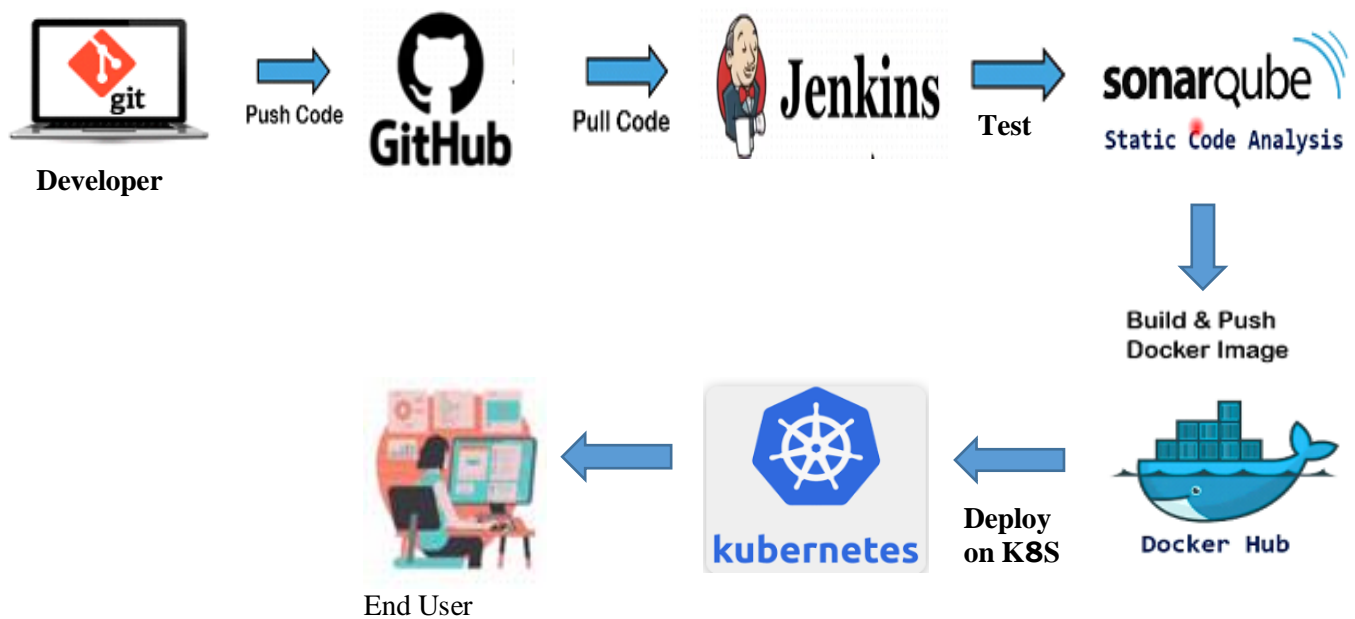


Continuous Integration & Continuous Deployment Pipeline for Web application development



Project Flow Summary

1. Development Phase (Local Development)

- The developer writes code using a local development environment.
- Version control is managed using Git.
- The developer pushes the code to GitHub, which acts as a central repository.

2. Continuous Integration (CI)

- Jenkins is configured to automatically pull the latest code from GitHub.
- Jenkins runs build jobs and executes automated tests to ensure the new changes do not break the application.

3. Code Quality & Security Checks

- SonarQube is used for static code analysis to check for bugs, vulnerabilities, and code quality issues. If issues are detected, the developer is alerted, and fixes are required before proceeding.

4. Build and Containerization

- After passing tests, Jenkins builds the application. The application is then containerized using Docker. A Docker image is built and pushed to Docker Hub, a container registry.

5. Continuous Deployment (CD)

- Once the Docker image is available, Jenkins triggers the deployment.
- The application is deployed to a Kubernetes (K8s) cluster.
- Kubernetes ensures the application is running with high availability and scalability.

6. End-User Access

- The end-user accesses the deployed web application through a browser or client interface.
- Kubernetes manages the traffic and scales the application based on demand.

Step: 1 To Create a Jenkins virtual server in AWS and login to Ubuntu linux with root user.

New EC2 Experience

Tell us what you think

Instances (3) Info

Connect

Instance state

Actions

Launch Instances

Find instance by attribute or tag (case-sensitive)

<

1

Name

Instance ID

Instance state

Instance type

Status check

Alarm status

Availability Zone

Jenkins

i-0c7f2e62a88a48681

Running

t2.medium

2/2 checks passed

No alarms

us-east-2

```
ubuntu@ip-172-31-11-179:~$ sudo su - root
root@ip-172-31-11-179:~# apt update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]
Get:9 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 B]
Get:10 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2498 kB]
Get:11 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [424 kB]
Get:12 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [16.4 kB]
Get:13 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1776 kB]
Get:14 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [249 kB]
Get:15 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 c-n-f Metadata [636 B]
```

Step: 2 -> To install Jenkins on Ubuntu linux

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

We have to install Java Development Kit before installing Jenkins on Ubuntu linux

```
~# apt search openjdk
~# apt install openjdk-17-jre-headless
```

Now the Java Development Kit has been successfully install. To install Jenkins on Ubuntu linux

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

```
root@ip-172-31-11-179:~# curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
> /usr/share/keyrings/jenkins-keyring.asc > /dev/null
root@ip-172-31-11-179:~# echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
> https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
> /etc/apt/sources.list.d/jenkins.list > /dev/null
root@ip-172-31-11-179:~# sudo apt-get update
~# sudo apt-get install jenkins
```

To check the processes active or not under Jenkins by using the below command

```
root@ip-172-31-11-179:~# ps -ef | grep jenkins
jenkins 4074      1 46 05:50 ?        00:00:38 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/
var/cache/jenkins/war --httpPort=8080
root      4468      1394  0 05:52 pts/0    00:00:00 grep --color=auto jenkins
```

To check the Jenkins installed or not by using the command # systemctl status Jenkins

We will build the Continuous Integration & Continuous Deployment Pipeline based on the five files such as Docker file, Form.html, Main.py, Pod.Yaml and requirement.txt file.

