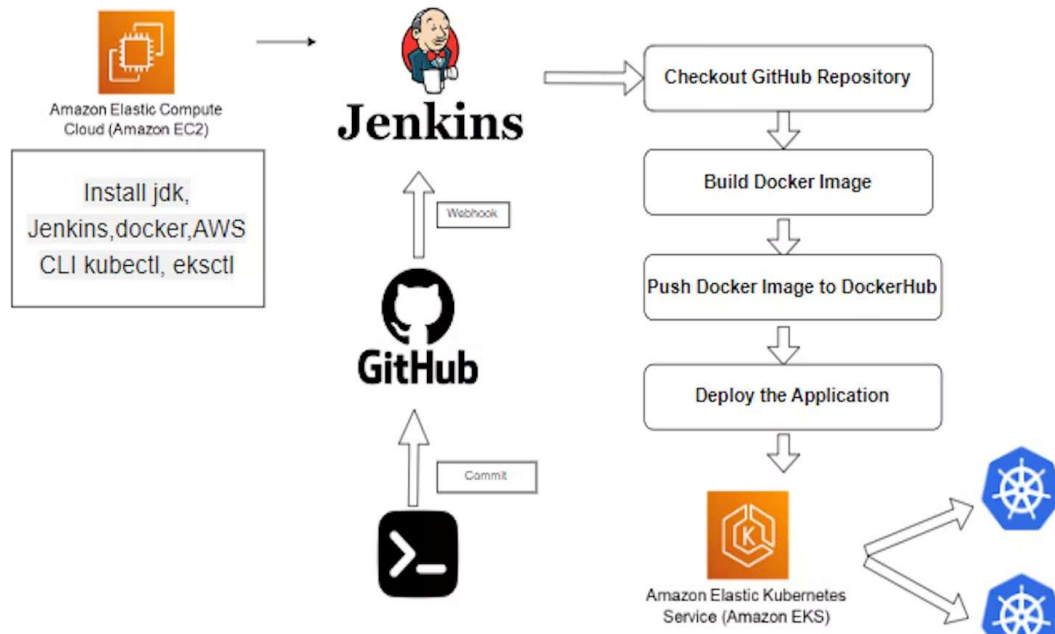


Jenkins CI/CD with Amazon EKS



Setup an AWS EC2 Instance

Login to an AWS account using a user with admin privileges and ensure your region is set to `us-east-1` N. Virginia.

Move to the EC2 console. Click `Launch Instance`.

For `name` use `Jenkins-EC2`.

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu


ubuntu

Windows

Microsoft

Red Hat

Red Hat



Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-0557a15b87f6559cf (64-bit (x86)) / ami-0f9bd9098aca2d42b (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-02-08

Architecture

AMI ID

64-bit (x86)

ami-0557a15b87f6559cf

Verified provider

Select **t2.medium** because we will be installing **Jenkins** and t2.micro will not be sufficient enough to set up Jenkins.

▼ Instance type

Info

Instance type

t2.medium

Family: t2 2 vCPU 4 GiB Memory

On-Demand Linux pricing: 0.0464 USD per Hour

On-Demand RHEL pricing: 0.1064 USD per Hour

On-Demand Windows pricing: 0.0644 USD per Hour

On-Demand SUSE pricing: 0.1464 USD per Hour

Compare instance types

Create and download the key pair(private key and public key)

Create key pair

×

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** [Learn more](#)

Key pair name

JenkinsEKS-KeyPair

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

☒ RSA
RSA encrypted private and public key pair

☐ ED25519
ED25519 encrypted private and public key pair (Not supported for Windows instances)

Private key file format

☒ .pem
For use with OpenSSH

Cancel


Create key pair

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

JenkinsEKS-KeyPair ▼

 [Create new key pair](#)

Configure Security Group - This is an important step because here we need to **add Custom TCP Port 8080**, if you do not add this port then you will not be able to access Jenkins using the public IP address of the AWS EC2 instance.

