ADQ-JAVA-APP-DEPLOYMENT

Pre-requisites:

1. Jenkins-sever – with Firewall rules allowing ports: 8080, 8081, 9000, 4243, 80, 22
2. Install SonarQube in Jenkins server using [sonar.sh] script in below GitHub Repository: https://github.com/SaravanaNani/ansible-setup-gcp.git

Note: After installing sonarqube – start the soanrqube follow the below steps to start SonarQube –> run this on server manually as sonar-user

- cd /opt/sonarqube-9.9.6.92038/bin/linux-x86-64/
- sh sonar.sh start
- sh sonar.sh status

3. Install Nexus in Jenkins Server manually using the steps in below GitHub Repository (nexus manual setup) –> https://github.com/SaravanaNani/ansible-setup-gcp.git
4. Tomcat Sever – Using playbook install Tomcat in Ubuntu desktop server

Note: The Jenkinsfile and Source code for this Deployment is taken from below GitHub Repo

Repository: https://github.com/SaravanaNani/jenkins-java-project.git

SonarQube step-up:

Step1 – Login into SonarQube console where initial Username & Password = admin

Step2 – Create a project in SonarQube:

–> Projects –> create project –> manually –> Project display name (adq-java-app)

–> Project key (adq-java-app) –> Main branch name (master) –> click setup

–> How do you want to analyze your repository?

–> locally –> Token name (adq-java-app) –> Generate token and copy it

Step3 – Get into Jenkins Console Paste SonarQube token in Global Credentials:

–> Manage Jenkins –> credentials –> global –> Add Credentials –> Kind (Secrete Text)

–> Paste token - ID (sonar_token) –> description (sonar-token) –> save

Step4 – Generate GitHub classic token and paste it in global credentials.

GitHub Token Generation:

–> Settings –> Developer Setting –> Personal Access Token –> Tokens (classic)

–> Generate new token –> new classic token –> Note (adq-java-app)

–> Select the required permission to token –> Generate Token.

–> Copy token and paste in Jenkins Global credentials:

–> Manage Jenkins –> credentials –> global –> Add Credentials –> Kind (Secrete Text)

–> Paste token - ID (github-pat) - description (GIT-PAT) - save

Step5 – Install Sonar Scanner plugin and Set SonarQube system configuration in Jenkins console:

1. Goto Jenkins Dashboard –> Manage Jenkins –> Plugins –> available plugins
   –> SonarSonarQube Scanner
2. Goto Jenkins Dashboard –> Manage Jenkins –> System configuration –> System
   –> SonarQube servers: –> Enable Environment variables
   –> SonarQube installations –> name (Sonar) –> Server URL (paste SonarQube URL)
   –> Server authentication token (select updated token credentials in above step) –> save

Step6 – Set SonarQube path in tools Jenkins console:

–> Goto Jenkins Dashboard –> Manage Jenkins –> tools –> SonarQube Scanner installations

–> name (sonar) –> select install automatically –> save

Nexus Setup:

Step1 – Create a Repository in Nexus:

To create Repository:

–> Login to Nexus Console (using nexus VM - ExternalIP:8081)

–> Initial Username & Password = admin –> Click on Setting Icon –> Repositories

–> Create repositories –> (maven2 hosted) –> Repo Name (adq-java-app)

–> Deployment Policy Select (Allow Redeploy) –> Create Repository

Step2 – Create a Pipeline Syntax for upload artifact to Nexus:

Click on –> Pipeline Syntax –> Sample Step (Nexus Artifact Uploader)

–> Nexus Version (NEXUS3) –> Protocol (HTTP)

–> Nexus URL (copy & Paste nexus EXTERNALIP:8081)

–> Credentials –> Add (Jenkins) –> kind (Username & Password)

–> Username (nexus-username [admin]) Password (nexus login password)

–> ID (nexus_token) –> Description(nexus) –> Add

–> Click on Artifacts –> Artifact (NEXUS_ARTIFACT_ID) –> Type (war)

–> Classifier (leave it blank) –> File (target/ JAVA_APP-1.2.${BUILD_NUMBER})

–> Generate Pipeline Syntax (copy & paste in pipeline stage nexus)

Note: The below given details are for the Jave application Code used here in this Task. Below Details vary for other Java application Code, you find it in POM.XML file

NEXUS_REPOSITORY = 'adq-java-app'

NEXUS_GROUP_ID = 'in.RAHAM'

NEXUS_ARTIFACT_ID = 'JAVA_APP'