

DevOps

# DevSecOps CI/CD Pipeline for an Uber Clone

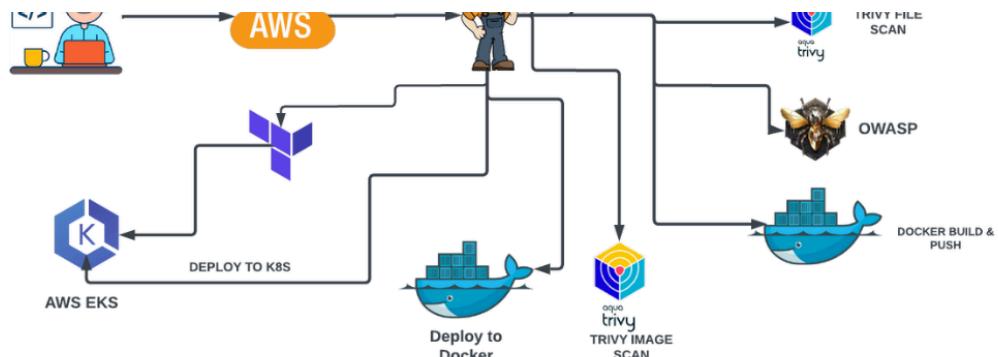


mrcloudbook.com · 29 December 2023

Mr Cloud Book

Home Blog DevSecOps Contact About Me Testimonials

Search Blogs



Welcome, tech enthusiasts! Today's journey is all about the cutting-edge realm of DevSecOps. Join us as we dive into deploying an Uber clone app using a comprehensive CI/CD approach. We'll explore the integration of SonarQube and OWASP for robust security measures, harness Docker Scout's power, and witness the orchestration of deployment onto an EKS cluster. This hands-on experience showcases Jenkins provisioning EKS via Terraform files, unlocking the seamless magic of DevSecOps in action.

## GITHUB REPO: <https://github.com/Aj7Ay/uber-clone.git>

### Step1: Launch an Ubuntu instance (T2.large)

1. **Sign in to AWS Console:** Log in to your AWS Management Console.
2. **Navigate to EC2 Dashboard:** Go to the EC2 Dashboard by selecting “Services” in the top menu and then choosing “EC2” under the Compute section.
3. **Launch Instance:** Click on the “Launch Instance” button to start the instance creation process.
4. **Choose an Amazon Machine Image (AMI):** Select an appropriate AMI for your instance. For example, you can choose Ubuntu image.
5. **Choose an Instance Type:** In the “Choose Instance Type” step, select `t2.large` as your instance type. Proceed by clicking “Next: Configure Instance Details.”
6. **Configure Instance Details:**
  - For “Number of Instances,” set it to 1 (unless you need multiple instances).
  - Configure additional settings like network, subnets, IAM role, etc., if necessary.
  - For “Storage,” click “Add New Volume” and set the size to 8GB (or modify the existing storage to 30GB).
  - Click “Next: Add Tags” when you’re done.
7. **Add Tags (Optional):** Add any desired tags to your instance. This step is optional, but it helps in organizing instances.
8. **Configure Security Group:**
  - Choose an existing security group or create a new one.
  - Ensure the security group has the necessary inbound/outbound rules to allow access as required.
9. **Review and Launch:** Review the configuration details. Ensure everything is set as desired.
10. **Select Key Pair:**

- Select “Choose an existing key pair” and choose the key pair from the drop down.
- Acknowledge that you have access to the selected private key file.
- Click “Launch Instances” to create the instance.

**11. Access the EC2 Instance:** Once the instance is launched, you can access it using the key pair and the instance's public IP or DNS.

Ensure you have necessary permissions and follow best practices while configuring security groups and key pairs to maintain security for your EC2 instance.

## Contents [ hide ]

[STEP 2: Create IAM role](#)

[Step3: Installations of Packages](#)

[Step4: Connect to Jenkins and Sonarqube](#)

[Step5: Terraform plugin install and EKS provision](#)

[Step6: Plugins installation & setup \(Java, Sonar, Nodejs, owasp, Docker\)](#)

[Step7: Configure in Global Tool Configuration](#)

[Step8: Configure Sonar Server in Manage Jenkins](#)

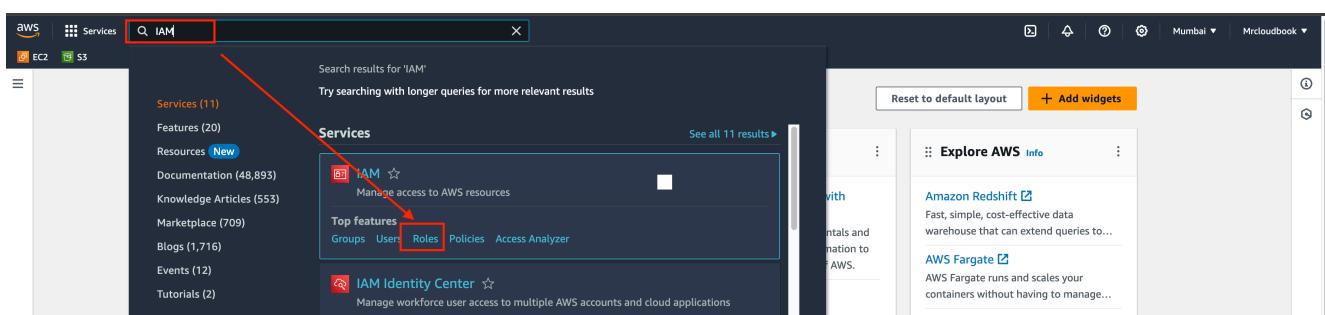
[Step9: Pipeline upto Docker](#)

[Step10: Kubernetes Deployment](#)

[Step11: Destruction](#)

## STEP 2: Create IAM role

Search for IAM in the search bar of AWS and click on roles.