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

Security group name - *required*

Jenkin-EC2-SG

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&!;\$*

Description - *required* [Info](#)

Jenkin-EC2-SG

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Remove

Type [Info](#)

ssh

Protocol [Info](#)

TCP

Port range [Info](#)

22

Source type [Info](#)

Custom

Source [Info](#)

Q Add CIDR, prefix list or secur

0.0.0.0/0 X

Description - *optional* [Info](#)

e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 8080, 0.0.0.0/0)

Remove

Type [Info](#)

Custom TCP

Protocol [Info](#)

TCP

Port range [Info](#)

8080

Source type [Info](#)

Custom

Source [Info](#)

Q Add CIDR, prefix list or secur

0.0.0.0/0 X

Description - *optional* [Info](#)

e.g. SSH for admin desktop

Click on launch Instance and once EC2 Instance started, connect to it with EC2 Instance Connect.

EC2 > Instances > i-0bc62e080d380dd44

Instance summary for i-0bc62e080d380dd44 (Jenkins-EC2) [Info](#)

Updated less than a minute ago



Connect

Instance state ▼

Actions ▼

Instance ID

i-0bc62e080d380dd44 (Jenkins-EC2)

Public IPv4 address

54.234.207.117 | [open address](#)

Private IPv4 addresses

172.31.30.151

IPv6 address

—

Instance state

Running

Public IPv4 DNS

ec2-54-234-207-117.compute-1.amazonaws.com | [open address](#)

Connect to instance [Info](#)

Connect to your instance i-0bc62e080d380dd44 (Jenkins-EC2) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID

i-0bc62e080d380dd44 (Jenkins-EC2)

Public IP address

54.234.207.117

User name

Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ubuntu.

ubuntu

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

Cancel

Connect

← → ↺ us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standa

aws Services [Alt+S]

* Ubuntu Pro delivers the most comprehensive open source security and compliance features.

<https://ubuntu.com/aws/pro>

* Introducing Expanded Security Maintenance for Applications. Receive updates to over 25,000 software packages with your Ubuntu Pro subscription. Free for personal use.

<https://ubuntu.com/aws/pro>

Expanded Security Maintenance for Applications is not enabled.

updates can be applied immediately.

of these updates is a standard security update.

o see these additional updates run: apt list --upgradable

additional security updates can be applied with ESM Apps.

earn more about enabling ESM Apps service at <https://ubuntu.com/esm>

**** System restart required ****

ast login: Mon Feb 27 08:52:09 2023 from 18.206.107.27

buntu@ip-172-31-30-151:~\$

Install JDK on AWS EC2 Instance

Before following the below steps, assume you have already launched the ubuntu-based ec2 medium instance.

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

COPY

COPY

```
sudo apt-get update
```

```
java --version
```

```
ubuntu@ip-172-31-30-151:~$ sudo apt-get update
```

```
ubuntu@ip-172-31-30-151:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install default-jre          # version 2:1.11-72build2, or
sudo apt install openjdk-11-jre-headless # version 11.0.17+8-1ubuntu2~22.04
sudo apt install openjdk-17-jre-headless # version 17.0.5+8-2ubuntu1~22.04
sudo apt install openjdk-18-jre-headless # version 18.0.2+9-2~22.04
sudo apt install openjdk-19-jre-headless # version 19.0.1+10-1ubuntu1~22.04
sudo apt install openjdk-8-jre-headless  # version 8u352-ga-1~22.04
```

If this command indicates that Java is not found, then it's not installed and you can proceed with the next steps.

You can install java by running the following command.

COPY

COPY

```
apt install openjdk-11-jre-headless -y
```

```
java --version
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-30-151:~$ java --version
openjdk 11.0.17 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu222.04, mixed mode, sharing)
ubuntu@ip-172-31-30-151:~$
```

Install and Setup Jenkins

Step 1: Install Jenkins

Follow the steps for installing Jenkins on the EC2 instance.

COPY

COPY

```
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'
```

```
sudo apt-get update
```

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

```
sudo systemctl status jenkins
```



```

ubuntu@ip-172-31-30-151:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
ubuntu@ip-172-31-30-151:~$

ubuntu@ip-172-31-30-151:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo gpg --dearmor -o /usr/share/keyrings/jenkins.gpg
ubuntu@ip-172-31-30-151:~$ sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

ubuntu@ip-172-31-30-151:~$ sudo apt update
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease

ubuntu@ip-172-31-30-151:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

ubuntu@ip-172-31-30-151:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-02-23 12:23:13 UTC; 1min 13s ago
     Main PID: 4973 (java)
    Tasks: 43 (limit: 4689)
   Memory: 1.2G
      CPU: 45.099s

```

Step 2: Setup Jenkins

Now go to `AWS dashboard -> EC2 -> Instances(running)` and click on Jenkins-EC2

Copy Public IPv4 address.

