



Expert Cloud Consulting

Enhance Optimise & Scale

ASCP GPUonCLOUD Pvt Ltd

“Expert Cloud Consulting” -

CI-CD Fundamentals [Title,18, Arial]

15.Jan.2025 [Subtitle,14, Arial]

version 1.0

—

Contributed by Shraddha Chaudhari [Normal text,14, Arial]

Approved by (In Review)

Expert Cloud Consulting

Office #811, Gera Imperium Rise,

Hinjewadi Phase-II Rd, Pune, India – 411057

“Expert Cloud Consulting”

CI-CD Fundamentals [Title,18, Arial]

1.0 Contents [Heading3,14, Arial]

1.0 Contents [Heading3,14, Arial]	1
2.0 General Information: [Heading3,14, Arial]	3
2.1 Document Jira/ Github Ticket(s) [Heading4,12, Arial].....	3
2.2 Document Purpose.....	3
2.3 Document Revisions.....	4
2.4 Document References	4
3.0 Document Overview:.....	5
4.0 Steps / Procedure	6
4.1 : Setup the ubuntu server for Jenkins application.....	6
4.2: Choose an Amazon Machine Image (AMI)	6
4.3: Key-Pair Configuration.....	7
4.4: Network settings	7
4.4.1: VPC Configuration:	7
4.4.2: Security Group Configuration	8
4.5: Launch Instance	8
4.5: SSH Configuration.....	9
4.6: Setting up the CI/CD pipeline with Jenkins	10



4.7: Access Jenkins.....	12
4.8: Install required Jenkins plugins:	14
4.9: Configured Jenkins Credentials :	16
5.0: Install Docker on Jenkins server:	16
5.1: Project Structure:.....	16
6.0: Initialize a git repository:	20
6.1: Push to GitHub:	21
7.0: Setup Jenkins pipeline job:	22
8.0: Test the pipeline:	24
9.0: Verify Deployment:	24
10: Integrate Notifications:	26
Step 1: Install Email Extension Plugin	26
Step 2: Configure Jenkins Email Notification Settings	27
11. Changes Made:	32
12. Steps to Test:	32



2.0 General Information: [Heading3,14, Arial]

2.1 Document Jira/ Github Ticket(s) [Heading4,12, Arial]

Ticket(s) Name	Url
CI-CD Fundamentals [Normal text,10, Arial]	https://github.com/shaanicha/CI-CD-pipeline/tree/main

2.2 Document Purpose

The purpose of this documentation is to provide a clear guide for setting up a **CI/CD pipeline using Jenkins**. It covers the steps to automate the process of:

1. **Continuous Integration (CI)**: Automatically pulling code from GitHub, running unit tests, and building Docker images.
2. **Continuous Deployment (CD)**: Automatically pushing the Docker image to Docker Hub and deploying it to a staging environment.
3. **Email Notifications**: Sending email alerts for both successful and failed pipeline runs.
4. **Troubleshooting**: Addressing common issues such as permission errors or failed builds.

Overall, this documentation helps automate the entire development workflow, ensuring faster, reliable, and consistent deployments.

4o mini

[Normal text,10, Arial, Justify Alignment]

2.3 Document Revisions

Date	Version	Contributor(s)	Approver(s)	Section(s)	Change(s)
15/Jan/2025	1.0	Shraddha Chaudhari	Akshay Shinde	All Sections	New Document Created

2.4 Document References

The following artifacts are referenced within this document. Please refer to the original documents for additional information.

Date	Document	Filename / Url
2017	Jenkins Documentation - Pipelines	https://www.jenkins.io/doc/book/pipeline/
2025	Jenkins User Handbook	https://www.jenkins.io/user-handbook.pdf
2025	Docker Documentation	https://docs.docker.com/
2025	GitHub Documentation	https://docs.github.com/en



3.0 Document Overview:

The goal is to streamline development workflows, enabling faster, reliable, and automated testing, building, and deployment processes. This guide includes setup instructions, best practices, and references to relevant tools and documentation.

1. Build a CI/CD pipeline using Jenkins to:

- Pull code from GitHub.
- Run unit tests automatically.
- Build and deploy a Docker container to a staging environment.



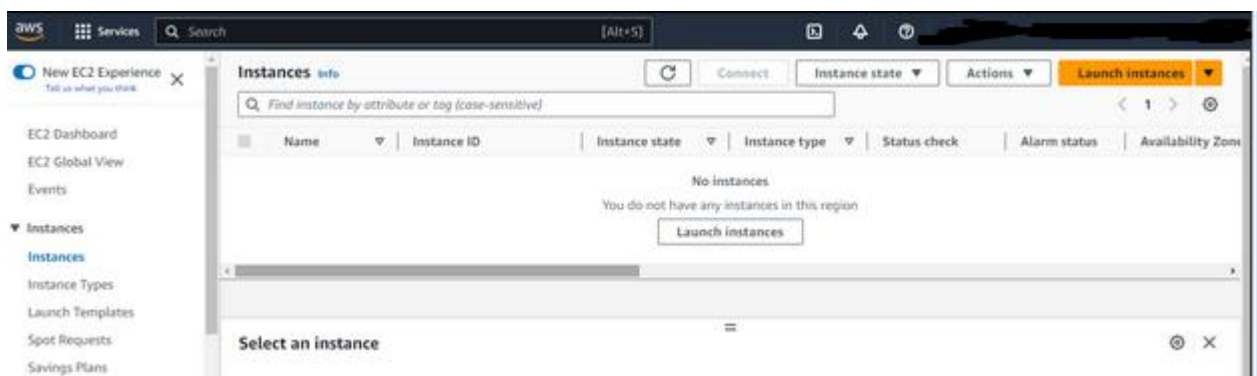
4.0 Steps / Procedure

4.1 : Setup the ubuntu server for Jenkins application

The following procedure has been done during the installation of ubuntu server:

