

How to setup kubernetes jenkins pipeline on AWS?

Jun 19, 2021 · 15 min read · Share on: [Twitter](#) [Facebook](#) [LinkedIn](#) [Email](#)

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

1. [Setup an AWS EC2 Instance](#)
2. [Connect to EC2 Instance](#)
3. [Install JDK on AWS EC2 Instance](#)
4. [Install and Setup Jenkins](#)
5. [Update visudo and assign administrative privileges to Jenkins user](#)
6. [Install Docker](#)
7. [Install and Setup AWS CLI](#)
8. [Install and Setup Kubectl](#)
9. [Install and Setup eksctl](#)
10. [Create eks cluster using eksctl](#)
11. [Add Docker and GitHub Credentials into Jenkins](#)
12. [Add jenkins stages](#)
13. [Build, deploy and test CI CD pipeline](#)

[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 Instance** 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.

[1. Choose AMI](#) [2. Choose Instance Type](#) [3. Configure Instance](#) [4. Add Storage](#) [5. Add Tags](#) [6. Configure Security Group](#) [7. Review](#)

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage

Number of instances	<input type="text" value="1"/>	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-5562eb3f aws_vpc (default)	Create new VPC
Subnet	No preference (default subnet in any Availability Zone)	Create new subnet
Auto-assign Public IP		

AWS EC2 configure instance details

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

[1. Choose AMI](#) [2. Choose Instance Type](#) [3. Configure Instance](#) [4. Add Storage](#) [5. Add Tags](#) [6. Configure Security Group](#) [7. Review](#)

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance. You can edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more about storage options in Amazon EC2](#).

Volume Type	Device	Snapshot	Size (GiB)	Volume Type
Root	/dev/sda1	snap-0ccde4e83b88fdd69	<input type="text" value="8"/>	General Purpose SSD (gp2)
Add New Volume				

AWS EC2 add storage

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

[1. Choose AMI](#) [2. Choose Instance Type](#) [3. Configure Instance](#) [4. Add Tags](#)

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could add a tag named `Environment` with the value `Production`. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more about tags](#)

Key <small>(128 characters maximum)</small>
<input type="text" value="jenkins-ec2"/>
Add another tag <small>(Up to 50 tags maximum)</small>

AWS EC2 add tags

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags **6. Configure Security Group** 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source
SSH	TCP	22	Custom 0.0.0.0/0
Custom TCP	TCP	8080	Custom 0.0.0.0/0, ::/0

[Add Rule](#)



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)**

Select an existing key pair or create a new key pair X

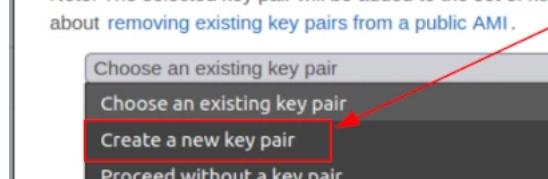
A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair
Choose an existing key pair
Create a new key pair
Proceed without a key pair

without this file, I won't be able to log into my instance.

[Cancel](#) [Launch Instances](#)

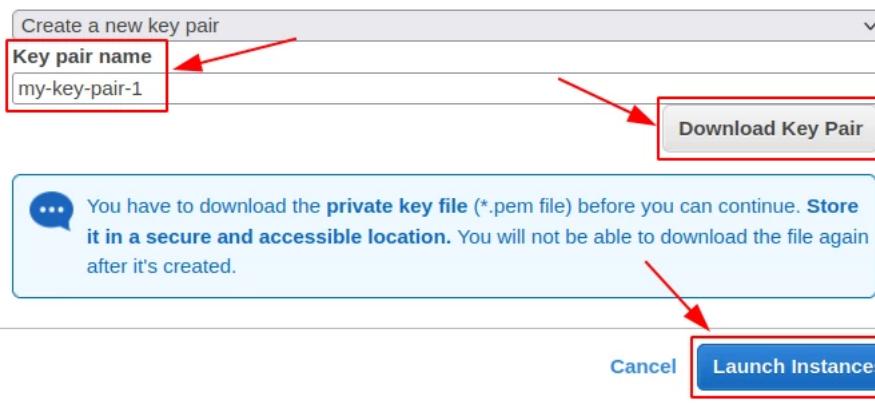


AWS EC2 create a new key pair

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

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).



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**.