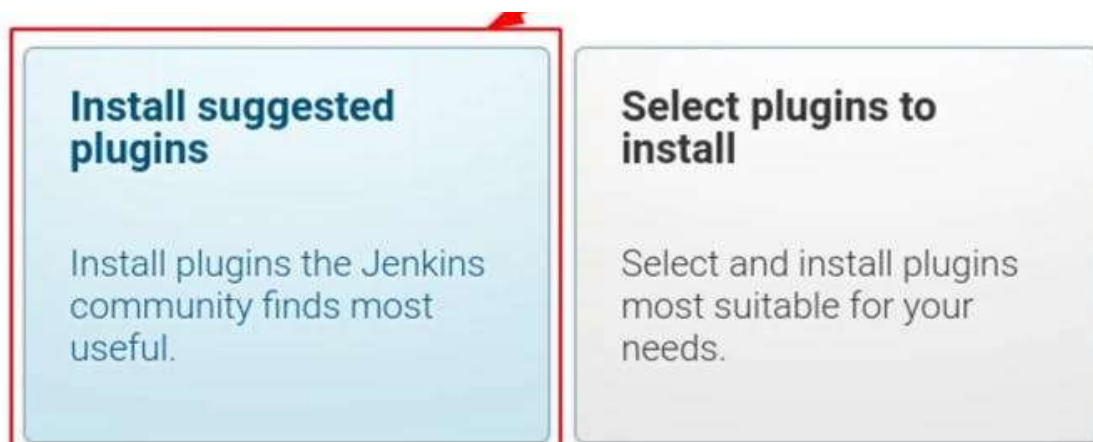
Alright now we know the public IP address of the EC2 machine, so now we can access Jenkins from the browser using the public IP address followed by port 8080.



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

```
ubuntu@ip-172-31-30-151:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
adc00600a863414bb9f01402695f55b9
ubuntu@ip-172-31-30-151:~$
```

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



Create First Admin User

Username :

admin

Password :

Confirm password :

Full name :

sunita

E-mail address :

sunita123@gmail.com

Jenkins 2.375.3

[Skip and continue as admin](#)

[Save and Continue](#)

Instance Configuration

Jenkins URL:

http://54.234.207.117:8080/

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.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

And now your Jenkins is ready for use

Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

Install Docker

The docker installation will be done by the **Jenkins** user because now it has root user privileges.

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.

COPY

COPY

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

```
docker --version
```

```
docker ps
```

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

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

```
jenkins@ip-172-31-30-151:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu4
jenkins@ip-172-31-30-151:~$
```

```
jenkins@ip-172-31-30-151:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
jenkins@ip-172-31-30-151:~$
```

Install and Setup AWS CLI

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**

COPY

COPY

```
sudo apt install awscli
```

```
aws --version
```

```
sudo apt install awscli
```

```
jenkins@ip-172-31-30-151:~$ aws --version
aws-cli/1.22.34 Python/3.10.6 Linux/5.15.0-1028-aws botocore/1.23.34
jenkins@ip-172-31-30-151:~$
```

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

COPY

COPY

```
aws configure
```

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

(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.

```
jenkins@ip-172-31-30-151:~$ aws configure
AWS Access Key ID [*****BLP4]: AKIAZ
AWS Secret Access Key [*****lVjL]: F
Default region name [None]: us-east-1
Default output format [None]:
jenkins@ip-172-31-30-151:~$
```

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

COPY
COPY

```
curl -LO [https://storage.googleapis.com/kubernetes-  
release/release/$(curl https://storage.googleapis.com/kubernetes-  
release/release/$(curl -s https://storage.googleapis.com/kubernetes-  
release/release/stable.txt)/bin/linux/amd64/kubect]l/bin/linux/amd64/kub  
ectl)
```

