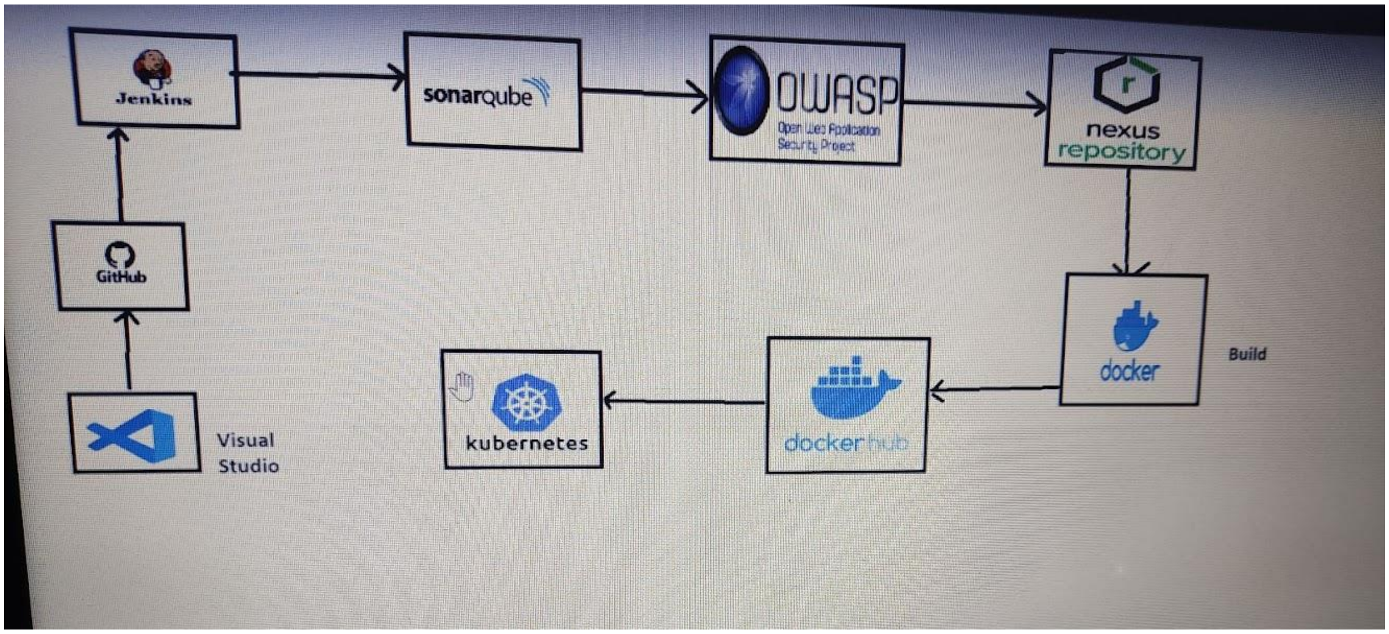
Client = ubs ,braclys ,sbi

# 1.Project-DevSecOps  (3 Tier - Java Based Application)

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



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

**Step 1: Automating:- installation**

A. git clone [https://github.com/ankit-jagtap-devops/terraform-vpc-jenkins-DB.git](https://github.com/ankit-jagtap-devops/terraform-vpc-jenkins-DB.git)
B. aws configure
C. Give access key and secret key
D. Terraform init
E. Terraform plan
F. Terraform apply

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

## Step 1.1:Manual :- create 3 EC2 instances with configurations  ubuntu, T2-Medium

A. Jenkins
B. SonarQube
C. Nexus

## Step 1.2: creating Jenkins Manually

1. # apt install openjdk-17-jre-headless

2. # 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

3. # sudo apt-get update

4. # sudo apt-get install Jenkins

**install docker on jenkins**

# apt install docker.io -y

## Step 1.3: creating SonarQube Manually

**install sonar by using docker**

1. apt update -y
2. apt install docker.io
3. docker pull sonarqube:latest
4. docker run -d -p 9000:9000 sonarqube:latest

## Step 1.4: creating nexus Manually

1. apt update -y
2. Install docker

3. apt install docker.io -y
4. docker run -d -p 8081:8081 sonatype/nexus3

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

## Step 2: login to Sonar nexus and Jenkins

a. For sonar : copy the IP address of Sonar Ec2 and paste on bowser (ex. 13.234.223.39:9000)
   A. Id => admin password => admin
   B. Change the password - > update

b. For Nexus: copy the IP address of Nexus Ec2 and paste on bowser (ex.12.123.15.233:8081)
   A. Take access of it on bash
   B. **docker ps** (you will get container ID for this command)
   C. **docker exec -it <ContainerID>/bin/bash** (gives password to add while login)
   D. ls
   E. cd/
   F. ls
   G. cd nexus-data/
   H. ls
   I. cat admin.password
   J. You will get password here (ex.7a27ddd6-80cd-46bf-8b7d-1bb25d1a234e )
   K. Now login on the browser with the id → admin and password that you got above.
   L. Give new password
   M. Enable anonymous access -> next -> finish

c. For Jenkins: copy the IP address of jenkins Ec2 and paste on bowser (ex.11.13.150.253:8080)