1. Main.py File

```
1 #uvicorn main:app --reload
2 from fastapi import FastAPI, Request, Form
3 from fastapi.templating import Jinja2Templates
4
5 app = FastAPI()
6 templates = Jinja2Templates(directory="/code")
7
8 @app.get("/")
9 def form_post(request: Request):
10     return templates.TemplateResponse('form.html', context={'request': request})
```

2. Requirement.txt

aiofiles==0.5.0	enrich==1.2.6	PyYAML==5.4.1
aniso8601==7.0.0	fastapi==0.68.1	requests==2.26.0
ansible-lint==5.0.7	graphene==2.1.9	rich==10.1.0
astroid==2.5.1	graphql-core==2.3.2	rsa==4.5
async-exit-stack==1.0.1	graphql-relay==2.0.1	ruamel.yaml==0.17.4
async-generator==1.10	h11==0.12.0	ruamel.yaml.clib==0.2.2
awscli==1.22.97	idna==3.2	Rx==1.6.1
boto3==1.21.42	isort==5.7.0	s3transfer==0.5.0
botocore==1.24.42	itsdangerous==1.1.0	six==1.15.0
bracex==2.1.1	Jinja2==2.11.3	starlette==0.14.2
certifi==2021.5.30	imespath==0.10.0	tenacity==7.0.0
cffi==1.14.5	lazy-object-proxy==1.5.2	toml==0.10.2
chardet==4.0.0	MarkupSafe==2.0.1	typing-extensions==3.7.4.3
charset-normalizer==2.0.4	mccabe==0.6.1	ujson==4.1.0
click==7.1.2	orjson==3.6.3	urllib3==1.26.4
colorama==0.4.3	packaging==20.9	uvicorn==0.13.4
commonmark==0.9.1	pathspec==0.8.1	watchgod==0.7
cryptography==3.4.7	pdfminer.six==20211012	wcmatch==8.1.2
dnspython==2.1.0	promise==2.3	websockets==8.1
docutils==0.15.2	py4j==0.10.9	wrapt==1.12.1
email-validator==1.1.3	pyasn1==0.4.8	yamllint==1.26.1

3. Form.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>k8 Page</title>
8 </head>
9 <body bgcolor="fRed">
10     <h1>Hello k8 Project
11 </body>
12 </html>
```

4. Docker File

```
FROM python:3.9
WORKDIR /code
COPY ./requirements.txt /code/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
COPY ./main.py /code/main.py
COPY ./form.html /code/form.html
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

5. Pod.Yaml

```
apiVersion: apps/V1
kind: Deployment
metadata:
  name: loadbalancer-pod
Spec:
  replicas: 3
  selector:
    matchLabels:
      app: loadbalancer-pod
  template:
    metadata:
      labels:
        app: loadbalancer-pod
    spec:
      containers:
        - name: loadbalancer-pod
          image: <dockerhub username>/image name
          resources:
            limits:
              memory: "256Mi"
              cpu: "500m"
          ports:
            - containerport: 8080
```

To access the Jenkins webpage using Jenkins server "IP address:8080"

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

Open the jenkins initial admin password path on ubuntu linux to get the credentials.


```
root@ip-172-31-11-179:~# cat /var/lib/jenkins/secrets/initialAdminPassword  
5f5db8ca74b94f838ccbe
```

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

Click on “Install 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.

Create the username, password, your name and email address to login jenkins

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

Jenkins 2.387.2

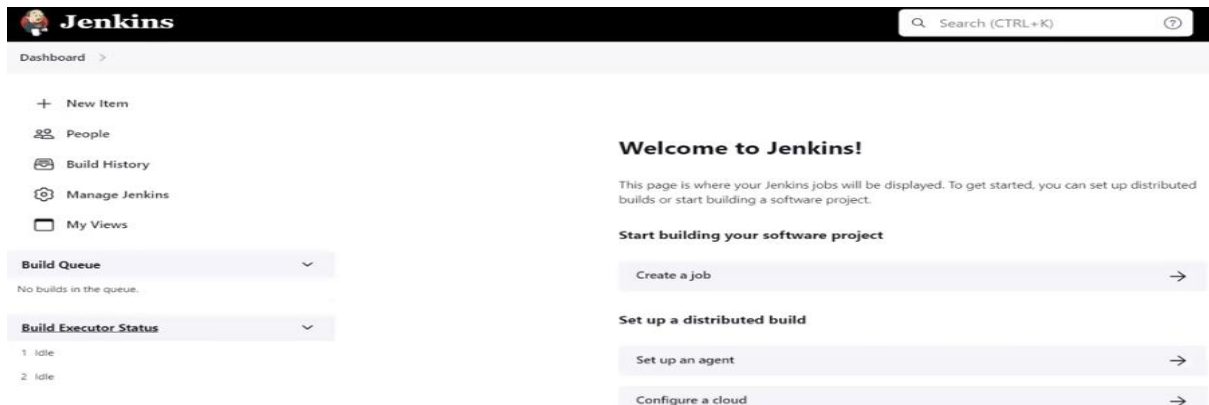
[Skip and continue as admin](#)

[Save and Continue](#)