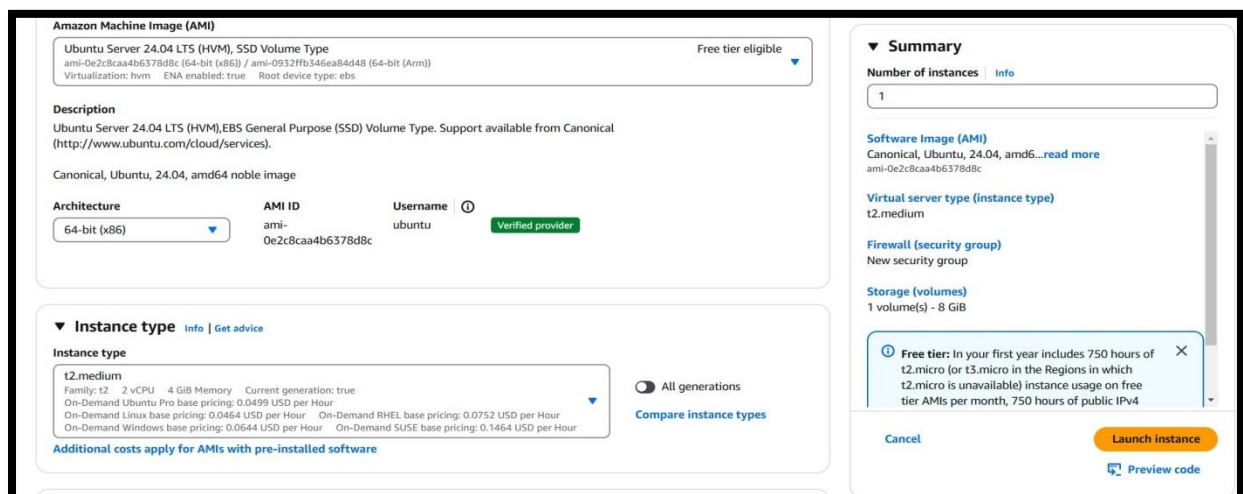
Log in to the AWS Management Console (Company-SandBox) and navigate to the EC2 dashboard.

Click on the "Launch Instance" button to start the process of launching a new EC2 instance.



4.2: Choose an Amazon Machine Image (AMI)

Select an instance type, configure your instance details (such as the number of instances, and storage)



Configure storage [Info](#) Advanced

1x 10 GiB gp3 Root volume 3000 IOPS (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

[Add new volume](#)

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems [Edit](#)

Advanced details [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd64...[read more](#)
ami-0e2c8caa4b6378d8c

Virtual server type (instance type)
t2.medium

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 10 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

[Cancel](#) [Launch instance](#) [Preview code](#)

4.3: Key-Pair Configuration

Select an instance type and create a new key-pair as name is vmkey.

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required
vmkey [Create new key pair](#)

Network settings [Info](#) [Edit](#)

Network [Info](#)
vpc-09e4fbb4d5401d76

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable
Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd64...[read more](#)
ami-0e2c8caa4b6378d8c

Virtual server type (instance type)
t2.medium

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 10 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

[Cancel](#) [Launch instance](#)

4.4: Network settings

4.4.1: VPC Configuration:

Select the vpc and subnet for the ec2 instance.

4.4.2: Security Group Configuration

Create a new security group with required security rules

Specified Security group rule for this ec2 instances are shown below:

Allow port 8080 for Jenkins server

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Action
sg-0b1217480e5316599	HTTPS	TCP	443	Custom	0.0.0.0/0	Delete
sg-0102712d974298a98	SSH	TCP	22	Custom	0.0.0.0/0	Delete
sg-0d8c61604a18a1d7e	HTTP	TCP	80	Custom	0.0.0.0/0	Delete
-	Custom TCP	TCP	8080	Anyw...	0.0.0.0/0	Delete

4.5: Launch Instance

Click on Launch instance

Configure storage [Info](#) [Advanced](#)

1x 10 GiB gp3 Root volume: 3000 IOPS (Not encrypted)

Free tier eligible customers can get up to 30 GiB of EBS General Purpose (SSD) or Magnetic storage

[Add new volume](#)

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems [Edit](#)

Advanced details [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd64...[read more](#)
ami-0e2c8caa4b6378d8c

Virtual server type (instance type)
t2.medium

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 10 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

[Cancel](#) [Launch instance](#) [Preview code](#)

Instances are created and they are ready to use.

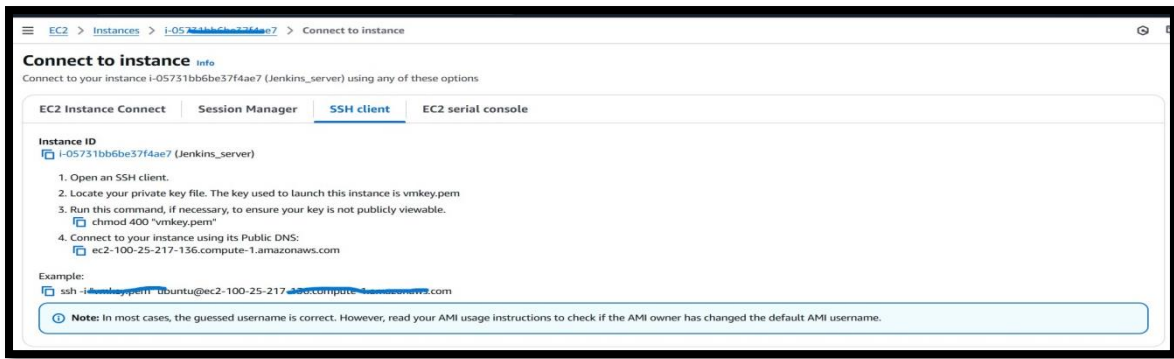
Instances (1) [Info](#) [Last updated less than a minute ago](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) [All states](#)

