

Mr Cloud Book

Home Blog DevSecOps Contact About Me Testimonials

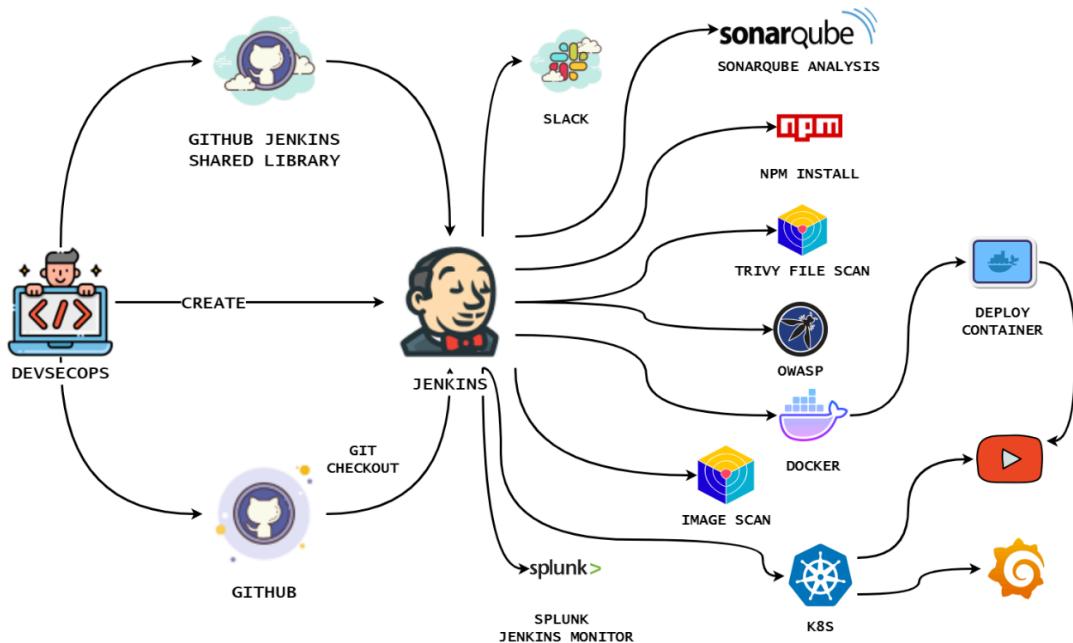
Search Blogs

DevOps

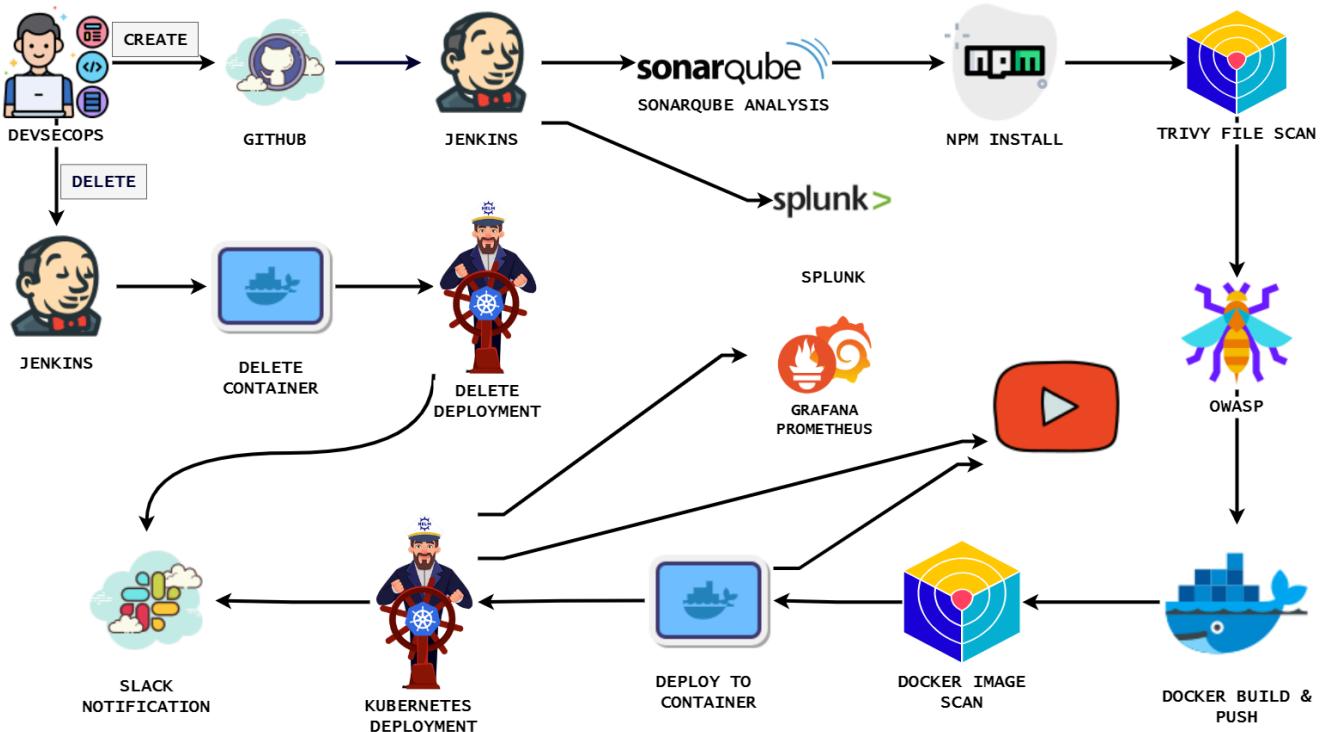
YouTube Clone App with DevSecOps and Jenkins Shared Library



mrcloobook.com · 8 January 2024



Hello and welcome! 🌟 Ready to create a YouTube clone that's rock-solid and feature-rich? 🔒



This blog is your gateway to a secure DevSecOps pipeline for your project. With Kubernetes, Docker, SonarQube, Trivy, OWASP Dependency Check, Prometheus, Grafana, Jenkins (and a shared library), Splunk, Rapid API, Slack notifications, and efficient parameters for creating and destroying your environment, we've got you covered.

Let's build a YouTube clone that's not just cutting-edge but also protected. Join us on this journey! 💪🌟

Github Repo: <https://github.com/Aj7Ay/Youtube-clone-app.git>

video on youtube: <https://youtu.be/OCZriuQe08U?si=INELPohIN-FZK-r6>

Normal Jenkinsfile you can use this one without the shared library

```

def COLOR_MAP = [
    'FAILURE' : 'danger',
    'SUCCESS' : 'good'
]
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/Youtu...
        }
    }
    stage("Sonarqube Analysis"){
        steps{
            withSonarQubeEnv('sonar-server') {
                sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectKey=youtube ...
                -Dsonar.projectKey=youtube '''
            }
        }
    }
    stage("quality gate"){
        steps {
            script {
                waitForQualityGate abortPipeline: false, credentials: ...
            }
        }
    }
    stage('Install Dependencies') {
        steps {
            sh "npm install"
        }
    }
    stage('OWASP FS SCAN') {
        steps {
            dependencyCheck additionalArguments: '--scan ./ --disab...
            dependencyCheckPublisher pattern: '**/dependency-check-i...
        }
    }
    stage('TRIVY FS SCAN') {
        steps {
            sh "trivy fs . > trivyfs.txt"
        }
    }
}
```

```

        }
    }

    stage("Docker Build & Push"){
        steps{
            script{
                withDockerRegistry(credentialsId: 'docker', toolName
                    sh "docker build --build-arg REACT_APP_RAPID_API_
                    sh "docker tag youtube sevenajay/youtube:latest "
                    sh "docker push sevenajay/youtube:latest "
                }
            }
        }
    }

    stage("TRIVY"){
        steps{
            sh "trivy image sevenajay/youtube:latest > trivyimage.t
        }
    }

    stage('Deploy to container'){
        steps{
            sh 'docker run -d --name youtube1 -p 3000:3000 sevenaja
        }
    }

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

    post {
        always {
            echo 'Slack Notifications'
            slackSend (
                channel: '#channel name', #change your channel name
                color: COLOR_MAP[currentBuild.currentResult],
                message: "*${currentBuild.currentResult}:* Job ${env.JOB_NAM
            )
        }
    }
}

```

Contents [hide]

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
- Step9E: K8S Deployment

STEPS:

Step 1: Launch an Ubuntu 22.04 instance for Jenkins

Log into AWS Console: Sign in to your AWS account.

Launch an Instance:

Choose “EC2” from services. Click “Launch Instance.”

Choose an AMI: Select an Ubuntu image.

Choose an Instance Type: Pick “t2.large.”

Key Pair: Choose an existing key pair or create a new one.

Configure Security Group:

Create a new security group. Add rules for HTTP, and HTTPS, and open all ports for learning purposes. Add Storage: Allocate at least 20 GB of storage.

Launch Instance: Review and launch the instance.

Access Your Instance: Use SSH to connect to your instance with the private key.

Keep in mind, that opening all ports is not recommended for production environments; it's just for educational purposes.

Instances (1) Info									
<input type="checkbox"/> Name		Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D	Launch instances
<input type="checkbox"/>	CI-CD	i-065c10200537a1eee	Running	t2.large	2/2 checks passed	No alarms	ap-south-1a	ec2-52-66-14	Launch instances

Connect to Your EC2 Instance and Install Jenkins:

Use MobaXterm or PuTTY to connect to your EC2 instance. Create a shell script named

```
sudo vi install_jenkins.sh
```

Paste the following script:



```
#!/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 main" | sudo tee /etc/apt/sources.list.d/adoptium.list >> /dev/null
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 /etc/apt/trusted.gpg.d/jenkins.io.key >> /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable/ jenkins io-2023 >> /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
```



Save and exit the text editor.

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.

You will need to go to your AWS EC2 Security Group and open Inbound Port 8080 since Jenkins works on Port 8080.

Now, grab your Public IP Address



EC2 Public IP Address: 8080

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



Getting Started

Unlock Jenkins

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
```

Please copy the password from either location and paste it below.

Administrator password

Continue

```
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

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

Build Queue

No builds in the queue.

Build Executor Status

Idle	2
Idle	1

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, we will create a Sonarqube container (Remember to add 9000 ports in the security group).

Run this command on your EC2 instance to create a SonarQube container:



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

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar  -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882fb8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b66c96bf9ad3d62289436af7f752fdb04993092d0ca5065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
4b66c96bf9ad        sonarqube:lts-community   "/opt/sonarqube/dock..."   9 seconds ago     Up 5 seconds      0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   sonar
```

Now copy the IP address of the ec2 instance



ec2-public-ip:9000



Enter username and password, click on login and change password



```
username admin
password admin
```



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

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

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 Set up global configuration	From Bitbucket Server Set up global configuration	From Bitbucket Cloud Set up global configuration	From GitHub Set up global configuration	From GitLab Set up global configuration
--	--	---	--	--

Step2B: Install Trivy on Jenkins machine

Create a shell script



```
sudo vi trivy.sh
```



Paste the below commands



```
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/ /" | sudo tee /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

[SPLUNK BLOG LINK](#)

[SPLUNK FULL VIDEO ON YOUTUBE](#)

Step 1: Launch Instances

1. Log in to your AWS console or your chosen cloud provider.

2. Navigate to the EC2 service and launch Ubuntu 22.04 instances. Ensure you select T2.medium as the instance type and allocate 24GB of storage to each instance.

Step 2: Install Splunk

At this point, the first machine is set up with Jenkins. You can now move to the second machine and proceed with the installation of Splunk.

Connect to your second instance using Putty or Mobaxtreme. To download and install Splunk on your Ubuntu instance use the `wget` command, use the following command:

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

To Depackage the Splunk use the below command

```
sudo dpkg -i splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb
```

```
ubuntu@ip-172-31-33-169:~$ ls
ubuntu@ip-172-31-33-169:~$ ls
splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb
ubuntu@ip-172-31-33-169:~$ 
ubuntu@ip-172-31-33-169:~$ sudo dpkg -i splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb
Selecting previously unselected package splunk.
(Reading database ... 64295 files and directories currently installed.)
Preparing to unpack splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb ...
Unpacking splunk (9.1.1+64e843ea36b1) ...
Setting up splunk (9.1.1+64e843ea36b1) ...
complete
ubuntu@ip-172-31-33-169:~$
```

```
sudo /opt/splunk/bin/splunk enable boot-start
```



By running this command, you ensure that Splunk Enterprise is configured to start automatically when your Ubuntu system boots, allowing you to seamlessly integrate it into your workflow.

Please note that after running this command, you should follow the on-screen prompts to accept the terms and complete the setup to 100%.

```
"C&I Services" means the services outlined in the Statement of Work.  
"C&I Services Materials" means the materials and other deliverables that are provided to you as part of the C&I Services, and any materials, technology, know-how and other innovations of any kind that we or our Personnel may create or reduce to practice in the course of performing the C&I Services, including without limitation all improvements or modifications to our proprietary technology, and all Intellectual Property Rights therein.  
"Customer Materials" means the data, information, and materials you provide to us in connection with your use of the C&I Services.  
"Fees" means the fees that are applicable to the C&I Services, as identified in the Statement of Work.  
"Intellectual Property Rights" means all worldwide intellectual property rights, including copyrights and other rights in works of authorship; rights in trademarks, trade names, and other designations of source or origin; rights in trade secrets and confidential information; and patents and patent applications.  
"Personnel" means any employee, consultant, contractor, or subcontractor of Splunk.  
"Splunk Preexisting IP" means, with respect to any C&I Services Materials, all associated Splunk technology and all Intellectual Property Rights created or acquired: (a) prior to the date of the Statement of Work that includes such C&I Services Materials, or (b) after the date of such Statement of Work but independently of the C&I Services provided under such Statement of Work.  
"Statement of Work" means the statements of work and/or any and all applicable Orders, that describe the specific services to be performed by Splunk, including any materials and deliverables to be delivered by Splunk.  
Do you agree with this license? [y/n]:  
Do you agree with this license? [y/n]: y  
This appears to be your first time running this version of Splunk.  
Splunk software must create an administrator account during startup. Otherwise, you cannot log in.  
Create credentials for the administrator account.  
Characters do not appear on the screen when you type in credentials.  
Please enter an administrator username: ■
```

After completing the initial setup and accepting the terms, you'll be prompted to create an admin user.

Administrator Username: Choose a username for the admin account. This should be a unique and secure username.

Administrator Password: Set a strong and secure password for the admin account. It's important to choose a password that combines upper and lower-case letters, numbers, and special characters for enhanced security.

Confirm your password to ensure it matches the one you initially entered.

By creating an administrator username and password, you'll have full access to your Splunk instance, allowing you to configure and manage it effectively.

```
Splunk software must create an administrator account during startup. Otherwise, you cannot log in.  
Create credentials for the administrator account.  
Characters do not appear on the screen when you type in credentials.  
  
Please enter an administrator username: ajay  
Password must contain at least:  
* 8 total printable ASCII character(s).  
Please enter a new password:  
Please confirm new password:  
Copying '/opt/splunk/etc/openldap/ldap.conf.default' to '/opt/splunk/etc/openldap/ldap.conf'.  
Generating RSA private key, 2048 bit long modulus  
.....+++++  
.....+++++  
e is 65537 (0x10001)  
writing RSA key  
  
Generating RSA private key, 2048 bit long modulus  
.....+++++  
.....+++++  
e is 65537 (0x10001)  
writing RSA key  
  
Moving '/opt/splunk/share/splunk/search_mrsparkle/modules.new' to '/opt/splunk/share/splunk/search_mrsparkle/modules'.  
Init script installed at /etc/init.d/splunk.  
Init script is configured to run at boot.  
ubuntu@ip-172-31-33-169:~$
```

The command `sudo ufw allow OpenSSH` is used to allow incoming SSH traffic through the UFW (Uncomplicated Firewall) on your Ubuntu system. It's essential for enabling SSH access to your server.



```
sudo ufw allow openSSH
```



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

```
ubuntu@ip-172-31-33-169:~$  
ubuntu@ip-172-31-33-169:~$ sudo ufw allow openSSH  
Rules updated  
Rules updated (v6)  
ubuntu@ip-172-31-33-169:~$
```

The command `sudo ufw allow 8000` is used to allow incoming network traffic on port 8000 through the UFW (Uncomplicated Firewall) on your Ubuntu system. It permits access to a specific port for network services or applications.



```
sudo ufw allow 8000
```



```
ubuntu@ip-172-31-33-169:~$  
ubuntu@ip-172-31-33-169:~$ sudo ufw allow 8000  
Rules updated  
Rules updated (v6)  
ubuntu@ip-172-31-33-169:~$ █
```

`sudo ufw status`: This command allows you to check the status of your UFW firewall. It will display information about whether the firewall is active, which rules are enabled, and whether it's set to allow or deny specific types of traffic.

