How to setup kubernetes jenkins pipeline on AWS?

Jun 19, 2021 · 15 min read · Share on:

Hello guys It is been quite some time I was planning to write a blog post on ***How to set up your Kubernetes CI CD Jenkins pipeline on AWS step by step?*** It took some time for me to figure out all the details for setting up the continuous pipeline but at last once I was able to deploy it successfully on AWS, it was a sigh of relief for me.

I have listed all the steps which you can find inside the table of content and also I will also record a session and will upload on youtube.

(If you are interested in setting up similar pipeline on premise or using virtual machine the checkout this post)

Table of Content

How to set up AWS Kubernetes Jenkins pipeline? - Part 13

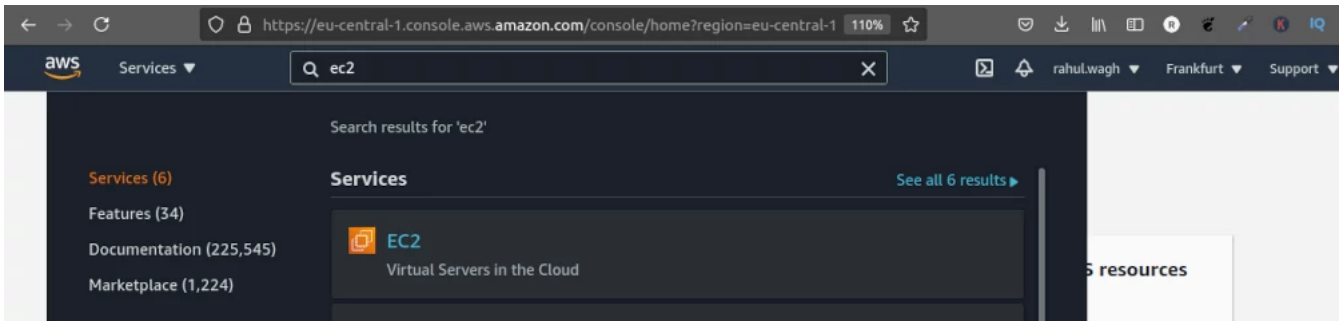
1. Setup an AWS EC2 Instance

The first step would be for us to set up an EC2 instance and on this instance, we will be installing -

1. JDK
2. Jenkins
3. eksctl
4. kubectl

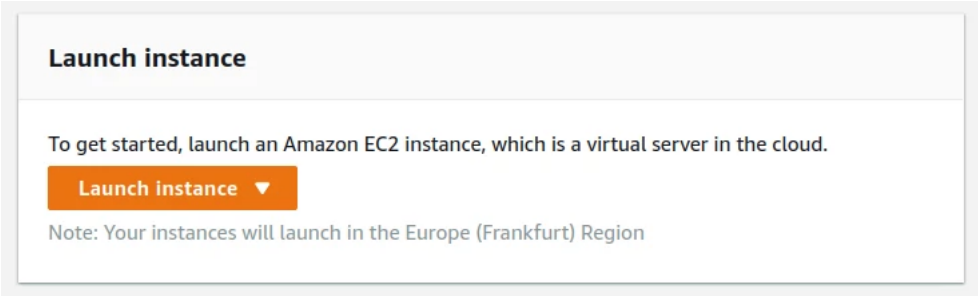
1.1 Launch EC2 instance

1. But first let's head over to AWS and in the search box type in `ec2`.



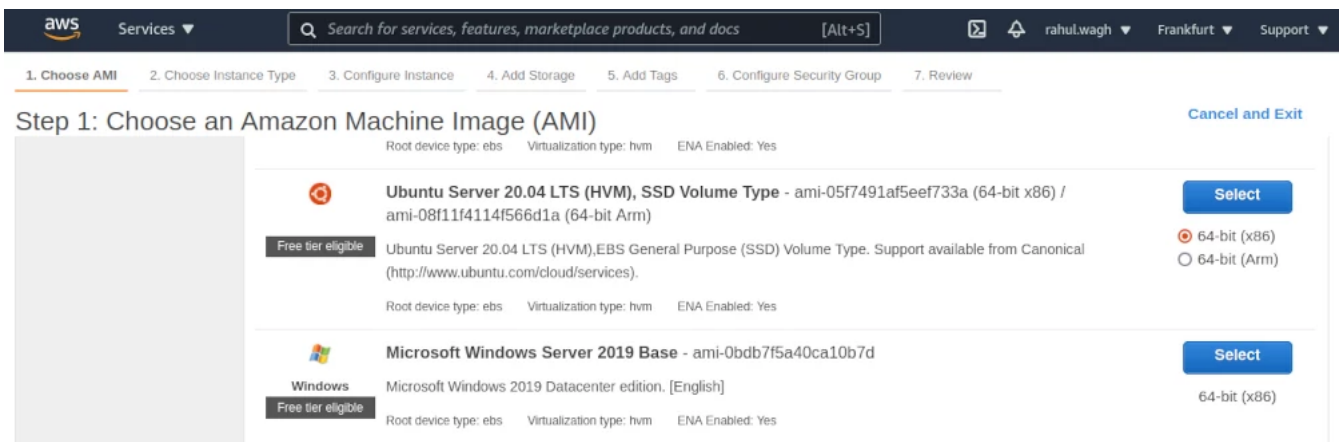
AWS EC2 setup

2. Click on the **EC2** and after that, you need to look for the `Launch Insance` option -



AWS EC2 Launch Instance

3. Now select the image type for the EC2 instance(For this article we are going to select `Ubuntu Server 20.04`)



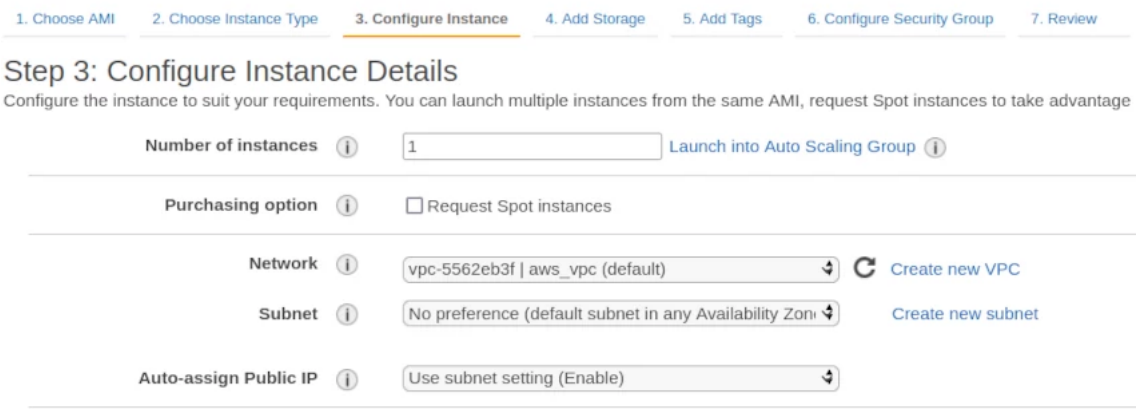
AWS EC2 Launch Instance

For this tutorial, we are going to use **t2.medium** because we will be installing **Jenkins** and t2.micro will not be sufficient enough to setup Jenkins