Click on Create Role

The screenshot shows the AWS IAM Roles page. On the left sidebar, under 'Access management', 'Roles' is selected and highlighted with a red box. At the top right, there is a 'Create role' button, which is also highlighted with a red box and has a red arrow pointing towards it.

Select entity type as AWS service

Use case as EC2 and click on Next.

This screenshot shows the 'Trusted entity type' section of the 'Create role' wizard. It includes five options: 'AWS service' (selected and highlighted with a red box), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this, the 'Use case' section shows 'EC2' selected in the 'Service or use case' dropdown (highlighted with a red box). Under 'Choose a use case for the specified service.', the 'EC2' option is selected (highlighted with a red box).

For permission policy select Administrator Access (Just for learning purpose), click Next.

This screenshot shows the 'Add permissions' step of the 'Create role' wizard. It displays a list of available policies. The 'AdministratorAccess' policy is selected (highlighted with a red box) and its details are shown: 'AWS managed - job function' and 'Provides full access to AWS services an...'. Other policies listed include 'AdministratorAccess-Amplify'.

Provide a Name for Role and click on Create role.

Role details

**Role name**  
Enter a meaningful name to identify this role.  
**MARIO**

**Description**  
Add a short explanation for this role.  
Allows EC2 instances to call AWS services on your behalf.

**Step 1: Select trusted entities**

**Trust policy**

```

1  {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "sts:AssumeRole"
8       ],
9       "Principal": [
10        {
11          "Service": [
12            "ec2.amazonaws.com"
13          ]
14        }
15      ]
16    }
  
```

Role is created.

Identity and Access Management (IAM)

**Roles (3) Info**

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linker)	-
<b>MARIO</b>	AWS Service: ec2	-

Now Attach this role to Ec2 instance that we created earlier, so we can provision cluster from that instance.

Go to EC2 Dashboard and select the instance.

Click on Actions -> Security -> Modify IAM role.

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

Instance state: running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Test	i-0c92d797a3813a128	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	192.168.0.11

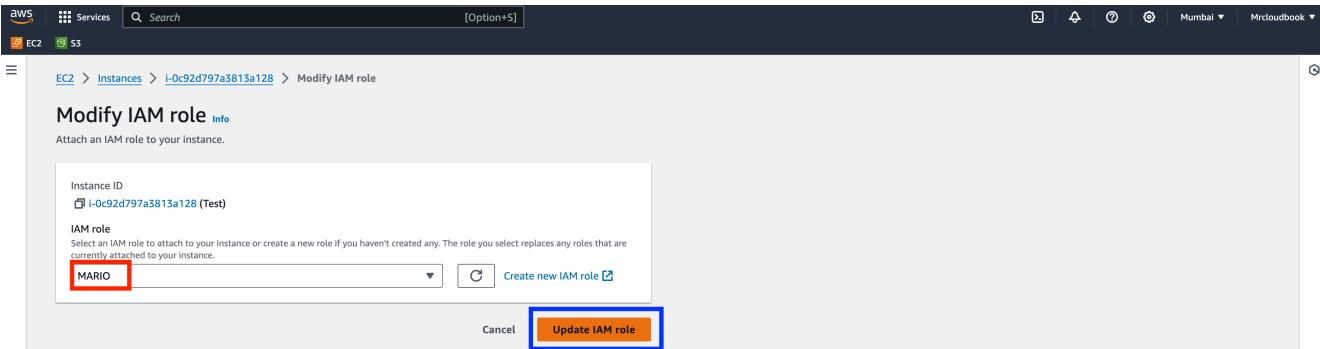
**Actions**

- Connect
- View details
- Manage instance state
- Instance settings
- Networking
- Security
- Image and templates
- Monitor and troubleshoot

**Security**

- Change security groups
- Get Windows password
- Modify IAM role**

Select the Role that created earlier and click on Update IAM role.



Connect the instance to Mobaxtreme or Putty

### Step3: Installations of Packages

create shell script in Ubuntu ec2 instance

```
sudo su    #run from inside root
vi script1.sh
```

Enter this script into it

This script installs Jenkins, Docker , Kubectl, Terraform, AWS Cli, Sonarqube

```
#!/bin/bash
sudo apt update -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/api/debian-stable/" | sudo tee /etc/apt/sources.list.d/adoptium.list
sudo apt update -y
sudo apt install temurin-17-jdk -y
/usr/bin/java --version
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo gpg --dearmor -o /usr/share/keyrings/jenkins-keyring.gpg
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable/ | sudo tee /etc/apt/sources.list.d/jenkins.list
sudo apt-get update -y
sudo apt-get install jenkins -y
sudo systemctl start jenkins
#install docker
# Add Docker's official GPG key:
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl gnupg -y
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --de
sudo chmod a+r /etc/apt/keyrings/docker.gpg
# Add the repository to Apt sources:
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/d
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-build
sudo usermod -aG docker ubuntu
newgrp docker
```

Now provide executable permissions to shell script



```
chmod 777 script1.sh
sh script1.sh
```



Let's Run the second script



```
vi script2.sh
```



Add this script



```
#!/bin/bash
# install trivy
sudo apt-get install wget apt-transport-https gnupg lsb-release -y
```

```
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor > /usr/share/keyrings/trivy.gpg
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb all main" | sudo tee /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy -y
#install terraform
sudo apt install wget -y
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
sudo apt update && sudo apt install terraform
#install Kubectl on Jenkins
sudo apt update
sudo apt install curl -y
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
#install Aws cli
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt-get install unzip -y
unzip awscliv2.zip
sudo ./aws/install
```