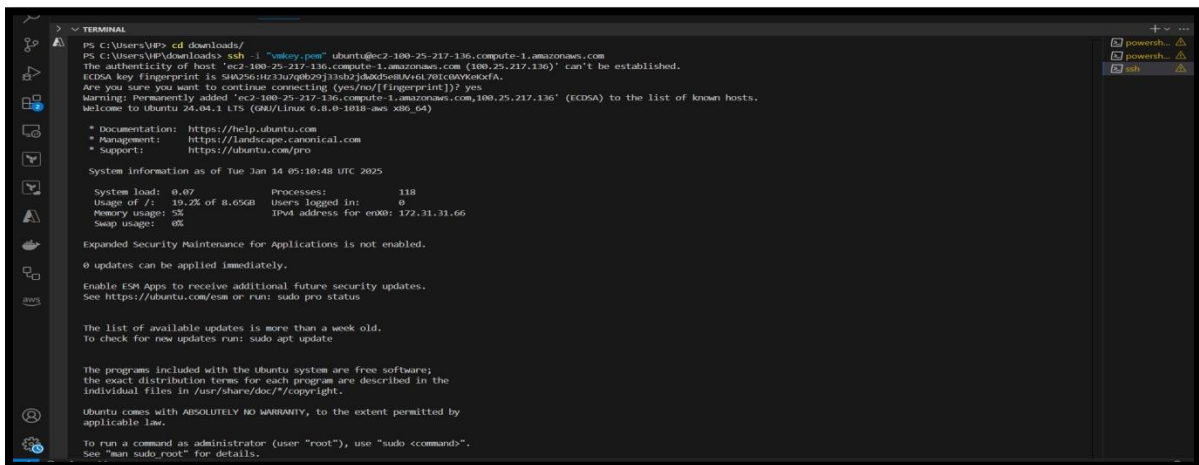
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Jenkins_server	i-05731bb6be37f4ae7	Running	t2.medium	Initializing	View alarms	us-east-1a	ec2-100-

4.5: SSH Configuration

Log in ec2 instance using SSH client.



Successfully able to connect the ec2-Instance by using ssh client.



4.6: Setting up the CI/CD pipeline with Jenkins

Here's a step-by-step guide to setting up the CI/CD pipeline with Jenkins:

So, first need to update package and install other dependencies which need for Jenkins.

```
sudo apt update
sudo apt install openjdk-11-jdk -y
wget -q -O https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key
add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins -y
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

```
root@ip-172-31-20-30:/home/ubuntu# history
1 apt update
2 apt upgrade -y
3 sudo apt install openjdk-17-jdk -y
4 java -version
5 wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
6 sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
7 apt update
8 sudo apt-key del 58A31D57EF5975CA
9 curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
10 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
11 apt update
12 sudo apt install jenkins -y
13 clear
14 systemctl status jenkins
```

This above mentioned all cmds need to setting up Jenkins.