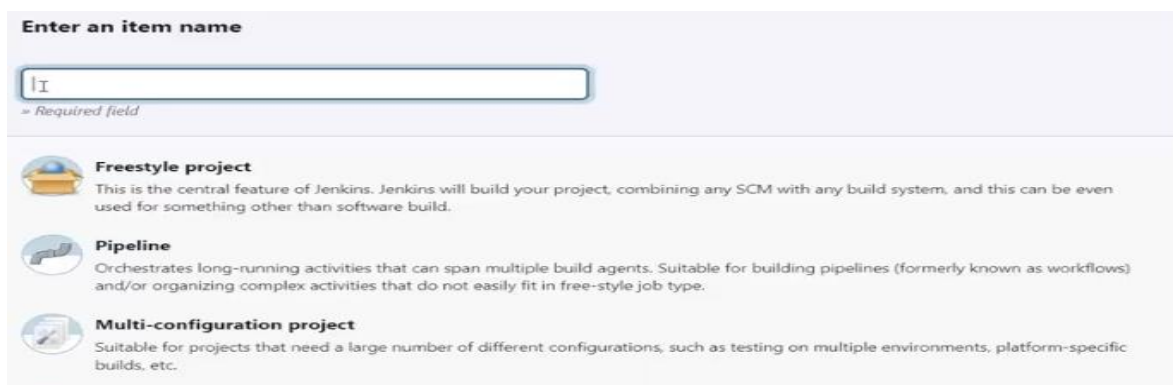
Getting Started



Click on “New Item”



Click on “Pipeline” project



Go to Ubuntu linux to check whether the git is installed or not by using the below command

```
root@ip-172-31-11-179:~# git --version
git version 2.25.1
```

Git has already been installed on the latest Ubuntu linux OS.

To create the directory folder then initialize empty git repository

```
root@ip-172-31-11-179:~# mkdir dev
root@ip-172-31-11-179:~# cd dev
root@ip-172-31-11-179:~/dev# git init
Initialized empty Git repository in /root/dev/.git/
```

To create all five files in Linux, then copy all the code from Visual Studio Code, and then paste & save the code inside the file.

```

root@ip-172-31-11-179:~/dev# vi requirements.txt
root@ip-172-31-11-179:~/dev# vi main.py
root@ip-172-31-11-179:~/dev# vi form.html
root@ip-172-31-11-179:~/dev# vi Dockerfile
root@ip-172-31-11-179:~/dev# vi pod.yaml

```

Now we will move all the files from Ubuntu linux to GitHub remote repository by using the below commands

To move all the files from working directory to staging area by using “**git add .**” command. Dot refers to select all the files then to move all the files from staging area to local repository by using “**git commit**” command

```

root@ip-172-31-11-179:~/dev# git add .
root@ip-172-31-11-179:~/dev# git commit -m "First Commit"

```

```

root@ip-172-31-11-179:~/dev# git remote add origin https://github.com/
<Github repository name>

```

To move the files from the local repository to remote repository by using “**git push origin master**” command

```

root@ip-172-31-11-179:~/dev# git push origin master

```

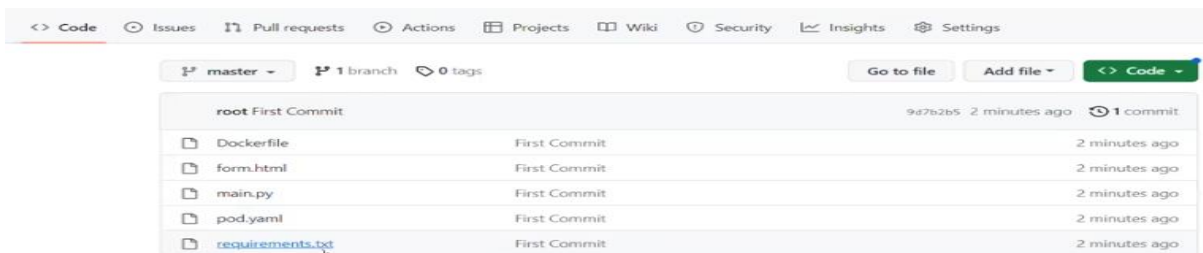
Enter the github username and password as access token

```

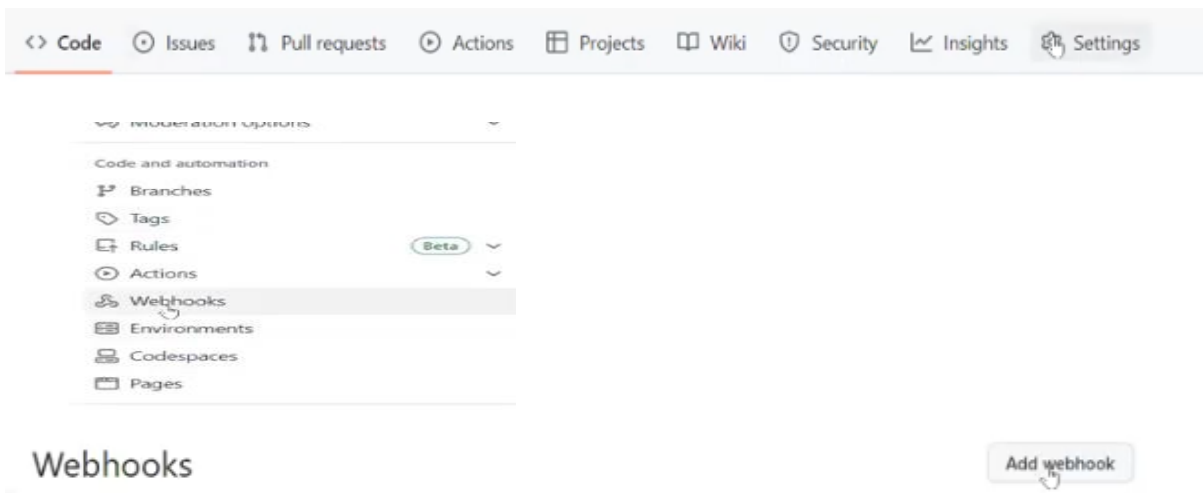
Username for
Password for

```


Now all the files has been successfully pushed to Github remote repository.



