green Deployment)

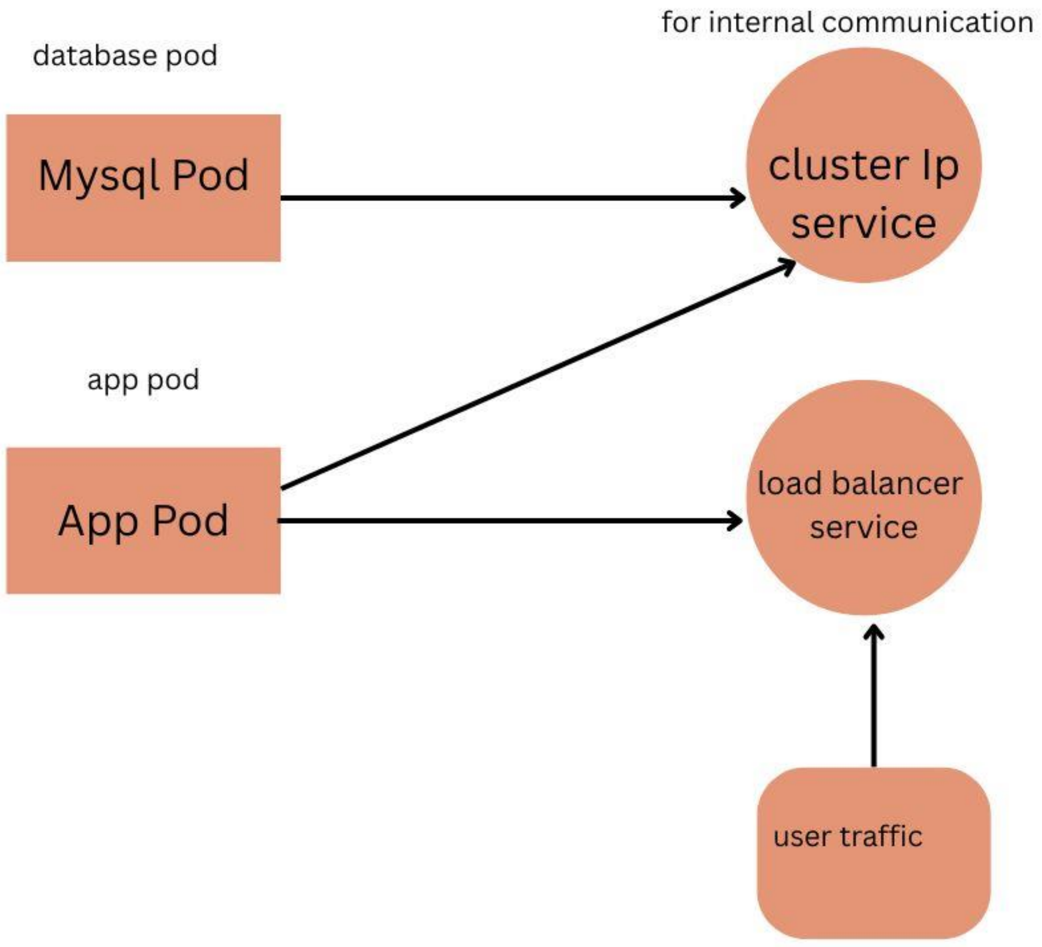

Login - Goldencat Bank
Not secure a9c5fae9a6e994b08b5d28bc01284555-730293615.ap-south-1.elb.amazonaws.com/login
devop
aws resources
project
AUTOMATION
pomodoro
posting folder
ai
my website
apps
All Bookmarks
Goldencat Bank
Login
Username:
Password:
Login
Don't have an account? Register here
© 2024 Code With Goldencat. All rights reserved. | Privacy Policy | Terms of Service

Project based on Blue-Green environment strategy..

Let’s say we have an application which is in environment called as blue environment and then we want to add some upgrade the application with some new features which we can deploy the application in green environment and we can switch the traffic from blue environment to green environment.

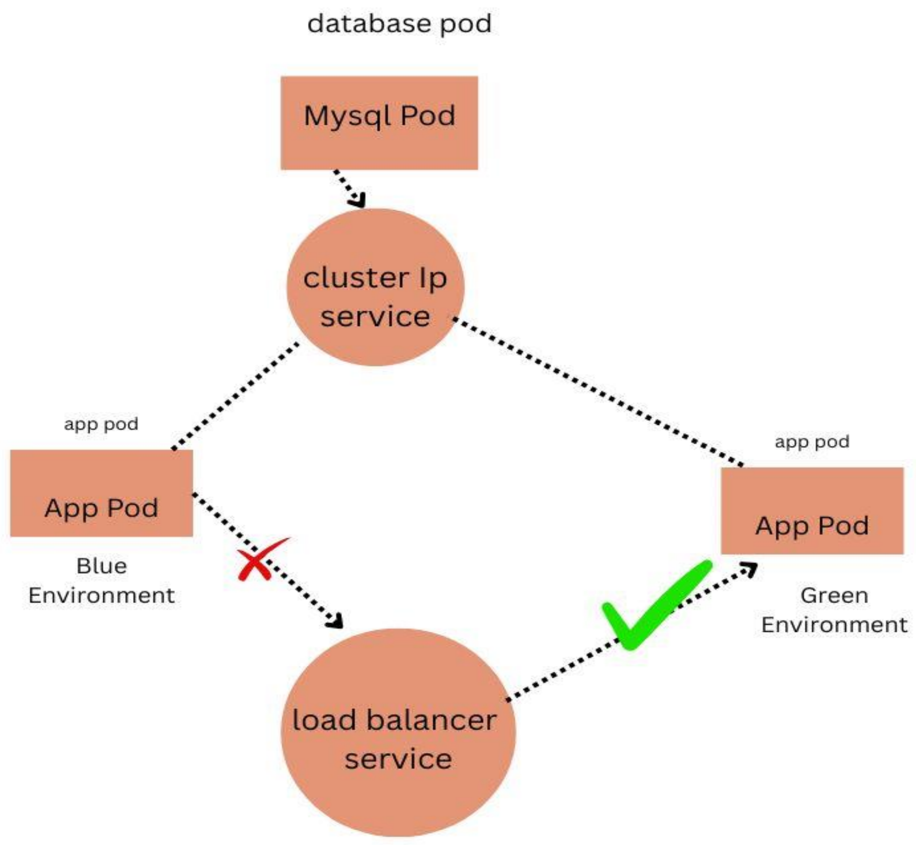
The best part of the blue-green deployment is zero downtime and rollback is very easy.

This application is in MySql database and is written in java, html type//..



(Normal structure)

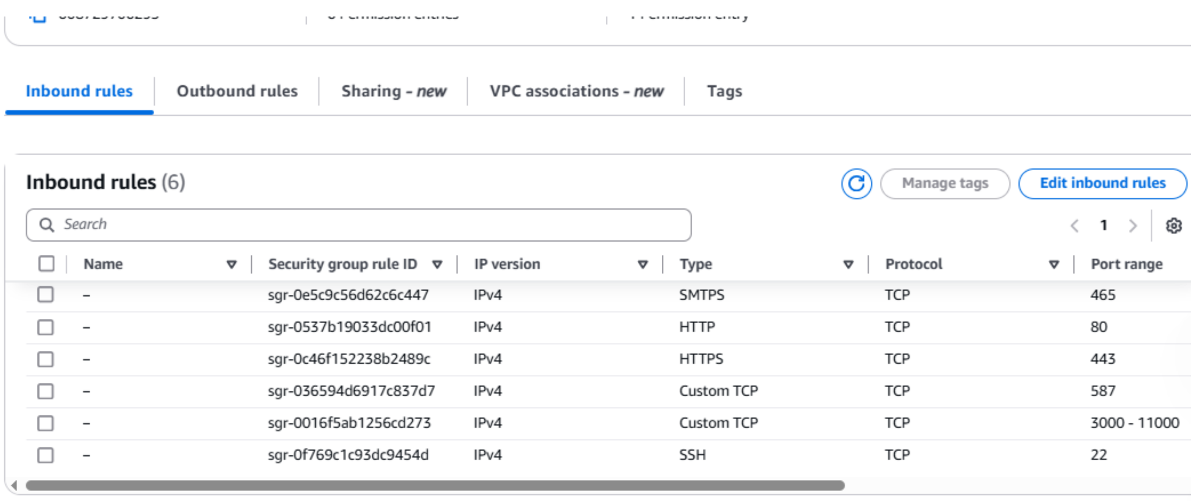
But we are doing this in this project below the picture is shown



Lets start the practical

Ist of all we have to set up our cluster for that we have to create a virtual machine where we can ran all out terraform commands to create a eks cluster.

