

**Java Based Application Deployment with EKS Cluster**

Launch the EC2 instance as per the requirement like here I chosen t2.medium. To configure Jenkins we have to follow the below steps.

1. Run the following command on the Jenkins server.( **Change Host Name to Jenkins**)

sudo hostnamectl set-hostname Jenkins

2. **Perform update first**

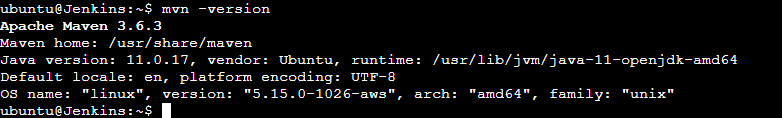
sudo apt update

3. **Maven Installation**

sudo apt install maven -y

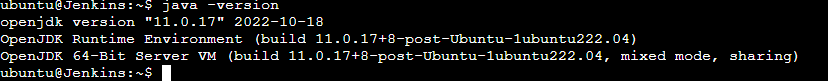
check the maven version by running the following command

mvn -version



4. Once maven installation done java automatically gets installed. Check java by running the following command

Java -version



**5. Jenkins setup**

6. Add Repository key to the system

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add –

7. Append debian package repo address to the system

echo deb http://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list

8. Update Ubuntu package

sudo apt update

9. Install Jenkins: sudo apt install jenkins -y

10. Get the initaill password from the below location on Jenkins server

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Tools Need To Be Installed On Jenkins Server

Install AWS CLI

* curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"
* sudo apt install unzip
* sudo unzip awscliv2.zip
* sudo ./aws/install
* aws –version

Install eksctl

* Download and extract the latest release of eksctl with the following command

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

* sudo mv /tmp/eksctl /usr/local/bin
* eksctl version ( to check the version)

Install kubectl

* sudo curl --silent --location -o /usr/local/bin/kubectl <https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl>
* sudo chmod +x /usr/local/bin/kubectl
* kubectl version --short –client

Install Docker

* sudo apt-get update
* sudo apt install docker.io -y
* Add Ubuntu user to Docker group

sudo usermod -aG docker $USER

* newgrp docker
* sudo systemctl start docker
* sudo systemctl enable docker
* sudo systemctl status docker

Docker Jenkins Integration

* Following plugins needs to be installed in Jenkins

Docker, Docker Pipeline

* Now Login to Jenkins EC2 instance, execute below commands
* Add jenkins user to Docker group

sudo usermod -a -G docker Jenkins

* Restart Jenkins service

sudo service jenkins restart

* Reload system daemon files

sudo systemctl daemon-reload

* Restart Docker service as well

sudo service docker stop

sudo service docker start

Plugins need to be installed in Jenkins

* Kubernetes CLI Plugin
* Quality Gates Plugin
* Sonar Gerrit Plugin
* Sonar Quality Gates Plugin
* SonarQube Generic Coverage Plugin Version
* SonarQube Scanner for Jenkins