`sudo ufw enable`: This command is used to enable the UFW firewall if it's not already active. Enabling the firewall ensures that the rules you've configured or will configure are enforced.

By running these commands, you can both check the current status of your firewall and activate it to apply the defined rules and settings.



```
sudo ufw status  
sudo ufw enable
```



```
ubuntu@ip-172-31-33-169:~$ sudo ufw status  
Status: inactive  
ubuntu@ip-172-31-33-169:~$  
ubuntu@ip-172-31-33-169:~$ sudo ufw enable  
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y  
Firewall is active and enabled on system startup  
ubuntu@ip-172-31-33-169:~$ █
```

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.



```
sudo /opt/splunk/bin/splunk start
```

```
ubuntu@ip-172-31-33-169:~$ sudo /opt/splunk/bin/splunk start
Splunk> Now with more code!
Checking prerequisites...
  Checking http port [8000]: open
  Checking mgmt port [8089]: open
  Checking appserver port [127.0.0.1:8065]: open
  Checking kvstore port [8191]: open
  Checking configuration... Done
    Creating: /opt/splunk/var/lib/splunk
    Creating: /opt/splunk/var/run/splunk/appserver/i18n
    Creating: /opt/splunk/var/run/splunk/appserver/modules/static/css
    Creating: /opt/splunk/var/run/splunk/upload
    Creating: /opt/splunk/var/run/splunk/search_telemetry
    Creating: /opt/splunk/var/run/splunk/search_log
    Creating: /opt/splunk/var/spool/splunk
    Creating: /opt/splunk/var/spool/dirmongocache
    Creating: /opt/splunk/var/lib/splunk/authDb
    Creating: /opt/splunk/var/lib/splunk/hashDb
New certs have been generated in '/opt/splunk/etc/auth'.
  Checking critical directories... Done
  Checking indexes...
    Validated: _audit _configtracker _internal _introspection _metrics _metrics_rollup _telemetry _thefishbucket history main summary
  Done
  Checking filesystem compatibility... Done
  Checking conf files for problems...
  Done
  Checking default conf files for edits...
  Validating installed files against hashes from '/opt/splunk/splunk-9.1.1-64e843ea36b1-linux-2.6-x86_64-manifest'
    All installed files intact.
  Done
All preliminary checks passed.

Starting splunk server daemon (splunkd)...
Generating a RSA private key
.....
```

```
Starting splunk server daemon (splunkd)...
Generating a RSA private key
.....+++++
writing new private key to 'privkeySecure.pem'
-----
Signature ok
subject=/CN=ip-172-31-33-169/0=SplunkUser
Getting CA Private Key
writing RSA key
PYTHONHTTPSVERIFY is set to 0 in splunk-launch.conf disabling certificate validation for the httplib and urllib libraries shipped with the embedded Python interpreter; must be set to "1" for increased security
Done

Waiting for web server at http://127.0.0.1:8000 to be available..... Done

If you get stuck, we're here to help.
Look for answers here: http://docs.splunk.com

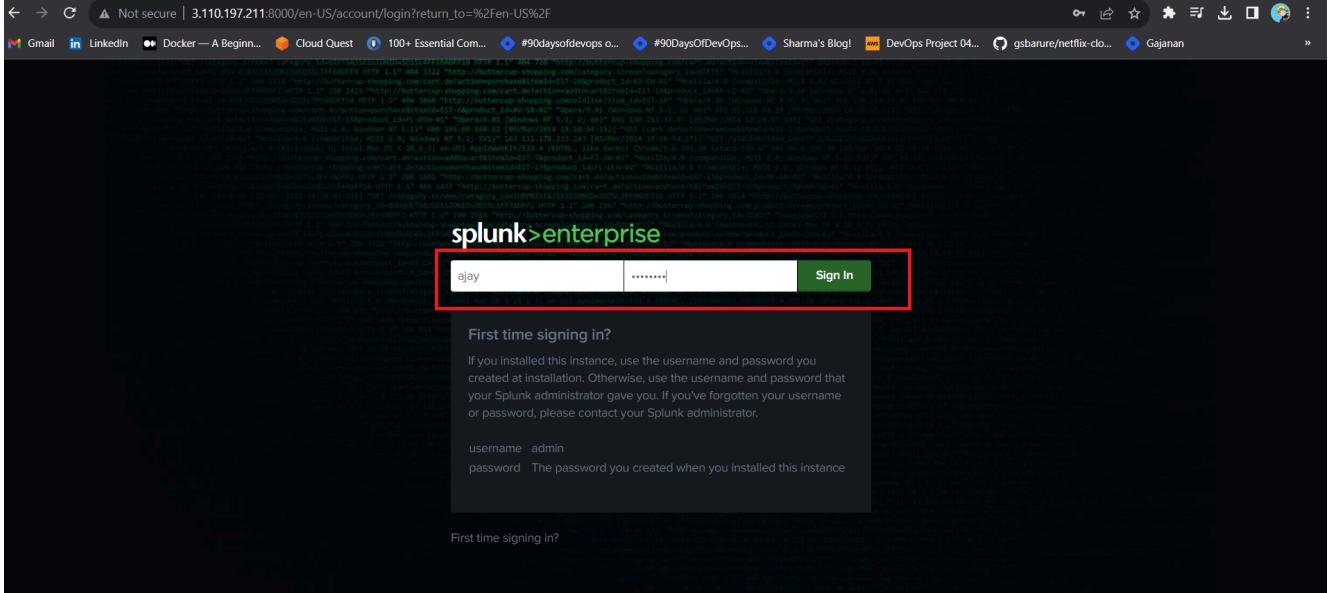
The Splunk web interface is at http://ip-172-31-33-169:8000
ubuntu@ip-172-31-33-169:~$
```

After successfully starting your Splunk instance, you can now access its web interface to start exploring and analyzing your data.

Copy Your Splunk Instance Public IP Address: Navigate to your cloud provider's console and find the public IP address of your Splunk instance.

Log in with Your Credentials: You'll be prompted to log in with the administrator username and password you created during the setup process (typically using the command `sudo /opt/splunk/bin/splunk enable boot-start`).

splunk-public-ip:8000



This is the Splunk Dashboard

Step3B: Install the Splunk app for Jenkins

In Splunk Dashboard

Click on Apps -> Find more apps

The screenshot shows the Splunk interface. In the top left, there's a search bar with the placeholder "Search app". Below it, there are links for "Splunk Secure Gateway", "Upgrade Readiness App", "Manage Apps", and "Find More Apps". On the right, there's a sidebar titled "Administrator" with sections for "Common tasks" and "Learning and resources". The "Common tasks" section includes options like "Add data", "Search your data", "Visualize your data", "Add team members", "Manage permissions", and "Configure mobile devices".

Search for Jenkins in the Search bar

You will get the Splunk app for Jenkins and click on install

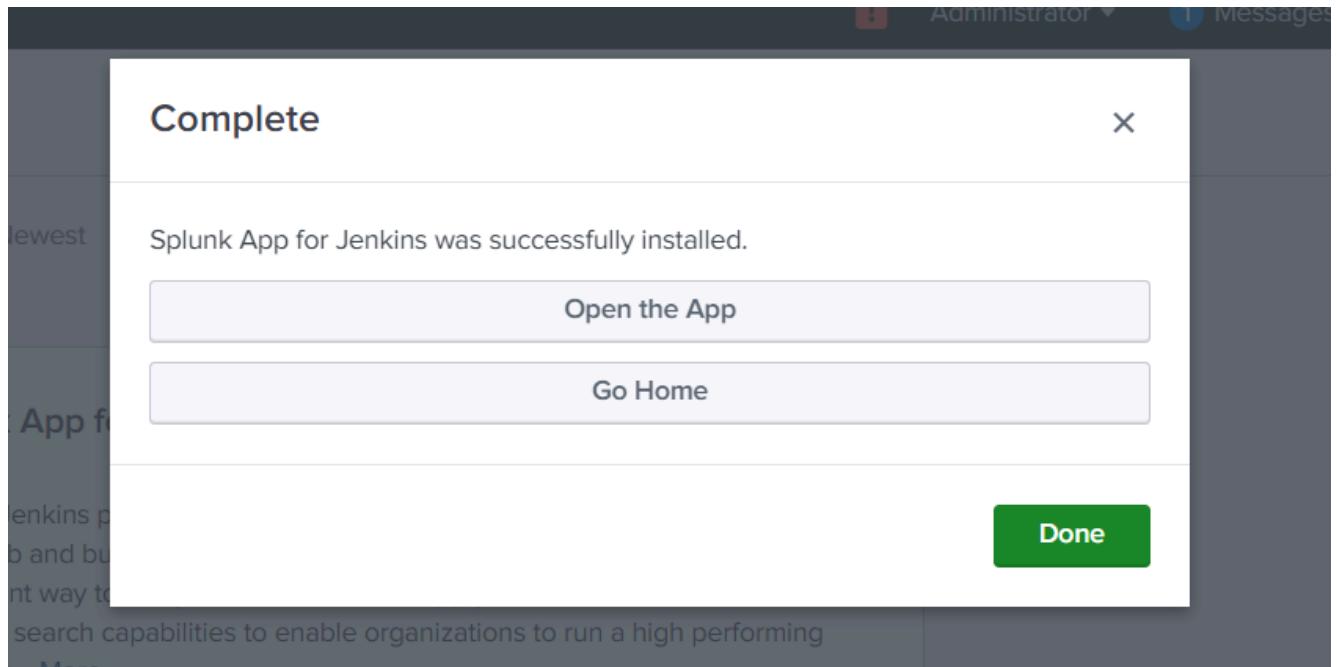
The screenshot shows the Splunk App Store search results for "Jenkins". The search bar at the top has "Jenkins" typed in. On the left, there's a sidebar with categories like IT Operations, Security, Fraud & Compliance, etc. The main area shows a single result: "Splunk App for Jenkins" by Splunk Works. The result card includes a thumbnail of a person, the app name, a green "Install" button, a brief description, category information (DevOps), and download details (17164 downloads, 4 years ago). A yellow box highlights the "Install" button.

You will be prompted to provide your Splunk credentials. That's why we created a Splunk account

The screenshot shows a modal dialog titled "Login and Install". It asks for the user's Splunk.com username and password. The username "postbox.aj99@gmail.com" is entered in the field. Below the password field is a link "Forgot your password?". The modal contains a detailed disclaimer about the app's license and dependency. At the bottom, there's a checkbox for accepting the "End User License Agreement for Third-Party Content" and a note about reading the terms and conditions. The "Agree and Install" button is highlighted with a red box.

Click on Agree and install

Now click on Go home



On the homepage of Splunk, you will see Jenkins has been added

A screenshot of the Splunk enterprise homepage. On the left, there is a sidebar with a search bar and a list of apps: "Search & Reporting", "Splunk App For Jenkins" (which is highlighted with a red box), "Splunk Secure Gateway", and "Upgrade Readiness App". Below the sidebar is a link "Find more apps". The main content area is titled "Hello, Administrator". It features a "Quick links" menu with options like "Dashboard", "Recently viewed", "Created by you", and "Shared with you". Below the menu is a section titled "Common tasks" with six items: "Add data", "Search your data", "Visualize your data", "Add team members", "Manage permissions", and "Configure mobile devices". The "Splunk App For Jenkins" item is also listed under "Common tasks".

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

The screenshot shows the Splunk Enterprise dashboard with the 'Data inputs' section highlighted by a red box. The dashboard includes sections for 'Hello, Administrator', 'Common tasks', 'Learning and resources', and various system and user management links.

Click on HTTP Event Collector.

The screenshot shows the 'Data inputs' page in Splunk. The 'HTTP Event Collector' option is highlighted by a red box. The page displays a table of local inputs, each with a type, count, and 'Add new' button.

Type	Inputs	Actions
Files & Directories	15	+ Add new
HTTP Event Collector	0	+ Add new
TCP	0	+ Add new
UDP	0	+ Add new
Scripts	25	+ Add new
Splunk Assist Instance Identifier	1	+ Add new

Click on Global Settings

The screenshot shows the 'HTTP Event Collector' page. The 'Global Settings' button is highlighted by a red box. The page includes a token search bar and a table for managing tokens.

Set All tokens to enabled

Uncheck SSL enable

Use 8088 port and click on save

The screenshot shows the 'Edit Global Settings' page in Splunk. The 'All Tokens' dropdown is set to 'Enabled'. The 'Enable SSL' checkbox is unchecked. The 'HTTP Port Number' field contains '8088'. The 'Save' button at the bottom right is highlighted with a green box.

Now click on New token

The screenshot shows the 'HTTP Event Collector' page in Splunk. The 'New Token' button at the top right is highlighted with a red box and has a red arrow pointing to it.

Provide a Name and click on the next

The screenshot shows the 'Add Data' wizard in Splunk. The 'Next >' button is highlighted with a red box. The 'Name' field in the configuration section is highlighted with a red box and contains 'Jenkins'.

Click Review

Add Data

Select Source Input Settings Review Done

Input Settings

Optionaly set additional input parameters for this data input as follows:

Source type

The source type is one of the default fields that the Splunk platform assigns to all incoming data. It tells the Splunk platform what kind of data you've got, so that the Splunk platform can format the data intelligently during indexing. And it's a way to categorize your data, so that you can search it easily.

Index

The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. [Learn More](#)

Automatic Select New

Click Submit

Add Data

Select Source Input Settings Review Done

Review

Input Type Token
 Name Jenkins
 Source name override N/A
 Description N/A
 Enable indexer acknowledg No
 Output Group N/A
 Allowed indexes N/A
 Default index default
 Source Type Automatic
 App Context launcher

Submit >

Click Start searching

Add Data

Select Source Input Settings Review Done

✓ Token has been created successfully.
 Configure your inputs by going to [Settings > Data Inputs](#)

Token Value: 757be2ee-eb37-4529-a8f2-9cf7c12e

Start Searching

Search your data now or see examples and tutorials. [\[link\]](#)

Add More Data Add more data inputs now or see examples and tutorials. [\[link\]](#)

Download Apps Apps help you do more with your data. Learn more. [\[link\]](#)

Build Dashboards Visualize your searches. Learn more. [\[link\]](#)

Now let's copy our token again

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

The screenshot shows the Splunk Enterprise interface. At the top, there's a navigation bar with 'splunk>enterprise' and various dropdowns like 'Administrator', 'Messages', 'Settings' (which is currently selected), 'Activity', 'Help', and 'Find'. Below the navigation is a search bar with the query 'source="http:Jenkins"'. A warning message says 'Search not executed: The minimum free disk space (500MB) reached for /opt/splunk/var/run/splunk/dispatch. user=ajay, concurrency_limit=5000'. Under the search bar are tabs for 'Events', 'Patterns', 'Statistics', and 'Visualization'. Below these are 'List', 'Format', and '20 Per Page' dropdowns. The main area shows a table with columns 'Time' and 'Event'. On the right side, there's a sidebar with several sections: 'KNOWLEDGE' (Searches, reports, and alerts, Data models, Event types, Tags, Fields, Lookups, User interface, Alert actions, Advanced search, All configurations), 'SYSTEM' (Server settings, Server controls, Health report manager, RapidDiag, Instrumentation, Licensing, Workload management, Mobile settings), 'DISTRIBUTED ENVIRONMENT' (Indexer clustering, Forwarder management, Federated search, Distributed search), and 'USERS AND AUTHENTICATION' (Roles, Users, Tokens, Password management, Authentication methods). The 'Data inputs' section under 'DATA' is specifically highlighted with a red box.

Click on the HTTP event collector

The screenshot shows the 'Data inputs' page in Splunk. The title is 'Data inputs' with a subtitle 'Set up data inputs from files and directories, network ports, and scripted inputs. If you want to set up forwarding and receiving between two Splunk instances, go to Forwarding and receiving.' Below this is a table titled 'Local inputs' with columns 'Type', 'Inputs', and 'Actions'. The rows are: 'Files & Directories' (15 inputs, '+ Add new'), 'HTTP Event Collector' (1 input, highlighted with a green box, '+ Add new'), 'TCP' (0 inputs, '+ Add new'), 'UDP' (0 inputs, '+ Add new'), and 'Scripts' (25 inputs, '+ Add new').