Ports to be opened



Inbound rules (6)

Name	Security group rule ID	IP version	Type	Protocol	Port range
–	sgr-0e5c9c56d62c6c447	IPv4	SMTPS	TCP	465
–	sgr-0537b19033dc00f01	IPv4	HTTP	TCP	80
–	sgr-0c46f152238b2489c	IPv4	HTTPS	TCP	443
–	sgr-036594d6917c837d7	IPv4	Custom TCP	TCP	587
–	sgr-0016f5ab1256cd273	IPv4	Custom TCP	TCP	3000 - 11000
–	sgr-0f769c1c93dc9454d	IPv4	SSH	TCP	22

VM requirement-

Ubuntu- 24.04 lts

T2 medium

Storage -20 gb with the existing security group

Then launch instance and after launching it go to the cli section and run the command-

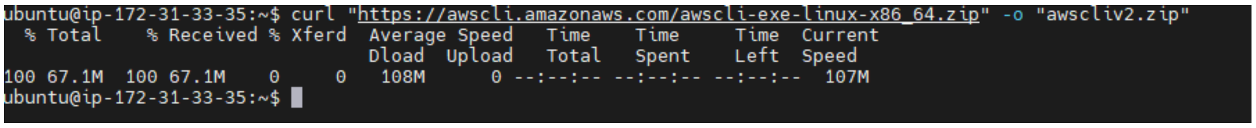
sudo apt update && sudo apt upgrade –y

Then to connect to our aws account with cli

We have to install aws cli tool

For that-

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

  sudo apt install unzip

  unzip awscliv2.zip

  sudo ./aws/install
```



```
ubuntu@ip-172-31-33-35:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 67.1M  100 67.1M    0     0   108M      0 --:--:-- --:--:-- --:--:--  107M
ubuntu@ip-172-31-33-35:~$ 
```

After installing aws cli tool then we have to configure with access key id and security access key ID

By going to our IAM console and create access key and download it and use it

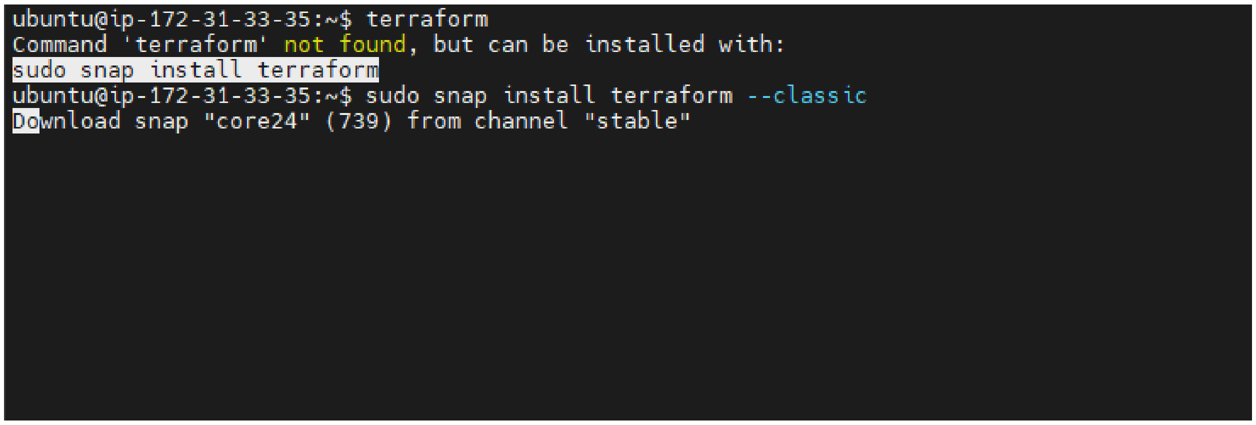
By entering

```
aws configure
```

```
after that enter all your detail as AWS access key ID:

                                   AWS Secret access key ID:

                                    region
```

Next we have to install terraform in this virtual machine because we have to create EKS cluster using terraform for that we have to install terraform

```
sudo snap install terraform --classic
```



```
ubuntu@ip-172-31-33-35:~$ terraform
Command 'terraform' not found, but can be installed with:
sudo snap install terraform
ubuntu@ip-172-31-33-35:~$ sudo snap install terraform --classic
Download snap "core24" (739) from channel "stable"
```

Meanwhile go to the repository and have to clone the repository

After installing terraform

Clone the repository named as-

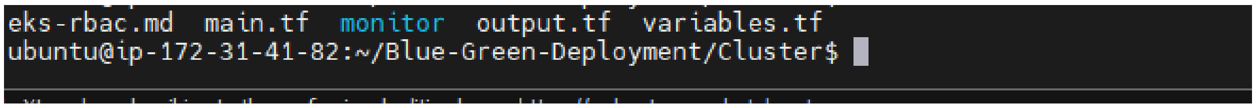
https://github.com/devops-methodology/Blue-Green-Deployment.git

before cloning the repository we have to small change in variable.tf "your private key" and in main.tf you have to change the region if you are outside of india….as well as availability Zone will also change

```
git clone https://github.com/devops-methodology/Blue-Green-Deployment.git
```

(as we are doing it public no need for going to generate token as usual)

Then cd go inside the cluster folder where we have terraform files to be deployed;;;



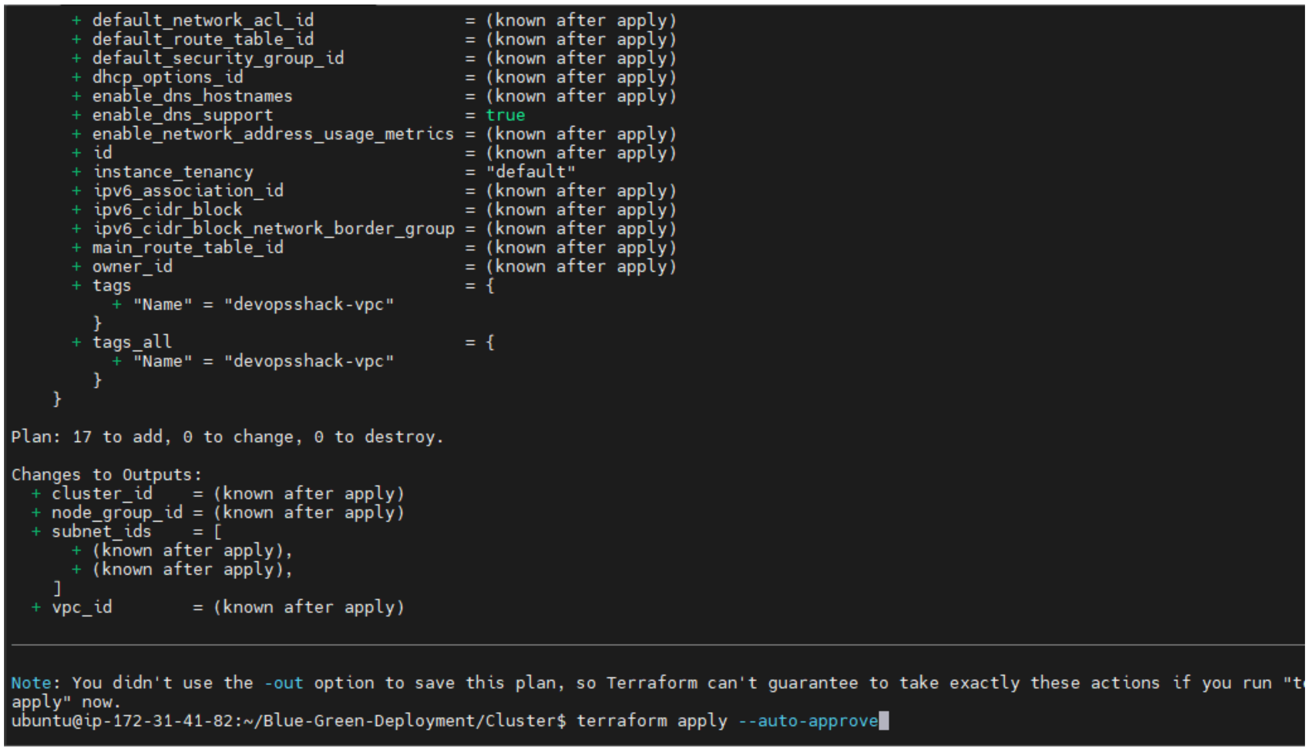
```
eks-rbac.md  main.tf  monitor  output.tf  variables.tf
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$
```

Then go for the list of commands

Terraform init-(initialize for terraform what terraform requires to create the cluster)

Terraform plan-(check and find out what resources to be created to create the cluster) 17 resources will create from scratch

Terraform apply –auto-approve –(create the resources basically the EKS cluster(will take 5 to 10 minutes))



```
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "devopsshack-vpc"
        }
      + tags_all                             = {
          + "Name" = "devopsshack-vpc"
        }
    }

Plan: 17 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + cluster_id    = (known after apply)
  + node_group_id = (known after apply)
  + subnet_ids    = [
      + (known after apply),
      + (known after apply),
    ]
  + vpc_id        = (known after apply)

