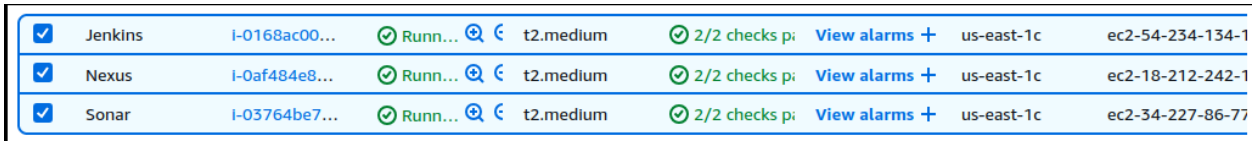
1. Setting up a multi-tier CI/CD pipeline for a Java application using MySQL with Jenkins, Nexus, and SonarQube involves several steps.

1. Environment Setup

Ensure all tools are installed and configured:

- **Jenkins**: Set up on a server (preferably with an agent/master configuration).
- **Nexus**: Install and configure as an artifact repository.
- **SonarQube**: Install and set up for code analysis.
- **MySQL**: Set up a database server for the application



2. **Steps to Create an EKS Cluster Using Terraform**
1. **Setup and Prerequisites**

    - Install Terraform.
    - Install AWS CLI and configure it with the appropriate IAM user/role credentials.
    - Ensure your AWS account has the necessary permissions to create EKS resources (e.g., VPC, subnets, EKS cluster, IAM roles).

Define Your Terraform Configuration

Create a new directory for your Terraform files and add the following main components:

Main Configuration File (e.g., `main.tf`):

```
provider "aws" {
  region = "us-east-1"  # Replace with your desired region
}

module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "~> 3.0"

  name = "eks-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["us-east-1a", "us-east-1b"]
  public_subnets  = ["10.0.1.0/24", "10.0.2.0/24"]
  private_subnets = ["10.0.3.0/24", "10.0.4.0/24"]
```

```
  enable_nat_gateway = true
  tags = {
    "Name" = "eks-vpc"
  }
}

module "eks" {
  source          = "terraform-aws-modules/eks/aws"
  version         = "~> 19.0"

  cluster_name    = "my-eks-cluster"
  cluster_version = "1.26"  # Replace with the desired Kubernetes version

  subnets         = module.vpc.private_subnets
  vpc_id          = module.vpc.vpc_id

  node_groups = {
    eks_nodes = {
      desired_capacity = 2
      max_capacity     = 3
      min_capacity     = 1

      instance_type = "t3.medium"

      key_name = "my-key-pair"  # Replace with your EC2 key pair name

      tags = {
        Name = "eks-node"
      }
    }
  }

  tags = {
    "Environment" = "dev"
    "Team"        = "DevOps"
  }
}
```

2. **Create Variables File (e.g., `variables.tf`)**

   Define your reusable variables:

   ```
   variable "aws_region" {
   ```

```
  description = "AWS region"
  default     = "us-east-1"
}

variable "cluster_name" {
  description = "Name of the EKS cluster"
  default     = "my-eks-cluster"
}

variable "vpc_cidr" {
  description = "CIDR block for the VPC"
  default     = "10.0.0.0/16"
}
```

3. **Create Outputs File (e.g., `outputs.tf`)**

   Define the outputs to extract useful data:

```
output "cluster_endpoint" {
  description = "EKS cluster endpoint"
  value       = module.eks.cluster_endpoint
}

output "cluster_security_group_id" {
  description = "Security Group ID of the EKS cluster"
  value       = module.eks.cluster_security_group_id
}

output "node_group_role_arn" {
  description = "ARN of the worker node IAM role"
  value       = module.eks.node_groups["eks_nodes"].iam_role_arn
}
```

4. **Initialize Terraform** Run the following command to download necessary provider plugins and modules:

   ```
   terraform init
   ```

5. **Validate the Configuration** Check the configuration syntax and validate it using:

   ```
   terraform validate
   ```

6. **Preview the Changes** Generate a plan to see what resources Terraform will create:

   ```
   terraform plan
   ```

7. **Apply the Configuration** Deploy the EKS cluster and associated resources:

   terraform apply
8. Confirm with `yes` when prompted.

