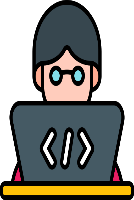
**Complete CI CD**

**Architecture Diagram**



**Developer(git)**



**GitHub**

Fetch Code

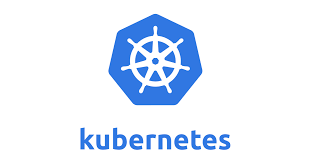
Build

Unit Test

Code Analysis



Docker Build



KOPS

Push

**Steps for Continuous Integration**

1. Jenkins Setup
2. SonarQube Setup
3. Security Setup
4. Plugins
5. Integrate

* SonarQube
* ECR

1. Write pipeline script
2. Set notifications

Jenkins Setup:

1. Create an EC2 instance

* Give name as you wish “Jenkins-server”
* Select AMI “Ubuntu 20.04 LTS”
* Select instance type : t2.small
* Create key pair
* Edit incoming securing group –
  + - Allow port 22 from anywhere
    - Allow port 8080 from anywhere
    - Allow port 80 from anywhere
* Copy the Jenkins setup shell script in the user data field
* Launch instance

**Build Tool installation:**

**Make sure you have installed Maven and JDK8 in Jenkins. Follow below steps to install JDK8**

1. **SSH to Jenkins instance**
2. **Run commands**

* **Sudo su -**
* **Sudo apt update**
* **Sudo apt install openjdk-8-jdk -y**
* **ls /usr/lib/jvm**
* **copy the path like - /usr/lib/jvm/java-1.8.0-openjdk-amd64**

1. **Go to Jenkins dashboard**
2. **Click on “Manage Jenkins” – Click on “Global Tool Configuration”**
3. **Go to “JDK Installations” – Click Add – Give name as “OracleJDK8”**
4. **Paste the JDK8 installation path which we have copied to “JAVA\_HOME”**
5. **Now go down and select “Maven Installations”**
6. **Give name as “MAVEN3” – select the version as 3.\* and Click Save**

SonarQube setup

1. Launch instance for Sonar Server

* Give name as “SonarServer”
* Use AMI as “Ubuntu 20.04 LTS”
* Instance type: t2.medium
* Create key pair – sonar key
* Create security group
  + - Name – sonarSG
    - Allow port 22 from anywhere
    - Allow port 80 from anywhere
    - Allow port 9000 from anywhere
* Copy pastes the Sonar shell script into user data field
* Launch instance

Check the services up and running

1. Login to Jenkins server and check Jenkins service is running using below command

Systemctl status Jenkins

Access from browser: Public\_ip:8080

1. Login to Sonar server and check the sonar service is running

Systemctl status sonarqube

Access from browser: Public\_ip

**Plugins Installation**

* Sonarqube
* Git
* Pipeline Maven Integration
* BuildTimeStamp

Go to Manage Jenkins 🡪 Manage Plugins 🡪 Click on Available 🡪 Search for “Sonarqube scanner” 🡪 Search for “Build timestamp” 🡪 Search for “Pipeline Maven Integration” 🡪 last one “Pipeline utility steps”

Click on “Install without restart”

**SonarQube Tools installation in Jenkins**

* Go to “Manage Jenkins” 🡪 “ go to “Global Tool Configuration”
* Go to ‘Sonarqube scanner” option 🡪 Click “Add Sonarqube scanner”
* Give name “sonar4.8” 🡪 save it

**Integrate SonarQube with Jenkins**

* Go to “Manage Jenkins” 🡪 Click on “Configure system” 🡪 Go to “SonarQube Servers” 🡪 Select “Add SonarQube server” 🡪 check mark “Environmet variables” 🡪 give a name “sonar” 🡪 type the sonarqube server IP address 🡪

<http://172.88.90.22> & save it

**Now create a token for authentication**

* Go to SonarQube server URL
* Login there
* Click on Profile logo
* Go to “My Account”
* Go to security
* Give a name for token
* Click on generate
* Copy the generated token

Now go to “Configure system” in Jenkins and go to SonarQube server & add the credentials by selecting “secret text” & give the token ID as “sonartoken”