Now copy your token and keep it safe

The screenshot shows the 'HTTP Event Collector' token page. The title is 'HTTP Event Collector' with a subtitle 'Data Inputs > HTTP Event Collector'. There are buttons for 'Global Settings' and 'New Token'. Below is a table with columns 'Name', 'Actions', 'Token Value', 'Source Type', 'Index', and 'Status'. One row for 'Jenkins' has its 'Token Value' ('757be2ee-eb37-4529-a8f2-9cf7c12e') highlighted with a green box.

Add Splunk Plugin in Jenkins

Go to Jenkins dashboard

Click on Manage Jenkins -> Plugins -> Available plugins

Search for Splunk and install it.

The screenshot shows the Jenkins Plugins page. A search bar at the top contains the text "splunk". Below it, a list of plugins is shown, with the "Splunk" plugin highlighted by a green box. The Splunk plugin entry includes the name "Splunk 1.10.1", a "Released" date of "7 mo 27 days ago", and a brief description: "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." There are tabs for "Build Notifiers", "Build Reports", and "Other Post-Build Actions".

Again Click on Manage Jenkins -> System

Search for Splunk

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. The "Enable" checkbox is checked. The "HTTP Input Host" field contains "3.110.197.211" and is labeled "SPLUNK PUBLIC IP". The "HTTP Input Port" field contains "8088". The "HTTP Input Token" field contains "757be2ee-eb37-4529-a8f2-9cf7c12a5c18" and is labeled "SPLUNK TOKEN". The "SSL Enabled" checkbox is unchecked. The "Send All Pipeline Console Logs" checkbox is checked. The "Jenkins Master Hostname" field contains "13.234.48.89" and is labeled "JENKINS PUBLIC IP". Below the form is a progress bar labeled "Testing...". At the bottom are "Save" and "Apply" buttons.

Now go to Putty or Mobaxtreme and In Splunk machine run this command



```
sudo ufw allow 8088
```

```
ubuntu@ip-172-31-33-169:~$ sudo ufw allow 8088
Rule added
Rule added (v6)
ubuntu@ip-172-31-33-169:~$ 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)

ubuntu@ip-172-31-33-169:~$
```

Now in the Jenkins dashboard Under Splunk click on Test connection



Restart Both Splunk and Jenkins

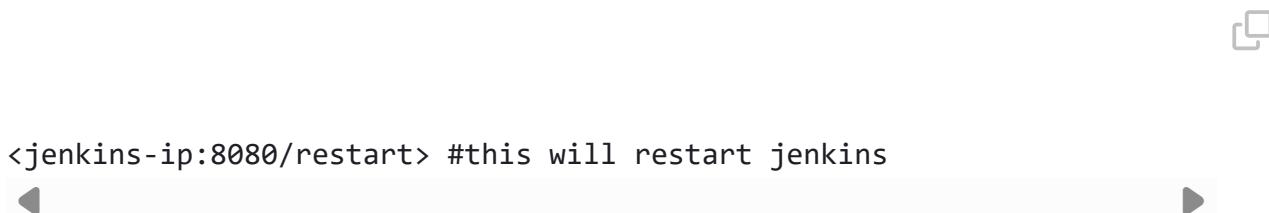
Let's Restart our Splunk machine

Click on Settings -> Server controls

Restart and log in again

The screenshot shows the Splunk interface with a 'Server controls' section. Inside, there's a 'Restart Splunk' button, which is highlighted with a red rectangular border. Below the button, there's a note: 'Click the button below to restart Splunk.'

Now restart Jenkins and log in again.

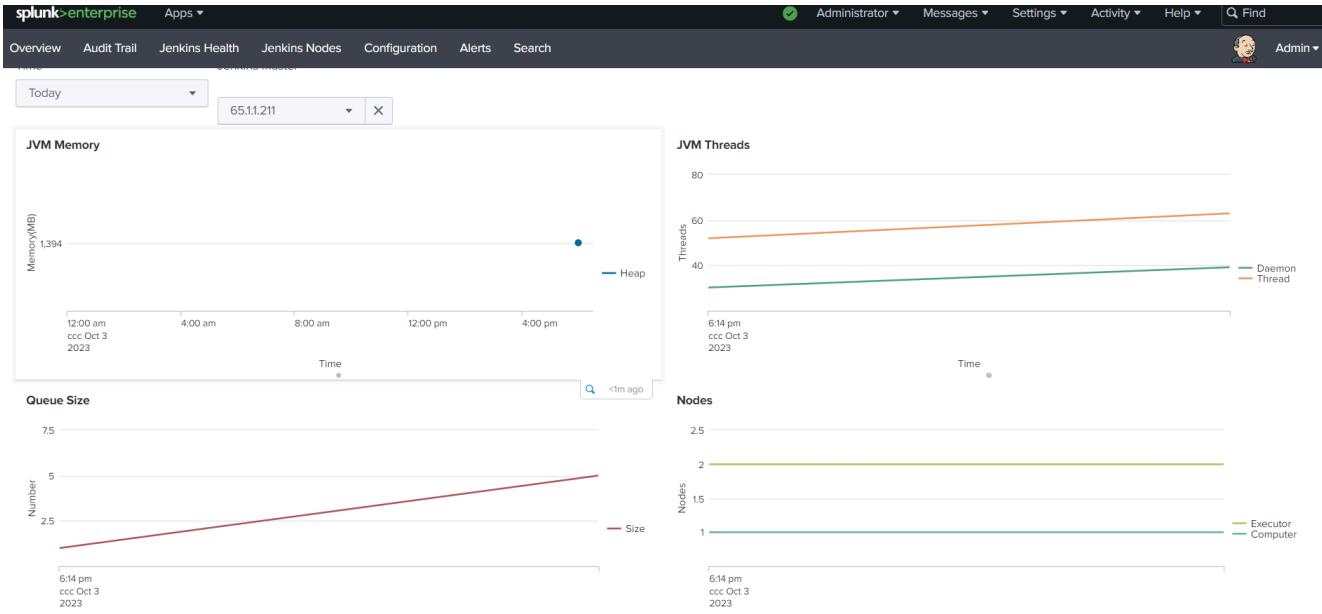


Now go to Splunk and click on the Jenkins app and you will get this output monitoring

Sample image.

The screenshot shows the Jenkins app within the Splunk interface. It displays a 'Build Analysis' section with various filters and a timeline. The timeline shows builds from 12:00 AM to 8:00 AM on Wednesday, October 11, 2023. Below the timeline is a table of build results:

Jenkins Master	Job	Build	StartTime	Jenkins Node	Duration	Status
65.11.211	Test2	17	2023-10-11 18:16:11	(built-in)	00:00:01.606	✓
65.11.211	Test2	16	2023-10-11 18:16:11	(built-in)	00:00:01.507	✓
65.11.211	Test2	15	2023-10-11 18:16:11	(built-in)	00:00:01.475	✓
65.11.211	Test2	14	2023-10-11 18:16:11	(built-in)	00:00:01.29	✓
65.11.211	Test2	13	2023-10-11 18:16:11	(built-in)	00:00:01.311	✓
65.11.211	Test1	42	2023-10-11 18:16:05	(built-in)	00:00:02.127	✓
65.11.211	Test1	41	2023-10-11 18:16:05	(built-in)	00:00:02.304	✓
65.11.211	Test1	40	2023-10-11 18:16:05	(built-in)	00:00:02.164	✓

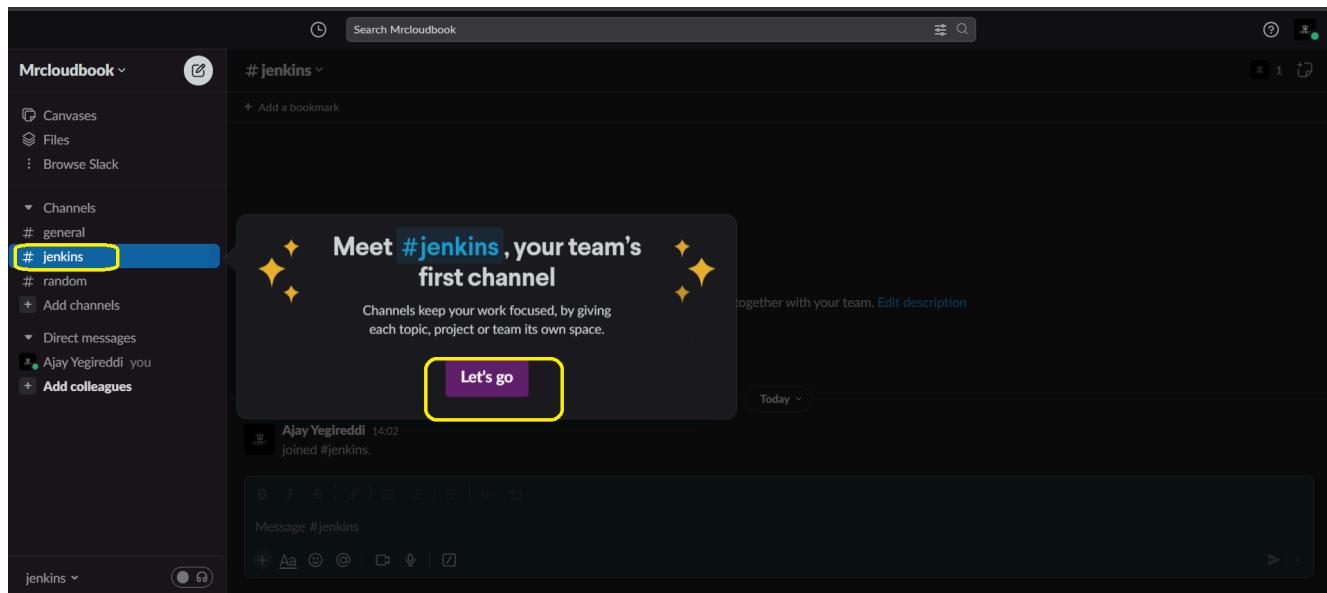


Step4A: Integrate Slack for Notifications

[SLACK BLOG](#)

[SLACK FULL VIDEO ON YOUTUBE](#)

Create a Slack account and create a channel Named Jenkins

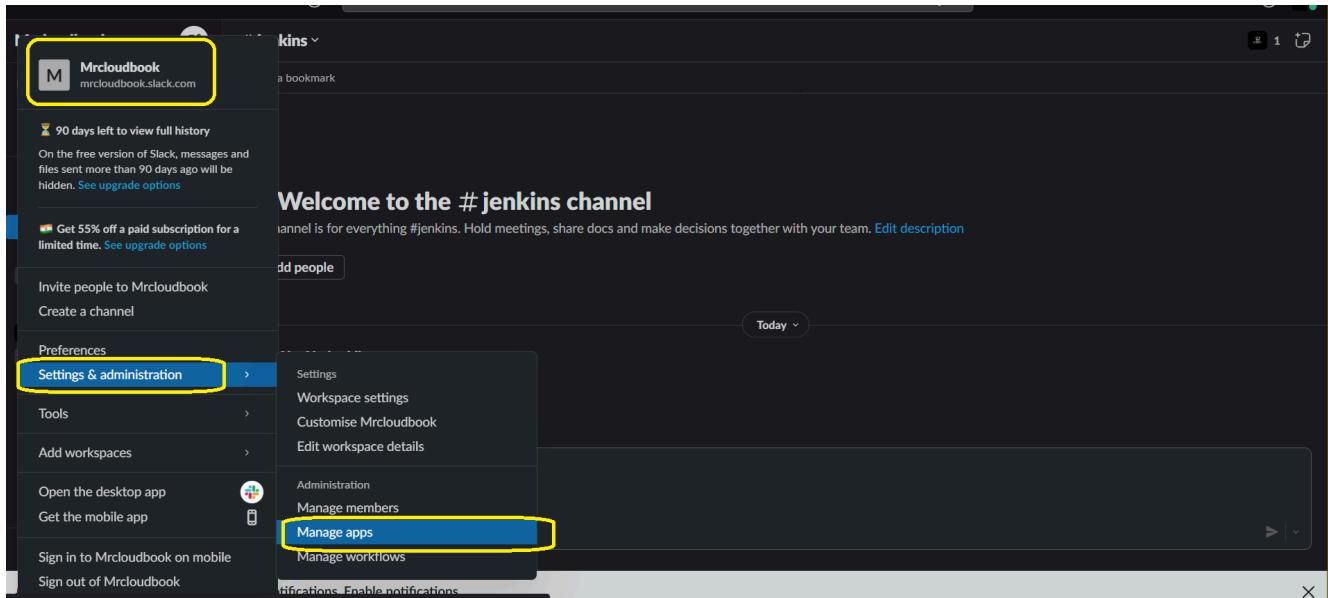


Step4B: Install the Jenkins CI app on Slack

Go to Slack and click on your name

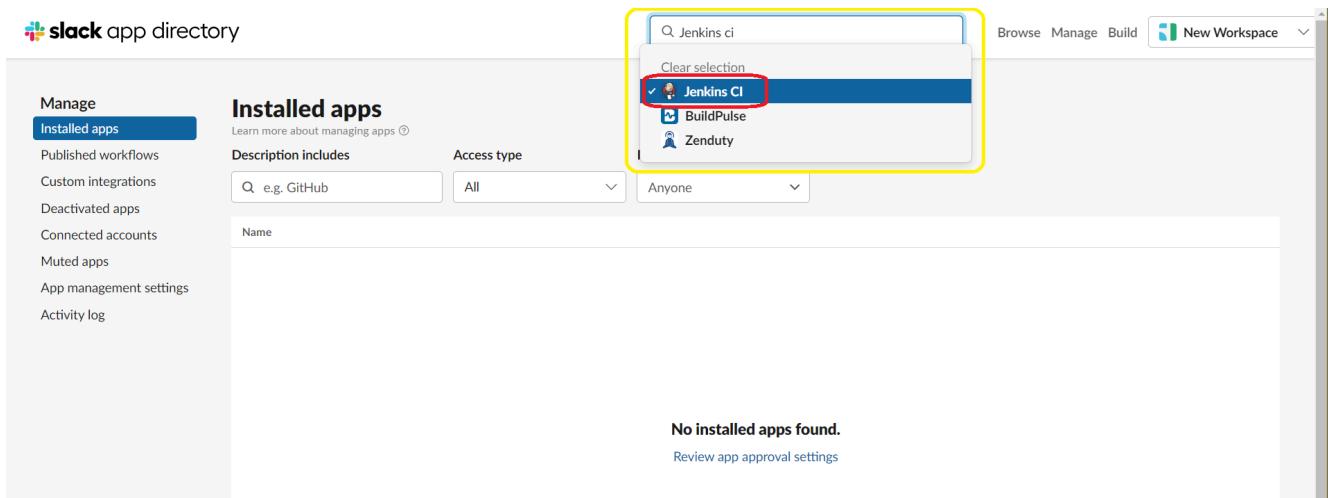
Select Settings and Administration

Click on Manage apps



It will open a new tab

Search for Jenkins CI and click on it



It will open another tab

Click on Add to Slack

The screenshot shows the Jenkins CI app page in the Slack App Directory. At the top, there's a search bar labeled "Search App Directory". To the right are links for "Browse", "Manage", and "Build", followed by a user icon for "Mrcloudbook". Below the header, there's a "Browse apps" link. The main content area features a cartoon character icon of a man in a suit holding a briefcase, with the text "Jenkins CI" next to it. Below the icon are three tabs: "Description" (which is underlined), "Permissions", and "Security & compliance". A description text states: "Jenkins CI is a customisable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs." Another line of text says: "This integration will post build notifications to a channel in Slack." A prominent green button labeled "Add to Slack" is centered at the bottom of the app card. This button is highlighted with a yellow border in the screenshot.

Now choose your Slack channel

Click on Add Jenkins CI integration.

The screenshot shows the "Post to channel" step for adding the Jenkins CI integration. It includes a search bar for channels and a red box highlighting the "Add Jenkins CI integration" button. The Jenkins CI app card is visible in the background.

You will be redirected to this page

slack app directory

Search App Directory

Browse apps > Jenkins CI > Edit configuration

Jenkins CI
Added by Ajay Yegireddi on 12th October 2023

Disable • Remove

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.

Setup instructions

Here are the steps required to add the Jenkins CI integration.