☑	t2	t2.medium	2	4	EBS only

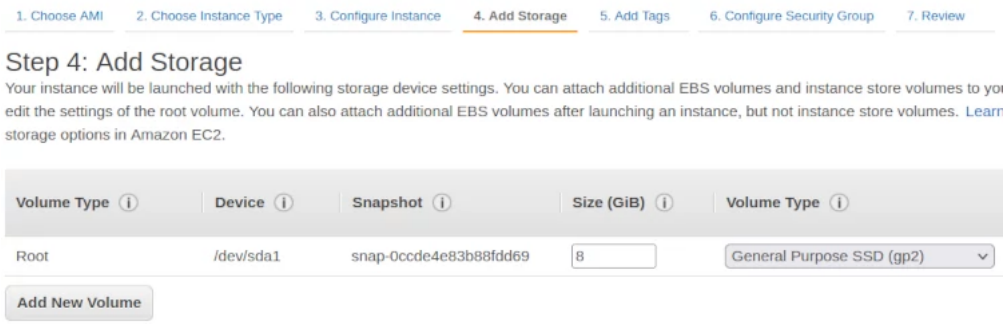
AWS EC2 Choose instance type

5. Configure Instance Details - you can simply verify the detail and proceed to add the storage part.



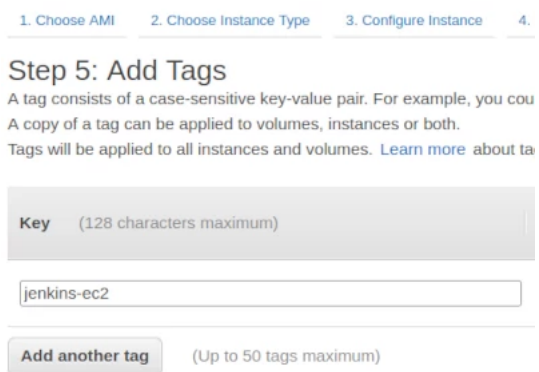
AWS EC2 configure instance details

6. Add storage - In general **8 Gib** of memory is sufficient enough for setting up Jenkins

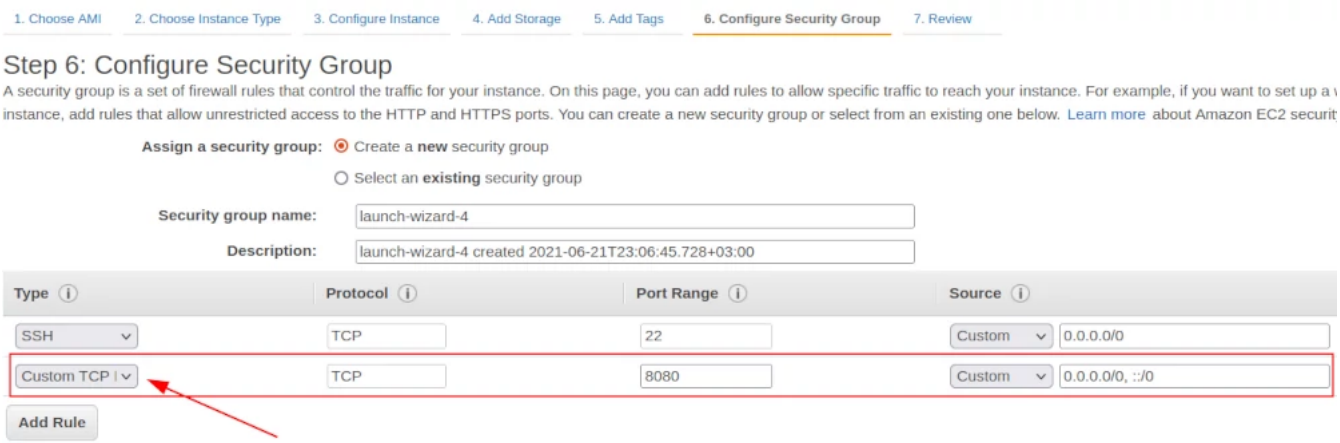


AWS EC2 add storage

7. Add tags - This part is optional but you can add some meaningful tag names to your EC2 instance.

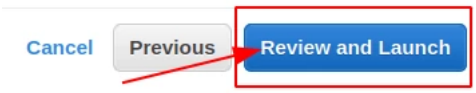


AWS EC2 add tags



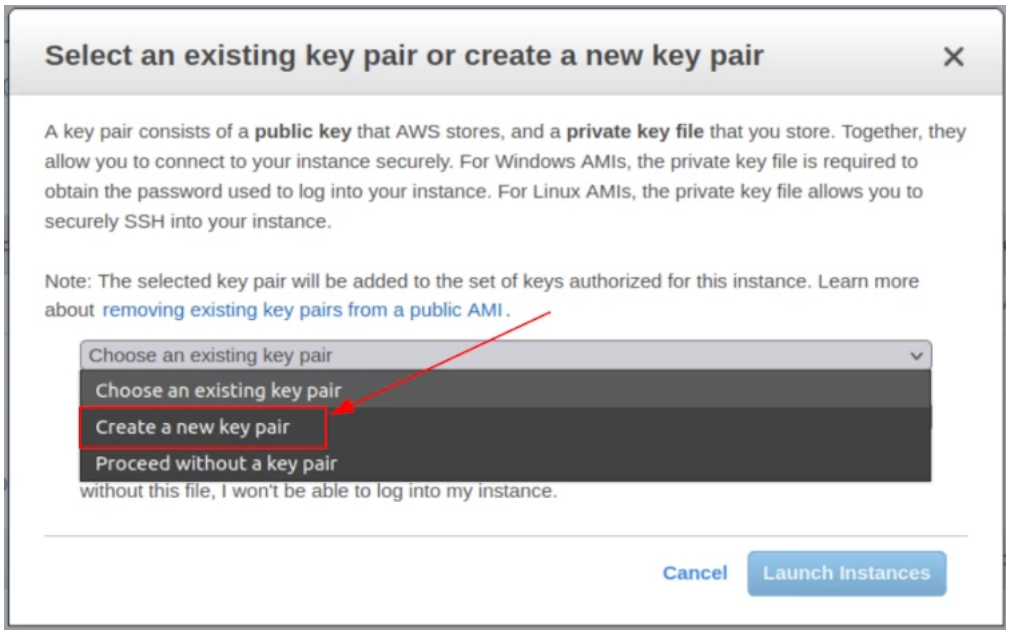
AWS EC2 add security group

9. Finally click on review and launch



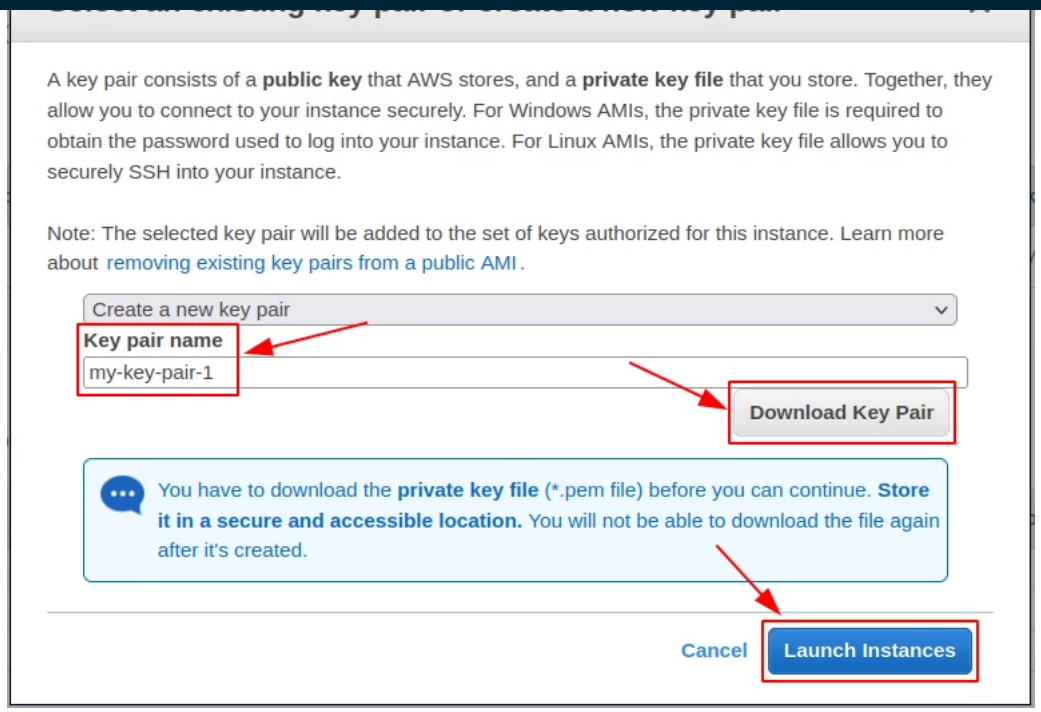
AWS EC2 review and launch

10. But before you launch your EC2 instance you need to ***create and download the key pair(private key and public key)***



AWS EC2 create a new key pair

11. Type in the key pair name and then click on the `Download Key Pair`.



AWS EC2 create a new key pair and download key

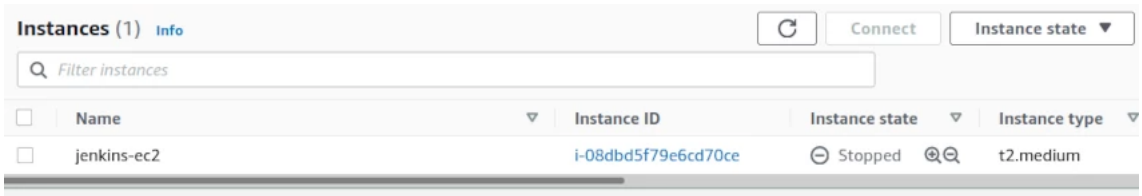
Your ec2 instance should be up and running.

2. Connect to EC2 Instance

Before you connect to your EC2 instance you must start your EC2 instance.

Goto your AWS EC2 dashboard and click on **EC2** after that click on **Instances(running)**.

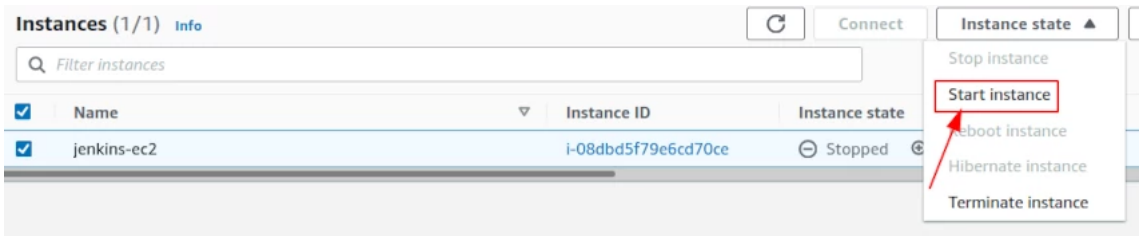
Remove the **running** filter and you should see your EC2 instance which you set up in Step 1.



