**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

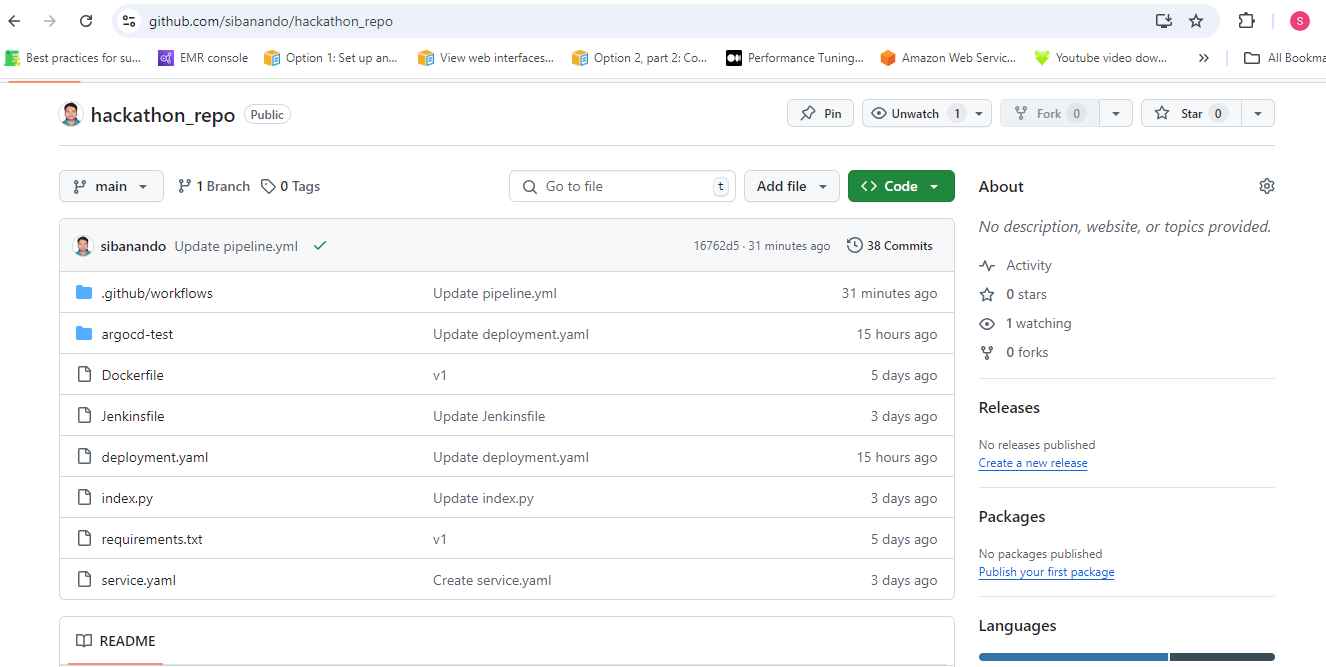
sudo snap install helm --classic

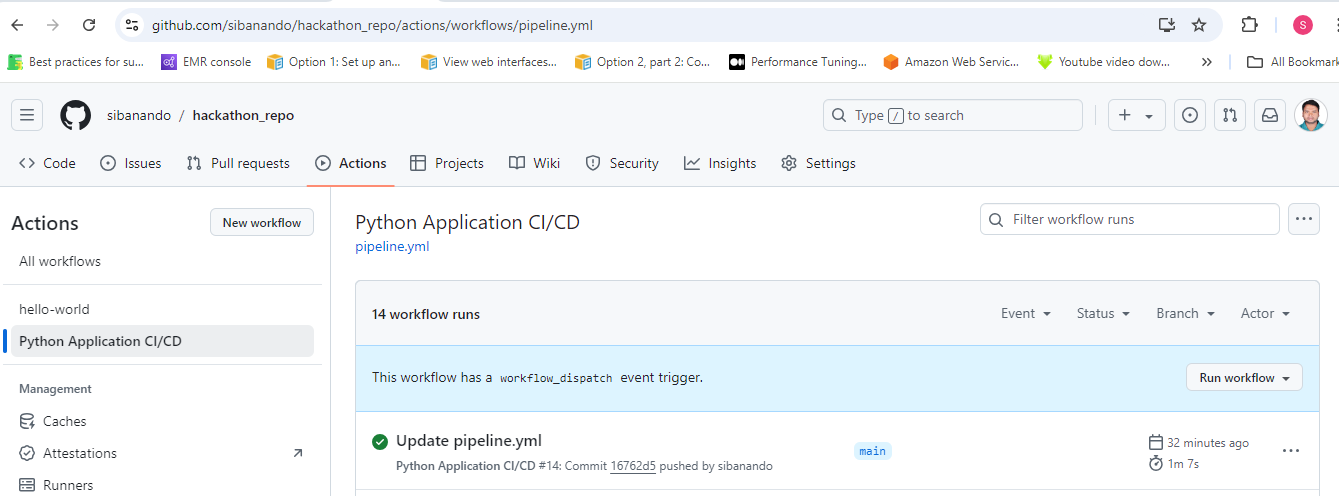
