Complete Flow CI/CD

Tools are used:

- ✓ Ubuntu
- ✓ Terraform
- ✓ Git
- ✓ Github
- ✓ Jenkins
- ✓ Docker
- ✓ Dockerhub
- ✓ SonarQube
- ✓ Trivy
- ✓ OWASP(NA)
- ✓ EKS
- ✓ ArgoCD(GITOPS)

Terraform to launch EC2(Jenkins) instance with pre-requisites:

Note : Install **aws cli** and **aws configure** to set **Accesskey and Secretkey** and add **Elastic IP** to the Jenkins server**(Optional but in Production/Dev is must)**

Terraform code:

provider.tf

========

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

==========

main.tf

=======

```
#Vpc
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "jenkins_vpc"
  cidr = var.vpc_cidr

  azs          = data.aws_availability_zones.azs.names
```

```
  public_subnets = var.public_subnets

  enable_dns_hostnames    = true
  map_public_ip_on_launch = true

  tags = {
    Name       = "jenkins_vpc"
    Terraform   = "true"
    Environment = "dev"
  }
  public_subnet_tags = {
    Name = "jenkins_subnet"
  }
}

#sg

module "sg" {
  source = "terraform-aws-modules/security-group/aws"

  name       = "jenkins_sg"
  description = "Security group for jenkins server"
  vpc_id      = module.vpc.vpc_id


  ingress_with_cidr_blocks = [
    {
      from_port   = 0
      to_port     = 0
      protocol    = "-1"
      description = "HTTP"
```

```
    cidr_blocks = "0.0.0.0/0"
   },
   {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    description = "SSH"
    cidr_blocks = "0.0.0.0/0"
   }
  ]
  egress_with_cidr_blocks = [
   {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = "0.0.0.0/0"
   }
  ]
  tags = {
   Name = "jenkins_sg"
  }
}

#ec2

module "ec2_instance" {
  source = "terraform-aws-modules/ec2-instance/aws"

  name = "jenkins_server"

  instance_type              = var.instance_type
```

```
  ami                    = data.aws_ami.example.id
  key_name               = "ayush2"
  monitoring             = true
  vpc_security_group_ids = [module.sg.security_group_id]
  subnet_id              = module.vpc.public_subnets[0]
  associate_public_ip_address = true
  availability_zone      = data.aws_availability_zones.azs.names[0]
  user_data              = file("jenkins-install.sh")


  tags = {
    Name       = "jankins_server"
    Terraform  = "true"
    Environment = "dev"
  }
}
============

variable.tf
===========
variable "vpc_cidr" {
  description = "Vpc CIDR"
  type       = string
}

variable "public_subnets" {
  description = "public_subnets CIDR"
  type       = list(string)
}

variable "instance_type" {
```

```
 description = "Instance Type"
 type        = string
}
=========
backend.tf
========
terraform {
 backend "s3" {
   bucket = "testayush"
   key    = "jenkins/terraform.tfstate"
   region = "us-west-2"
 }
}
===========
data.tf
=========
data "aws_ami" "example" {

 most_recent = true
 owners      = ["amazon"]

 filter {
   name   = "name"
   values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20231207"]
 }

 filter {
   name   = "root-device-type"
   values = ["ebs"]
 }
```

```
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

data "aws_availability_zones" "azs" {}
```

==========

jenkins-install.sh

==========

```
#!/bin/bash
# For Ubuntu 22.04
# Intsalling Java
sudo apt update -y
sudo apt install openjdk-17-jre -y
sudo apt install openjdk-17-jdk -y
java --version

# Installing Jenkins
curl -fsSL https://pkg.jenkins.io/debian/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 binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update -y
sudo apt-get install jenkins -y

# Installing Docker
sudo apt update -y
sudo apt install docker.io -y
sudo usermod -aG docker jenkins
```

```
sudo usermod -aG docker ubuntu
sudo systemctl restart docker
sudo chmod 777 /var/run/docker.sock

# If you don't want to install Jenkins, you can create a container of Jenkins
# docker run -d -p 8080:8080 -p 50000:50000 --name jenkins-container jenkins/jenkins:lts

# Run Docker Container of Sonarqube
#docker run -d  --name sonar -p 9000:9000 sonarqube:lts-community
docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube


# Installing AWS CLI
#!/bin/bash
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt install unzip -y
unzip awscliv2.zip
sudo ./aws/install

# Installing Kubectl
#!/bin/bash
sudo apt update
sudo apt install curl -y
sudo curl -LO "https://dl.k8s.io/release/v1.28.4/bin/linux/amd64/kubectl"
sudo chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version --client


# Installing eksctl
#! /bin/bash
```