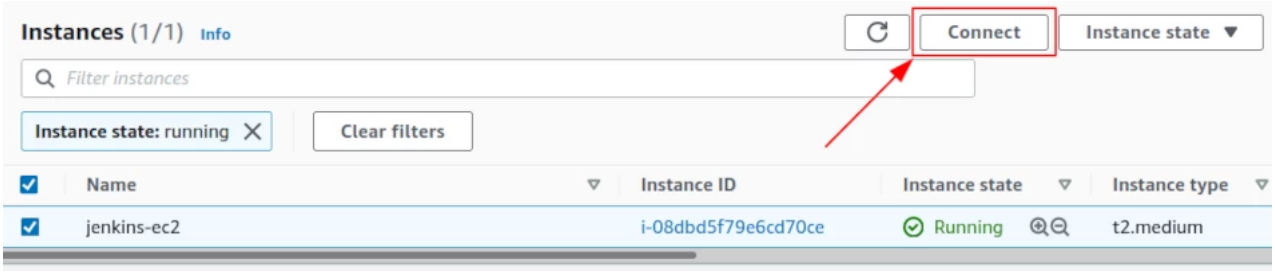
Connect AWS EC2

Now we need to start the EC2 instance and it can be done by first selecting the instance and then

```
Goto-> Instance Start -> Start Instance
```

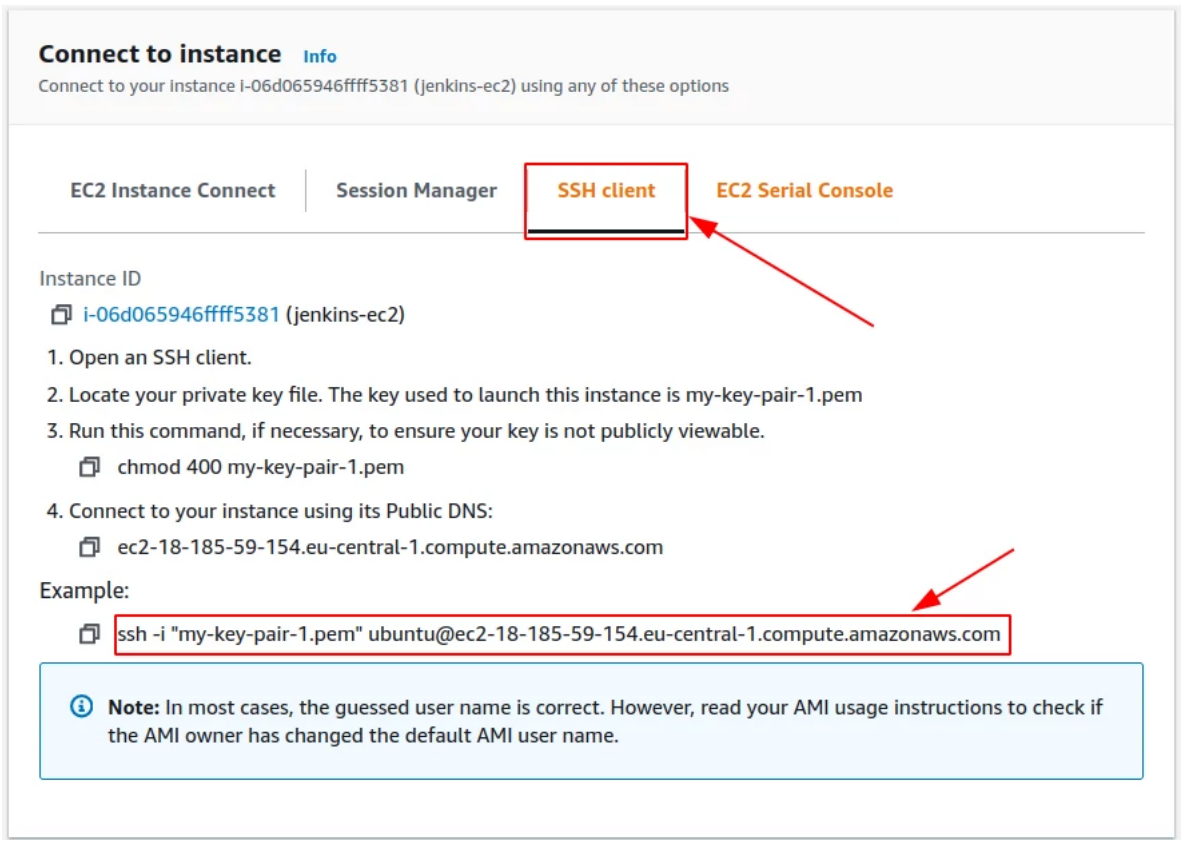


Connect AWS EC2



Connect AWS EC2

We will connect using SSH Client -

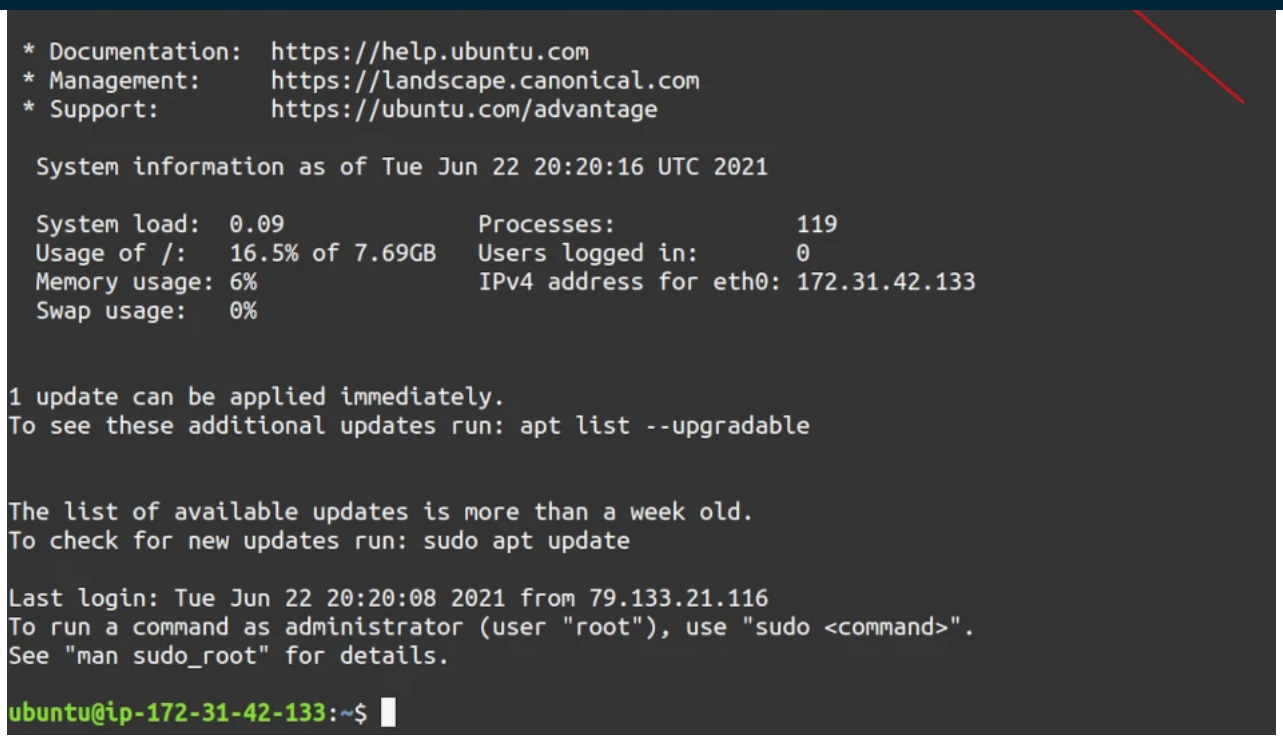


AWS EC2 Connect to instance SSH Client

We will use the `jenkins-ec2.pem` file to connect, so carefully copy the ssh command. (*Following command will be different for you because the IP address of EC2 instance will always be different for you and also you need to supply your server pem file*)

```
ssh -i "my-key-pair-1.pem" ubuntu@ec2-18-185-59-154.eu-central-1.compute.amazonaws.com
```

After successful login, you should see something similar on your terminal -



SSH into AWS ec2 instance using pem file

3. Install JDK on AWS EC2 Instance

The next requirement is we need to install JAVA(JDK) on the EC2 instance.

In the previous step we have seen how to connect and `ssh` into the EC2 instance.

Now before we do the JDK installation lets first update the package manager of the virtual machine -

```
sudo apt-get update
```

Check if you have java already installed onto your EC2 machine by running the following command -

```
java -version
```

In case if you do not have java installed then you will see the following message -

BASH

```
Command 'java' not found, but can be installed with:

sudo apt install openjdk-11-jre-headless  # version 11.0.11+9-0ubuntu2~20.04, or
sudo apt install default-jre              # version 2:1.11-72
sudo apt install openjdk-13-jre-headless  # version 13.0.7+5-0ubuntu1~20.04
sudo apt install openjdk-16-jre-headless  # version 16.0.1+9-1~20.04
sudo apt install openjdk-8-jre-headless   # version 8u292-b10-0ubuntu1~20.04
sudo apt install openjdk-14-jre-headless  # version 14.0.2+12-1~20.04
```

```bash
sudo apt install openjdk-11-jre-headless
```

If you see the following message then you have installed java successfully -

```bash
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
```

4. Install and Setup Jenkins

The next step would be to install the Jenkins. You can follow the official Jenkins Installation guide also. But here I have listed down the steps for installing the Jenkins on the EC2 instance.

