<https://core.digit.org/>  
<https://github.com/egovernments>  
https://jagan.kumar@[egov.org.in/](https://egov.org.in/)  
<https://www.youtube.com/watch?v=unfsCR__O0s&list=PLyQ2-xBpWNe5ufyowv5du6PPwOriOBLHx&index=6>  
<https://digit-discuss.atlassian.net/wiki/spaces/DOPS/pages/112721941/DIGIT+Architecture+and+Technical+overview>

**Project:01 06-01-2024**

**Ci/cd Pipeline Implementation Using Jenkins to deploy Reditt -clone application. For ubuntu 20 LTS**

**Objective :**

**How Should I Start : Work Flow Of Project**

Step:1Terraform to create an EC2 instance for Jenkins, Docker and SonarQube

Step:2 Configure the Jenkins

Step3: Configure SonarQube and Integrate SonarQube with Jenkins

Step4: Create Pipeline Script(Jenkinsfile) and Create CI Job on Jenkins

Step5: Setup Email Notification Through Jenkins

Step6: Create AWS EKS Cluster.

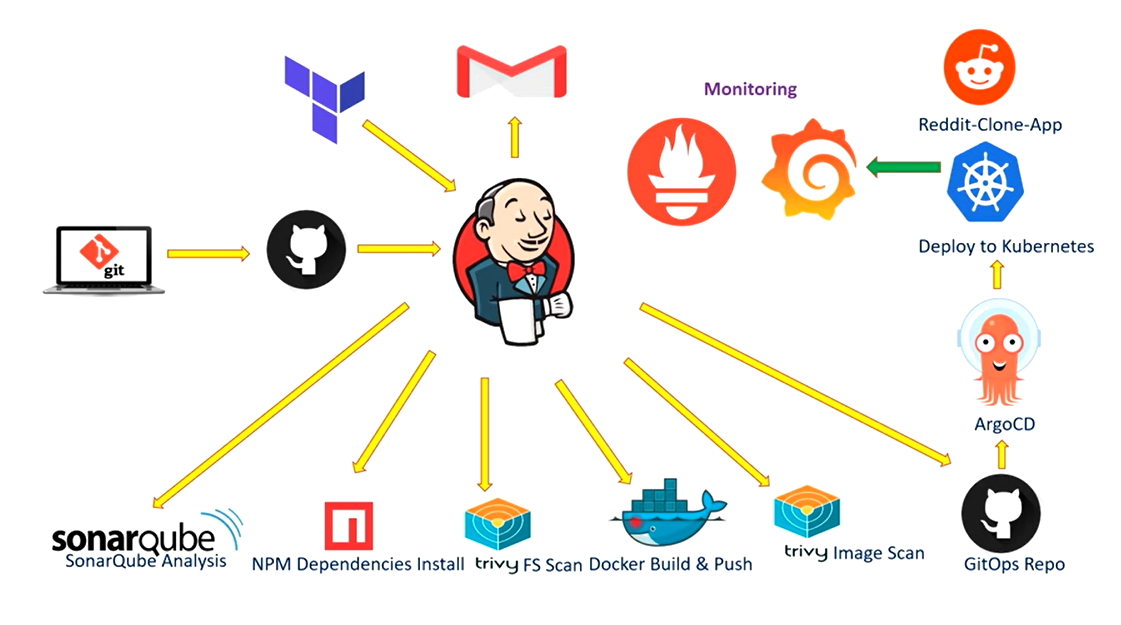
Step 7:Setup Monitoring for Kubernetes using Helm, Prometheus and Grafana Dashboard

Step8: ArgoCD Installation on Kubernetes Cluster and Add AWS EKS Cluster to ArgoCD

Step9: Configure ArgoCD to Deploy Pods on EKS Cluster and Automate ArgoCD Deployment using GitOps GitHub Repository

Step:10 Set the Trigger using GitHub Webhook and Verify the CI/CD Pipeline

**Below The Diagram Representation Of Work Flow Of CI/CD Pipeline Implementation Using Jenkins to Make Reditt-App Clone**

****

**Explanation Of Work Flow**

1.we will used terraform to built ec-2 machine for Jenkins and Sonarqube.

2.If the user makes any change to the git hub repository so that will trigger the CI jobs in Jenkins and CI job will do sonarqube analysis of code .

3.Then It will Install dependencies and do trivy scan then built and push docker image to the docker hub

After that it will scan docker hub image through trivy

Now CI Jobs ends

CI Job Will Push Image To The Docker Hub with different tag Invoved

On every built. Docker Image Push Docker hub by CI job will be having tag according to built number of the Jenkins CI job .

