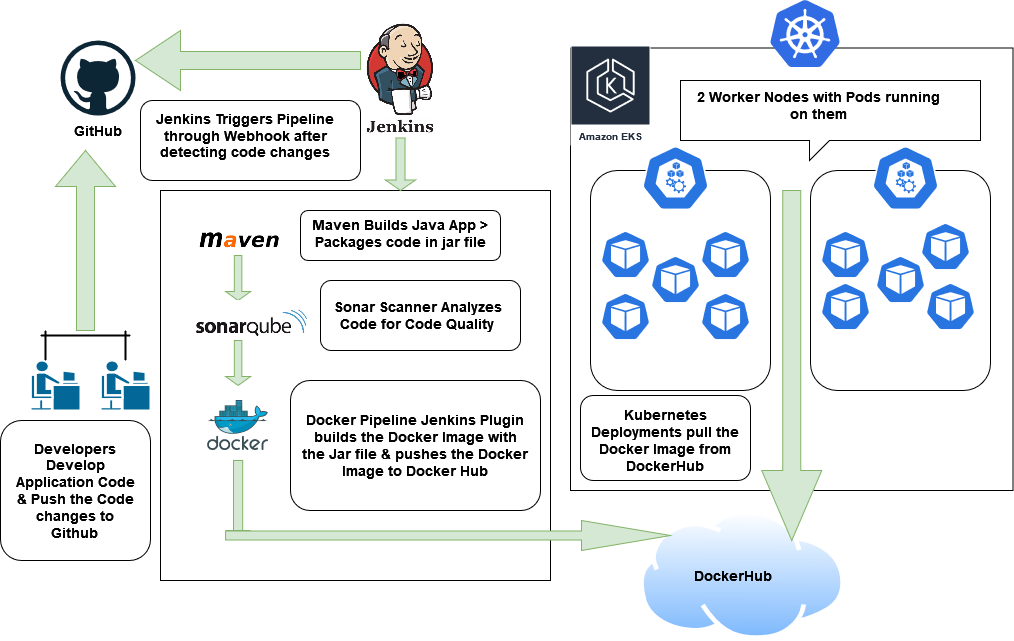
**Devops Project: Implement End to End Continuous Integration, Deployment & Delivery with Jenkins, Maven, SonarQube, Docker, Dockerhub, GitHub, AWS EC2 & AWS EKS (Kubernetes).**

**Objective:** Build, Test & Deploy Java Application through a Jenkins Declarative Pipeline on AWS EKS Cluster (Kubernetes).

**Prerequisites:** AWS, GitHub & Dockerhub Accounts. Any Code Editor or IDE like VS Code, Eclipse etc. & Git for Windows installed on your local machine.

**High Level Architecture Diagram of the Entire Project:**

****

**Install Git for Windows:** [**https://nerdschalk.com/how-to-install-and-use-git-on-windows-11/**](https://nerdschalk.com/how-to-install-and-use-git-on-windows-11/)

Test to see if Git work from Code Editor Terminal Window by typing **Git** command.

Login to AWS & Create **t2.medium** Ubuntu 20.04 Instance with **40GB** Hard drive in existing default Security Group.

**Login to EC2 with username ubuntu:**

ubuntu@ec2-instance$ sudo su -#Change to root account

**Install & Configure Openjdk (Java)**:

root@ec2-instance$ apt install openjdk-11-jdk

root@ec2-instance$ export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

root@ec2-instance$ export PATH=$PATH:$JAVA\_HOME/bin

#**OR** for Permanent Changes, do the following:

root@ec2-instance$ vim /etc/profile

#Add following lines:

JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

PATH=$PATH:$HOME/bin:$JAVA\_HOME/bin

export JAVA\_HOME

export JRE\_HOME

export PATH

root@ec2-instance$ source /etc/profile

root@ec2-instance$ echo $JAVA\_HOME ## Verify if this command outputs the Java Home path. If yes, proceed with next steps.

**Install Jenkins: https://pkg.jenkins.io/debian-stable/**

root@ec2-instance$ usermod -aG docker jenkins # So jenkins user has permissions in docker group

root@ec2-instance$ systemctl restart jenkins

**Configure Jenkins (Install Plugins):** Go to Jenkins Dashboard > Manage Jenkins > Manage plugins > Available > **Maven Integration,** **SonarQube Scanner**, **Docker Pipeline Plugin, Pipeline** & **Kubernetes CLI** > Install.