9. **Access Your EKS Cluster**

Update your local Kubernetes configuration to access the cluster:
aws eks update-kubeconfig --region us-east-1 --name my-eks-cluster

-

Verify connectivity:
kubectl get nodes

-

Clean Up Resources If you want to destroy the created resources:

terraform destroy

10. Confirm with `yes` when prompted.

Notes:

- Replace placeholders (e.g., `us-east-1`, `my-key-pair`) with your specific values.
- Use Terraform state management practices to ensure proper resource tracking.
- Add backend configuration for state storage if you plan to collaborate with a team.

3. CI/CD Workflow Overview

The pipeline involves:

1. **Code Build**: Compile and package the Java application.
2. **Static Code Analysis**: Run SonarQube for quality checks.
3. **Artifact Management**: Upload build artifacts to Nexus.
4. **Deployment**: Deploy artifacts to testing/staging/production environments.
5. **Database Integration**: Apply database migrations via tools like Flyway.

4. Detailed Steps

A. Jenkins Configuration

1. **Install Plugins:**
    - SonarQube Scanner
    - Nexus Artifact Uploader
    - Git/GitHub integration
    - Pipeline (Declarative or Scripted)
2. **Configure Tools in Jenkins:**
    - Add JDK, Maven, and SonarQube scanner in Jenkins' global tool configuration.
    - Configure SonarQube and Nexus credentials in Jenkins.

3. **Nexus**: Install and configure as an artifact repository.
   sudo apt update -y
   sudo apt install docker.io -y
   sudo docker run -d --name nexus -p 8081:8081 sonatype/nexus3
   http://18.212.242.13:8081
   Your **admin** user password is located in
   /nexus-data/admin.password on the server.
   sudo docker ps
   sudo docker exec -it 31cc96dafaa3 /bin/bash
   cat /nexus-data/admin.password

4. **SonarQube**: Install and set up for code analysis.
   sudo apt update -y
   sudo apt install docker.io -y
   sudo docker run -d --name sonardevops -p 9000:9000 sonarqube:lts-community

5. **Jenkins**: Set up on a server (preferably with an agent/master configuration)
   Installation commands, first java we should install
   sudo apt update
   sudo snap install openjdk
   java -version
   openjdk version "17.0.13" 2024-10-15
   OpenJDK Runtime Environment (build 17.0.13+11-Ubuntu-2ubuntu124.04)
   OpenJDK 64-Bit Server VM (build 17.0.13+11-Ubuntu-2ubuntu124.04, mixed mode, sharing)

   sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
     https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
   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
   sudo apt-get install jenkins

sudo systemctl start jenkins
sudo systemctl enable jenkins
sudo systemctl status jenkins

A. Jenkins Configuration

1. **Install Plugins:**
    - SonarQube Scanner
    - Nexus Artifact Uploader
    - Git/GitHub integration
    - Pipeline (Declarative or Scripted)
2. **Configure Tools in Jenkins:**
    - Add JDK, Maven, and SonarQube scanner in Jenkins' global tool configuration.
    - Configure SonarQube and Nexus credentials in Jenkins.

Latest docker install in ubuntu jenkins server

Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

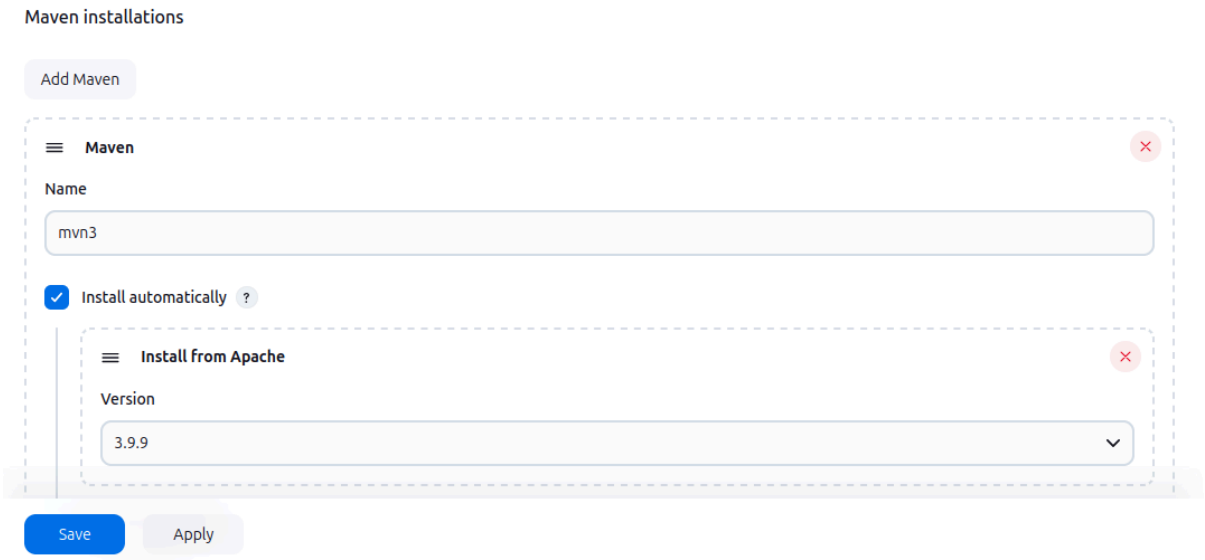
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