```

root@ip-172-31-20-30:/home/ubuntu# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-01-14 06:21:16 UTC; 24s ago
     Main PID: 15684 (java)
        Tasks: 52 (limit: 4676)
      Memory: 686.6M (peak: 695.1M)
         CPU: 15.803s
        CGroup: /system.slice/jenkins.service
                └─15684 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jan 14 06:21:12 ip-172-31-20-30 jenkins[15684]: 8915e8be11c749e2974296a34560f9ae
Jan 14 06:21:12 ip-172-31-20-30 jenkins[15684]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jan 14 06:21:12 ip-172-31-20-30 jenkins[15684]: *****
Jan 14 06:21:12 ip-172-31-20-30 jenkins[15684]: *****
Jan 14 06:21:12 ip-172-31-20-30 jenkins[15684]: *****
Jan 14 06:21:16 ip-172-31-20-30 jenkins[15684]: 2025-01-14 06:21:16.318+0000 [id=32] INFO jenkins.InitReactorRunner$1onAttained: Completed initialization
Jan 14 06:21:16 ip-172-31-20-30 jenkins[15684]: 2025-01-14 06:21:16.332+0000 [id=23] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Jan 14 06:21:16 ip-172-31-20-30 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Jan 14 06:21:16 ip-172-31-20-30 jenkins[15684]: 2025-01-14 06:21:16.468+0000 [id=48] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson
Jan 14 06:21:16 ip-172-31-20-30 jenkins[15684]: 2025-01-14 06:21:16.472+0000 [id=48] INFO hudson.util.Retrier#start: Performed the action check updates server successf

```

4.7: Access Jenkins

Once Jenkins installation done next take access of jenkins.

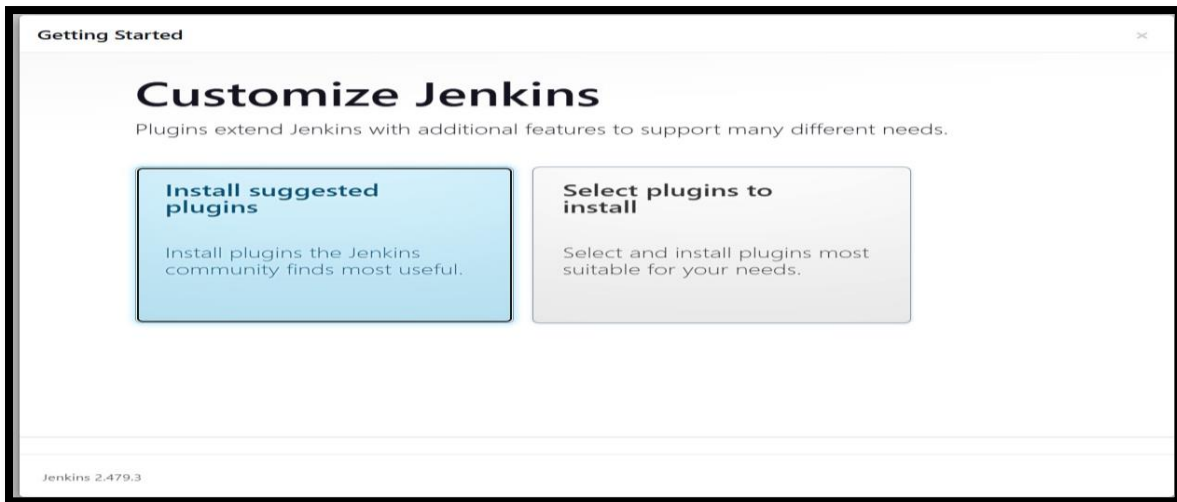
- Open http://<your_server_ip>:8080 in your browser.

Retrieve the initial admin password:

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