Nov 15, 2023

```
chmod 777 script2.sh
sh script2.sh
```

Now check the versions of packages

```
docker --version
trivy --version
aws --version
```

```
terraform --version
```

```
kubectl version
```

```
ubuntu@ip-172-31-11-71:~$  
ubuntu@ip-172-31-11-71:~$ trivy --version  
Version: 0.46.0  
ubuntu@ip-172-31-11-71:~$  
ubuntu@ip-172-31-11-71:~$ terraform --version  
Terraform v1.6.2  
on linux_amd64  
ubuntu@ip-172-31-11-71:~$  
ubuntu@ip-172-31-11-71:~$ kubectl --version  
error: unknown flag: --version  
See 'kubectl --help' for usage.  
ubuntu@ip-172-31-11-71:~$ kubectl version  
Client Version: v1.28.3  
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3  
Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fvers<br>timeout%3D32s'>;</script></head><body style='background-color:white; color:white;'>  
  
Authentication required  
<!--  
-->  
  
</body></html>  
ubuntu@ip-172-31-11-71:~$
```

Provide executable permissions from Mobaxtreme

```
sudo chmod 777 /var/run/docker.sock  
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

## Step4: Connect to Jenkins and Sonarqube

Now copy the public IP address of ec2 and paste it into the browser

<Ec2-ip:8080> #you will Jenkins login page

The screenshot shows a 'Getting Started' screen for Jenkins. The title 'Unlock Jenkins' is prominently displayed. A note states: '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'. Below this is a text input field labeled 'Administrator password' with a placeholder 'Enter password here'. A blue 'Continue' button is located at the bottom right.

Connect your Instance to Putty or Mobaxtreme and provide the below command for the Administrator password

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

```
ubuntu@ip-172-31-33-57:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
0ed1cb07ea7447c5a47d723022e74968
ubuntu@ip-172-31-33-57:~$ █
```

Now, install the suggested plugins.

[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 will now get installed and install all the libraries.

Create an admin user

[Getting Started](#)

# Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.414.1

[Skip and continue as admin](#)

[Save and Continue](#)

Click on save and continue.

Jenkins Dashboard

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

**Start building your software project**

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle  
2 Idle

**Set up a distributed build**

- Create a job →
- Set up an agent →
- Configure a cloud →
- Learn more about distributed builds ↗

Now Copy the public IP again and paste it into a new tab in the browser with 9000



<ec2-ip:9000> #runs sonar container



Log in to SonarQube

Login

Password

Log in Cancel

Enter username and password, click on login and change password