First, we need to add the Jenkins repository to the package manager -

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

```
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

After adding the repository link of Jenkins update the package manager

```
sudo apt-get update
```

Then finally install Jenkins using the following command

```bash
sudo apt-get install jenkins
```

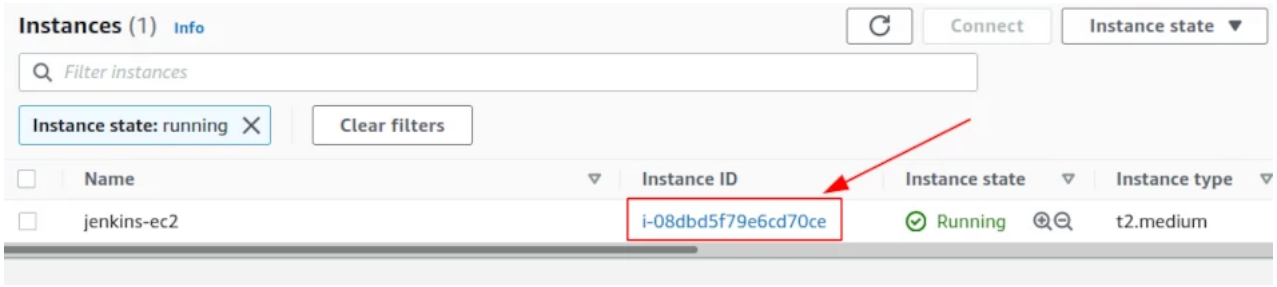
On successful installation, you should see ***Active Status***

```
sudo service jenkins status
```

```bash
● jenkins.service - LSB: Start Jenkins at boot time
     Loaded: loaded (/etc/init.d/jenkins; generated)
     Active: active (exited) since Tue 2021-06-22 20:31:18 UTC; 37s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 16297 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCESS)
```

4.1 Setup Jenkins

After installing jenkins lets go back to `AWS dashboard -> EC2 -> Instances(running)`



AWS EC2 click on instance ID for public IP address

Click on the instance ID as mentioned in the above image.

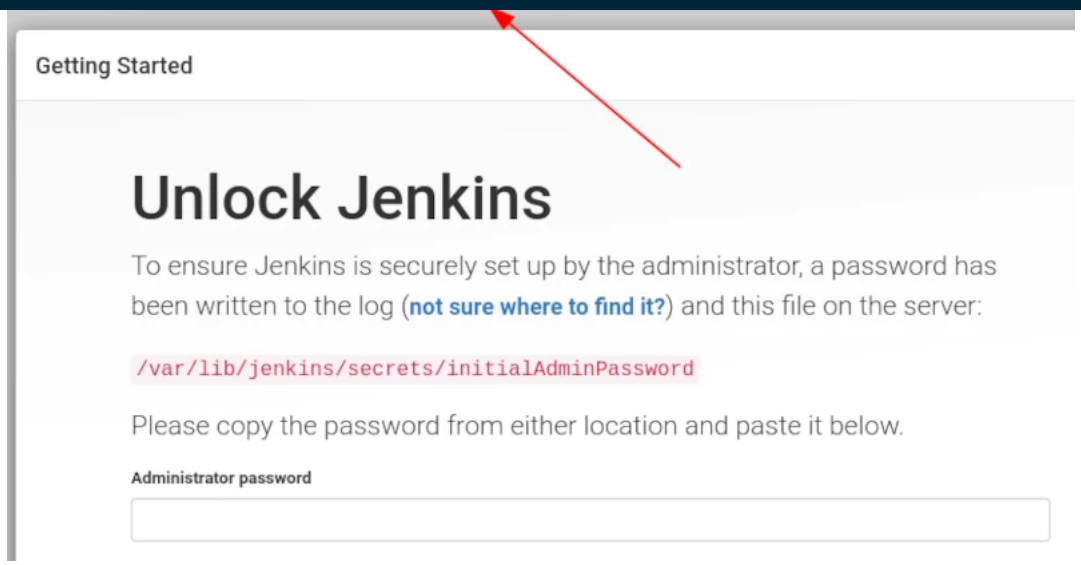
Now we need to find the public IP address of the EC2 machine so that we can access the Jenkins.

Once you click on the instance ID you should see the following page with lots of information about the EC2 instance.

We need to look for `Public IPv4 address`

AWS EC2 click on instance ID for public IP address

Alright now we know the public IP address of the EC2 machine, so now we can access the Jenkins from the browser using the public IP address followed by the port 8080



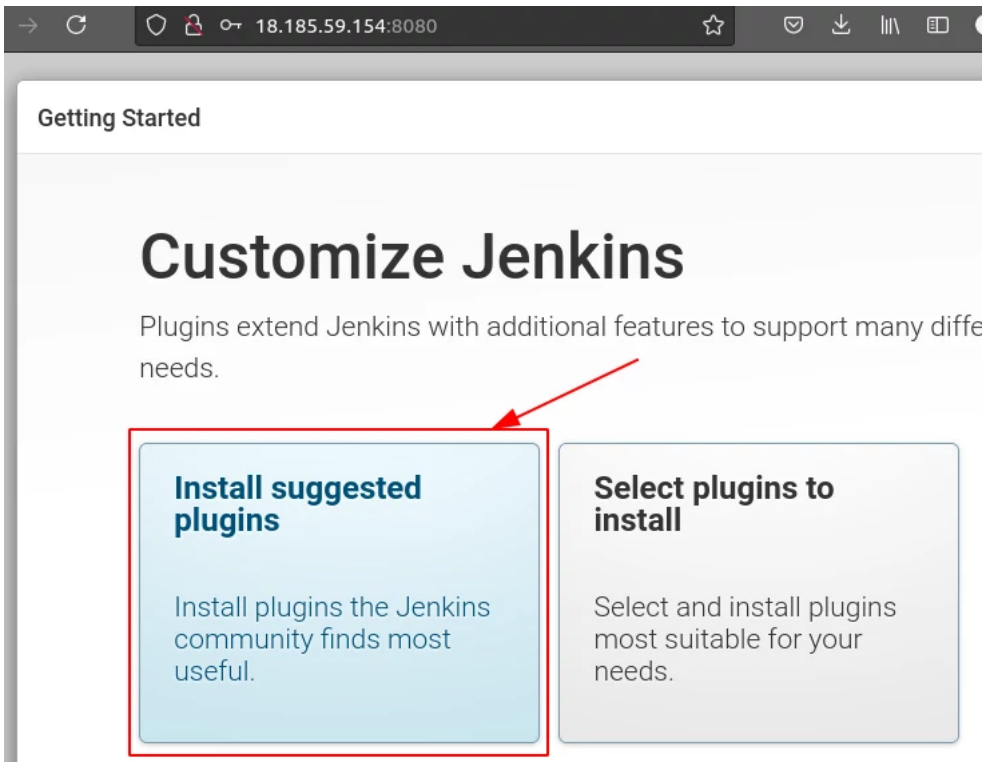
Jenkins access url after installing

If you are installing the Jenkins for the first time then you need to supply the **initialAdminPassword** and you can obtain it from -

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

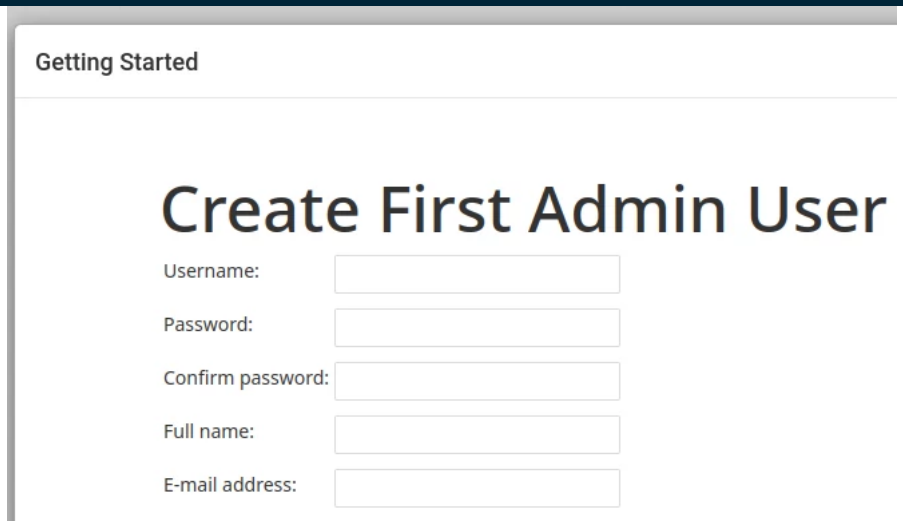
Copy the password and paste it into the initial page of the Jenkins. After that, Jenkins will prompt you for installing the plugins.

Opt for ***install suggested plugin*** -



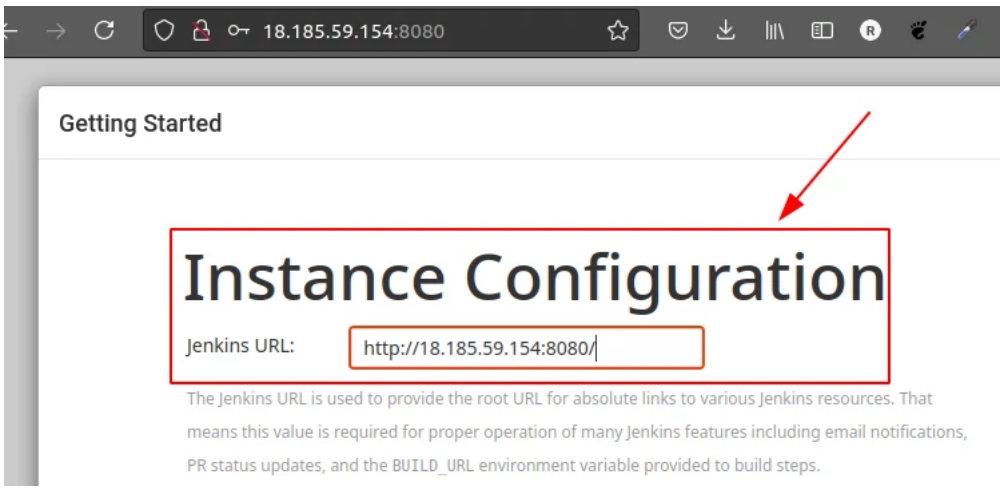
Jenkins install suggested plugins

After completing the installation of the suggested plugin you need to set the ***First Admin User*** for Jenkins



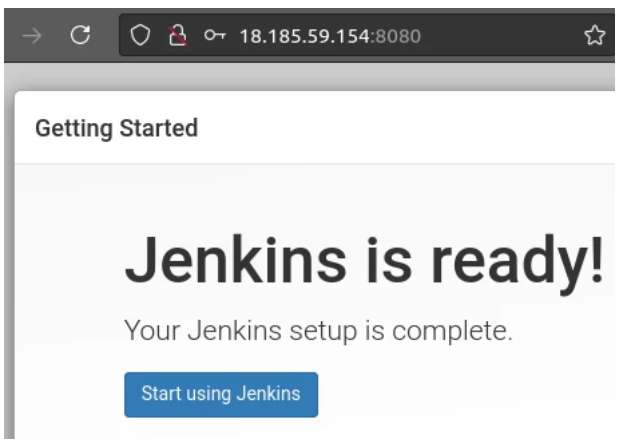
Jenkins set first admin user

Also, check the instance configuration because it will be used for accessing the Jenkins



jenkins instance configuration

And now your Jenkins is ready for use

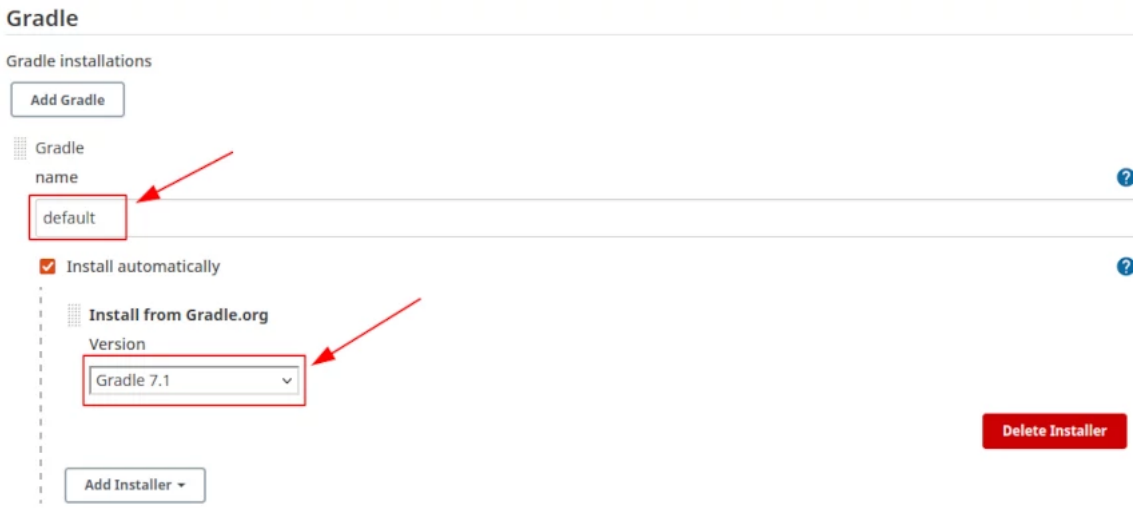


Jenkins is ready for use

4.2 Setup Gradle

For setting up the gradle `Goto -> Manage Jenkins -> Global Tool Configuration -> Gradle`

Here is the screenshot for your reference -



Setup gradle as preferred build

5. Update visudo and assign administration privileges to jenkins user

Now we have installed the Jenkins on the EC2 instance. To interact with the Kubernetes cluster Jenkins will be executing the `shell script` with the Jenkins user, so the Jenkins user should have an administration(superuser) role assigned forehand.

Let's add `jenkins user` as an administrator and also ass `NOPASSWD` so that during the pipeline run it will not ask for `root` password.

Open the file `/etc/sudoers` in ***vi*** mode