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

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

```
kubectl version
```

```
jenkins@ip-172-31-30-151:~$ curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl"
jenkins@ip-172-31-30-151:~$
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
100 45.7M  100 45.7M    0     0  43.3M      0  0:00:01  0:00:01 --:--:-- 43.4M
jenkins@ip-172-31-30-151:~$ chmod +x ./kubectl
jenkins@ip-172-31-30-151:~$ sudo mv ./kubectl /usr/local/bin
jenkins@ip-172-31-30-151:~$
```

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

```
Client Version: version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.1", GitCommit:"8f94681cd294aa9cfd3407b819f6c70214973a4", GitTreeState:"clean", BuildDate:"2023-01-18T19:58:16Z", GoVersion:"go1.19.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"24", GitVersion:"v1.24.10-eks-48643af", GitCommit:"9176fb99b52f8d5ff73d67f6a27f3a630f679f8a", GitTreeState:"clean", BuildDate:"2023-03-24T19:17:48Z", GoVersion:"go1.19.5", Compiler:"gc", Platform:"linux/amd64"}
WARNING: version difference between client (1.26) and server (1.24) exceeds the supported minor version skew of +/-1
```

Install and Setup eksctl

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

Download and extract the latest release of `eksctl` with the following command.

COPY

COPY

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

Move the extracted binary to `/usr/local/bin`.

COPY

COPY

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

Test that your installation was successful with the following command.

COPY

COPY

```
eksctl version
```

```
Client Version: version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.1", GitCommit:"0f94661cd294a8cfd3407b8191f6c70214973a4", GitTreeState:"clean", BuildDate:"2023-01-10T13:58:16Z", GoVersion:"go1.19.5", Compiler:"gc", Platform:"linux/amd64"}
Kubernetes Version: v1.26.1
Server Version: version.Info{Major:"1", Minor:"24", GitVersion:"v1.24.10-eks-48e43af", GitCommit:"9176f699b52f0d5ff73d67fca27f3a638f679f8a", GitTreeState:"clean", BuildDate:"2023-01-24T15:17:48Z", GoVersion:"go1.19.5", Compiler:"gc", Platform:"linux/amd64"}
WARNING: version difference between client (1.26) and server (1.24) exceeds the supported minor version skew of +/-1
```

Creating an Amazon EKS cluster using eksctl

Now in this step, we are going to create Amazon EKS cluster using `eksctl`

You need the following in order to run the eksctl command

1. **Name of the cluster :** `--name first-eks-cluster1`
2. **Version of Kubernetes :** `--version 1.24`
3. **Region :** `--region us-east-1`

4. **Nodegroup name/worker nodes** : `--nodegroup-name worker-nodes`
5. **Node Type** : `--nodegroup-type t2.micro`
6. **Number of nodes**: `--nodes 2`

Here is the eksctl command -

COPY

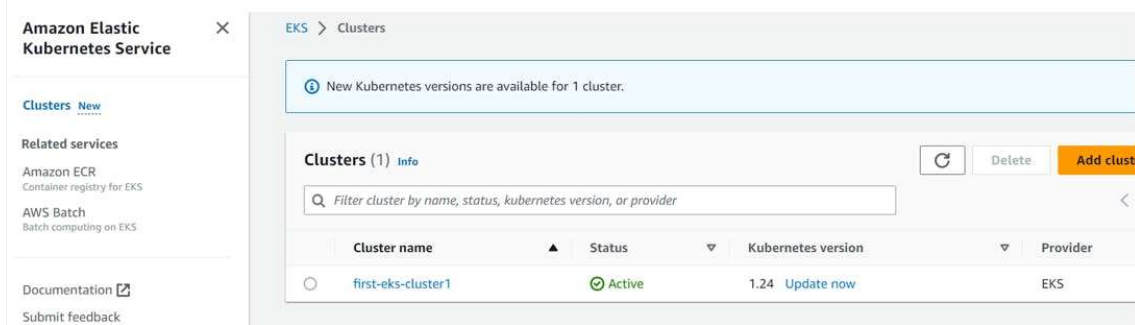
COPY

```
eksctl create cluster --name first-eks-cluster1 --version 1.24 --region us-east-1  
--nodegroup-name worker-nodes --node-type t2.micro --nodes 2
```

It took me 20 minutes to complete this EKS cluster. If you get any error for not having sufficient data for mentioned availability zone then try it again.