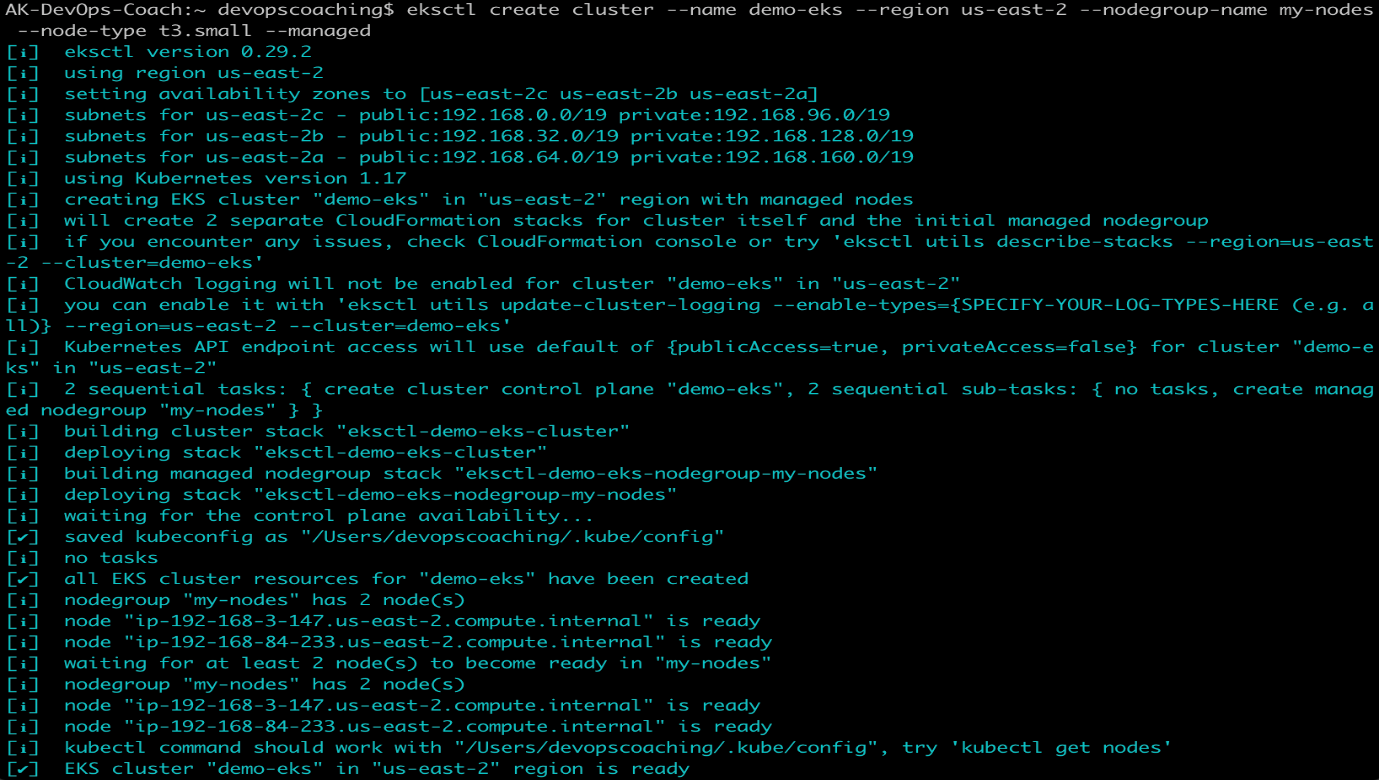
EKS SETUP

1. Switch to Jenkins user

sudo su – jenkins

1. Create EKS Cluster with two worker nodes using eksctl

eksctl create cluster --name demo-eks --region us-east-2 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2



1. This should confirm that EKS cluster is up and running.

eksctl get cluster --name demo-eks --region us-east-2

1. kubectl get nodes. (This should confirm the access of eks cluster from jenkins server)

Sonar Integration with Jenkins

Add the below property in pom.xml to integrate the sonar qube with jenkins

<sonar.organization>user\_id </sonar.organization>

<sonar.host.url>https://sonarcloud.io</sonar.host.url>

And generate the secret token from sonar dashboard and save it as secret text in jenkins dashboard.

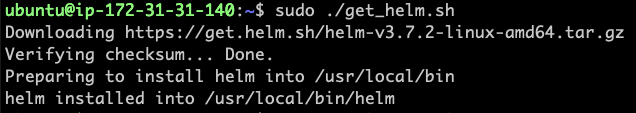
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_HELM INTEGRATION FOR K8’s:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is Helm?

As the Kubernetes platform and ecosystem continued to expand, deploying one and only one Kubernetes configuration file (ie: a single YAML) was not the norm anymore. As number of K8S deployment files increased, how to manage those files? Helm solves that problem.

Installation of helm on the jenkins server to access the eks cluster.

1. curl -fsSL -o get\_helm.sh <https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3>
2. sudo chmod 700 get\_helm.sh
3. sudo ./get\_helm.sh



1. Verify the helm version

helm version –client

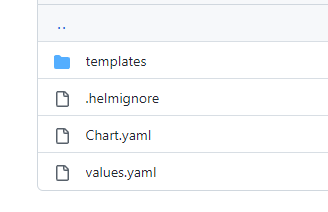
Now Helm is installed on the server, and we are good to use the helm command.

To Run the deployments through helm first we need to create the helm chart for the respective project, and we need to modify helm charts and related file like deployment.yml and service.yml file and so on.

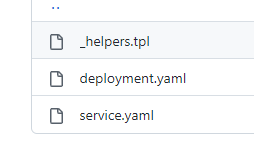
To create the charts first we need to create the chart with the mentioned command below

helm create <chart name>

By running the above command, we will see the file created at that location as shown below

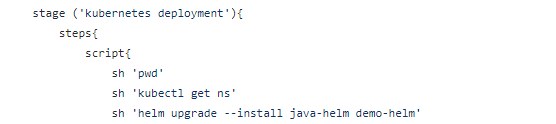


Now we can edit these files as per our requirements and once all the files is modified here we can go in templates folder where the actual deployments and other related files are present



As shown in above screenshot … here we required only tow files for now

After that to start the deployment of your application run the below command either on the server terminal from where EKS cluster is accessible, or we can put the same command in the jenkins file on the place of “kubectl apply” command.



helm upgrade --install java-helm demo-helm

helm upgrade –install <release name> <helm-charts directory name>

NOTE: --install will first check if the release is created or not, if it is already present then it will simply upgrade the release version, if not present then it will first create the chart and then apply the things \*it is basically for the first time of deployment for the newly created application.

Once the command run successful, we will get the output like below.