```bash
sudo vi /etc/sudoers
```

Add the following line at the end of the file

```bash
jenkins ALL=(ALL) NOPASSWD: ALL
```

After adding the line save and quit the file.

Now we can use Jenkins as root user and for that run the following command -

```bash
sudo su - jenkins
```

6. Install Docker

Now we need to install the docker after installing the Jenkins.

Use the following command for installing the docker -

```BASH
sudo apt install docker.io
```

After installing the docker you can verify it by simply typing the `docker --version` onto the terminal

It should return you with the latest version of the docker

```BASH
Docker version 20.10.2, build 20.10.2-0ubuntu1~20.04.2
```

6.1 Add jenkins user to Docker group

Jenkins will be accessing the Docker for building the application Docker images, so we need to add the Jenkins user to the docker group.

```
sudo usermod -aG docker jenkins
```

7. Install and Setup AWS CLI

Okay so now we have our EC2 machine and Jenkins installed. Now we need to set up the AWS CLI on the EC2 machine so that we can use ***eksctl*** in the later stages

Let us get the installation done for ***AWS CLI***

```BASH
sudo apt install awscli
```

Verify your ***AWS CLI*** installation by running the following command -

```
aws --version
```

It should return you with the version of CLI

```
aws-cli/1.18.69 Python/3.8.5 Linux/5.4.0-1045-aws botocore/1.16.19
```

7.1 Configure AWS CLI

Okay now after installing the AWS CLI, let's configure the ***AWS CLI*** so that it can authenticate and communicate with the AWS environment.

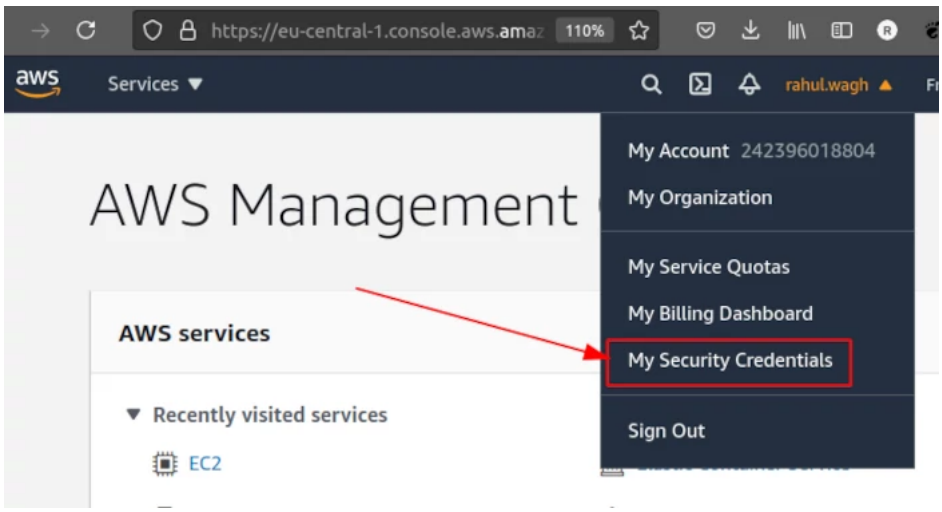
To configure the AWS the first command we are going to run is -

```
aws configure
```

Once you execute the above command it will ask for the following information -

1. AWS Access Key ID [None]:
2. AWS Secret Access Key [None]:
3. Default region name [None]:
4. Default output format [None]:

You can find this information by going into `AWS -> My Security Credentials`

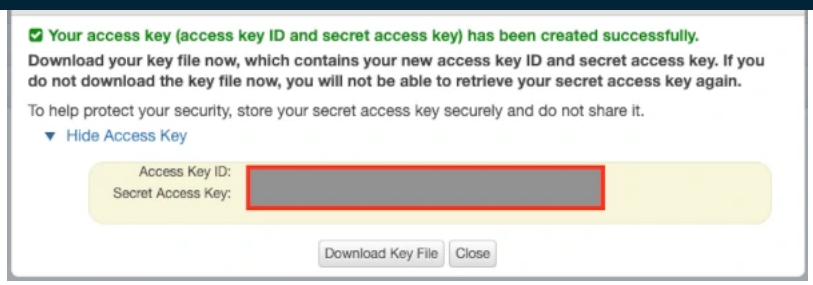


AWS EC2 my security credentials for AWS CLI

Then navigate to ***Access Keys (access key ID and secret access key)***

AWS EC2 access keys for setting up AWS CLI

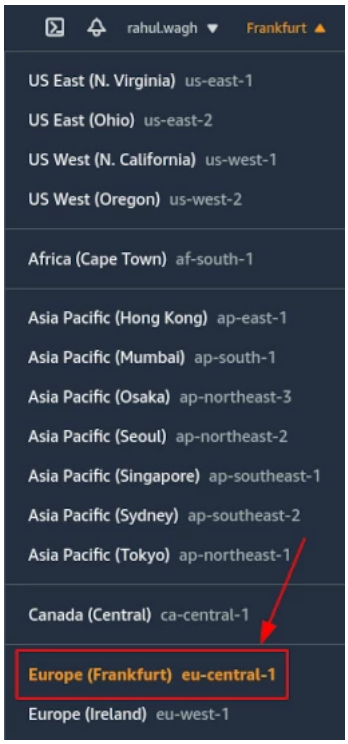
You can click on the ***Create New Access Key*** and it will let you generate - `AWS Access Key ID`, `AWS Secret Access Key`.



AWS EC2 download access key id and secret access key for aws configure

(Note: - Always remember you can only download your access id and secret once, if you misplace the secret and access then you need to recreate the keys again)

Default region name - You can find it from the menu



AWS EC2 region for AWS CLI Default region name

Alright now we have installed and set up AWS CLI.

8. Install and Setup Kubectl

Moving forward now we need to set up the kubectl also onto the EC2 instance where we set up the Jenkins in the previous steps.

Here is the command for installing kubectl