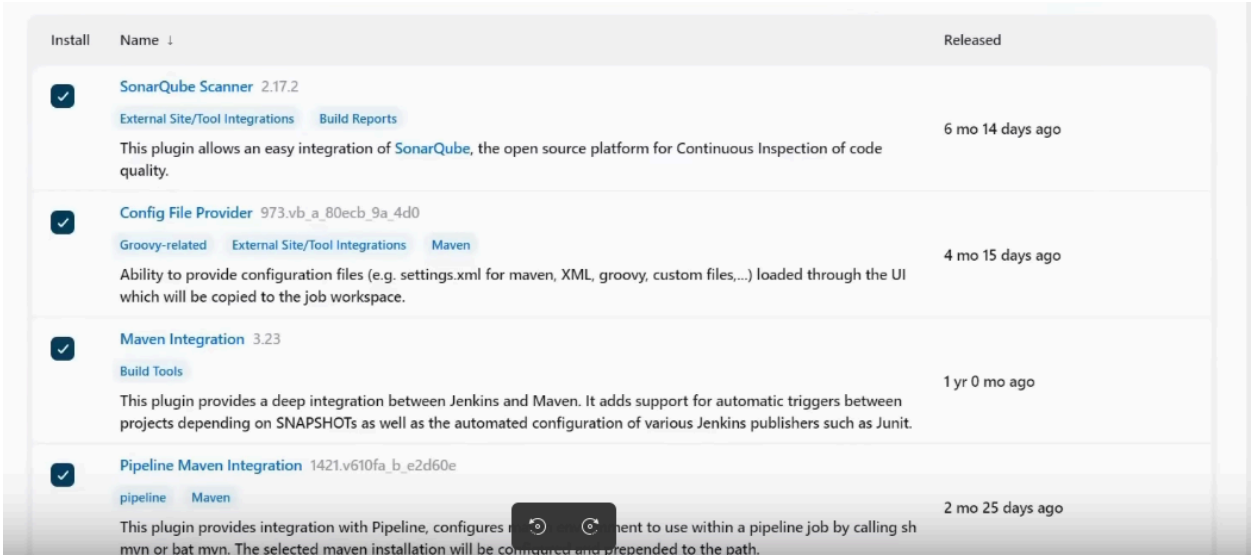
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo chmod 666 /var/run/docker.sock

- 

3. 

Pipeline Structure

1. **pipeline**
    - The pipeline block defines the entire Jenkins Declarative Pipeline.
    - It includes global configurations such as agent, tools, environment, and multiple stages.

Global Settings

2. **agent any**
    - Specifies that the pipeline can run on any available Jenkins agent.
3. **tools**
    - Configures the tools required for the pipeline.
        - `maven 'mvn3'`: Sets Maven version `mvn3` for the pipeline.
4. **environment**
    - Declares environment variables for the pipeline.
        - `SCANNER_HOME = tool 'sonar'`: Configures SonarQube Scanner and assigns its path to `SCANNER_HOME`.

Stages

Stage 1: Git Checkout

- **Purpose**: Clone the source code repository.

4. **Steps**:
   git branch: 'main', url: 'https://github.com/mohammadshoaib8/Multi-Tier-With-Database.git'

-
    - `branch: 'main'`: Checks out the main branch of the GitHub repository.
    - `url`: Specifies the repository URL.

Stage 2: Compile

- **Purpose**: Compile the Java code.

5. **Steps**:
   sh "mvn compile"

-
  - Runs the Maven compile command, which compiles the source code.

Stage 3: Test

- **Purpose**: Run unit tests (currently skipping them).

6. **Steps**:
   sh "mvn test -DskipTests=true"

-
  - `mvn test`: Executes unit tests.
  - `-DskipTests=true`: Skips running the tests.

Stage 4: Trivy FS Scan

- **Purpose**: Perform a security scan on the file system using Trivy.

