

Jenkins Pipeline 3 - CICD with Terraform

CI -> Continuous Integration

CD -> Continuous Deployment or Delivery

CI Job:

Multiple stages will be there

Pipeline1: Git Clone -> Maven Build -> Docker Image -> Push Image -> Deployment

CD Job:

Pipeline2: Git Clone (to clone K8s manifest file and if we are writing manifest file in pipeline then need not clone) -> K8s Deployment

If Pipeline1 is successful, it should automatically trigger the CD job

Starting EKS clusters in EKS-host VM


```
eksctl create cluster --name my-eks-cluster --region ca-central-1 --node-type t2.medium --zones ca-central-1a,ca-central-1b
```

New Item


New Item

Enter an item name


Select an item type




Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



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

This repo has both manifest and Dockerfile

```
pipeline {
  agent any

  environment {
    IMAGE_NAME = "CI-web-app"
    DOCKER_TAG = "latest"
  }

  tools {
    maven "maven-3.9.10"
  }
}
```

```

stages {
  stage('git clone') {
    steps {
      git branch: 'main', url: 'https://github.com/Haider7214/WebAppMaven.git'
    }
  }
  stage('maven build') {
    steps {
      sh 'mvn clean package'
    }
  }
  stage('Build Docker Image') {
    steps {
      script {
        writeFile file: 'Dockerfile', text: '''
          # Use an official Tomcat base image
          FROM tomcat:latest
          LABEL maintainer="DemoDockerfile"

          # Remove default webapps
          RUN rm -rf /usr/local/tomcat/webapps/*

          # Copy WAR to Tomcat webapps
          COPY target/*.war /usr/local/tomcat/webapps/ROOT.war

          # Expose port
          EXPOSE 8080
        '''
        echo "✔ Dockerfile generated"

        sh "docker build -t ${IMAGE_NAME}:${DOCKER_TAG} ."
      }
    }
  }
  stage('Docker push') {
    steps {
      withCredentials([string(credentialsId: 'Sai-Docker-Pwd', variable: 'Docker_Hub_PWD_New')]) {
        sh 'docker login -u saidocker567 -p ${Docker_Hub_PWD_New}'
        sh 'docker tag ${IMAGE_NAME}:${DOCKER_TAG} saidocker567/${IMAGE_NAME}:${DOCKER_TAG}'
        sh 'docker push saidocker567/${IMAGE_NAME}:${DOCKER_TAG}'
      }
    }
  }
}

```

```

9574addb8357: Layer already exists
37d26a060906: Layer already exists
4d8cb8462bc9: Layer already exists
78635f3af26b: Layer already exists
45a01f98e78c: Layer already exists
2c4ffe76ce80: Pushed
latest: digest: sha256:c81ddbd8b41fe7a6656c426df9f19e765148eb8d5e97b41add93a9549ba8ba4d size: 2619
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Change the image name

```
pipeline {
```

```
agent any
```

```
environment {
```

```
    IMAGE_NAME = "ci-web-app"
```

```
    DOCKER_TAG = "latest"
```

```
}
```

```
tools {
```

```
    maven "maven-3.9.10"
```

```
}
```

```
stages {
```

```
    stage('git clone') {
```

```
        steps {
```

```
            git branch: 'main', url: 'https://github.com/Haider7214/WebAppMaven.git'
```

```
        }
```

```
    }
```

```
    stage('maven build') {
```

```
        steps {
```

```
            sh 'mvn clean package'
```

```
        }
```

```
    }
```

```
    stage('Build Docker Image') {
```

```
        steps {
```

```
            script {
```

```
                writeFile file: 'Dockerfile', text: '''
```

```
                # Use an official Tomcat base image
```

```
                FROM tomcat:latest
```

```
                LABEL maintainer="DemoDockerfile"
```

```
                # Remove default webapps
```

```
                RUN rm -rf /usr/local/tomcat/webapps/*
```

```

# Copy WAR to Tomcat webapps
COPY target/*.war /usr/local/tomcat/webapps/ROOT.war





# Expose port
EXPOSE 8080

'''
echo "✔ Dockerfile generated"