Note: These instructions are for v2.8. To install an older version, go down to [Previous setup instructions](#).

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 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: `mrcloudbook`
- Integration token credential ID: Create a secret text credential using `ZYMK1vg6NiPdwraMnV3HjDg` as the value

The other fields are optional. You can click on the question mark icons next to them for more information. Press **Save** once you've finished.

Note: Please remember to replace the integration token in the screenshot below with your own.



Copy the team subdomain and integration token credential ID for later use.

slack app directory

Search App Directory

Step 2 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: `mrcloudbook`
- Integration token credential ID: Create a secret text credential using `ZYMK1vg6NiPdwraMnV3HjDg` as the value

The other fields are optional. You can click on the question mark icons next to them for more information. Press **Save** once you've finished.

Note: Please remember to replace the integration token in the screenshot below with your own.



slack app directory

Search App Directory

Here are the instructions for installing older versions of the Slack notifier.

expand

Integration settings

Post to channel
Notifications that come in from Jenkins CI will be posted here.

#jenkins	or create a new channel
----------	-------------------------

Token
This token is used as the key to your Jenkins CI integration.

ZYMK1vg6NiPdwraMnV3HjDg	Regenerate
-------------------------	------------

Descriptive label
Use this label to provide extra context in your list of integrations (optional).

Optional description of this integration
--

Install Slack Notification Plugin in Jenkins

Go to Jenkins Dashboard

Click on manage Jenkins -> Plugins -> Available plugins

Search for Slack Notification and install

The screenshot shows the Jenkins Plugins page. In the top navigation bar, 'Manage Jenkins' is selected. Under 'Available plugins', a search bar contains the text 'Slack'. A list of available plugins is shown, with 'Slack Notification' by slack being the first result. This plugin has a version of 684.v833089650554 and a release date of 2 mo 1 day ago. A description below the plugin states: 'Integrates Jenkins with Slack, allows publishing build statuses, messages and files to Slack channels.' The 'Install' button is visible on the right.

Click on Manage Jenkins -> Credentials -> Global

Select kind as Secret Text

At Secret Section Provide Your Slack integration token credential ID

Id and description are optional and create

The screenshot shows the Slack App Directory. It includes two main sections: 'Step 2' and 'Step 3'.

- Step 2:** A screenshot of the Jenkins Plugins page showing the 'Available' tab. The 'Slack Notification' plugin by slack is listed with a checked checkbox, indicating it is selected for installation. The description below says: 'This plugin is a Slack notifier that can publish build status to Slack channels.'
- Step 3:** A screenshot of the Jenkins 'Global Slack Notifier Settings' configuration page. It shows fields for 'Slack compatible app URL (optional)', 'Team Subdomain' (set to 'jenkins-slack-plugin'), 'Integration Token' (containing the value 'ZYMk1vg6N1PdwvJaInV3HJDg'), and 'Integration Token Credential ID' (containing the value 'some text (bot user slack token)'). A note at the bottom of this section says: 'Note: Please remember to replace the integration token in the screenshot below with your own.'

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

New credentials

Kind

Secret text

Scope ?

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

Secret

..... integration token credential ID

ID ?

slack

Description ?

slack

Create

Click on Manage Jenkins -> System

Go to the end of the page

Workspace -> team subdomain

Credential -> Select your Credential for Slack

Default channel -> Provide your Channel name

Test connection

slack app directory

Search App Directory

Browse Manage Build Mrcloudbo... ▾

<p>Step 2</p> <p>Click on Manage plugins and search for Slack notification in the Available tab. Click the tick box and install the plugin.</p> <p>The screenshot shows the Jenkins Manage Plugins interface. A search bar at the top contains 'slack notification'. Below it, there are tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. Under the 'Available' tab, there is a table with columns 'Name' and 'Version'. A single row for 'Slack Notification' is shown, with a checkbox next to it. A tooltip below the table says: 'This plugin is a Slack notifier that can publish build status to Slack channels.'</p>	<p>Step 3</p> <p>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:</p> <ul style="list-style-type: none"> Team subdomain: <code>mrcloudbook</code> Integration token credential ID: Create a secret text credential using <code>ZYMK1vg6N1PdwrJaInV3HJ0g</code> as the value <p>The other fields are optional. You can click on the question mark icons next to them for more information. Press Save once you've finished.</p> <p>Note: Please remember to replace the integration token in the screenshot below with your own.</p> <p>The screenshot shows the 'Global Slack Notifier Settings' configuration page. It includes fields for 'Slack compatible app URL (optional)', 'Team Subdomain' (set to 'jenkins-slack-plugin'), 'Integration Token' (empty), and 'Integration Token Credential ID' (set to 'some text (bot user slack token)'). A note at the bottom states: '⚠ Exposing your Integration Token is a security risk. Please use the Integration Token Credential ID'.</p>
--	--

Dashboard > Manage Jenkins > System >

Slack

Workspace ?
mrcloudbook **team subdomain**

Credential ?
slack **Select Credential**

Default channel / member id ?
#jenkins **Your Slack channel name**

Custom slack app bot user ?

Advanced

Click on Apply and save

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}"
        )
    }
}
```

You will get a Notification in Slack

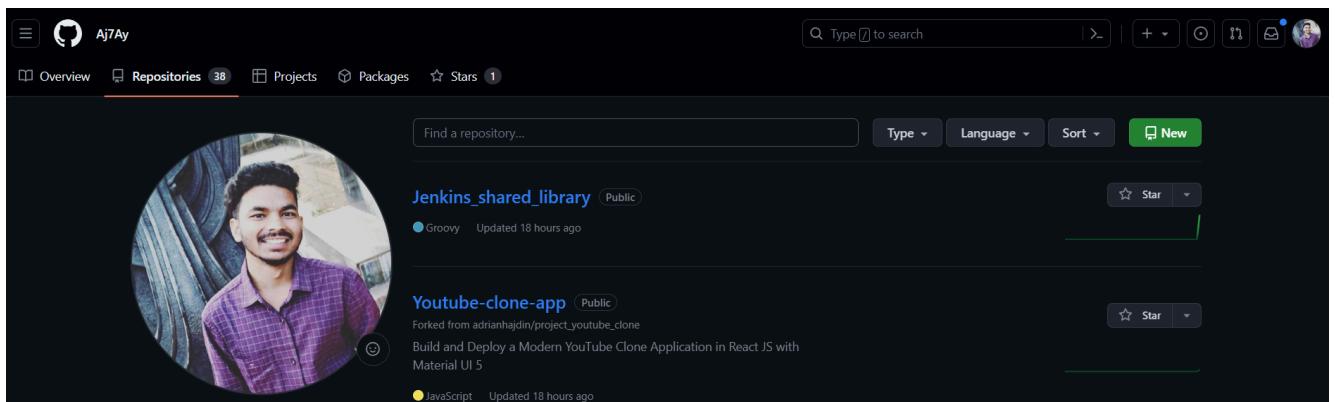
Step5A: Start Job

Go to Jenkins dashboard and click on New Item.

Provide a name for the Job & click on Pipeline and click on OK.

Step5B: Create a Jenkins shared library in GitHub

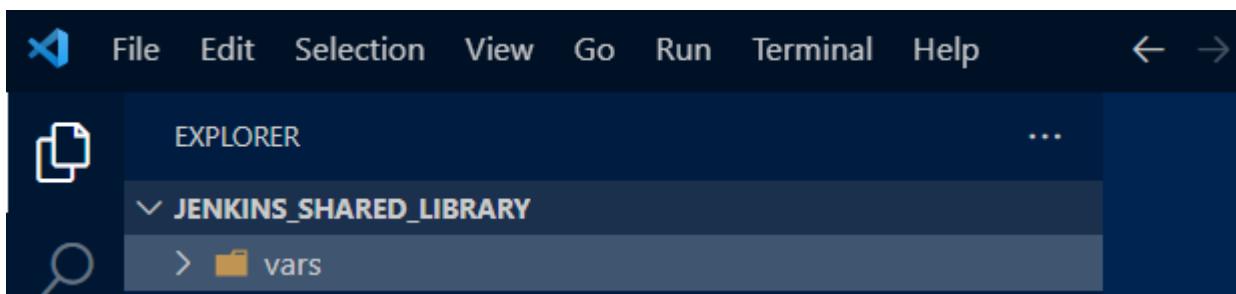
Create a new repository in GitHub named Jenkins_shared_library.



Connect to your VS Code

Create a directory named Jenkins_shared_library

Create a Vars directory inside it



Open Terminal

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/Aj7Ay/Jenkins_shared_library.g
git push -u origin main
```

Nov



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

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 ?	<input type="text" value="Jenkins_shared_library"/> provide a name	X
Default version	?	<input type="text" value="main"/> MAIN	
Currently maps to revision: 6bcb83719364d62ca8a846731a69695fe2f2dcde			
<input type="checkbox"/> Load implicitly ? <input checked="" type="checkbox"/> Allow default version to be overridden ? <input checked="" type="checkbox"/> Include @Library changes in job recent changes ? <input type="checkbox"/> Cache fetched versions on controller for quick retrieval ?			
Retrieval method			
<input type="text" value="Modern SCM"/> SELECT MODERN SCM			

Lets a library from an SCM plugin using newer interfaces optimized for this purpose. The recommended option when available.

Source Code Management

Git **Select Git**

Project Repository ?
https://github.com/Aj7Ay/Jenkins_shared_library.git **Provide Jenkins shared library git URL**

Credentials ?
- none - **Add**

Behaviors

- Discover branches ? **X**
- Add**

Fresh clone per build ?

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 :
def COLOR_MAP = [
    'FAILURE' : 'danger',
    'SUCCESS' : 'good'
]
pipeline{
    agent any
    parameters {
        choice(name: 'action', choices: 'create\ndelete', description:
    }
    stages{
        stage('clean workspace'){
            steps{
                cleanWorkspace()
            }
        }
        stage('checkout from Git'){
            steps{
                checkoutGit('https://github.com/Aj7Ay/Youtube-clone-app')
            }
        }
    }
}
```

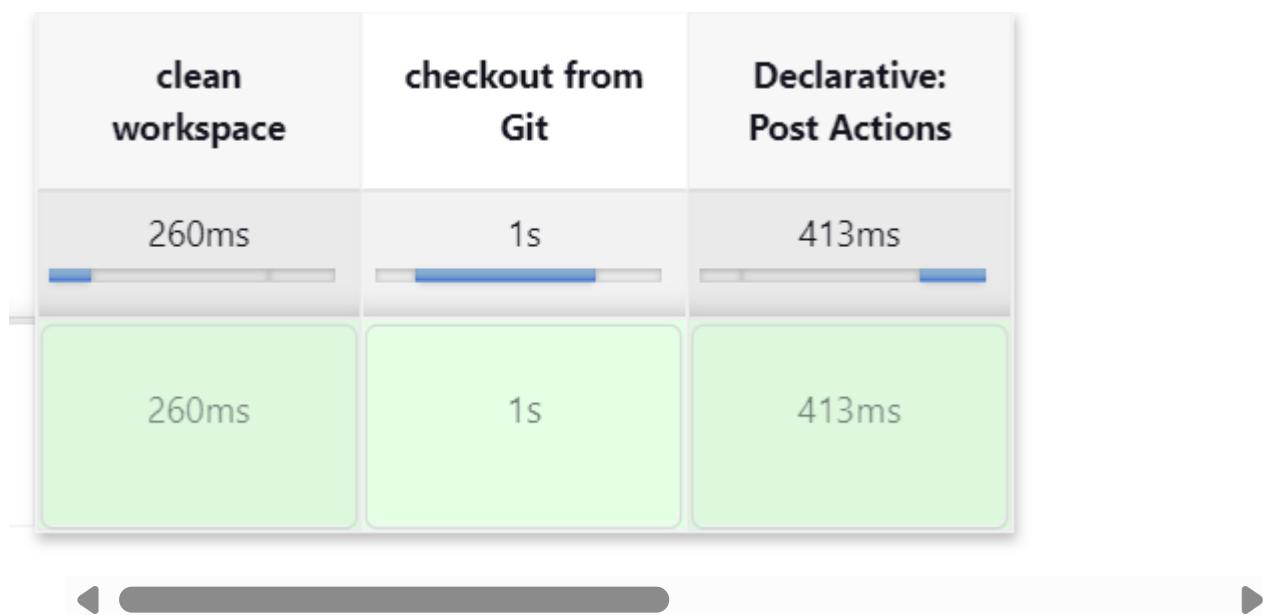
```

        }
    }
    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} ${env.BUILD_NUMBER} started"
            )
        }
    }
}

```

Build with parameters and build

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. On the left, there's a sidebar with links for Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress. The main area is titled "Plugins" and has tabs for Install and Name (sorted). It lists two plugins:

- Eclipse Temurin installer 1.5**: Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net. Status: Released 11 mo ago. A note says: "This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information."
- SonarQube Scanner 2.15**: External Site/Tool Integrations, Build Reports. Status: Released 9 mo 19 days ago. A note says: "This plugin allows an easy integration of [SonarQube](#), the open source platform for Continuous Inspection of code quality."

Below this, another section shows the "NodeJS 1.6.1" plugin installed, with a note: "NodeJS Plugin executes [NodeJS](#) script as a build step." Status: Released 1 mo 2 days ago.

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 page under "JDK installations". There's a button to "Add JDK". A configuration dialog is open for a new JDK named "jdk17". The "Install automatically" checkbox is checked. Under "Install from adoptium.net", the "Version" dropdown is set to "jdk-17.0.8.1+1". There's also a "Add installer" button.

Step6C: 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

click on update Token

Create a token with a name and generate

Tokens of Administrator**Generate Tokens**

Name	Expires in
<input type="text" value="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!



`squ_21d162904c1c72cf8b39665f96480185c99dc2f9`

Name	Type	Project	Last use	Created	Expiration	
Jenkins	User		Never	September 8, 2023	October 8, 2023	<input type="button" value="Revoke"/>

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:

Scope:

Secret:

ID:

Description:

You will 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	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"/>	X
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 style="background-color: #0072bc; color: white; border-radius: 5px; padding: 5px; margin-right: 10px;" type="button" value="Save"/> <input type="button" value="Apply"/>		

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.

SonarQube Scanner installations**SonarQube Scanner****Name**

Install automatically ?

Install from Maven Central**Version**

In the Sonarqube Dashboard add a quality gate also

Administration-> Configuration->Webhooks

The screenshot shows the SonarQube administration interface. The top navigation bar has tabs for Gmail, YouTube, Amazon Web Servic..., Coaching on DevO..., LinkedIn, Docker — A Beginn..., Cloud Quest, 100+ Essential Com..., Advanced End-to-E..., LINUX - YouTube T..., and How to Install Jenki... The main navigation bar includes Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration (which is highlighted with a red box), and a search bar for projects.

The Administration page has sub-tabs: General Settings, Encryption, and Webhooks (also highlighted with a red box). Below these tabs is a search bar labeled "Search by login or name...".

The main content area displays user information in a table:

	SCM Accounts	Last connection	Groups	Tokens
Administrator admin		< 1 hour ago	sonar-administrators sonar-users	1

At the bottom of the table, it says "1 of 1 shown".

Click on Create

The screenshot shows the SonarQube configuration page for Webhooks. The top navigation bar is identical to the previous screenshot. The main content area has a "Create" button highlighted with a black box.

The "Webhooks" section contains the following text:

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#).

Below this text, there is a message: "No webhook defined."

Add details

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

Create Webhook

All fields marked with * are required

Name *
jenkins

URL *
http://43.204.36.242:8090/sonarqube-webhook/

Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header.

Create **Cancel**

Get the most out of SonarQube! Take advantage of the whole ecosystem by using SonarLint, a free IDE plugin that helps you find and fix issues earlier in your workflow. Learn More Dismiss

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-Clone -Dsonar.host.url=http://43.204.36.242:8090'''
    }
}
```

