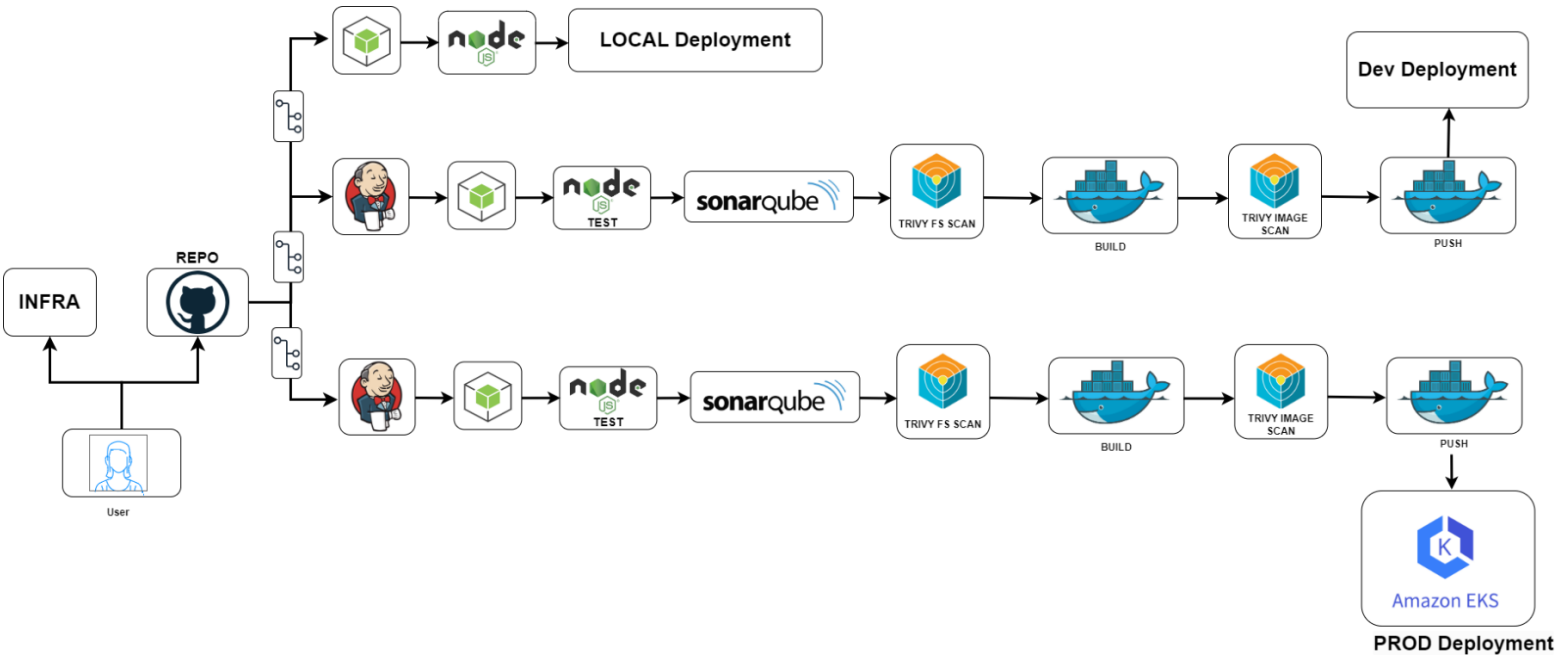
3-Tier Full Stack Project



Setting up Jenkins:

Installing Jenkins on Ubuntu

```
#!/bin/bash

# Install OpenJDK 17 JRE Headless
sudo apt install openjdk-17-jre-headless -y

# Download Jenkins GPG key
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

# Add Jenkins repository to package manager sources
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

# Update package manager repositories
sudo apt-get update

# Install Jenkins
sudo apt-get install jenkins -y
```

Installing Docker for Future Use

```
#!/bin/bash

# Update package manager repositories
sudo apt-get update

# Install necessary dependencies
sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key
sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker 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

# Update package manager repositories
sudo apt-get update

# Install Docker
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin

sudo chmod 666 /var/run/docker.sock
```

Setting up SonarQube

Install docker from above command & run below command to setup SonarQube

```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

Setting up EKS:

AWS CLI Installation

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install
aws configure
```

kubectl Installation

```
curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-
01-05/bin/linux/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin
kubectl version --short --client
```

eksctl Installation

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(una
me -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

Creating EKS Cluster

```
eksctl create cluster --name=my-eks22 \
                      --region=ap-south-1 \
                      --zones=ap-south-1a,ap-south-1b \
                      --without-nodegroup

eksctl utils associate-iam-oidc-provider \
    --region ap-south-1 \
    --cluster my-eks22 \
    --approve

eksctl create nodegroup --cluster=my-eks22 \
                       --region=ap-south-1 \
                       --name=node2 \
                       --node-type=t3.medium \
                       --nodes=3 \
                       --nodes-min=2 \
                       --nodes-max=4 \
                       --node-volume-size=20 \
                       --ssh-access \
                       --ssh-public-key=Key \
```

```
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
```