    sh "docker build -t ${IMAGE_NAME}:${DOCKER_TAG} ."
}
}
}

stage('Docker push') {
  steps {
    withCredentials([string(credentialsId: 'Sai-Docker-Pwd', variable: 'Docker_Hub_PWD_New')])
    {
      sh 'docker login -u saidocker567 -p ${Docker_Hub_PWD_New}'
      sh 'docker tag ${IMAGE_NAME}:${DOCKER_TAG}
saidocker567/${IMAGE_NAME}:${DOCKER_TAG}'
      sh 'docker push saidocker567/${IMAGE_NAME}:${DOCKER_TAG}'
    }
  }
}
}
}

```


Repositories		
All repositories within the saidocker567 namespace.		
<input type="text" value="Search by repository name"/> <input type="button" value="All content"/>		
Name	Last Pushed 	Contains
saidocker567/ci-web-app	less than a minute ago	
saidocker567/my-web-app	about 1 hour ago	
saidocker567/img-1	2 months ago	


1-3 of 3


To trigger CD, we need Pipeline Stage


Dashboard > Manage Jenkins > Plugins




Plugins

 Updates 21

 Available plugins

 Installed plugins

 Advanced settings

Install	Name ↓
<input type="checkbox"/>	Pipeline: Stage View 2.38  User Interface Pipeline Stage View Plugin.
<input type="checkbox"/>	Artifact Manager on S3 942.vc32d61381d05  aws A Jenkins plugin to keep artifacts and Pipeline stashes in Amazon S3.
<input type="checkbox"/>	Azure Artifact Manager 187.vc45952e6c7fc  azure Jenkins Azure Artifact Manager plugin enables keeping artifacts and Pipeline stashes in Azure Blob storage

Different view due to Pipeline Stage View Plugin

CI_WebApp_Pipeline

CI Job with Git Maven Docker DockerHub

Stage View



CD as another pipeline

git clone (to clone k8s manifest file and if we are writing manifest file in pipeline then need not to clone)

k8s deployment

```
pipeline {
  agent any
  tools {
    maven "maven-3.9.10"
  }

  stages {
    stage('git clone') {
      steps {
        git branch: 'main', url: 'https://github.com/Haider7214/WebAppMaven.git'
      }
    }
    stage('k8s - deployment') {
      steps {
        sh 'kubectl apply -f k8s-deployment.yaml'
      }
    }
  }
}
```

35:44

New Item

Enter an item name

CD_WebApp_Pipeline

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



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.

We see k8s-deployment.yaml and Dockerfile are there in the repo

[Actions](#) [Projects](#) [Security](#) [Insights](#)



WebAppMaven

Public

Watch 0

main



1 Branch



0 Tags

Go to file



Add file

Code



Haider7214 Update Dockerfile

8d7b5d4 · 19 hours ago

7 Commits



.mvn/wrapper

first commit

2 days ago



src

first commit

2 days ago



.gitattributes

first commit

2 days ago



.gitignore

first commit

2 days ago



Dockerfile

Update Dockerfile

19 hours ago



k8s-deployment.yaml

Create k8s-deployment.yaml

2 days ago



mvnw

first commit

2 days ago



mvnw.cmd

first commit

2 days ago



pom.xml

first commit

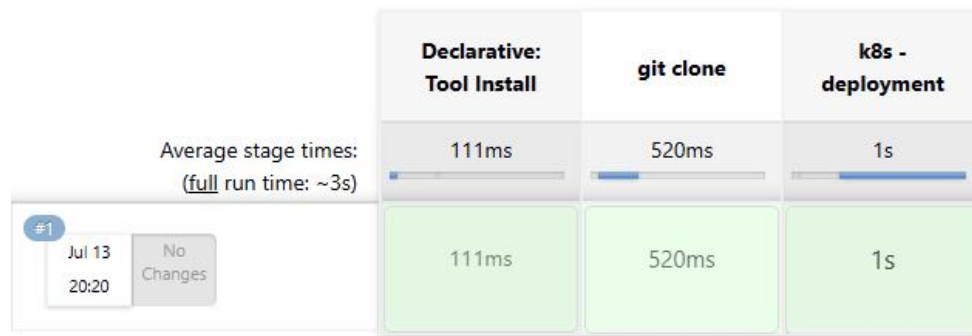
2 days ago

CD Pipeline was successful

CD_WebApp_Pipeline

CD deployment pipeline

Stage View



Permalinks

Once the CI pipeline is successful, it needs to trigger CD pipeline automatically. How do we do it?

Modifying CI Pipeline Script

CI pipeline Pipeline Syntax

Definition

Pipeline script

```
Script ?
1 pipeline {
2   agent any
3
4   environment {
5     IMAGE_NAME = "ci-web-app"
6     DOCKER_TAG = "latest"
7   }
8
9   tools {
10    maven "maven-3.9.10"
11  }
12
13  stages {
14    stage('git clone') {
15      steps {
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Build Build a Job

Select : CD_WebApp_Pipeline

Copy Script: build 'CD_WebApp_Pipeline'

```
pipeline {
  agent any
```

```
environment {
```

```

        IMAGE_NAME = "ci-web-app"
        DOCKER_TAG = "latest"
    }

    tools {
        maven "maven-3.9.10"
    }

    stages {
        stage('git clone') {
            steps {
                git branch: 'main', url: 'https://github.com/Haider7214/WebAppMaven.git'
            }
        }
        stage('maven build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    writeFile file: 'Dockerfile', text: '''
                        # Use an official Tomcat base image
                        FROM tomcat:latest
                        LABEL maintainer="DemoDockerfile"

                        # Remove default webapps
                        RUN rm -rf /usr/local/tomcat/webapps/*

                        # Copy WAR to Tomcat webapps
                        COPY target/*.war /usr/local/tomcat/webapps/ROOT.war

                        # Expose port
                        EXPOSE 8080
                    '''
                    echo "✔ Dockerfile generated"

                    sh "docker build -t ${IMAGE_NAME}:${DOCKER_TAG} ."
                }
            }
        }
        stage('Docker push') {
            steps {
                withCredentials([string(credentialsId: 'Sai-Docker-Pwd', variable: 'Docker_Hub_PWD_New')]) {
                    sh 'docker login -u saidocker567 -p ${Docker_Hub_PWD_New}'
                    sh 'docker tag ${IMAGE_NAME}:${DOCKER_TAG}'
                    sh 'docker push saidocker567/${IMAGE_NAME}:${DOCKER_TAG}'
                }
            }
        }
        stage('Trigger CD job if CI is successful') {
            steps {