4. When CI job is done it will trigger CD Job Will Actually update images tag on Gitops repo which will have Kubernetes manifest file which will trigger to argoCD which will deploy pods on Kubernetes and this application is Reditt Clone App will setup Kubernetes monitoring using Grafana and Prometheus after completion of CI job the will alert on your gmail

Note : three attachment on mails Trivy Fs Scan,trivy image scan and logistic built.

Now Discus Brief Information About All Processes Involved

1.Terraform (for ec-2 machine)

2.Jenkins

3.github

4.Sonarqube Analysis

5.Dependencies Install

6.Trivy Fs Scan

7.Docker Built And Push

8.Trivy Image Scan

9.Gitops Repo

10.ArgoCD

11.Kubernetes(pods)

12.Reddit Clone-App

Task 1: Terraform to Create an EC2 instance for Jenkins ,Docker and Sonarqube

Terraform Must Be Installed First

Aws Cli and VsCode

Ports used in Security Groups

* 22: SSH (Secure Shell) - used for secure remote administration.
* 80: HTTP (Hypertext Transfer Protocol) - used for regular web browsing.
* 443: HTTPS (Hypertext Transfer Protocol Secure) - used for secure web browsing.
* 8080: HTTP alternate (commonly used for proxy and caching services).
* 9000: Used by various applications, including some web servers and network devices.
* 3000: Generally used for web servers or development environments.

Let's use Terraform to create an EC2 instance for Jenkins, Docker and SonarQube

1--main.tf

resource "aws\_instance" "web" {

ami = "ami-0287a05f0ef0e9d9a" #change ami id for different region

instance\_type = "t2.large"

key\_name = "Linux-VM-Key7" #change key name as per your setup

vpc\_security\_group\_ids = [aws\_security\_group.Jenkins-VM-SG.id]

user\_data = templatefile("./install.sh", {})

tags = {

Name = "Jenkins-SonarQube"

}

root\_block\_device {

volume\_size = 40

}

}

resource "aws\_security\_group" "Jenkins-VM-SG" {

name = "Jenkins-VM-SG"

description = "Allow TLS inbound traffic"

Script To Create EC2 Instance Via Terraform

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-VM-SG"

}

}

2.Provider.tf

2--provider.tf

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

}

}

# Configure the AWS Provider

provider "aws" {

region = "ap-south-1" #change region as per your requirement

}

3. Install.tf

**3--install.sh**

**#!/bin/bash**

**sudo apt update -y**

**wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc**

**echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION\_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list**

**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 | 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-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null**

**sudo apt-get update -y**

**sudo apt-get install jenkins -y**

**sudo systemctl start jenkins**

**sudo systemctl status jenkins**

**##Install Docker and Run SonarQube as Container**

**sudo apt-get update**

**sudo apt-get install docker.io -y**

**sudo usermod -aG docker ubuntu**

**sudo usermod -aG docker jenkins**

**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 | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null**

**echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] 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 -y**

**2. Go to aws**

**IAM…..create user ….attach policy …administrator policy…..security credentials …..access key and secret key**

**Practical View:**

**Step1:**

Configure the Jenkins

1.copy public ip of ec2 instance and browse it port 8080

Go to terminal and open the path(given on Jenkins) and cat write path name and get the password after that configure the Jenkins install suggested plugins

…..Now enters in the Jenkins Dashboard

1.Manage Jenkins …plugin..available plugin(eclipse temurin,sonarqube scanner,nodejs,gates(plugins quality gates),docker,docker commons,pipeline,api,build step and cloudbees docker build and publish) install it

2.go to tools---> nodejs….add nodejs….name(node16) version 16.2.0

Install automatically

Jdk….add jdk…name(jdk17) add installer install from odoptium version jdk17.0.8.1+1

Docker…docker install automatically ….download from docker.com name docker ….latest version

SonarQube-Scanner….add sonarqube scanner….name sonar-scanner…install automatically …..maven version …latest …apply and save.