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

Installing Terraform

#!/bin/bash

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update

sudo apt install terraform -y

Installing Trivy

#!/bin/bash

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

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 update

sudo apt install trivy -y

Intalling Helm

#! /bin/bash

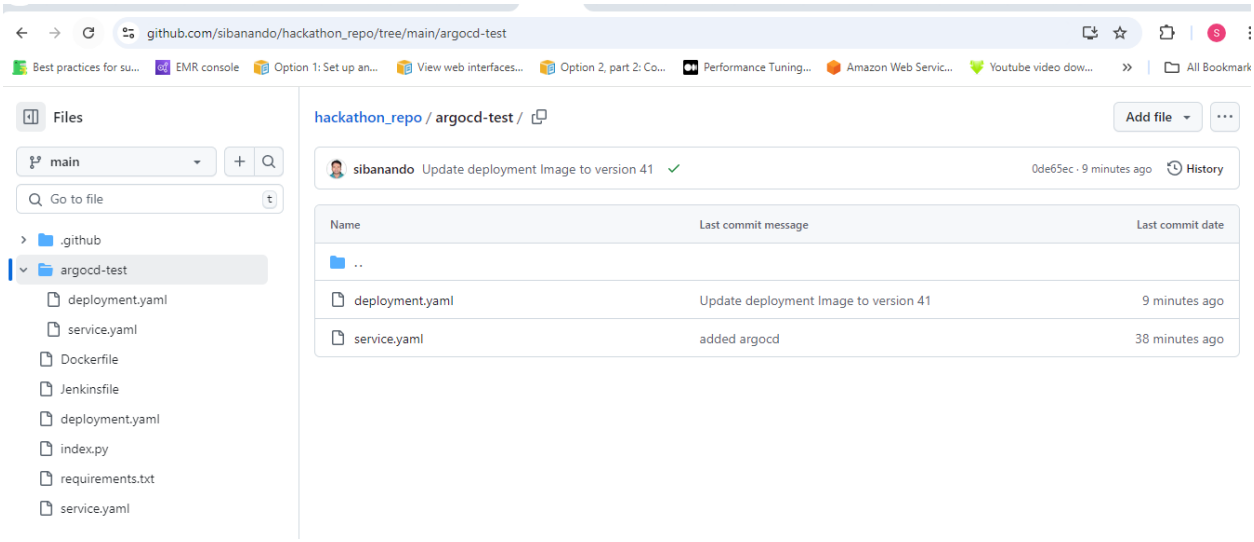
sudo snap install helm --classic

==========

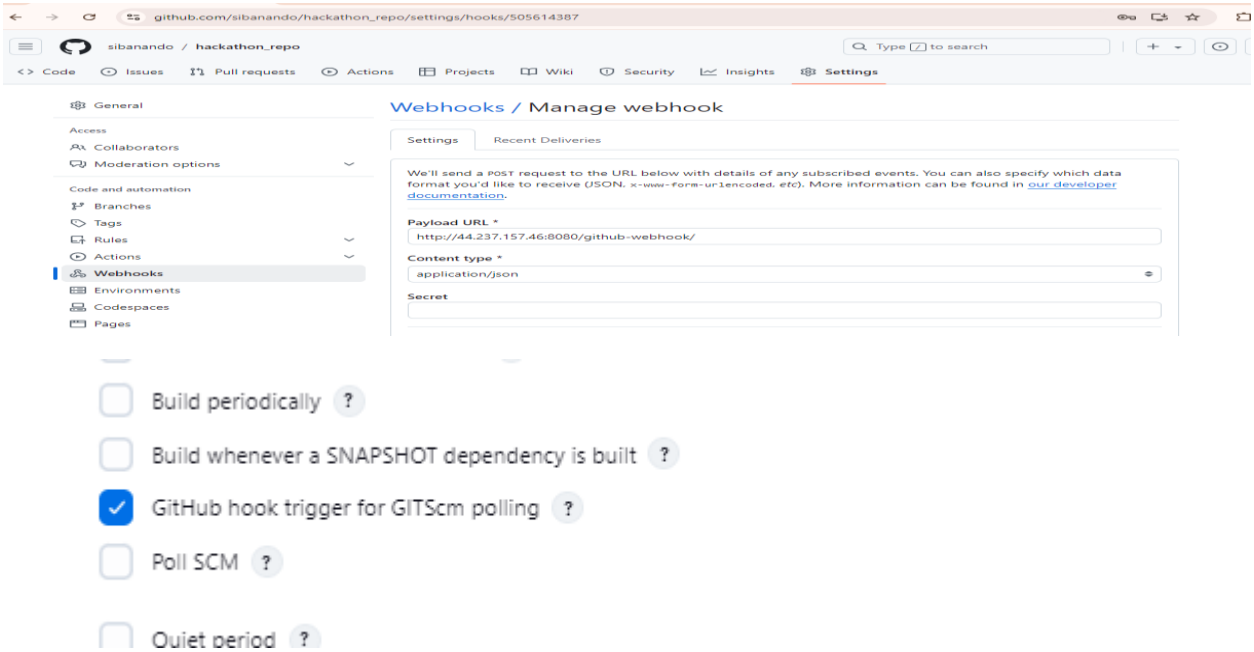
Github

==========

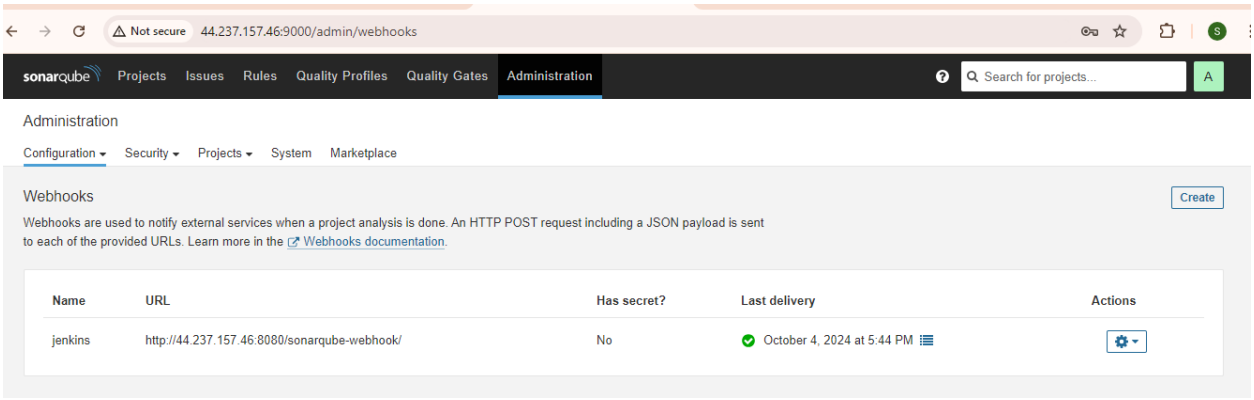
Repo : https://github.com/sibanando/hackathon_repo.git