```



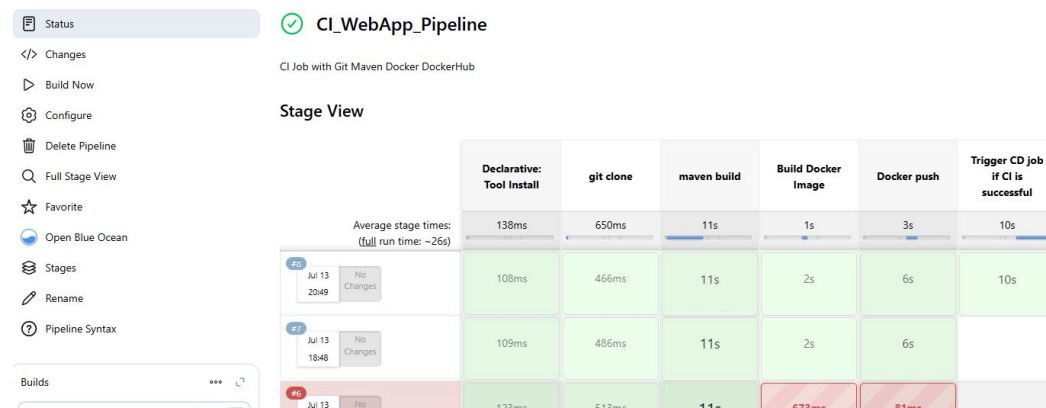
```

    build 'CD_WebApp_Pipeline'
  }
}
}
}

```

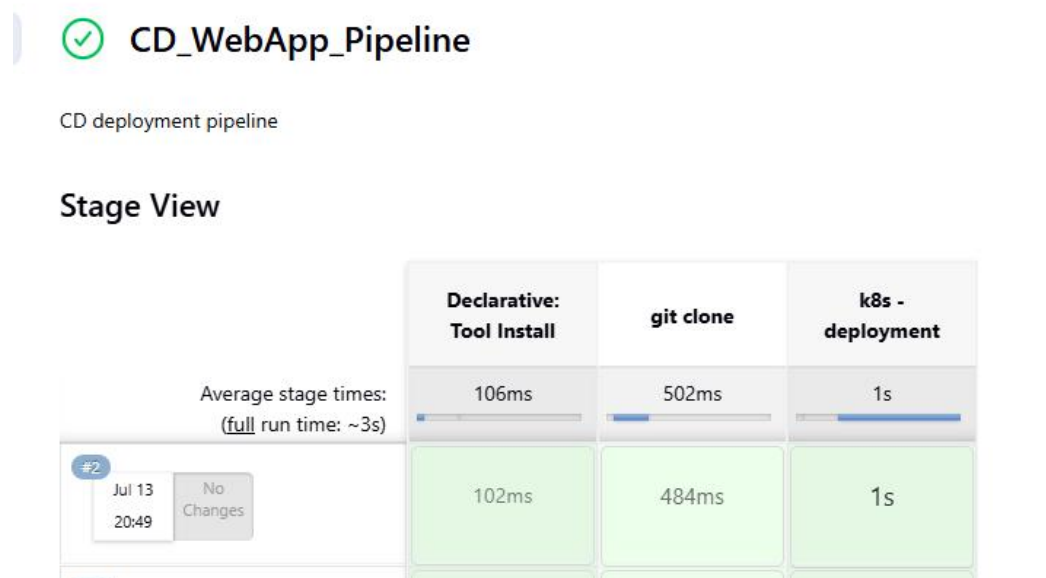
Apply and Save

Now it triggers CD job successfully



1:11

CD pipeline automatically getting executed



Go to Configure --> Poll SCM --> ***** means every minute we have a new commit it will trigger automatically

☐ GitHub hook trigger for GITScm polling ?

☒ Poll SCM ?

Schedule ?

No schedules so will only run due to SCM changes if triggered by a post-commit hook

☐ Ignore post-commit hooks ?

☐ Trigger builds remotely (e.g., from scripts) ?