Verify the EKS kubernetes cluster on AWS Console.

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



Overview

Resources

Compute

Networking

Add-ons

Authentication

Logging

Update history

Tags

Resource types

▶ Workloads

▼ Cluster

Nodes

Namespaces

APIServices

Leases

RuntimeClasses

FlowSchemas

PriorityLevelConfigurations

Cluster: Nodes (2)

View details

A node is a worker machine in Kubernetes. [Learn more](#)

Filter Nodes by property or value

< 1 >


	Node name	Instance type	Node group	Created	Status
<input type="radio"/>	ip-192-168-48-227.ec2.internal	t2.micro	worker-nodes	Created 3 hours ago	✓ Read y
<input type="radio"/>	ip-192-168-5-98.ec2.internal	t2.micro	worker-nodes	Created 3 hours ago	✓ Read y

Add Docker and GitHub Credentials on Jenkins

Step 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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Secret text

Scope ?


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

Secret

.....

ID ?

DOCKER_HUB_PASSWORD

 This ID is already in use

Description ?

DOCKER_HUB_CREDENTIAL

Create

Step 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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Username with password

Scope ?

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

Username ?


sunitabachhav2007

☐ Treat username as secret ?

Password ?

ID ?

GIT_HUB_CREDENTIALS

 This ID is already in use





Description ?

GITHUBCREDENTIALS

Create

Global credentials (unrestricted) [+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 DOCKER_HUB_PASSWORD	Docker Hub Credentials	Secret text	Docker Hub Credentials 
 GIT_HUB_CREDENTIALS	sunitabachhav2007/***** (Git Hub Credentials)	Username with password	Git Hub Credentials 

Build, deploy and test CI/CD pipeline

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

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



+ New Item

Pipeline Name: Now enter Jenkins pipeline name and select Pipeline

Enter an item name

test-pipeline

» A job already exists with the name 'test-pipeline'



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

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

Dashboard > pipeline-node-todo-eks > Configuration

Configure

General

Advanced Project Options

Pipeline

Advanced Project Options

Advanced...

