This documentation provides step-by-step instructions for setting up Jenkins, SonarQube, Nexus, Trivy, and Kubernetes, along with a Jenkins pipeline for a sample project.

Jenkins Setup Documentation

Prerequisites

Before setting up Jenkins, ensure that Java Development Kit (JDK) is installed on your system.

Jenkins Installation

Follow the steps in the official Jenkins installation documentation to install Jenkins on your Linux machine.

SonarQube Setup Documentation

Installation

1. Install Docker on your VM.
2. Run the following command to start SonarQube in a Docker container:

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

Nexus Setup Documentation

Installation

1. Install Docker on your VM.
2. Run the following command to start Nexus in a Docker container:

   ```
   docker run -d -p 8081:8081 sonatype/nexus3
   ```

Trivy Installation on Jenkins Documentation

1. Install required dependencies:

   ```
   sudo apt-get install wget apt-transport-https gnupg lsb-release
   ```

2. Add Trivy GPG key:

   ```
   wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg -
   -dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
   ```

3. Add Trivy repository:

   ```
   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
   ```

4. Update package list:

   ```
   sudo apt-get update
   ```

5. Install Trivy:

   ```
   sudo apt-get install trivy -y
   ```

Kubernetes Setup Documentation

On Master and Worker Nodes

1. Update and install Docker:

   ```
   sudo apt-get update -y
   sudo apt-get install docker.io -y
   sudo service docker restart
   ```

2. Add Kubernetes repository:

   ```
   sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-
   key add -

   echo "deb http://apt.kubernetes.io/ kubernetes-xenial main"
   >/etc/apt/sources.list.d/kubernetes.list
   sudo apt-get update
   ```

3. Install Kubernetes components:

   ```
   sudo apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
   ```

On Master Node

1. Initialize Kubernetes with a pod network CIDR:

   ```
   kubeadm init --pod-network-cidr=192.168.0.0/16
   ```

2. Set up kubeconfig:

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   ```

3. Apply network plugins:

   ```
   kubectl apply -f https://docs.projectcalico.org/v3.20/manifests/calico.yaml

   kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-
   nginx/controller-v0.49.0/deploy/static/provider/baremetal/deploy.yaml
   ```

Create Service Account, Role, Bind Role, Createtoken for service account

Creating Service Account

```
apiVersion:  v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: webapps
```

Create Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: app-role
  namespace: webapps
rules:
  - apiGroups:
        - ""
        - apps
        - autoscaling
        - batch
        - extensions
        - policy
        - rbac.authorization.k8s.io
    resources:
      - pods
      - componentstatuses
      - configmaps
      - daemonsets
      - deployments
      - events
      - endpoints
      - horizontalpodautoscalers
      - ingress
      - jobs
      - limitranges
      - namespaces
      - nodes
      - pods
      - persistentvolumes
      - persistentvolumeclaims
      - resourcequotas
      - replicasets
      - replicationcontrollers
      - serviceaccounts
      - services
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Bind the role to service account

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-rolebinding
  namespace: webapps
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: app-role
subjects:
- namespace: webapps
  kind: ServiceAccount
  name: jenkins
```

Generate token using service account in the namespace

Create Token

Jenkins Pipeline Documentation

```
ppipeline {
    agent any

    tools {
        maven 'maven3'
        jdk 'jdk17'
    }

   environment {
       SCANNER_HOME= tool 'sonar-scanner'
   }

    stages {
        stage('Git Checkout') {
            steps {
                git branch: 'main', url:
'https://github.com/jaiswaladi246/Ekart.git'
            }
        }

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

        stage('Unit Tests') {
            steps {
                sh "mvn test -DskipTests=true"
            }
        }

        stage('SonarQube Analysis') {
            steps {
               withSonarQubeEnv('sonar') {
                  sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectKey=EKART
-Dsonar.projectName=EKART \
                  -Dsonar.java.binaries=. '''
               }
            }
        }

        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: 'jdk17',
maven: 'maven3', mavenSettingsConfig: '', traceability: true) {
                  sh "mvn deploy -DskipTests=true"
               }
            }
        }

        stage('Docker Build & tag Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred', toolName:
'docker') {
                           sh "docker build -t adijaiswal/ekart:latest -f
docker/Dockerfile ."
                    }
                }
            }
        }

         stage('Trivy Scan') {
            steps {
                sh "trivy image adijaiswal/ekart:latest > trivy-report.txt "
            }
        }

        stage('Docker Push Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred', toolName:
'docker') {
                           sh "docker push adijaiswal/ekart:latest"
                  }
                }
            }
        }
        stage('Kubernetes Deploy') {
            steps {
                withKubeConfig(caCertificate: '', clusterName: '', contextName:
'', credentialsId: 'k8-token', namespace: 'webapps', restrictKubeConfigAccess:
false, serverUrl: 'https://172.31.8.162:6443') {
                  sh "kubectl apply -f deploymentservice.yml -n webapps"
                  sh "kubectl get svc -n webapps"
              }
            }
        }

    }
}


}
```