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' }}
```

```

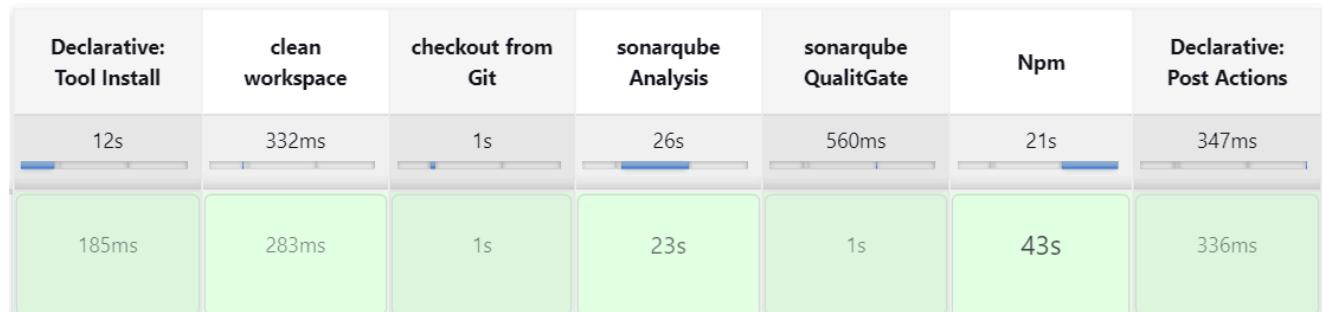
steps{
    script{
        def credentialsId = 'Sonar-token'
        qualityGate(credentialsId)
    }
}
stage('Npm'){
when { expression { params.action == 'create'}}

steps{
    npmInstall()
}
}

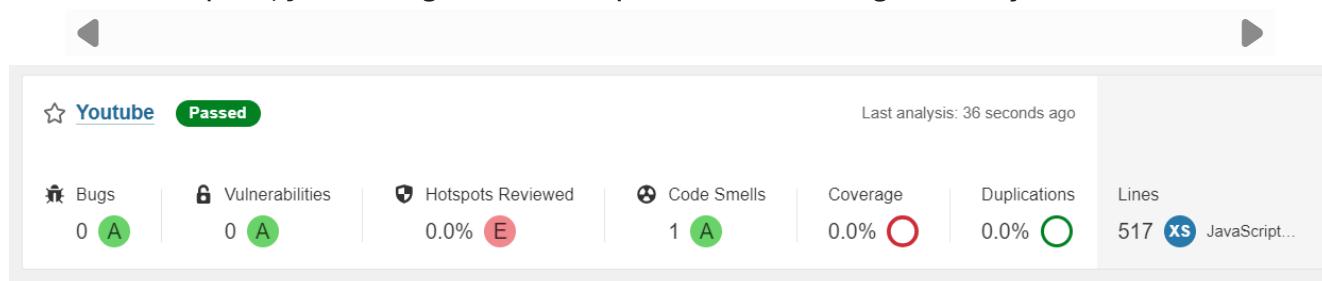
```

Build now.

Stage view



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



You can see the report has been generated and the status shows as passed. You can see that there are 517 lines scanned. To see a detailed report, you can go to issues.

Slack Notification

The screenshot shows a Jenkins job history for a job named "Job Youtube". It displays four builds:

- Build 1: SUCCESS at 12:17. More info at <http://43.205.116.178:8080/job/Youtube/1/>
- Build 2: SUCCESS at 12:39. More info at <http://43.205.116.178:8080/job/Youtube/2/>
- Build 3: FAILURE at 12:43. More info at <http://43.205.116.178:8080/job/Youtube/3/>
- Build 4: SUCCESS at 12:43. More info at <http://43.205.116.178:8080/job/Youtube/4/>

Step7: Install OWASP Dependency Check Plugins

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

The screenshot shows the Jenkins Plugins page under the "Available plugins" tab. The "OWASP Dependency-Check" plugin is selected for installation. The details shown are:

- Name: OWASP Dependency-Check 5.4.2
- Released: 8 days 17 hr ago
- Category: Security
- Sub-categories: DevOps, Build Tools, Build Reports
- 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.

First, we configured the Plugin and next, we had to configure the Tool

Goto Dashboard → Manage Jenkins → Tools →

[Dashboard](#) > [Manage Jenkins](#) > [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 ▾](#)

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 ./ --disab'
        dependencyCheckPublisher pattern: '**/dependency-check-i
    }
}
```

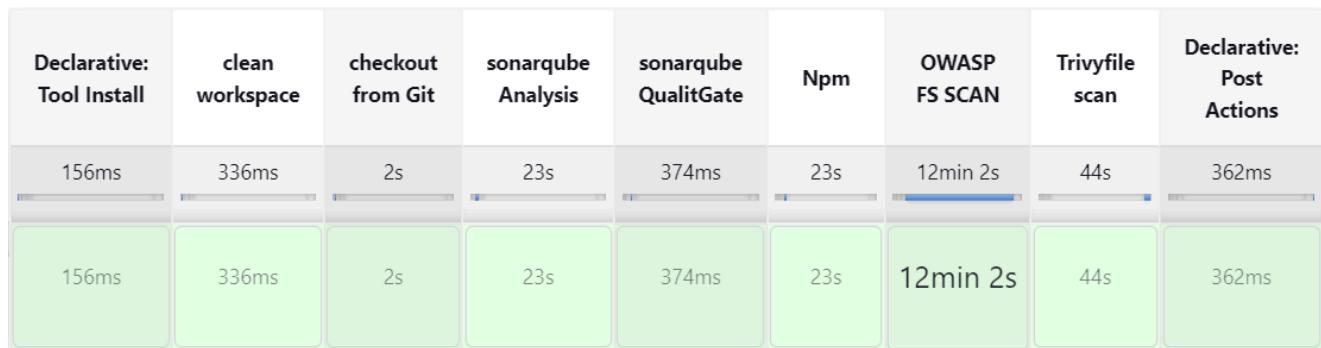


Stage with the Dependency Check steps cannot be directly used inside a shared library.

The main reason is that pipelines loaded from shared libraries have more restrictive script security by default. So the `dependencyCheck` and `dependencyCheckPublisher` steps would fail with rejected signature errors.

Build now

Stage view



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

Dependency-Check Results

Severity Distribution			
6	15	5	
File Name	Vulnerability	Severity	Weakness
+ css-what:3.4.2	OSSINDEX CVE-2022-21222	High	CWE-1333
+ ejs:3.1.8	OSSINDEX CVE-2023-29827	High	CWE-74
+ 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
+ loader-utils:2.0.2	NVD CVE-2022-37601	Critical	CWE-1321
+ loader-utils:2.0.2	NVD CVE-2022-37603	High	CWE-1333
+ loader-utils:3.2.0	NVD CVE-2022-37599	High	CWE-1333
+ loader-utils:3.2.0	NVD CVE-2022-37603	High	CWE-1333

Step8A: Docker Image Build and Push

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

Docker

Docker Commons

Docker Pipeline

Docker API

docker-build-step

and click on install without restart

The screenshot shows the Jenkins Plugins page with a search bar for "docker". The results list three plugins:

- Docker 1.5** (Cloud Providers, Cluster Management, docker) - Last updated 3 days 15 hr ago. Status: Released. Install button.
- Docker Commons 439.va_3cb_0a_6a_fb_29** (Library plugins (for use by other plugins), docker) - Last updated 1 mo 29 days ago. Status: Released.
- Docker Pipeline 572.v950f58993843** (pipeline, DevOps, Deployment, docker) - Last updated 27 days ago. Status: Released. This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.
- Docker API 3.3.1-79.v20b_53427e041** (Library plugins (for use by other plugins), docker) - Last updated 3 mo 4 days ago. Status: Released. This plugin provides `docker-java` API for other plugins.

Now, goto Dashboard → Manage Jenkins → Tools →

The screenshot shows the Jenkins Tools page under Manage Jenkins → Tools. It displays the "Docker installations" section with a form to add a new Docker installation:

- Add Docker** button.
- Docker** configuration section:
 - Name**: docker
 - Install automatically** checkbox (checked).
 - Download from docker.com** configuration section:
 - Docker version**: latest
 - Add Installer** dropdown menu.

Add DockerHub Username and Password under Global Credentials

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

Kind

Username with password

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

Username ?
sevenajay

Treat username as secret ?

Password ?
.....

ID ?
docker

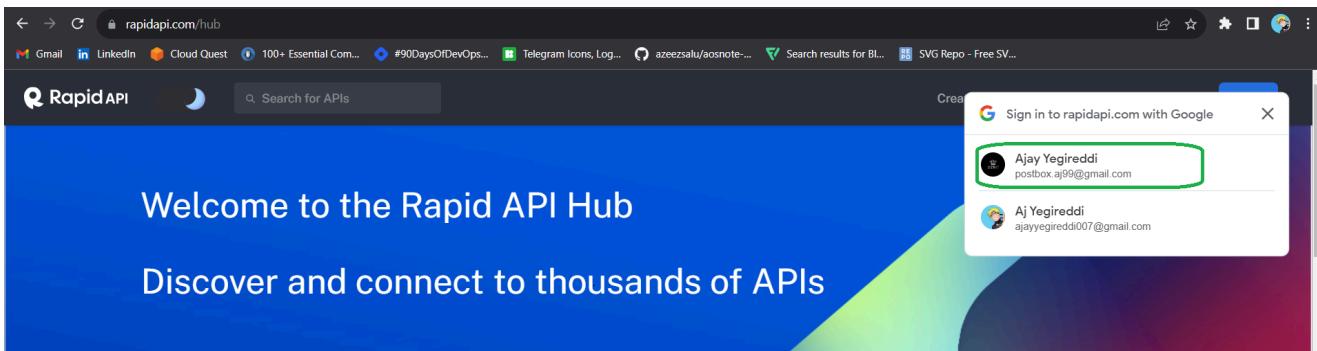
Description ?
docker

Create

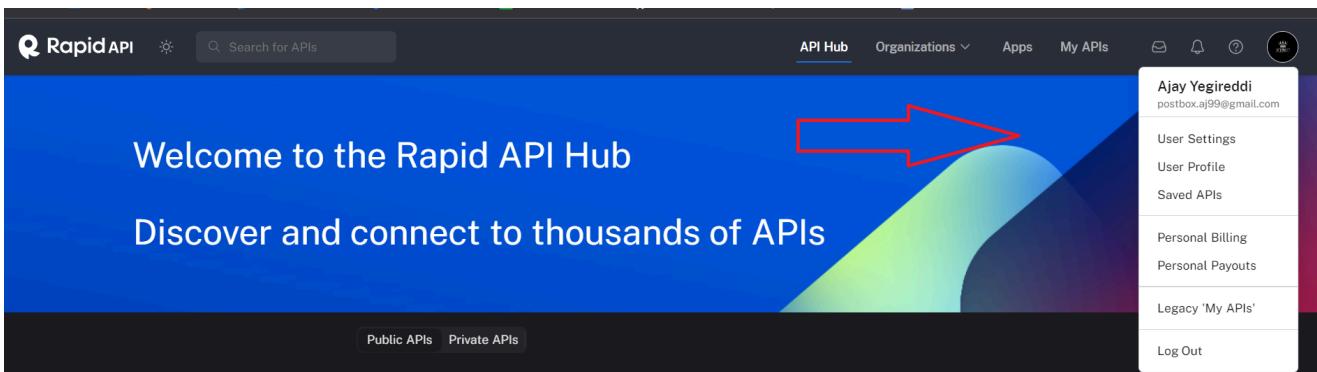
Step8B: Create an API key from Rapid API

Open a new tab in the browser and search for rapidapi.com

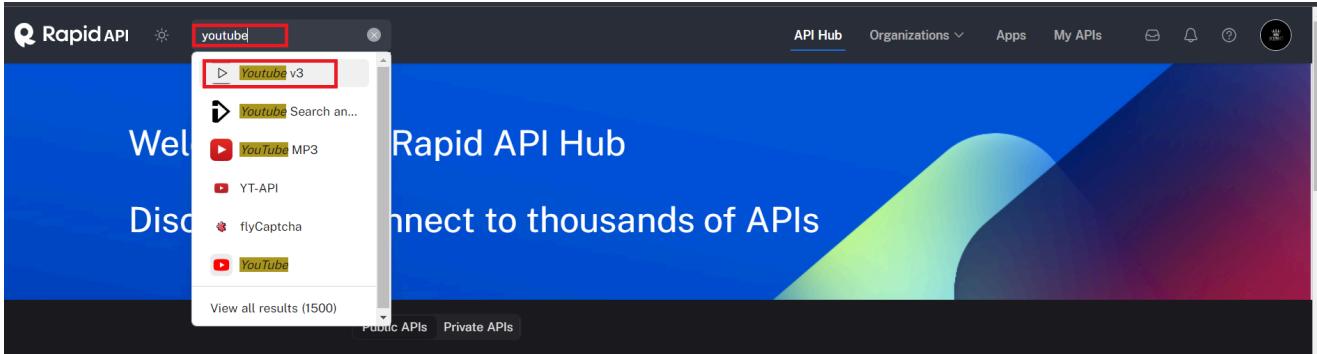
It will automatically provide your mail and select a mail to create an account



Account is created

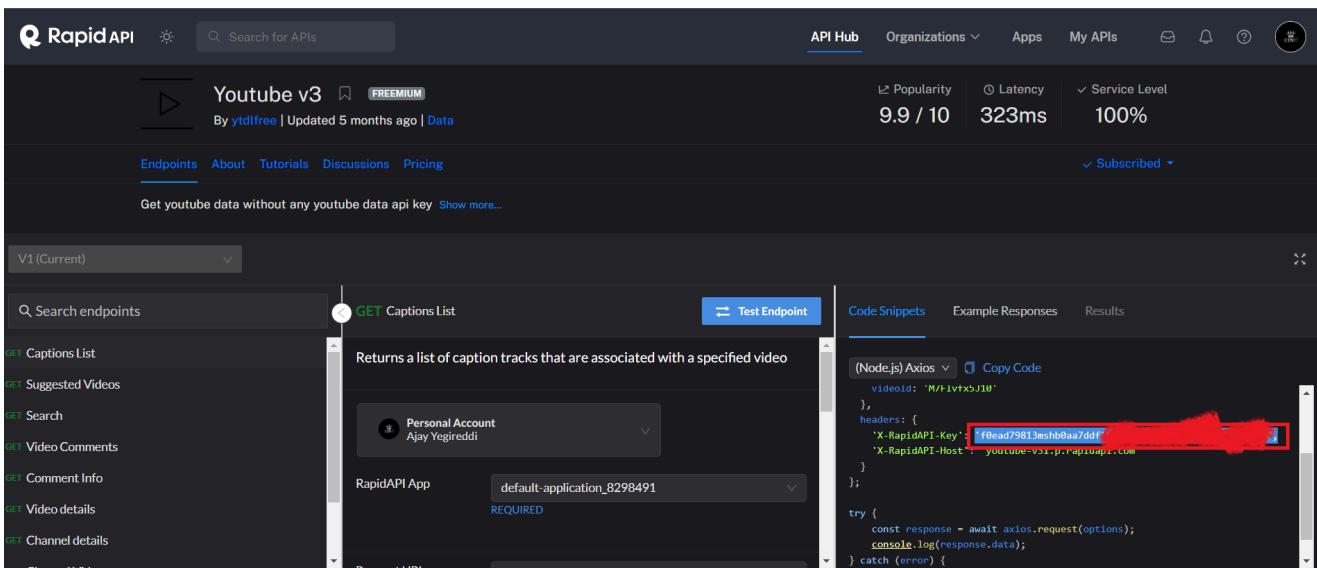


Now in the search bar search for YouTube and select YouTube v3



Copy API and use it in the groovy file

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



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=f0ead79813mshb6aa7ddf"
    // Tag the Docker image
    sh "docker tag ${imageName} ${dockerHubUsername}/${imageName}:latest"
    // Push the Docker image
    withDockerRegistry([url: 'https://index.docker.io/v1/', credentialsId: 'Docker Registry']):
        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

Pipeline Youtube

This build requires parameters:

action
Select create or destroy.
create

DOCKER_HUB_USERNAME
Docker Hub Username
sevenajay

IMAGE_NAME
Docker Image Name
youtube

Build History trend ▾ **Build** Cancel

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

sevenajay / youtube

Description
This repository does not have a description

Last pushed: 3 minutes ago

Docker commands
To push a new tag to this repository:
docker push sevenajay/youtube:tagname

Public View

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'