Pipeline

Definition

Pipeline script

Script ?

```
1= node {
2=
3=   stage("Git Clone"){
4=     |
5=     | git credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/sunitabachhav2007/node-todo-cicd.git', branch: 'master'
6=     |
7=   }
8=   stage("Build") {
9=     |
10=    | sh 'docker build -t sunitabachhav2007/node-todo-test:latest'
11=    | sh 'docker image list'
12=    |
13=   }
14=   withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD', variable: 'PASSWORD')]) {
15=     | sh 'docker login -u sunitabachhav2007 -p $PASSWORD'
16=   }
17= }
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save

Apply

Copy the below script and paste into Pipeline Script.

COPY

COPY

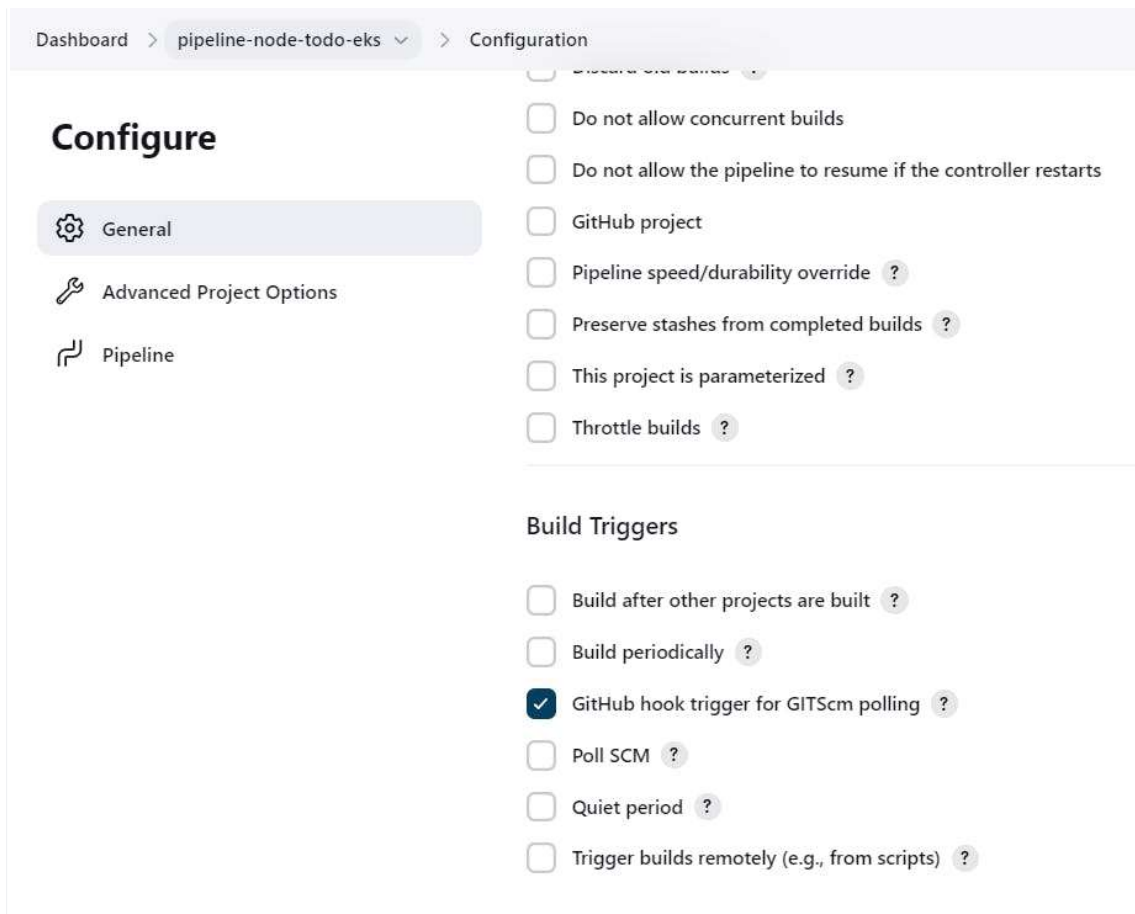
```
node {  
  
    stage("Git Clone"){  
  
        git credentialsId: 'GIT_HUB_CREDENTIALS', url:  
        'https://github.com/sunitabachhav2007/node-todo-cicd.git', branch: 'master'  
  
    }  
  
    stage("Build") {  
  
        sh 'docker build . -t sunitabachhav2007/node-todo-test:latest'  
  
        sh 'docker image list'  
  
    }  
  
    withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD',  
variable: 'PASSWORD')]) {  
  
        sh 'docker login -u sunitabachhav2007 -p $PASSWORD'
```

```
}  
  
stage("Push Image to Docker Hub"){  
  
    sh 'docker push sunitabachhav2007/node-todo-test:latest'  
  
}  
  
stage("kubernetes deployment"){  
  
    sh 'kubectl apply -f deployment.yml'  
  