##################################################################################################

## Step 3: GO on Jenkins and follow these steps.

**A. Installing and Configuring Plugins**

1. jenkins -> manage jenkins - > plugins -> Available plugins -> Install below Plugin

a. SonarQube Scanner
b. Nexus Artifact Uploader
c. Docker
d. docker-build-step
e. docker pipeline
f. OWASP Dependency-Check
g. Eclipse Temurin installer
h. Pipeline Maven Integration

2. Select option Restart Jenkins .

3. **now configure tool that we have install plugin**

a. **for JDK**

Manage jenkins - -> tools - -> JDK installation => jdk-17 --> install automatically - ->
Install from adoptium.net --> jdk17.09+9

b. **For SonarQube**

SonarQube configure scanner -> Add sonar scanner -> name => Sonar scanner -> default version

**c. For Maven**

Maven Installations -> install automatically -> version => 3- 3.6.3

**d. Dependency check**

dependency check installation -> name => DC -> install automatically -> add installer => dependency-check 6.5.1

**e.Docker**

Docker Installations -> name => docker -> install automatically -> add installer => docker version => latest

4. Apply and save

**B. Creating Pipeline for Kubernetes cluster**

a. New item -> item name => infra-Deployment -> select pipeline -> OK
b. Pipeline -> pipeline script from SCM -> SCM => Git -> Repository URL = https://github.com/ankit-jagtap-devops/terraform-eks-nodegroup.git -> branch => master -> script path => JenkinsPath -> Apply -> save
c. Build the pipeline

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

## Step 4: Creating the Sonar Token

A. Administrator -> Security -> User -> Token -> Token_name => sonar -> generate
B. Copy the token (ex squ_cd243b280afc47f4c8c36f357c0c48d4fe26bce7 )

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

## Step 5: Create credentials for sonar-token , dockerhub-pwd and docker-hub

a. Jenkins → manage jenkins → security → credentials → system -→ global credentials(unrestricted ) → add credential → secrete text → secrete ⇒(copy the above generated token of sonar) → ID ⇒sonar-token → create
b. Credentials → kind ⇒username with password → scope ⇒global username ⇒ (docker hub username) → password ⇒(password of docker hub) → ID ⇒docker-cred

c. Credentials → kind ⇒secret text → scope ⇒global → secret ⇒(password of docker hub) → ID ⇒dockerhub-pwd
d. in ekart repo JenkinsFile match this name and also change the docker hub name to yours in the docker hub push step
e. Also match the cluster name from aws and jenkins file in EKS and kubectl cluster.

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

## Step 6: Create pipeline for Deployment

a. New pipeline → name ⇒ Deployment → pipeline → Ok

b. Pipeline → pipeline script → script ⇒add below pipeline here → Apply → save

```
pipeline {
agent any

environment {
    SCANNER_HOME = tool 'sonar-scanner'
}

tools {
    maven 'maven3'
    jdk 'jdk-17'
}

stages {
    stage('git checkout') {
        steps {
            git branch: 'main', url: 'https://github.com/Kausar1011/Ekart.git'
        }
    }

    stage('compile') {
        steps {
            sh "mvn compile"
        }
    }

    stage('unit tests') {
        steps {
            sh "mvn test -DskipTests=true"
        }
    }
```

```
        stage('SonarQube analysis') {
            steps {
                withSonarQubeEnv('sonar') {
                    sh "${env.SCANNER_HOME}/bin/sonar-scanner \
                        -Dsonar.projectKey=EKART \
                        -Dsonar.projectName=EKART \
                        -Dsonar.java.binaries=target/classes"
                }
            }
        }

        stage('OWASP Dependency Check') {
            steps {
                dependencyCheck additionalArguments: '--scan ./', odcInstallation: 'DC'
                dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
            }
        }

        stage('Build') {
            steps {
                sh "mvn package -DskipTests=true"
            }
        }

        stage('deploy to Nexus') {
            steps {
                withMaven(globalMavenSettingsConfig: 'global-Maven', jdk: 'jdk-17', maven:
'maven3', mavenSettingsConfig: '', traceability: true) {
                    sh "mvn deploy -DskipTests=true"
                }
            }
        }

        stage('build and Tag docker image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred') {
                        sh "docker build -t kausar/ekart:latest -f docker/Dockerfile ."
                    }
                }
            }
        }

        stage('kubernetes Deploy') {
            steps {
                script {
                    // Replace with your Kubernetes deployment commands
                    // Example: kubectl apply -f deployment.yaml
                    sh "kubectl apply -f deployment.yaml"
                }
            }
        }
    }
}
```

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

## Step 7: Integrating of sonar and Nexus with jenkins

a. **Jenkins - Nexus - Integration**

1. Deployment (our pipeline ) → configure → Manage Jenkins → Managed files → Add new config → global-maven setting.xml → ID ⇒global-Maven → next

2. Content → Line -119 →

```
<server>
            <id>deploymentRepo</id>
            <username>repouser</username>
            <password>repopwd</password>
</server>
-->
```

3. Go to nexus → maven release → copy → from repo copy last folder (ex maven-release ) and copy that to above given code in id.