Usually people use Terraform to provision the infrastructure
 Jenkins will execute Terraform script to provision infrastructure
 1:16

Run on EKS host
 kubectl delete all --all
 eksctl delete cluster --name my-eks-cluster --region ca-central-1

Make sure cluster is created on EKS-Host
 Create EKS Cluster using eksctl

Create K8s cluster in **EKS-host VM**
 eksctl create cluster --name my-eks-cluster --region ca-central-1 --node-type t2.medium --zones ca-central-1a,ca-central-1b

From EKS-host VM
 ubuntu@ip-172-31-9-165:~\$ cat .kube/config
 apiVersion: v1
 clusters:
 - cluster:
 certificate-authority-data:
 LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJUKlsSGRPcno3SjR3RFFZ
 SktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmIaWEp1WlhSbGN6QWVGdzB5TIRB
 M01UVXdNREEzTXpKYUZ3MHpOVEEzTVRNd01ERXINekphTUJVeApFekFSQmdOVkIBTVRDbXQxWW1
 WeWJtVjBaWE13Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUUN2OXNy
 TWpOZ3c3bTM3amFCZlZldGNLMXhBa1hBUHZvVmZlVzdrdzFSaU45SEQRtjd4dStRQzZocjgKdkN5ZWsv
 WC9vVmxEWEJmenRxUXVhYU1pOERSYkhLamI2NUF4QWwhXaE5rdUw4elpyQ0EyWGRLaHJpNjZMVA0
 awpmdUNVnNkJEVnAvcGozaDIrem91YXg5UHKvTituUWZnK3Y5QzN6WCtsOGI2TEIzTzRQNGJ0MncrVDZ
 GK2tOTURQCKRmZUxsSHIUWitCK1FHcloybk9VRFFTWENrN0tuSForY3RGZ2F6UW03aEtXSVDZnNGT2
 RibmVEMXdZMnB4bmgKeUxaTVVnUmoySVIvbDVZUUtXOEJDDkKjY0h4QmROSkFvekV0YUdnSWp6S2p
 ZTC9BckZDbExQTnVxU3VvTHgvQQpzSDZMb3lxcGtEUDNOeW5RVUINK3VuODRJK2EvQWdNQkFBR2pX
 VEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCKJnTIZlUk1CQWY4RUJUURBUUgVtUIlwR0ExVWREZ
 1FXQkJSYIZ4SnF6ak03dG5vUkZpV0VvUzVtdGtMZ1FqQVYKQmdOVkhSRUVEakFNZ2dwcmlRSmxjbTVs
 ZEdWek1BMEdDU3FHU0liM0RRRUJDD1VBQTRJQkFRQTFaRkVabjhLQgpia3VxMmFKNHl6UDFCNm13U
 mNVclVdi90OWtTNm1YRWdPd1ZXZGlaeWR4RC9NWTdKNlFjOHVKamlyMmQrL3B1CmVqZG5GaFFFY
 VZHOFpOMmo4K2E0eGRNSlJSjJNNkpzdEZWVXVqYUdYd1MyU2VFYkFtYXZKWXcvUHcwNTZ2T1cKWx
 BValc1Y2pUMTV3dDR3Zk9qdTN4ajh3NEtOMUIRtmxtYXVebjZCWWR3WmlMWW5WV0RqMit6U3VaV
 Hk1RTE4LwpUQU1FMzFPd2IzeTZvd3AzMWO4NXBHektXaU1mYkFpRnJYOXdjF5NnQySxhCUiFaM1J
 zTS9uTVBNvk1LYktHCjB5OFhaQTJodTnMl2o2RTRFMETndXR4VnFUUTkzVkrXN0E3L3ZGTGUzcgloY2cz
 VklZ0NMQnNWTmRSN1Z1TFQKekhuV2hYWWtqQW5QCj0tLS0tRU5EIENFUlRJRkdDQVRFLS0tLS0K
 server: https://E359CB3780B483E3DEE92E09CDE915C8.gr7.ca-central-1.eks.amazonaws.com
 name: my-eks-cluster.ca-central-1.eksctl.io
 contexts:

```
- context:
  cluster: my-eks-cluster.ca-central-1.eksctl.io
  user: i-01289fc5ca918b25f@my-eks-cluster.ca-central-1.eksctl.io
  name: i-01289fc5ca918b25f@my-eks-cluster.ca-central-1.eksctl.io
current-context: i-01289fc5ca918b25f@my-eks-cluster.ca-central-1.eksctl.io
kind: Config
preferences: {}
users:
- name: i-01289fc5ca918b25f@my-eks-cluster.ca-central-1.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
        - eks
        - get-token
        - --output
        - json
        - --cluster-name
        - my-eks-cluster
        - --region
        - ca-central-1
      command: aws
    env:
      - name: AWS_STS_REGIONAL_ENDPOINTS
        value: regional
    provideClusterInfo: false
```

Update in Jenkins Server

```
ubuntu@ip-172-31-11-116:~$ sudo vi .kube/config
```

<input type="checkbox"/>	my-eks-cluster...	i-07a5567a005072046	Running	t2.medium	Initializing	View ala
<input type="checkbox"/>	EKS-host	i-01289fc5ca918b25f	Running	t2.micro	2/2 checks passec	View ala
<input type="checkbox"/>	Jenkins-server	i-0abe9191466188c59	Running	t2.medium	2/2 checks passec	View ala
<input type="checkbox"/>	my-eks-cluster...	i-0f57f9b66fcc6964f	Running	t2.medium	Initializing	View ala

