Github Action CI/CD pipeline flow

Tools are used:

- ✓ Ubuntu
- ✓ Terraform
- ✓ Git
- ✓ Github Action
- ✓ Docker
- ✓ Dockerhub
- ✓ SonarQube
- ✓ EKS

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

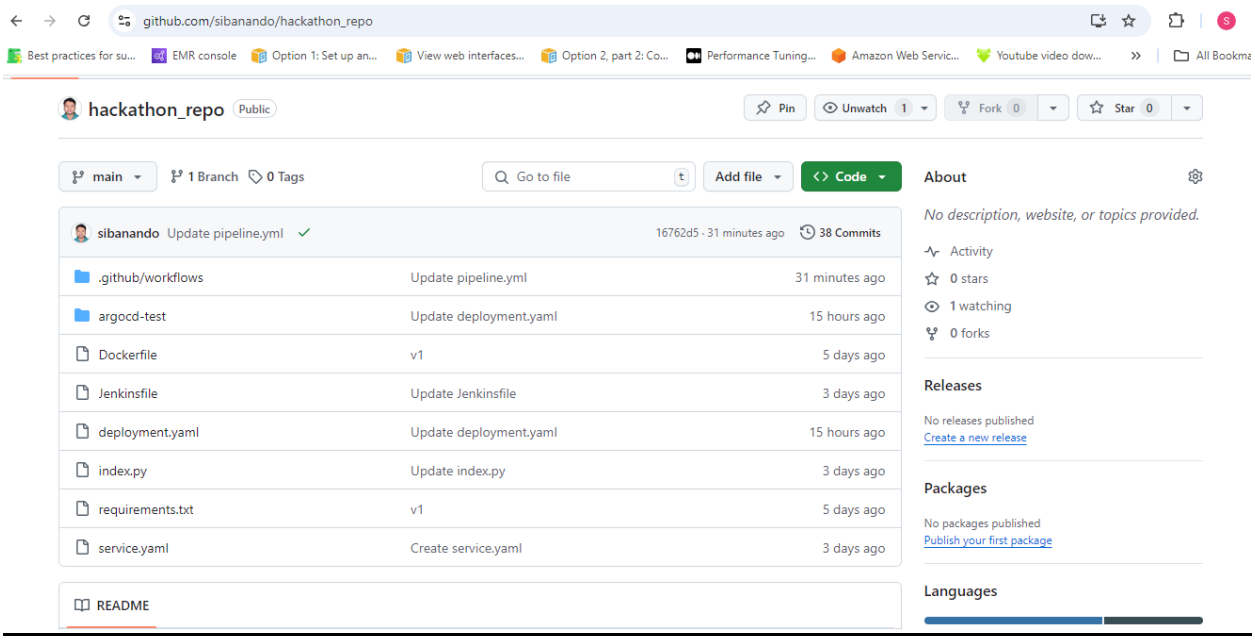