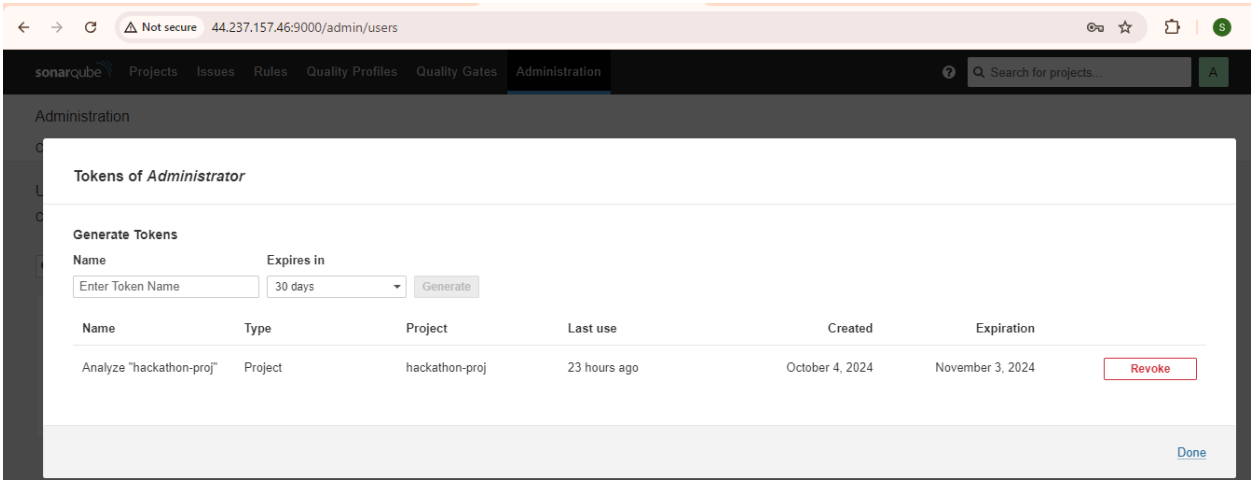


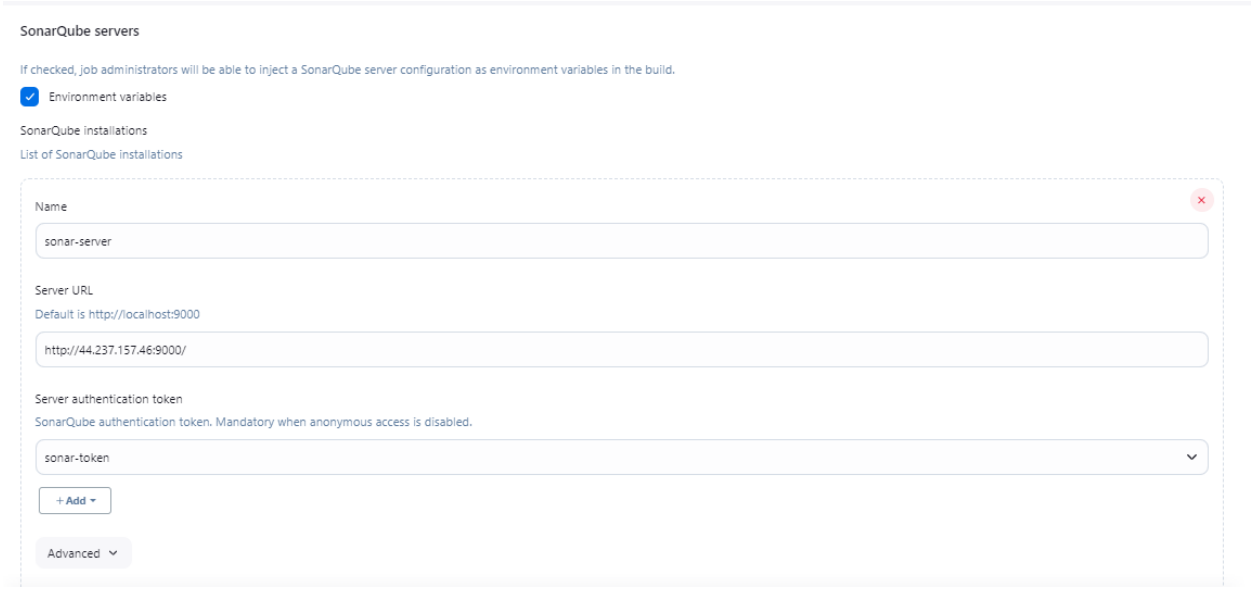
Github webhook

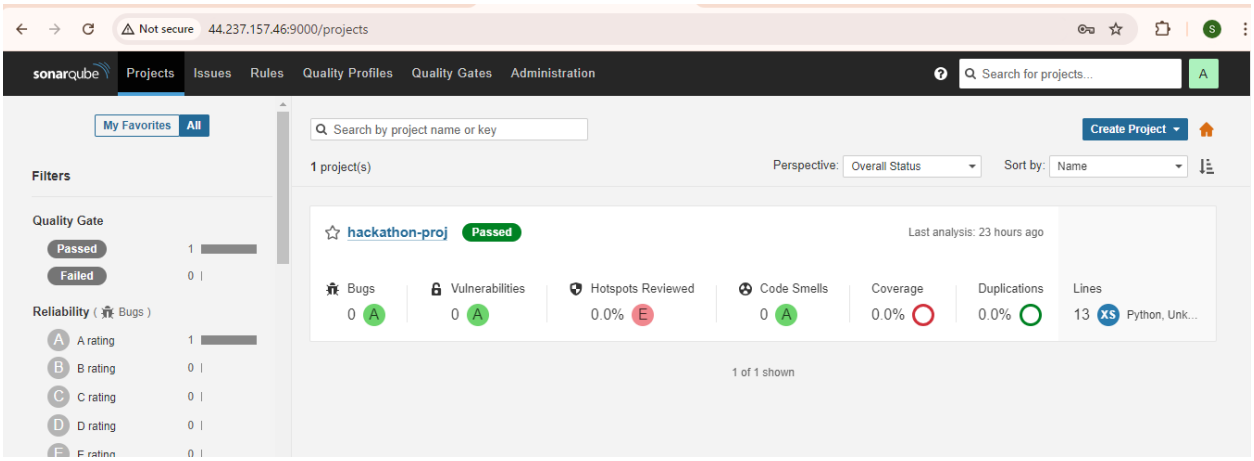


SonarQube:

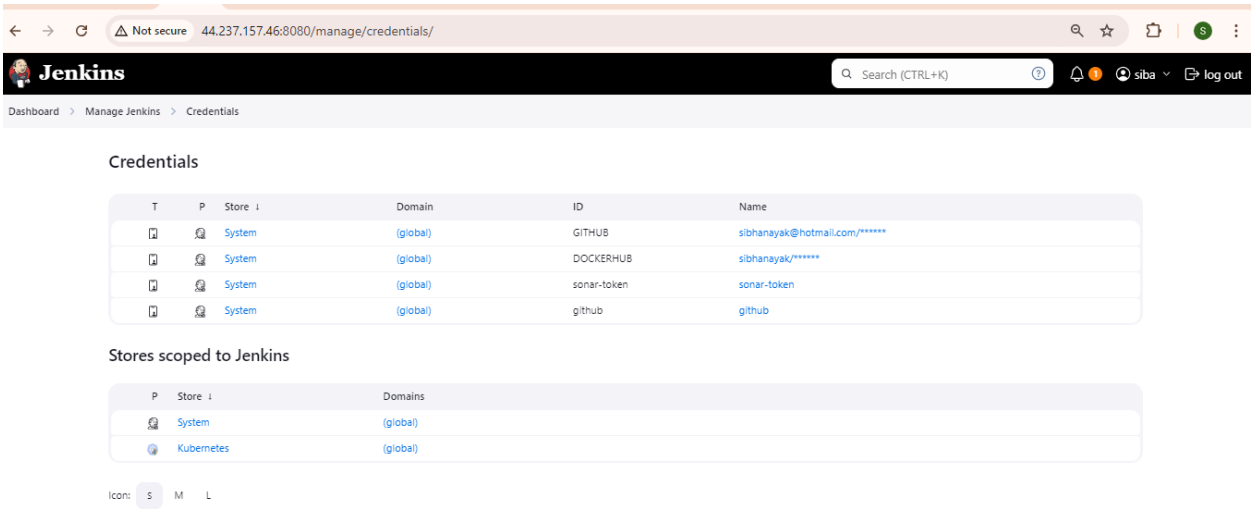

Not secure 44.237.157.46:9000/admin/webhooks
sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects...
Administration
Configuration Security Projects System Marketplace
Webhooks
Create
Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the Webhooks documentation.
Name URL Has secret? Last delivery Actions
jenkins http://44.237.157.46:8080/sonarqube-webhook/ No October 4, 2024 at 5:44 PM


Not secure 44.237.157.46:9000/admin/users
sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects...
Administration
Tokens of Administrator
Generate Tokens
Name
Expires in
Enter Token Name
30 days
Generate
Name Type Project Last use Created Expiration
Analyze "hackathon-proj" Project hackathon-proj 23 hours ago October 4, 2024 November 3, 2024
Revoke
Done


SonarQube servers
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
Server URL
Default is http://localhost:9000
http://44.237.157.46:9000/
Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
sonar-token
+ Add
Advanced



In Jenkins -> manage jenkins-> Credential



==========

Jenkinsfile

==========

```
pipeline {
  agent any
   environment {
   DOCKERHUB_CREDENTIALS= credentials('DOCKERHUB')
 }
  stages{

     stage('Cleaning Workspace') {
```

