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:

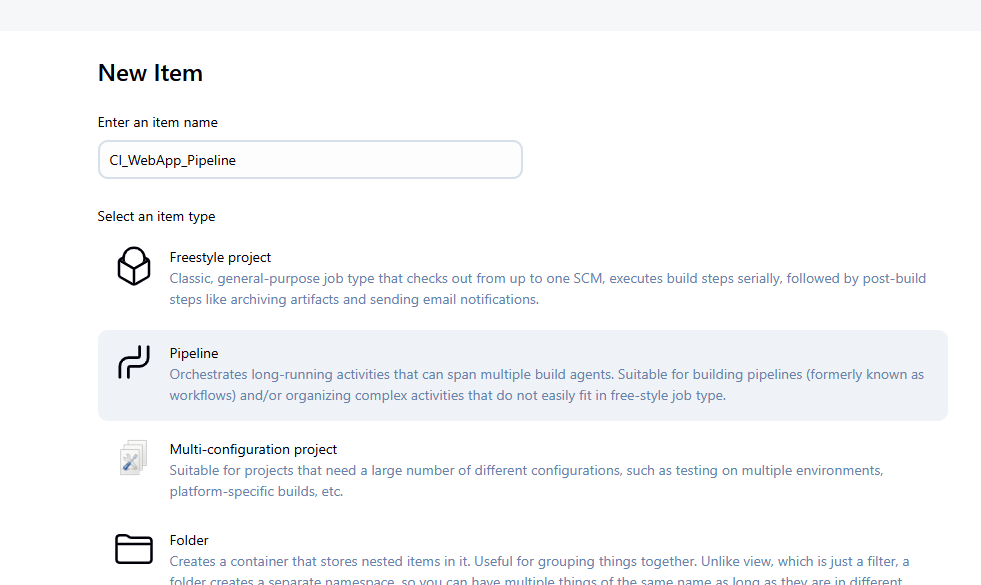
Pipeline2: Git Clone (to clone K8s manifest file and if we are writing manifest file in pipeline then need not clone) -> K8s Deployment

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



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}'

}

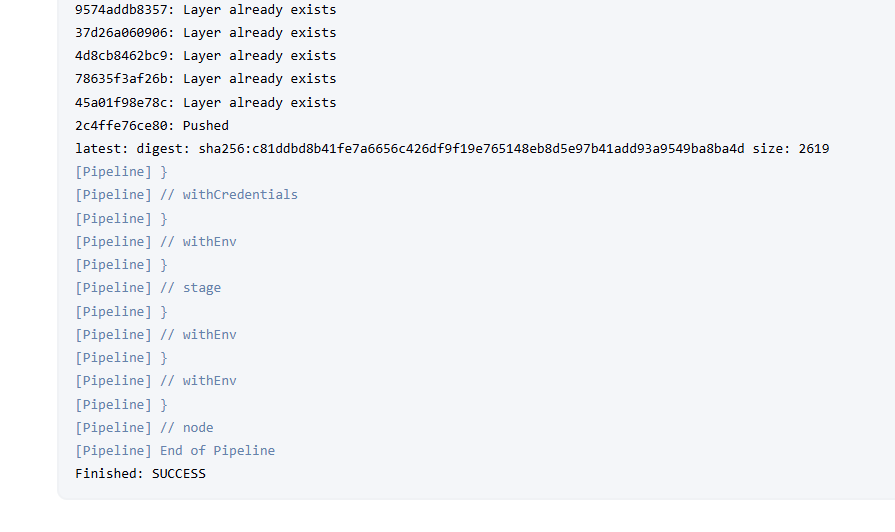
}

}

}

}

Make sure cluster is created on EKS-Host



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}'