After this it showing this page click on install suggested plugins.

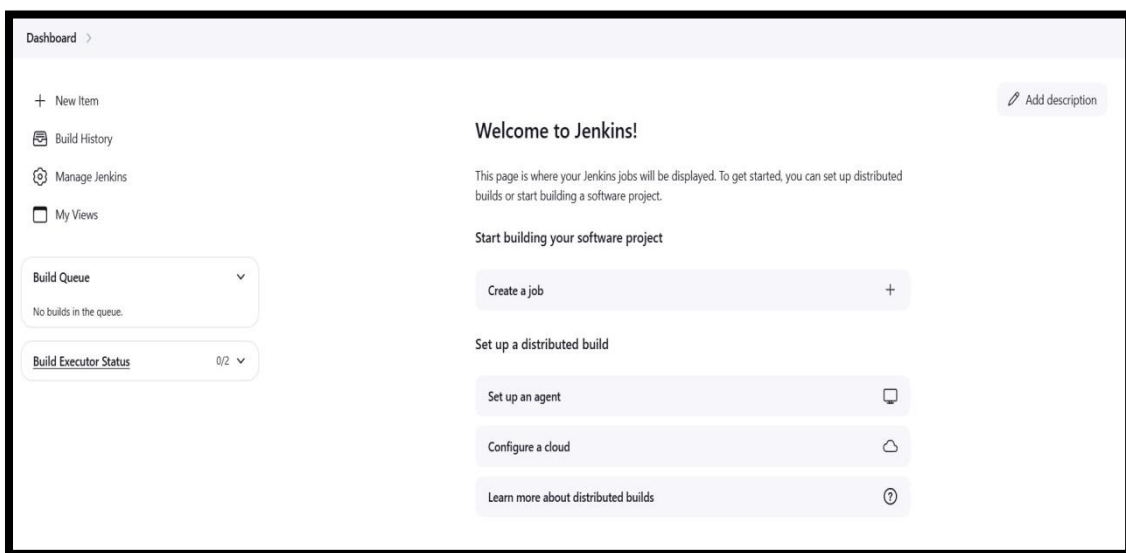
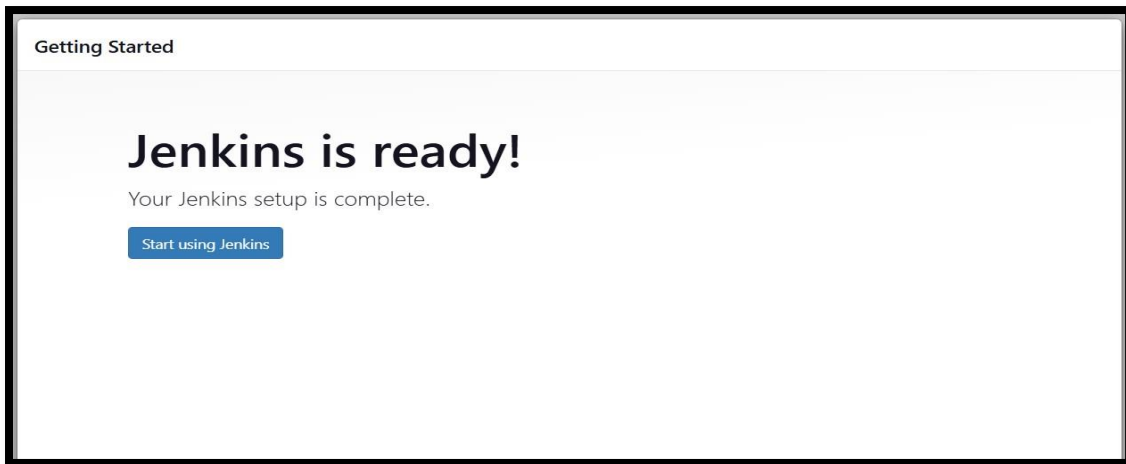


Next, we have to create admin user for login to Jenkins.

The screenshot shows the 'Getting Started' window for Jenkins, specifically the 'Create First Admin User' screen. The form contains the following fields: 'Username' (filled with 'Shraddha'), 'Password' (filled with dots), 'Confirm password' (filled with dots), 'Full name' (filled with 'shraddha chaudhari'), and 'E-mail address' (empty). At the bottom, there is a 'Skip and continue as admin' link and a 'Save and Continue' button. The Jenkins version 'Jenkins 2.479.3' is displayed at the bottom left.

Once filled all needed information then click to save and continue.

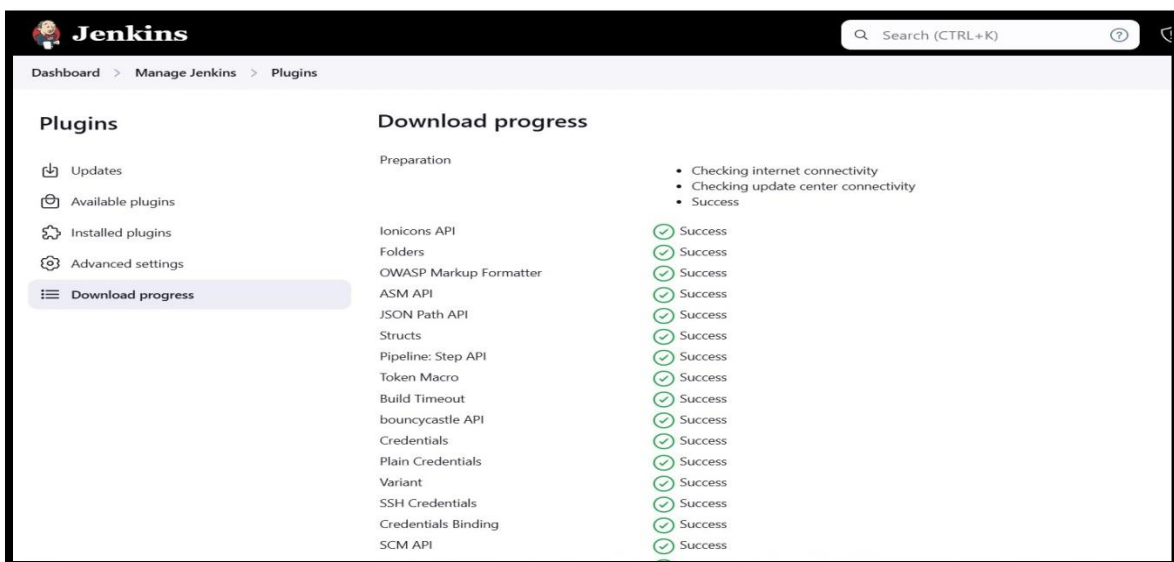
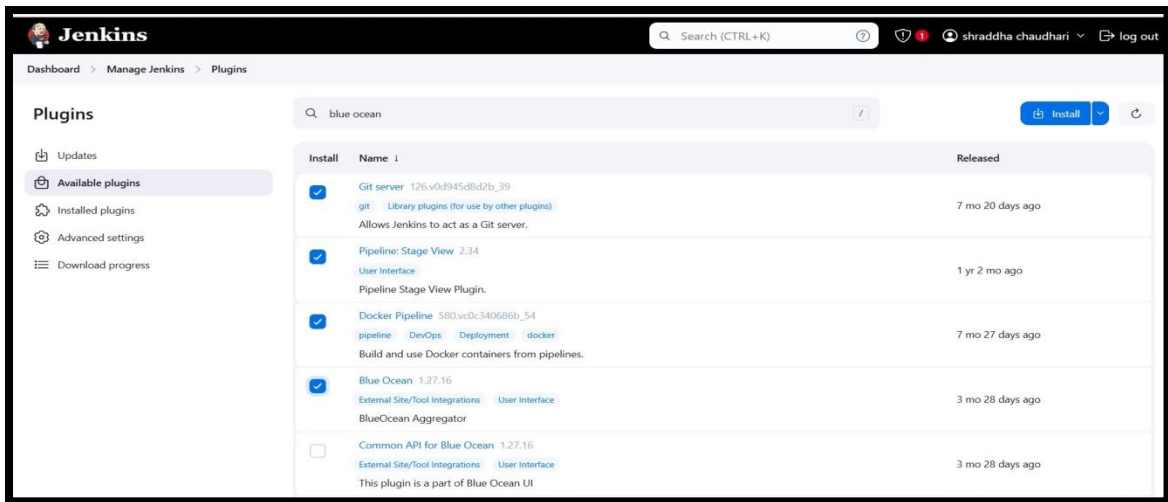
So then it will show that setup is ready.



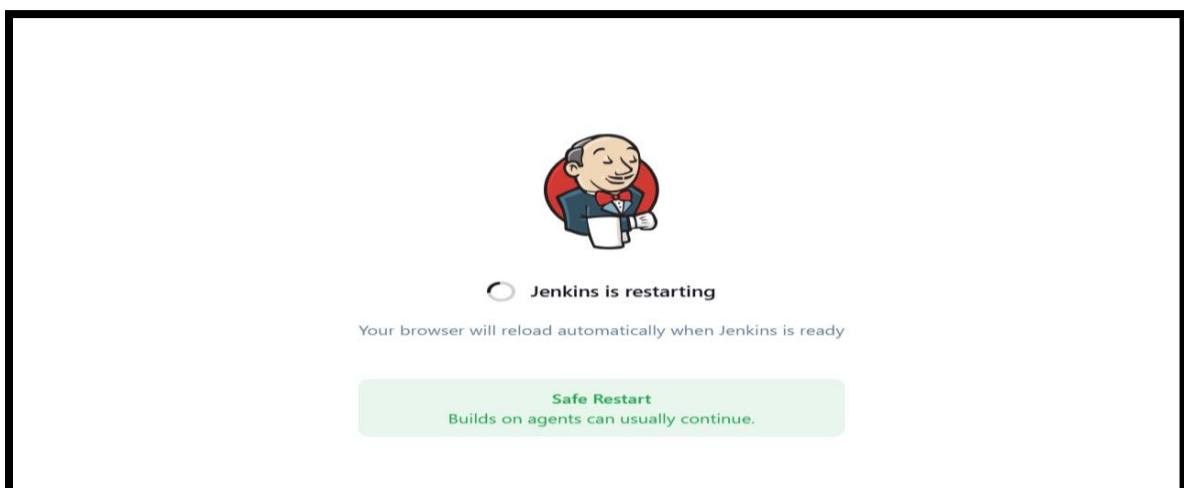
Once done with all setup next have to install required plugins.

4.8: Install required Jenkins plugins:

- Go to **Manage Jenkins** → **Manage Plugins** → **Available**.
- Install:
 - GitHub Integration Plugin
 - Pipeline
 - Docker Pipeline
 - Docker Commons



Once install all plugins then next have to restart jenkins.



It's done with setup and plugins.

4.9: Configured Jenkins Credentials :

For that we need to follow below steps:

- Go to **Manage Jenkins** → **Manage Credentials**.
- Add GitHub credentials (username and personal access token).
- Add DockerHub credentials if pushing images.

5.0: Install Docker on Jenkins server:

Follow below cmds to install docker on jenkins server