Select an instance

```
ubuntu@ip-172-31-11-116:~$ sudo vi .kube/config
```

```
ubuntu@ip-172-31-11-116:~$ kubectl get nodes
```

```
NAME                                STATUS ROLES AGE  VERSION
ip-192-168-25-28.ca-central-1.compute.internal Ready <none> 25m v1.32.3-eks-473151a
ip-192-168-61-42.ca-central-1.compute.internal Ready <none> 24m v1.32.3-eks-473151a
```

EKS-VM deleting cluster

```
kubectl delete all --all
```

```
eksctl delete cluster --name my-eks-cluster --region ca-central-1
```

We don't need these machines anymore so stopped them

<input type="checkbox"/>	JenkinsSlave	i-01f4ee18204bd61fd	⏻ Stopped	🔍	t2.micro	–	View
<input type="checkbox"/>	sonar-server	i-03605f14a60c45415	⏻ Stopped	🔍	t2.medium	–	View
<input type="checkbox"/>	my-eks-cluster...	i-07a5567a005072046	⏻ Terminated	🔍	t2.medium	–	View
<input checked="" type="checkbox"/>	EKS-host	i-01289fc5ca918b25f	⏻ Stopping	🔍	t2.micro	✅ 2/2 checks passed	View
<input checked="" type="checkbox"/>	Jenkins-server	i-0abe9191466188c59	⏻ Stopping	🔍	t2.medium	✅ 2/2 checks passed	View
<input type="checkbox"/>	my-eks-cluster...	i-0f57f9b66fcc6964f	⏻ Terminated	🔍	t2.medium	–	View

2 instances selected

In the next Pipeline, we are going to automate Jenkins with Terraform to provision EKS cluster
Terraform --> Jenkins --> EKS cluster

1. Have a machine to launch an EC2 instance (Ubuntu --> t2.medium) so we can have Terraform in it
2. Install Terraform, Java and Jenkins in this machine
- 3.

Name and tags
Info

Name
TerraformFinal
Add additional tags

Application and OS Images (Amazon Machine Image)
Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents
Quick Start

Amazon Linux
aws

macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

Red Hat
Red Hat

SUSE Linux
SUSE

Debian
debian

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-0c0a551d0459e9d39 (64-bit (x86)) / ami-0a47b03c99d29f7c2 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Free tier eligible

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture

AMI ID

Publish Date

Username

64-bit (x86)

ami-0c0a551d0459e9d39

2025-06-10

ubuntu

Verified provider

▼ Instance type

Info | Get advice

Instance type

t2.medium

Family: t2 2 vCPU 4 GiB Memory Current generation: true

On-Demand Ubuntu Pro base pricing: 0.0547 USD per Hour On-Demand RHEL base pricing: 0.08 USD per Hour

On-Demand SUSE base pricing: 0.1512 USD per Hour On-Demand Windows base pricing: 0.0692 USD per Hour

On-Demand Linux base pricing: 0.0512 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ 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

DevOpsMar30

Create new key pair

▼ Network settings

Info

Edit

Network

Info

vpc-0a752647f0a021f2e | default_VPC

Subnet

Info

No preference (Default subnet in any availability zone)

Auto-assign public IP

Info

Enable

Additional charges apply when outside of free tier allowance

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

Common security groups

Info

Select security groups

DevOps-sg sg-031a081efd38c0e3a

VPC: vpc-0a752647f0a021f2e

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Configure storage

Info

Advanced

1x 8 GiB gp3

Root volume, 3000 IOPS, Not encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Launch Instance

Instances (1/9) [Info](#)

Find Instance by attribute or tag (case-sensitive) All states ▾

<input type="checkbox"/>	Name ↗	Instance ID	Instance state ↗	Instance type ↗	Status check	Alarm s
<input type="checkbox"/>	sonar-server	i-03605f14a60c45415	⏸ Stopped 🔍 🔍	t2.medium	–	View al
<input type="checkbox"/>	my-eks-cluster...	i-07a5567a005072046	⏹ Terminated 🔍 🔍	t2.medium	–	View al
<input type="checkbox"/>	EKS-host	i-01289fc5ca918b25f	⏸ Stopped 🔍 🔍	t2.micro	–	View al
<input type="checkbox"/>	Jenkins-server	i-0abe9191466188c59	⏸ Stopped 🔍 🔍	t2.medium	–	View al
<input type="checkbox"/>	my-eks-cluster...	i-0f57f9b66fcc6964f	⏹ Terminated 🔍 🔍	t2.medium	–	View al
<input checked="" type="checkbox"/>	TerraformFinal	i-03c973945e2bdb536	🟢 Running 🔍 🔍	t2.medium	🕒 Initializing	View al

i-03c973945e2bdb536 (TerraformFinal)

Instance ID

[🔍](#) i-03c973945e2bdb536

IPv6 address

Public IPv4 address

[🔍](#) 99.79.55.27 | [open address](#) [🔗](#)