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

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "t
apply" now.
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ terraform apply --auto-approve
```

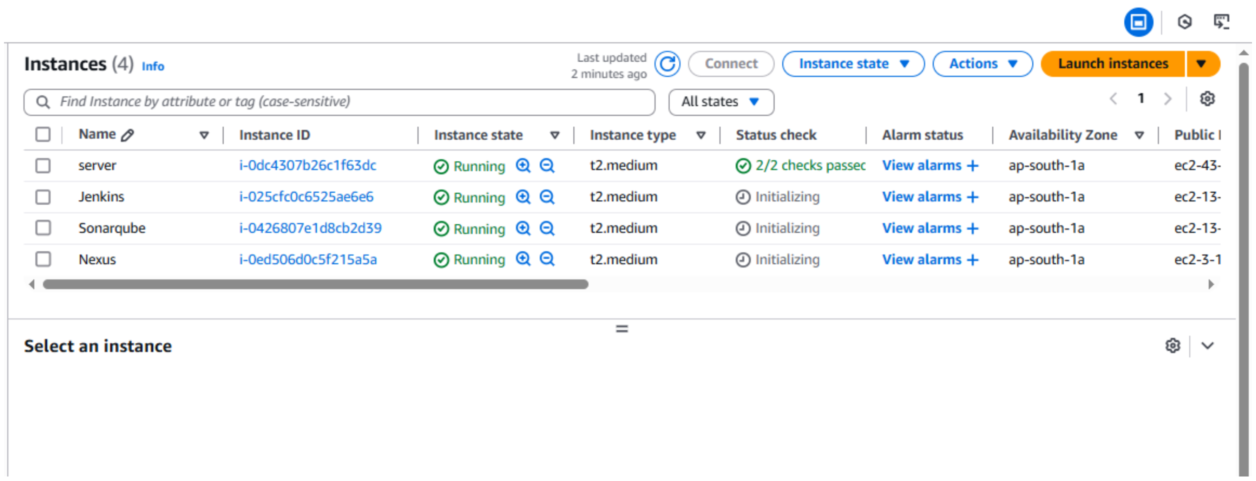
Now we have to create 3 Server

1/Jenkins

2/Nexus

3/SonarQube

Size of the machine will be individually t2 medium,same security group and key with 25 gb Storage



Instances (4) Info

Last updated 2 minutes ago

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public I
server	i-0dc4307b26c1f63dc	Running	t2.medium	2/2 checks passec	View alarms +	ap-south-1a	ec2-43-
Jenkins	i-025cfc0c6525ae6e6	Running	t2.medium	Initializing	View alarms +	ap-south-1a	ec2-13-
Sonarqube	i-0426807e1d8cb2d39	Running	t2.medium	Initializing	View alarms +	ap-south-1a	ec2-13-
Nexus	i-0ed506d0c5f215a5a	Running	t2.medium	Initializing	View alarms +	ap-south-1a	ec2-3-1

For Jenkins installation

In Jenkins server-after connecting through moba -xterm

```
sudo apt update

sudo apt install openjdk-17-jre-headless –y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

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

sudo apt-get install Jenkins

sudo systemctl enable Jenkins

sudo systemctl start  Jenkins
```

After that go for nexus

```
Sudo apt update

#Install docker
```

Because through container we have to create nexus server

For that

```
sudo apt install docker.io –y

sudo chmod –aG docker $user

newgrp docker

docker run –d –name nexus3 –p 8081:8081 sonatype/nexus3
```

after some time go to instance ip:8081

then in nexus server login

userid: admin

and for password you have to go inside of the container

```
docker exec --it <container id> /bin/bash
```

ls

cd sonatype-work/

cd nexus3/

cat admin.password----copy and paste in nexus3 server password area.

Then reset and give your new password

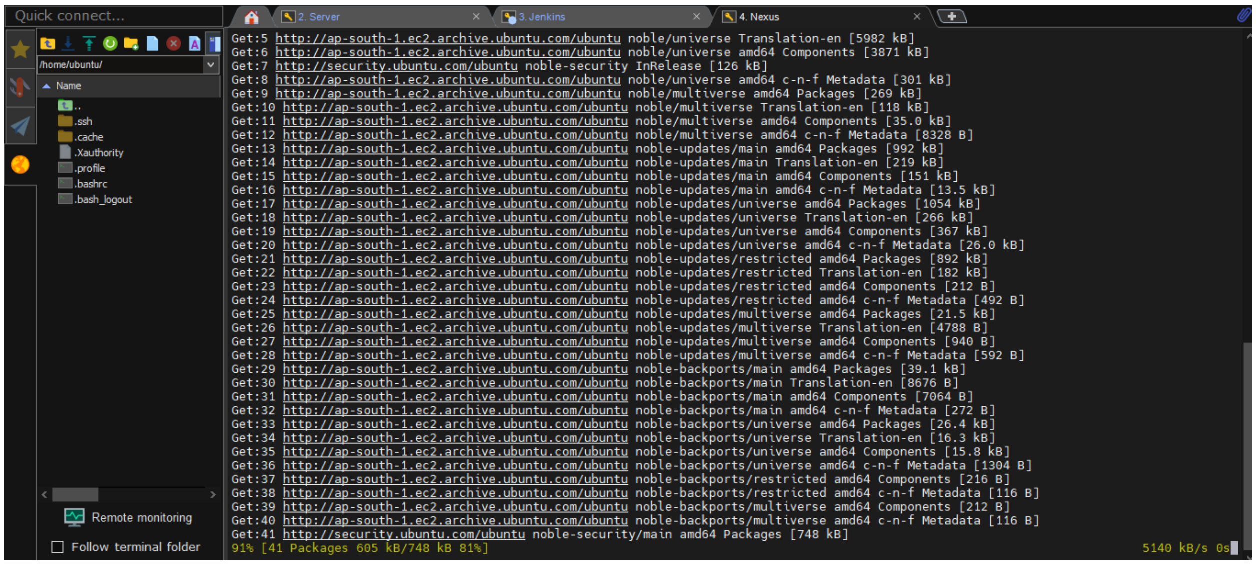
&&&FOR SonarQube

```
sudo apt update

sudo apt install docker.io –y

after installation docker we have to

sudo usermod –aG docker ubuntu

newgrp docker

docker run –d –p 9000:9000 sonarqube:lts-community
```

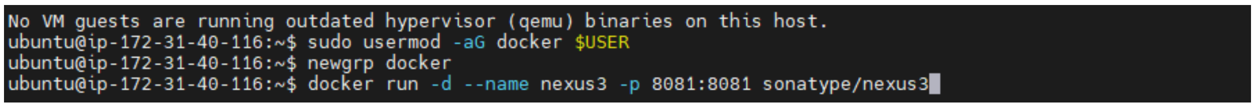
then access sonarqube through sonraqube instanceip:9000

user:admin

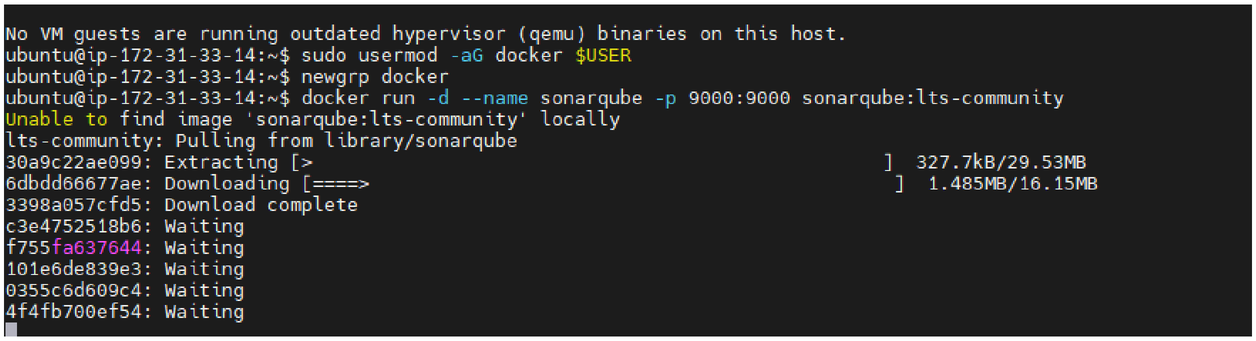
password:admin



```
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [992 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [219 kB]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [13.5 kB]
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1054 kB]
Get:18 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [266 kB]
Get:19 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [367 kB]
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [26.0 kB]
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [892 kB]
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [182 kB]
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:24 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 c-n-f Metadata [492 B]
Get:25 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [21.5 kB]
Get:26 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [4788 B]
Get:27 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:28 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [592 B]
Get:29 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [39.1 kB]
Get:30 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main Translation-en [8676 B]
Get:31 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7064 B]
Get:32 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [272 B]
Get:33 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [26.4 kB]
Get:34 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [16.3 kB]
Get:35 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [15.8 kB]
Get:36 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1304 B]
Get:37 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:38 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:39 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:40 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [748 kB]
91% [41 Packages 605 kB/748 kB 81%]                                                    5140 kB/s 0s
```



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-116:~$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-40-116:~$ newgrp docker
ubuntu@ip-172-31-40-116:~$ docker run -d --name nexus3 -p 8081:8081 sonatype/nexus3
```



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-33-14:~$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-33-14:~$ newgrp docker
ubuntu@ip-172-31-33-14:~$ docker run -d --name sonarqube -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
30a9c22ae099: Extracting [>                                                  ]  327.7kB/29.53MB
6dbdd66677ae: Downloading [====>                                              ]  1.485MB/16.15MB
3398a057cfd5: Download complete
c3e4752518b6: Waiting
f755fa637644: Waiting
101e6de839e3: Waiting
0355c6d609c4: Waiting
4f4fb700ef54: Waiting
```

<u>IN Jenkins server install docker</u>

access all of these and then install docker from the official website

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc



# Add the repository to Apt sources:

echo \

  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \

  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

after installing docker we have to give permission to Jenkins user

sudo usermod –aG docker Jenkins
```

instance ip :8080/restart

then install trivy in Jenkins server

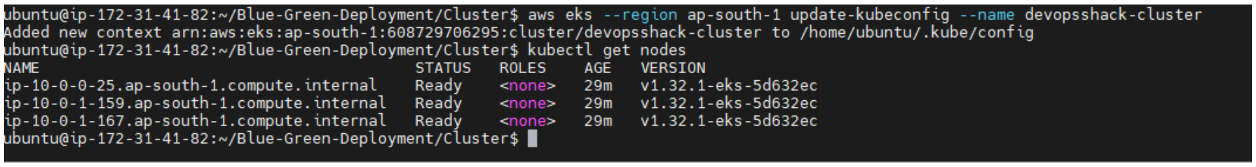
```
sudo apt-get install wget apt-transport-https gnupg lsb-release

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -

echo deb 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
```

then install kubectl in jenkins

```
sudo snap install kubectl –classic
```

install kubectl in server because we have use kubectl

```
sudo snap install kubectl –classic –y



 aws eks --region ap-south-1 update-kubeconfig --name devopsshack-cluster
```



```
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ aws eks --region ap-south-1 update-kubeconfig --name devopsshack-cluster
Added new context arn:aws:eks:ap-south-1:608729706295:cluster/devopsshack-cluster to /home/ubuntu/.kube/config
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ kubectl get nodes
NAME                                         STATUS   ROLES    AGE   VERSION
ip-10-0-0-25.ap-south-1.compute.internal     Ready    <none>   29m   v1.32.1-eks-5d632ec
ip-10-0-1-159.ap-south-1.compute.internal    Ready    <none>   29m   v1.32.1-eks-5d632ec
ip-10-0-1-167.ap-south-1.compute.internal    Ready    <none>   29m   v1.32.1-eks-5d632ec
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$
```