**Install Docker on Ubuntu 20.4:** [**https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04**](https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04)

**Install Sonarqube Docker Container on Jenkins EC2:**

root@ec2-instance$ sysctl -w vm.max\_map\_count=262144

root@ec2-instance$ sysctl -w fs.file-max=65536

root@ec2-instance$ ulimit -n 65536

root@ec2-instance$ ulimit -u 4096

root@ec2-instance$ docker run -d -p 9000:9000 -v sonarqube\_conf:/opt/sonarqube/conf -v sonarqube\_extensions:/opt/sonarqube/extensions -v sonarqube\_logs:/opt/sonarqube/logs -v sonarqube\_data:/opt/sonarqube/data sonarqube

**Add Webhook in GitHub & SonarQube:**

Add Github Webhook for Jenkins in Github: Github Repo > settings > webhook > payload url= <http://jenkins-url:8080/github-webhook/> > Click “Update Webhook”.

Add Webhook in Sonarqube for Jenkins: Under "Administration"> Configuration > Create Webhook- [jenkins-url:8080/sonarqube-webhook/](Devops-Project-High-Level-Architecture-Diagram_files)

**On Jenkins UI, Global Tool Configuration Setting for Tools like Maven, JDK, Docker & Sonarqube:**

* Global Tool Configuration > Add maven (install automatically)
* JDK (JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64 ##Verify the JAVA\_HOME path by running ‘echo $JAVA\_HOME on EC2 Instance’
* Sonarqube Scanner (install automatically)
* Docker /var/lib/docker/

Tools Names: jdk11, maven3, sonar, docker

**Create Sonar Token in Sonar UI:**

**User > My Account > Security > Click ‘Generate Token’ for Admin user> Copy & Save the Token in a notepad > Done**

**Configure SonarQube in Jenkins UI:**

On Jenkins UI > Configure systems > Under Sonarqube scanner, enter sonar url:9000, Add Credentials- Secret text > copy paste Token from Sonarqube user Administrator from previous step.

**Fix GitHub's 'support for password authentication was removed' error:**

<https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Fix-GitHubs-support-for-password-authentication-was-removed-error>

1. Log into GitHub with your username and password
2. Navigate to your GitHub account settings
3. Scroll down and click ‘Developer settings’ in the list of links to the left
4. Click the Personal access tokens link
5. Click the ‘Generate new token’ button
6. Add a ‘Note’ to describe the tokens usage
7. Set an expiration date for the GitHub personal access token
8. Select the appropriate authentication scope
9. Click the ‘Generate token’ button

**Jenkins Pipeline Job Configuration:**

* New Item > Enter a project Name > Select “Pipeline” > Click ‘OK’.
* Under “Build Triggers”, Select “GitHub hook trigger for GITScm polling
* Under “Advanced Project Options > Pipeline”, Select “Pipeline Script from SCM” from the dropdown. Under SCM, Select “Git” & enter the Github Repo URL.
* Under “Branches to build > Branch Specifier (blank for 'any')”, Type \*/\* > click ‘Save’.

**Jenkins Configuration for Email Notification:**

Under Extended E-mail Notification Section, enter the below details:

- SMTP server: smtp.gmail.com

- SMTP Port: 465

- Default Recipients: <your-email@email.com>

- Tick/Select the Checkbox "Allow Sending to unregistered users" > Save

- Click Test Configuration after typing your email address where you want Build mails to be sent. Verify if you received the Test Email from Jenkins.

- Save

**Note:** If Google has blocked the email from Jenkins for Security reasons, Please follow the steps from the link below to ‘Allow less Secure apps to connect to Google’.

<https://support.google.com/accounts/answer/6010255?hl=en>

**Note:** After every restart of ec2 where sonarqube & jenkins are installed, webhook urls need to be changed with the new public ip of ec2 in these 4 places:

1 - github repo > settings > webhook > payload url

2 - sonarqube UI > "Administration"> Configuration > Webhook > Update

3 & 4 - jenkins UI > Manage Jenkins > Configure Systems > Jenkins Location & Sonarqube Servers.

**Create Dockerhub Repo:**

Login to DockerHub & Create a Repository.

**Add Credentials in Jenkins UI:**

Manage Jenkins > Manage Credentials > Under “Credentials”, Click Jenkins > Global Credentials > Add Credentials > (Create secret text credential for Sonar) Select ‘secret text’ from drop down > copy paste token from sonarqube user administrator > give a meaningful name like ‘sonar-token’.