Instance state

```
Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

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

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

/usr/bin/xauth: file /home/ubuntu/.Xauthority does not exist
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-11-18:~$
ubuntu@ip-172-31-11-18:~$
ubuntu@ip-172-31-11-18:~$
```

1. Create Linux VM on AWS Cloud - Ubuntu (preferred to use min t2.medium as instance type)
Get connected to Linux VM using ssh gitbash or terminal or any medium

2. install Java

- 1.sudo apt update -> update the package manager
- 2.sudo apt install openjdk-21-jdk -> install java
- java -version -> To check java is installed or not

sudo apt update

sudo apt install openjdk-17-jdk

3. Install Jenkins

Create keyring directory if it doesn't exist

```
sudo mkdir -p /etc/apt/keyrings
```

Download and add the Jenkins GPG key

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

Add Jenkins repo to your sources list

```
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" \
```

```
| sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
sudo apt update
```

```
sudo apt install -y jenkins
```

4. Start and verify Jenkins

```
sudo systemctl enable jenkins
```

```
sudo systemctl start jenkins
```

Verify Jenkins

```
sudo systemctl status jenkins
```

5. Open Jenkins server in browser (also make sure edit inbond rules and add 8080 in security group)

<http://public-ip:8080/>

6: Copy Jenkins admin password

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

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

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-11-18:~$ sudo systemctl enable jenkins
sudo systemctl start jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-11-18:~$ sudo systemctl status Jenkins
Unit Jenkins.service could not be found.
ubuntu@ip-172-31-11-18:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-07-15 01:21:24 UTC; 37s ago
     Main PID: 4365 (java)
       Tasks: 49 (limit: 4670)
      Memory: 568.5M (peak: 580.9M)
         CPU: 19.351s
    CGroup: /system.slice/jenkins.service
            └─4365 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war

Jul 15 01:21:20 ip-172-31-11-18 jenkins[4365]: d47f0116307a4d23ace70aafa71e62d1
Jul 15 01:21:20 ip-172-31-11-18 jenkins[4365]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jul 15 01:21:20 ip-172-31-11-18 jenkins[4365]: *****
Jul 15 01:21:20 ip-172-31-11-18 jenkins[4365]: *****
Jul 15 01:21:20 ip-172-31-11-18 jenkins[4365]: *****
```

Install Terraform

```
ubuntu@ip-172-31-11-18:~$ sudo vi terraform.sh
```

```
#!/bin/bash
```

```
set -e
```

```
echo " Updating system packages..."
sudo apt-get update -y
sudo apt-get install -y curl unzip gnupg software-properties-common
```

```
echo " Adding HashiCorp GPG key..."
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
echo " Adding HashiCorp repo to apt sources..."
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" \
| sudo tee /etc/apt/sources.list.d/hashicorp.list > /dev/null
```

```
echo " Updating package list and installing Terraform..."
sudo apt-get update -y
sudo apt-get install terraform -y
```

```
echo "✔ Verifying Terraform version..."
terraform -version
```

```
echo " Terraform installation completed!"
```

```
ubuntu@ip-172-31-11-18:~$ sudo vi terraform.sh
ubuntu@ip-172-31-11-18:~$ ls
terraform.sh
ubuntu@ip-172-31-11-18:~$ sudo chmod +x terraform.sh
ubuntu@ip-172-31-11-18:~$ ls -l
total 4
-rwxr-xr-x 1 root root 798 Jul 15 01:28 terraform.sh
```

```
ubuntu@ip-172-31-11-18:~$ sudo sh terraform.sh
```



```

Fetchd 204 kB in 1s (354 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  terraform
0 upgraded, 1 newly installed, 0 to remove and 59 not upgraded.
Need to get 28.4 MB of archives.
After this operation, 93.6 MB of additional disk space will be used.
Get:1 https://apt.releases.hashicorp.com noble/main amd64 terraform amd64 1.12.2-1 [28.4 MB]
Fetchd 28.4 MB in 0s (87.4 MB/s)
Selecting previously unselected package terraform.
(Reading database ... 86268 files and directories currently installed.)
Preparing to unpack .../terraform_1.12.2-1_amd64.deb ...
Unpacking terraform (1.12.2-1) ...
Setting up terraform (1.12.2-1) ...
Scanning processes...
Scanning linux images...

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.
[✓] Verifying Terraform version...
Terraform v1.12.2
on linux_amd64
* Terraform installation completed!
ubuntu@ip-172-31-11-18:~$

```

1:40

```
ubuntu@ip-172-31-11-18:~$ sudo vi k8s.sh
```

```
#!/bin/bash
```

```
set -e
```

```
echo " Updating system packages..."
```

```
sudo apt-get update -y
```

```
echo " Downloading latest kubectl binary..."
```

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
echo " Verifying the binary checksum..."
```

```
curl -LO "https://dl.k8s.io/release/$(curl -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(cat kubectl.sha256) kubectl" | sha256sum --check
```

```
echo " Installing kubectl..."
```

```
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```