==================

**Github and Github Action**

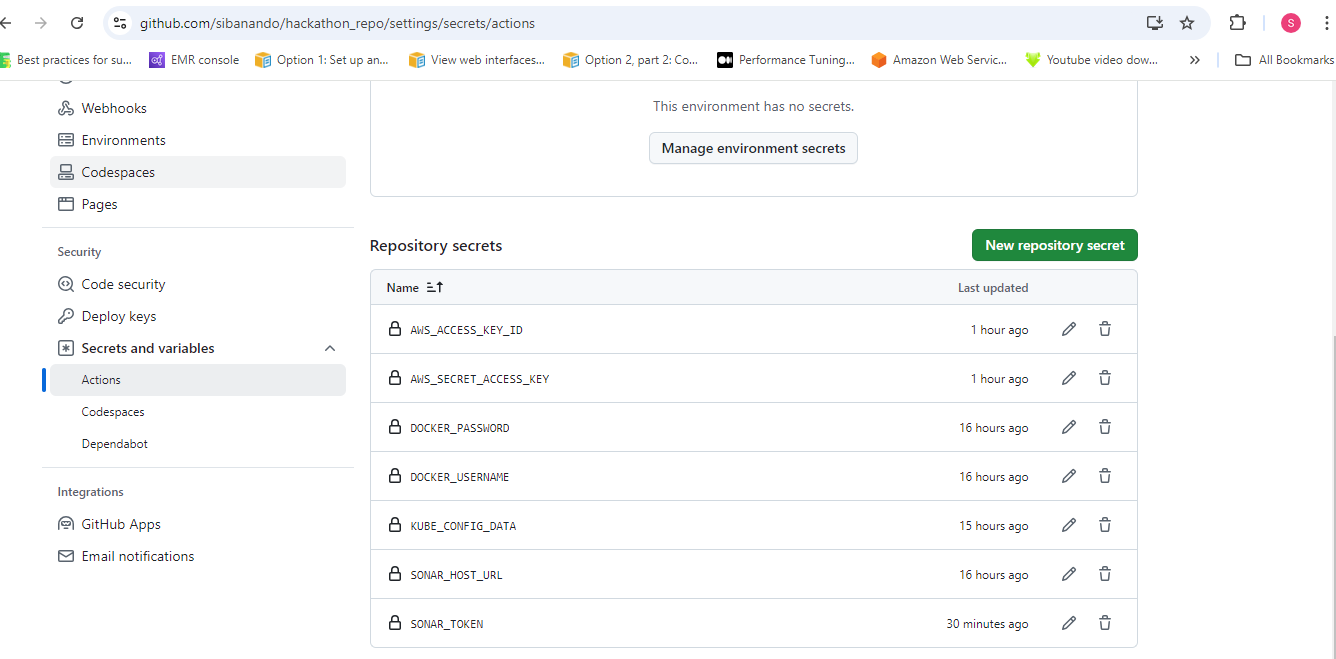
===============

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