Instances (1) Info					
C Connect Instance state ▾					
<input type="text"/> Filter instances					
Name	Instance ID	Instance state	Instance type		
jenkins-ec2	i-08dbd5f79e6cd70ce	Stopped	t2.medium		

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

Instances (1/1) Info					
C Connect Instance state ▾					
<input type="text"/> Filter instances					
Name	Instance ID	Instance state	Instance type		
jenkins-ec2	i-08dbd5f79e6cd70ce	Stopped	t2.medium	Stop instance Start instance Start instance Reboot instance Hibernate instance Terminate instance	

Connect AWS EC2

Instances (1/1) [Info](#)

Filter instances

Instance state: running [X](#) [Clear filters](#)

Name	Instance ID	Instance state	Instance type
<input checked="" type="checkbox"/> jenkins-ec2	i-08dbd5f79e6cd70ce	<input checked="" type="checkbox"/> Running Details	t2.medium

Connect AWS EC2

We will connect using SSH Client -

Connect to instance [Info](#)

Connect to your Instance i-06d065946ffff5381 (jenkins-ec2) using any of these options

EC2 Instance Connect [Session Manager](#) [SSH client](#) [EC2 Serial Console](#)

Instance ID
 [i-06d065946ffff5381 \(jenkins-ec2\)](#)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is my-key-pair-1.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 chmod 400 my-key-pair-1.pem

4. Connect to your instance using its Public DNS:
 [ec2-18-185-59-154.eu-central-1.compute.amazonaws.com](#)

Example:
 [ssh -i "my-key-pair-1.pem" ubuntu@ec2-18-185-59-154.eu-central-1.compute.amazonaws.com](#)

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

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 -

```
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
```

System information as of Tue Jun 22 20:20:16 UTC 2021

```
System load: 0.09          Processes: 119
Usage of /: 16.5% of 7.69GB Users logged in: 0
Memory usage: 6%           IPv4 address for eth0: 172.31.42.133
Swap usage: 0%
```

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

```
Last login: Tue Jun 22 20:20:08 2021 from 79.133.21.116
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

ubuntu@ip-172-31-42-133:~\$ █

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)
```

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

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

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** -

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins install suggested plugins

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

Getting Started

Create First Admin User

Username:	<input type="text"/>
Password:	<input type="password"/>
Confirm password:	<input type="password"/>
Full name:	<input type="text"/>
E-mail address:	<input type="text"/>

Jenkins set first admin user

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

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

jenkins instance configuration

And now your Jenkins is ready for use

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

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 -

Gradle

Gradle installations

Add Gradle

Gradle name
default

Install automatically

Install from Gradle.org

Version
Gradle 7.1

Delete Installer

Add Installer ▾

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 beforehand.

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 -

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

BASH

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

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

BASH

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**

```
sudo apt install awscli
```

BASH

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

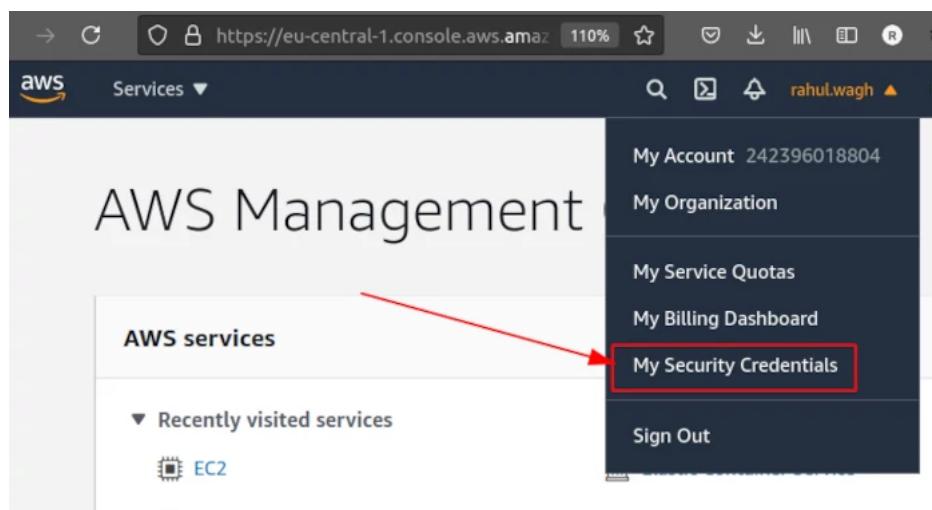
Okay now after installing the AWS CLI, let's configure the **AWS CLI** so that it can authenticate and communicate with the AWS environment.

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

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)**

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage or change your password, see [Change your password](#). To learn more about the types of AWS credentials and how they're used, see [AWS credentials](#).