```
echo "✔ Verifying kubectl version..."
```

```
kubectl version --client
```

```
echo " kubectl installed successfully!"
```

```
ubuntu@ip-172-31-11-18:~$ sudo vi k8s.sh
```

```
ubuntu@ip-172-31-11-18:~$ sudo chmod +x k8s.sh
```

```
ubuntu@ip-172-31-11-18:~$ sh k8s.sh
#!/bin/bash
```

```
set -e
```

```
echo " Updating system packages..."
sudo apt-get update -y
```

```
echo " Fetching latest stable version..."
KUBECTL_VERSION=$(curl -L -s https://dl.k8s.io/release/stable.txt)
```

```
echo " Downloading kubectl version $KUBECTL_VERSION..."
curl -LO "https://dl.k8s.io/release/${KUBECTL_VERSION}/bin/linux/amd64/kubectl"
```

```
echo " Downloading checksum..."
curl -LO "https://dl.k8s.io/release/${KUBECTL_VERSION}/bin/linux/amd64/kubectl.sha256"
```

```
echo "✓ Verifying checksum..."
echo "$(cat kubectl.sha256) kubectl" | sha256sum --check -
```

```
echo " Installing kubectl..."
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```

```
echo " Verifying installation..."
kubectl version --client
```

```
echo " kubectl installed successfully!"
```

```
100  138  100  138   0   0  1745    0  ---:--:  ---:--:  ---:--:  1769
100  57.3M  100  57.3M   0   0  3281k    0  0:00:17  0:00:17  ---:--:  3315k
📦 Downloading checksum...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  138  100  138   0   0  1737    0  ---:--:  ---:--:  ---:--:  1746
100  64  100  64   0   0  446    0  ---:--:  ---:--:  ---:--:  446
☑ Verifying checksum...
kubectl: OK
🔧 Installing kubectl...
🔧 Verifying installation...
Client Version: v1.33.2
Kustomize Version: v5.6.0
🌟 kubectl installed successfully!
ubuntu@ip-172-31-11-18:~$
```

```
ubuntu@ip-172-31-11-18:~$ sudo vi aws-cli.sh
ubuntu@ip-172-31-11-18:~$ sudo chmod +x aws-cli.sh
```

```
#!/bin/bash
```

```
set -e
```

```
echo " Updating system packages..."
sudo apt-get update -y
sudo apt-get install -y unzip curl
```

```
echo " Downloading AWS CLI v2..."
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```

echo " Unzipping..."
unzip -q awscliv2.zip

echo " Installing AWS CLI..."
sudo ./aws/install

echo " Cleaning up..."
rm -rf aws awscliv2.zip

echo "✔ Verifying AWS CLI installation..."
aws --version

echo " AWS CLI installed successfully!"

```

```
ubuntu@ip-172-31-11-18:~$ sh aws-cli.sh
```

```

Hit:3 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://apt.releases.hashicorp.com noble InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
unzip is already the newest version (6.0-28ubuntu4.1).
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.
📦 Downloading AWS CLI v2...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 63.2M  100 63.2M    0     0  32.5M      0  0:00:01  0:00:01 --:--:-- 32.4M
📦 Unzipping...
🔧 Installing AWS CLI...
Found preexisting AWS CLI installation: /usr/local/aws-cli/v2/current. Please rer
ubuntu@ip-172-31-11-18:~$

```

1:46

```
install kubectl
```

```

curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
# Make executable and move to /usr/local/bin
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
# Verify installation
kubectl version --client --output=yaml

```

```
install AWS CLI
```

```

# Install unzip (adjust for your package manager)
# For Debian/Ubuntu
$ sudo apt update && sudo apt install -y unzip

```

```

# For RHEL/CentOS
# sudo yum install -y unzip
# Download and install AWS CLI v2