Dashboard > Youtube >

Pipeline Youtube

Status

Changes

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Splunk

SonarQube

Rename

Pipeline Syntax

DOCKER_HUB_USERNAME

Docker Hub Username

sevenajay

IMAGE_NAME

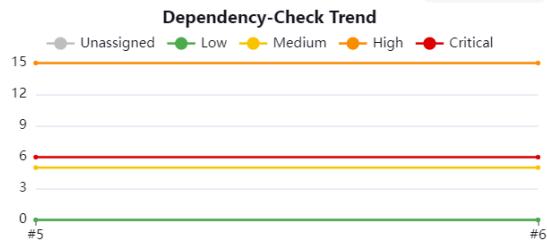
Docker Image Name

youtube

Build History trend ▾

Build Cancel

Stage view



Declarative: Tool Install	clean workspace	checkout from Git	sonarqube Analysis	sonarqube QualitGate	Npm	OWASP FS SCAN	Trivyfile scan	Docker Build	Trivy iamge	Run container	Remove container	Declarative: Post Actions
159ms	371ms	1s	25s	346ms	21s	7min 41s	40s	5min 40s	3min 7s	1s	0ms	366ms
159ms	371ms	1s	25s	346ms	21s	7min 41s	40s	5min 40s	3min 7s	1s		366ms

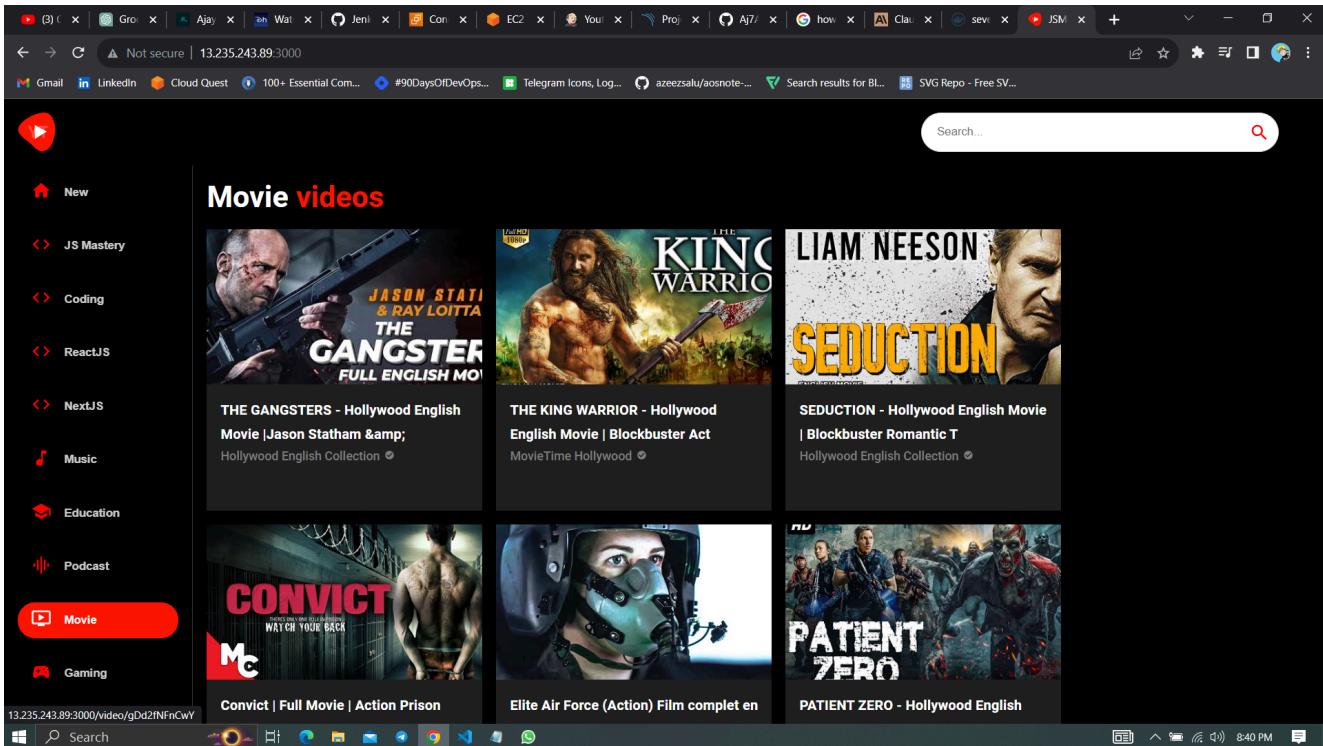
It will start the container



<public-ip-jenkins:3000>



Output:



Build with parameters 'delete'

Dashboard > Youtube >

Pipeline Youtube

This build requires parameters:

- action
- Select create or destroy.
- delete

DOCKER_HUB_USERNAME

Docker Hub Username

sevenajay

IMAGE_NAME

Docker Image Name

youtube

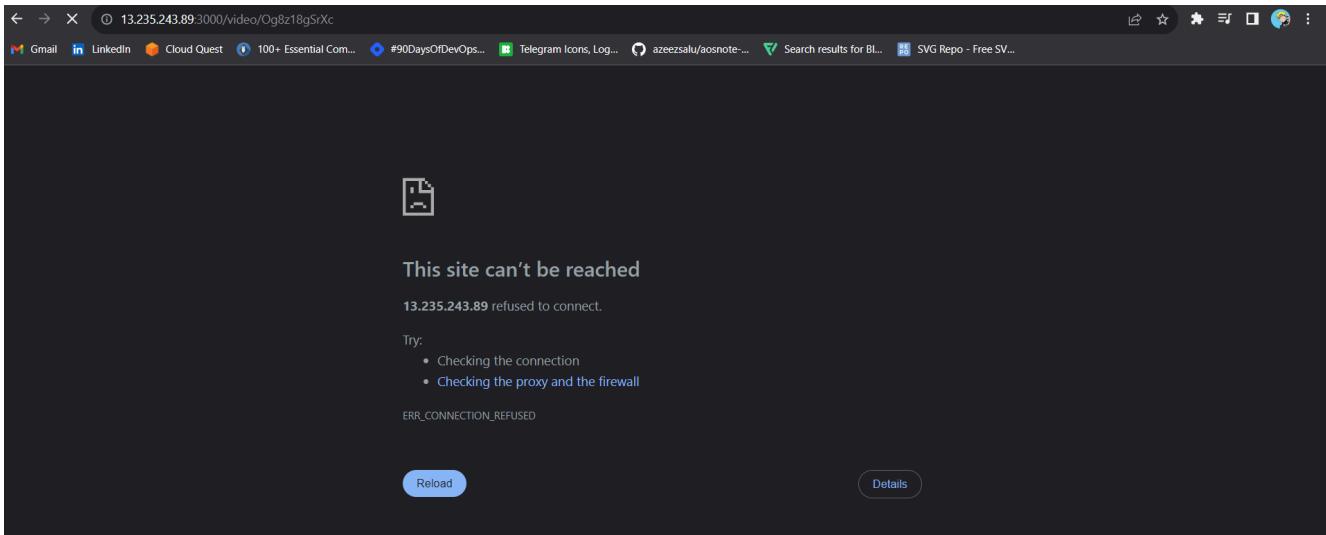
Build History trend ▾

Build Cancel

Filter builds /

It will stop and remove the Container

Declarative: Tool Install	clean workspace	checkout from Git	sonarqube Analysis	sonarqube QualitGate	Npm	Trivy file scan	Docker Build	Trivy iamge	Run container	Remove container
125ms	281ms	1s	24s	314ms	24s	2s	2min 42s	2min 8s	1s	2s
133ms	283ms		SKIPPED SOME STAGES BECUASE WE USED DELETE PARAMETER & REMOVED CONTAINER						2s	



Step9A: Kubernetes Setup

Connect your machines to Putty or Mobaxtreme

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/releas  
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
```



```
ubuntu@ip-172-31-45-224:~$  
ubuntu@ip-172-31-45-224:~$ sudo hostnamectl set-hostname K8s-Master  
ubuntu@ip-172-31-45-224:~$ exec bash  
ubuntu@K8s-Master:~$ █
```

-----Worker Node-----

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



```
ubuntu@ip-172-31-42-176:~$  
ubuntu@ip-172-31-42-176:~$ sudo hostnamectl set-hostname K8s-worker  
ubuntu@ip-172-31-42-176:~$ exec bash  
ubuntu@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 apt-get update
# apt-transport-https may be a dummy package; if so, you can skip that part
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor - > /etc/apt/trusted.gpg.d/kubernetes.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
sudo snap install kube-apiserver

```



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



```

sudo kubeadm init --pod-network-cidr=10.244.0.0/16
# in case your 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-flanneld-deployment.yaml

```



-----Worker Node-----



```

sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discovery-token-ca-cert-hash sha256:342823611225701786a93d19e063b06a8e03470b7658ccb1d98ac02e94d32c07

```



Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 172.31.37.230:6443 --token z2rpwv.mzu6q6k0j1y36z6y \
--discovery-token-ca-cert-hash sha256:342823611225701786a93d19e063b06a8e03470b7658ccb1d98ac02e94d32c07
```

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

A screenshot of a Windows desktop environment. At the top, there's a taskbar with icons for File Explorer, Task View, Start, and several pinned applications. Below the taskbar, there are two rows of open browser tabs. The first row includes 'Quick connect...', '1 / drives/c/Users/Admin/Downloads', and '2 / drives/c/Users/Admin/Downloads'. The second row includes '3 / drives/c/Users/Admin/Downloads'. In the center of the screen, there's a large terminal window titled 'Terminal' with a black background and white text. The terminal output is a long command-line session, likely a penetration test or exploit development process. It includes various commands like 'cat', 'apt-get', 'curl', and 'nc' along with complex JSON and XML payloads. The bottom of the screen shows the Windows taskbar again with various pinned apps like File Explorer, Task View, Start, and others.

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 (selected) Installed plugins Advanced settings Download progress

Kuber / Install Refresh

Install	Name	Released
<input checked="" type="checkbox"/>	Kubernetes Credentials 0.11 kubernetes credentials Common classes for Kubernetes credentials	9 days 16 hr ago
<input checked="" type="checkbox"/>	Kubernetes Client API 6.8.1-224.vd388fca_4db_3pb... 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"/>	Kubernetes 4029.v571223ccb_f8 Cloud Providers Cluster Management kubernetes Agent Management This plugin integrates Jenkins with Kubernetes	9 days 15 hr ago
<input checked="" type="checkbox"/>	Kubernetes CLI 1.12.1 kubernetes Configure kubectl for Kubernetes	8 days 22 hr ago

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

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

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

Step9D: Install Helm & Monitoring K8S using Prometheus and Grafana

On Kubernetes Master install the helm



```
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
```



```
ubuntu@Master:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@Master:~$ ls
get_helm.sh
ubuntu@Master:~$
```

```
ubuntu@Master:~$ sudo chmod 700 get_helm.sh
ubuntu@Master:~$ ls -la
total 56
drwxr-xr-x 5 ubuntu ubuntu 4096 Oct 14 17:30 .
drwxr-xr-x 3 root  root 4096 Oct  9 06:57 ..
-rw----- 1 ubuntu ubuntu 114 Oct 14 17:30 .Xauthority
-rw----- 1 ubuntu ubuntu 1627 Oct 14 15:24 .bash_history
-rw-r--r-- 1 ubuntu ubuntu 220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Feb 25 2020 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Oct  9 07:00 .cache
drwxrwxr-x 3 docker  docker 4096 Oct  9 07:08 .kube
-rw-r--r-- 1 ubuntu ubuntu 807 Feb 25 2020 .profile
drwx----- 2 ubuntu ubuntu 4096 Oct  9 06:57 .ssh
-rw-r--r-- 1 ubuntu ubuntu 0 Oct  9 07:00 .sudo_as_admin_successful
-rw-rw-r-- 1 docker  docker 165 Oct  9 07:12 .wget-hsts
-rwx----- 1 ubuntu ubuntu 11715 Oct 14 17:30 get_helm.sh
ubuntu@Master:~$
```

```
ubuntu@Master:~$ sudo ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.13.1-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ubuntu@Master:~$
```

See the Helm version



helm version --client



```
ubuntu@Master:~$ helm version --client
version.BuildInfo{Version:"v3.13.1", GitCommit:"3547a4b5bf5edb5478ce352e18858d8a552a4110", GitTreeState:"clean", GoVersion:"go1.20.8"}
ubuntu@Master:~$
```

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
```



```
ubuntu@Master:~$  
ubuntu@Master:~$ helm repo add stable https://charts.helm.sh/stable  
"stable" has been added to your repositories  
ubuntu@Master:~$
```

Add Prometheus Helm repo



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



```
ubuntu@Master:~$  
ubuntu@Master:~$  
ubuntu@Master:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts  
"prometheus-community" has been added to your repositories  
ubuntu@Master:~$
```

Create Prometheus namespace



```
kubectl create namespace prometheus
```



```
ubuntu@Master:~$  
ubuntu@Master:~$ kubectl create namespace prometheus  
namespace/prometheus created  
ubuntu@Master:~$
```

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
```



```
ubuntu@Master:~$ helm install stable prometheus-community/kube-prometheus-stack -n prometheus
NAME: stable
LAST DEPLOYED: Sat Oct 14 17:33:54 2023
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@Master:~$
```

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



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



```
ubuntu@ip-172-31-47-213:~$ kubectl get pods -n prometheus
NAME                                         READY   STATUS    RESTARTS   AGE
alertmanager-stable-kube-prometheus-sta-alertmanager-0   2/2     Running   0          37s
prometheus-stable-kube-prometheus-sta-prometheus-0       2/2     Running   0          37s
stable-grafana-688dd8b9f7-2rrbm                   3/3     Running   0          52s
stable-kube-prometheus-sta-operator-79698d5c9f-djcwl    1/1     Running   0          52s
stable-kube-state-metrics-859f6c9466-g2zsv           1/1     Running   0          52s
stable-prometheus-node-exporter-85ghg                1/1     Running   0          52s
stable-prometheus-node-exporter-d529q                1/1     Running   0          52s
ubuntu@ip-172-31-47-213:~$
```

Now See the services



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



```
ubuntu@ip-172-31-47-213:~$  
ubuntu@ip-172-31-47-213:~$  
ubuntu@ip-172-31-47-213:~$ kubectl get svc -n prometheus  
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE  
alertmanager-operated  ClusterIP  <none>       9093/TCP,9094/TCP,9094/UDP  53s  
prometheus-operated  ClusterIP  <none>       9090/TCP                53s  
stable-grafana        ClusterIP  10.97.78.239  <none>       80/TCP     69s  
stable-kube-prometheus-sta-alertmanager  ClusterIP  10.98.194.55  <none>       9093/TCP,8080/TCP  69s  
stable-kube-prometheus-sta-operator       ClusterIP  10.108.220.28  <none>       443/TCP    69s  
stable-kube-prometheus-sta-prometheus   ClusterIP  10.107.180.169 <none>       9090/TCP,8080/TCP  69s  
stable-kube-state-metrics      ClusterIP  10.111.13.212  <none>       8080/TCP   69s  
stable-prometheus-node-exporter        ClusterIP  10.106.128.227 <none>       9100/TCP   69s  
ubuntu@ip-172-31-47-213:~$ █
```

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
```



```

meta.helm.sh/release-name: stable
meta.helm.sh/release-namespace: prometheus
creationTimestamp: "2023-10-14T17:34:10Z"
labels:
  app: kube-prometheus-stack-prometheus
  app.kubernetes.io/instance: stable
  app.kubernetes.io/managed-by: Helm
  app.kubernetes.io/part-of: kube-prometheus-stack
  app.kubernetes.io/version: 51.7.0
  chart: kube-prometheus-stack-51.7.0
  heritage: Helm
  release: stable
  self-monitor: "true"
name: stable-kube-prometheus-sta-prometheus
namespace: prometheus
resourceVersion: "6596"
uid: 6205569e-3967-41be-9366-59b70b7abce9
spec:
  clusterIP: 10.97.53.48
  clusterIPs:
    - 10.97.53.48
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: http-web
      port: 9090
      protocol: TCP
      targetPort: 9090
    - appProtocol: http
      name: reloader-web
      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: {}
-- INSERT --

```



```

ubuntu@Master:~$ kubectl edit svc stable-kube-prometheus-sta-prometheus -n prometheus
service/stable-kube-prometheus-sta-prometheus edited
ubuntu@Master:~$ 

```

Edit Grafana Service



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



```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  annotations:
    meta.helm.sh/release-name: stable
    meta.helm.sh/release-namespace: prometheus
  creationTimestamp: "2023-10-14T17:34:10Z"
  labels:
    app.kubernetes.io/instance: stable
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: grafana
    app.kubernetes.io/version: 10.1.4
    helm.sh/chart: grafana-6.60.4
  name: stable-grafana
  namespace: prometheus
  resourceVersion: "6603"
  uid: 6ff89333-2743-428d-9c78-b8ec727a6261
spec:
  clusterIP: 10.102.79.119
  clusterIPs:
  - 10.102.79.119
  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
status:
  loadBalancer: {}
~
-- INSERT --
```