```
username admin
password admin
```



Instances | EC2 | ap-south-1    petstore Config [Jenkins]    SonarQube

6.140.95.9000/account/reset\_password

Web Servic... Coaching on DevO... LinkedIn Docker — A Beginn... Cloud Quest 100+ Essential Com... Advanced End-to-E...

**Update your password**

This account should not use the default password.

Enter a new password

All fields marked with \* are required

**Old Password \***

**New Password \***

**Confirm Password \***

**Update**

Update New password, This is Sonar Dashboard.

Not secure | 52.66.140.95:9000/projects/create

Gmail YouTube Amazon Web Service... Coaching on DevO... LinkedIn Docker — A Beginn... Cloud Quest 100+ Essential Com... Advanced End-to-E... LINUX - YouTube T... How to Install Jenk...

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration Search for projects... A

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform. First, you need to set up a DevOps platform configuration.

**From Azure DevOps** **From Bitbucket Server** **From Bitbucket Cloud** **From GitHub** **From GitLab**

Set up global configuration Set up global configuration Set up global configuration Set up global configuration Set up global configuration

## Step5: Terraform plugin install and EKS provision

Now go to Jenkins and add a terraform plugin to provision the AWS EKS using the Pipeline Job.

Go to Jenkins dashboard -> Manage Jenkins -> Plugins

Available Plugins, Search for Terraform and install it.

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'Plugins' section. A search bar at the top contains the text 'Terraform'. Below the search bar, there is a button labeled 'Install' with a blue background and white text. A dropdown arrow is positioned next to the 'Install' button. The main list area has a header with 'Install' and 'Name' columns, and a 'Released' column. A single plugin entry is listed: 'Terraform 1.0.10' with a 'Build Wrappers' link below it. To the right of the plugin name is the release date '3 yr 8 mo ago'. A note below the plugin entry states: 'This plugin provides a build wrapper for Terraform to launch and destroy infrastructure.'

Go to Putty and use the below command

let's find the path to our Terraform (we will use it in the tools section of Terraform)

The screenshot shows a terminal window with a dark theme. At the top, there is a search bar containing the text 'Quick connect...'. Below the search bar, the terminal prompt is 'ubuntu@ip-172-31-42-229:~\$'. The user then types the command 'which terraform' and presses enter. The terminal displays the output: '/usr/bin/terraform'. The terminal window has a standard window title bar with minimize, maximize, and close buttons.

Now come back to Manage Jenkins -> Tools

Add the terraform in Tools

## Terraform installations

Add Terraform

**Terraform**

Name

Install directory

Install automatically ?

Add Terraform

Save
Apply

Apply and save.

CHANGE YOUR S3 BUCKET NAME IN THE BACKEND.TF

Now create a new job for the EKS provision

**Enter an item name**

» 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.

---

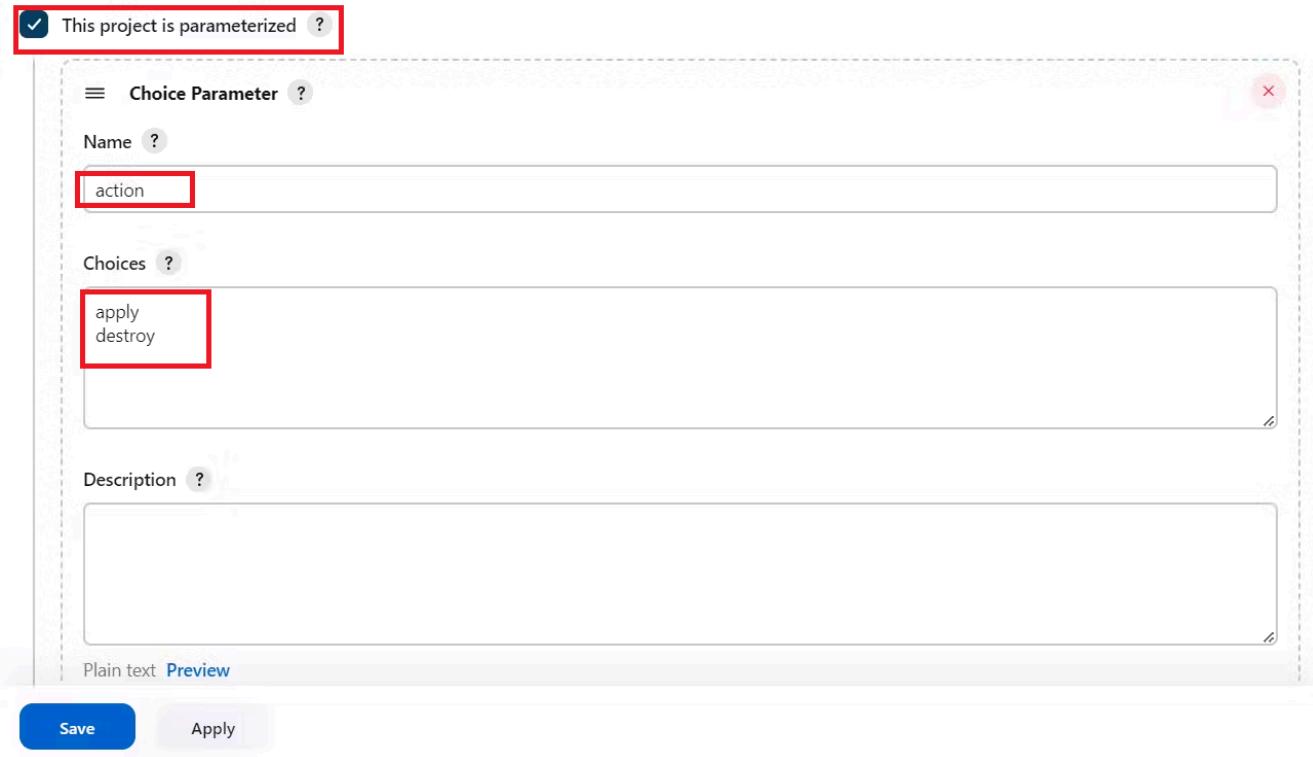
**Multi-configuration project**



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

I want to do this with build parameters to apply and destroy while building only.

you have to add this inside job like the below image



Let's add a pipeline



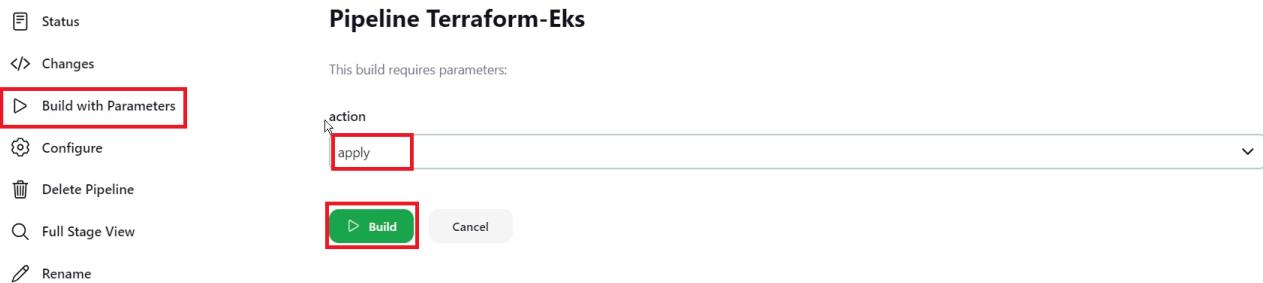
```
pipeline{
    agent any
    stages {
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Aj7Ay/uber'
            }
        }
        stage('Terraform version'){
            steps{
                sh 'terraform --version'
            }
        }
        stage('Terraform init'){
            steps{
                dir('EKS_TERRAFORM') {
                    sh 'terraform init'
                }
            }
        }
    }
}
```

