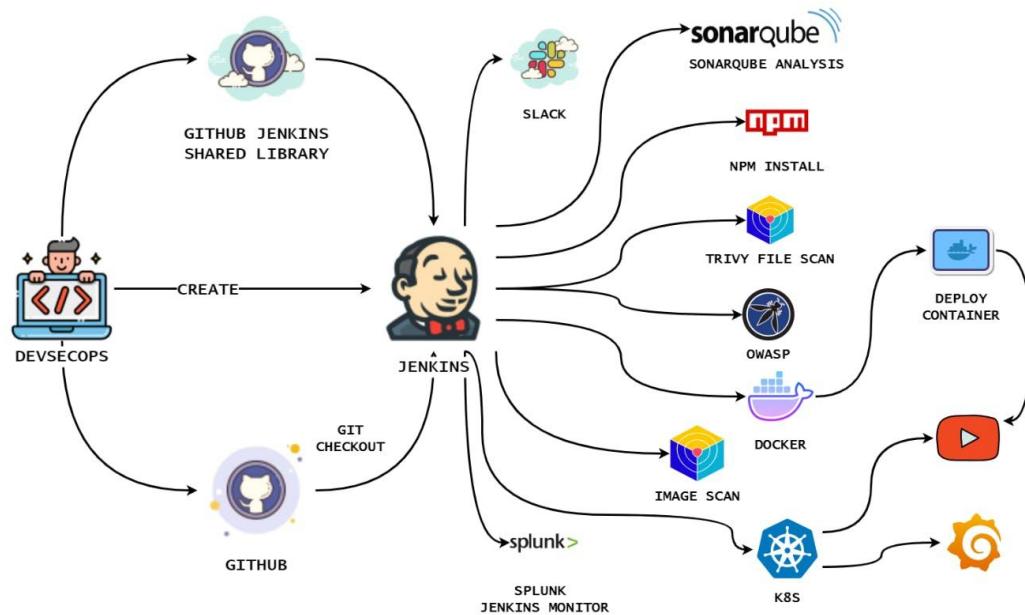


# Deploying a YouTube Clone App with DevSecOps and Jenkins Shared Library



## Table of contents

### STEPS:

Step 1: Launch an Ubuntu 22.04 instance for Jenkins

Step2A: Install Docker on the Jenkins machine

Step2B: Install Trivy on Jenkins machine

Step3A: Launch an Ubuntu instance for Splunk

Step3B: Install the Splunk app for Jenkins

Add Splunk Plugin in Jenkins

Restart Both Splunk and Jenkins

Step4A: Integrate Slack for Notifications

Step4B: Install the Jenkins CI app on Slack

Install Slack Notification Plugin in Jenkins

Step5A: Start Job

Step5B: Create a Jenkins shared library in GitHub

Step5C: Add Jenkins shared library to Jenkins system

Step5D: Run Pipeline

Step6: Install Plugins like JDK, Sonarqube Scanner, NodeJs

Step6A: Install Plugin

Step6B: Configure Java and Nodejs in Global Tool Configuration

Step6C: Configure Sonar Server in Manage Jenkins

Step6D: Add New stages to the pipeline

Step7: Install OWASP Dependency Check Plugins

Step8A: Docker Image Build and Push

Step8B: Create an API key from Rapid API

Step8C: Run the Docker container

Step9A: Kubernetes Setup

Step9B: Kubectl is to be installed on Jenkins

Step9C: K8S Master-Slave setup

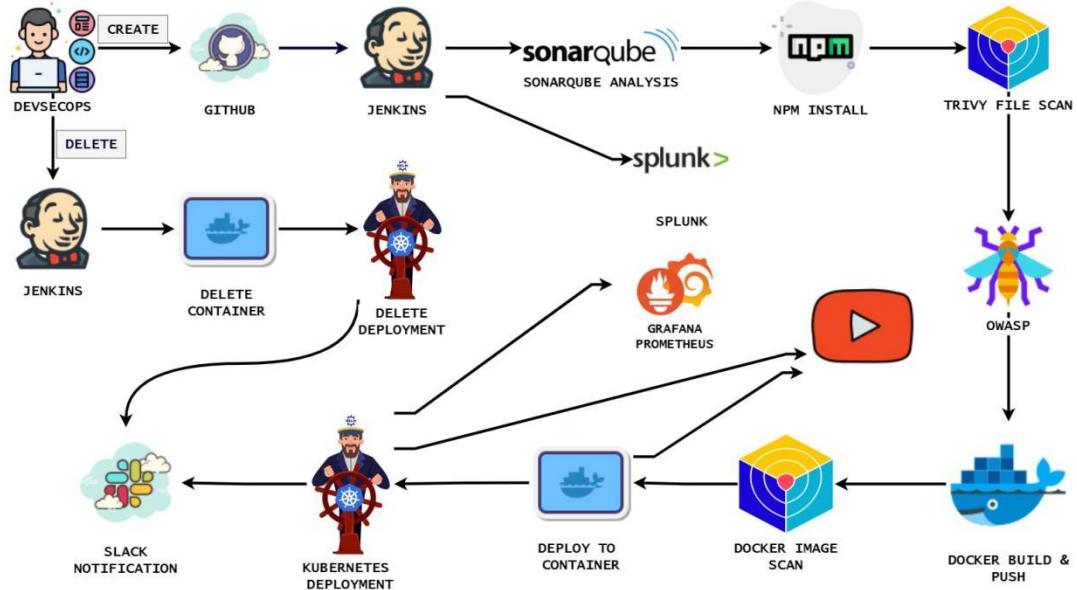
Step9D: Install Helm & Monitoring K8S using Prometheus and Grafana

Install kube-Prometheus-stack

Edit Prometheus Service

Edit Grafana Service

Step9E: K8S Deployment



GitHub Repo: <https://github.com/chandrushal/Youtube-clone-app.git>

GitHub Repo: [https://github.com/chandrushal/Jenkins\\_shared\\_library.git](https://github.com/chandrushal/Jenkins_shared_library.git)

## Step 1: Launch an Ubuntu 22.04 instance for Jenkins

**Choose an Instance Type:** Pick "t2.large."

<input checked="" type="checkbox"/> jenkin	i-0f9abf1aaa451f058	<span>Running</span>	<span>2/2 checks passed</span>	<span>View alarms +</span>	us-east-1d
--	---------------------	----------------------	--------------------------------	----------------------------	------------

Create a shell script to install Jenkins service

`sudo vi install_jenkins.sh`

**Past the following script**

```
#!/bin/bash
sudo apt update -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee
/etc/apt/keyrings/adoptium.asc
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb
$(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | 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 tee
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable
binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update -y
```

```
sudo apt-get install jenkins -y  
sudo systemctl start jenkins  
sudo systemctl status jenkins
```

#### Make the script executable:

```
sudo chmod +x install_jenkins.sh
```

#### Run the script:

```
./install_jenkins.sh
```

The script will install Jenkins and start the Jenkins service.

## Step2A: Install Docker on the Jenkins machine

Run the below commands to install the docker

```
sudo apt-get update  
sudo apt-get install docker.io -y  
sudo usermod -aG docker $USER #my case is ubuntu  
newgrp docker  
sudo chmod 777 /var/run/docker.sock
```

After the docker installation, create a Sonarqube container (add 9000 ports in the security group).

```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

Now copy the IP address of the ec2 instance



Enter username and password, click on login and change password

**username :** admin

**password :** admin

## Step2B: Install Trivy on Jenkins machine

[Create a shell script to install trivy](#)

```
sudo vi trivy.sh
```

```
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 | sudo tee /usr/share/keyrings/trivy.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy -y
```

#### Provide executable permissions and run the shell script

```
sudo chmod +x trivy.sh
```

```
./trivy.sh
```

This will install Trivy on our Jenkins machine.

## Step3A: Launch an Ubuntu instance for Splunk

Step 1: Launch Instances , select T2.medium with 24GB of storage

Create a splunk account in splunk.com

Step 2: Install Splunk