Step:2 Configure SonarQube and Integrate SonarQube With Jenkins:

1.public ip address and browse with port 9000

2.go to administration ….security user …token (to integrate with Jenkins)

3.go to Jenkins…..manage Jenkins….credentials …add credential kind…secret text…..SonarQube-Token(name) and create

4.go to manage Jenkins….system……SonarQube-Server(under name)…add sonarqube…name(SonarQube-Server)…..url <https://private> ip :9000(because install Jenkins and sonarqube on same ec2 machine)

5.go to sonarqube …..quality gate ….create ….Sonarqube quality gate (name)……..set the webhook on sonarqube for Jenkins server

Configuration…webhook….name jenkins….url…http://privateip address of ec2 instance:8080/sonarqube-webhook/ (create)

Step 3:

Create Pipeline Script(Jenkinsfile) and create Ci Job on Jenkins

1. Add git hub and dockerhub credentials in Jenkins.
2. Github …go to setting..developer setting….create personal access tokens
3. Go to manage Jenkins credential ….kind user name with password ….username: and password:(create)
4. Go to dockerhub create token …go to Jenkins …credentials….add credentials ……username and password …id …dockerhub and create

5.go to sonarqube dashboard ….go to project and display name …Reditt clone ci setup locally …generate continue …linux …

1. Jenkins job is a declarative job we will add jenkinsfile in our repo…..

Jenkinsfile: script

pipeline {

agent any

tools {

jdk 'jdk17'

nodejs 'node16'

}

environment {

SCANNER\_HOME = tool 'sonar-scanner'

APP\_NAME = "reddit-clone-pipeline"

RELEASE = "1.0.0"

DOCKER\_USER = "ashfaque9x"

DOCKER\_PASS = 'dockerhub'

IMAGE\_NAME = "${DOCKER\_USER}" + "/" + "${APP\_NAME}"

IMAGE\_TAG = "${RELEASE}-${BUILD\_NUMBER}"

JENKINS\_API\_TOKEN = credentials("JENKINS\_API\_TOKEN")

}

stages {

stage('clean workspace') {

steps {

cleanWs()

}

}

stage('Checkout from Git') {

steps {

git branch: 'main', url: 'https://github.com/Ashfaque-9x/a-reddit-clone.git'

}

}

stage("Sonarqube Analysis") {

steps {

withSonarQubeEnv('SonarQube-Server') {

sh '''$SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=Reddit-Clone-CI \

-Dsonar.projectKey=Reddit-Clone-CI'''

}

}

}

stage("Quality Gate") {

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'SonarQube-Token'

}

}

}

stage('Install Dependencies') {

steps {

sh "npm install"

}

}

stage('TRIVY FS SCAN') {

steps {

sh "trivy fs . > trivyfs.txt"

}

}

1.go to Jenkins dashboard new item ….redittclone-ci(name)…..pipeline(ok)…discard old build …02…..pipelinescript from scm…git…provide repo(which have jenkinsfile)….branch …main…script path…jenkins path….apply and save……build now

1.after job is completed go to sonarjobs and in project sonarqube analyasisof project and go to issues find issues of project

Now add stage in our pipeline script for docker build and push

Edit jenkinsfile which have pipeline script and add one more stage.

stage("Build & Push Docker Image") {

steps {

script {

docker.withRegistry('',DOCKER\_PASS) {

docker\_image = docker.build "${IMAGE\_NAME}"

}

docker.withRegistry('',DOCKER\_PASS) {

docker\_image.push("${IMAGE\_TAG}")

docker\_image.push('latest')

}

}

}

}

If I go to dockerhub there is no image called Reddit clone and I go to the pipeline and build now

Again go to dockerhub so Reditt clone available here

If I go to image note the tag 1.0.0.2 and if I go to my job I have build number 2

I will add one more stage and the stage will be trivy scan image and this image scan docker hub image which we push to docker hub

stage("Trivy Image Scan") {

steps {

script {

sh ('docker run -v /var/run/docker.sock:/var/run/docker.sock aquasec/trivy image ashfaque9x/reddit-clone-pipeline:latest --no-progress --scanners vuln --exit-code 0 --severity HIGH,CRITICAL --format table > trivyimage.txt')

}

}

}

Now Setup Email Notification through Jenkins

1

your account should be enable for two factors authentication.

app password....vamn pwzt sqsk yqfo

go to jenkins manage jenkins..credentials ...username with password....username....cloudindia001@gmail.com password...app password...create and i will go to manage jenkins ..system...email notification....smtp server....smtp.gmail.com......default user e mail cloudindia001.....advanced setting.....use ssl .....smtp port 465......user name ....cloudindia001....password.....app password....use smtp password......

test configuration check provide recepient and click on test configuration.....extended email notification .....smtp.gmail.com......smtp port 465....advanced....select gmail credentials ....use ssl...default mail suffix....default content type html, default triggers....select always and success ...apply + save

go to jenkins file attached code for email

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

to: 'ashfaque.s510@gmail.com',

attachmentsPattern: 'trivyfs.txt,trivyimage.txt'

}now build on the jobs completed

if go to email find three attachment

1.result of trivy fs scan

2.result of trivy image scan

3.complete build log for our job.