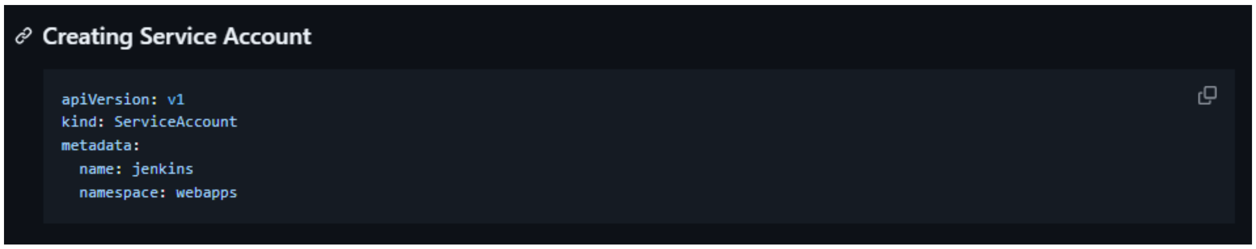
Next we want rbac for permission to perform or execute update,delete or deployment those permission only available to specific service account to the user that is we are using Jenkins for that

So we will create a service account in webapps namespace but before that we have to create a namespace webapps

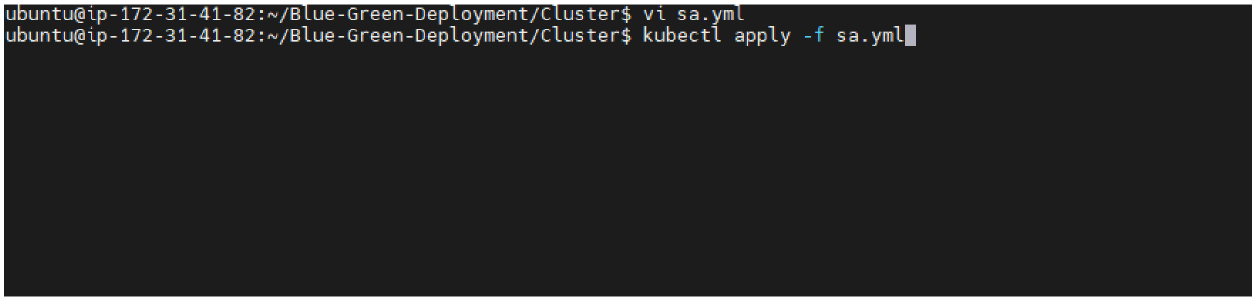
Kubectl create ns webapps

Kubectl apply –f sa.yml (for service account)

```
apiVersion: v1

kind: ServiceAccount

metadata:

  name: jenkins

  namespace: webapps
```



```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: webapps
```



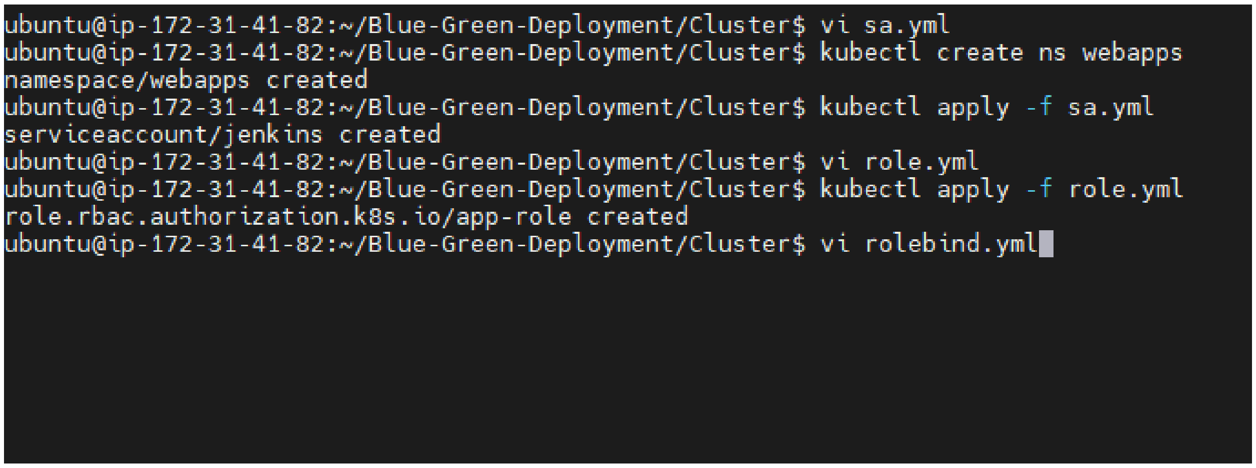
```
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ vi sa.yml
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ kubectl apply -f sa.yml
```

Then create a role which will perform through this

```
apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

  name: app-role

  namespace: webapps

rules:

  - apiGroups:

        - ""

        - apps

        - autoscaling

        - batch

        - extensions

        - policy

        - rbac.authorization.k8s.io

    resources:

      - pods

      - secrets

      - componentstatuses

      - configmaps
      - daemonsets
      - deployments
      - events
      - endpoints
      - horizontalpodautoscalers
      - ingress
      - jobs
      - limitranges
      - namespaces
      - nodes
      - pods
      - persistentvolumes
      - persistentvolumeclaims
      - resourcequotas
      - replicasets
      - replicationcontrollers
      - serviceaccounts
      - services
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

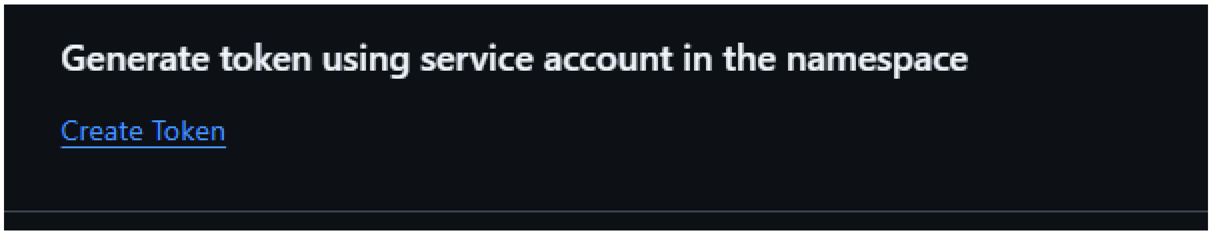
Next step is to bind this role to the service account so we will use role binding for that

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-rolebinding
  namespace: webapps
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: app-role
subjects:
- namespace: webapps
  kind: ServiceAccount
  name: Jenkins
```



```
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ vi sa.yml
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ kubectl create ns webapps
namespace/webapps created
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ kubectl apply -f sa.yml
serviceaccount/jenkins created
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ vi role.yml
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ kubectl apply -f role.yml
role.rbac.authorization.k8s.io/app-role created
ubuntu@ip-172-31-41-82:~/Blue-Green-Deployment/Cluster$ vi rolebind.yml
```

We are going to create a token



Generate token using service account in the namespace

Create Token

```
apiVersion: v1

kind: Secret

type: kubernetes.io/service-account-token

metadata:

  name: mysecretname

  annotations:

    kubernetes.io/service-account.name: Jenkins
```

```
kubectl apply –f sec.yaml –n webapps
```

now we have to get that token from that secret

for that for authentication

```
kubectl describe secret mysecretname –n webapps
```

then copy the secret

then go the Jenkins ui

manage Jenkins>secret credentials>global credential>add credential>secret text>paste the secret>k8-token

now we have setup rback and cluster also

Now we have to install certain plugins in jenkins

Sonarqube scanner

Maven integration

Config file provider

Pipeline maven integration

Docker pipeline

Pipeline stage view

Generic Webhook trigger

Kubernetes

Kubernetes cli

Kubernetes credentials

Kubernetes client api

Lets create our pipeline

```
pipeline {
    agent any

    parameters {
        choice(name: 'DEPLOY_ENV', choices: ['blue', 'green'], description: 'Choose which environment to deploy: Blue or Green')
        choice(name: 'DOCKER_TAG', choices: ['blue', 'green'], description: 'Choose the Docker image tag for the deployment')
        booleanParam(name: 'SWITCH_TRAFFIC', defaultValue: false, description: 'Switch traffic between Blue and Green')
    }

    environment {
        IMAGE_NAME = "premd91/bankapp"
        TAG = "${params.DOCKER_TAG}"  // The image tag now comes from the parameter
        KUBE_NAMESPACE = 'webapps'
        SCANNER_HOME= tool 'sonar-scanner'
    }
    stages {

        stage('Git Checkout') {

            steps {

                git branch: 'main', credentialsId: 'git-cred', url: https://github.com/devops-methodology/Blue-
Green-Deployment.git'

            }

        }
```