Pipelines & Deployments

Local Execution

To deploy your application locally following the steps you provided, you'll need to execute the following commands on your T2.Medium Ubuntu machine:

1. **Connect to the machine using ssh-key in Mobaxterm:**

2. **Clone the repository:**

   ```
   git clone <repository_url>
   cd <repository_directory>
   ```

3. **Install Node.js using NVM:**

```
# installs NVM (Node Version Manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash

# Execute below commands
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"

# download and install Node.js
nvm install 20

# verifies the right Node.js version is in the environment
node -v # should print `v20.12.2`

# verifies the right NPM version is in the environment
npm -v # should print `10.5.0`
```

4. **Obtain API keys:**

   - Create an account on Cloudinary and obtain your cloud name, API key, and secret.
   - Create an account on Mapbox and obtain your public access token.
   - Sign up for MongoDB Atlas and create a database. Retrieve your connection URL.

5. **Create a .env file:**

   ```
   vi .env   file in the project directory
   ```

   Add the following lines to the file and replace placeholders with your actual values:

   ```
   CLOUDINARY_CLOUD_NAME=[Your Cloudinary Cloud Name]
   CLOUDINARY_KEY=[Your Cloudinary Key]
   CLOUDINARY_SECRET=[Your Cloudinary Secret]
   MAPBOX_TOKEN=[Your Mapbox Token]
   DB_URL=[Your MongoDB Atlas Connection URL]
   SECRET=[Your Chosen Secret Key]
   ```

6. **Install project dependencies:**

   ```
   npm install
   ```

7. **Start the application:**

   ```
   npm start
   ```

8. **Access the app:** Open a web browser and navigate to `http://VM_IP:3000` (replace `VM_IP` with the IP address of your Ubuntu machine).

Dev Deployment Pipeline:

```
pipeline {
    agent any

    tools {
        nodejs 'node21'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git credentialsId: 'git-cred', url:
'https://github.com/jaiswaladi246/DevOps-Shack-3Tier.git'
            }
        }

        stage('Install Package Dependencies') {
            steps {
                sh "npm install"
            }
        }

        stage('Unit Tests') {
            steps {
               sh "npm test"
            }
```

```
        }

        stage('Trivy FS Scan') {
            steps {
                sh "trivy fs --format table -o fs-report.html ."
            }
        }

        stage('SonarQube') {
            steps {
                withSonarQubeEnv('sonar') {
                    sh "$SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectKey=Campground -Dsonar.projectName=Campground"
                }
            }
        }

        stage('Docker Build & Tag') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {
                        sh "docker build -t adijaiswal/camp:latest ."
                    }
                }
            }
        }

        stage('Trivy Image Scan') {
            steps {
                sh "trivy image --format table -o fs-report.html
adijaiswal/camp:latest"
            }
        }

        stage('Docker Push Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {
                        sh "docker push bittush8789/camp:latest"
                    }
                }
            }
        }

        stage('Docker Deploy To Local') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {
                        sh "docker run -d -p 3000:3000
adijaiswal/camp:latest"
                    }
                }
            }
        }
    }
}
```

Production Deployment Pipeline:

```
pipeline {
    agent any

    tools {
        nodejs 'node21'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git credentialsId: 'git-cred', url:
'https://github.com/jaiswaladi246/DevOps-Shack-3Tier.git'
            }
        }

        stage('Install Package Dependencies') {
            steps {
                sh "npm install"
            }
        }

        stage('Unit Tests') {
            steps {
               sh "npm test"
            }
        }

        stage('Trivy FS Scan') {
            steps {
                sh "trivy fs --format table -o fs-report.html ."
            }
        }

        stage('SonarQube') {
            steps {
                withSonarQubeEnv('sonar') {
                    sh "$SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectKey=Campground -Dsonar.projectName=Campground"
                }
            }
        }

        stage('Docker Build & Tag') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {
                        sh "docker build -t adijaiswal/campa:latest ."
                    }
                }
```

```
            }
        }

        stage('Trivy Image Scan') {
            steps {
                sh "trivy image --format table -o fs-report.html
adijaiswal/campa:latest"
            }
        }

        stage('Docker Push Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {
                        sh "docker push bittush8789/campa:latest"
                    }
                }
            }
        }

        stage('Deploy To EKS') {
            steps {
                withKubeCredentials(kubectlCredentials: [[caCertificate:
'', clusterName: 'EKS-5', contextName: '', credentialsId: 'k8-token',
namespace: 'webapps', serverUrl:
'https://0CB794B1E81785AF1D98888B0FB36B19.gr7.ap-south-
1.eks.amazonaws.com']]) {
                    sh "kubectl apply -f Manifests/"
                    sleep 60
                }
            }
        }

        stage('Verify the Deployment') {
            steps {
                withKubeCredentials(kubectlCredentials: [[caCertificate:
'', clusterName: 'EKS-5', contextName: '', credentialsId: 'k8-token',
namespace: 'webapps', serverUrl:
'https://0CB794B1E81785AF1D98888B0FB36B19.gr7.ap-south-
1.eks.amazonaws.com']]) {
                    sh "kubectl get pods -n webapps"
                    sh "kubectl get svc -n webapps"
                }
            }
        }
    }
}
```