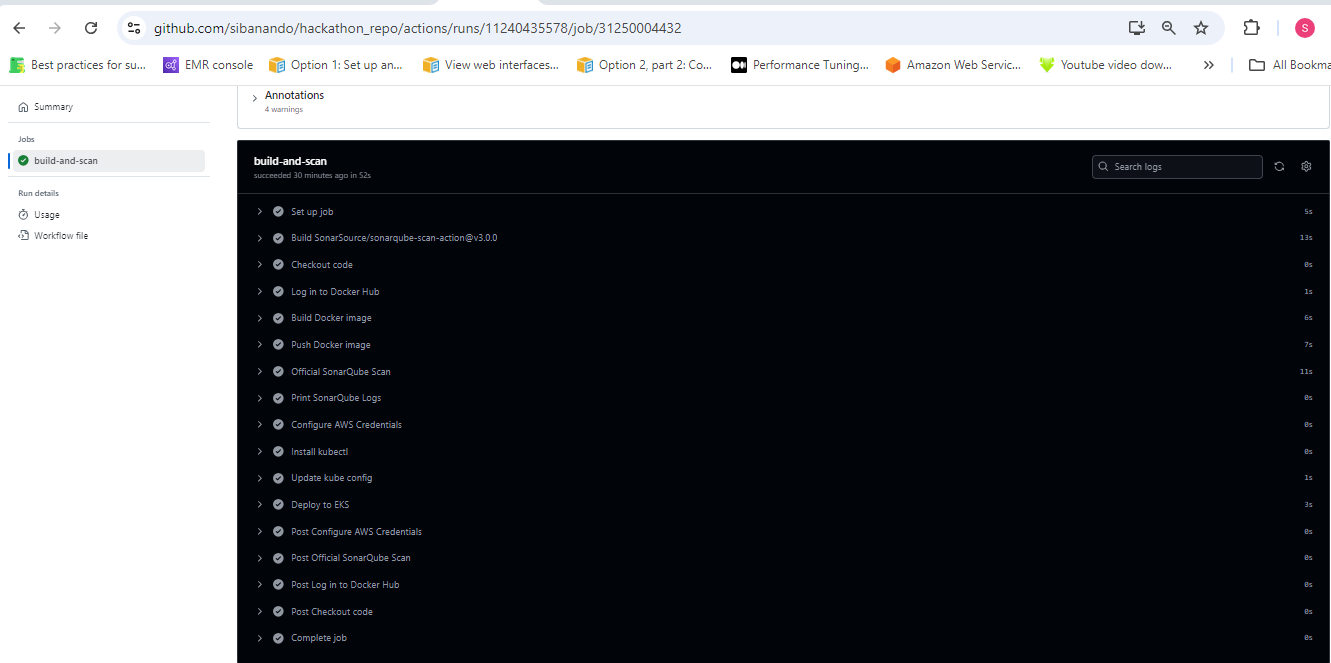
run: |

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

============

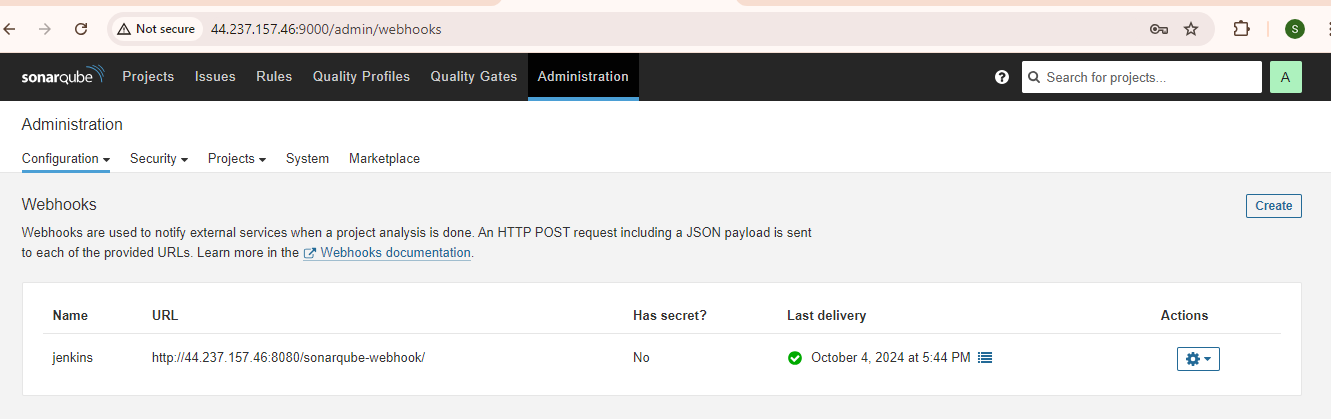
Output :