```
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stab
```

```
chmod +x ./kubectl
```

```BASH
sudo mv ./kubectl /usr/local/bin
```

Verify the kubectl installation

Verify the kubectl installation by running the command `kubectl version` and you should see the following output

```
Client Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.2", GitCommit:"092fbfbf53427de67cac1e9fa54aaa09a28371d7", GitTreeState
Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fversion%3Ftimeout%3D32s'/><script>window
```

9. Install and Setup eksctl

The next thing which we are gonna do is to install the eksctl, which we will be using to create AWS EKS Clusters.

Okay, the first command which we are gonna run to install the ***eksctl***

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

```BASH
sudo mv /tmp/eksctl /usr/local/bin
```

Verify the installation by running the command -

```BASH
eksctl version
```

And it will return you with the version -

```BASH
0.52.0
```

So at the time of installation, I had `0.52.0`

```
$ eksctl version
0.54.0
```

eksctl version after installation

Installing eksctl on other OS

For Mac OS

```
brew tap weaveworks/tap
brew install weaveworks/tap/eksctl
```

For Windows

```BASH
chocolatey install eksctl
```

or Scoop

```BASH
scoop install eksctl
```

10. Create eks cluster using eksctl

In all the previous 9 steps we were preparing our AWS environment. Now in this step, we are going to create EKS cluster using `eksctl`

You need the following in order to run the eksctl command

1. **Name of the cluster :** --name jhooq-test-cluster
2. **Version of Kubernetes :** --version 1.17
3. **Region :** --name eu-central-1
4. **Nodegroup name/worker nodes :** worker-nodes
5. **Node Type :** t2.micro
6. **Number of nodes:** -nodes 2

Here is the eksctl command -

```
eksctl create cluster --name jhooq-test-cluster --version 1.17 --region eu-central-1 --nodegroup-name worker-nodes --node-type t2.micro --no
```

*(*Note - Be patient with the above command because it may take 20-30 minutes to complete)*

For me it almost took 20 minutes, here are the timestamps

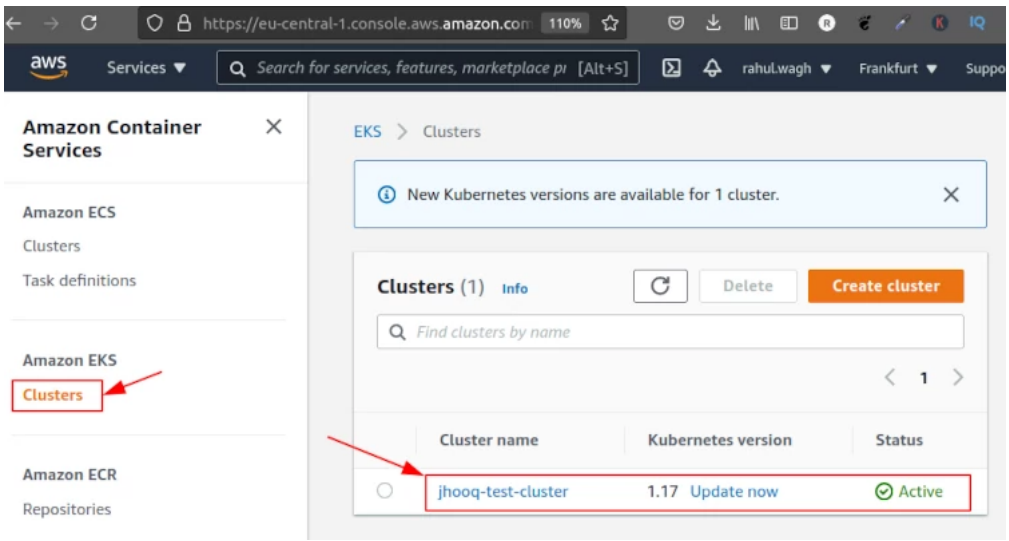
```
jenkins@ip-172-31-42-133:~$ eksctl create cluster --name jhooq-test-cluster --version 1.17 --region eu-central-1 --noo
egroup-name worker-nodes --node-type t2.micro --nodes 2
2021-06-23 18:49:04 [ℹ]  eksctl version 0.54.0
2021-06-23 18:49:04 [ℹ]  using region eu-central-1
2021-06-23 18:49:04 [ℹ]  setting availability zones to [eu-central-1a eu-central-1b eu-central-1c]
2021-06-23 18:49:04 [ℹ]  subnets for eu-central-1a - public:192.168.0.0/19 private:192.168.96.0/19
2021-06-23 18:49:04 [ℹ]  subnets for eu-central-1b - public:192.168.32.0/19 private:192.168.128.0/19
2021-06-23 18:49:04 [ℹ]  subnets for eu-central-1c - public:192.168.64.0/19 private:192.168.160.0/19
```

eksctl kuberenetes setup start time

```
2021-06-23 19:08:43 [ℹ]  nodegroup "worker-nodes" has 2 node(s)
2021-06-23 19:08:43 [ℹ]  node "ip-192-168-15-31.eu-central-1.compute.internal" is ready
2021-06-23 19:08:43 [ℹ]  node "ip-192-168-82-201.eu-central-1.compute.internal" is ready
2021-06-23 19:08:44 [ℹ]  kubectl command should work with "/var/lib/jenkins/.kube/config", try 'kubectl get nodes'
2021-06-23 19:08:44 [✔]  EKS cluster "jhooq-test-cluster" in "eu-central-1" region is ready
```

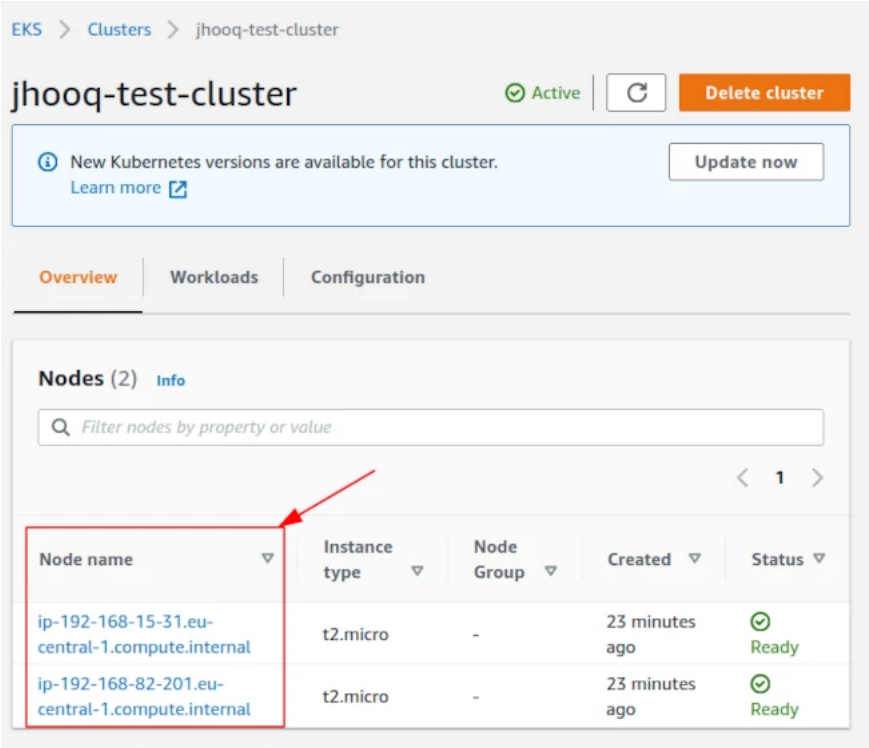
eksctl kuberenetes setup end time

You can go back to your AWS dashboard and look for `Elastic Kubernetes Service -> Clusters`



use eksctl to setup AWS EKS cluster

Click on the **Cluster Name** to verify the worker nodes -



eksctl worker nodes

11. Add Docker and GitHub Credentials into Jenkins

As we know Kubernetes is a container orchestration tool and container management we are using docker.

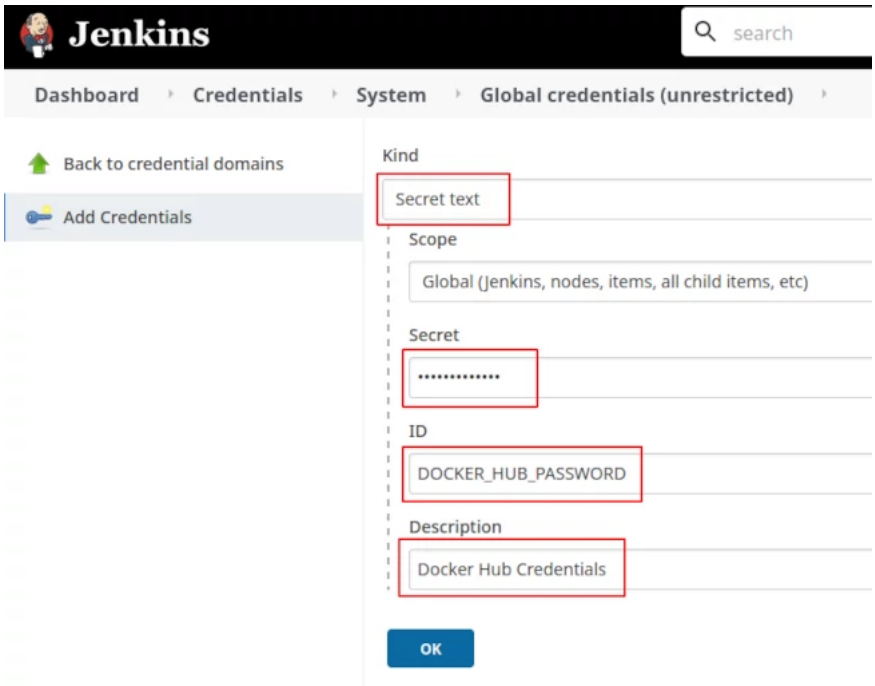
(*In case if you haven't set up Docker Hub Account then please create a DockerHub Account because we are gonna need it.*)

Alright so if you are reading this line then I am assuming you have a DockerHub Account and GitHub Account.

11.1 Setup Docker Hub Secret Text in Jenkins

You can set the docker credentials by going into -

