**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.

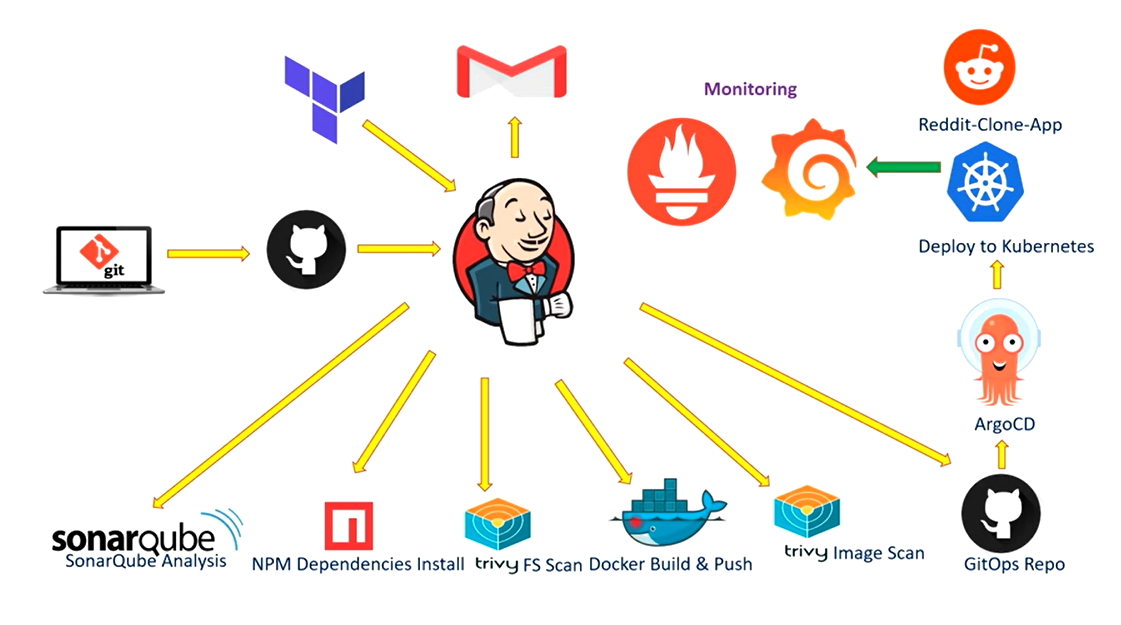
Step 7:Setup Monitoring for Kubernetes using Helm, Prometheus and Grafana Dashboard

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**

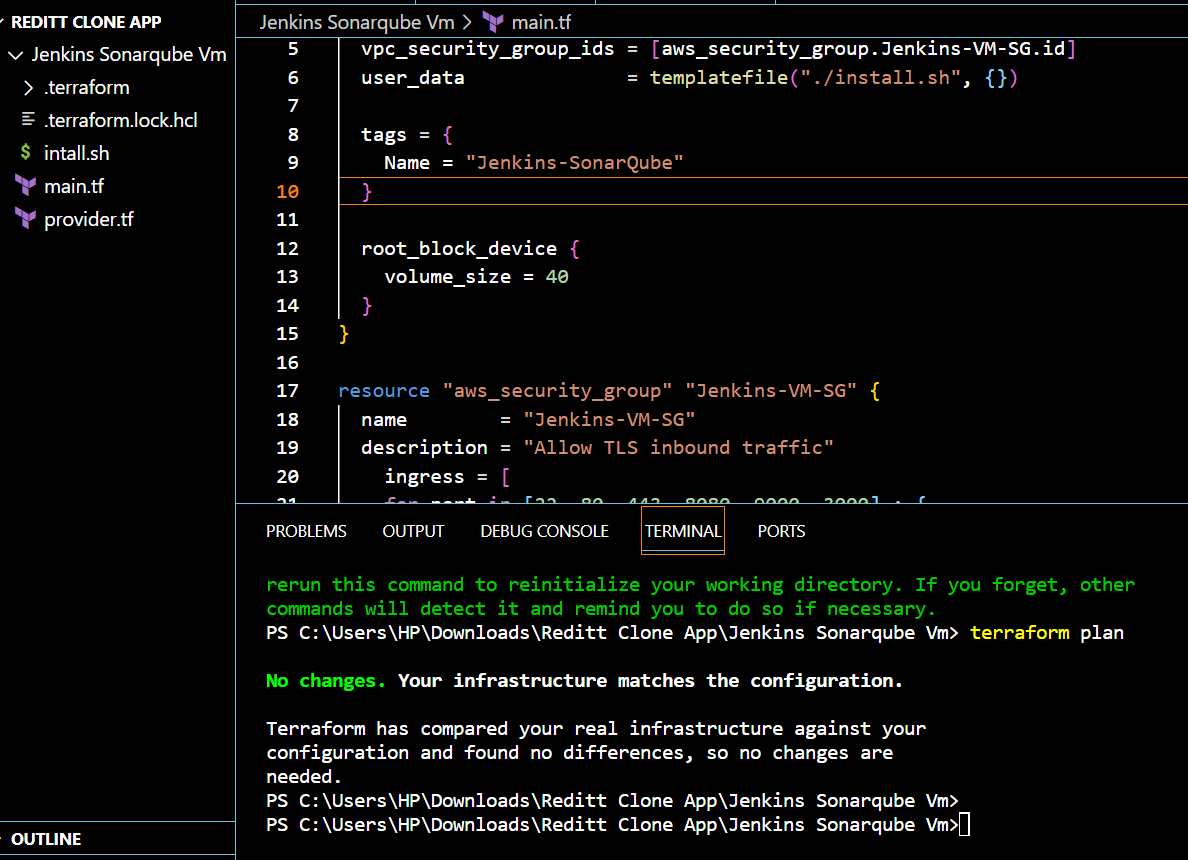
**echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list**

**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**

****