```

```

$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip

```

```
sudo ./aws/install

# Clean up
$ rm -rf awscliv2.zip aws

# Verify installation
$ aws --version

ubuntu@ip-172-31-11-18:~$ sudo chmod +x eksctl.sh
Install eksctl

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

# Verify installation
eksctl version
#!/bin/bash

set -e

echo " Updating system packages..."
sudo apt-get update -y
sudo apt-get install -y curl tar

echo " Fetching latest eksctl version..."
LATEST_VERSION=$(curl -s https://api.github.com/repos/eksctl-io/eksctl/releases/latest | grep
tag_name | cut -d '"' -f 4)

if [[ -z "$LATEST_VERSION" ]]; then
    echo "❌ Failed to retrieve eksctl version. Check your network or GitHub rate limits."
    exit 1
fi

echo " Latest version is $LATEST_VERSION"

TAR_NAME="eksctl_Linux_amd64.tar.gz"
DOWNLOAD_URL="https://github.com/eksctl-
io/eksctl/releases/download/${LATEST_VERSION}/${TAR_NAME}"

echo " Downloading from: $DOWNLOAD_URL"
curl -LO "$DOWNLOAD_URL"

echo " Extracting..."
tar -xzf "$TAR_NAME"

echo " Installing eksctl to /usr/local/bin..."
sudo mv eksctl /usr/local/bin/

echo " Cleaning up..."
rm -f "$TAR_NAME"

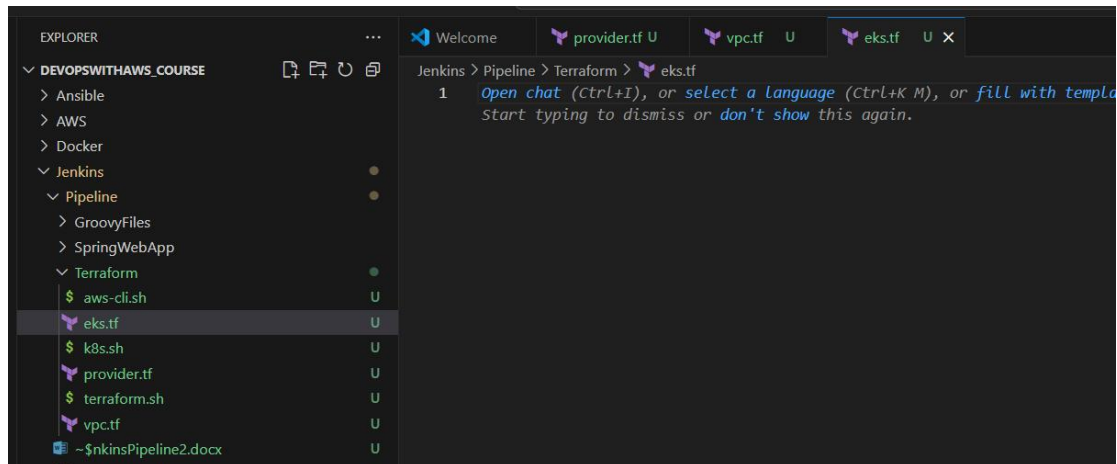
echo "✅ Verifying installation..."
eksctl version

echo " eksctl installed successfully!"
```

```
- Latest version is v0.210.0
📦 Downloading from: https://github.com/eksctl-io/eksctl/releases/download/v0.210.0/eksctl\_Linux\_amd64.tar.gz
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Dload  Upload   Total   Spent    Left   Speed
  0     0    0     0    0     0     0      0  0:00:00  0:00:00  0:00:00     0
100 33.3M 100 33.3M    0     0 2950k      0  0:00:11  0:00:11  0:00:00 3508k
📦 Extracting...
🔧 Installing eksctl to /usr/local/bin...
🧹 Cleaning up...
☑ Verifying installation...
0.210.0
🎉 eksctl installed successfully!
ubuntu@ip-172-31-44-18:~$
```

1:50

Go to VSCode create 3 tf files



1:51

To access resources over internet, we use Public subnets

provider.tf

```
locals {
  region = "ap-south-1"
  name    = "telusko-eks-cluster"
  vpc_cidr = "10.123.0.0/16"
  azs     = ["ap-south-1a", "ap-south-1b"]
  public_subnets = ["10.123.1.0/24", "10.123.2.0/24"]
  private_subnets = ["10.123.3.0/24", "10.123.4.0/24"]
  intra_subnets = ["10.123.5.0/24", "10.123.6.0/24"]
  tags = {
    Example = local.name
  }
}
```

```
provider "aws" {
  region = "ap-south-1"
}
```

vpc.tf

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  version = "~> 4.0" # Latest 4.x version of VPC to be used

  name = local.name
  cidr = local.vpc_cidr
}
```

```

    azs            = local.azs
    private_subnets = local.private_subnets
    public_subnets  = local.public_subnets
    intra.intra_subnets = local.intra_subnets

    enable_nat_gateway = true # if Private subnets require internet access to access external APIs or
resources then we need to enable the NAT Gateway

    public_subnets_tags = {
        "kubernetes.io/role/elb" = 1
    }

    private_subnets_tags = {
        "kubernetes.io/role/internal-elb" = 1
    }
}

eks.tf

module "eks" {
    source = "terraform-aws-modules/eks/aws" # specifies location of module from Terraform AWS
registry
    version = "19.15.1"

    cluster_name            = local.name
    cluster_endpoint_public_access = true # to enable public access to all cluster endpoints

    cluster_addons = {
        coredns = {
            most_recent = true
        }
        kube-proxy = { # for kubernetes API communication within the cluster for networking purpose,
            kube-proxy is required
            most_recent = true
        }
        vpc-cni = {
            most_recent = true
        }
    }

    # for networking
    vpc_id            = module.vpc.vpc_id
    subnet_ids        = module.vpc.private_subnets
    control_plane_subnet_ids = module.vpc.intra_subnets

    # what type of machines you want
    eks_managed_node_group_defaults = {
        ami_type     = "AL2_x86_64"
        instance_types = ["t2.medium"]
        attach_cluster_primary_security_group = true
    }

    eks_managed_node_groups = {
        pipeline-cluster-wg = {
            min_size = 2
            max_size = 2

```

```
desired_size = 2

instance_types = ["t2.medium"]
capacity_type = "SPOT"

tags = {
    ExtraTag = "full_pipeline"
}
}
}
```

tags = local.tags

```
}
```

Next class building the pipeline