```
wget -O splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb
"https://download.splunk.com/products/splunk/releases/9.1.1/linux/splunk-9.1.1-64e843ea36b1-
linux-2.6-amd64.deb"
```

running the below command, you ensure that SSH access is permitted through your firewall, which is crucial for remote server management and administration.

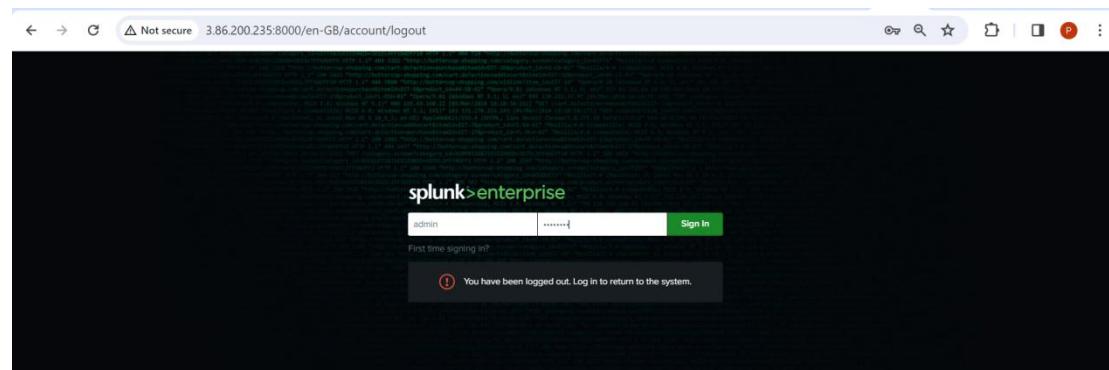
```
sudo ufw allow openSSH
sudo ufw allow 8000
sudo ufw status
sudo ufw enable
```

**sudo /opt/splunk/bin/splunk start**

The command `sudo /opt/splunk/bin/splunk start` is used to start the Splunk Enterprise application on your system. When you run this command with superuser privileges (using `sudo`), it initiates the Splunk service, allowing you to begin using the Splunk platform for data analysis, monitoring, and other data-related tasks.

Access with

<splunk-public-ip:8000>



## Step3B: Install the Splunk app for Jenkins

In Splunk Dashboard .Click on Apps --> Find more apps

The screenshot shows the Splunk enterprise dashboard with the search bar containing 'Jenkins'. The search results show one app: 'Splunk App for Jenkins'. This app card includes a brief description, download count (17164), release date (4 years ago), and update information ('Last Updated: a year ago'). A green 'Install' button is visible at the top right of the card.

provide your Splunk credentials. That's why we created a Splunk account.

The screenshot shows the Splunk enterprise dashboard with the user 'poorni' logged in. The 'Common tasks' section is highlighted with a yellow box. It contains six tasks: 'Add data', 'Search your data', 'Visualize your data', 'Add team members', 'Manage permissions', and 'Configure mobile devices'. Each task has a small icon and a brief description.

Splunk app for jenkins will be installed

In the Splunk web interface, go to Settings > Data Inputs.

The screenshot shows the Splunk enterprise dashboard with the 'Build Analysis' section active. On the right side, a navigation menu is open, showing categories like KNOWLEDGE, DATA, DISTRIBUTED ENVIRONMENT, SYSTEM, and USERS AND AUTHENTICATION. Under 'DATA', 'Data inputs' is selected, which is described as 'Forwarding and receiving Indexes'. Other options include 'Virtual indexes', 'Source types', and 'Ingest actions'.

Click on HTTP Event Collector.

The screenshot shows the Splunk Data Inputs page under the Local inputs section. It lists two types of inputs:

Type	Inputs	Actions
Files & Directories	15	+ Add new
HTTP Event Collector	0	+ Add new

Click on Global Settings

The screenshot shows the 'Edit Global Settings' dialog box. It contains the following configuration options:

- All Tokens tab selected (Enabled)
- Default Source Type: Select Source Type ▾
- Default Index: Default ▾
- Default Output Group: None ▾
- Use Deployment Server:
- Enable SSL:
- HTTP Port Number: 8088

Buttons at the bottom: Cancel and Save.

Generate a token

The screenshot shows the 'HTTP Event Collector' page. It features a 'New Token' button in the top right corner. Below it, there's a search bar and a table header with columns: Name, Actions, Token Value, Source Type, Index, and Status.

A message at the bottom states: 'No tokens found.'

**splunk>enterprise** Apps ▾ poorni ▾ Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

Add Data Select Source Input Settings Review Done < Back Next >

**Files & Directories**  
Upload a file, index a local file, or monitor an entire directory.

**HTTP Event Collector**  
Configure tokens that clients can use to send data over HTTP or HTTPS.

**TCP / UDP**  
Configure the Splunk platform to listen on a network port.

**Scripts**  
Get data from any API, service, or database with a script.

**Splunk Assist Instance Identifier**  
Assigns a random identifier to every node.

**Systemd Journald Input for Splunk**  
This is the input that gets data from journald (systemd's logging component) into Splunk.

**Logd Input for the Splunk platform**  
This input collects data from logd on macOS and sends it to the

Configure a new token for receiving data over HTTP. Learn More [\[?\]](#)

Name

Source name override?

Description?

Output Group (optional)

Enable indexer acknowledgement

**FAQ**

- › What is the HTTP Event Collector?
- › How do I set up the HTTP Event Collector?

Click next and submit

**splunk>enterprise** Apps ▾ poorni ▾ Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

Add Data Select Source Input Settings Review Done < Back Submit >

**Review**

Input Type ..... Token  
 Name ..... jenkins  
 Source name override ..... N/A  
 Description ..... N/A  
 Enable Indexer acknowledgement ..... No  
 Output Group ..... N/A  
 Allowed indexes ..... N/A  
 Default Index ..... default  
 Source Type ..... Automatic  
 App Context ..... splunk\_app\_jenkins

In the Splunk web interface, go to **Settings > Data Inputs**. Click on the HTTP event collector

**Local inputs**

Type	Inputs	Actions
Files & Directories Index a local file or monitor an entire directory.	15	+ Add new
HTTP Event Collector Receive data over HTTP or HTTPS.	1	+ Add new
TCP Listen on a TCP port for incoming data, e.g. syslog.	0	+ Add new

Now copy your token and keep it saved

**splunk>enterprise** Apps ▾ poorni ▾ Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

**HTTP Event Collector**

Data Inputs > HTTP Event Collector Global Settings New Token

1 Tokens App: All ▾ filter 20 per page ▾

Name	Actions	Token Value	Source Type	Index	Status
jenkins	Edit Disable Delete	5086ab5b-9e8e-4566-a4bd-f5a384ff839e	default		Enabled

# Add Splunk Plugin in Jenkins

The screenshot shows the Jenkins Plugins page. In the top navigation bar, there is a Jenkins logo, a search bar with the placeholder "Search (CTRL+K)", a shield icon with the number "1", a Jenkins icon, and a "log out" link. Below the navigation bar, the URL "Dashboard > Manage Jenkins > Plugins" is visible. On the left, a sidebar has four options: "Updates" (disabled), "Available plugins" (selected and highlighted in blue), "Installed plugins", and "Advanced settings". The main content area has a search bar with the text "splunk". A table lists the "Splunk 1.10.1" plugin, which is "Released" and "11 mo ago". The plugin description states: "Splunk plugin for Jenkins provides deep insights into your Jenkins controller and agent infrastructure, job and build details such as console logs, status, artifacts, and an incredibly efficient way to analyze test results." Below the table are three tabs: "Build Notifiers", "Build Reports", and "Other Post-Build Actions". At the bottom right of the table, there is an "Install" button with a gear icon and a "Reinstall" button.

Again Click on Manage Jenkins --> System

Search for Splunk then Check to enable

HTTP input host as SPLUNK PUBLIC IP

HTTP token that you generated in Splunk

Jenkins IP and apply.

The screenshot shows the "Splunk for Jenkins Configuration" page under the "Manage Jenkins > System" path. The title is "Splunk for Jenkins Configuration". There is a checked checkbox labeled "Enable". Below it are fields for "HTTP Input Host" (3.86.200.235) and "HTTP Input Port" (8088). There is also a field for "HTTP Input Token" containing the value "e68a2278-6278-4a97-92a6-7df39abcf9a8". Under "SSL Enabled", there is an unchecked checkbox. Under "Send All Pipeline Console Logs", there is a checked checkbox. Below these is a field for "Jenkins Master Hostname" with the value "3.87.216.58". At the bottom right, there is a "Test Connection" button and a message "Splunk connection verified".

In Splunk machine run this command and test the connection

**sudo ufw allow 8088**

```

aws Services Search [Alt+S]
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun Feb  4 15:26:07 2024 from 18.206.107.27
ubuntu@ip-172-31-87-248:~$ sudo ufw allow 8088
Rule added
Rule added (v6)
ubuntu@ip-172-31-87-248:~$ sudo ufw status
Status: active

To          Action    From
--          ----     ---
OpenSSH      ALLOW     Anywhere
8000        ALLOW     Anywhere
8088        ALLOW     Anywhere
OpenSSH (v6) ALLOW     Anywhere (v6)
8000 (v6)   ALLOW     Anywhere (v6)
8088 (v6)   ALLOW     Anywhere (v6)

```

Restart Both Splunk and Jenkins .Let's Restart our Splunk machine ,Click on Settings --> Servecontrols

The screenshot shows the Splunk Enterprise interface with the 'Build Analysis' tab selected. The page displays Jenkins build logs for a specific job. The search bar at the top contains the IP address '54.163.80.203'. Below the search bar, there are filters for Jenkins Node, Job, Build Parameters, Build, Status, and Time. A timeline below the filters shows the build's progress from 00:00 on Sunday, February 4, 2024, to 08:00. The build log table includes columns for Jenkins Master, Job, Build, StartTime, Jenkins Node, Duration, and Status. The first entry in the table is '54.163.80.203 test 1 2024-02-04 17:14:24 (built-in) 00:00:03.08 ✓'.

## Step4A: Integrate Slack for Notifications

Create a Slack account and create a channel

The screenshot shows the initial step of creating a new Slack workspace. The title is 'devops'. The question asked is 'What's the name of your company or team?'. The input field contains the text 'devops'. Below the input field is a note: 'This will be the name of your Slack workspace – choose something that your team will recognise.' A 'Next' button is visible at the bottom.

The screenshot shows the Slack workspace settings menu. The 'Manage apps' option is highlighted in blue, indicating it is selected. Other options visible include Tools & settings, Customise workspace, Workflow Builder, Analytics, Settings, Workspace settings, Edit workspace details, Administration, Manage members, and Manage workflows.

Search for Jenkins CI and click on it

The screenshot shows the Slack App Directory. The search bar at the top contains the text 'jenkins'. Below the search bar, the 'Installed apps' section is displayed. The Jenkins CI app is listed first, followed by Zenduty, marbot, and BuildPulse. A message below the list states 'No matching installed apps found.' with a note: 'You might want to try using different keywords, checking for typos or adjusting your filters.' There is also a 'Clear filters' button and a link to 'Review app approval settings'.

Click on Add to Slack

← → ⌛ devops-b3m5257.slack.com/apps/A0F7VRFKN-jenkins-ci?tab=more\_info

slack app directory Search App Directory Browse Manage Build devops

< Browse apps

## Jenkins CI

Description Permissions Security & compliance

Jenkins CI is a customisable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.

This integration will post build notifications to a channel in Slack.

Add to Slack

Learn more & Support  
Privacy policy  
Terms

← → ⌛ devops-b3m5257.slack.com/apps/new/A0F7VRFKN-jenkins-ci

slack app directory Search App Directory Browse Manage Build devops

## Jenkins CI

An open-source continuous integration server.

Jenkins CI is a customisable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.

This integration will post build notifications to a channel in Slack.

**Post to channel**

Start by choosing a channel where Jenkins notifications will be posted.

# jenkins

or create a new channel

Add Jenkins CI integration

← → ⌛ devops-b3m5257.slack.com/services/B06H6G943NY?added=1

slack app directory Search App Directory Browse Manage Build devops

**Step 1**  
In your Jenkins dashboard, click on **Manage Jenkins** from the left navigation.



**Step 2**  
Click on **Manage plugins** and search for **Slack notification** in the **Available** tab. Click the tick box and install the plugin.

**Step 2** Click on **Manage plugins** and search for **Slack notification** in the **Available** tab. Click the tick box and install the plugin.

**Step 3** Once it's installed, click on **Manage Jenkins** again in the left navigation and then go to **Configure system**. Find the **Global Slack notifier settings** section and add the following values:

- Team subdomain: `devops-b3m5257`
- Integration token credential ID: Create a secret text credential using `yDcAgehTYU6dkQnIbeNYkof` as the value

Click on Manage Jenkins --> Credentials --> Global

Select kind as Secret Text

At Secret Section Provide Your Slack integration token credential ID

Click on manage Jenkins --> Plugins --> Available plugins ,Search for Slack Notification and install

Plugin Name	Version	Last Updated
WMI Windows Agents	1.8.1	1 yr 9 mo ago
Slack Notification	684.v833089650554	5 mo 26 days ago

Success Test Connection

Click on Apply and save and Add this to the pipeline