Before compiling we have to configure maven

Manage Jenkins> maven

Also configure sonarqube scanner in manage Jenkins> tools section

Before starting pipeline we have to setup some credentials go to manage Jenkins> credentials>

This is for token for sonarqube

Secret txt where to copy it

Go to sonarqube server>administration>security>users>generate tokens

And paste in Jenkins credentials section named it as “”sonar-token””

Now we have to manage sonarqube server inside Jenkins

So to managejenkins>system>

Name- sonar

Sonarserver url

Select the token we have to create

Then go to manage Jenkins> manage files>global maven settings>

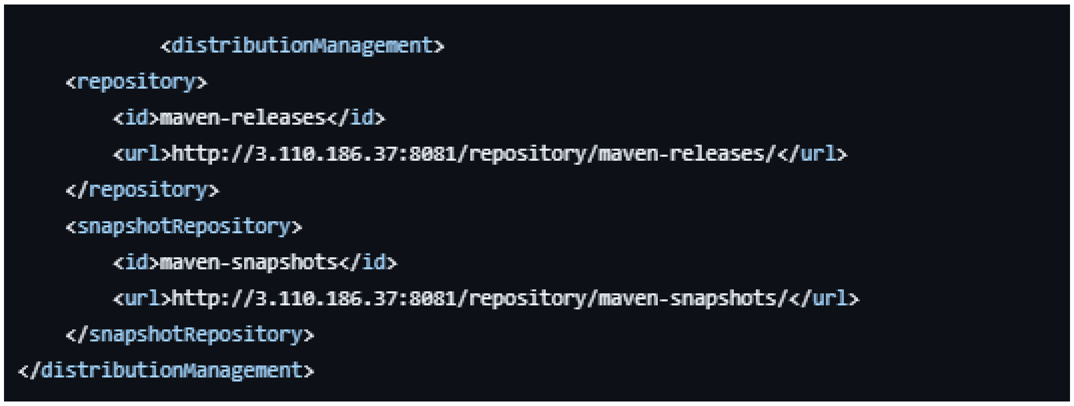
ID named as maven-settings

Ok>

Here we add our credentials for nexus server

ADD snapshot and release credentials

Then another thing go to nexus server>browse>copy the link of maven releases and snapshot url and paste in pom.xml>distribution management



```
            <distributionManagement>
    <repository>
        <id>maven-releases</id>
        <url>http://3.110.186.37:8081/repository/maven-releases/</url>
    </repository>
    <snapshotRepository>
        <id>maven-snapshots</id>
        <url>http://3.110.186.37:8081/repository/maven-snapshots/</url>
    </snapshotRepository>
</distributionManagement>
```

1/git checkout

2/mvn compile (tools {

Maven ‘maven3’

}

3/mvn test

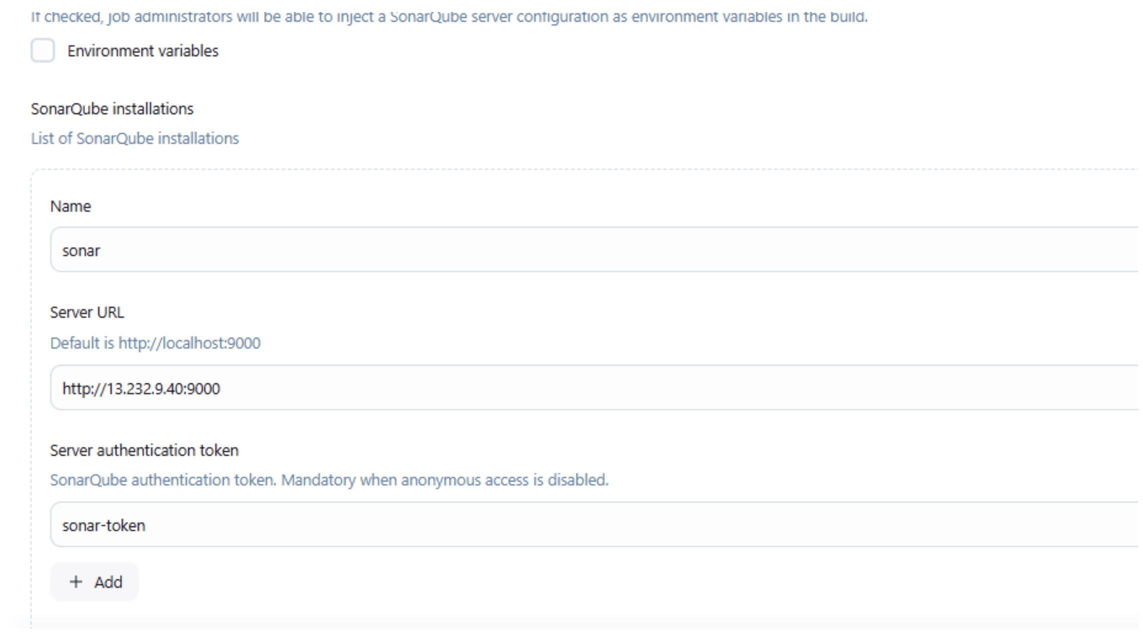
4/trivy FS scan(“trivy fs –format table –o fs.html .”)

5/sonarqube analysis(environment {

SCANNER_HOME= tool ‘sonar-scanner’

}

Sh “SCANNER_HOME/bin/sonar-scanner –Dsonar.projectKey=Multitier –Dsonar.projectName=Multtier –Dsonar.java.binaries=target


If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Name

sonar

Server URL

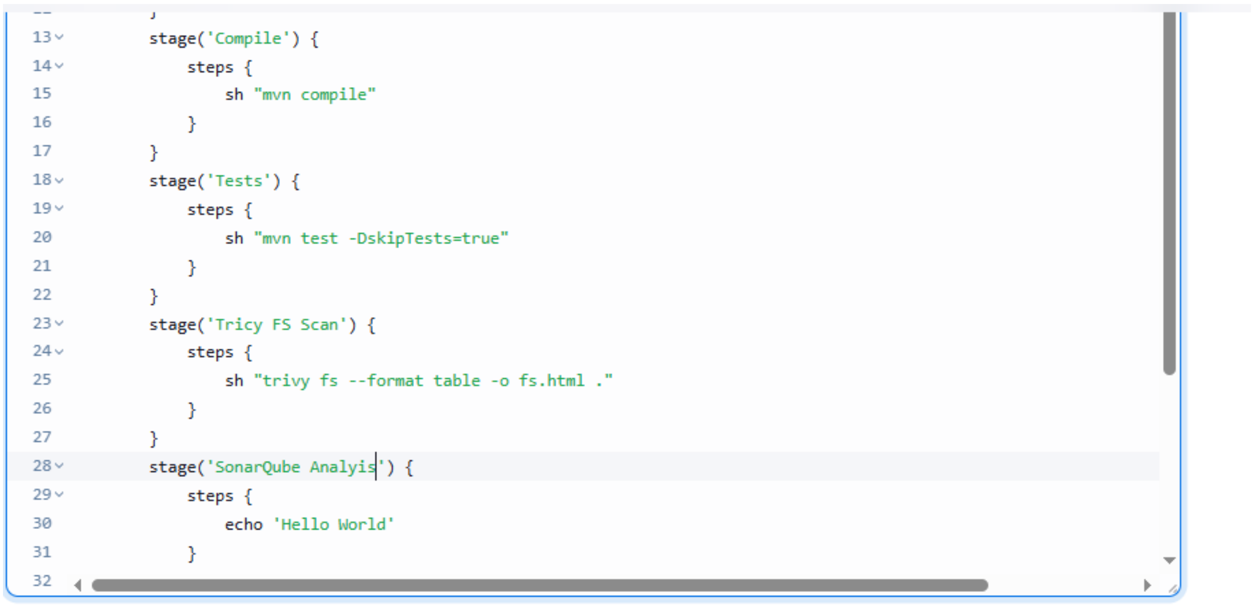
Default is http://localhost:9000

http://13.232.9.40:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

sonar-token


```
        stage('Compile') {
            steps {
                sh "mvn compile"
            }
        }
        stage('Tests') {
            steps {
                sh "mvn test -DskipTests=true"
            }
        }
        stage('Tricy FS Scan') {
            steps {
                sh "trivy fs --format table -o fs.html ."
            }
        }
        stage('SonarQube Analyis') {
            steps {
                echo 'Hello World'
            }
```

(Note: we have to create credentials for kubernetes =k8-token that we have generated,git-cred,sonar-token)

6/Then we go for webhook for quality gate check

Go to sonarqube server

Administration>configuration>webhook

Name-jenkins

http://ip:8080/sonarqube-webhook/

then create

after that lets say if the quality gate check taking too much time so we have another option go to pipeline syntax and there we choose time out option and generate the pipeline script and copy and paste it(for 1 hours)…I have choosen 60seconds(as per your requirement)

7/after that we have to build the application

“mvn package –DskipTests=true”

8/next publish the artifact to nexus

Sh “mvn deploy –DskipTests=true”

9/Docker Build & Tag

sh “docker build –t ${IMAGE_NAME}: ${TAG} .”

so before creating docker image we have 2 environment one is blue environment and another is green environment so we have to setup parameterized

For that

```
parameters {

        choice(name: 'DEPLOY_ENV', choices: ['blue', 'green'], description: 'Choose which environment to
deploy: Blue or Green')

        choice(name: 'DOCKER_TAG', choices: ['blue', 'green'], description: 'Choose the Docker image tag
for the deployment')

        booleanParam(name: 'SWITCH_TRAFFIC', defaultValue: false, description: 'Switch traffic between
Blue and Green')

    }
```