```
sudo apt install docker.io -y  
sudo usermod -aG docker jenkins  
sudo systemctl restart jenkins
```

5.1: Project Structure:

We need to follow this below project structure

```
project-root/  
├─ app/  
│  ├─ __init__.py  
│  ├─ main.py  
│  └─ ... (other app files)  
└─ tests/
```

```
| |— test_main.py
| |— ... (other test files)
|— requirements.txt
|— Dockerfile
|— Jenkinsfile
|— README.md
```

5.2: Create app directory

`mkdir app/`

Then navigate to app directory

`cd app/`

Once navigate to app/ directory so in this directory we have to create main.py file

`nano main.py`

In this file setup flask application

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, World!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

5.3: Create tests directory

`mkdir tests/`

Navigate to tests/ directory

`cd tests/`

We have to create test_main.py created this file for run unit test

`nano test_main.py`

```
def test_sample():  
    assert 1 + 1 == 2
```

5.4: Create requirements.txt

Used in python project to list all the dependencies required for the project to run.

`nano requirements.txt`

```
flask  
pytest
```

5.5: Create Dockerfile

Used to create docker image all necessary configurations includes in dockerfile required to build image

`nano Dockerfile`

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python", "app/main.py"]
```

5.6: Create Jenkinsfile

It is script that defines a Jenkins pipeline which automates the build, test, and deployment process of software.

nano Jenkinsfile

```
jenkinsfile
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'shaani/ci-cd-pipeline-image'
        STAGING_ENV = 'staging-container'
    }
    stages {
        stage('Checkout Code') {
            steps {
                echo 'Checking out code from Github...'
                git credentialsId: 'github-credentials', branch: 'main', url: 'https://github.com/shaanicha/CI-CD-pipeline.git'
            }
        }
        stage('Install Dependencies') {
            steps {
                echo 'Installing Python dependencies...'
                script {
                    // Create a virtual environment
                    sh 'python3 -m venv venv'
                    // Activate the virtual environment and install dependencies
                    sh '. venv/bin/activate && pip install -r requirements.txt'
                }
            }
        }
        stage('Run Unit Tests') {
            steps {
                echo 'Running unit tests...'
                script {
                    // Activate virtual environment and run tests
                    sh '. venv/bin/activate && pytest tests/'
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                echo 'Building Docker image...'
                sh 'docker build -t $DOCKER_IMAGE .'
            }
        }
    }
}
```

```

pipeline {
  stages {
    stage('Build Docker Image') {
    }
    stage('Push Docker Image') {
      steps {
        echo 'Pushing Docker image to Docker Hub...'
        withDockerRegistry(credentialsId: 'docker-credentials', url: '') {
          sh 'docker push $DOCKER_IMAGE'
        }
      }
    }
    stage('Deploy to Staging') {
      steps {
        echo 'Deploying Docker container to staging environment...'
        sh """
        docker stop $STAGING_ENV || true
        docker rm $STAGING_ENV || true
        docker run -d --name $STAGING_ENV -p 5000:5000 $DOCKER_IMAGE
        """
      }
    }
  }
  post {
    always {
      echo 'Pipeline execution completed.'
    }
    success {
      echo 'Pipeline executed successfully.'
    }
    failure {
      echo 'Pipeline failed. Check the logs for details.'
    }
  }
}

```

This provided files and project structure mentioned in above GitHub link.

6.0: Initialize a git repository:

We have to push all created project structure to github

So first need to install git on Ubuntu server. Once done then do git initialize

`git init`

`git add .`

Add modify changes

`git commit -m "new ci-cd pipeline"`

```

root@ip-172-31-20-30:/home/ubuntu# git init
Reinitialized existing Git repository in /home/ubuntu/.git/
root@ip-172-31-20-30:/home/ubuntu# git add .
root@ip-172-31-20-30:/home/ubuntu# git commit -m "ci/cd pipeline"
[master (root-commit) c2c474e] ci/cd pipeline
Committer: root <root@ip-172-31-20-30.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

12 files changed, 243 insertions(+)
create mode 100644 .bash_history
create mode 100644 .bash_logout
create mode 100644 .bashrc
create mode 100644 .cache/motd.legal-displayed
create mode 100644 .profile
create mode 100644 .ssh/authorized_keys
create mode 100644 .sudo_as_admin_successful
create mode 100644 Dockerfile
create mode 100644 Jenkinsfile
create mode 100644 app/main.py
create mode 100644 requirements.txt
create mode 100644 tests/test_main.py
root@ip-172-31-20-30:/home/ubuntu#

```

6.1: Push to GitHub:

- Create a new repository on GitHub (e.g., [my-ci-cd-pipeline](#)).