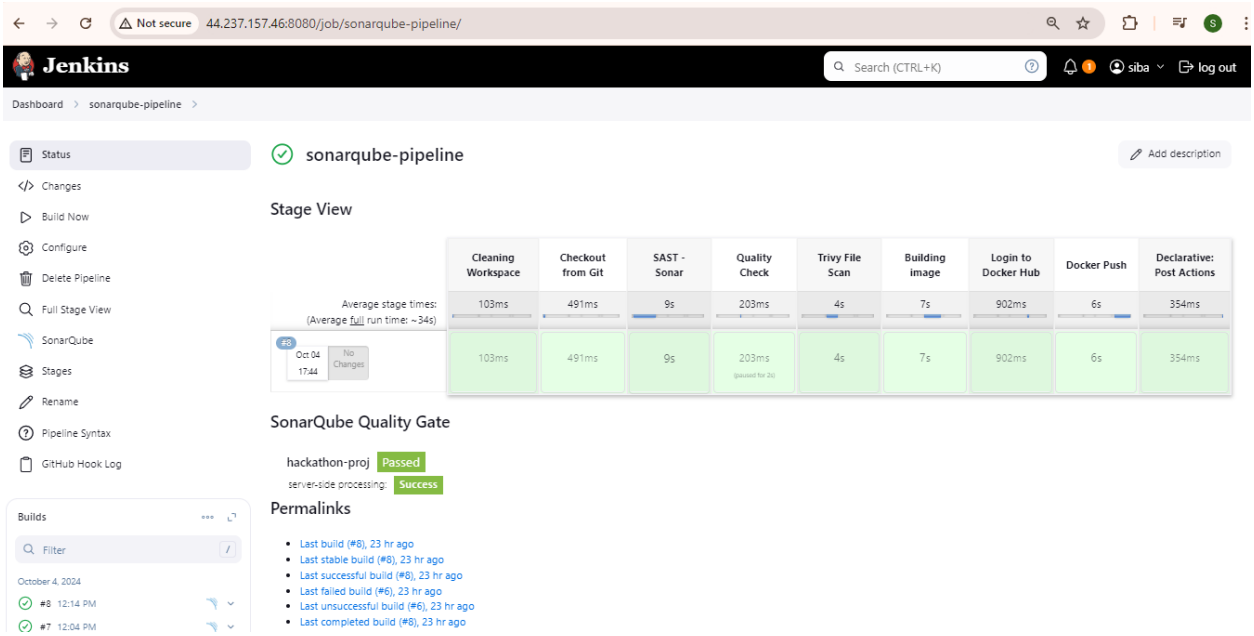
```
        steps {
            cleanWs()
        }
    }
    stage('Checkout from Git') {
        steps {
            git branch: 'main', credentialsId: 'github', url: 'https://github.com/sibanando/hackathon_repo.git'
        }
    }
        stage('Trivy File Scan') {
        steps {

            sh 'trivy fs . > trivyfs.txt'

        }
    }
  stage('SAST - Sonar') {
      environment {
        scannerHome = tool 'sonar-scanner';
      }
      steps {
       withSonarQubeEnv(credentialsId: 'sonar-token', installationName: 'sonar-server') {
        sh "${scannerHome}/bin/sonar-scanner -Dsonar.projectName=hackathon-proj -Dsonar.projectKey=hackathon-proj "
       }
      }
    }
  stage('Quality Check') {
      steps {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'
```

```
            }
          }
       }
            // stage('OWASP Dependency-Check Scan') {
    //      steps {
    //
    //                      dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit',
odcInstallation: 'DP-Check'
    //            dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
     //
     //     }
      // }


   // Building Docker images


    // Uploading Docker images into Docker Hub
     stage('Building image') {
       steps{
         script {
           sh 'docker system prune -f'
           sh 'docker container prune -f'
           sh 'docker build -t  sibhanayak/pythonapp:$BUILD_NUMBER .'
         }
       }
     }


  stage('Login to Docker Hub') {
     steps{
           sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --
password-stdin'
           echo 'Login Completed'
```

```
  }
}
```

```
// Running Docker container, make sure port 8096 is opened in
 stage('Docker Push') {
   steps{
     script {
       sh 'docker push sibhanayak/pythonapp:$BUILD_NUMBER'
     }
  }
}
stage("TRIVY Image Scan") {
        steps {
            sh 'trivy image sibhanayak/pythonapp:$BUILD_NUMBER > trivyimage.txt'
        }
    }
    stage('Checkout2 from Git') {
        steps {
            git branch: 'main', credentialsId: 'github', url: 'https://github.com/sibanando/hackathon_repo.git'
        }
    }
    stage('Update Deployment file') {
        environment {
            GIT_REPO_NAME = "hackathon_repo"
            GIT_USER_NAME = "sibanando"
        }
        steps {
```