7. **Steps**:
   sh "trivy fs --format table -o fs-report.html ."

-
  - `trivy fs`: Scans the file system for vulnerabilities.
  - `--format table`: Formats the output as a table.
  - `-o fs-report.html`: Saves the scan report to `fs-report.html`.

Stage 5: SonarQube Analysis

- **Purpose**: Perform static code analysis using SonarQube.

8. **Steps**:
   withSonarQubeEnv('sonar') {
9. sh "$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=multitier -Dsonar.projectKey=multitier -Dsonar.java.binaries=target"
10. }

-
  - `withSonarQubeEnv('sonar')`: Provides the SonarQube environment.
  - `$SCANNER_HOME/bin/sonar-scanner`: Executes the SonarQube Scanner binary.
  - `-Dsonar.projectName=multitier`: Sets the project name in SonarQube.
  - `-Dsonar.projectKey=multitier`: Assigns a unique key to the project.
  - `-Dsonar.java.binaries=target`: Specifies the compiled binaries location.

Stage 6: Build

- **Purpose**: Build the application into a package.

11. **Steps**:
    sh "mvn package -DskipTests=true"
-
    - `mvn package`: Creates the deployable artifact (e.g., .jar or .war).
    - `-DskipTests=true`: Skips running tests during the build.

Stage 7: Publish to Nexus Repository

- **Purpose**: Deploy the built package to a Nexus repository.
12. **Steps**:
    withMaven(globalMavenSettingsConfig: 'settings-maven', maven: 'mvn3') {
13. sh "mvn deploy -DskipTests=true"
14. }
-
    - `withMaven`: Configures Maven settings for deploying to Nexus.
    - `mvn deploy`: Uploads the built package to Nexus.
    - `-DskipTests=true`: Skips running tests during deployment.

Stage 8: Docker Build Image

- **Purpose**: Build a Docker image for the application.
15. **Steps**:
    withDockerRegistry(credentialsId: 'docker-cred') {
16. sh "docker build -t msshoaib2255457/bankApp:latest ."
17. }
-
    - `withDockerRegistry`: Authenticates with the Docker registry using `docker-cred`.
    - `docker build`: Builds the Docker image.
    - `-t msshoaib2255457/bankApp:latest`: Tags the image as `bankApp:latest`.

Stage 9: Trivy Scan Image

- **Purpose**: Scan the Docker image for vulnerabilities using Trivy.
18. **Steps**:
    sh "trivy image --format table -o image-report.html msshoaib2255457/bankApp:latest"
-
    - `trivy image`: Scans the Docker image.
    - `--format table`: Formats the output as a table.
    - `-o image-report.html`: Saves the scan report to `image-report.html`.

Stage 10: Docker Push Image

- **Purpose**: Push the Docker image to the Docker registry.

19. **Steps**:

```
withDockerRegistry(credentialsId: 'docker-cred') {
    sh "docker push msshoaib2255457/bankApp:latest"
}
```

-
    - `docker push`: Uploads the image to the registry.

Stage 11: K8S Deploy

- **Purpose**: Deploy the application to Kubernetes.

22. **Steps**:

```
withKubeConfig(
    credentialsId: 'k8s-cred',
    namespace: 'webapps',
    serverUrl: 'https://840EBDE3727CA221BB19C036BC654334.gr7.us-east-1.eks.amazonaws.com'
) {
    sh "kubectl apply -f ds.yml -n webapps"
    sleep 30
}
```

-
    - `withKubeConfig`: Configures Kubernetes credentials.
    - `kubectl apply -f ds.yml`: Deploys the application using the `ds.yml` manifest file.
    - `-n webapps`: Targets the `webapps` namespace.
    - `sleep 30`: Waits for 30 seconds to ensure deployment stability.

30. **Start the pipeline**:

```
pipeline {
        agent any

        tools {
        maven 'mvn3'  // Corrected "tool" to "tools"
        }

        environment {
        SCANNER_HOME = tool 'sonar'
        }

        stages {
        stage('Git Checkout') {
```