We will setup GitHub webhook. Go to GitHub settings -> Click on “Webhooks”



Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).



Confirm access

Password

[Forgot password?](#)

Confirm

Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.

Give the Jenkins server URL in payload URL with adding **/github-webhook/**

Payload URL *

Content type

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me everything.

☐ Let me select individual events.

☒ **Active**
We will deliver event details when this hook is triggered.

Add webhook

Go to Jenkins website and Click on “General” and tick the check box of GitHub hook trigger for GITscm polling

Configure

General

Advanced Project Options

Pipeline

☐ Pipeline speed/durability override ?

☐ Preserve stashes from completed builds ?

☐ This project is parameterized ?

☐ Throttle builds ?

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ **GitHub hook trigger for GITscm polling ?**

☐ Poll SCM ?

☐ Quiet period ?

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

Give the github your project URL then click on “Save”

Configure ☐ [Plain text] [Preview](#)

☐ Discard old builds ?

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the controller restarts

☒ **GitHub project**

Project url ?

GitHub webhook has been successfully created

To create one virtual server for SonarQube on the AWS.

☐ SonarQube [i-0eebd6ddbe4fe1262](#) Running t2.medium 2/2 checks passed

Allow the sonarQube port 9000 inbound security groups on the AWS virtual server.

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	
sgr-016ca6f0ac5e9f015	SSH	TCP	22	Custom		Delete
-	Custom TCP	TCP	9000	Anywh...	SonarQube	Delete

We will install the sonarqube on Ubuntu linux by using wget command

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip
```

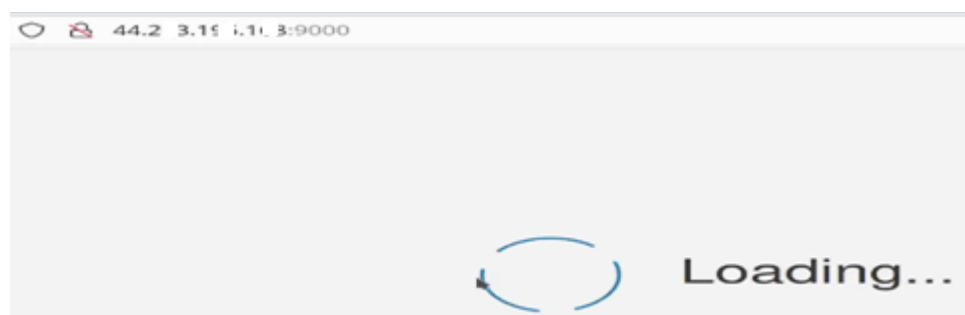
We will unzip the sonarqube by using the below command

```
sudo apt install unzip
unzip sonarqube-9.9.0.65466.zip
```

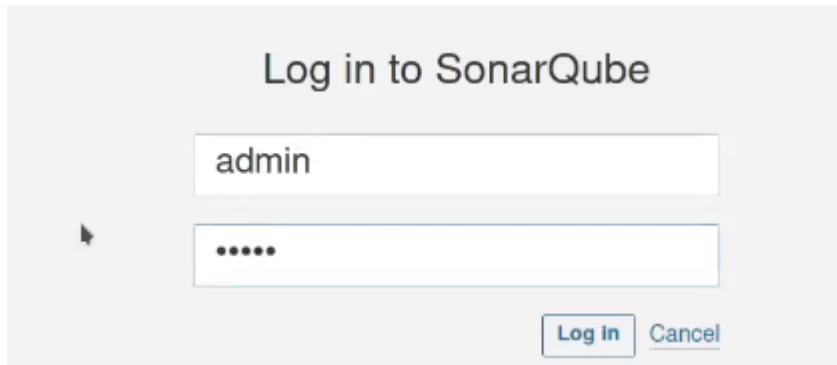
After Unzipped sonarqube, we are going to execute it