Link the repository:

```
git remote add origin https://github.com/your-username/my-ci-cd-pipeline.git
```

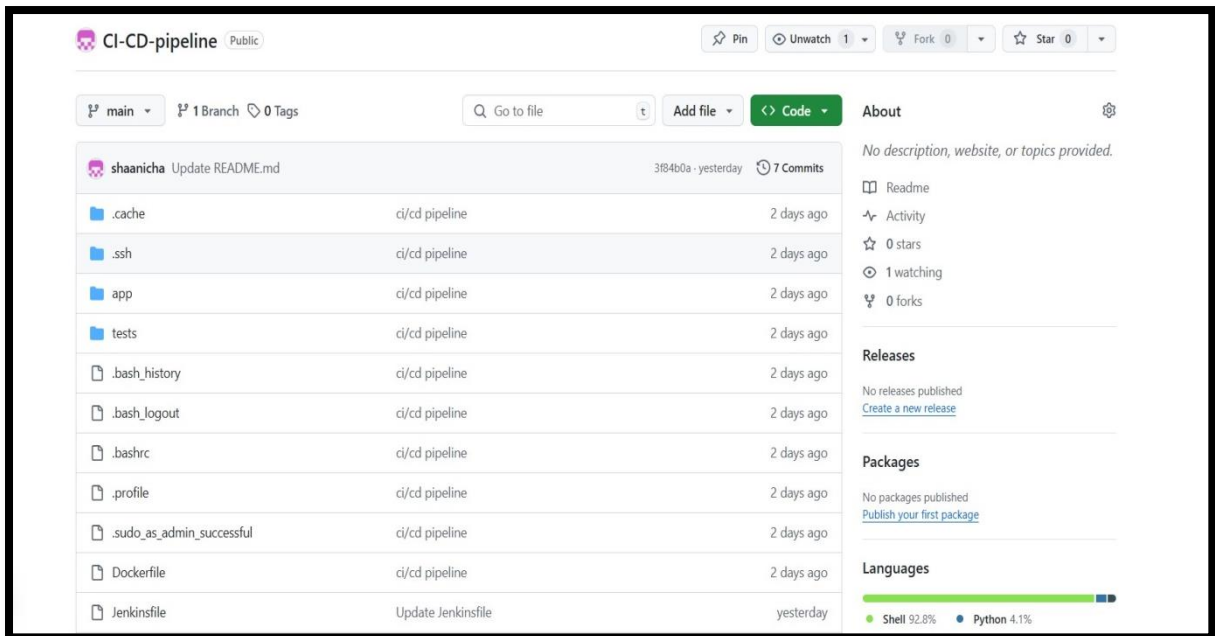
```
git branch -M main
```

- `git push -u origin main`

```

root@ip-172-31-20-30:/home/ubuntu# git push origin main
Username for 'https://github.com': shaanicha
Password for 'https://shaanicha@github.com':
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 2 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (17/17), 4.40 KiB | 4.40 MiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/shaanicha/CI-CD-pipeline.git
e3d35c0..c38fc15  main -> main

```



7.0: Setup Jenkins pipeline job:

Already we setup Jenkins now we have to create new pipeline so we have to create new job and pipeline

7.1: Create a New Job:

- Go to the Jenkins dashboard, click **New Item**.
- Enter a name (e.g., **My-CI-CD-Pipeline**), select **Pipeline**, and click **OK**.

New Item

Enter an item name

my-ci-cd-pipeline

Select an item type

- Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different namespaces.

OK

7.2: Configure the Pipeline:

Under **Pipeline**, select **Pipeline script from SCM**.

Set:

SCM: Git.

Repository URL: <https://github.com/your-username/my-ci-cd-pipeline.git>.

Branch: **main**.

7.3: Save the Job.

Configure

Advanced

General

Advanced Project Options

Pipeline

Pipeline

Definition

Pipeline script from SCM

SCM ?

None

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

Pipeline Syntax

Save Apply

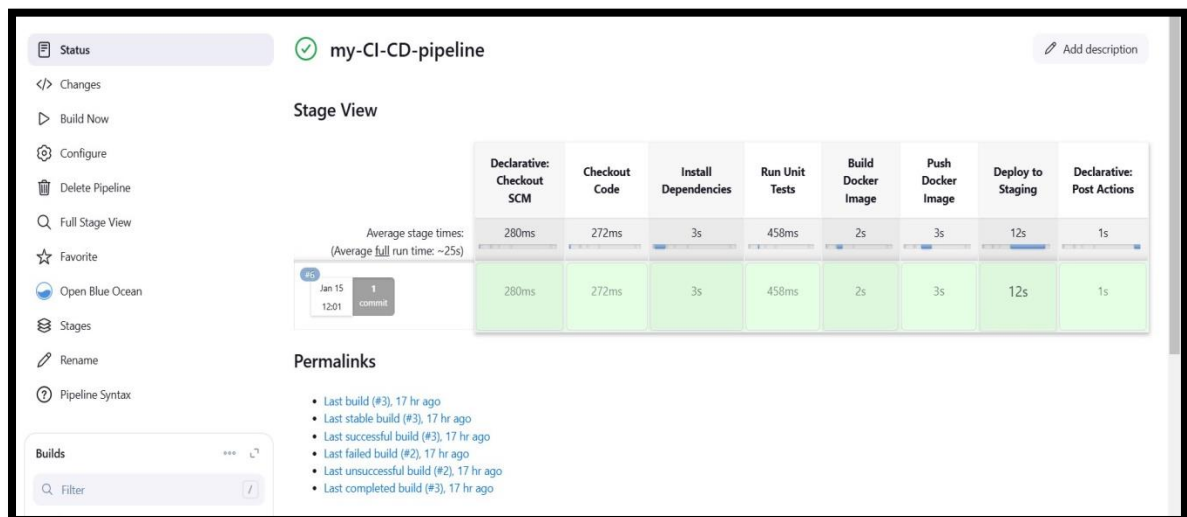
8.0: Test the pipeline:

8.1: Trigger a Build:

- Click **Build Now**.