```
ubuntu@Master:~$ 
ubuntu@Master:~$ 
ubuntu@Master:~$ kubectl edit svc stable-grafana -n prometheus
service/stable-grafana edited
ubuntu@Master:~$ 
```

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



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	3m41s
prometheus-operated	ClusterIP	None	<none>	9090/TCP	3m41s
stable-grafana	LoadBalancer	10.102.79.119	<pending>	80:32525/TCP	3m49s
stable-kube-prometheus-sta-alertmanager	ClusterIP	10.104.104.67	<none>	9093/TCP, 8080/TCP	3m49s
stable-kube-prometheus-sta-operator	ClusterIP	10.104.152.148	<none>	443/TCP	3m49s
stable-kube-prometheus-sta-prometheus	LoadBalancer	10.97.53.48	<pending>	9090:30549/TCP, 8080:30105/TCP	3m49s
stable-kube-state-metrics	ClusterIP	10.106.45.195	<none>	8080/TCP	3m49s
stable-prometheus-node-exporter	ClusterIP	10.101.188.117	<none>	9100/TCP	3m49s

Access Grafana UI in the browser

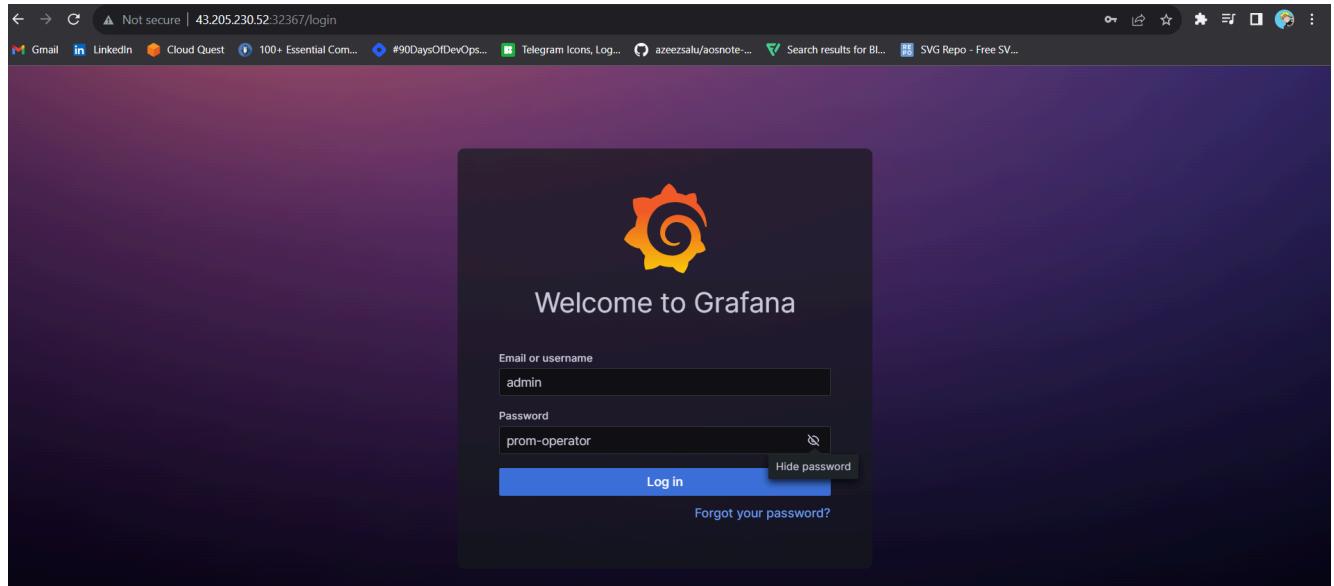
Get the external IP from the above screenshot and put it in the browser



<k8s-master-public-ip:external-ip>



```
ubuntu@ip-172-31-47-213:~$ 
ubuntu@ip-172-31-47-213:~$ kubectl get svc -n prometheus
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP     PORT(S)          AGE
alertmanager-operated   ClusterIP  <none>       9093/TCP,9094/TCP,9094/UDP   2m48s
prometheus-operated    ClusterIP  <none>       9090/TCP      9090/TCP        2m48s
stable-grafana         LoadBalancer 10.97.78.239  <pending>     80:32367/TCP      3m4s
stable-kube-prometheus-sta-alertmanager   ClusterIP  10.98.194.55  <none>        9093/TCP,8080/TCP   3m4s
stable-kube-prometheus-sta-operator        ClusterIP  10.108.220.28  <none>        443/TCP          3m4s
stable-kube-prometheus-sta-prometheus    LoadBalancer 10.107.180.169 <pending>    9090:30650/TCP,8080:32643/TCP  3m4s
stable-kube-state-metrics               ClusterIP  10.111.13.212  <none>        8080/TCP          3m4s
stable-prometheus-node-exporter        ClusterIP  10.106.128.227 <none>        9100/TCP          3m4s
ubuntu@ip-172-31-47-213:~$
```



Login to Grafana



UserName: admin

Password: prom-operator



Create a Dashboard in Grafana

In Grafana, we can create various kinds of dashboards as per our needs.

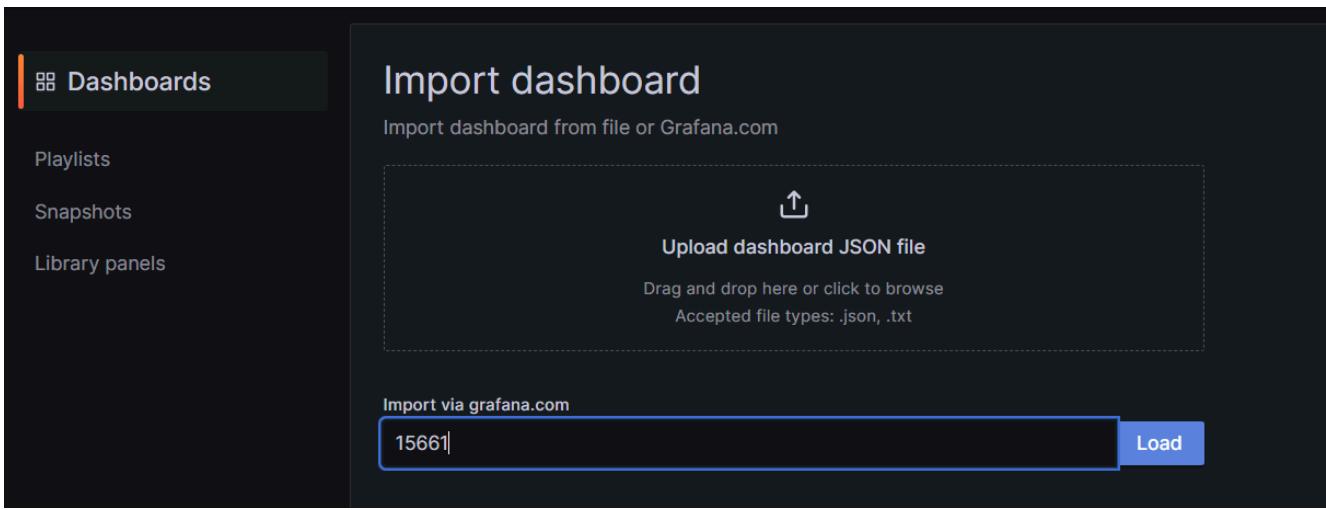
How to Create Kubernetes Monitoring Dashboard?

For creating a dashboard to monitor the cluster:

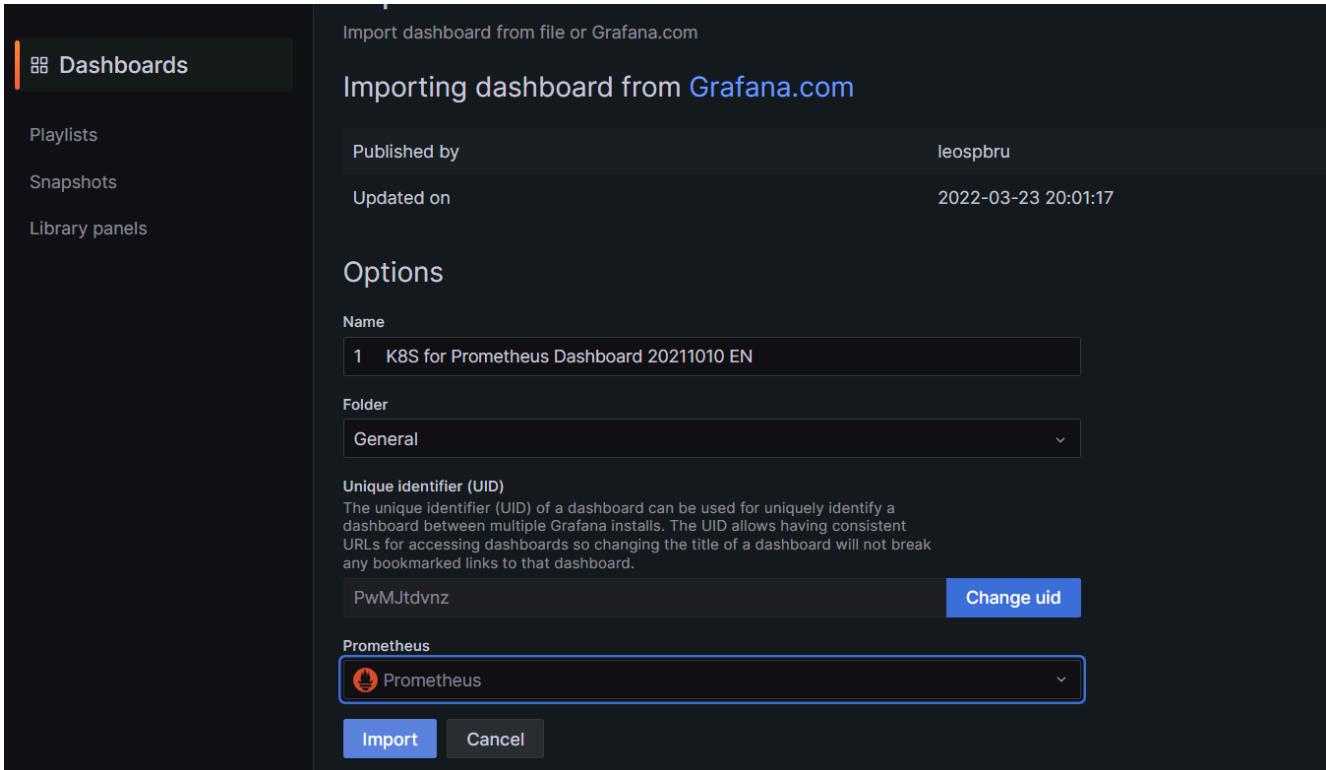
Click the '+' button on the left panel and select 'Import'.

Enter the 15661 dashboard id under [Grafana.com](#) Dashboard.

Click 'Load'.

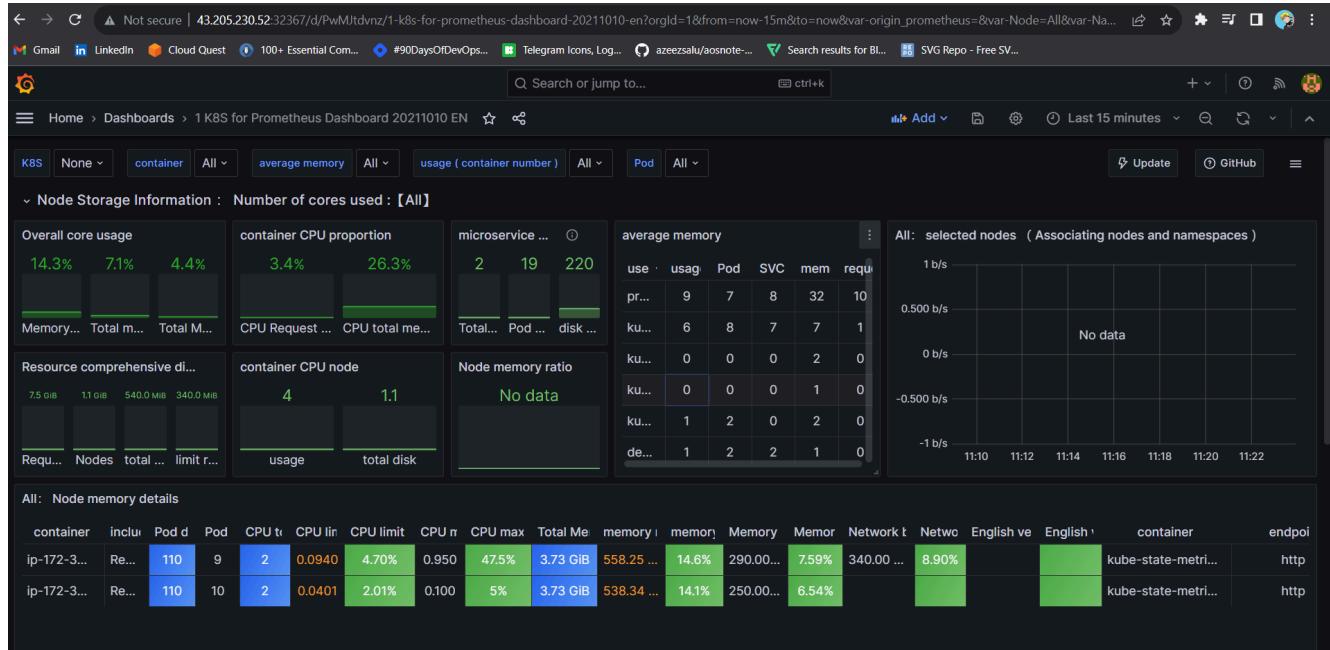


Select 'Prometheus' as the endpoint under the Prometheus data sources drop-down.



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'.

This will show the monitoring dashboard for all cluster nodes



Create a POD Monitoring Dashboard

For creating a dashboard to monitor the cluster:

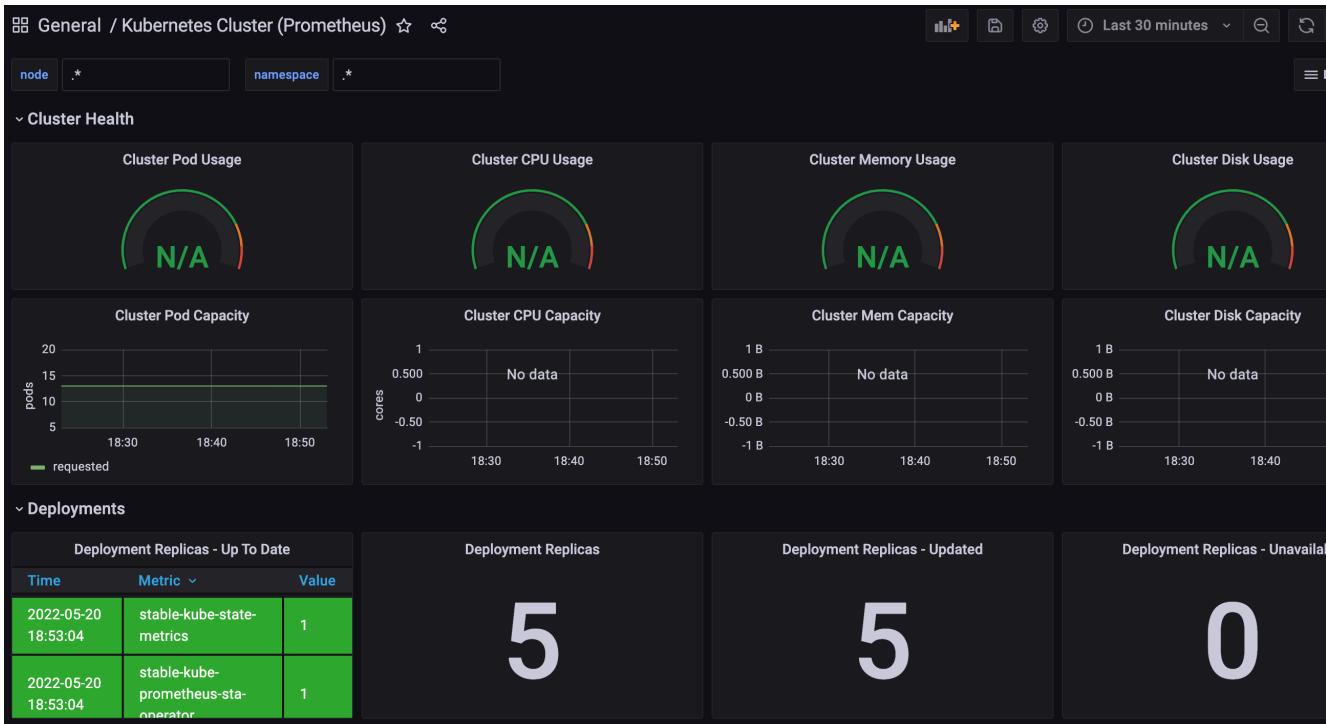
Click the '+' button on the left panel and select 'Import'.

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: '',
        sh "kubectl apply -f deployment.yml"
    )
}
```