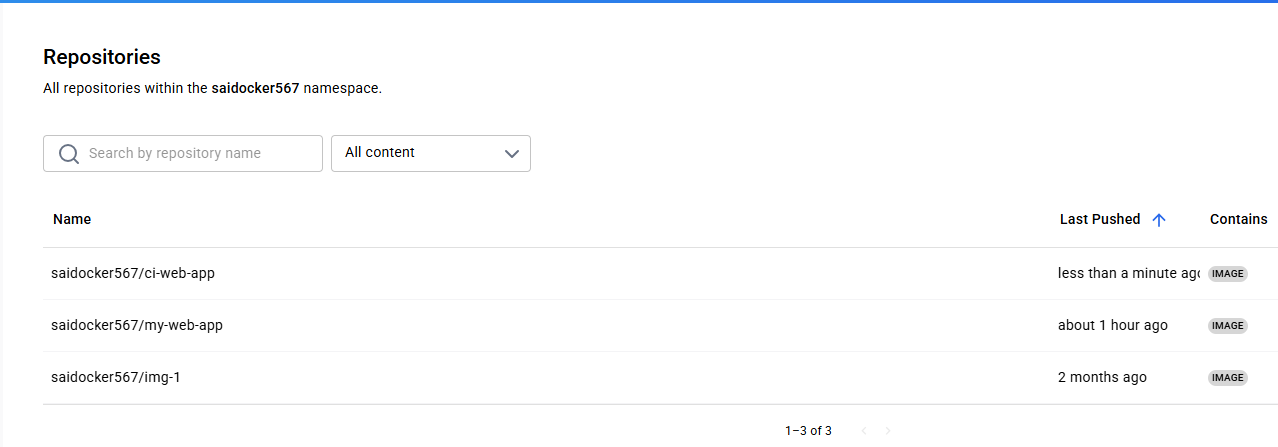
}

}

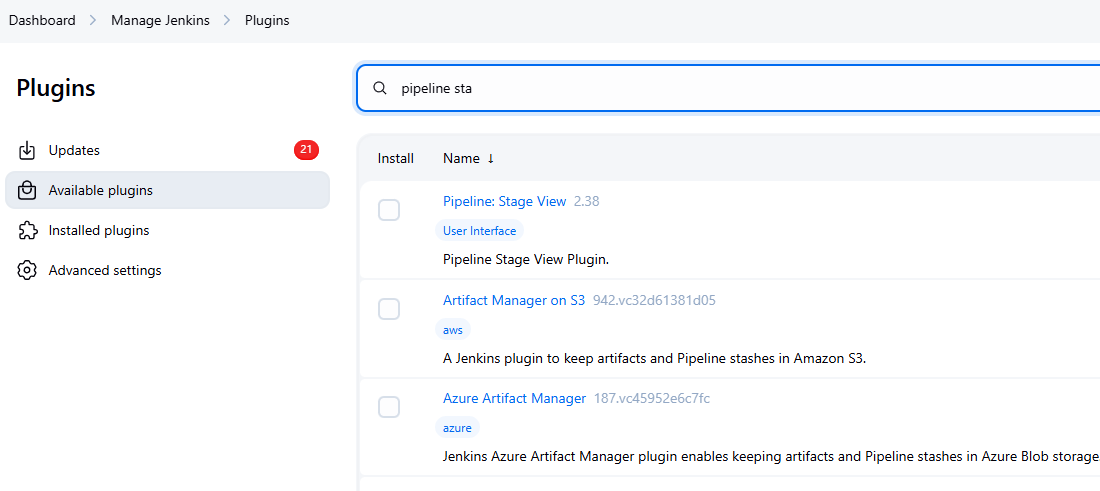
}

}

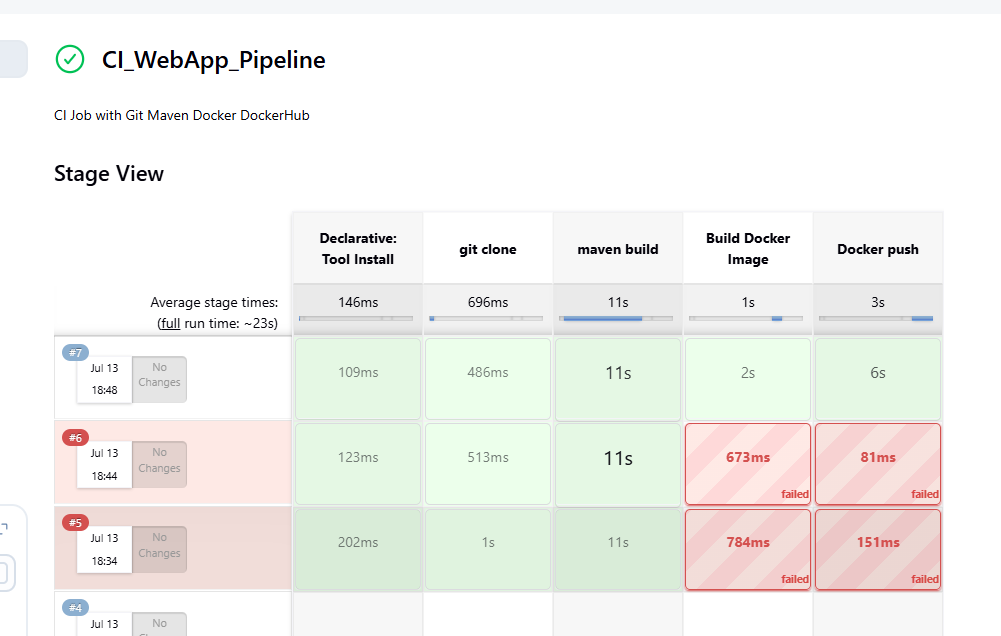
}



To trigger CD, we need Pipeline Stage



Different view due to Pipeline Stage View Plugin



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'