In this line we used to 3 aspects ---environment,tag and then traffic

After that we have to set two variables

image name and the tag

in environment section we have to add…

IMAGE_NAME = "adijaiswal/bankapp"

TAG = "${params.DOCKER_TAG}" // The image tag now comes from the parameter

10/then docker image scanning

sh " trivy image –format table –o fs.html ${IMAGE_NAME}: ${TAG}

11/push of docker image to docker hub

sh "docker push ${IMAGE_NAME}: ${TAG}"

12/deployment the mysql deployment

13/deployment mysql service

14/deployment to kubernetes

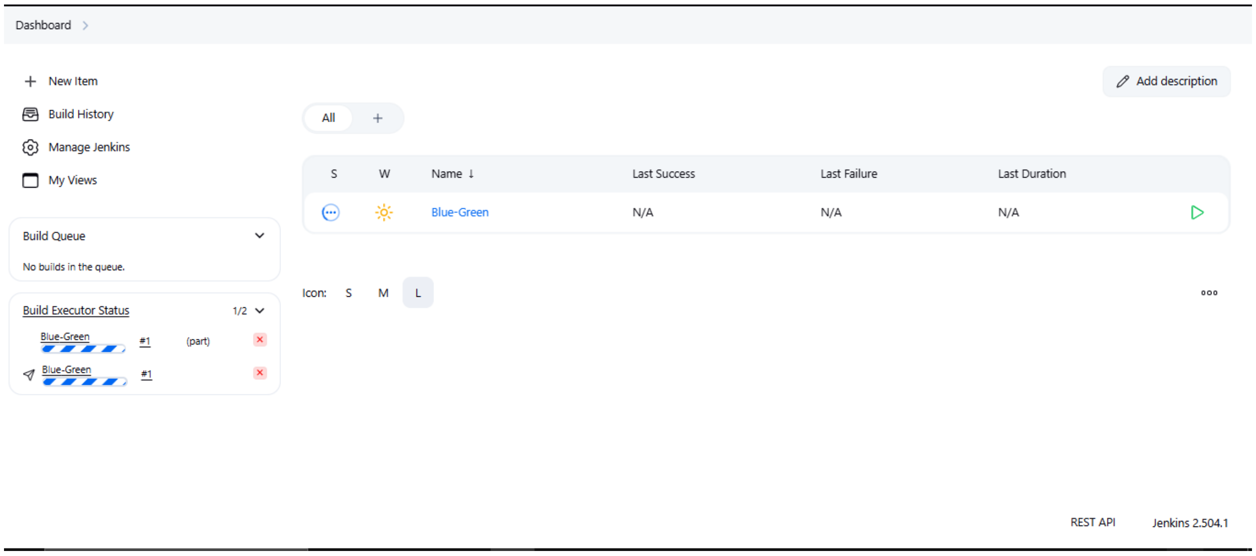
15/switching the traffic between blue and green environment based on the parameter selected.

(kubectl pacth service bank-app-service)

16/verify deployment

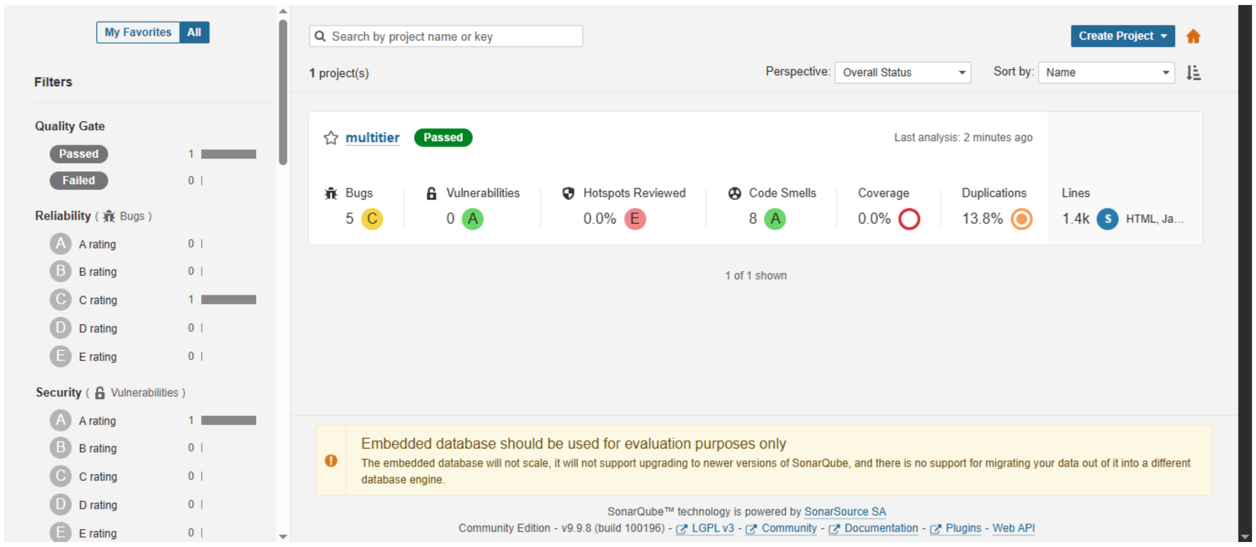
(the code is in the github repo check for that)---any issue dm me I will definitely help you after my work

When I first build the application there is no "Build With Parameter" option but after 2nd time it appears



S	W	Name ↓	Last Success	Last Failure	Last Duration
		Blue-Green	N/A	N/A	N/A

Checking the code any vulnerabilities or any credentials seen or any bug or code smell



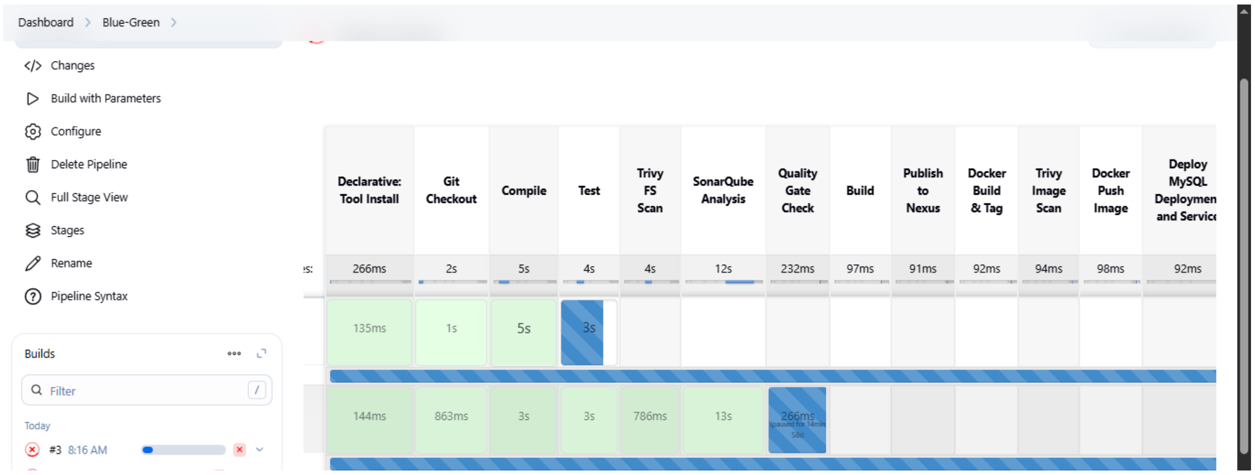
multitier Passed — Last analysis: 2 minutes ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
5 C	0 A	0.0% E	8 A	0.0%	13.8%	1.4k S HTML, Ja...

1 of 1 shown

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

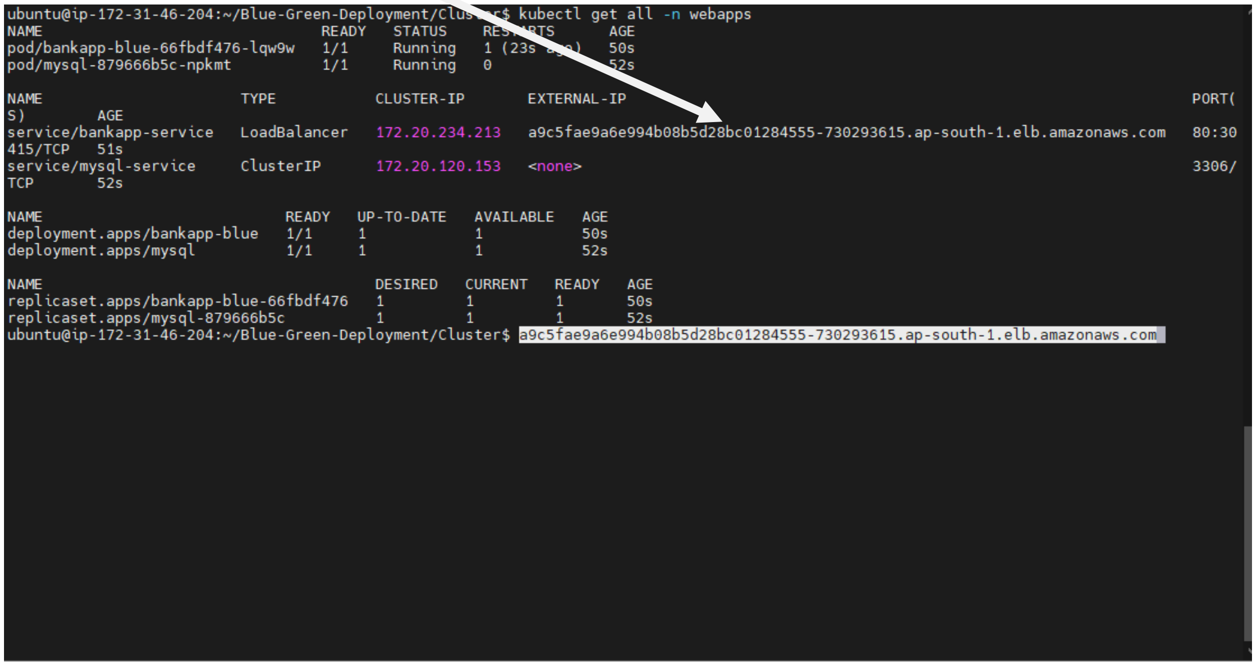
SonarQube™ technology is powered by SonarSource SA
Community Edition - v9.9.8 (build 100196) - LGPL v3 - Community - Documentation - Plugins - Web API