```
Goto -> Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials
```

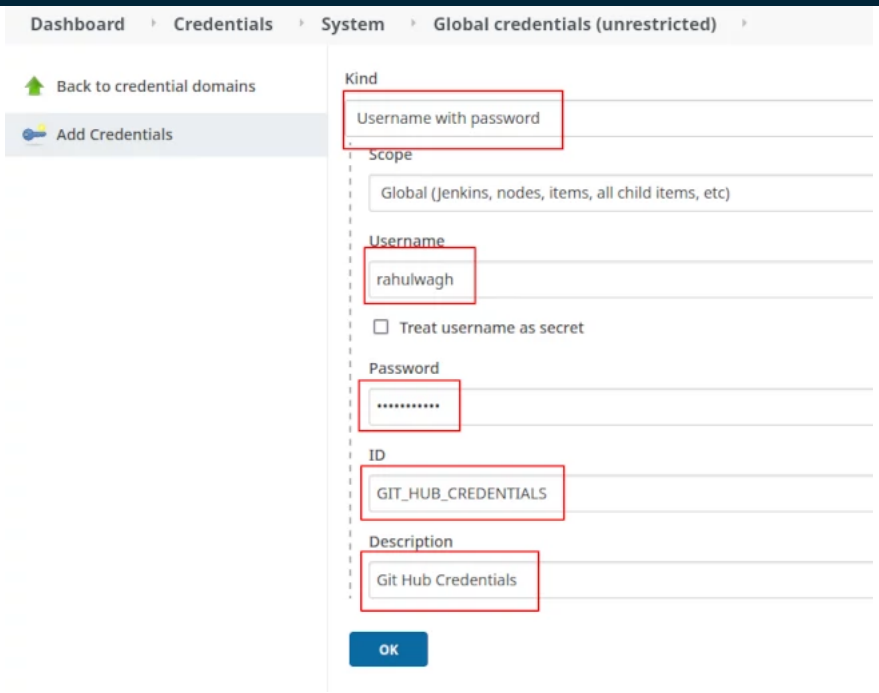


Jenkins add credentials as secret text for docker

11.2 Setup GitHub Username and password into Jenkins

Now we add one more username and password for GitHub.

```
Goto -> Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials
```



Jenkins add GitHub credentials

12. Add jenkins stages

Okay, now we can start writing out the Jenkins pipeline for deploying the Spring Boot Application into the Kubernetes Cluster.

12.1 Jenkins stage-1 : Checkout the GitHub Repository

Add the following Jenkins script for checking out the GitHub Repository -

```BASH
    stage("Git Clone"){

       git credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/rahulwagh/k8s-jenkins-aws'
   }
```

12.2 Jenkins stage-2 : Gradle compilation and build

Now after checking out the repository let compile and build the application using Gradle

```
stage('Gradle Build') {
    sh './gradlew build'
}
```

12.3 Jenkins stage-3 : Create Docker Container and push to Docker Hub

After successful compilation and build let's create a Docker image and push to the docker hub

```
    sh 'docker version'
    sh 'docker build -t jhooq-docker-demo .'
    sh 'docker image list'
    sh 'docker tag jhooq-docker-demo rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'
}

stage("Push Image to Docker Hub"){
        sh 'docker push  rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'
}
```

12.4 Jenkins stage-4 : Kubernetes deployment

Finally, do the Kubernetes deployment

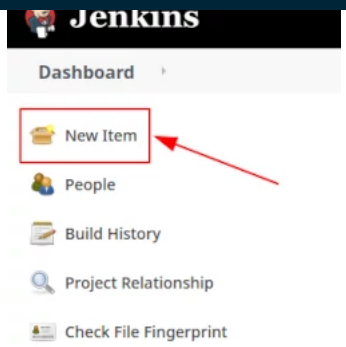
```
BASH
stage("kubernetes deployment"){
  sh 'kubectl apply -f k8s-spring-boot-deployment.yml'
}
```

Here is the complete final script for Jenkins pipeline -

```
node {

    stage("Git Clone"){

        git credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/rahulwagh/k8s-jenkins-aws'
    }

     stage('Gradle Build') {

       sh './gradlew build'

    }

    stage("Docker build"){
        sh 'docker version'
        sh 'docker build -t jhooq-docker-demo .'
        sh 'docker image list'
        sh 'docker tag jhooq-docker-demo rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'
    }

    withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD', variable: 'PASSWORD')]) {
        sh 'docker login -u rahulwagh17 -p $PASSWORD'
    }

    stage("Push Image to Docker Hub"){
        sh 'docker push  rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'
    }