<https://www.jenkins.io/doc/pipeline/steps/sonar/>

**Quality Gates in SonarQube**

* Lets create gate for our SonarQube project
* Login to SonarQube server
* Go to Quality Gates option on the top
* Create a Quality Gate – give any name e.g: “sscademy-QG”
* Click “Add condition” 🡪 select “On overall Code”
* Select metrics 🡪 “Bugs” 🡪 Give “60”
* Click Add condition
* Come to “Project” option on SonarQube
* Click on our project
* Go to “Project settings”
* Click on “Quality gate”
* Select the newly created Quality Gate
* Click again on “Project settings”
* Click on “Webhooks”
* Click “Create”
* Give name as “Jenkins-ci-webhooks”
* Type URL as 🡪 <http://jenkins_public_ip:8080/sonarqube-webhook>
* Click Create

1. Create a GIT repository

* Login to github account
* Create a new repository – give name as “ci-cd”
* Select “Private” repo
* Create repository

1. Create SSH key

* Create SSH key 🡪 Open gitbash
* Type “ssh-keygen.exe”
* Create a SSH key
* You can see the SSH keys in “ **ls ~/.ssh ”**
* Copy the content of the public key
* Come to GITHUB account 🡪 go to GITHUB settings
* Go to “SSH and GPG key”
* Click on “New ssh key”
* Give a title “projectkey”
* Go to repository and copy the ssh link not http link
* Clone the repo
* Create a simple pipeline as a Code – give name as “**Jenkinsfile**”
* Commit and push it to the repository

1. Create Jenkins job to access Jenkinsfile from GIT repo

* Go to Jenkins dashboard
* Go to “Manage Jenkins”
* Click on “Configure Global Security”
* Go to “Git Host Key Verification Configuration” option
* Select option as “Accept first connection”
* Save
* Now create a pipeline
* In the pipeline section select “Pipeline script from SCM”
* Select SCM as “git”
* Give the Repository URL of SSH
* Click Add credential 🡪 Select kind as “SSH username with private key”
* Give ID as “gitsshkey” 🡪 username, type github account name
* Select private key 🡪 enter directly
* Copy the SSH private from which we have generated
* Click add
* Select the credential which we added
* Click save
* Click on Build now

**WEBHOOKS**

* Copy the Jenkins server URL ( http://public\_ip:8080)
* Go to GitHub repository 🡪 go to GitHub Repository settings (not account settings)
* Go to option “Webhooks”
* Click on “Add webhook”
* Paste the Jenkins URL along with /github-webhook/
* Like this <http://Jenkins_public_ip:8080/github-webhook/>
* In Content type 🡪 select “application/json”
* Select “Just the push event”
* Click Add webhook
* Now to go to Jenkins job
* Click on Configure
* In the “Build Triggers” 🡪 Select “GitHub hook trigger or GITScm polling”
* Click Save

Now do the commit to the repository

Install Docker Engine in Jenkins server

1. Login to Jenkins server
2. Run below commands to install Docker

$ curl -fsSL https://get.docker.com -o get-docker.sh

$ sh get-docker.sh

1. Add Jenkins user to Docker group

$ id jenkins

$ usermod -a -G docker jenkins

$ id jenkins

$ apt install awscli -y

$ reboot

AWS Setup

* Login to AWS console
* Search for IAM
* Click on “Users”
* Click “Add user”
* Give the username as “Jenkins”
* Select “Access Key – Programmatic access”
* Attach existing policy
* Search for “Registry”
* Select “AmazonEC2ContainerRegistryFullAccess”
* Search for “ecs”
* Select “AmazonECS\_FullAccess”
* Click next
* Create user
* Download .csv file
* Now create the ECR repository 🡪 search for ecr in the console search bar
* Select “Elastic Container Registry”
* Click “Create Repository”
* Keep it “Private”
* Give a name like “sscademyaapimg”
* Click “Create respository”

Jenkins Plugins setup

* Login to Jenkins dashboard
* Click on “Manage Jenkins”
* Click on “Manage plugins”
* Click on “Available”
* Search for “Docker pipeline”
* Search for “Amazon ecr”
* Search for “Amazon Web Services SDK::All”
* Search for “CloudBees Docker Build and Publish”
* Click install without restart

Store AWS credentials in Jenkins

* Go to Jenkins dashboard
* Manage Jenkins
* Manage Credentials
* Click on Jenkins
* Click on “Global credentials”
* Click Add credentials
* Select “AWS credentials”
* Give name as “awscreds”
* Give “Access key ID” & “Secret access key” which we have downloaded while creating the IAM user