▼ Password

You use an email address and password to sign in to secure pages on AWS. For your protection, create a password that contains many characters, including uppercase and lowercase letters, numbers, and symbols. Click here to change the password, name, or email address for your root AWS account.

▲ Multi-factor authentication (MFA)

▲ Access keys (access key ID and secret access key) **(This section is highlighted with a red box)**

▲ CloudFront key pairs

▲ X.509 certificate

▲ Account identifiers

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](#).

Your access key (access key ID and secret access key) has been created successfully.

Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

▼ Hide Access Key

Access Key ID:

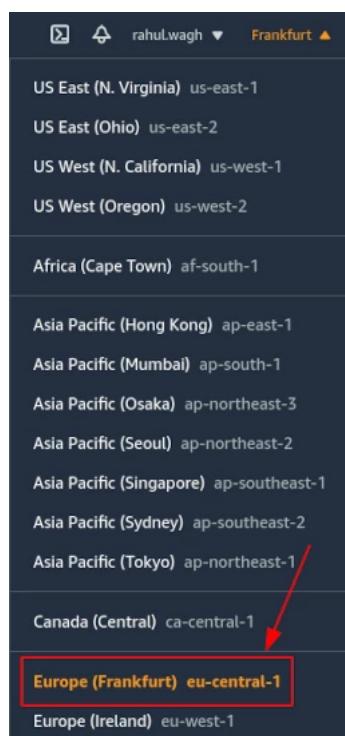
Secret Access Key:

[Download Key File](#) [Close](#)

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

```
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

chocolatey install eksctl

BASH

or Scoop

scoop install eksctl

BASH

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 --nodes 2
```

(*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 --nodegroup-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 kubernetes 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 kubernetes setup end time

You can go back to your AWS dashboard and look for [Elastic Kubernetes Service -> Clusters](#)

The screenshot shows the AWS EKS Clusters page. On the left, there's a sidebar for 'Amazon Container Services' with 'Amazon ECS' and 'Amazon EKS' sections. The 'Clusters' section under EKS is highlighted with a red box and has a red arrow pointing to it. The main area shows a table with one cluster entry:

Cluster name	Kubernetes version	Status
jhoog-test-cluster	1.17 Update now	Active

use `eksctl` to setup AWS EKS cluster

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

The screenshot shows the AWS EKS Cluster Overview page for 'jhoog-test-cluster'. The navigation bar includes 'Overview', 'Workloads', and 'Configuration'. The 'Overview' tab is selected. Below it, the 'Nodes (2)' section is shown with a red box around the first two rows of the table. A red arrow points from the 'Nodes' link in the navigation bar to this table.

Node name	Instance type	Node Group	Created	Status
ip-192-168-15-31.eu-central-1.compute.internal	t2.micro	-	23 minutes ago	Ready
ip-192-168-82-201.eu-central-1.compute.internal	t2.micro	-	23 minutes ago	Ready

`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](#).

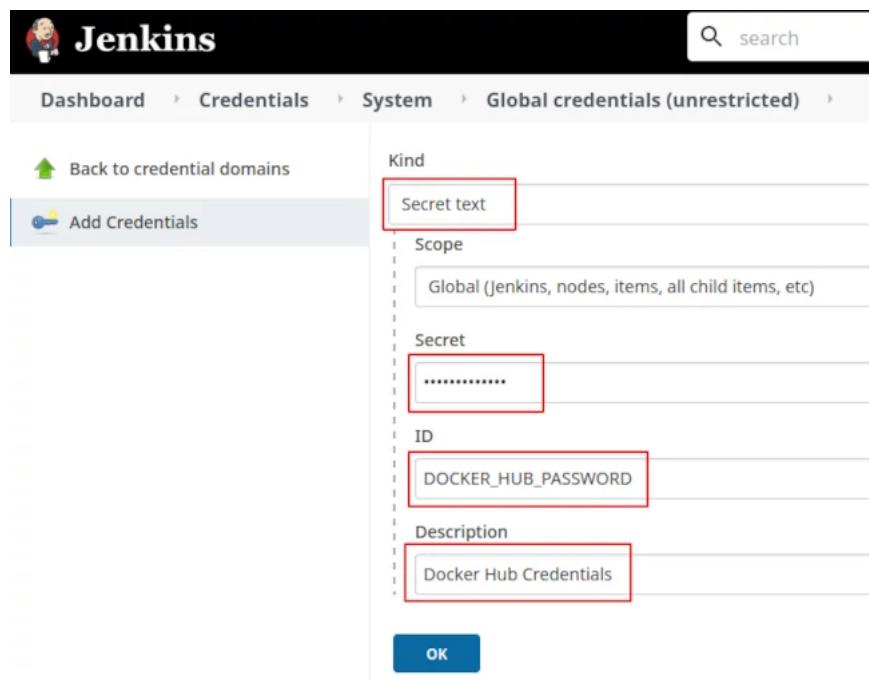
(In case if you haven't set up Docker Hub Account then please [create a DockerHub Account](#) because we are gonna need it.)