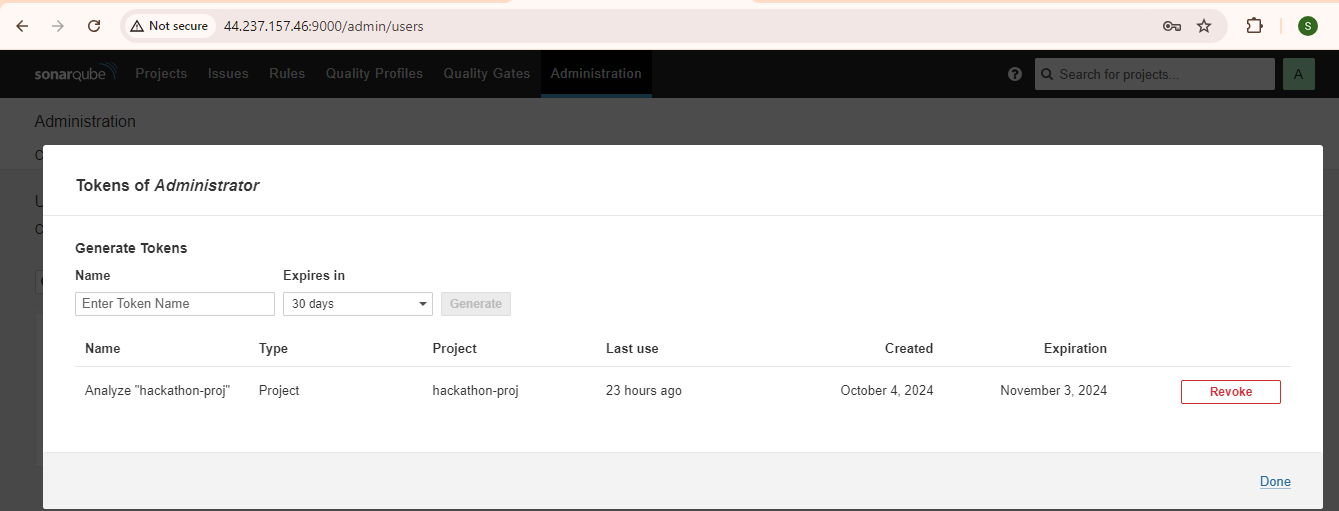


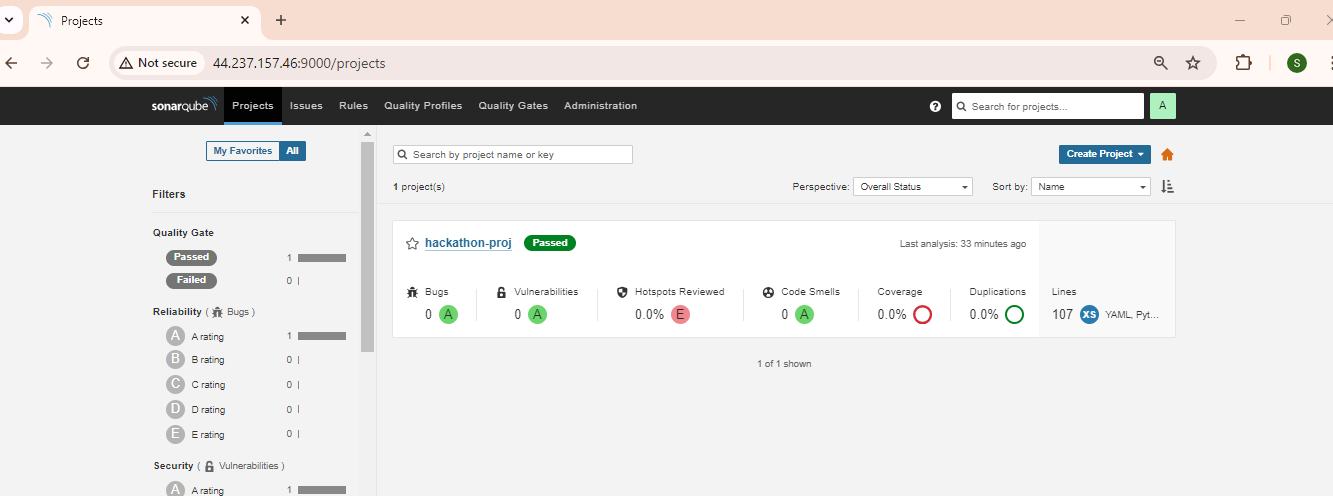
=============

SonarQube:

==================