```

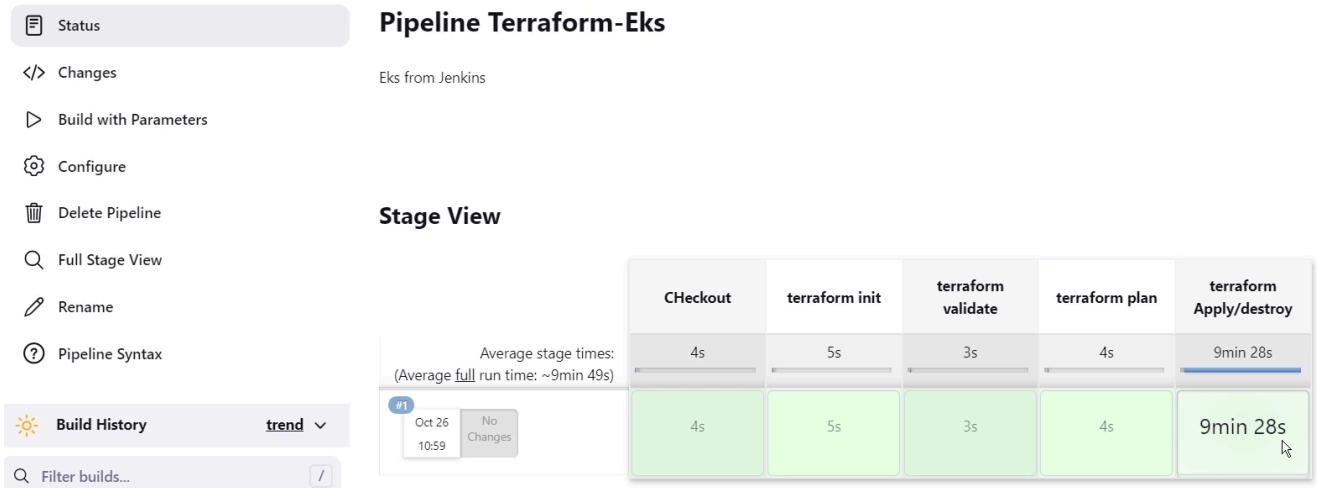
stage('Terraform validate'){
    steps{
        dir('EKS_TERRAFORM') {
            sh 'terraform validate'
        }
    }
}
stage('Terraform plan'){
    steps{
        dir('EKS_TERRAFORM') {
            sh 'terraform plan'
        }
    }
}
stage('Terraform apply/destroy'){
    steps{
        dir('EKS_TERRAFORM') {
            sh 'terraform ${action} --auto-approve'
        }
    }
}
}
}

```

Let's apply and save and Build with parameters and select action as apply



Stage view it will take max 10mins to provision



Check in Your Aws console whether it created EKS or not.

EKS > Clusters

Clusters (1) Info

Filter clusters

Cluster name	Status	Kubernetes version	Provider
EKS_CLOUD	Active	1.28	EKS

Ec2 instance is created for the Node group

Instances (1/2) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Put
Jenkins-ARGO	i-0323f37f837248e53	Running	t2.large	2/2 checks passed	No alarms	ap-south-1b	ec2
	i-049634a401c64808b	Running	t2.medium	2/2 checks passed	No alarms	ap-south-1b	ec2

## Step6: Plugins installation & setup (Java, Sonar, Nodejs, owasp, Docker)

Go to Jenkins dashboard

Manage Jenkins -> Plugins -> Available Plugins

Search for the Below Plugins

Eclipse Temurin installer

## Sonarqube Scanner

### NodeJs

### Owasp Dependency-Check

### Docker

### Docker Commons

### Docker Pipeline

### Docker API

### Docker-build-step

 <a href="#">Eclipse Temurin installer 1.5</a>	Provides an installer for the JDK tool that downloads the JDK from <a href="https://adoptium.net">https://adoptium.net</a>	<a href="#">This plugin is up for adoption! We are looking for new maintainers. Visit our <u>Adopt a Plugin</u> initiative for more information.</a>	1 yr 0 mo ago
 <a href="#">SonarQube Scanner 2.16.1</a>	<a href="#">External Site/Tool Integrations</a> <a href="#">Build Reports</a>	This plugin allows an easy integration of <a href="#">SonarQube</a> , the open source platform for Continuous Inspection of code quality.	15 days ago
 <a href="#">NodeJS 1.6.1</a>	<a href="#">npm</a>	NodeJS Plugin executes <a href="#">NodeJS</a> script as a build step.	2 mo 10 days ago
 <a href="#">OWASP Dependency-Check 5.4.3</a>	<a href="#">Security</a> <a href="#">DevOps</a> <a href="#">Build Tools</a> <a href="#">Build Reports</a>	This plug-in can independently execute a <a href="#">Dependency-Check</a> analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.	1 mo 16 days ago
 <a href="#">Docker 1.5</a>	<a href="#">Cloud Providers</a> <a href="#">Cluster Management</a> <a href="#">docker</a>	This plugin integrates Jenkins with <a href="#">Docker</a>	1 mo 21 days ago

**Docker Commons** 439.va\_3cb\_0a\_6a\_fb\_29

Provides the common shared functionality for various Docker-related plugins.

**Docker Pipeline** 572.v950f58993843

Build and use Docker containers from pipelines.

**Docker API** 3.3.1-79.v20b\_53427e041

This plugin provides `docker-java` API for other plugins.

**This plugin is up for adoption!** We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.

**docker-build-step** 2.10

## Step7: Configure in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16) → Click on Apply and Save

Dashboard > Manage Jenkins > Tools

JDK installations

Add JDK

**JDK**

Name: jdk17

Install automatically

**Install from adoptium.net**

Version: jdk-17.0.8.1+1

Add Installer

Dashboard > Manage Jenkins > Tools

**NodeJS**

Name: node16

Install automatically

**Install from nodejs.org**

Version: NodeJS 16.2.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see `npm install -g`. Note that you can fix the packages version by using the syntax `'packageName@version'`

For Sonarqube use the latest version

Dashboard &gt; Manage Jenkins &gt; Tools

## SonarQube Scanner installations

[Add SonarQube Scanner](#)**SonarQube Scanner**

sonar-scanner

 Install automatically [?](#)**Install from Maven Central**

## Version

SonarQube Scanner 5.0.1.3006

[Add Installer ▾](#)[Add SonarQube Scanner](#)[Save](#) [Apply](#)**For Owasp use the 6.5.1 version**

Dashboard &gt; Manage Jenkins &gt; Tools

## Dependency-Check installations

[Add Dependency-Check](#)**Dependency-Check**

## Name

DP-Check

 Install automatically [?](#)**Install from github.com**

## Version

dependency-check 6.5.1

[Add Installer ▾](#)**Use the latest version of Docker**