sudo snap install helm --classic

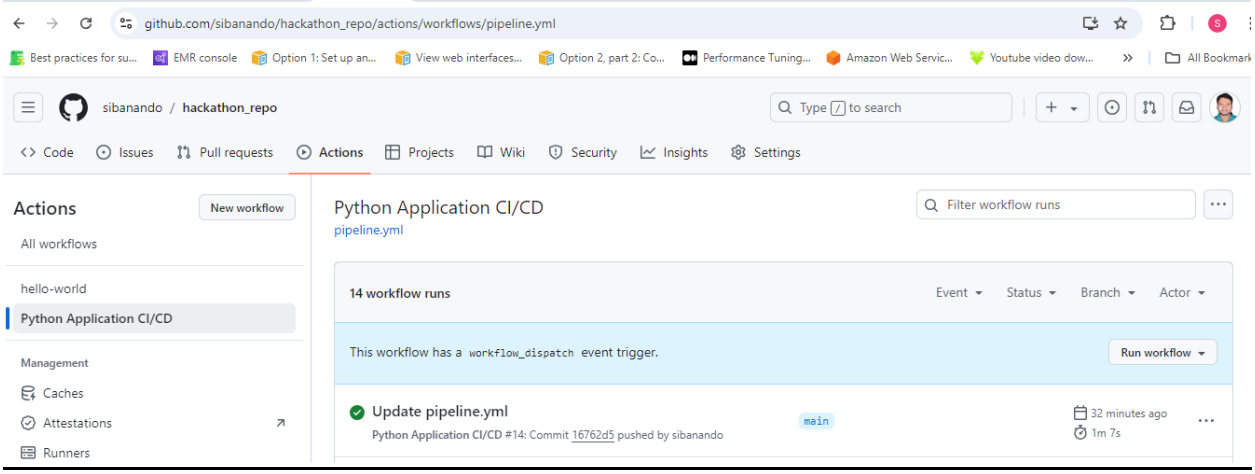
==========

Github and Github Action

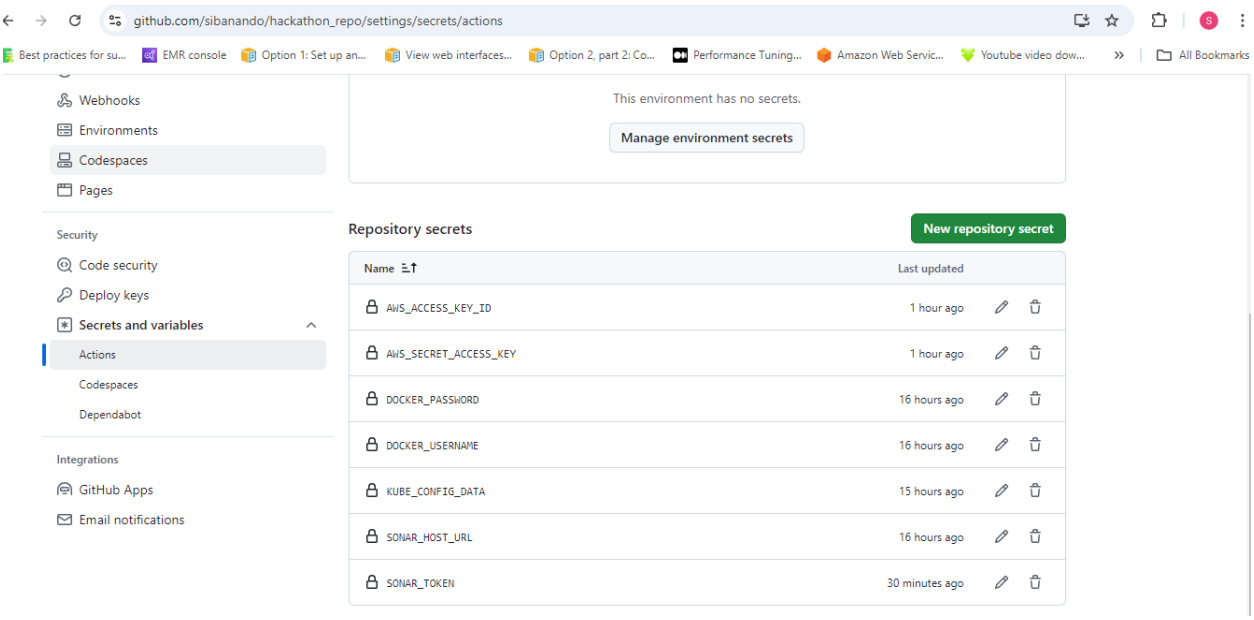
==========

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





Github Env and secret:



Github action pipeline code:

==========

.github/workflows/pipeline.yml

==========

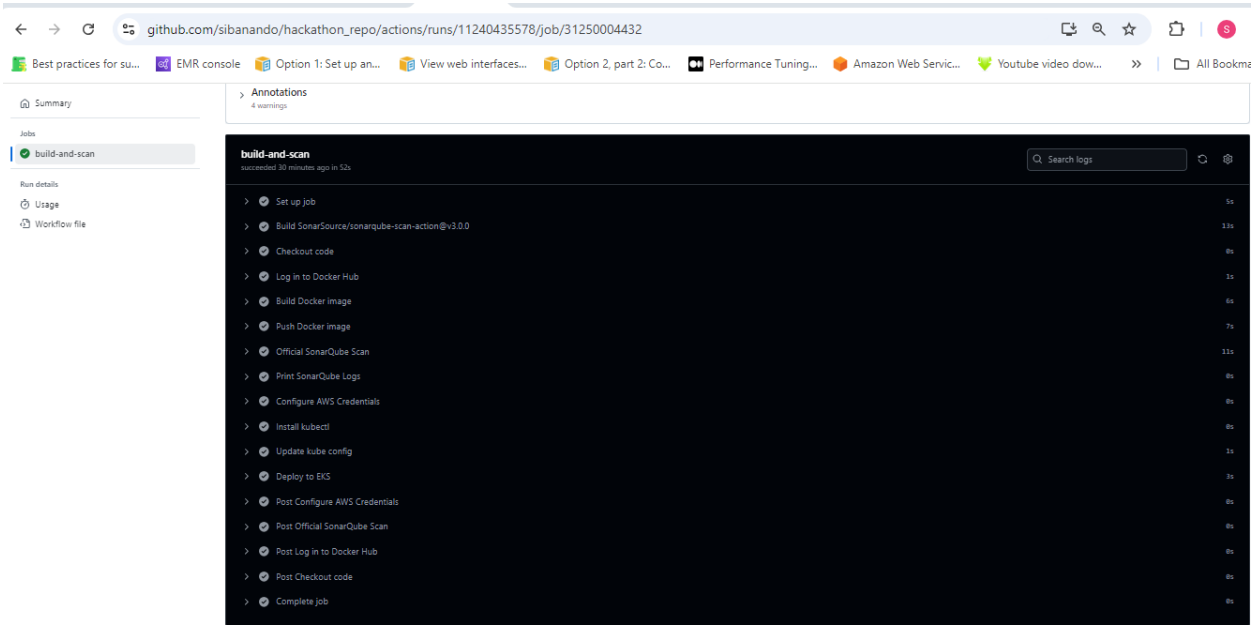
```
name: Python Application CI/CD
on:
  push:
    branches:
      - main
  workflow_dispatch: null
  pull_request:
    branches:
      - main
jobs:
  build-and-scan:
    runs-on: ubuntu-latest
    env:
```

```
  DOCKER_IMAGE: sibhanayak/pythonapp
  SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
  SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
  KUBE_CONFIG_DATA: ${{ secrets.KUBE_CONFIG_DATA }}
  AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
  AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
steps:
  - name: Checkout code
    uses: actions/checkout@v3
  - name: Log in to Docker Hub
    uses: docker/login-action@v2
    with:
      username: ${{ secrets.DOCKER_USERNAME }}
      password: ${{ secrets.DOCKER_PASSWORD }}
  - name: Build Docker image
    run: |
      docker build -t $DOCKER_IMAGE:${{ github.sha }} .
      docker tag $DOCKER_IMAGE:${{ github.sha }} $DOCKER_IMAGE:latest
  - name: Push Docker image
    run: |
      docker push $DOCKER_IMAGE:${{ github.sha }}
      docker push $DOCKER_IMAGE:latest
  - name: Official SonarQube Scan
    uses: SonarSource/sonarqube-scan-action@v3.0.0
    with:
      projectBaseDir: .
      args: >
```

```
      -Dsonar.projectKey=hackathon-proj -Dsonar.host.url=${{ env.SONAR_HOST_URL
      }} -Dsonar.login=${{ env.SONAR_TOKEN }}
      -Dsonar.working.directory=./.scannerwork
- name: Print SonarQube Logs
  run: >
    ls -al .scannerwork
    cat .scannerwork/report-task.txt || echo "report-task.txt not found"
- name: Configure AWS Credentials
  uses: aws-actions/configure-aws-credentials@v1
  with:
    aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
    aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
    aws-region: us-west-2
- name: Install kubectl
  uses: azure/setup-kubectl@v2.0
  with:
     version: 'v1.24.0' # default is latest stable
  id: install
- name: Update kube config
  run: aws eks update-kubeconfig --region us-west-2 --name hackathon-k8s
- name: Deploy to EKS
  run: |
    kubectl apply -f deployment.yaml
    kubectl apply -f service.yaml
```

==========

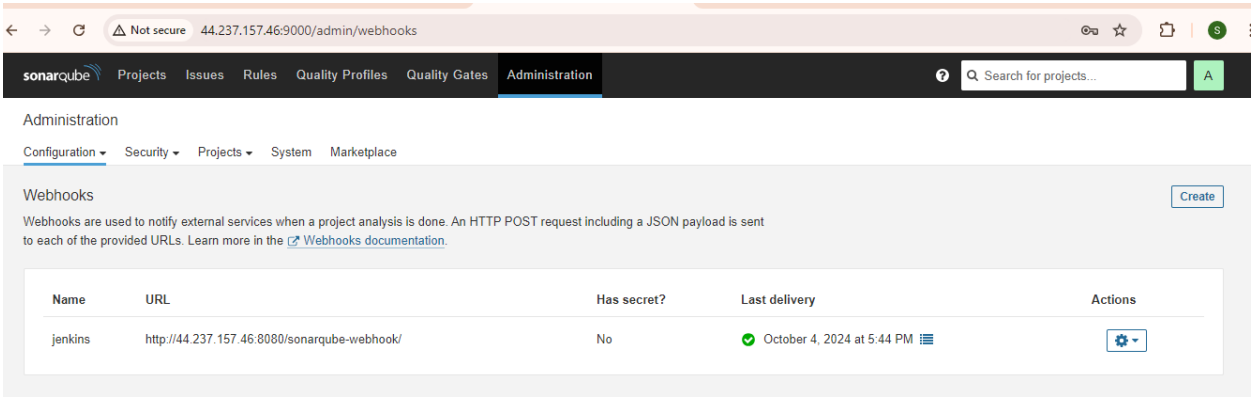
Output :