}  
  
}
```

To set up Jenkins - GitHub Webhook

Now, go to the "Build Triggers" tab.

Here, choose the "GitHub hook trigger for GITScm pulling" option, which will listen for triggers from the given GitHub repository, as shown in the image below.



Jenkins GitHub Webhook is **used to trigger the action whenever Developers commit something into the repository**. It can automatically build and deploy applications.

Switch to your GitHub account, go to "**Settings**" option. Here, select the "**Webhooks**" option and then click on the "**Add Webhook**"

It will provide you the blank fields to add the Payload URL where you will paste your Jenkins address, Content type, and other configuration.

Go to your Jenkins tab and copy the URL then paste it in the text field named "**Payload URL**", as shown in the image below. Append the "**/github-webhook/**" at the end of the URL.

<> Code

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub apps

Email notifications

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our [developer documentation](#).

Payload URL *
http://54.234.207.117:8080/github-webhook/

Content type
application/json

Secret

Which events would you like to trigger this webhook?
☒ Just the push event.
☐ Send me everything.
☐ Let me select individual events.

☒ Active
We will deliver event details when this hook is triggered.

Add webhook

Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at <https://docs.github.com/webhooks/#ping-event>.

sunitabachhav2007 / node-todo-cicd Public

forked from LondheShubham153/node-todo-cicd

<> Code Pull requests Actions Projects Wiki Security Insights Settings

General

Webhooks

Add webhook

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

☒ http://54.234.207.117:8080/github-... (push) Edit Delete

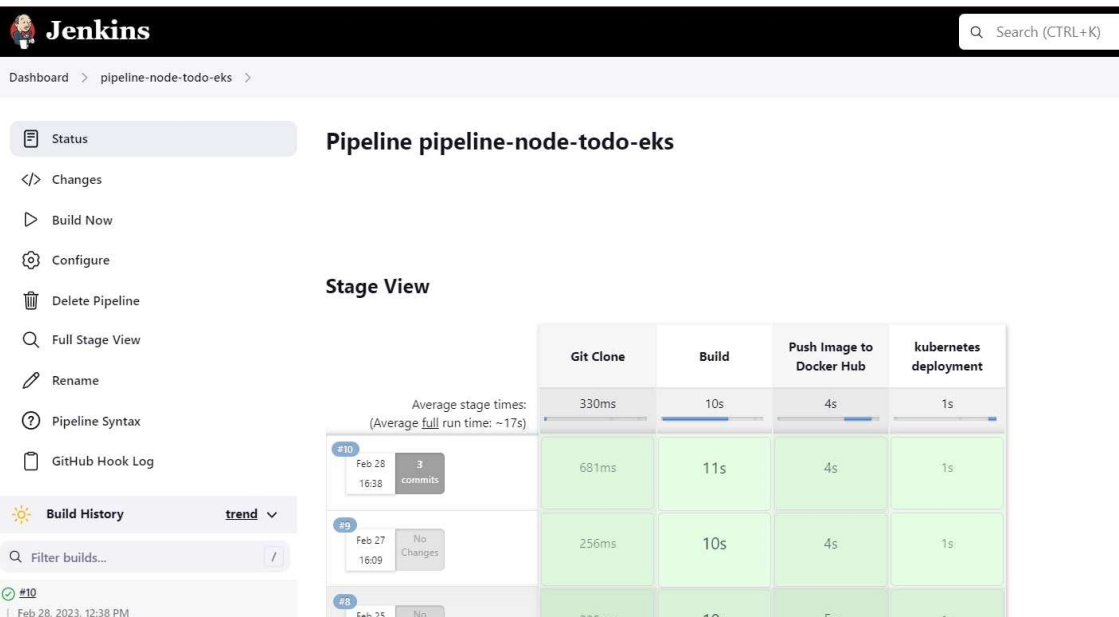
You completed Jenkins GitHub Webhook. Now for any commit in the GitHub repository, Jenkins will trigger the event specified

```
views > <> todo.ejs > html > head > title
1 <!DOCTYPE html>
2
3 <html>
4
5   <head>
6     <title>My Awesome todoclist. Created today on 28th Feb 2023</title>
7     <style>
8       a {
9         text-decoration: none;
10        color: black;
11      }
12    </style>
13  </head>
14
15
```

```
PS C:\Sunita\DevOps-Project\CICD-EKS-node-todo-app\node-todo-cicd> git remote -v
origin https://github.com/sunitabachhav2007/node-todo-cicd.git (fetch)
origin https://github.com/sunitabachhav2007/node-todo-cicd.git (push)
PS C:\Sunita\DevOps-Project\CICD-EKS-node-todo-app\node-todo-cicd>
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + v
PS C:\Sunita\DevOps-Project\CICD-EKS-node-todo-app\node-todo-cicd> git commit -m "Changed message with today's date"
[master d8562a3] Changed message with today's date
1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Sunita\DevOps-Project\CICD-EKS-node-todo-app\node-todo-cicd> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 387 bytes | 387.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/sunitabachhav2007/node-todo-cicd.git
19461fb..d8562a3 master -> master
PS C:\Sunita\DevOps-Project\CICD-EKS-node-todo-app\node-todo-cicd>
```

After pushing code to Github repository



The Jenkins dashboard shows the pipeline **pipeline-node-todo-eks** in the **Stage View**. The pipeline consists of four stages: **Git Clone**, **Build**, **Push Image to Docker Hub**, and **kubernetes deployment**. The average stage times are 330ms, 10s, 4s, and 1s respectively. The average full run time is approximately 17s.

Stage	Git Clone	Build	Push Image to Docker Hub	kubernetes deployment
#10 (Feb 28, 16:38, 3 commits)	681ms	11s	4s	1s
#9 (Feb 27, 16:09, No Changes)	256ms	10s	4s	1s
#8 (Feb 25, No Changes)	205ms	10s	5s	1s