```

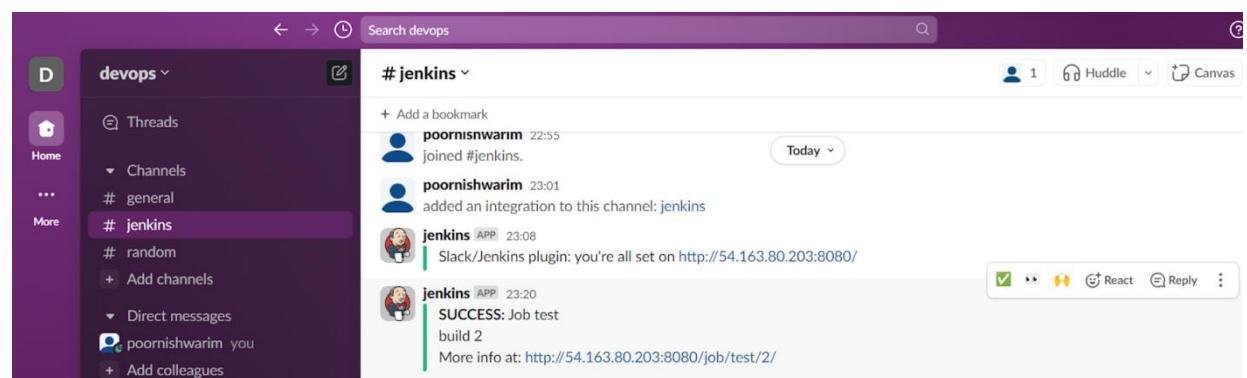
def COLOR_MAP = [
    'FAILURE' : 'danger',
    'SUCCESS' : 'good'
]

]

post {
    always {
        echo 'Slack Notifications'

        slackSend (
            channel: '#channel name', #change your channel name
            color: COLOR_MAP[currentBuild.currentResult],
            message: "*${currentBuild.currentResult}:* Job ${env.JOB_NAME} \n build
${env.BUILD_NUMBER} \n More info at: ${env.BUILD_URL}"
        )
    }
}

```



You will get a Notification in Slack

## **Step5A: Start Job**

Go to Jenkins dashboard and click on New Item. Provide a name for the Job & click on Pipeline.

## **Step5B: Create a Jenkins shared library in GitHub**

Create a directory named Jenkins\_shared\_library

Create a Vars directory inside it

Run the below commands to push to GitHub

```
echo "# Jenkins_shared_library" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
# make sure to change your repo Url here
git remote add origin https://github.com/git_user/Jenkins_shared_library.git
git push -u origin main
```

Create a **cleanWorkspace.groovy** file and add the below code

```
#cleanWorkspace.groovy //cleans workspace
def call() {
    cleanWs()
}
```

Create **checkoutGit.groovy** file and add the below code

```
def call(String gitUrl, String gitBranch) {
    checkout([
        $class: 'GitSCM',
        branches: [[name: gitBranch]],
        userRemoteConfigs: [[url: gitUrl]]
    ])
}
```

Now push them to GitHub using the below commands from vs code

```
git add .
git commit -m "message"
git push origin main
```

## **Step5C: Add Jenkins shared library to Jenkins system**

Go to Jenkins Dashboard

Click on Manage Jenkins --> system

Search for Global Pipeline Libraries and click on Add

GitHub Enterprise Servers

Add

Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Grab.

Add

Now Provide a name that we have to call in our pipeline

Dashboard > Manage Jenkins > System >

Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Grab.

**Library**

Name ?  
Jenkins\_shared\_library

Default version ?  
main

Cannot validate default version until after saving and reconfiguring.

Load implicitly ?

Allow default version to be overridden ?

Include @Library changes in job recent changes ?

Cache fetched versions on controller for quick retrieval ?

Retrieval method  
Maven Central

Save Apply

Dashboard > Manage Jenkins > System >

Git

Project Repository ?  
https://github.com/chandrushal/Jenkins\_shared\_library.git

Credentials ?  
chandrushal/\*\*\*\*\*\*\*\* (github)

+ Add ▾

Behaviours

Discover branches ?

Add ▾

Save Apply

Click apply and save

## Step5D: Run Pipeline

Go to Jenkins Dashboard again & select the job and add the below pipeline

```
@Library('Jenkins_shared_library') _ #name used in jenkins system for library
def COLOR_MAP = [
    'FAILURE' : 'danger',
    'SUCCESS' : 'good'
```

```

]

pipeline{
    agent any
    parameters {
        choice(name: 'action', choices: 'create\ndelete', description: 'Select create or destroy.')
    }
    stages{
        stage('clean workspace'){
            steps{
                cleanWorkspace()
            }
        }
        stage('checkout from Git'){
            steps{
                checkoutGit('https://github.com/chandrushal/Youtube-clone-app.git', 'main')
            }
        }
    }
    post {
        always {
            echo 'Slack Notifications'
            slackSend (
                channel: 'jenkins', #change your channel name
                color: COLOR_MAP[currentBuild.currentResult],
                message: "*${currentBuild.currentResult}*: Job ${env.JOB_NAME} \n build
${env.BUILD_NUMBER} \n More info at: ${env.BUILD_URL}"
            )
        }
    }
}

```

Build with parameters and build

Stage view

**Stage View**



## Step6: Install Plugins like JDK, Sonarqube Scanner, NodeJs

### Step6A: Install Plugin

Goto Manage Jenkins → Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)

3 → NodeJs Plugin (Install Without restart)

The screenshot shows the Jenkins 'Plugins' page. A search bar at the top right contains the text 'nodejs'. Below the search bar, there are three listed plugins:

- Eclipse Temurin installer 1.5
- SonarQube Scanner 2.17.1
- NodeJS 1.6.1

The 'NodeJS 1.6.1' plugin is highlighted with a blue checkmark and has the word 'npm' next to it. To the right of the plugin names, there are release dates: '1 yr 3 mo ago', '26 days ago', and '5 mo 22 days ago'. At the bottom right of the list, there is a note: 'NodeJS Plugin executes NodeJS script as a build step.'

## Step6B: Configure Java and Nodejs in Global Tool Configuration

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

The screenshot shows the Jenkins 'Tools' configuration page. It displays two sections: 'JDK' and 'NodeJS'.

**JDK Configuration:**

- Name: jdk17
- Install automatically: checked
- Install from adoptium.net:
  - Version: jdk-17.0.8.1+1
- Add Installer: button

**NodeJS installations:**

- Add NodeJS: button
- NodeJS Configuration:**
  - Name: node16
  - Install automatically: checked
  - Install from nodejs.org:
    - Version: NodeJS 16.2.0

At the bottom of the page, there are 'Save' and 'Apply' buttons.

Apply and save it.

## Step6C: Configure Sonar Server in Manage Jenkins

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 under the 'Users' tab. In the 'Tokens' section, there is one entry for 'Administrator admin'. The token value is 'sonar-administrators'. The 'Groups' column shows 'sonar-administrators' and 'sonar-users'. The 'Tokens' column shows '0'.

copy Token

Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text.

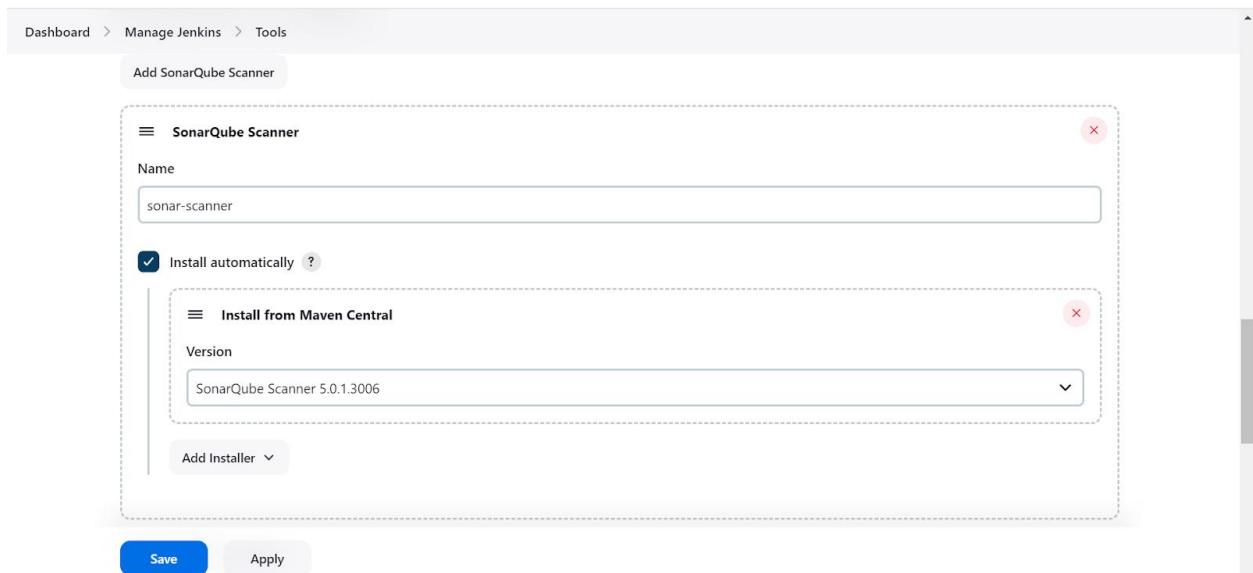
The screenshot shows the Jenkins System Configuration page under the 'System' tab. It displays a 'List of SonarQube installations' with one entry. The 'Name' field is set to 'sonar-server'. The 'Server URL' field contains 'http://52.90.9.174:9000'. The 'Server authentication token' field contains 'sonar-token'. There is also an 'Advanced' dropdown menu.

Click on Apply and Save

**The Configure System option** is used in Jenkins to configure different server

**Global Tool Configuration** is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.



In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks

Click on Create

## Step6D: Add New stages to the pipeline

Go to vs code and create a file **sonarqubeAnalysis.groovy** & add the below code and push to Jenkins shared library GitHub Repo.