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

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: rahulwagh

Password: (Red box)

ID: GIT_HUB_CREDENTIALS

Description: Git Hub Credentials

OK

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

```

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'
}

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

```

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*

[Dashboard](#) >[People](#)[Project Relationship](#)

Jenkins new item for creating the pipeline

Pipeline Name: Now enter Jenkins pipeline name and select Pipeline

Enter an item name

jhooc-pipeline Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Copy from

OK

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.

General Build Triggers Advanced Project Options Pipeline

Pipeline

Definition

Pipeline script

```

1 node {
2   stage("Git Clone"){
3     git credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/rahulwagh/k8s-jenkins-aws'
4   }
5   stage('Gradle Build') {
6     sh './gradlew build'
7   }
8   stage("Docker build"){
9     sh 'docker version'
10    sh 'docker build -t jhooq-docker-demo .'
11    sh 'docker image list'
12    sh 'docker tag jhooq-docker-demo rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'
13  }
14}
15
16
17
18
19
20

```

Use Groovy Sandbox

[Pipeline Syntax](#)

[Save](#) [Apply](#)

Add the pipeline script into the pipeline definition script section

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

Jenkins

Dashboard > jhooq-pipeline

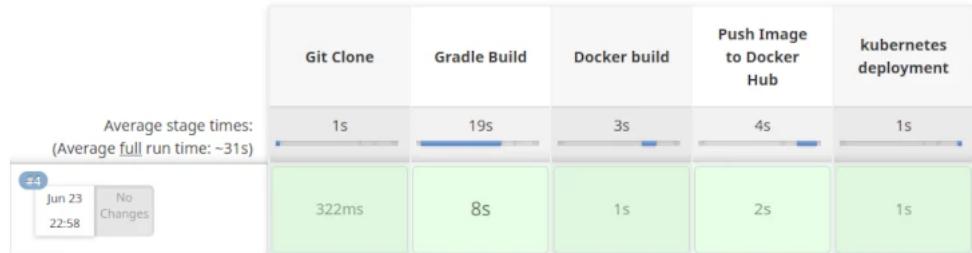
- Back to Dashboard
- Status
- Changes
- Build Now** (highlighted with a red box and arrow)
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Add the pipeline script into the pipeline definition script section

Verify the build status:



Stage View



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`

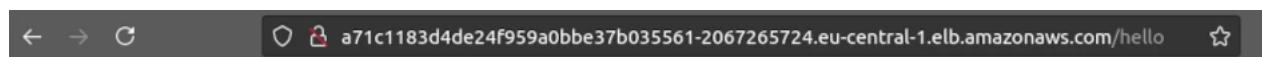
```
jenkins@ip-172-31-42-133:~$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
jhooc-springboot   2/3       3           2          33m
```

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
jhooc-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



Hello - Jhooc-k8s i Have updated the message

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 -

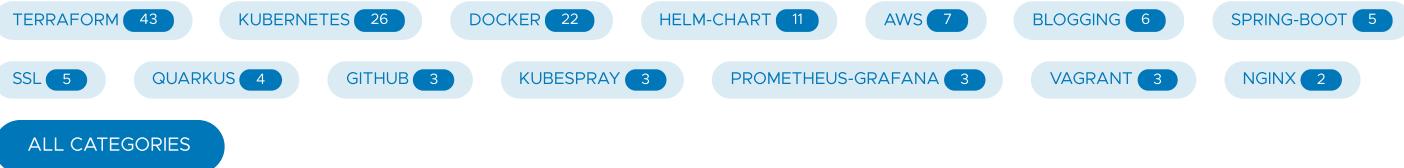
1. [Setup kubernetes on Ubuntu](#)
2. [Setup Kubernetes on CentOs](#)
3. [Setup HA Kubernetes Cluster with Kubespray](#)
4. [Setup HA Kubernetes with Minikube](#)
5. [Setup Kubernetes Dashboard for local kubernetes cluster](#)
6. [Setup Kubernetes Dashboard On GCP\(Google Cloud Platform\)](#)
7. [How 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



Series



Tags



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 command?
- 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.



Copyright 2023 JHOOQ. All Rights Reserved