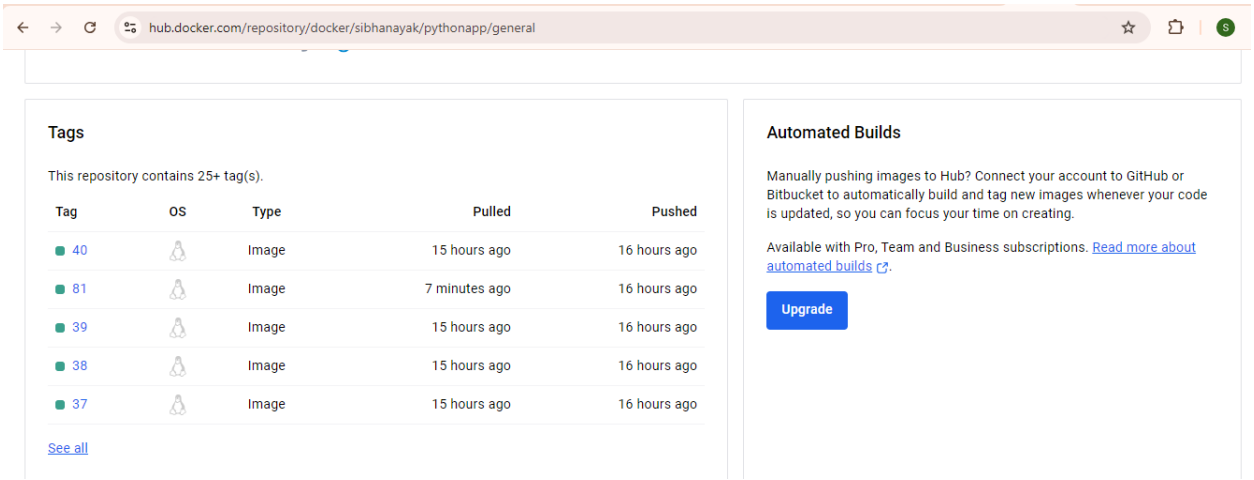
```
                withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
                  sh '''
                    git config user.email "sibhanayak@hotmail.com"
                    git config user.name "sibanando"
                    BUILD_NUMBER=${BUILD_NUMBER}
                    echo $BUILD_NUMBER
                   sed -i "s/replaceImageTag/${BUILD_NUMBER}/g" argocd-test/deployment.yaml
                    git add argocd-test/deployment.yaml
                    git commit -m "Update deployment Image to version \${BUILD_NUMBER}"
                    git push https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME}
HEAD:main
                  '''
                }
            }
        }


}
post{
   always {
     sh 'docker logout'
   }
 }
}
```



==========

Dockerhub

==========



==========

EKS cluster

==========

Creating eks:

eksctl create cluster --name hackathon-k8s --region us-west-2 --node-type t2.medium --zones us-west-2a,us-west-2b

Update-kubeconfig to access Kubernetes in kubectl :

aws eks update-kubeconfig --region us-west-2 --name hackathon-k8s

Delete Kubernetes cluster

eksctl delete cluster --name hackathon-k8s --region us-west-2

==========

ARGOCD

==========

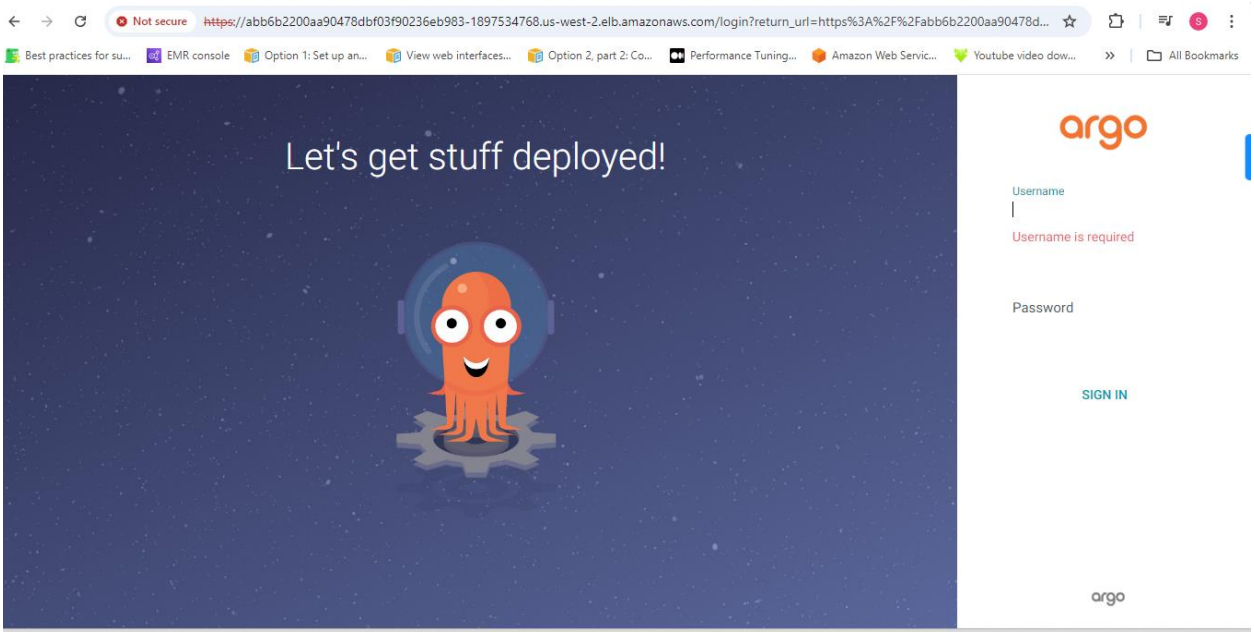
kubectl create namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

kubectl get pods -n argocd -w

kubectl edit svc argocd-server -n argocd

kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'