Dashboard &gt; Manage Jenkins &gt; Tools

Docker installations

Add Docker

**Docker**

Name: docker

Install automatically ?

**Download from docker.com**

Docker version: latest

Add Installer ▾

Click apply and save.

## Step8: Configure Sonar Server in Manage Jenkins

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token

The screenshot shows the Sonarqube administration interface. The top navigation bar has tabs for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration tab is highlighted with a red box. Below the navigation, there's a sub-navigation for Configuration, Security (which is currently selected), Projects, System, and Marketplace. A dropdown menu is open under the Security tab, with 'Users' highlighted by a red box. Other options in the dropdown include Groups, Global Permissions, and Permission Templates. The main content area shows general settings like 'Edit global configuration' and a search bar.

click on update Token

This screenshot shows the 'Users' section of the Sonarqube administration. At the top right, there's a 'Tokens' button highlighted with a red box. Below it, a table lists two users: 'sonar-administrators' and 'sonar-users'. Each user entry includes a small icon, the user name, and a 'Tokens' button. At the bottom right of the table is a large 'Update Tokens' button.

Create a token with a name and generate

**Tokens of Administrator**

**Generate Tokens**

Name	Expires in
Enter Token Name	30 days
<input type="button" value="Generate"/>	

**New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!**

**Copy** :squ\_21d162904c1c72cf8b39665f96480185c99dc2f9

Name	Type	Project	Last use	Created	Expiration
Jenkins	User		Never	September 8, 2023	October 8, 2023
					<b>Revoke</b>

**copy Token**

Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this

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

### New credentials

Kind: Secret text

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

Secret: POST THE TOKEN HERE

ID: Sonar-token

Description: Sonar-token

**Create**

You will see this page once you click on create

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

ID	Name	Kind	Description
Sonar-token	sonar	Secret text	sonar

Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

**SonarQube servers**

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

**Environment variables** Enable injection of SonarQube server configuration as build environment variables

**SonarQube installations**

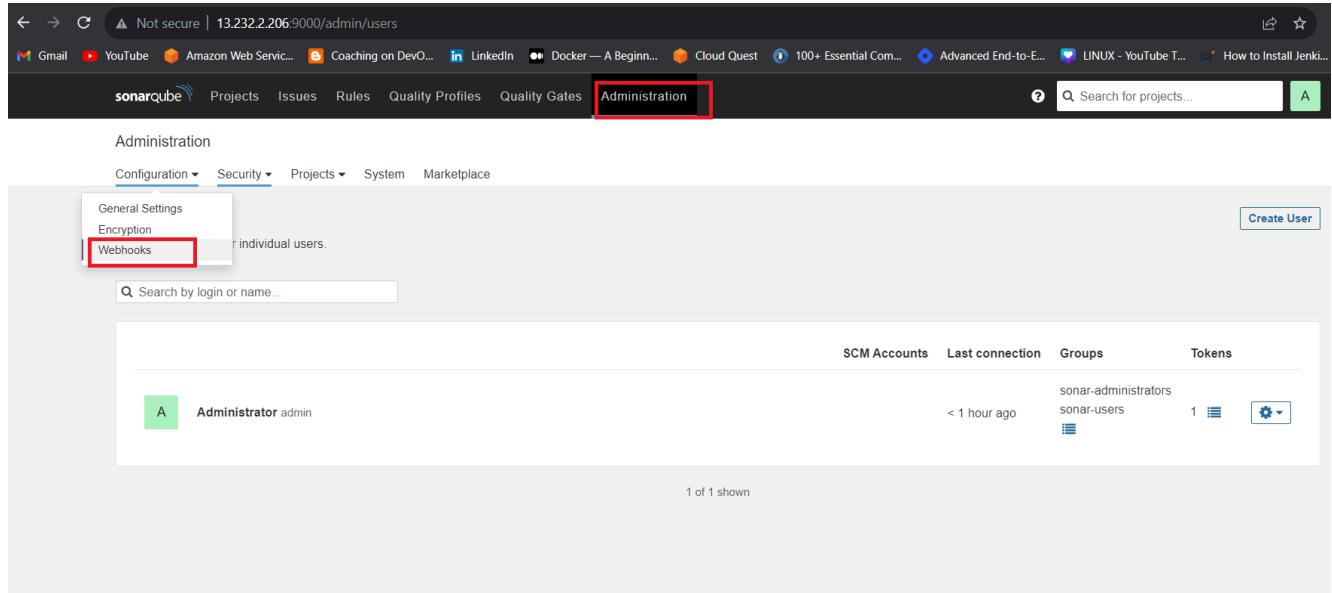
List of SonarQube installations

Name	<input type="text" value="sonar-server"/>	<span style="color: red;">X</span>
Server URL	Default is http://localhost:9000 <input type="text" value="http://13.232.17.191:9000"/>	
Server authentication token	SonarQube authentication token. Mandatory when anonymous access is disabled. <input type="text" value="Sonar-token"/>	
<input style="margin-right: 10px;" type="button" value="Add"/> <input type="button" value="Save"/> <input type="button" value="Apply"/>		

Click on Apply and Save

In the Sonarqube Dashboard add a quality gate also

Administration-> Configuration->Webhooks



The screenshot shows the SonarQube Administration interface. The top navigation bar has a 'Administration' tab highlighted with a red box. Below it, there's a sub-navigation bar with 'Configuration', 'Security', 'Projects', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. The 'Administration' tab is active. On the left, there's a sidebar with 'General Settings', 'Encryption', and 'Webhooks' (which is also highlighted with a red box). A search bar at the bottom of the sidebar says 'Search by login or name...'. The main content area shows a table with one row for 'Administrator admin'. The table includes columns for 'SCM Accounts', 'Last connection', 'Groups', and 'Tokens'. The 'Last connection' column shows '< 1 hour ago'. The 'Groups' column lists 'sonar-administrators' and 'sonar-users'. The 'Tokens' column shows '1' and a gear icon. At the bottom of the main content area, it says '1 of 1 shown'.

Click on Create

The screenshot shows the SonarQube Administration interface under the 'Webhooks' section. It displays a message stating 'No webhook defined.' and a 'Create' button in the top right corner.

## Add details