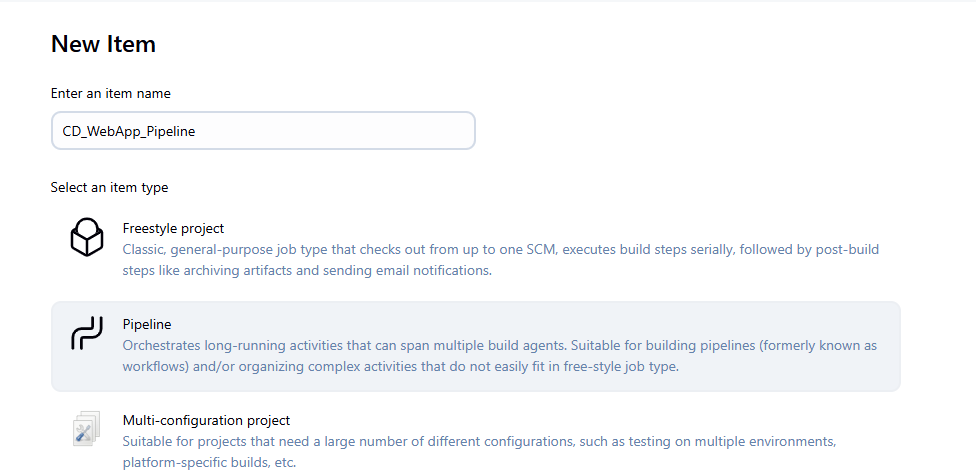
}

}

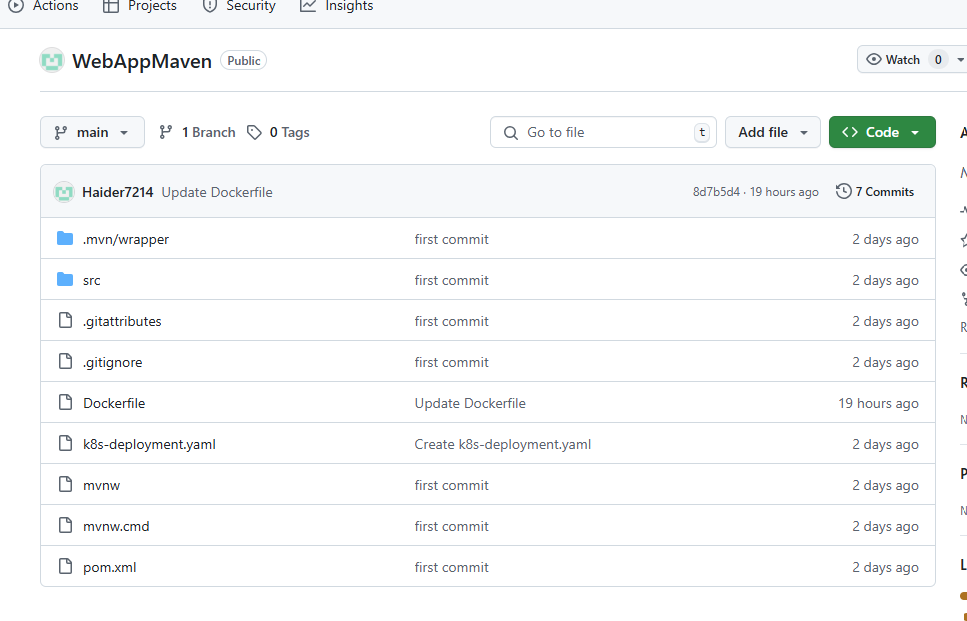
}

}

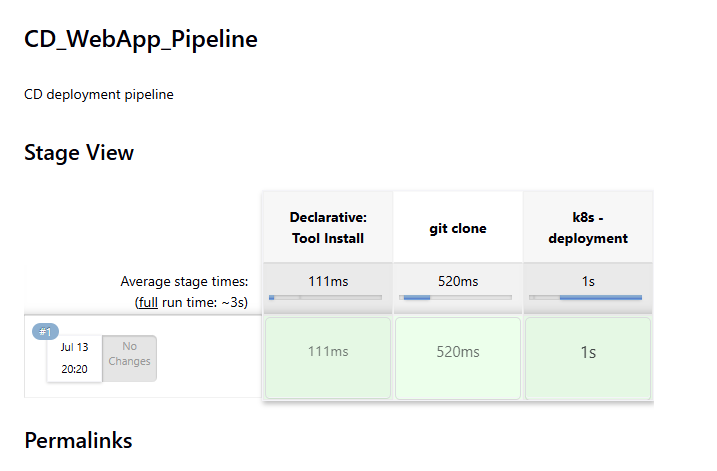
35:44



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



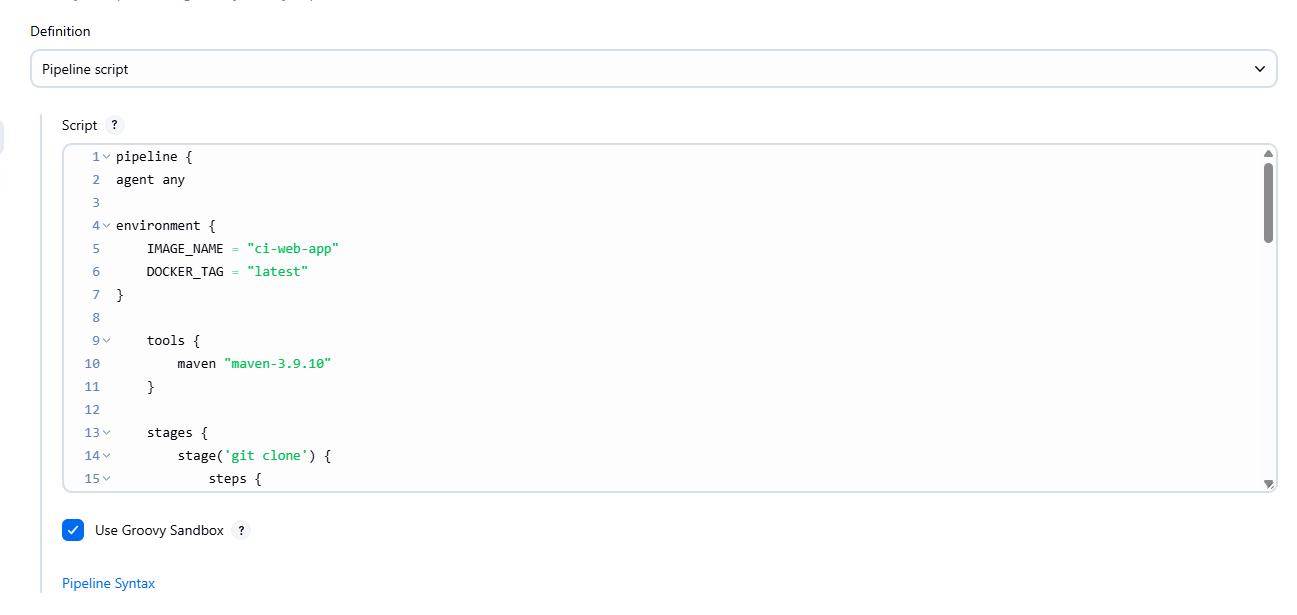
CD Pipeline was successful



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



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} saidocker567/${IMAGE\_NAME}:${DOCKER\_TAG}'

sh 'docker push saidocker567/${IMAGE\_NAME}:${DOCKER\_TAG}'