```

def call() {
    withSonarQubeEnv('sonar-server') {
        sh "" $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Youtube -Dsonar.projectKey=Youtube ""
    }
}

```

Create another file for **qualityGate.groovy**

```

def call(credentialsId) {
    waitForQualityGate abortPipeline: false, credentialsId: credentialsId
}

```

Create another file for **npmInstall.groovy**

```

def call() {
    sh 'npm install'
}

```

Push them to the GitHub Jenkins shared library

```

git add .
git commit -m "message"
git push origin main

```

Add these stages to the pipeline now

```

#under parameters
tools{
    jdk 'jdk17'
    nodejs 'node16'
}
environment {
    SCANNER_HOME=tool 'sonar-scanner'
}

# add in stages
stage('sonarqube Analysis'){
    when { expression { params.action == 'create' } }
    steps{
        sonarqubeAnalysis()
    }
}
stage('sonarqube QualitGate'){
    when { expression { params.action == 'create' } }
    steps{
        script{
            def credentialsId = 'Sonar-token'
            qualityGate(credentialsId)
        }
    }
}
stage('Npm'){
    when { expression { params.action == 'create' } }
    steps{
        npmInstall()
    }
}

```

}

Build now.

Stage view



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

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

My Favorites All

Search for projects...

1 project(s)

Perspective: Overall Status Sort by: Name

**Youtube** Passed

Last analysis: 2 minutes ago PRIVATE

Bugs: 0 A | Vulnerabilities: 0 A | Hotspots Reviewed: 0.0% E | Code Smells: 1 A | Coverage: 0.0% O | Duplications: 0.0% G | Lines: 549 xs JavaScript...

## Step7: Install OWASP Dependency Check Plugins

GotoDashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install it without restart.

Plugins

Search: owasp

Available plugins

OWASP Dependency-Check 5.4.3

Released 4 mo 28 days ago

Description: This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.

Goto Dashboard → Manage Jenkins → Tools →

Dependency-Check installations

Add Dependency-Check

**Dependency-Check**

Name: DP-Check

Install automatically: checked

Install from github.com

Version: dependency-check 6.5.1

Add Installer

Save Apply

Click on Apply and Save here.

Create a file for **trivyFs.groovy**

```
def call() {
    sh 'trivy fs . > trivyfs.txt'
}
```

Push to GitHub

```
git add .
git commit -m "message"
git push origin main
```

Add the below stages to the Jenkins pipeline

```
stage('Trivy file scan'){
    when { expression { params.action == 'create' } }
    steps{
        trivyFs()
    }
}
stage('OWASP FS SCAN'){
when { expression { params.action == 'create' } }
steps {
    dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit',
    odcInstallation: 'DP-Check'
    dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
}
}
```

## Step8A: Docker Image Build and Push

We need to install the Docker tool in our system, Goto Dashboard → Manage Jenkins → Available plugins → Search for Docker and install these plugins

The screenshot shows the Jenkins management interface for available plugins. The search bar at the top contains the text 'docker'. Below the search bar, there are two tabs: 'Installed plugins' (which is currently selected) and 'Available plugins'. A large list of Docker-related plugins is displayed, each with a checkmark icon, the plugin name, its version, a brief description, and the date it was last updated. The plugins listed are:

- Docker Commons** 439.va\_3cb\_0a\_6a\_fb\_29  
Provides the common shared functionality for various Docker-related plugins.  
Last updated 6 mo 29 days ago.
- Docker Pipeline** 572.v950f58993843  
Build and use Docker containers from pipelines.  
Last updated 5 mo 27 days ago.
- Docker API** 3.3.4-86.v39b\_a\_5ede342c  
This plugin provides docker-java API for other plugins.  
Last updated 2 mo 8 days ago.  
A yellow box highlights this plugin with the text: "This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information."
- docker-build-step** 2.11  
This plugin allows to add various docker commands to your job as build steps.  
Last updated 1 mo 1 day ago.

Now, goto Dashboard → Manage Jenkins → Tools

The screenshot shows the Jenkins 'Docker installations' configuration page. At the top, there is a breadcrumb navigation: Dashboard > Manage Jenkins > Tools. Below the navigation, the title 'Docker installations' is displayed. A button labeled 'Add Docker' is present. The main configuration area is titled 'Docker' and contains the following fields:

- Name:** docker
- Install automatically:**  (with a question mark icon)
- Download from docker.com:**  (with a question mark icon)
- Docker version:** latest
- Add Installer:** A dropdown menu.

At the bottom of the configuration area are two buttons: 'Save' (highlighted in blue) and 'Apply'.

Add DockerHub Username and Password under Global Credentials

## Step8B: Create an API key from Rapid API

Open a new tab in the browser and search for [rapidapi.com](https://rapidapi.com)

It will automatically provide your mail and select a mail to create an account

The screenshot shows a web browser window with the URL 'rapidapi.com/hub' in the address bar. The page displays the 'API Hub' section of the RapidAPI website. On the left, there is a sidebar with the text 'Welcome' and 'Discover'. The main content area features a large blue banner with the text 'Find API Hub to thousands of APIs'. To the right of the banner, there is a search bar containing the query 'youtubev3'. Below the search bar, a list of search results is shown, including:

- Youtubev3
- Youtubev3 - altern...
- YouTube v3
- YouTube Search an...
- YouTube Scraper 2...
- Youtube V2

At the bottom of the search results, there is a link 'View all results (339)' and a filter section with 'PUBLIC APIs' and 'PRIVATE APIs' options.

Copy API and use it in the groovy file

```
docker build --build-arg REACT_APP_RAPID_API_KEY=<API-KEY> -t ${imageName} .
```

The screenshot shows the RapidAPI interface for the YouTube v3 API. At the top, there's a search bar and navigation links for API Hub, Organizations, Apps, My APIs, and account settings. The main card displays the API name "Youtube v3" with a "FREEMIUM" badge, created by "ytdlfree" 24 days ago. It shows metrics: Popularity (9.9 / 10), Latency (327ms), Service Level (100%), and Health Check (100%). Below the card are tabs for Endpoints, About, Tutorials, Discussions, and Pricing, with "Subscribed" checked. A note says "Get youtube data without any youtube data api key" with a "Show more..." link. The "Endpoints" tab is active, showing V1 (Current) selected. A search bar and a "Test Endpoint" button are present. On the left, a sidebar lists endpoints: GET Activities, GET Captions, GET Channels, and GET ChannelSections. The "GET Captions" endpoint is detailed, describing it as returning a list of channel activity events. To the right, sections for Code Snippets (Node.js Axios example), Example Responses, and Results are shown.

### Create a shared library file for `dockerBuild.groovy`

```
def call(String dockerHubUsername, String imageName) {
    // Build the Docker image
    sh "docker build --build-arg REACT_APP_RAPID_API_KEY=f0ead79813mshb0aa -t ${imageName} ."
    // Tag the Docker image
    sh "docker tag ${imageName} ${dockerHubUsername}/${imageName}:latest"
    // Push the Docker image
    withDockerRegistry([url: 'https://index.docker.io/v1/', credentialsId: 'docker']) {
        sh "docker push ${dockerHubUsername}/${imageName}:latest"
    }
}
```

### Create another file for `trivyImage.groovy`

```
def call() {
    sh 'trivy image sevenajay/youtube:latest > trivyimage.txt'
```

Push the above files to the GitHub shared library.

```
git add .
git commit -m "message"
git push origin main
```

Add this stage to your pipeline with parameters

```
#add inside parameter
string(name: 'DOCKER_HUB_USERNAME', defaultValue: 'sevenajay', description: 'Docker Hub Username')
string(name: 'IMAGE_NAME', defaultValue: 'youtube', description: 'Docker Image Name')
#stage
stage('Docker Build'){
    when { expression { params.action == 'create' } }
    steps{
        script{
```

```

def dockerHubUsername = params.DOCKER_HUB_USERNAME
def imageName = params.IMAGE_NAME

    dockerBuild(dockerHubUsername, imageName)
}
}
}
stage('Trivy iamge'){
when { expression { params.action == 'create'}}
steps{
    trivyImage()
}
}

```

Build now with parameters

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

## Step8C: Run the Docker container

Create a new file **runContainer.groovy**

```

def call(){
    sh "docker run -d --name youtube1 -p 3000:3000 sevenajay/youtube:latest"
}

```

Create Another file to remove container **removeContainer.groovy**

```

def call(){
    sh 'docker stop youtube1'
    sh 'docker rm youtube1'
}

```

Push them to the Shared library GitHub repo

```

git add .
git commit -m "message"
git push origin main

```

Add the below stages to the Pipeline