Declarative: Tool Install	Git Checkout	Compile	Test	Trivy FS Scan	SonarQube Analysis	Quality Gate Check	Build	Publish to Nexus	Docker Build & Tag	Trivy Image Scan	Docker Push Image	Deploy MySQL Deploymen and Service
266ms	2s	5s	4s	4s	12s	232ms	97ms	91ms	92ms	94ms	98ms	92ms
135ms	1s	5s	3s									
144ms	863ms	3s	3s	786ms	13s	266ms						

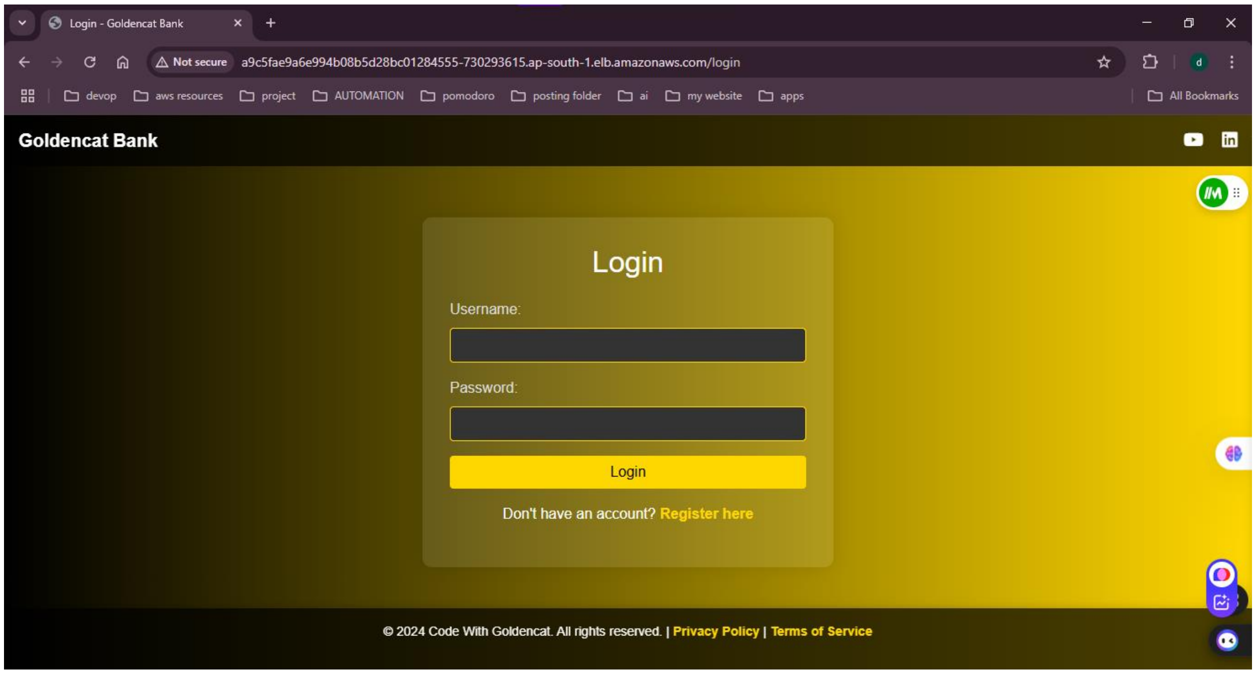
After that check by using in server

{kubectl get all –n webapps }

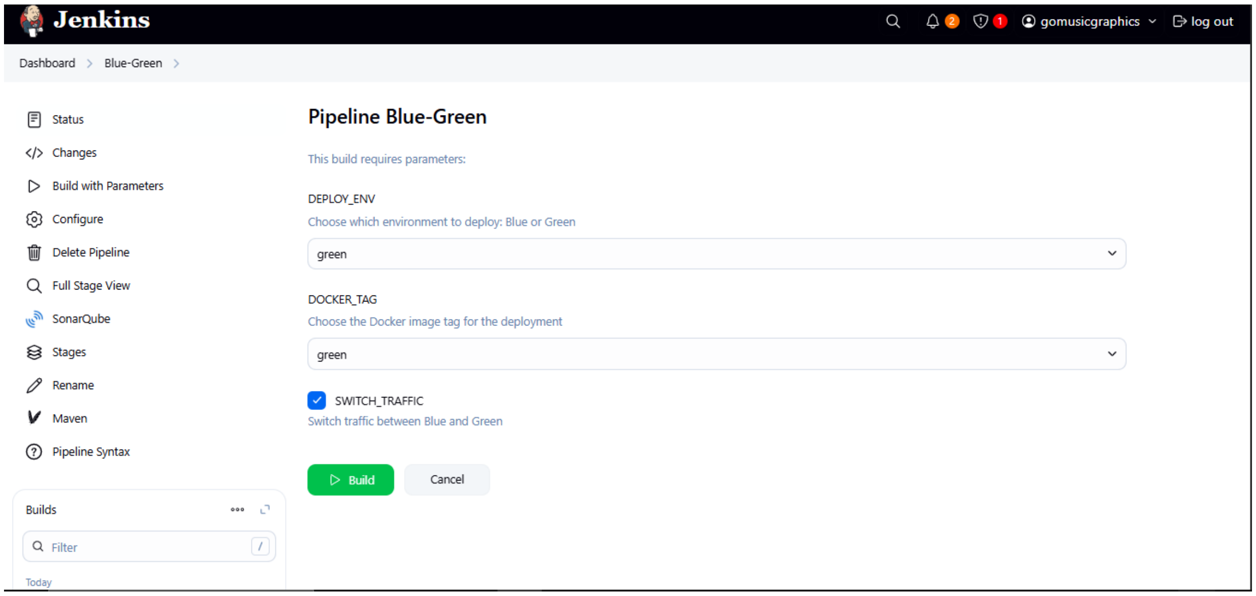


```
ubuntu@ip-172-31-46-204:~/Blue-Green-Deployment/Cluster$ kubectl get all -n webapps
NAME                               READY   STATUS    RESTARTS      AGE
pod/bankapp-blue-66fbdf476-lqw9w   1/1     Running   1 (23s ago)   50s
pod/mysql-879666b5c-npkmt          1/1     Running   0             52s

NAME                       TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(
S)         AGE
service/bankapp-service    LoadBalancer   172.20.234.213  a9c5fae9a6e994b08b5d28bc01284555-730293615.ap-south-1.elb.amazonaws.com   80:30
415/TCP    51s
service/mysql-service      ClusterIP      172.20.120.153  <none>                                                                     3306/
TCP        52s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bankapp-blue   1/1     1            1           50s
deployment.apps/mysql          1/1     1            1           52s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/bankapp-blue-66fbdf476   1         1         1       50s
replicaset.apps/mysql-879666b5c          1         1         1       52s
ubuntu@ip-172-31-46-204:~/Blue-Green-Deployment/Cluster$ a9c5fae9a6e994b08b5d28bc01284555-730293615.ap-south-1.elb.amazonaws.com
```

Paste in browser you will find after 5 to 10 seconds Goldencat bank app


Login - Goldencat Bank
Not secure
a9c5fae9a6e994b08b5d28bc01284555-730293615.ap-south-1.elb.amazonaws.com/login
devop
aws resources
project
AUTOMATION
pomodoro
posting folder
ai
my website
apps
All Bookmarks
Goldencat Bank
Login
Username:
Password:
Login
Don't have an account? Register here
© 2024 Code With Goldencat. All rights reserved. | Privacy Policy | Terms of Service

As I have added some features so created green env for that so I have switch the traffic from blue to green envirornment..