Step:5

Create Aws Eks cluster

1.kubernetes cluster

On Jenkins first install kubectl

Sudo apt update

Sudo apt install curl

curl -LO [https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)(download package for kubectl)

\*sudo apt-get update && sudo apt-get install -y kubectl  
(

To install kubectl)

Kubectl version –client(

…install aws cli

curl -o awscliv2.zip <https://awscli.amazonaws.com/AWSCLIv2.zip>

unzip file unzip awscliv2.zip

sudo apt install unzip

sudo ./aws/install(executable file)

aws –-version

curl -o eksctl -s https://github.com/weaveworks/eksctl/releases/download/<version>/eksctl\_Linux\_amd64.tar.gz | tar -xz -C /tmp(package for eksctl)

cd/tmp

tmp/eksctl/bin

eksctl version

now go to aws console go to iam ….roles….create roles..aws service…..ec2….administrator access…eksctl\_role(name)…..create role…go to ec2 instance….security(action)…modify…iam role….update iam role…role attached to ec2 instance

eksctl create cluster –name virtualtechbox.cluster \

--region us-east-1\

--node.type t2.small

--nodes 3\

Ekscluster creation done

Kubectl get nodes

Kubectl get svc

Step 6

Setup Monitoring For Kubernetes Using Helm,Prometheus and Grafana dashboard

1.first of all install helm chart on system

**$** curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

**$** chmod 700 get\_helm.sh

**$** ./get\_helm.sh

helm version

helm stable chart for local client

helm repo add stable <https://charts.helm.sh/stable>

2.prometheus helm repo

helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

3.create name space using the command

Kubectl create namespace Prometheus

4.install Prometheus using helm with this command

helm install prometheus prometheus-community/kube-prometheus-stack -n Prometheus

5.kubectl get pods -n Prometheus

6.kubectl get svc -n Prometheus

Expose Prometheus to external world

Kubectl edit svc stable-kube-prometheus-sta-prometheus -n Prometheus

Press i

Type ….cluster ip….change to load balancer

:wq enter

Kubectl get srv Prometheus

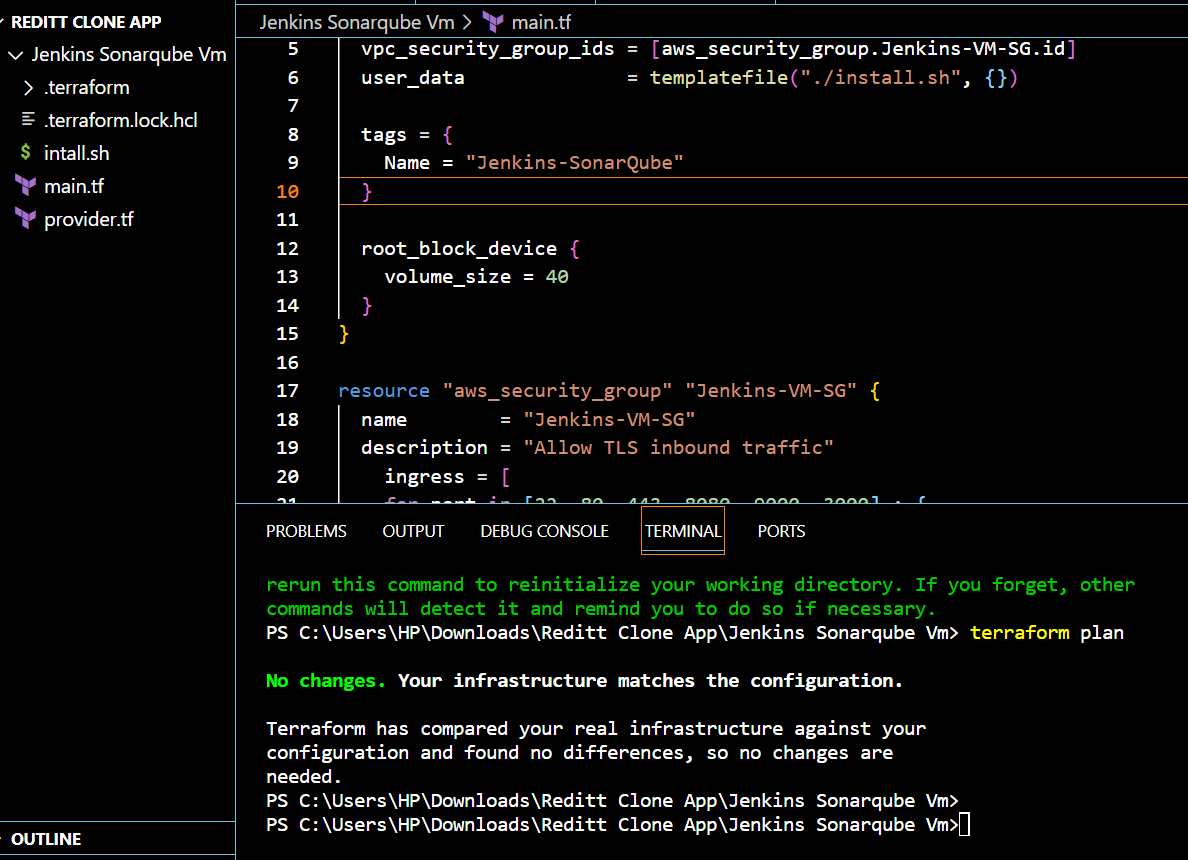
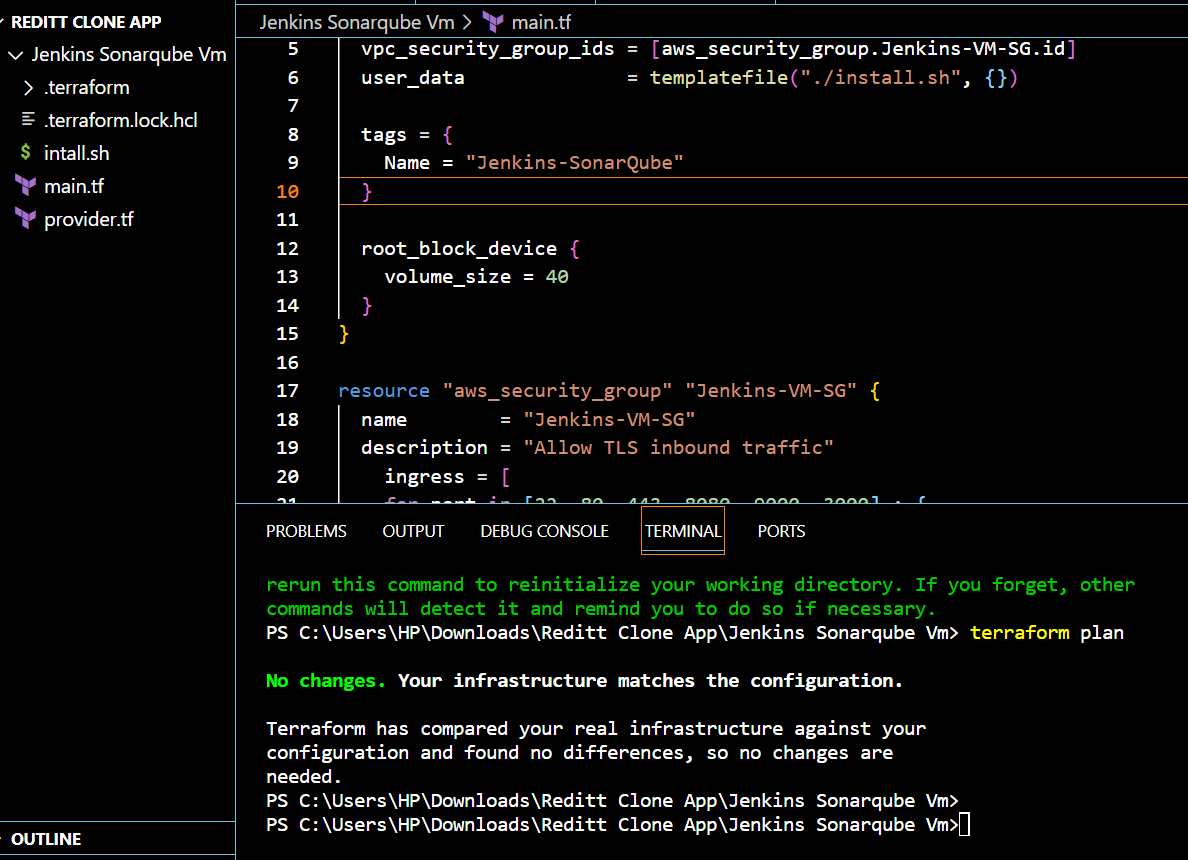
(load balancer which is dna service)

Dns name:9090

Prometheus server opens

Now Grafana(open service file for Grafana)

Kubectl edit svc stable-grafana -n Prometheus

****