```
stage('Run container'){
    when { expression { params.action == 'create' } }
    steps{
        runContainer()
    }
}
stage('Remove container'){
when { expression { params.action == 'delete' } }
steps{
    removeContainer()
}
}
```

Build with parameters "create"

The screenshot shows the Jenkins Pipeline configuration for the 'Pipeline youtube' job. On the left, there's a sidebar with various options like Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Splunk, SonarQube, Rename, and Pipeline Syntax. The main area is titled 'Pipeline youtube' and shows a single build step. The 'action' parameter is set to 'create'. The 'DOCKER\_HUB\_USERNAME' parameter is set to 'shalinichandru'. The 'IMAGE\_NAME' parameter is set to 'youtube'. At the bottom, there are buttons for 'Build' and 'Cancel', along with links for 'Atom feed for all' and 'Atom feed for failures'.

Stage view

Declarative: Tool Install	clean workspace	checkout from Git	sonarqube Analysis	sonarqube QualitGate	Npm	Trivy file scan	OWASP FS SCAN	Docker Build	Trivy iamge	Run container	Remove container	Declarative: Post Actions
364ms	691ms	1s	28s	474ms	24s	3s	7min 37s	2min 38s	4min 23s	1s	0ms	427ms
364ms	691ms	1s	28s	474ms	24s	3s	7min 37s	2min 38s	4min 23s	1s		427ms

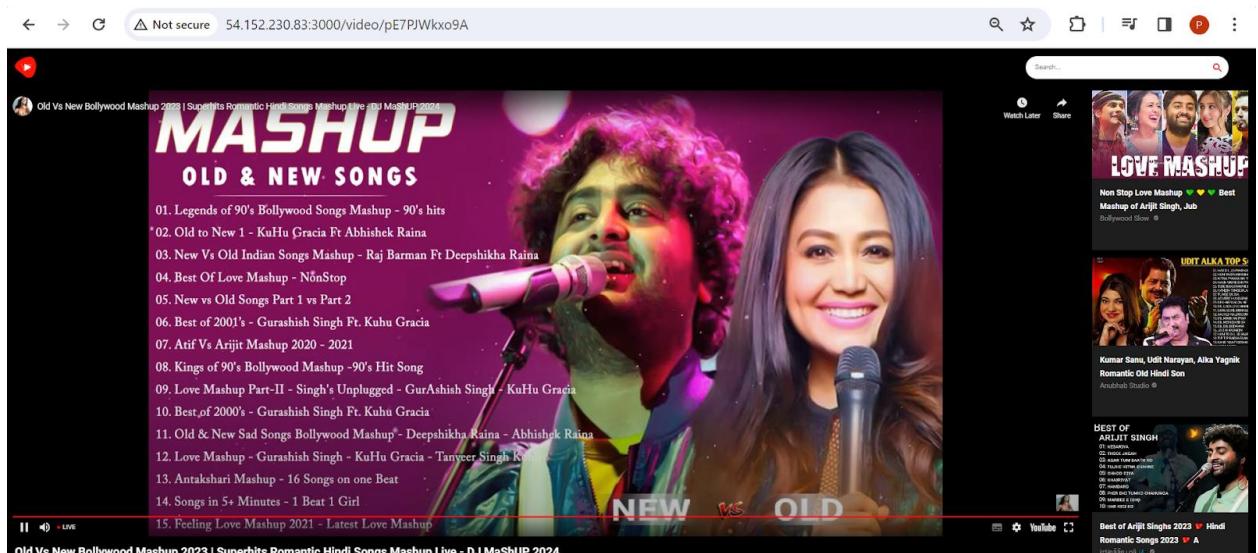
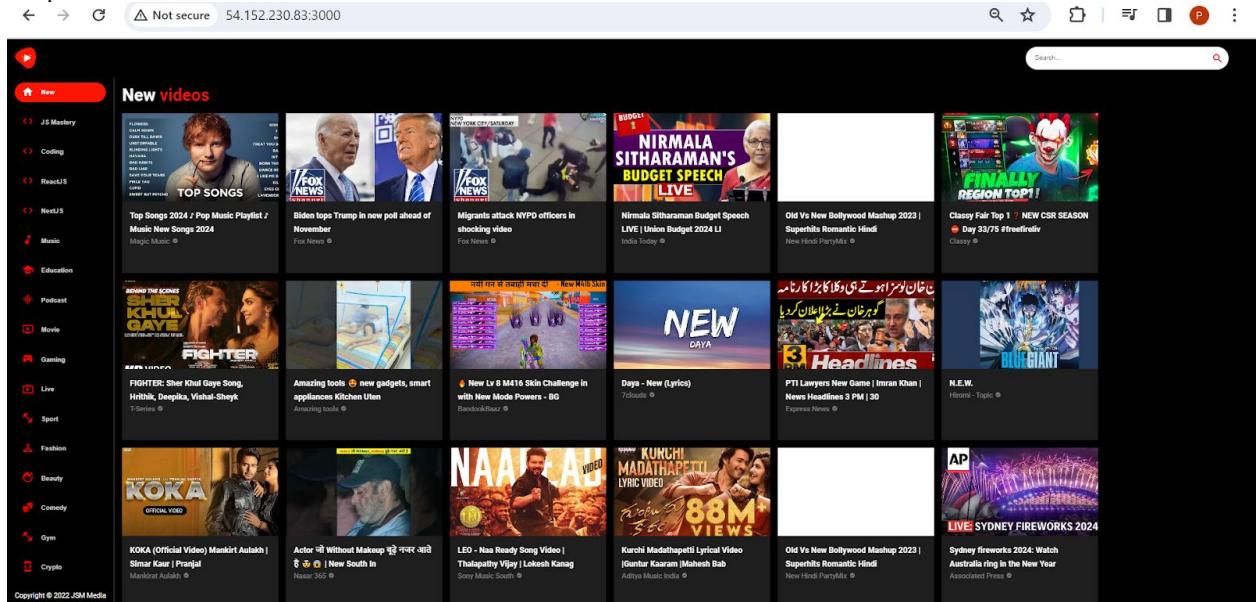
#### Dependency-Check Results

SEVERITY DISTRIBUTION				Search
File Name	Vulnerability	Severity	Weakness	
axios.js	NVD CVE-2023-45857	Medium	CWE-352	
axios.min.js	NVD CVE-2023-45857	Medium	CWE-352	
axios@2.7.2	OSINDEX CVE-2023-45857	Medium	CWE-352	
css-what:3.4.2	OSINDEX CVE-2022-21222	High	CWE-1333	
ejs@3.1.8	OSINDEX CVE-2023-29827	High	CWE-74	
follow-redirects@1.15.1	NVD CVE-2023-26159	Medium	CWE-601	
json5@1.0.1	NVD CVE-2022-46175	High	CWE-1321	
json5@2.2.1	NVD CVE-2022-46175	High	CWE-1321	
jsonpointer@5.0.1	NVD CVE-2022-4742	Critical	CWE-1321	
loader-utils@2.0.2	NVD CVE-2022-37599	High	CWE-1333	

It will start the container

<public-ip-jenkins:3000>

Output:



Build with parameters 'delete'

Dashboard > youtube >

**Pipeline youtube**

This build requires parameters:

**action**  
Select create or destroy.

**DOCKER\_HUB\_USERNAME**  
Docker Hub Username

**IMAGE\_NAME**  
Docker Image Name

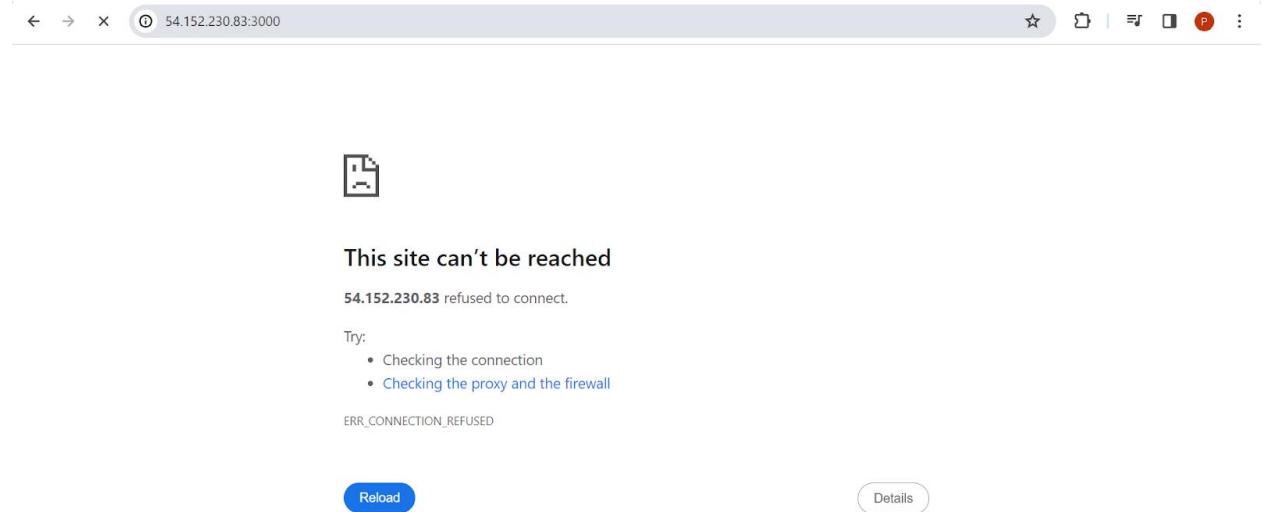
**Build History** trend ▾ **Build** Cancel

Atom feed for all Atom feed for failures

It will stop and remove the Container

Declarative: Tool Install	clean workspace	checkout from Git	sonarqube Analysis	sonarqube QualitGate	Npm	Trivy file scan	OWASP FS SCAN	Docker Build	Trivy iamge	Run container	Remove container	Declarative: Post Actions
294ms	516ms	939ms	28s	474ms	24s	3s	7min 37s	2min 38s	4min 23s	1s	1s	415ms
224ms	342ms	783ms									1s	403ms

Skipped some of the stages due to delete parameter



## Step9A: Kubernetes Setup

Take-Two Ubuntu 20.04(t2.medium) instances one for k8s master and the other one for worker.

Install Kubectl on Jenkins machine also.

## Step9B: Kubectl is to be installed on Jenkins

Connect your Jenkins machine

Create a shell script file kube.sh

**sudo vi kube.sh**

Paste the below commands

```
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)/bin/linux/amd64/kubectl
```

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
kubectl version --client
```

## Step9C: K8S Master-Slave setup

### Part 1 -----Master Node-----

```
sudo hostnamectl set-hostname K8s-Master
```

### -----Worker Node-----

```
sudo hostnamectl set-hostname K8s-Worker
```

### Part 2 -----Both Master & Node -----

```
sudo apt-get update
```

```
sudo apt-get install -y docker.io  
sudo usermod -aG docker Ubuntu  
newgrp docker  
sudo chmod 777 /var/run/docker.sock
```

```
sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

```
sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF  
deb https://apt.kubernetes.io/ kubernetes-xenial main  
EOF
```

```
sudo apt-get update
```

```
sudo apt-get install -y kubelet kubeadm kubectl
```

```
sudo snap install kube-apiserver
```

### Part 3 ----- Master -----

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16  
# in case you're in root exit from it and run below commands  
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config  
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

```
[addons] Applied essential addon: kube-proxy  
  
Your Kubernetes control-plane has initialized successfully!  
  