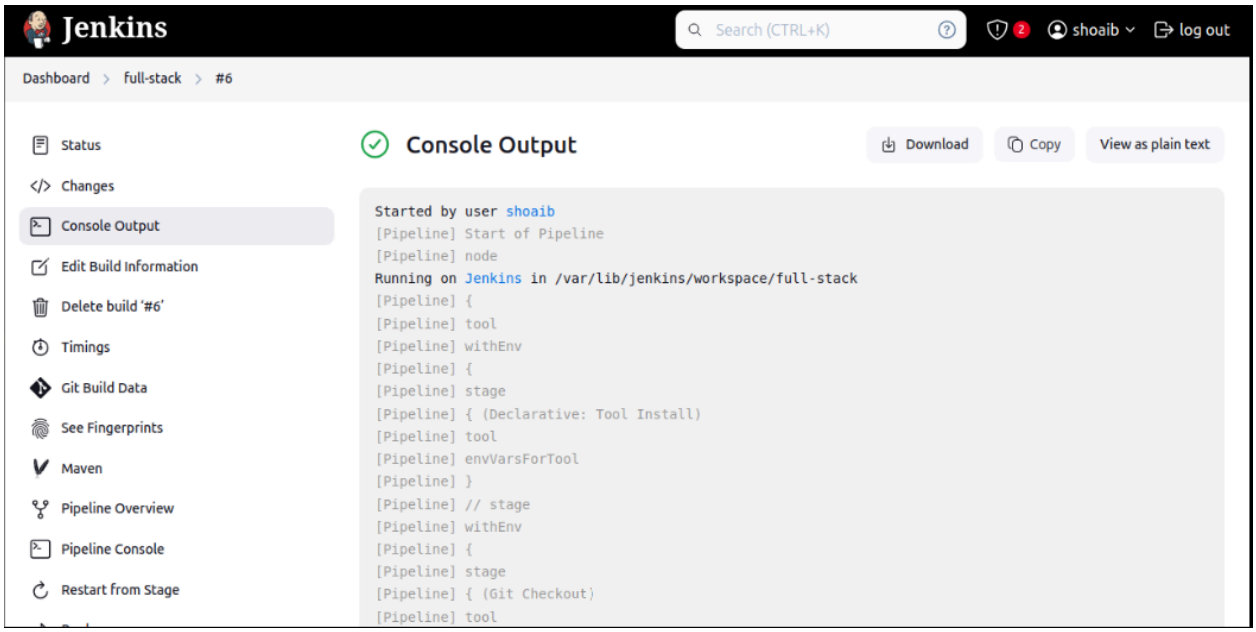
```
        steps {
                git branch: 'main', url:
'https://github.com/mohammadshoaib8/Multi-Tier-With-Database.git'
        }
        }
        stage('Compile') {
        steps {
                sh "mvn compile"
        }
        }
        stage('Test') {
        steps {
                sh "mvn test -DskipTests=true"
        }
        }
        stage('Trivy FS Scan') {
        steps {
                sh "trivy fs --format table -o fs-report.html ."
        }
        }
        stage('SonarQube Analysis') {
        steps {
                withSonarQubeEnv('sonar') {
                sh "$SCANNER_HOME/bin/sonar-scanner
-Dsonar.projectName=multitier -Dsonar.projectKey=multitier
-Dsonar.java.binaries=target"
                }
        }
        }
        stage('Build') {
        steps {
                sh "mvn package -DskipTests=true"
        }
        }
        stage('Publish to Nexus Repository') {
        steps {
                withMaven(globalMavenSettingsConfig: 'settings-maven', maven:
'mvn3') {
                sh "mvn deploy -DskipTests=true"
                }
        }
        }
        stage('Docker Build Image') {
        steps {
```

```
                script {
                withDockerRegistry(credentialsId: 'docker-cred') {
                sh "docker build -t msshoaib2255457/bankApp:latest ."
                }
                }
        }
        }
        stage('Trivy Scan Image') {
        steps {
                sh "trivy image --format table -o image-report.html
msshoaib2255457/bankApp:latest"
        }
        }
        stage('Docker Push Image') {
        steps {
                script {
                withDockerRegistry(credentialsId: 'docker-cred') {
                sh "docker push msshoaib2255457/bankApp:latest"
                }
                }
        }
        }
        stage('K8S Deploy') {
        steps {
                withKubeConfig(
                credentialsId: 'k8s-cred',
                namespace: 'webapps',
                serverUrl:
'https://840EBDE3727CA221BB19C036BC654334.gr7.us-east-1.eks.amazonaws.co
m'
                ) {
                sh "kubectl apply -f ds.yml -n webapps"
                sleep 30
                }
        }
        }
        }
}
```

Install trivy to scan:

sudo apt-get install wget apt-transport-https gnupg lsb-release
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-get update
sudo apt-get install trivy

31. 

32.