```
jenkins@ip-172-31-30-151:~$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
node-todo-app-8ff465f74-b887c      1/1     Running   0           3m13s
node-todo-app-8ff465f74-tf4v6      1/1     Running   0           3m13s
jenkins@ip-172-31-30-151:~$ kubectl get service
NAME    TYPE    CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes  ClusterIP  10.100.0.1     <none>         443/TCP    2d22h
node-todo-app  LoadBalancer  10.100.0.123  aa33fia9ff964a67bbbc5cac9ff56-1844308549.us-east-1.elb.amazonaws.com  80:32002/TCP  3m23s
jenkins@ip-172-31-30-151:~$ kubectl get deployments
NAME    READY    UP-TO-DATE    AVAILABLE    AGE
node-todo-app  2/2     2             2           16m
jenkins@ip-172-31-30-151:~$
```

You can access the rest endpoint from a browser using the EXTERNAL-IP address.

Finally uploaded node-todo-app on AWS EKS - Elastic Kubernetes Service

Today date is 28th Feb 2023. Hurray !!!

What should I do?

Clean up

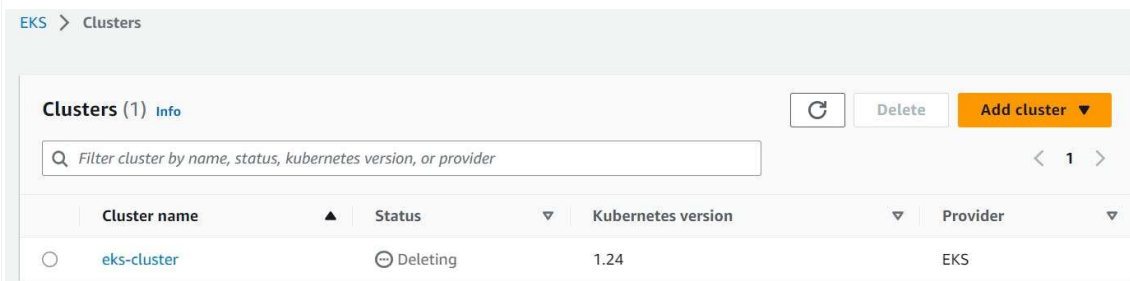
Copy Deployment.yml file (From Github Repository) to EC2 server and run with below command.

COPY

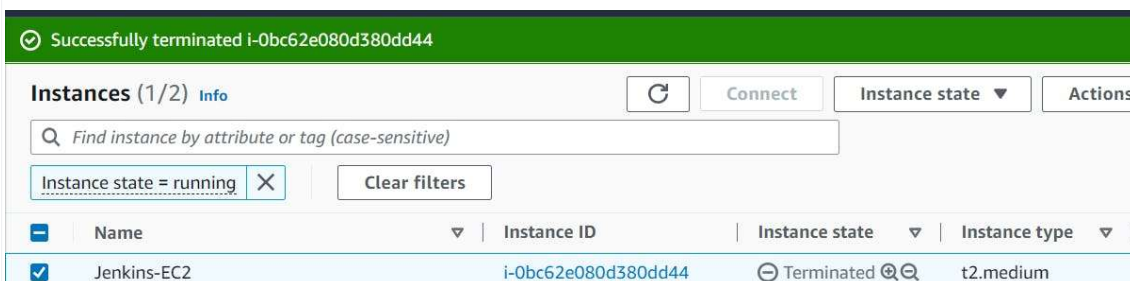
COPY

```
kubectl delete -f deployment.yml
```

Delete EKS Cluster from AWS Console.



Terminate EC2 Instance.



Conclusion

We have successfully **deployed our Node.js App in Amazon EKS cluster using AWS EC2, Jenkins, Docker, Kubernetes, GitHub, Webhook.**