To start using your cluster, you need to run the following as a regular user:  
  
    mkdir -p $HOME/.kube  
    sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
    sudo chown $(id -u):$(id -g) $HOME/.kube/config  
  
Alternatively, if you are the root user, you can run:  
  
    export KUBECONFIG=/etc/kubernetes/admin.conf  
  
You should now deploy a pod network to the cluster.  
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:  
    https://kubernetes.io/docs/concepts/cluster-administration/addons/  
  
Then you can join any number of worker nodes by running the following on each as root:  
  
kubeadm join 172.31.81.6:6443 --token ncxrog.od0ao7t7d4mulj82 \  
    --discovery-token-ca-cert-hash sha256:15bab2a7e6eaeb701a61c15aa98cf31a78258687dc0c50a65316662dec176535
```

## -----Worker Node-----

```
sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discovery-token-ca-cert-hash <hash>
```

```
ubuntu@K8s-slave:~$ sudo kubeadm join 172.31.81.6:6443 --token ncxrog.od0ao7t7d4mulj82 \
>           --discovery-token-ca-cert-hash sha256:15bab2a7e6aeab701a61c15aa98cf31a78258687dc0c50a65316662dec17653
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

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

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,

The screenshot shows the Jenkins Plugins page with a search bar containing 'kuber'. Below the search bar, there are four listed plugins:

- Kubernetes Client API** (version 6.10.0) - Released 10 days ago. Description: Kubernetes Client API plugin for use by other Jenkins plugins.
- Kubernetes Credentials** (version 0.11) - Released 5 months 10 days ago. Description: Common classes for Kubernetes credentials.
- Kubernetes** (version 4.186) - Released 7 days 21 hours ago. Description: This plugin integrates Jenkins with Kubernetes.
- Kubernetes CLI** (version 1.12.1) - Released 5 months 10 days ago. Description: Kubernetes Client API plugin for use by other Jenkins plugins.

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

The screenshot shows the 'New credentials' form in Jenkins. The fields are filled as follows:

- Kind:** Secret file
- Scope:** Global (Jenkins, nodes, items, all child items, etc.)
- File:** Choose File Secret File.txt
- ID:** k8s
- Description:** k8s

## Step9D: Install Helm & Monitoring K8S using Prometheus and Grafana

install the helm in Kubernetes Master

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
```

**See the Helm version:**  
helm version --client

**We need to add the Helm Stable Charts for your local client. Execute the below command:**

```
helm repo add stable https://charts.helm.sh/stable
```

**Add Prometheus Helm repo:**

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

**Create Prometheus namespace:**

```
kubectl create namespace prometheus
```

## Install kube-Prometheus-stack

Below is the command to install kube-prometheus-stack. The helm repo kube-stack-Prometheus (formerly Prometheus-operator) comes with a Grafana deployment embedded.

```
helm install stable prometheus-community/kube-prometheus-stack -n prometheus
```

Let's check if the Prometheus and Grafana pods are running or not

```
kubectl get pods -n prometheus
```

Now See the services

```
kubectl get svc -n prometheus
```

```
ubuntu@K8s-Master:~$ kubectl get nodes
NAME      STATUS    ROLES     AGE      VERSION
k8s-master Ready     control-plane   2m37s   v1.28.2
k8s-slave Ready     <none>    19s     v1.28.2
ubuntu@K8s-Master:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@K8s-Master:~$ chmod 700 get_helm.sh
ubuntu@K8s-Master:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.14.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ubuntu@K8s-Master:~$ helm version --client
version.BuildInfo{Version:"v3.14.0", GitCommit:"3fc9f4b2638e76f26739cd77c7017139be81d0ea", GitTreeState:"clean", GoVersion:"go1.21.5"}
ubuntu@K8s-Master:~$ 
ubuntu@K8s-Master:~$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
ubuntu@K8s-Master:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
ubuntu@K8s-Master:~$ kubectl create namespace prometheus
namespace/prometheus created
ubuntu@K8s-Master:~$ helm install stable prometheus-community/kube-prometheus-stack -n prometheus
NAME: stable
LAST DEPLOYED: Tue Feb 13 10:46:01 2024
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace prometheus get pods -l "release=stable"
Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
```

```
ubuntu@K8s-Master:~$ kubectl get pods -n prometheus
NAME                               READY   STATUS    RESTARTS   AGE
alertmanager-stable-kube-prometheus-sta-alertmanager-0   2/2     Running   0          24s
prometheus-stable-kube-prometheus-sta-prometheus-0       1/2     Running   0          22s
stable-grafana-6889bcb489-srdtc                         2/3     Running   0          33s
stable-kube-prometheus-sta-operator-c548659cd-tj6dg        1/1     Running   0          33s
stable-kube-state-metrics-7fdbdb97cb-qqm54               1/1     Running   0          33s
stable-prometheus-node-exporter-fhthc                     1/1     Running   0          33s
stable-prometheus-node-exporter-qpgqg                     1/1     Running   0          33s
ubuntu@K8s-Master:~$ kubectl get svc -n prometheus
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
alertmanager-operated   ClusterIP   None          <none>        9093/TCP,9094/TCP,9094/UDP   28s
prometheus-operated   ClusterIP   None          <none>        9090/TCP          26s
stable-grafana        ClusterIP   10.102.174.220 <none>        80/TCP           37s
stable-kube-prometheus-sta-alertmanager   ClusterIP   10.110.121.0  <none>        9093/TCP,8080/TCP          37s
stable-kube-prometheus-sta-operator        ClusterIP   10.109.78.45   <none>        443/TCP          37s
stable-kube-prometheus-sta-prometheus    ClusterIP   10.108.127.30 <none>        9090/TCP,8080/TCP          37s
stable-kube-state-metrics                ClusterIP   10.102.180.145 <none>        8080/TCP          37s
stable-prometheus-node-exporter         ClusterIP   10.106.61.84   <none>        9100/TCP          37s
```

This confirms that Prometheus and grafana have been installed successfully using Helm.

To make Prometheus and grafana available outside the cluster, use LoadBalancer or NodePort instead of ClusterIP.

## Edit Prometheus Service

```
kubectl edit svc stable-kube-prometheus-sta-prometheus -n prometheus
```

```
ipFamilyPolicy: SingleStack
ports:
- name: http-web
  nodePort: 32548
  port: 9090
  protocol: TCP
  targetPort: 9090
- appProtocol: http
  name: reloader-web
  nodePort: 32558
  port: 8080
  protocol: TCP
  targetPort: reloader-web
selector:
  app.kubernetes.io/name: prometheus
  operator.prometheus.io/name: stable-kube-prometheus-sta-prometheus
sessionAffinity: None
type: LoadBalancer
status:
loadBalancer:
  ingress:
  - hostname: a785b472f5aaa413ea0c5e1576444326-1069929702.us-east-1.elb.amazonaws.com
```

58,1

Bot

## Edit Grafana Service

```
kubectl edit svc stable-grafana -n prometheus
```

```
name: stable-grafana
namespace: prometheus
resourceVersion: "6259"
uid: 97295be-5d44-4ab5-b96b-c21d0bd5ecfb
spec:
  clusterIP: 10.100.206.88
  clusterIPs:
  - 10.100.206.88
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http-web
    port: 80
    protocol: TCP
    targetPort: 3000
  selector:
    app.kubernetes.io/instance: stable
    app.kubernetes.io/name: grafana
  sessionAffinity: None
  type: LoadBalancer
```

```
kubectl get svc -n prometheus
```

Verify if the service is changed to LoadBalancer and also get the Load BalancerPorts OR NodePort.