ubuntu@ip-10-0-1-36:~$ kubectl get secret -n argocd

NAME TYPE DATA AGE

argocd-initial-admin-secret Opaque 1 4h

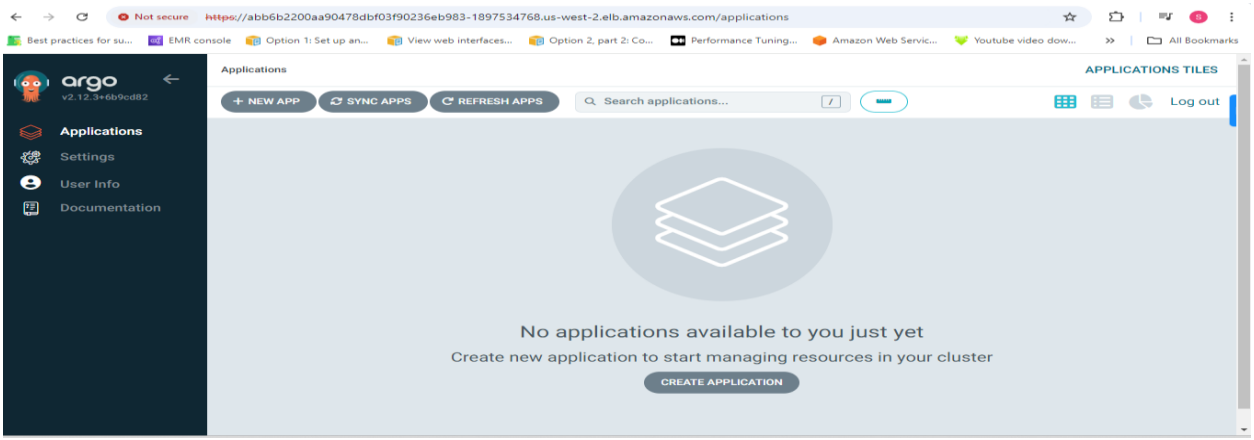
argocd-notifications-secret Opaque 0 4h

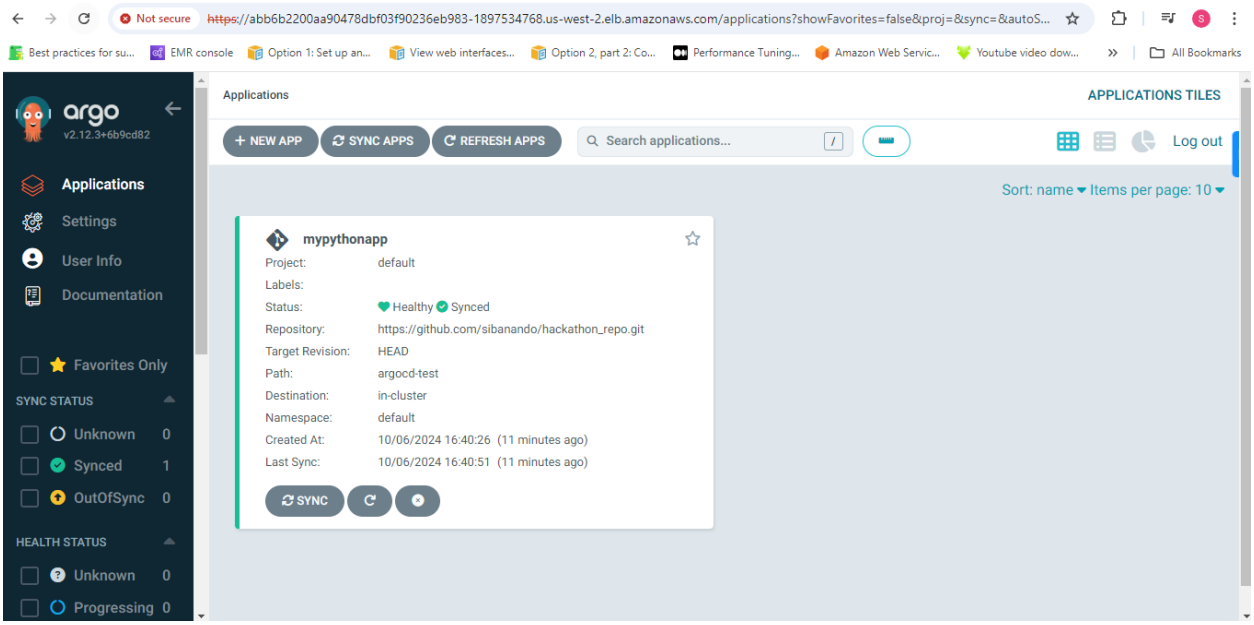
argocd-redis Opaque 1 4h

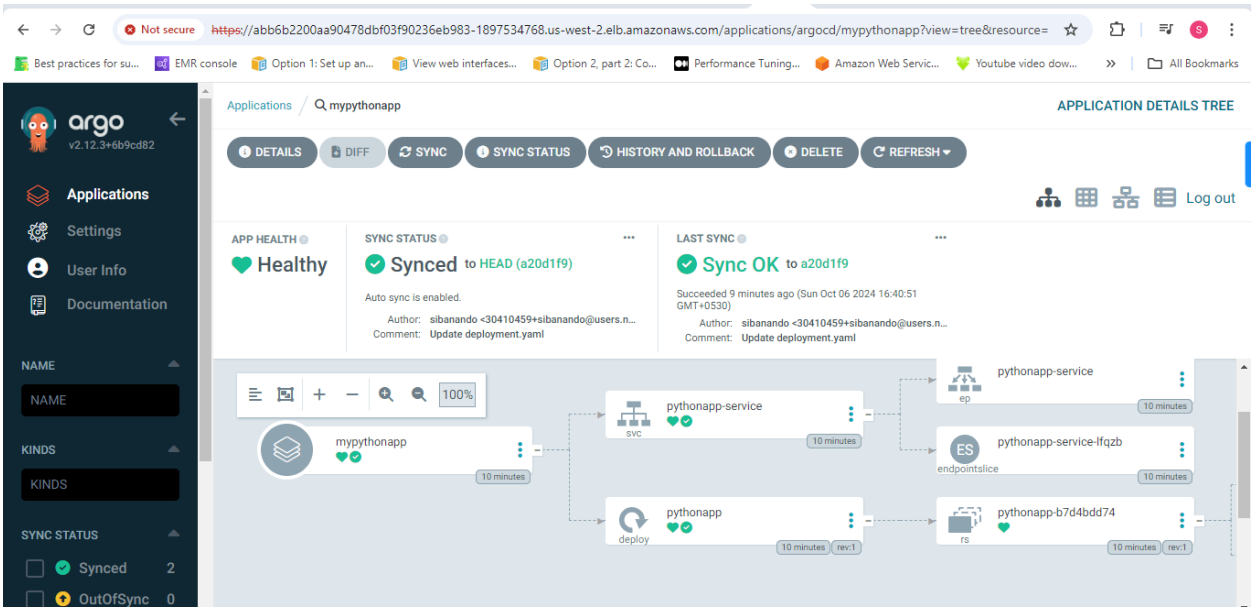
argocd-secret Opaque 5 4h

kubectl edit secret argocd-initial-admin-secret -n argocd

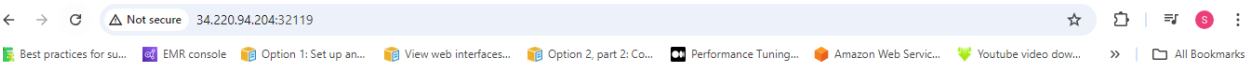
echo WVlZd24tWUVZbFdCdTBCUw== |base64 –decode



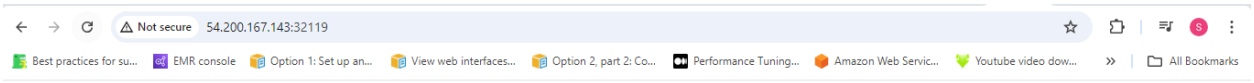





```
ubuntu@ip-10-0-1-36:~/hackathon_repo$ kubectl get svc -n argocd
NAME                                      TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)
        AGE
argocd-applicationset-controller          ClusterIP      10.100.240.8     <none>                                                                    7000/TCP,8080/TCP
        5h1m
argocd-dex-server                         ClusterIP      10.100.1.139     <none>                                                                    5556/TCP,5557/TCP,
58/TCP   5h1m
argocd-metrics                            ClusterIP      10.100.140.127   <none>                                                                    8082/TCP
        5h1m
argocd-notifications-controller-metrics   ClusterIP      10.100.203.193   <none>                                                                    9001/TCP
        5h1m
argocd-redis                              ClusterIP      10.100.142.193   <none>                                                                    6379/TCP
        5h1m
argocd-repo-server                        ClusterIP      10.100.103.25    <none>                                                                    8081/TCP,8084/TCP
        5h1m
argocd-server                             LoadBalancer   10.100.129.161   abb6b2200aa90478dbf03f90236eb983-1897534768.us-west-2.elb.amazonaws.com   80:30218/TCP,443:3
70/TCP   5h1m
argocd-server-metrics                     ClusterIP      10.100.108.81    <none>                                                                    8083/TCP