    stage("kubernetes deployment"){
        sh 'kubectl apply -f k8s-spring-boot-deployment.yml'
    }
}
```

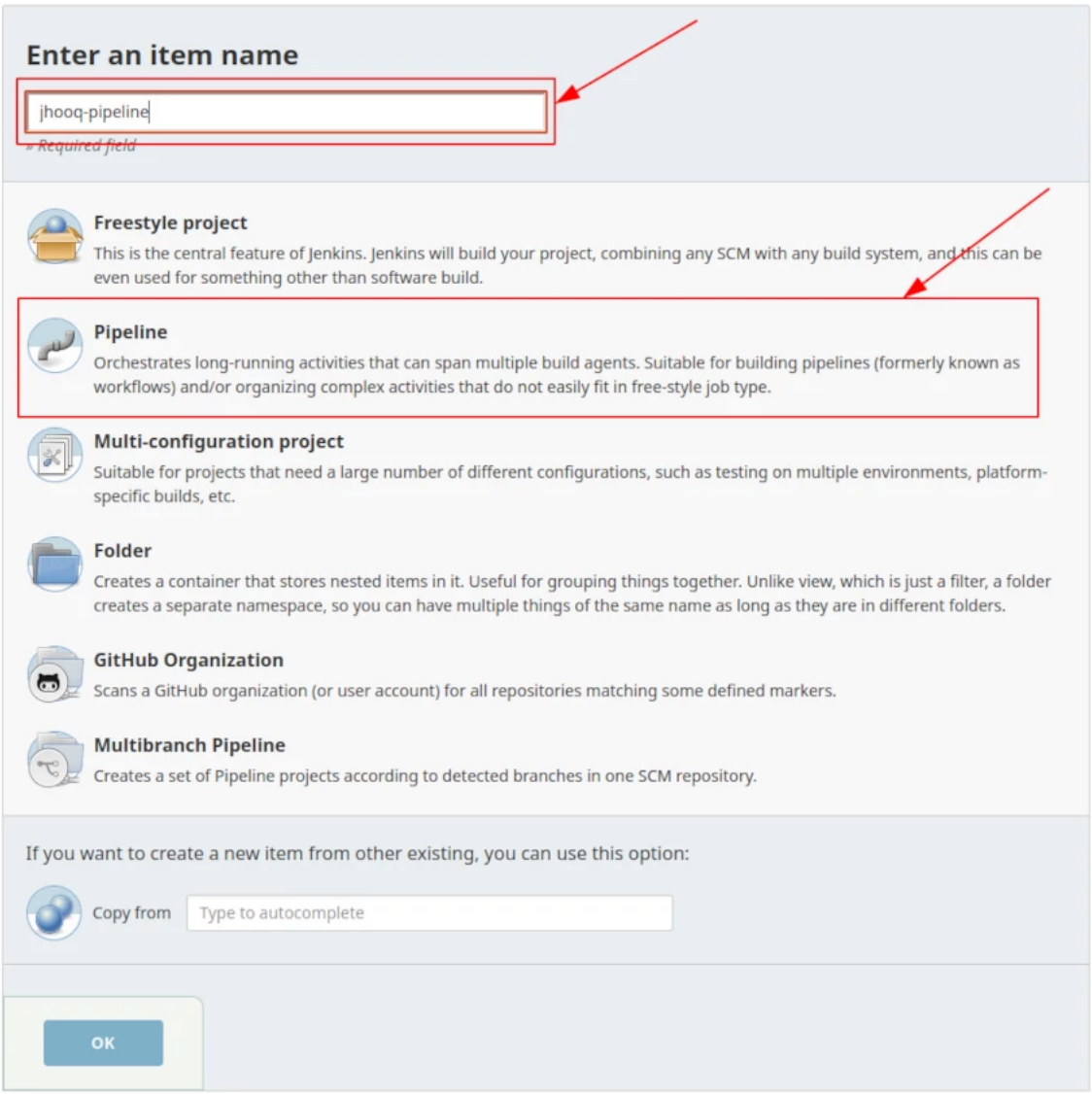
13. Build, deploy and test CI/CD pipeline

Create new Pipeline: Goto Jenkins Dashboard or Jenkins home page click on *New Item*



Jenkins new item for creating the pipeline

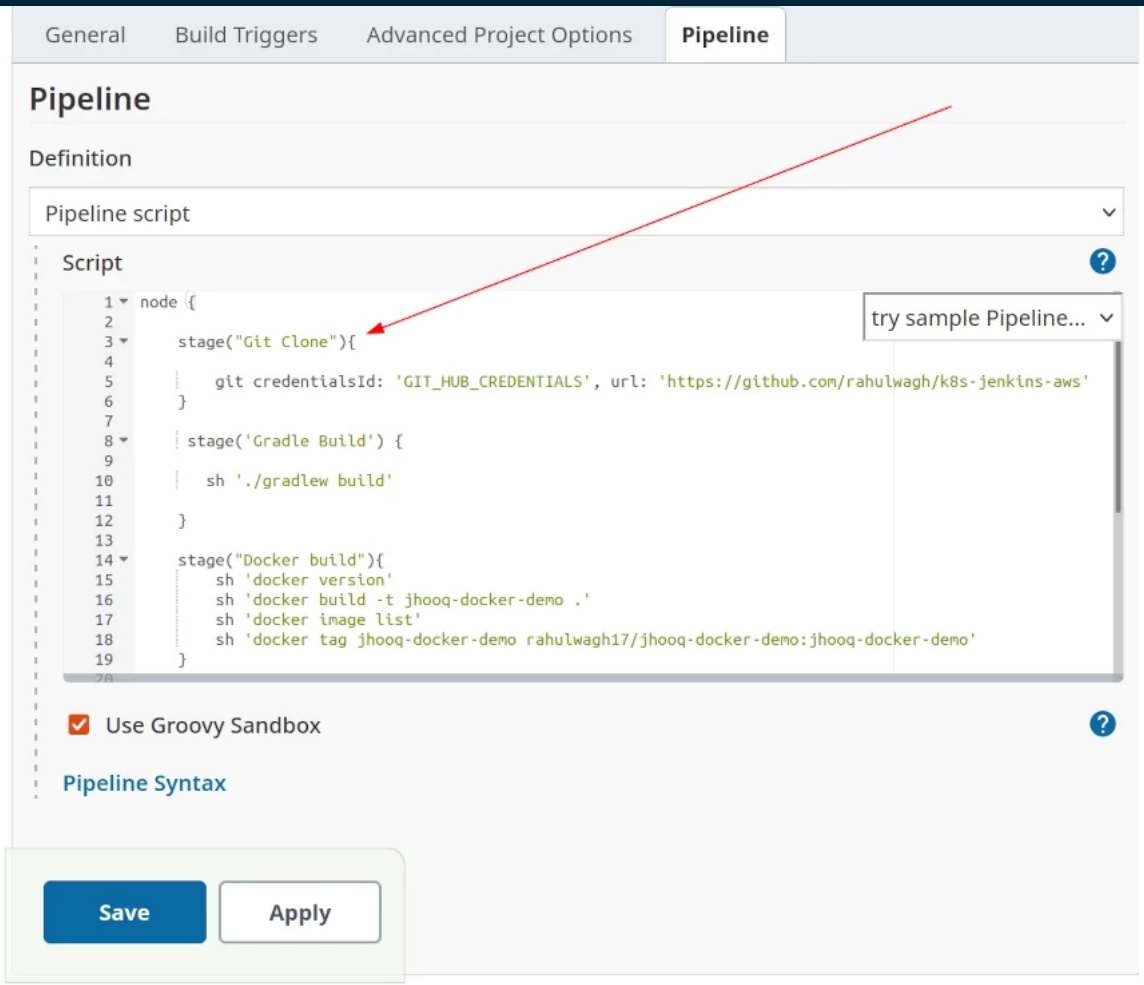
Pipeline Name: Now enter Jenkins pipeline name and select Pipeline



Enter an item name for jenkins pipeline

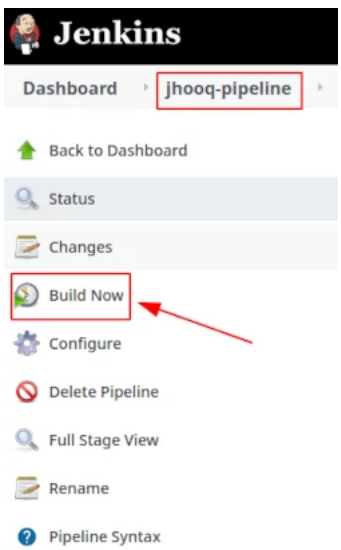
Add pipeline script: `Goto -> Configure` and then pipeline section.

Copy the Jenkins script from Step 12 and paste it there.



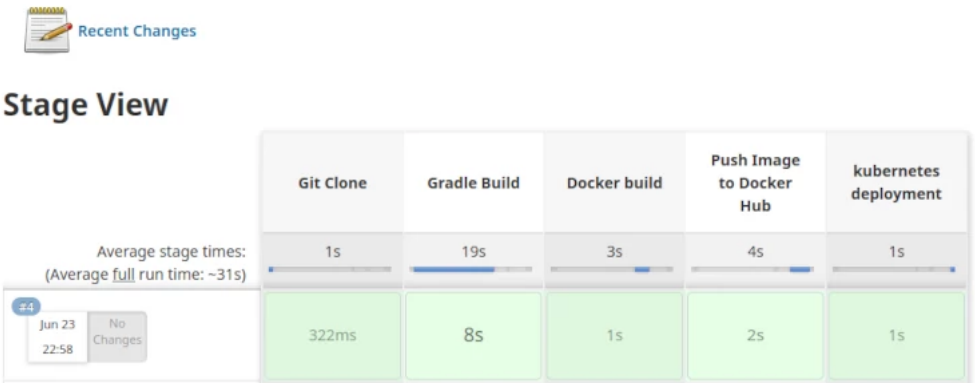
Add the pipeline script into the pipeline definition script section

Build and Run Pipeline: Now goto pipeline and click on build now



Add the pipeline script into the pipeline definition script section

Verify the build status:



Jenkins stages status

Verify using kubectl commands

You can also verify the Kubernetes deployment and service with `kubectl` command .e.g `kubectl get deployments`, `kubectl get service`

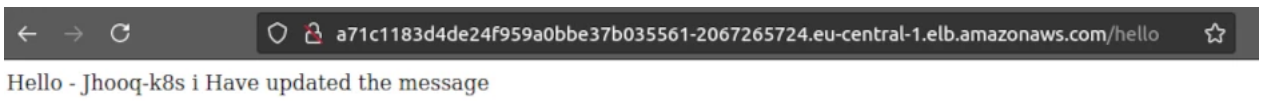
```
jenkins@ip-172-31-42-133:~$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
jhooq-springboot    2/3     3            2           33m
```

Kubectl deployment status check after deploying application on AWS

```
jenkins@ip-172-31-42-133:~$ kubectl get service
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)          AGE
jhooq-springboot  LoadBalancer   10.100.230.63   a71c1183d4de24f959a0bbe37b035561-2067265724.eu-central-1.elb.amazonaws.com   80:30965/TCP     35m
kubernetes        ClusterIP      10.100.0.1      <none>                                                                   443/TCP          95m
```

Kubectl service status check after deploying application on AWS

You can access the rest end point from browser using the EXTERNAL-IP address



Jenkins stages status

I hope you enjoyed this article. For other devops topic like Terraform, Helm Chart you can check the category and my YouTube Channel

Learn more On Kubernetes -

1. Setup kubernetes on Ubuntu
2. Setup Kubernetes on CentOs
3. Setup HA Kubernetes Cluster with Kubespray
4. Setup HA Kubernetes with Minikube
5. Setup Kubernetes Dashboard for local kubernetes cluster
6. Setup Kubernetes Dashboard On GCP(Google Cloud Platform)
7. How to use Persistent Volume and Persistent Volume Claims in Kubernetes

10. Setting up Ingress controller NGINX along with HAproxy inside Kubernetes cluster
11. CI/CD Kubernetes | Setting up CI/CD Jenkins pipeline for kubernetes
12. kubectl export YAML | Get YAML for deployed kubernetes resources(service, deployment, PV, PVC....)
13. How to setup kubernetes jenkins pipeline on AWS?
14. Implementing Kubernetes liveness, Readiness and Startup probes with Spring Boot Microservice Application?
15. How to fix kubernetes pods getting recreated?
16. How to delete all kubernetes PODS?
17. How to use Kubernetes secrets?
18. Share kubernetes secrets between namespaces?
19. How to Delete PV(Persistent Volume) and PVC(Persistent Volume Claim) stuck in terminating state?
20. Delete Kubernetes POD stuck in terminating state?

Search...

Categories

TERRAFORM 43 KUBERNETES 26 DOCKER 22 HELM-CHART 11 AWS 7 BLOGGING 6 SPRING-BOOT 5
SSL 5 QUARKUS 4 GITHUB 3 KUBESPRAY 3 PROMETHEUS-GRAFANA 3 VAGRANT 3 NGINX 2
ALL CATEGORIES

Series

TERRAFORM 42 KUBERNETES 27 DOCKER 22 HELM-CHART 11 AWS 2 LINUX-COMMANDS 1

Tags

KUBERNETES 18 HELM-CHART 10 BLOGGING 4 QUARKUS 4 DOCKER 3 GITHUB 3 SSL 3
KUBESPRAY 2 SPRING-BOOT 2 HADOOP 1 INDEX 1 NGINX 1 TERRAFORM 1

Recent Posts

- What is user_data in Terraform?
- Why you should not store terraform state file(.tfstate) inside Git Repository?
- How to import existing resource using terraform import comand?
- Terraform - A detailed guide on setting up ALB(Application Load Balancer) and SSL?
- How to release(delete) Elastic IP from AWS?
- Testing Infrastructure as Code with Terraform?
- How to remove a resource from Terraform state?
- What is Terraform null Resource?

Rahul Wagh

Its all about Open Source and DevOps, here I talk about Kubernetes, Docker, Java, Spring boot and practices.