```
./sonar.sh
```

We will login to SonarQube by using "Virtual server IP address : sonarqube running port"



Now the sonarqube is working fine. Default username and password of sonarqube is admin.



Log in to SonarQube

admin

.....

Log in Cancel

Change your password



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 *

.....

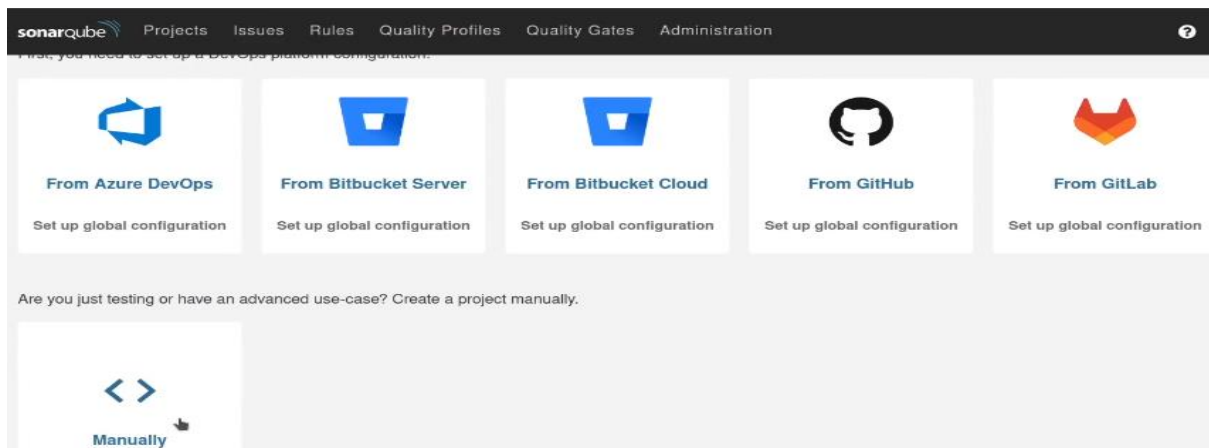
Confirm Password *

.....

Update

✓ The password has been changed!

Click on “Manually” to create a project manually.



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

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

Are you just testing or have an advanced use-case? Create a project manually.

Manually



Create a project

All fields marked with * are required

Project display name *

.....

Up to 255 characters. Some scanners might override the v

Project key *

.....

The project key is a unique identifier for your project. It max 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one no

Main branch name *

main


The name of your project's default branch [Learn More](#)


Set Up


Overview Issues Security Hotspots Measures Code Activity

How do you want to analyze your repository?

Do you want to integrate with your favorite CI? Choose one of the following tutorials.


With Jenkins


With GitHub Actions


With Bitbucket Pipelines

Select your DevOps platform

Bitbucket Cloud Bitbucket Server **GitHub** GitLab

Configure Analysis >

1 Create a Pipeline Job

Create a Pipeline in order to automatically analyze your project.

1. From Jenkins' dashboard, click **New Item** and create a **Pipeline Job**.
2. Under **Build Triggers**, choose **Trigger builds remotely**. You must set a unique, secret token for this field.
3. Under **Pipeline**, make sure the parameters are set as follows:
 - **Definition**: Pipeline script from SCM
 - **SCM**: Configure your SCM. Make sure to only build your main branch. For example, if your main branch is called "main", put `*/main` under **Branches to build**.
 - **Script Path**: Jenkinsfile
4. Click **Save**.

Continue >

2 Create a GitHub Webhook

Create a Webhook in your repository to trigger the Jenkins job on push. Already have a Webhook configured? [Skip this step](#).

1. Go to the **GitHub Webhook creation page for your repository** and enter the following information:
 - **URL**: Enter the following URL, replacing the values between `***` as needed:

```
***JENKINS_SERVER_URL*** / job / ***JENKINS_JOB_NAME*** / build? token=***JENKINS_BUILD_TRIGG
```

Copy

2. Under **Which events would you like to trigger this webhook?** select **Let me select individual events** and check the following:
 - **Pushes**
3. Click **Add webhook**.

Continue >

3 Create a Jenkinsfile

1. What option best describes your build?

Maven Gradle .NET **Other (for JS, TS, Go, Python, PHP, ...)**

Copy the sonarqube project key. Once copied then click on "finish this tutorial"

2. Create a `sonar-project.properties` file in your repository and paste the following code:

```
sonar.projectKey=
```

Copy

Finish this tutorial >

Go to your sonarqube account and click on “security”

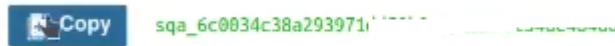


Now we have to create a token

Generate Tokens

Name	Type	Expires in	
<input type="text" value="Sonarqube-Token"/>	<input type="text" value="Global Analysis Token"/>	<input type="text" value="30 days"/>	<input type="button" value="Generate"/>

Copy this token

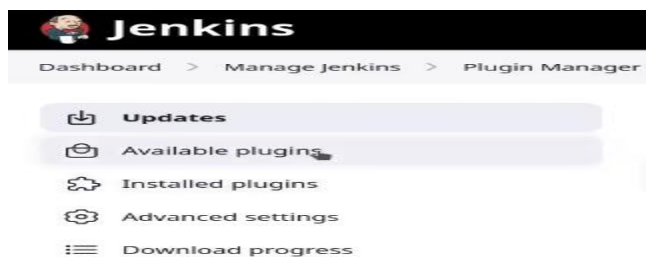


Now we go back to Jenkins to install plugins

Click on “Manage Jenkins” & “Manage plugins”



Click on “Available Plugins”



Search for SonarQube Scanner and SSH2

Plugins



Plugins

Install	Name	Released
<input checked="" type="checkbox"/>	SSH2 Easy 1.4 This plugin allows you to ssh2 remote server to execute linux commands , shell , sftp upload, download etc	6 yr 9 mo ago

Install without restart

Download now and install after restart

Update information obtained: 19 min ago

Check now

SonarQube Scanner

✓ Success

SSH2 Easy

✓ Success

Click on “manage Jenkins” & “Global Tool Configuration”

Dashboard > Manage Jenkins > Plugin Manager

System Configuration



Configure System

Configure global settings and paths.



Global Tool Configuration

Configure tools, their locations and automatic installers.

Click on “Add SonarQube Scanner”

SonarQube Scanner

SonarQube Scanner installations

List of SonarQube Scanner installations on this system

Add SonarQube Scanner

Give any name that you want

≡ SonarQube Scanner

Name

Required

☒ Install automatically ?

≡ Install from Maven Central

Version

Add Installer

Add SonarQube Scanner

Save

Apply

After saving it. We have to go to configure system

System Configuration



Configure System

Configure global settings and paths.

Scroll down to SonarQube servers & Click on “Add SonarQube”

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

Add SonarQube

Give any name that you want

Name ✕

Sonar-Server

Give URL of the SonarQube

Server URL

Default is http://localhost:9000

http://44.21 58:9000

Click on “Add Jenkins”

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

Add

Jenkins

Advanced

Select “secret text” and paste the token secret code (refer page no 12)

Kind

Secret text

Scope ?

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

Secret

.....

ID ?

Sonar-Token

Description ?

Add Cancel

Select -> “Sonar Token” & “Save it”

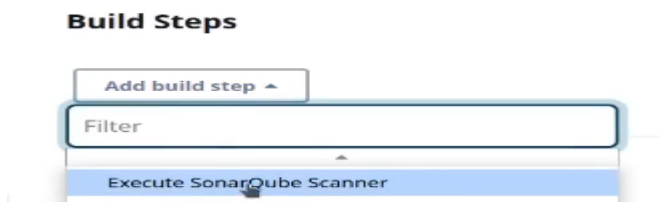
Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Sonar-Token

Save Apply

Go to Build Steps and select “Execute SonarQube Scanner”



Paste the Sonar project key (Refer Page no 11) and save it.

Analysis properties ?

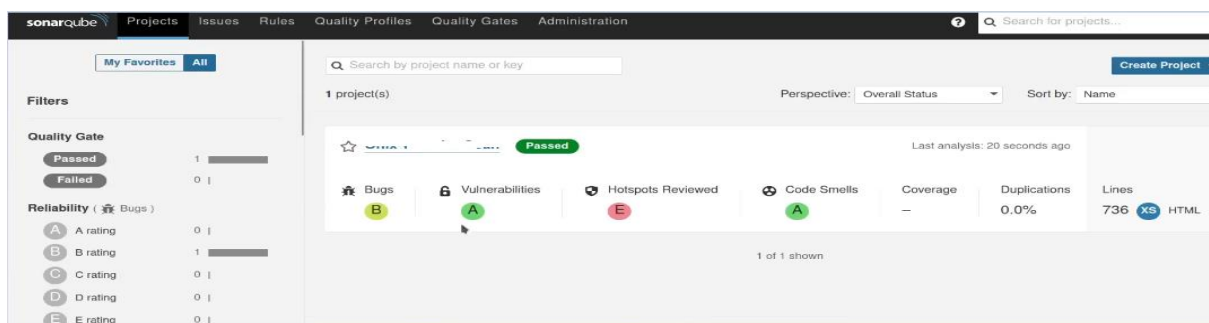
sonar.projectKey=

Save Apply

Click on “Build Now”



Now the Sonarqube scanning is working fine



Now we have to write the four stages of declarative groovy pipeline scripts on the Jenkins.

Dashboard > Kubeproject > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Definition

Pipeline script

Script ?