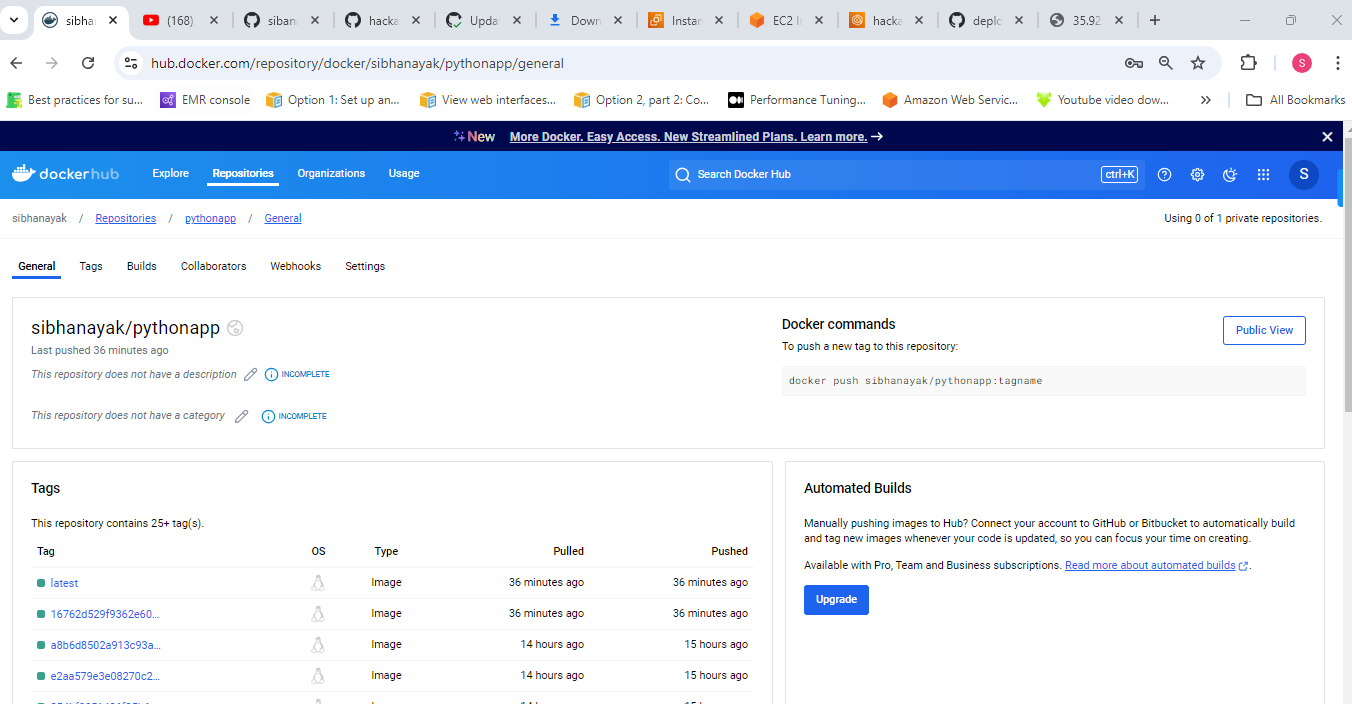




===========

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:~$