Pipeline Blue-Green

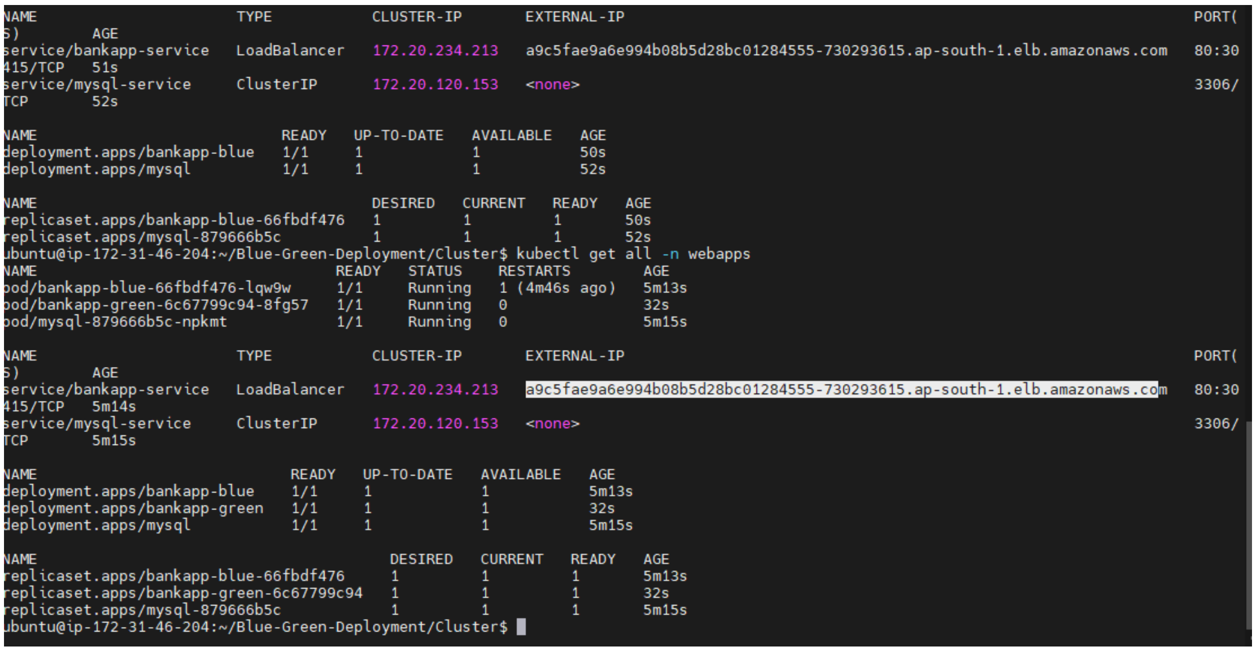
This build requires parameters:

DEPLOY_ENV
Choose which environment to deploy: Blue or Green
green

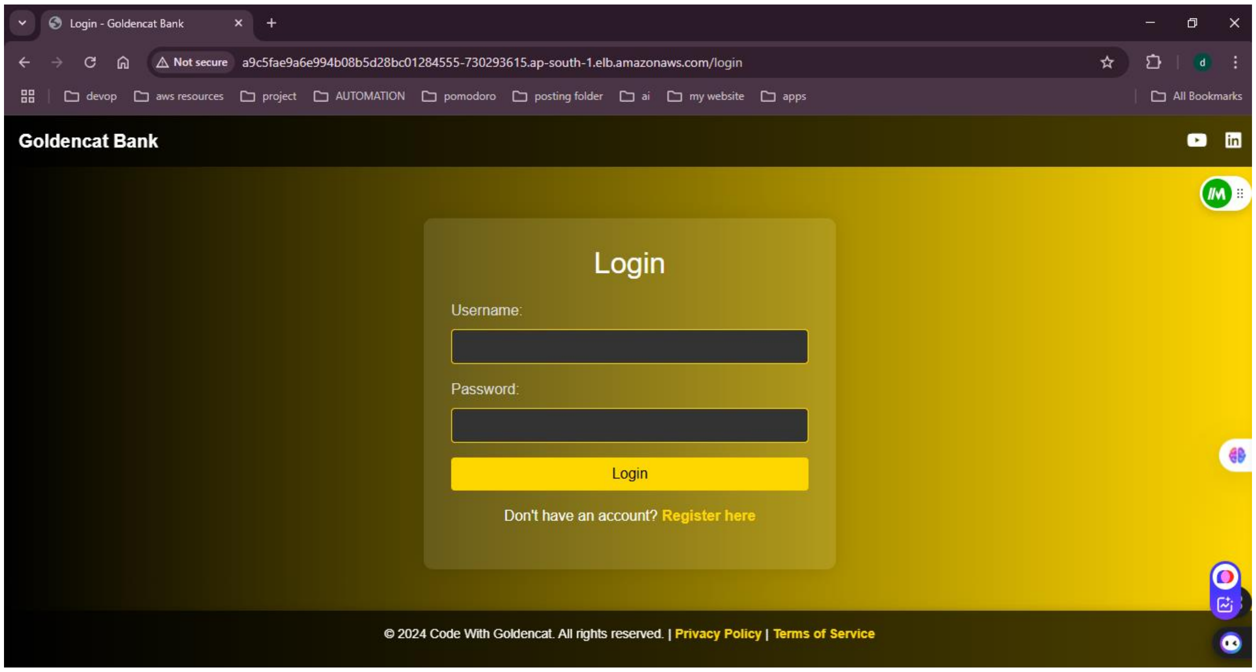
DOCKER_TAG
Choose the Docker image tag for the deployment
green

☑ SWITCH_TRAFFIC
Switch traffic between Blue and Green

Again I have done the same or you can restart the browser you will find the below goldencat bank app



```
NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP                                                                  PORT(
S)          AGE
service/bankapp-service     LoadBalancer   172.20.234.213   a9c5fae9a6e994b08b5d28bc01284555-730293615.ap-south-1.elb.amazonaws.com   80:30
415/TCP   51s
service/mysql-service       ClusterIP      172.20.120.153   <none>                                                                       3306/
TCP       52s

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bankapp-blue    1/1     1            1           50s
deployment.apps/mysql           1/1     1            1           52s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/bankapp-blue-66fbdf476    1         1         1       50s
replicaset.apps/mysql-879666b5c           1         1         1       52s
ubuntu@ip-172-31-46-204:~/Blue-Green-Deployment/Cluster$ kubectl get all -n webapps
NAME                                  READY   STATUS    RESTARTS        AGE
pod/bankapp-blue-66fbdf476-lqw9w      1/1     Running   1 (4m46s ago)   5m13s
pod/bankapp-green-6c67799c94-8fg57    1/1     Running   0               32s
pod/mysql-879666b5c-npkmt             1/1     Running   0               5m15s

NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP                                                                  PORT(
S)          AGE
service/bankapp-service     LoadBalancer   172.20.234.213   a9c5fae9a6e994b08b5d28bc01284555-730293615.ap-south-1.elb.amazonaws.com   80:30
415/TCP   5m14s
service/mysql-service       ClusterIP      172.20.120.153   <none>                                                                       3306/
TCP       5m15s

NAME                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bankapp-blue     1/1     1            1           5m13s
deployment.apps/bankapp-green    1/1     1            1           32s
deployment.apps/mysql            1/1     1            1           5m15s

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/bankapp-blue-66fbdf476      1         1         1       5m13s
replicaset.apps/bankapp-green-6c67799c94    1         1         1       32s
replicaset.apps/mysql-879666b5c             1         1         1       5m15s
ubuntu@ip-172-31-46-204:~/Blue-Green-Deployment/Cluster$
```



Goldencat Bank

Login

Username:

Password:

Login

Don't have an account? Register here

© 2024 Code With Goldencat. All rights reserved. | Privacy Policy | Terms of Service

(Remember To delete these resources or you will be charged ....)

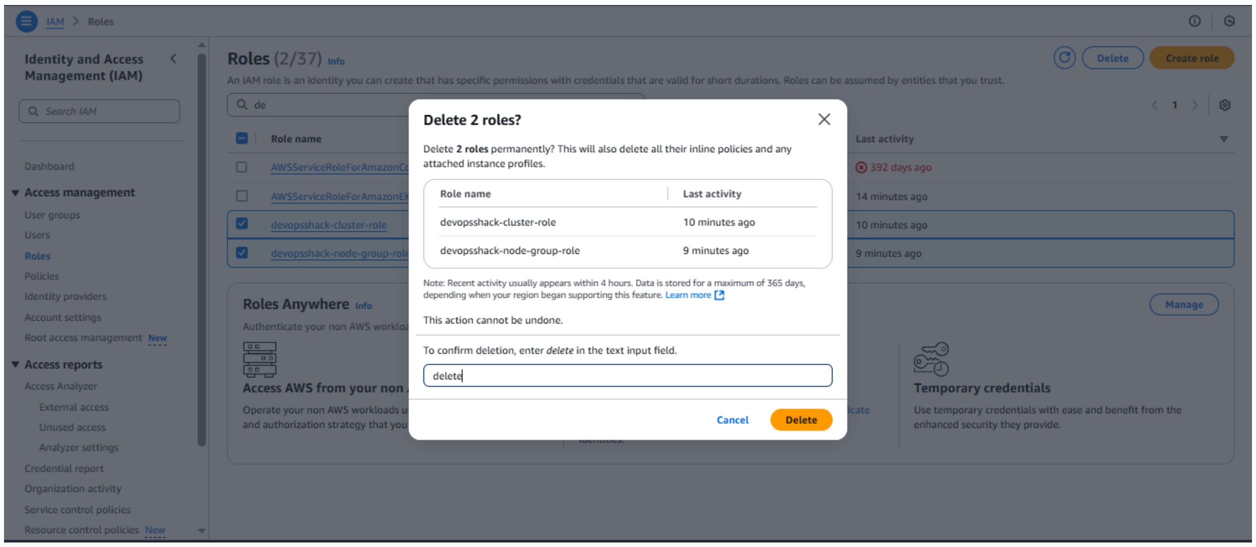
1/instance

2/eks cluster node-group then eks cluster(will take 2 to 3 minutes)

3/roles and policies(because when you use again this terraform code it will gave some error)

4/load balancer(very important)

5/check ec2 dasboard(for vpc,subnet,route -table)(if needed)—



An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Delete 2 roles?

Delete **2 roles** permanently? This will also delete all their inline policies and any attached instance profiles.

Role name	Last activity
devopsshack-cluster-role	10 minutes ago
devopsshack-node-group-role	9 minutes ago

Note: Recent activity usually appears within 4 hours. Data is stored for a maximum of 365 days, depending when your region began supporting this feature. Learn more

This action cannot be undone.

To confirm deletion, enter *delete* in the text input field.

delete

Cancel Delete