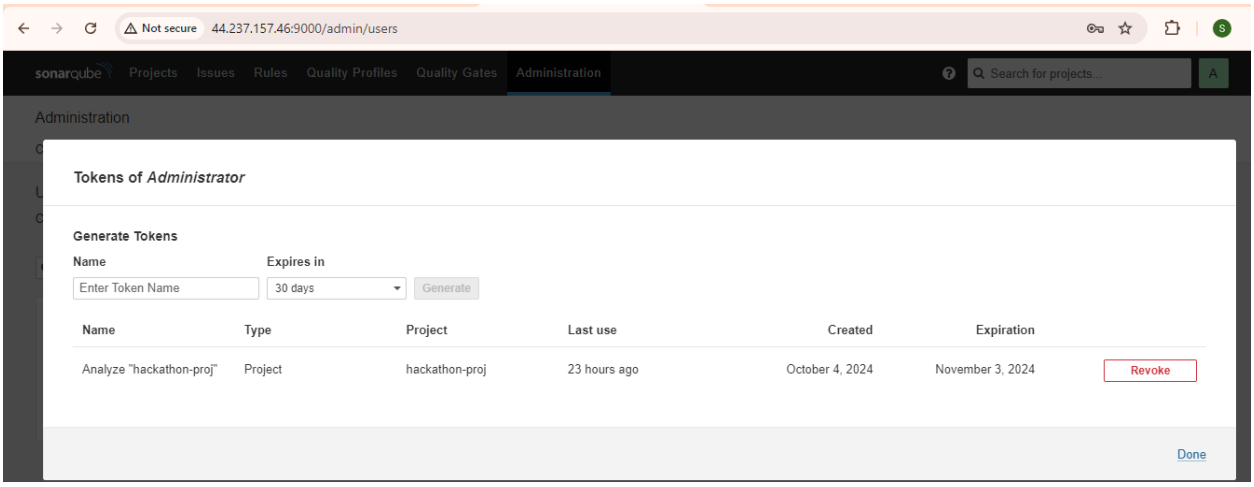


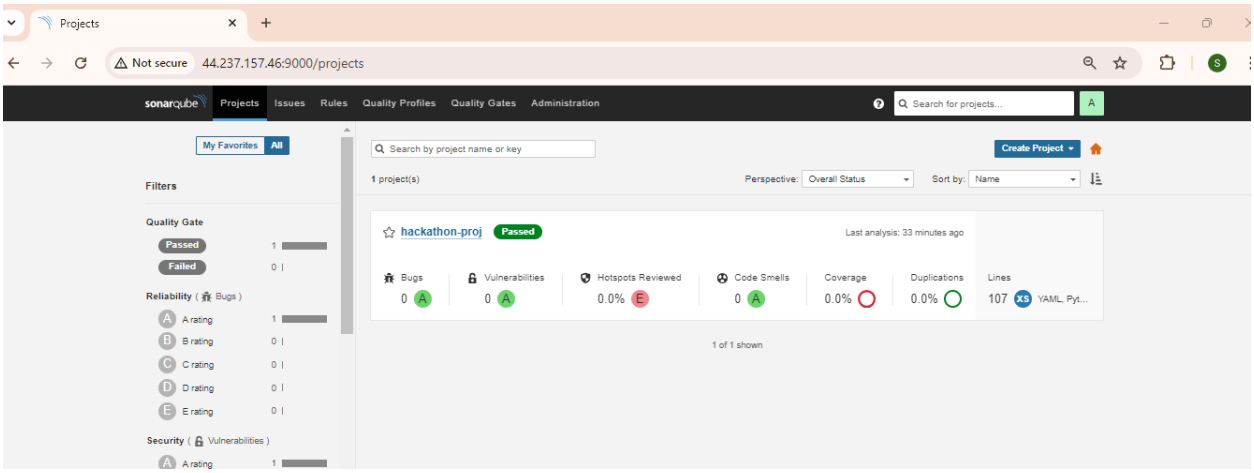
==========

SonarQube:

==========



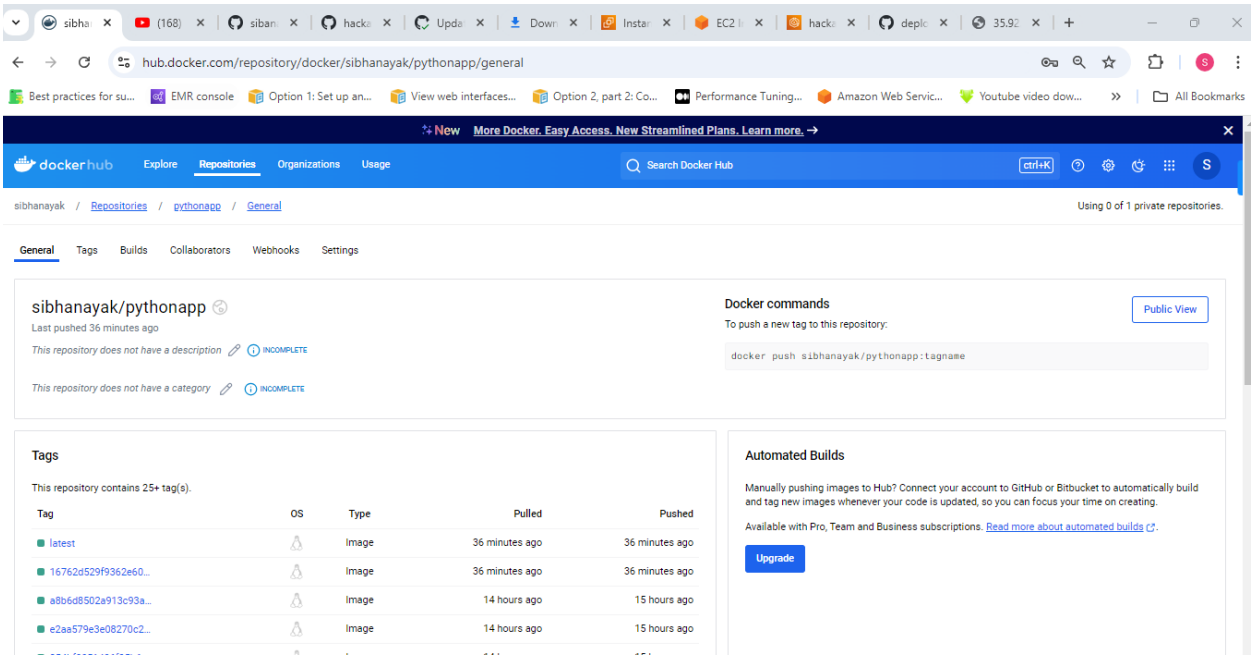

Not secure 44.237.157.46:9000/admin/users
sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects...
Administration
Tokens of Administrator
Generate Tokens
Name
Expires in
Enter Token Name
30 days
Generate
Name Type Project Last use Created Expiration
Analyze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


Projects
Not secure 44.237.157.46:9000/projects
sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects...
My Favorites All
Filters
Quality Gate
Passed
Failed
Reliability (Bugs)
Security (Vulnerabilities)
Search by project name or key
Create Project
1 project(s)
Perspective: Overall Status
Sort by: Name
hackathon-proj Passed
Last analysis: 33 minutes ago
Bugs Vulnerabilities Hotspots Reviewed Code Smells Coverage Duplications Lines
0 0 0.0% 0 0.0% 0.0% 107 YAML, Pyt...
1 of 1 shown

==========

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

========

ubuntu@ip-10-0-1-76:~$ kubectl get pod

NAME READY STATUS RESTARTS AGE

pythonapp-99bd946d4-76pzf 1/1 Running 0 37m

pythonapp-99bd946d4-ms584 1/1 Running 0 37m

```
ubuntu@ip-10-0-1-76:~$ kubectl get svc
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes       ClusterIP   10.100.0.1      <none>        443/TCP          51m
pythonapp-service   NodePort    10.100.217.23   <none>        3000:30840/TCP   38m
ubuntu@ip-10-0-1-76:~$ kubectl get pod -o wide
NAME                         READY   STATUS    RESTARTS   AGE   IP              NODE                                           NOMINATED NODE   READINESS GATES
pythonapp-99bd946d4-76pzf    1/1     Running   0          38m   192.168.30.96   ip-192-168-26-147.us-west-2.compute.internal   <none>           <none>
pythonapp-99bd946d4-ms584    1/1     Running   0          38m   192.168.50.203  ip-192-168-43-59.us-west-2.compute.internal    <none>           <none>
ubuntu@ip-10-0-1-76:~$
```