```
ubuntu@ip-172-31-42-150:~$ kubectl get svc -n prometheus
NAME           AGE      TYPE        CLUSTER-IP   EXTERNAL-IP
S)             AGE      ClusterIP  None         <none>
alertmanager-operated   6m20s    ClusterIP  None         <none>
TCP, 9094/TCP, 9094/UDP 6m20s    ClusterIP  None         <none>
prometheus-operated    6m20s    ClusterIP  None         <none>
TCP, 9090/TCP          6m20s    LoadBalancer 10.100.206.88 a972955be5d444ab5b96bc21d0bd5ecf-1946628392.us-east-1.elb.amazonaws.com
stable-grafana          6m27s    ClusterIP  10.100.33.74 <none>
379/TCP                6m27s    ClusterIP  10.100.145.137 <none>
stable-kube-prometheus-sta-alertmanager 6m27s    ClusterIP  10.100.146.3 a785b472f5aaa413ea0c5e1576444326-1069929702.us-east-1.elb.amazonaws.com
TCP, 8080/TCP          6m27s    LoadBalancer 10.100.112.226 <none>
stable-kube-state-metrics 6m27s    ClusterIP  10.100.114.191 <none>
```

## Access Grafana UI in the browser

Get the external IP from the above screenshot and put it in the browser

Login to Grafana

UserName: admin

Password: prom-operator

## For creating a dashboard to monitor the cluster:

Click the '+' button on the left panel and select 'Import'.

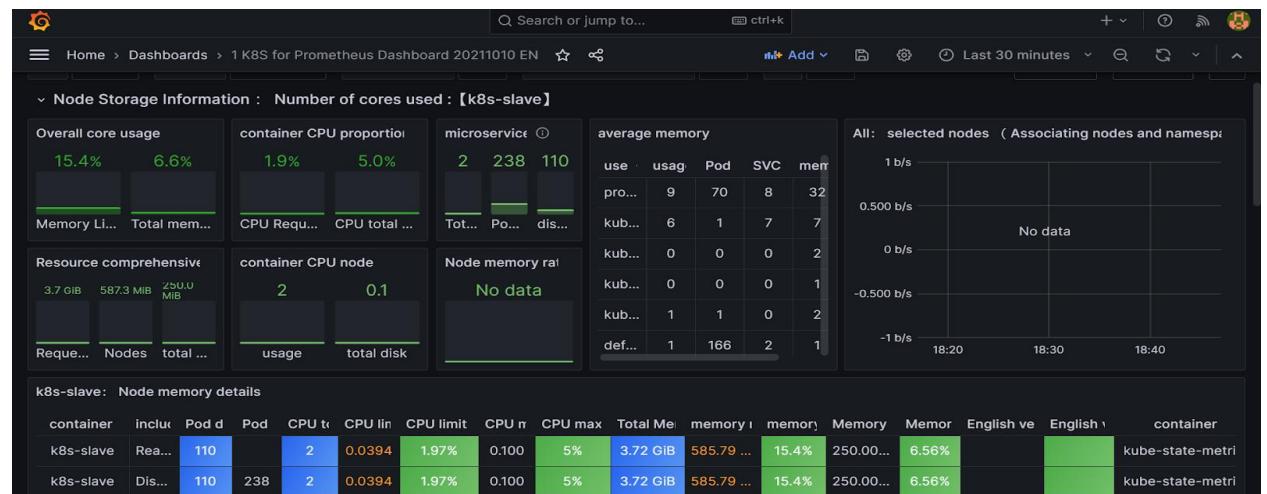
Enter the 15661 dashboard id under [Grafana.com Dashboard](https://grafana.com/Dashboard). Click 'Load'.

The screenshot shows the 'Import dashboard' screen in Grafana. On the left, there's a sidebar with 'Dashboards', 'Playlists', 'Snapshots', 'Library panels', and 'Public dashboards'. The main area has a title 'Import dashboard' and a sub-instruction 'Import dashboard from file or Grafana.com'. Below this is a large input field with a placeholder 'Upload dashboard JSON file' and a note 'Drag and drop here or click to browse Accepted file types: .json, .txt'. Underneath is a search bar with '15661' and a blue 'Load' button. At the bottom, there's a code editor showing a JSON snippet for the dashboard's configuration.

This screenshot shows the 'Importing dashboard from Grafana.com' screen. It displays information about the imported dashboard, such as 'Published by leospbru' and 'Updated on 2022-03-23 20:01:17'. Below this is an 'Options' section with fields for 'Name' (set to 'K8S for Prometheus Dashboard 20211010 EN'), 'Folder' (set to 'Dashboards'), and 'Unique identifier (UID)' (set to 'PwMjtJdvnz'). There's also a 'Prometheus' dropdown set to 'Prometheus'. At the bottom are 'Import' and 'Cancel' buttons.

Click 'Import'.

This will show the monitoring dashboard for all cluster nodes



## How to Create Kubernetes Cluster Monitoring Dashboard?

For creating a dashboard to monitor the cluster:

Click the '+' button on the left panel and select 'Import'.

Enter 3119 dashboard ID under [Grafana.com](#) Dashboard.

Click 'Load'.

Select 'Prometheus' as the endpoint under the Prometheus data sources drop-down.

Click 'Import'.

## Create a POD Monitoring Dashboard

For creating a dashboard to monitor the cluster:

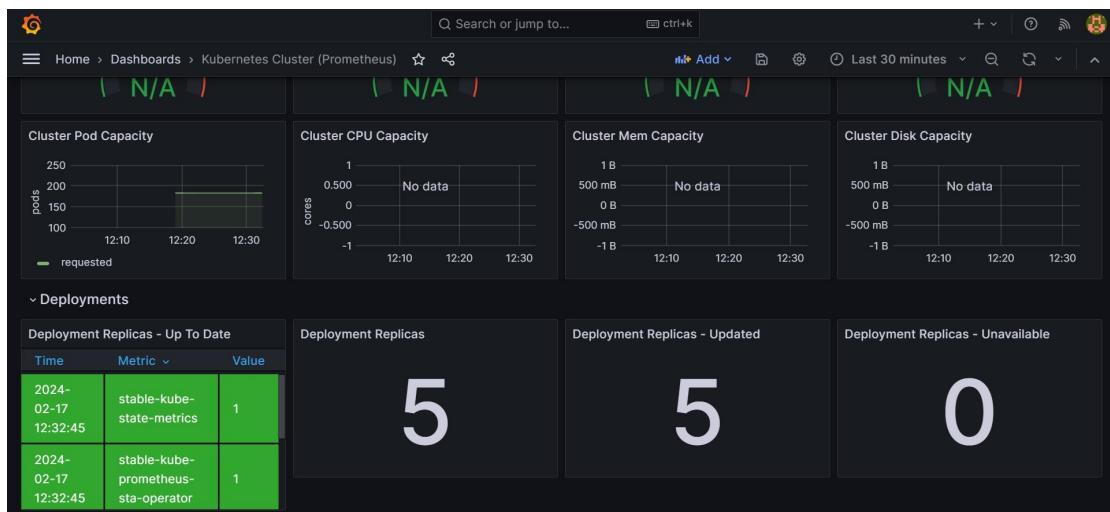
Click the '+' button on the left panel and select 'Import'.

Enter 6417 dashboard ID under [Grafana.com](#) Dashboard.

Click 'Load'.

Select 'Prometheus' as the endpoint under the Prometheus data sources drop-down.

Click 'Import'.



## Step9E: K8S Deployment

Let's Create a Shared Jenkins library file for K8s deploy and delete

Name **kubeDeploy.groovy**

```
def call() {
```

```
    withKubeConfig(caCertificate: "", clusterName: "", contextName: "", credentialsId: 'k8s', namespace: "", restrictKubeConfigAccess: false, serverUrl: "") {
```

```
        sh "kubectl apply -f deployment.yml"
```

```
}
```

```
}
```

### To delete deployment

Name **kubeDelete.groovy**

```
def call() {
```

```
    withKubeConfig(caCertificate: "", clusterName: "", contextName: "", credentialsId: 'k8s', namespace: "", restrictKubeConfigAccess: false, serverUrl: "") {
```

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

```
}
```

```
}
```

### Let's push them to GitHub

```
git add .
```

```
git commit -m "message"
```

```
git push origin main
```

### The final stage of the Pipeline

```
stage('Kube deploy') {
```

```
    when { expression { params.action == 'create' } }
```

```
    steps {
```

```
        kubeDeploy()
```

```
}
```

```
}
```

```
stage('kube deleter') {
```

```
    when { expression { params.action == 'delete' } }
```

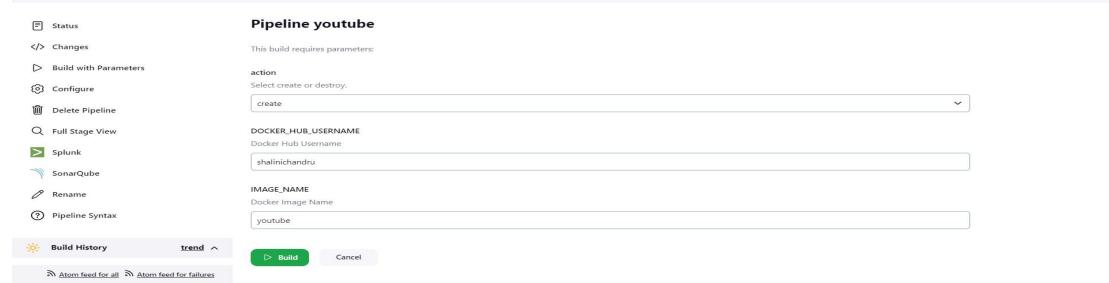
```
    steps {
```

```
        kubeDelete()
```

```
}
```

```
}
```

## Build Now with parameters 'create'



The screenshot shows the Jenkins Pipeline configuration interface for a pipeline named 'youtube'. The 'action' dropdown is set to 'create'. The 'DOCKER.HUB\_USERNAME' field contains 'shalinichandru' and the 'IMAGE NAME' field contains 'youtube'. At the bottom, there is a 'Build' button.

It will apply the deployment

kubectl get all (or)

kubectl get svc

<kubernetes-worker-ip:svc port>

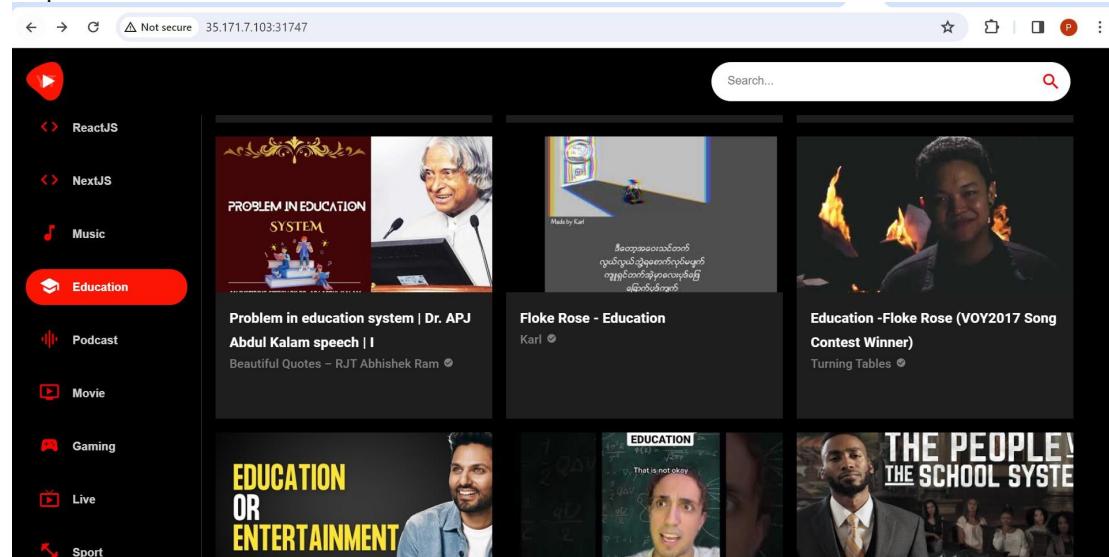
```
NAME           READY   STATUS    RESTARTS   AGE
pod/my-app-deployment-97f7c58bc-kx5dk   0/1     Error      0          15m
pod/my-app-deployment-97f7c58bc-t5zxx   1/1     Running     0          8m37s
pod/my-app-deployment-97f7c58bc-wbdfk   0/1     Error      0          14m