To delete deployment

Name **kubeDelete.groovy**

```
def call() {
    withKubeConfig(caCertificate: '', clusterName: '', contextName: ''
```

```
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'

Dashboard > Youtube >

Pipeline Youtube

Status

Changes

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Splunk

SonarQube

Rename

Pipeline Syntax

This build requires parameters:

action

Select create or destroy.

DOCKER_HUB_USERNAME

Docker Hub Username

sevenajay

IMAGE_NAME

Docker Image Name

youtube

Build History

trend

Build Cancel

It will apply the deployment

stage view

Declarative: Tool Install	clean workspace	checkout from Git	sonarqube Analysis	sonarqube QualitGate	Npm	OWASP FS SCAN	Trivyfile scan	Docker Build	Trivy iamge	Run container	Remove container	Kube deploy	kube deleter	Declarative: Post Actions
152ms	330ms	1s	25s	351ms	24s	6min 26s	40s	2min 35s	3min 22s	1s	0ms	3s	0ms	350ms
152ms	330ms	1s	25s	351ms	24s	6min 26s	40s	2min 35s	3min 22s	1s		3s		350ms



kubectl get all (or)
kubectl get svc



<kubernetes-worker-ip:svc port>



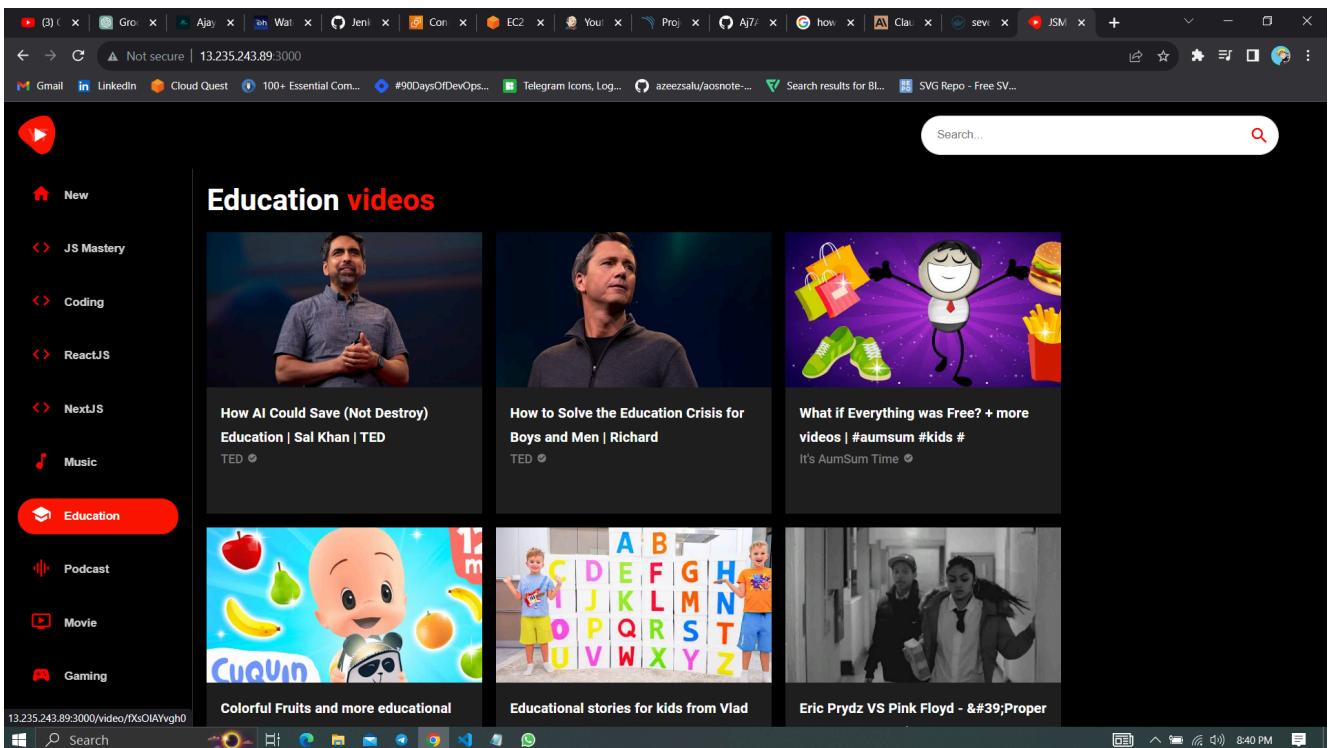
```
ubuntu@K8s-master:~$ kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/my-app-deployment-79f9d49c54-qxf2q   1/1     Running   0          3m10s

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP      58m
service/my-app-service   NodePort    10.111.246.72  <none>        80:32089/TCP  3m10s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-app-deployment   1/1       1           1          3m10s

NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/my-app-deployment-79f9d49c54   1         1         1          3m10s
ubuntu@K8s-master:~$
```

output:



The screenshot shows a web browser window with the URL <https://13.235.243.89:3000/>. The page title is "YouTube Clone App with DevSecOps and Jenkins Shared Library - Mr Cloud Book". On the left, there is a sidebar with various categories: New, JS Mastery, Coding, ReactJS, NextJS, Music (which is highlighted in red), Education, Podcast, Movie, and Gaming. The main content area is titled "Music videos" and displays four video thumbnails:

- Relaxing Music For Stress Relief, Anxiety and Depressive Sta** by Open Heart Music - Helios 4K
- Rema - Charm (Official Music Video)** by Rema
- Relaxing Music For Stress Relief, Anxiety and Depressive Sta** by Open Heart Music - Helios 4K
- WEDNESDAY MORNING JAZZ: Smooth** by RAGMC Records

Below the thumbnails, there is a list titled "Top Hits 2023" with many song titles. The bottom of the screen shows the Windows taskbar with various pinned icons.

The screenshot shows a web browser window with the URL <https://3.11.31.186:32232/>. The page title is "YouTube Clone App with DevSecOps and Jenkins Shared Library - Mr Cloud Book". On the left, there is a sidebar with categories: NextJS, Music, Education, Podcast, Movie, Gaming, Live, Sport, and Fashion (which is highlighted in red). The main content area is titled "Fashion videos" and displays three video thumbnails:

- Britney Manson - FASHION (Single, 2023)** by Britney Manson
- Britney Manson - FASHION (Lyrics)** by Blissful Mind
- britney manson - FASHION (sped up)** by mel

Below the thumbnails, there are two more rows of fashion-related content:

- 10 Wearable Fall Shoe Trends | 2023**
- WHAT TO WEAR - WINTER 2023!**
- Britney Manson - Fashion (Extended)**

The bottom of the screen shows the Windows taskbar with various pinned icons.

Build with parameter 'delete'

It will destroy Container and Kubernetes deployment.

Declarative: Tool Install	clean workspace	checkout from Git	sonarqube Analysis	sonarqube QualitGate	Npm	OWASP FS SCAN	Trivyfile scan	Docker Build	Trivy iamge	Run container	Remove container	Kube deploy	kube deleter	Declarative: Post Actions
157ms	338ms	1s	25s	351ms	24s	6min 26s	40s	2min 35s	3min 22s	1s	1s	3s	604ms	346ms
162ms	346ms	2s									1s		604ms	343ms
152ms	330ms	1s	25s	351ms	24s	6min 26s	40s	2min 35s	3min 22s	1s		3s		350ms

Slack Notifications

The screenshot shows a Slack channel named '# Jenkins-youtube'. The sidebar on the left includes sections for Threads, Canvases, Slack Connect, Files, More, Channels (# ajay, # general, # jenkins, # Jenkins-youtube), Add channels, Direct messages, and Add colleagues. The '# Jenkins-youtube' channel has the following messages:

- Ajay Yegireddi joined #jenkins-youtube at 11:49.
- Ajay Yegireddi added an integration to this channel: jenkins at 11:52.
- Jenkins (APP) Slack/Jenkins plugin: you're all set on <http://43.205.116.178:8080/> at 11:55.
- Jenkins (APP) SUCCESS: Job Youtube build 1 More info at: <http://43.205.116.178:8080/job/Youtube/1/> at 12:17.
- Jenkins (APP) SUCCESS: Job Youtube build 2 More info at: <http://43.205.116.178:8080/job/Youtube/2/> at 12:39.
- Jenkins (APP) FAILURE: Job Youtube build 3 More info at: <http://43.205.116.178:8080/job/Youtube/3/> at 12:39.
- Jenkins (APP) SUCCESS: Job Youtube build 4 More info at: <http://43.205.116.178:8080/job/Youtube/4/> at 12:39.
- Jenkins (APP) SUCCESS: Job Youtube build 5 More info at: <http://43.205.116.178:8080/job/Youtube/5/> at 13:04.
- Jenkins (APP) SUCCESS: Job Youtube build 6 More info at: <http://43.205.116.178:8080/job/Youtube/6/> at 13:26.
- Jenkins (APP) SUCCESS: Job Youtube build 7 More info at: <http://43.205.116.178:8080/job/Youtube/7/> at 13:42.
- Jenkins (APP) SUCCESS: Job Youtube build 8 More info at: <http://43.205.116.178:8080/job/Youtube/8/> at 13:49.
- Jenkins (APP) SUCCESS: Job Youtube build 9 More info at: <http://43.205.116.178:8080/job/Youtube/9/> at 14:08.

Splunk

splunk>enterprise Apps ▾

Overview Audit Trail Jenkins Health Jenkins Nodes Configuration Alerts Search

Jenkins Master Select...

Jenkins Build Status History

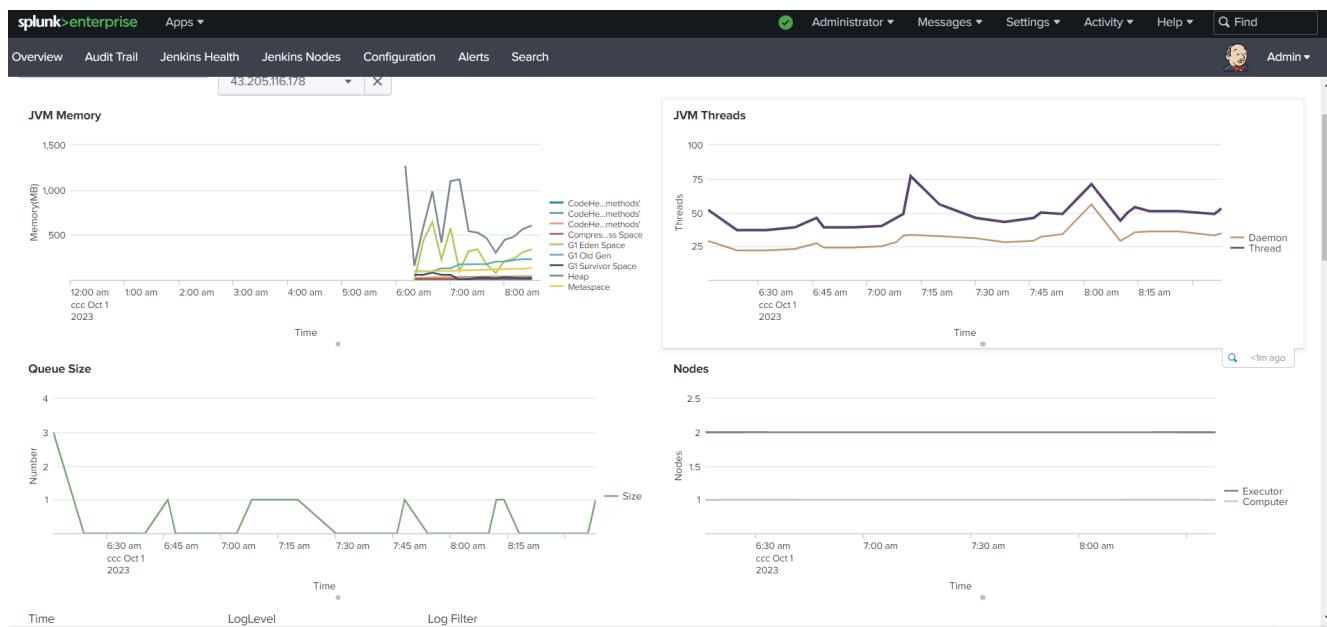
Last 7 days ▾

On Oct 9 Tue Oct 10 Wed Oct 11 Thu Oct 12 Fri Oct 13 Sat Oct 14 Sun Oct 15 Mon Oct 16

Latest Builds

Today ▾

Jenkins Master	Job	Build	StartTime	Duration	Status
43.205.116.178	Youtube	9	2023-10-16 08:38:10	00:00:09.24	✓
43.205.116.178	Youtube	8	2023-10-16 08:15:44	00:14:05.243	✓
43.205.116.178	Youtube	7	2023-10-16 08:12:29	00:00:10.94	✓
43.205.116.178	Youtube	6	2023-10-16 07:48:51	00:18:04.618	✓
43.205.116.178	Youtube	5	2023-10-16 07:20:35	00:13:40.041	✓



Pipeline



```
@Library('Jenkins_shared_library') _
def COLOR_MAP = [
    'FAILURE' : 'danger',
    'SUCCESS' : 'good'
]
pipeline{
    agent any
    parameters {
        choice(name: 'action', choices: 'create\ndelete', description:
        string(name: 'DOCKER_HUB_USERNAME', defaultValue: 'sevenajay', (
        string(name: 'IMAGE_NAME', defaultValue: 'youtube', description

```

```
}

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/Aj7Ay/Youtube-clone-app')
        }
    }
    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 ./ --disab:
        dependencyCheckPublisher pattern: '**/dependency-check-i
    }
}

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: '#channel name',    #change your channel name
            color: COLOR_MAP[currentBuild.currentResult],
            message: "*${currentBuild.currentResult}:* Job ${env.JOB_NAME} - ${currentBuild.result} - ${currentBuild.durationSec}s"
        )
    }
}
}
```

In conclusion, deploying a YouTube clone app on Docker and Kubernetes with a strong DevSecOps approach and Jenkins Shared Library is a powerful combination that can take your application deployment to the next level. It provides the security, scalability, and efficiency needed to ensure a smooth transition to production.

By implementing DevSecOps practices from the very beginning and utilizing the orchestration capabilities of Kubernetes, your app will be more secure and ready to handle the demands of your growing user base. Docker containers will give you the flexibility to move your app across different environments with ease, ensuring consistency and reducing deployment-related issues.

Jenkins Shared Library is your key to streamlined, automated, and standardized CI/CD workflows, simplifying the deployment process and making it reproducible and consistent. With these tools and practices in your arsenal, you'll be well-prepared to face the challenges of modern application deployment.

We hope this tutorial has been valuable in helping you understand the essential components of a DevSecOps-driven deployment. As you embark on your journey to deploy your own YouTube clone app, remember that learning and refining these skills takes time and practice. Don't be discouraged by challenges; instead, see them as opportunities to grow and improve your expertise.

If you found this tutorial helpful, please share it with your network, and consider subscribing to our blog or channel for more in-depth technical guides. We're here to

support you and answer any questions you may have along the way.

Thank you for joining us on this deployment journey. We wish you the best of luck in your application deployment endeavours. Stay curious, keep learning, and continue to explore the exciting world of DevSecOps, containers, and orchestration!

Here's to your success in the world of tech and DevSecOps!



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

2 responses to “YouTube Clone App with DevSecOps and Jenkins Shared Library”



manoj

13 March 2024

Hi,

This was great. I learned a lot here. Keep up the good work.

[Reply](#)



Jamal

21 December 2024

Thank you very much, I learned a lot.

[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**

**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

22 July 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](#)

© 2024 · Powered by [Mr Cloud Book](#)

[Follow Us on YouTube](#)