Likewise, Create Credential with docker username & password for DockerHub.

Finally, After EKS has been setup in the next Topic, Create Credential for AWS EKS Cluster with “Secret file” & copy paste the contents of kubeconfig file (Run the below command) after running the following command as jenkins user on jenkins EC2:

jenkins@ip-172-31-90-204:~$ cat /var/lib/jenkins/.kube/config

**AWS EKS Cluster Creation:**

Create IAM role with AdministratorAccess & attach the role to Jenkins EC2.

Install awscli, kubectl & eksctl (Refer to Installation Documentation for each of these tools).

Awscli Installation: pip3 install awscli boto boto3

jenkins@ip-172-31-90-204:~$ aws configure

Kubectl Installation: <https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>

Eksctl Installation: <https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>

[**https://www.coachdevops.com/2022/02/create-amazon-eks-cluster-by-eksctl-how.html**](https://www.coachdevops.com/2022/02/create-amazon-eks-cluster-by-eksctl-how.html)

jenkins@ip-172-31-90-204:~$ eksctl create cluster --name shubhradeep-eks --region us-east-1 --nodegroup-name shubhradeep-nodes --node-type t2.medium --managed --nodes 1

------------------------OPTIONAL COMMANDS-------------------------------

jenkins@ip-172-31-90-204:~$ eksctl get cluster --name shubhradeep-eks --region us-east-1

jenkins@ip-172-31-90-204:~$ kubectl create deployment nginx --image=nginx

jenkins@ip-172-31-90-204:~$ eksctl delete cluster --name shubhradeep-eks --region us-east-1

------------------------OPTIONAL COMMANDS-------------------------------

jenkins@ip-172-31-90-204:~$ kubectl get nodes #To check if worker nodes are in Ready Status

jenkins@ip-172-31-90-204:~$ kubectl get all # #To check all Resources deployed likes Deployments & Services in Default Namespace

jenkins@ip-172-31-90-204:~$ vim service.yaml

#Copy Paste the below content into service.yaml:

apiVersion: v1

kind: Service

metadata:

labels:

app: springboot-app

k8s-app: springboot-app

name: springboot-app

spec:

ports:

- name: http

port: 80

protocol: TCP

targetPort: 8085

type: LoadBalancer

selector:

app: springboot-app

jenkins@ip-172-31-90-204:~$ kubectl apply -f service.yaml

jenkins@ip-172-31-90-204:~$ kubectl get all ## Check if service called “springboot app” is showing in the output. If yes, then proceed with the next steps.

**Download my Devops project Github Repo on local machine:**

1. Download & Extract Zip bundle from <https://github.com/shubhradeep23/jenkins-sonar-eks>
2. Create Empty Github Repository with a Meaningful Repo name.
3. Create a Empty Folder on your local machine & Navigate to the Folder in your Code Editor/IDE from Terminal Window.
4. git init
5. Copy the files from Step1 into the newly created folder in Step3.
6. Make changes to Jenkinsfile Line 3: imagename = <your-dockerhub-username>/<your dockerhub repository-name>:latest
7. jenkins@ip-172-31-90-204:~$ ‘kubectl get all’.
8. Copy the ELB URL from output of above command in step 7.
9. Make changes to Jenkinsfile Line 6: CHECK\_URL = <Paste the ELB URL in the value>/greeting
10. Make changes to Jenkinsfile Line 69: withKubeConfig([credentialsId: '<your-eks-cluster-name>', serverUrl: ''])
11. git add .
12. git commit -m "first commit"
13. git branch -M main
14. git remote add origin [https://github.com/<your-github-username>/<your-github-repository>.git](https://github.com/%3cyour-github-username%3e/%3cyour-github-repository%3e.git) which you created in from Step2.
15. git push -u origin main

**Verify if Pods/Deployment is live/Running on EKS & if the Java app is Reachable from a Web Browser:**

jenkins@ip-172-31-90-204:~$ kubectl get all

Copy the ELB URL from the output of the above command or the value of CHECK\_URL from Jenkinsfile Line 6.

From a Web Browser, Paste the ‘ELB-URL/greeting’. If you get a Welcome message, then you have successfully completed the project. Congratulations !!

Make changes to the Java App (follow the below steps) & see if the changes are being reflected in the Web browser.

From Code Editor, Navigate to the following path: src/main/java/com/example/k8s/springbootkubernetes/controller/WelcomeController.java

Make changes to the Text Message in Line 11.