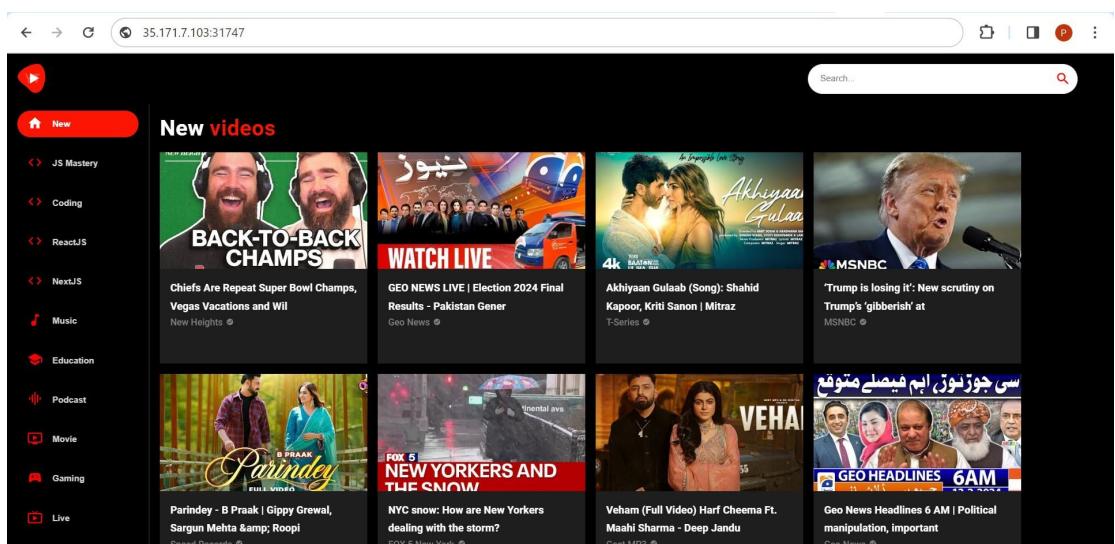
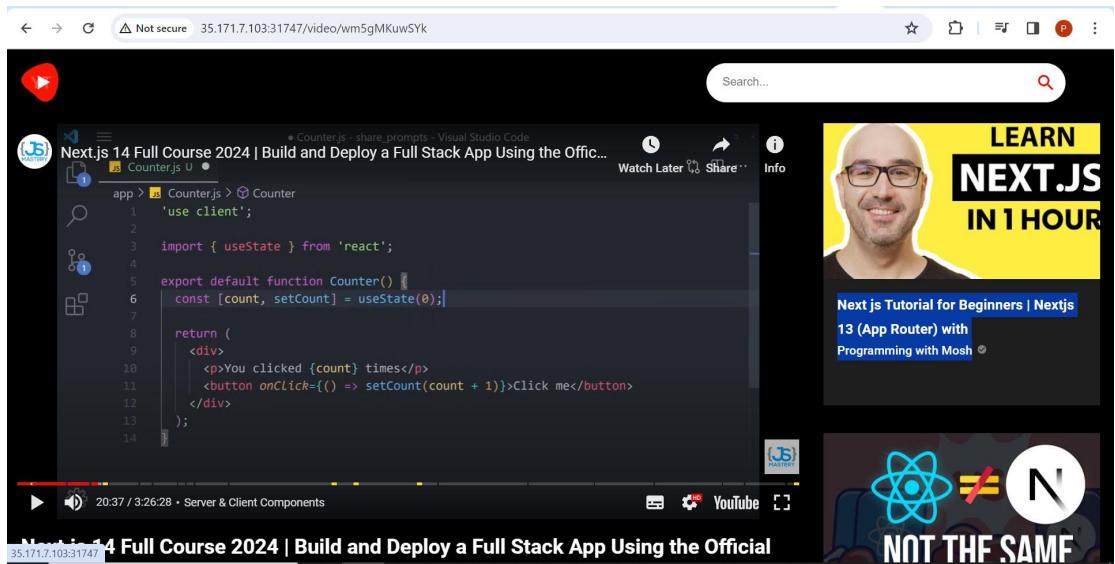
NAME            TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
service/kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP       3d20h
service/my-app-service   NodePort    10.97.215.183   <none>        80:31747/TCP   15m

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-app-deployment   1/1     1           1          15m

NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/my-app-deployment-97f7c58bc   1         1         1      15m
ubuntu@K8s-Master:~$
```

output:





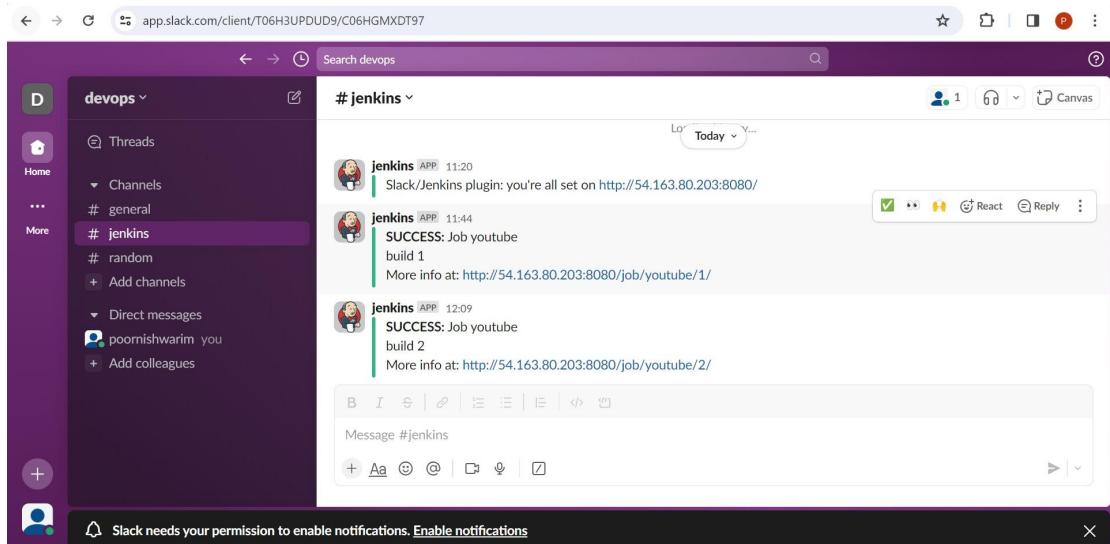
Build with parameter 'delete'

It will destroy Container and Kubernetes deployment.

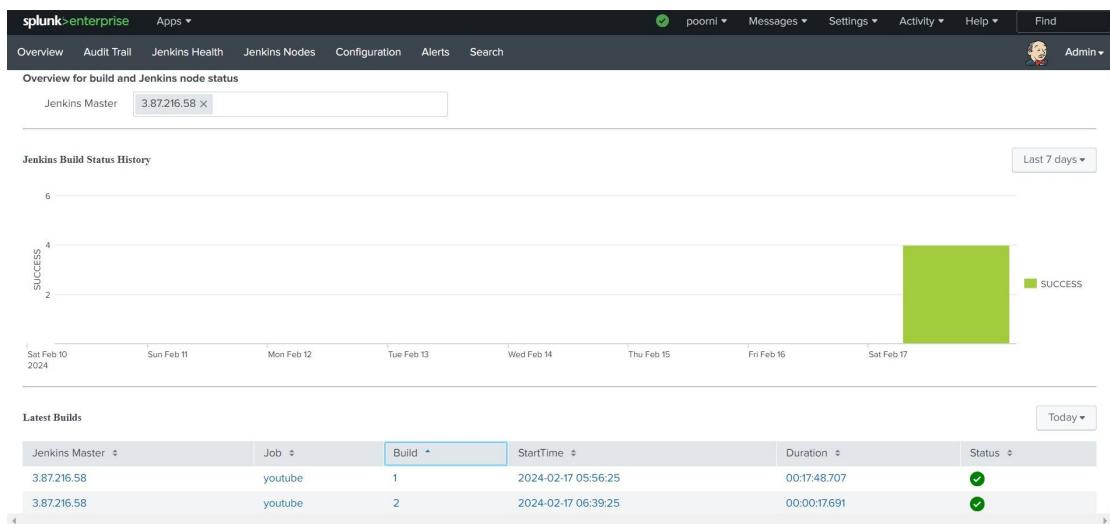
Stage view

Declarative: Tool Install	clean workspace	checkout from Git	sonarqube Analysis	sonarqube QualiGate	Npm	Trivy file scan	OWASP FS SCAN	Docker Build	Trivy image	Run container	Remove container	Kube deploy	kube deleter	Declarative: Post Actions
325ms	572ms	998ms	32s	848ms	25s	7s	4min 30s	3min 23s	4min 10s	2s	1s	4s	678ms	205ms
192ms	378ms	866ms					11s				1s		678ms	179ms
458ms	766ms	1s	32s	848ms (paused for 2s)	25s	7s	8min 50s	3min 23s	4min 10s	2s		4s		231ms

## Slack Notifications:



## Splunk:



## Pipeline:

```
@Library('Jenkins_shared_library') _  
  
def COLOR_MAP = [  
  
    'FAILURE' : 'danger',  
  
    'SUCCESS' : 'good'  
  
]
```

```
pipeline{

    agent any

    parameters {
        choice(name: 'action', choices: 'create\\ndelete', description: 'Select create or destroy.')
        string(name: 'DOCKER_HUB_USERNAME', defaultValue: 'shalinichandru', description: 'Docker Hub Username')
        string(name: 'IMAGE_NAME', defaultValue: 'youtube', description: 'Docker Image Name')
    }

    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }

    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }

    stages{
        stage('clean workspace'){
            steps{
                cleanWorkspace()
            }
        }

        stage('checkout from Git'){
            steps{
                checkoutGit('https://github.com/chandrushal/Youtube-clone-app.git', 'main')
            }
        }

        stage('sonarqube Analysis'){
            when { expression { params.action == 'create' } }
            steps{

```

```

        sonarqubeAnalysis()

    }

}

stage('sonarqube QualitGate'){

when { expression { params.action == 'create' } }

steps{

script{

    def credentialsId = 'Sonar-token'

    qualityGate(credentialsId)

}

}

}

stage('Npm'){

when { expression { params.action == 'create' } }

steps{

npmInstall()

}

}

stage('Trivy file scan'){

when { expression { params.action == 'create' } }

steps{

trivyFs()

}

}

stage('OWASP FS SCAN') {

steps {

    dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit',
    odcInstallation: 'DP-Check'

    dependencyCheckPublisher pattern: '**/dependency-check-report.xml'

}

}

```

```
}

stage('Docker Build'){

when { expression { params.action == 'create' } }

steps{

script{

def dockerHubUsername = params.DOCKER_HUB_USERNAME

def imageName = params.IMAGE_NAME


dockerBuild(dockerHubUsername, imageName)

}

}

}

stage('Trivy iamge'){

when { expression { params.action == 'create' } }

steps{

trivyImage()

}

}

stage('Run container'){

when { expression { params.action == 'create' } }

steps{

runContainer()

}

}

stage('Remove container'){

when { expression { params.action == 'delete' } }

steps{

removeContainer()

}

}
```

```

}

stage('Kube deploy'){

when { expression { params.action == 'create' } }

steps{
    kubeDeploy()
}

}

stage('kube deleter'){

when { expression { params.action == 'delete' } }

steps{
    kubeDelete()
}

}

post {

always {

echo 'Slack Notifications'

slackSend (

    channel: 'jenkins', / #change your channel name

    color: COLOR_MAP[currentBuild.currentResult],

        message: "*${currentBuild.currentResult}:* Job ${env.JOB_NAME} \n build
${env.BUILD_NUMBER} \n More info at: ${env.BUILD_URL}"


)

}

}

}

```