```

1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8       }
9     }
10  }
11 }
12

```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save Apply

Stage 1: Pull the code from GitHub

```

pipeline {
  agent any
  stages {
    stage('pull code from GitHub') {
      steps {
        git <git hub remote repository URL>
      }
    }
  }
}

```

Click on “Build Now”

Dashboard > Kubeproject >

- Status
- Changes
- Build Now**
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Pipeline Kubeproject

Stage View

Average stage times:
(Average full run time: ~5s)

Pull Code From GitHub

4s

Stage View

Average stage times:
(Average full run time: ~5s)

Pull Code From GitHub

Success

logs

Now the code has been successfully pulled from github

```

root@ip-172-31-11-179:~# cd /var/lib/jenkins
root@ip-172-31-11-179:/var/lib/jenkins# ls
config.xml                               jenkins.telemetry.Correlator.xml      secret.key
hudson.model.UpdateCenter.xml           jobs                                  secret.key.not-so-secret
hudson.plugins.git.GitTool.xml          nodeMonitors.xml                     secrets
identity.key.enc                        nodes                                updates
jenkins.install.InstallUtil.lastExecVersion  org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml  userContent
jenkins.install.UpgradeWizard.state      plugins                               users
jenkins.model.JenkinsLocationConfiguration.xml  queue.xml                           workspace
root@ip-172-31-11-179:/var/lib/jenkins# cd workspace/
root@ip-172-31-11-179:/var/lib/jenkins/workspace# ls
Kubeproject
root@ip-172-31-11-179:/var/lib/jenkins/workspace# cd Kubeproject/
root@ip-172-31-11-179:/var/lib/jenkins/workspace/Kubeproject# ls
Dockerfile  form.html  main.py  pod.yaml  requirements.txt

```

We will install the Docker on Ubuntu linux

```

root@ip-172-31-11-179:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 26 not upgraded.
Need to get 66.1 MB of archives.
After this operation, 293 MB of additional disk space will be used.

```

After the Docker installation, Login to Docker on linux.

```

root@ip-172-31-11-179:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.

```

Username:

Password:

Login Succeeded

We will build the docker image from the docker file

Stage 2: Build the Docker image and run on Jenkins declarative pipeline script