        5h1m
ubuntu@ip-10-0-1-36:~/hackathon_repo$ kubectl get pods -n argocd
NAME                                               READY   STATUS    RESTARTS   AGE
argocd-application-controller-0                    1/1     Running   0          5h2m
argocd-applicationset-controller-744b76d7fd-qbhws  1/1     Running   0          5h2m
argocd-dex-server-5bf5dbc64d-brwh4                 1/1     Running   0          5h2m
argocd-notifications-controller-84f5bf6896-gkpxd   1/1     Running   0          5h2m
argocd-redis-74b8999f94-hvgcm                      1/1     Running   0          5h2m
argocd-repo-server-57f4899557-t787z                1/1     Running   0          5h2m
argocd-server-7bc7b97977-wfqdf                     1/1     Running   0          5h2m
ubuntu@ip-10-0-1-36:~/hackathon_repo$ kubectl get pod
NAME                         READY   STATUS    RESTARTS   AGE
pythonapp-b7d4bdd74-bspn2    1/1     Running   0          119s
pythonapp-b7d4bdd74-lfcs4    1/1     Running   0          119s
ubuntu@ip-10-0-1-36:~/hackathon_repo$ kubectl get pod -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP               NODE                                              NOMINATED NODE   READINESS GATES
pythonapp-b7d4bdd74-bspn2    1/1     Running   0          2m11s   192.168.36.142   ip-192-168-44-172.us-west-2.compute.internal   <none>           <none>
pythonapp-b7d4bdd74-lfcs4    1/1     Running   0          2m11s   192.168.11.30    ip-192-168-27-203.us-west-2.compute.internal   <none>           <none>
ubuntu@ip-10-0-1-36:~/hackathon_repo$ kubectl get svc  -o wide
NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE     SELECTOR
kubernetes           ClusterIP   10.100.0.1     <none>        443/TCP          5h23m   <none>
pythonapp-service    NodePort    10.100.74.92   <none>        3000:32119/TCP   2m46s   app=pythonapp
ubuntu@ip-10-0-1-36:~/hackathon_repo$
```



ay learning python easy?



hay learning python easy?