4. Give the username and password of the nexus .

5. Example
```
<server>
        <id>maven-release</id>
        <username>admin</username>
        <password>admin@123</password>

</server>
```

6. Go to nexus → maven snapshots→ copy → from repo copy last folder (ex maven-snapshot ) and copy that to above given code in id.
7. Example
```
        <server>
        <id>maven-snapshot</id>
        <username>admin</username>
        <password>admin@123</password>

</server>
```
8. Submit
9. In repo go to [https://github.com/Kausar1011/Ekart.git](https://github.com/Kausar1011/Ekart.git)
10. Make this changes in pom.xml
11. Give the url of the nexus maven-release and maven snapshots.

**b. Jenkins - Sonar - Integration**

Deployment (our pipeline ) → configure → Manage Jenkins → System → SonarQube installations → name ⇒sonar —>

Server url ⇒ give url of sonar server → server authentication server ⇒sonar token → apply and save

**c. Build now**

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

## Step 7: For deleting resources

A. Delete cluster first
B. Terraform destroy

Nore : the project will be the same in the company as well ,just the difference will be that we will not use docker hub we will use  nexus for image  or azure container registry elastic container.
==================================================================================================

### Step 8: to see your application

a. AWS → load balancer →  copy the DNS name → go on browser
b. <DNS name>:8070

****************************************************************************************************************
****************************************************************************************************************

## 2. Project

1  Install visual studio, AWS CLI & Terraform.

2. Create S3 bucket.

https://github.com/ankit-jagtap-devops/terraform-s3-bucket.git

3. Create a EC2 instance and install jenkins, maven, terraform, k8s, git.

https://github.com/ankit-jagtap-devops/terraform-vpc-jenkins-DB.git

4. Create a K8s cluster

https://github.com/ankit-jagtap-devops/terraform-eks-nodegroup.git

5. K8s cluster configuration and deployment

https://github.com/ankit-jagtap-devops/devops-cicd-ygminds

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

## 3. Steps to create a helm chart.

**1.Create An EC2 Instance and take the access of the instance.**

**2. Install kubectl - follow the below document to install Kubectl**

https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/

a. ```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/arm64/kubectl"
```

b. ```
curl -LO https://dl.k8s.io/release/v1.30.0/bin/linux/arm64/kubectl
```

c. ```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/arm64/kubectl.sha256
```

d. `echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check`
e. `sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl`
f. `chmod +x kubectl`
g. `mkdir -p ~/.local/bin`
h. `mv ./kubectl ~/.local/bin/kubectl`
i. `kubectl version --client`

**2. Install helm**

https://helm.sh/docs/intro/install/

a. ```
$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
```

b. `$ chmod 700 get_helm.sh`
c. `$ ./get_helm.sh`
d. `curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null`
e. `sudo apt-get install apt-transport-https --yes`
f. `echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg]`
g. `https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list`
h. `sudo apt-get update`
i. `sudo apt-get install helm`

**3. Assign Admin role to EC2 instance**

**4. Install Terraform on EC2 instance-**

https://www.cherryservers.com/blog/install-terraform-ubuntu

a. `wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg`
b. `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`
c. `sudo apt update && sudo apt install terraform`
d. `cd ~`
e. `mkdir terraform`
f. `cd terraform`

**5. Install Amazon CLI on EC2 instance**

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

a. **curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"**
b. **unzip awscliv2.zip**
c. **sudo ./aws/install**
d. **aws config (to check installed or not)**

**6. Create K8s cluster using terraform**

https://github.com/ankit-jagtap-devops/eks-deployment-with-cicd

a. cd /eks-deployment-with-cicd
b. ls
c. cd infra
d. ls
e. terraform init -reconfigure
f. terraform plan
g. terraform apply

Create bucket and give buckent name in backend file.

**5. Connect to K8s Cluster**

https://docs.aws.amazon.com/eks/latest/userguide/getting-started-console.html

aws eks update-kubeconfig --region ap-south-1 --name ankit-cluster

**6. # kubectl get nodes**

**7. Create a new Helm chart:**

a. helm create my-nginx
b. Ls

c. cd my-nginx

**8. Navigate to the chart directory:**

a. cd my-nginx
b. Vim values.yml

**9. Customize the Helm Chart**

**Note: verify the below  steps/setting of file .and match them accordingly**

a. replicaCount:
b. image:

   repository: nginx

   tag: latest

   pullPolicy: IfNotPresent

c. service:

      type: LoadBalancer

       port: 80

d. ingress:

      enabled: false

e. resources: {}

f. nodeSelector: {}
g. tolerations: []
h. affinity: {}

**10. Deploy the chart to your EKS cluster:**

```
# helm install my-nginx .
```

**11. Verify the deployment:**

```
# helm list
```

```
# kubectl get pods
```

```
# kubectl get svc
```

**12. Access the NGINX Application**

```
# kubectl get svc my-nginx
```

**Upgrade the Release:** If you need to make changes, update the values.yaml or other chart files, and then upgrade the release:

```
# helm upgrade my-nginx .
```

**Rollback the Release:**

```
# helm rollback my-nginx <revision-number>
```

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