```

stage('Build the Docker image')
{
  steps {
    sh 'sudo docker build -t <dockerhub username>/image name: latest /var/lib/jenkins/workspace/kubeprojects'
    sh 'sudo docker tag <dockerhub username>/image name: latest <dockerhub username>/image name:${BUILD_NUMBER}'
  }
}

```

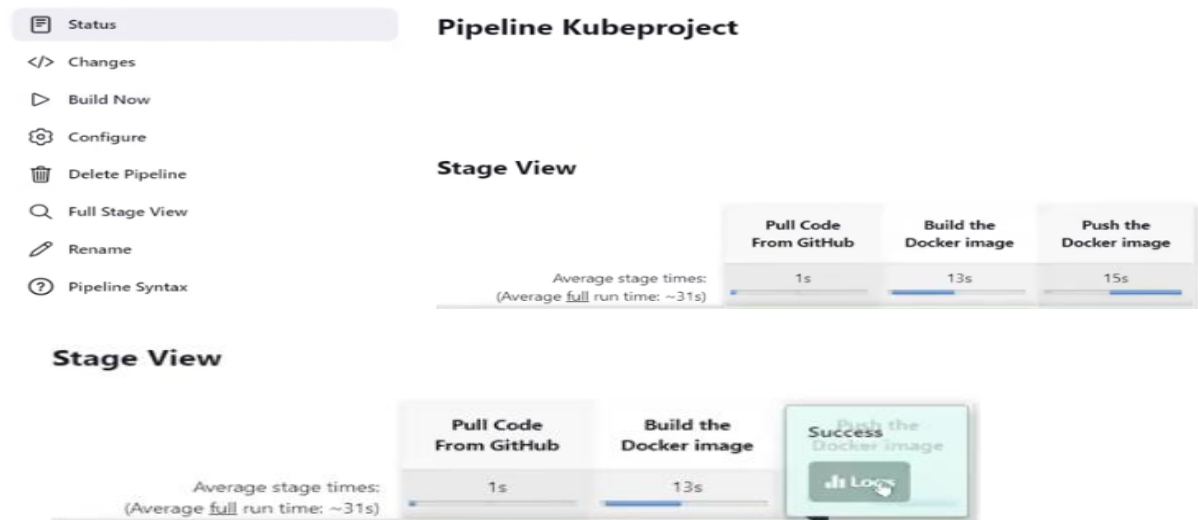
The screenshot shows the Jenkins Pipeline Stage View for a pipeline named 'Kubeproject'. On the left, there is a sidebar with actions: 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'Rename', and 'Pipeline Syntax'. The main area displays two stages: 'Pull Code From GitHub' and 'Build the Docker image'. Below the stage names, a progress bar shows the average stage times: 'Pull Code From GitHub' at 1s and 'Build the Docker image' at 17s. At the bottom, it indicates the average full run time is approximately 32 seconds.

Stage 3: Push the Docker image and run on Jenkins declarative pipeline script

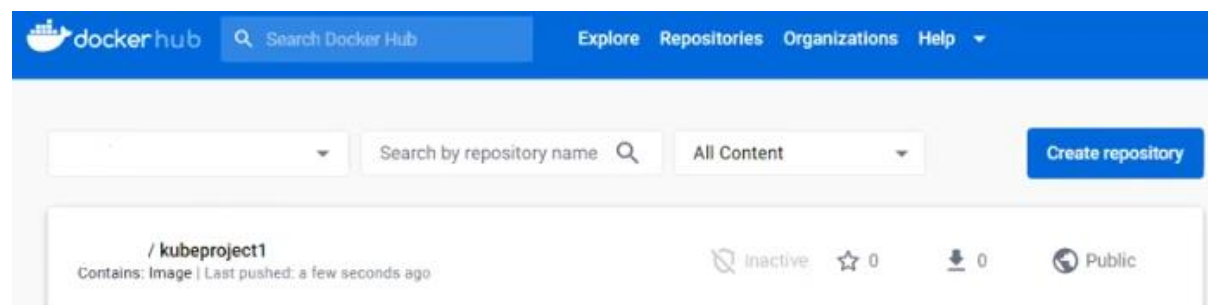
```

stage('push the Docker image'){
    steps {
        sh 'sudo docker image push <dockerhub username>/image name: latest
        sh 'sudo docker image push <dockerhub username>/image name:${BUILD_NUMBER}'
    }
}
}
}
}

```



Now the Docker image has been successfully pushed to docker hub



KOPS (Kubernetes Operations) install on Ubuntu linux

PREREQUIREMENTS

1. Kops binary (kubernetes cluster initiate)
2. Kubectl binary (kubernetes deployments)

KOPS BINARY SETUP

```

# curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl
-s https://api.github.com/repos/kubernetes/kops/releases/latest | grep
tag_name | cut -d '"' -f 4)/kops-linux-amd64
# chmod +x ./kops
# sudo mv ./kops /usr/local/bin/

```

KUBECTL BINARY SETUP

```

# curl -Lo kubectl https://storage.googleapis.com/kubernetes-
release/release/$(curl -s https://storage.googleapis.com/kubernetes-
release/release/stable.txt)/bin/linux/amd64/kubectl
# chmod +x ./kubectl
# sudo mv ./kubectl /usr/local/bin/kubectl

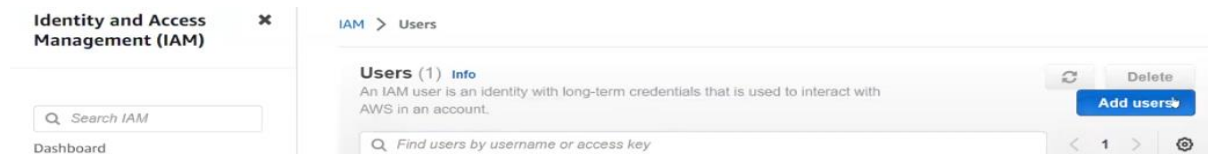
```


Kops and Kubectl binary setup has been successfully installed on Ubuntu linux by using the above commands

3. Setup IAM User (Kops access aws resource) -> kindly configure AWS Command line interface packages in your linux machines.

```
root@ip-172-31-11-179:~# apt install awscli
```

In order to build clusters within AWS, we will create a dedicated IAM user for kops. This user requires API credentials in order to use kops. Create the user and credentials using the AWS console.



Set username

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Info If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel **Next**

Permissions options

☐ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

	Policy name ↗	Type	Attached entities
<input type="checkbox"/>	AccessAnalyzerServ...	AWS managed	0
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	1
<input checked="" type="checkbox"/>	AdministratorAcces...	AWS managed	1

Permissions boundary - optional
Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. [Learn more](#)

Cancel Previous **Next**

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy
AdministratorAccess-Amplify	AWS managed	Permissions policy

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#)
[Previous](#)
[Create user](#)

After username creation. Click on user name

User created successfully
[View user](#)

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

IAM > Users

Users (2) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

<input type="checkbox"/>	User name	Groups	Last activity	MFA	Password
<input type="checkbox"/>	kube			None	

Click on create access key

Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

[Create access key](#)

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

[Create access key](#)

Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Command Line Interface (CLI)

You plan to use this access key to enable the AWS CLI to access your AWS account.

[Cancel](#)
[Next](#)

[Cancel](#)
[Previous](#)
[Create access key](#)

Access key	Secret access key
------------	-------------------

Copy the access key and secret access key and paste on Ubuntu linux