}

}

}

stage('Trigger CD job if CI is successful') {

steps {

build 'CD\_WebApp\_Pipeline'

}

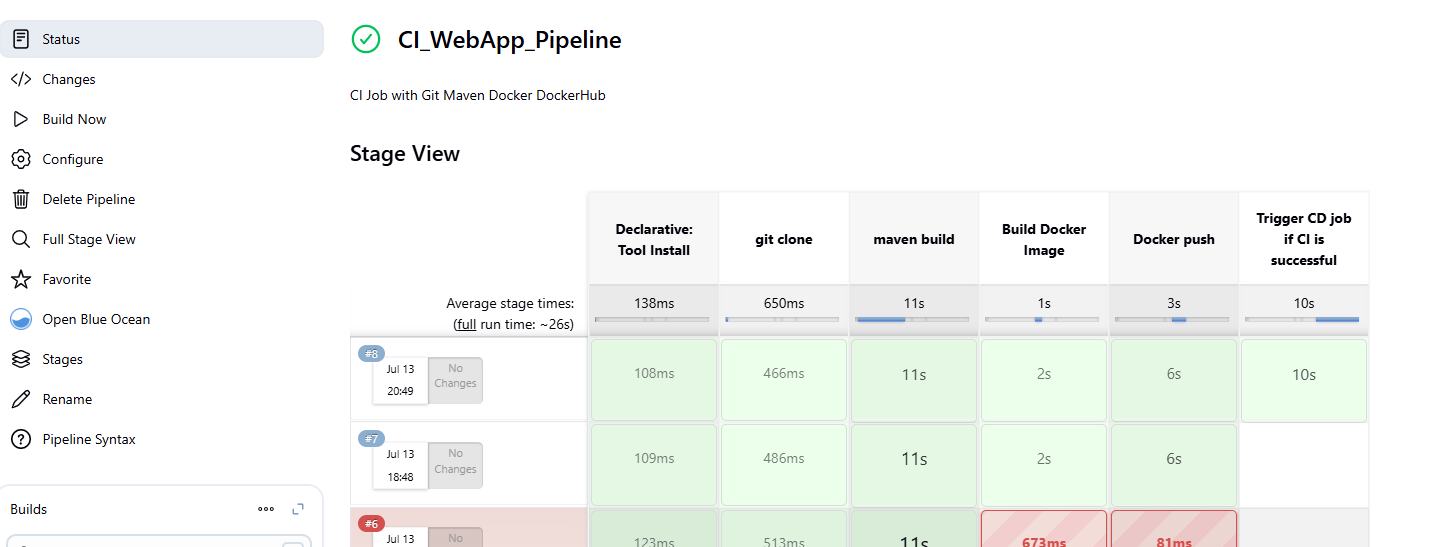
}

}

}

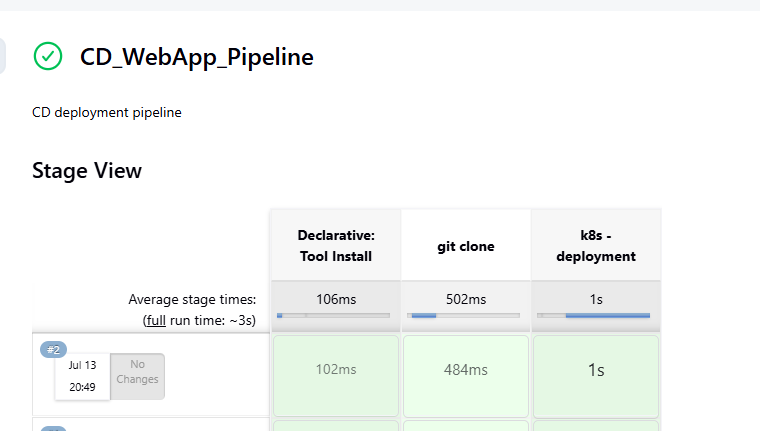
Apply and Save

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



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