**Steps for Continuous Deployment / Delivery**

1. **Setup KOPS Kubernetes Cluster**

**Setup Kubernetes with Kops**

**Pre-requisites**

1. Domain for Kubernetes DNS records
   * E.g., sscademy-k8s.com
2. Create a Linux VM and setup
   * Kops, kubectl, ssh keys, awscli
3. Login to AWS account and setup
   * S3 bucket, IAM User for AWSCli, Route53 Hosted Zones.
4. Login to Domain Registrar (E.g. Hostinger.com, GoDaddy.com)
   * Create NS records for subdomain pointing to Route 53 hosted zone NS servers

**Let’s start the setup**

1. **Login to AWS cloud**
   * Launch an EC2 instance
     + Give name as - Kops
     + Select Ubuntu 20.04 Image
     + Select Server type – t2.micro
     + Create a security group
       - Name – kops-SG
       - Allow port 22 from anywhere
     + Create a security key
     + Click on Launch instance
2. **S3 Bucket**
   * Create a s3 bucket
   * sscademy-kops-state
   * Click on Create
3. **Create an IAM user**
   * Give username as – kopsadmin
   * Select Programmatic access
   * Next
   * Attach policy – AdministratorAccess
   * Create user
   * Download th .csv credential file
4. Create Hosted Zone (sub domain)
   * Click on Create Hosted Zone
   * Type sub domain name as – k8s.sscademy.in
   * Select Public hosted zone
   * Click on Create hosted zone
5. Add the NS server records in domain registrar
   * Login to Hostinger.com/godaddy.com
   * Add 4 NS records for your domain
   * Select record type – Nameserver
   * Host – k8s
   * Points to – copy & paste the NS server
   * Likewise create 4 records
6. Login to EC2 instance which we have created
   * Generate the ssh keys
     1. $ ssh-keygen
   * Install AWS cli
     1. $ sudo apt update && sudo apt install awscli -y
     2. $ aws configure
        1. Copy & paste the access key and secret access key
        2. Enter the region – us-east-1
        3. Output format – json
   * Install kubectl

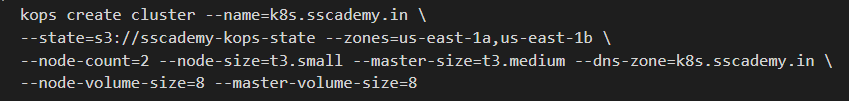
<https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

* + Install kops

<https://kubernetes.io/docs/setup/production-environment/tools/kops/>

* + Select Linux OS
  + Follow the instructions to install kops
  + Verify it using command – kops –help
  + Verify the domain
    1. $ nslookup -type=ns k8s.ssdevops.in

1. Now let’s create the cluster. For that follow the below command





After this we have to wait for 15min, then we can verify it.

$ kops validate cluster –state=s3://sscademy-kops-state

**Kubernetes integration with Jenkins**

Go to manage Jenkins

Manage plug ins

Search for “Docker”, “Docker pipeline”, “Pipeline utility steps”,

**Install helm in kops EC2 instance**

[**https://helm.sh/**](https://helm.sh/)

[**https://helm.sh/docs/intro/install/**](https://helm.sh/docs/intro/install/)

Refer the Kubernetes folder for creating the helm chart. We can use the directory called app for complete stack deployment.

We have to install helm locally in order to create helm chart out of the app folder inside the Kubernetes directory.

$ choco install kubernetes-helm

Create a directory called helm

$ cd helm

$ helm create sscademychart

$ cd sscademychart/templates

$ rm -rf \*

$ cp kubernetes/app/\* helm/sscademychart/templates/

**Add KOPS EC2 instance as Jenkins slave machine**

Ssh into KOPS EC2 instance

$ sudo mkdir /opt/jenkins-slave

$ sudo chown ubuntu.ubuntu /opt/jenkins-slave -R

$ sudo apt update

$ sudo apt install openjdk-11-jre -y

$ sudo apt install openjdk-8-jdk -y

Allow kops vm security group to access ssh connection from Jenkins security group

**Now go to Jenkins dashboard**

Manage Jenkins

Manage Nodes and Clouds

New node

Give name as “KOPS”

Label as “KOPS”

Remote root directory -> /opt/Jenkins-slave

Usage -> Only build jobs with label expression matching this node

Launch method -> Launch agents via SSH

Host -> private IP of KOPS EC2 instance

Credentials -> Add -> SSH Username with private key

Username -> ubuntu

Copy & paste the .pem key of KOPS EC2 instance which we have downloaded

Host Key verification strategy -> Non verifying verification strategy

Select the credential and save it

Click on Relaunch agent