```
#in url section of quality gate
<http://jenkins-public-ip:8080>/sonarqube-webhook/>
```



The screenshot shows the 'Create Webhook' dialog box. It has fields for 'Name' (set to 'jenkins') and 'URL' (set to 'http://43.204.36.242:8090/sonarqube-webhook/'). A note below the URL field explains the server endpoint. At the bottom, there is a 'Create' button.

Now add Docker credentials to the Jenkins to log in and push the image

Manage Jenkins -> Credentials -> global -> add credential

Add DockerHub Username and Password under Global Credentials

Kind

Username with password

Scope ?

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

Username ?

sevenajay

 Treat username as secret ?

Password ?

.....

ID ?

docker

Description ?

docker

**Create**

Create.

## Step09: Pipeline upto Docker

Now let's create a new job for our pipeline

**Enter an item name**

» 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.

 **Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **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.

Add this to Pipeline

```
pipeline{
    agent any
    tools{
```

```
        jdk 'jdk17'
        nodejs 'node16'
    }
environment {
    SCANNER_HOME=tool 'sonar-scanner'
}
stages {
    stage('clean workspace'){
        steps{
            cleanWs()
        }
    }
    stage('Checkout from Git'){
        steps{
            git branch: 'main', url: 'https://github.com/Aj7Ay/uber'
        }
    }
    stage("Sonarqube Analysis"){
        steps{
            withSonarQubeEnv('sonar-server') {
                sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectKey=Uber'''
            }
        }
    }
    stage("quality gate"){
        steps {
            script {
                waitForQualityGate abortPipeline: false, credentials:
            }
        }
    }
    stage('Install Dependencies') {
        steps {
            sh "npm install"
        }
    }
    stage('OWASP FS SCAN') {
        steps {
            dependencyCheck additionalArguments: '--scan ./ --disable'
            dependencyCheckPublisher pattern: '**/dependency-check-'
        }
    }
    stage('TRIVY FS SCAN') {
        steps {

```

```

        sh "trivy fs . &gt; trivyfs.txt"
    }
}

stage("Docker Build & Push"){
    steps{
        script{
            withDockerRegistry(credentialsId: 'docker', toolName
                sh "docker build -t uber ."
                sh "docker tag uber sevenajay/uber:latest "
                sh "docker push sevenajay/uber:latest "
            }
        }
    }
}

stage("TRIVY"){
    steps{
        sh "trivy image sevenajay/uber:latest &gt; trivyimage.txt"
    }
}

stage("deploy_docker"){
    steps{
        sh "docker run -d --name uber -p 3000:3000 sevenajay/uber"
    }
}
}
}

```

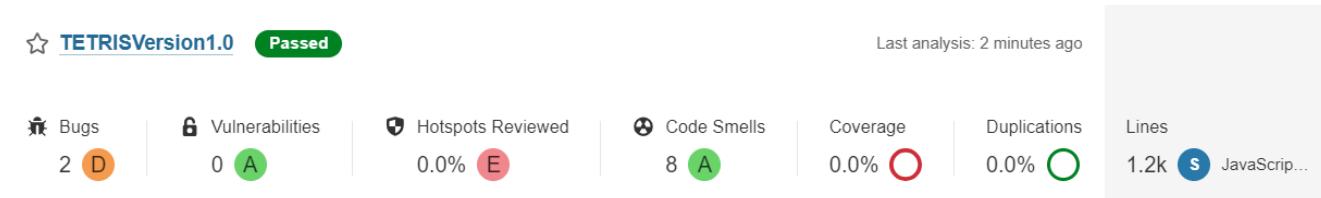
Click on Apply and save.

Build now

Stage view

Declarative: Tool Install	CO	Sonar analysis	QG	NPM	TRIVY FS	OWASP FS SCAN	Docker build and push	TRIVY IMAGE
10s	1s	29s	327ms	20s	3s	12min 29s	1min 9s	57s
199ms	1s	35s	346ms	12s	4s	6min 47s	3min 27s	2min 52s

To see the report, you can go to Sonarqube Server and go to Projects.



You can see the report has been generated and the status shows as passed. You can see that there are 1.2k lines it scanned. To see a detailed report, you can go to issues.

OWASP, You will see that in status, a graph will also be generated and Vulnerabilities.

### Dependency-Check Results

Severity Distribution			
15	47	25	
			Search <input type="button" value="🔍"/>
File Name	Vulnerability	Severity	Weakness
⊕ @babel/preset-env:7.12.1	NVD CVE-2023-45133	High	CWE-697
⊕ ansi-html:0.0.7	NVD CVE-2021-23424	High	NVD-CWE-noinfo
⊕ ansi-regex:4.1.0	NVD CVE-2021-3807	High	CWE-1333
⊕ ansi-regex:5.0.0	NVD CVE-2021-3807	High	CWE-1333
⊕ async:2.6.3	NVD CVE-2021-43138	High	CWE-1321
⊕ browserslist:4.14.2	NVD CVE-2021-23364	Medium	CWE-1333
⊕ browserslist:4.16.3	NVD CVE-2021-23364	Medium	CWE-1333
⊕ css-what:3.4.2	OSSINDEX CVE-2022-21222	High	CWE-1333
⊕ decode-uri-component:0.2.0	NVD CVE-2022-38778	Medium	CWE-20
⊕ decode-uri-component:0.2.0	NVD CVE-2022-38900	High	CWE-20

When you log in to Dockerhub, you will see a new image is created

sevenajay / tetrисv1

Description

This repository does not have a description

Last pushed: a minute ago

Docker commands

To push a new tag to this repository:

```
docker push sevenajay/tetrисv1:tagname
```

## Step10: Kubernetes Deployment

Go to Putty of your Jenkins instance SSH and enter the below command



```
aws eks update-kubeconfig --name <CLUSTER NAME> --region <CLUSTER REGION>
aws eks update-kubeconfig --name EKS_CLOUD --region ap-south-1
```



```
ubuntu@ip-172-31-11-71:~$ aws eks update-kubeconfig --name EKS_CLOUD --region ap-south-1
Added new context arn:aws:eks:ap-south-1:07201807785:cluster/EKS_CLOUD to /home/ubuntu/.kube/config
ubuntu@ip-172-31-11-71:~$
```

Let's see the nodes



```
kubectl get nodes
```



```
ubuntu@ip-172-31-11-71:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE     VERSION
ip-172-31-13-85.ap-south-1.compute.internal   Ready    <none>    111m   v1.28.1-eks-43840fb
ubuntu@ip-172-31-11-71:~$
```

Copy the config file to Jenkins master or the local file manager and save it



```
cat .kube/config
```



copy it and save it in documents or another folder save it as secret-file.txt

Note: create a secret-file.txt in your file explorer save the config in it and use this at the kubernetes credential section.

Install Kubernetes Plugin, Once it's installed successfully

Dashboard > Manage Jenkins > Plugins

## Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Kuber

Install

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<a href="#">Kubernetes Credentials</a> 0.11 kubernetes credentials Common classes for Kubernetes credentials	9 days 16 hr ago
<input checked="" type="checkbox"/>	<a href="#">Kubernetes Client API</a> 6.8.1-224.vd388fca_4db_3b kubernetes Library plugins (for use by other plugins) Kubernetes Client API plugin for use by other Jenkins plugins.	9 days 17 hr ago
<input checked="" type="checkbox"/>	<a href="#">Kubernetes</a> 4029.v5712230ccb_f8 Cloud Providers Cluster Management kubernetes Agent Management This plugin integrates Jenkins with Kubernetes	9 days 15 hr ago
<input checked="" type="checkbox"/>	<a href="#">Kubernetes CLI</a> 1.12.1 kubernetes Configure kubectl for Kubernetes	8 days 22 hr ago

goto manage Jenkins -> manage credentials -> Click on Jenkins global -> add credentials

**New credentials**

Kind

Secret file

Scope ?

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

File

Choose File Secret File.txt

ID ?

k8s

Description ?

k8s

**Create**

final step to deploy on the Kubernetes cluster



```
stage('Deploy to kubernets'){
    steps{
        script{
            dir('K8S') {
                withKubeConfig(caCertificate: '', clusterName:
                    sh 'kubectl apply -f deployment.yml'
                    sh 'kubectl apply -f service.yml'
                }
            }
        }
    }
}
```



You will see output like this.

## Step11: Destruction

Now Go to Jenkins Dashboard and click on Terraform-Eks job

And build with parameters and destroy action

It will delete the EKS cluster that provisioned

The screenshot shows the AWS CloudFormation console interface for a pipeline named 'Pipeline Terraform-Eks'. On the left, there's a sidebar with options like Status, Changes, Build with Parameters (which is selected and highlighted with a red box), Configure, Delete Pipeline, Full Stage View, Rename, and a minus sign. The main area is titled 'Pipeline Terraform-Eks' and contains a message: 'This build requires parameters:'. Below this is a dropdown labeled 'action' with the value 'destroy' selected (also highlighted with a red box). At the bottom right are two buttons: a green 'Build' button (highlighted with a red box) and a 'Cancel' button.

After 10 minutes cluster will delete and wait for it. Don't remove ec2 instance till that time.



Cluster deleted

The screenshot shows the AWS Lambda console with a section titled 'Clusters (0)'. There's a 'Info' link, a 'Delete' button, and an 'Add cluster' button. Below is a search bar with the placeholder 'Filter clusters'. A table has columns: Cluster name, Status, Kubernetes version, and Provider. The table body contains the message: 'No clusters' and 'You do not have any clusters.' At the bottom is a 'Create cluster' button.

Delete the Ec2 instance.



**Ajay Kumar Yegireddi** is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

## Comments

3 responses to “DevSecOps CI/CD Pipeline for an Uber Clone”

» **DevopsTechPulse**

29 December 2023

[...] DevSecOps CI/CD Pipeline for an Uber Clone [...]

[Reply](#)



**Nazir**

31 December 2023

Sir, Could you make video on above article with brief explanation it will really helpful.

[Reply](#)



**mrcloudbook.com**

31 December 2023

The process is same as old video projects

[Reply](#)

## Leave a Reply

Your email address will not be published. Required fields are marked \*

Comment \*

Name \*

Email \*

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot

reCAPTCHA  
[Privacy](#) - [Terms](#)

Post Comment

Uncategorized

AI

AI, DevOps

**How to Automate  
Incident Response :  
How Q Developer  
Helped Me Automate a  
Daily Pain Point**

22 July 2025

**How to Run Docker  
Model Runner on  
Ubuntu 24.04**

11 July 2025

**How to Install docker-ai on Ubuntu 24.04**

15 June 2025

## Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.



## Important Links

[Privacy Policy](#)

[Terms & Conditions](#)

[Contact](#)

## Resources

[Blog](#)

[YouTube Channel](#)