```
root@ip-172-31-11-179:~# aws configure
```

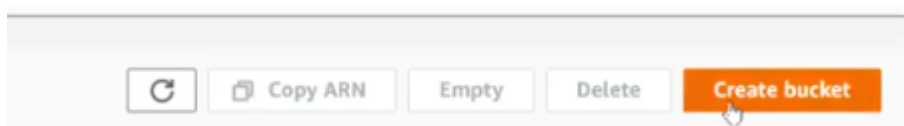
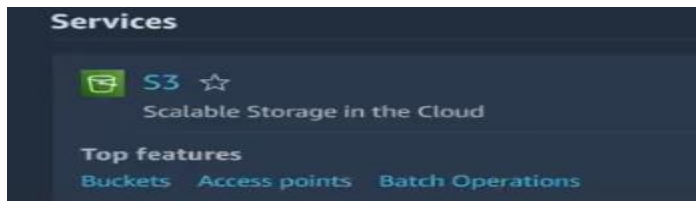
```
AWS Access Key ID [None]:
```

```
AWS Secret Access Key [None]:
```

```
Default region name [None]: us-east-2
```

```
Default output format [None]: json
```

To create one S3 bucket on AWS



General configuration

Bucket name

Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming [🔗](#)

AWS Region

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

<input type="radio"/> ACLs disabled (recommended) All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.	<input checked="" type="radio"/> ACLs enabled Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.
---	---

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☐ Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ Block public access to buckets and objects granted through any access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

☐ Block public access to buckets and objects granted through new public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

- ☐ Disable
☒ Enable

- ☒ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Default encryption [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption key type [Info](#)

- ☒ Amazon S3 managed keys (SSE-S3)
☐ AWS Key Management Service key (SSE-KMS)

Bucket Key

When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS.

[Learn more](#)

- ☒ Disable
☐ Enable

► Advanced settings

After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel

Create bucket

S3 bucket has been successfully created

```
root@ip-172-31-11-179:~# export KOPS_STATE_STORE=s3://<S3 Bucker name>
```

```
root@ip-172-31-11-179:~# kops create cluster --zones us-east-2a ${NAME}
```

```
root@ip-172-31-11-179:~# kops update cluster --name
```

```
root@ip-172-31-11-179:~# kops validate cluster
```

Validating cluster

Now the cluster is ready.

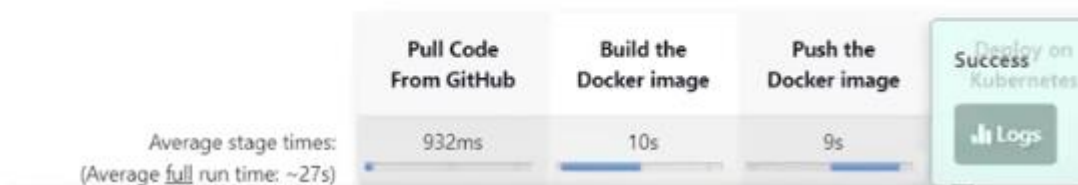
Stage 4: Deploy on kubernetes

```
    stage('Deploy on kubernetes') {
      steps {
        sh 'kubectl apply -f /var/lib/jenkins/workspace/pod.yaml'
        sh 'kubectl rollout restart deployment loadbalancer-pod'
      }
    }
  }
}
```

Stage View



Stage View



Four stages of Jenkins Declarative pipeline script:



```
pipeline {
  agent any
  stages {
    stage('pull code from GitHub') {
      steps {
        git <git hub remote repository URL>
      }
    }
    stage('Build the Docker image') {
      steps {
        sh 'sudo docker build -t <dockerhub username>/image name: latest /var/lib/jenkins/workspace/kubeprojects'
        sh 'sudo docker tag <dockerhub username>/image name: latest <dockerhub username>/image name:${BUILD_NUMBER}'
      }
    }
    stage('push the Docker image') {
      steps {
        sh 'sudo docker image push <dockerhub username>/image name: latest'
        sh 'sudo docker image push <dockerhub username>/image name:${BUILD_NUMBER}'
      }
    }
    stage('Deploy on kubernetes') {
      steps {
        sh 'kubectl apply -f /var/lib/jenkins/workspace/pod.yaml'
        sh 'kubectl rollout restart deployment loadbalancer-pod'
      }
    }
  }
}
```



```

root@ip-172-31-11-179:~# kubectl get svc
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes           ClusterIP           100.64.0.1      <none>            443/TCP          10m
loadbalancer-svc     LoadBalancer       100.64.208.9    <pending>         80:30421/TCP     63s
root@ip-172-31-11-179:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
loadbalancer-pod-c68658d-6kx4b     1/1     Running   0           4m32s
loadbalancer-pod-c68658d-7b46v     1/1     Running   0           4m51s
loadbalancer-pod-c68658d-wg2hx     1/1     Running   0           4m30s

```

Load balancers (4)							⌂	Actions ▾	Create load balancer
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.									
<input type="text" value="Filter by property or value"/>							< 1 >		
<input type="checkbox"/>	Name	 DNS name	State	VPC ID	Availability Zones	Type			
<input type="checkbox"/>	a5c35cdd18e504733bde0c847314cf50	 a5c35cdd18e504733bde0...	-	vpc-0a703ebcde6917ad8	us-east-2a (use2-az1)	classic			

Now webpage has been successfully triggered



.....