8.2: Observe the Stages:

- Jenkins will:
 - Pull code from GitHub.
 - Install dependencies (**pip install**).
 - Run unit tests (**pytest**).
 - Build a Docker image.
 - Push the image to Docker Hub.
 - Deploy the container to the staging environment.



Pipeline run successfully.

9.0: Verify Deployment:

9.1: Check the Docker Container:

SSH into your EC2 instance and run:

`docker ps`



```
root@ip-172-31-20-30:/home/ubuntu# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
606c40fc089f	shaani/ci-cd-pipeline-image	"python app/main.py"	2 minutes ago	Up 2 minutes	0.0.0.0:5000->5000/tcp, :::5000->5000/tcp	staging-container

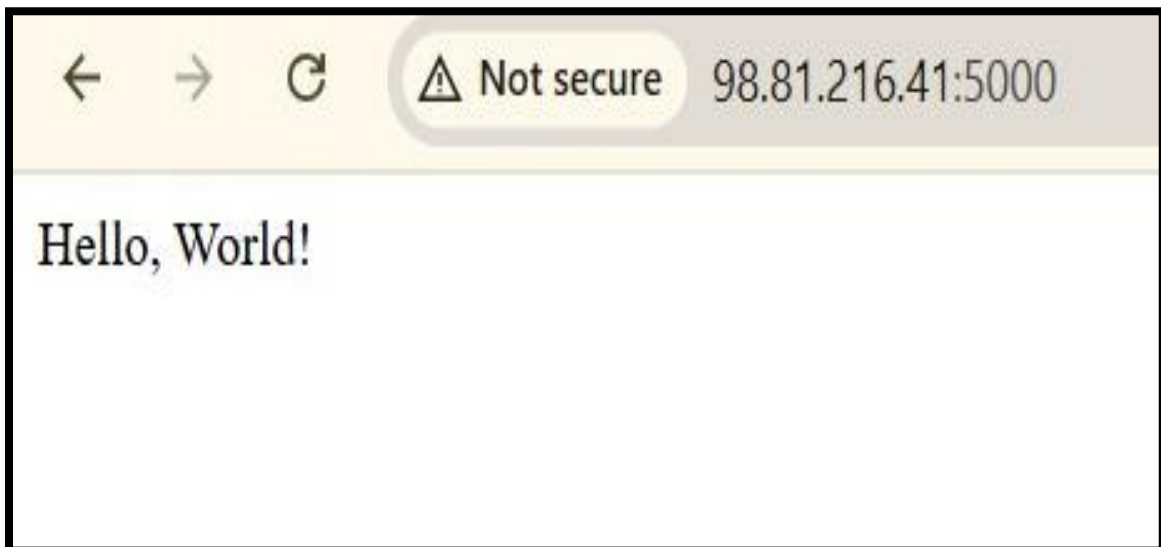
```
root@ip-172-31-20-30:/home/ubuntu#
```

You should see your container running.

9.2: Access the Application:

Visit <http://<EC2-PUBLIC-IP>:5000>.

You should see the message: **"Hello, World!"**.



Application access successfully.

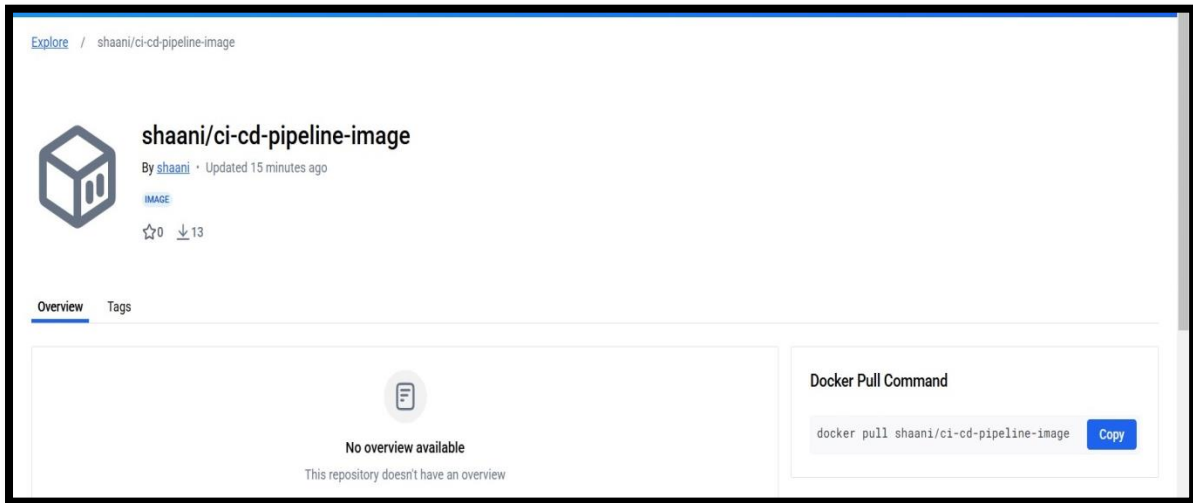
9.3: Verify Docker image:

Once our application access then check docker image correctly push to dockerhub.

Go to dockerhub

Login to dockerhub

Then It will showing our updates image



10: Integrate Notifications:

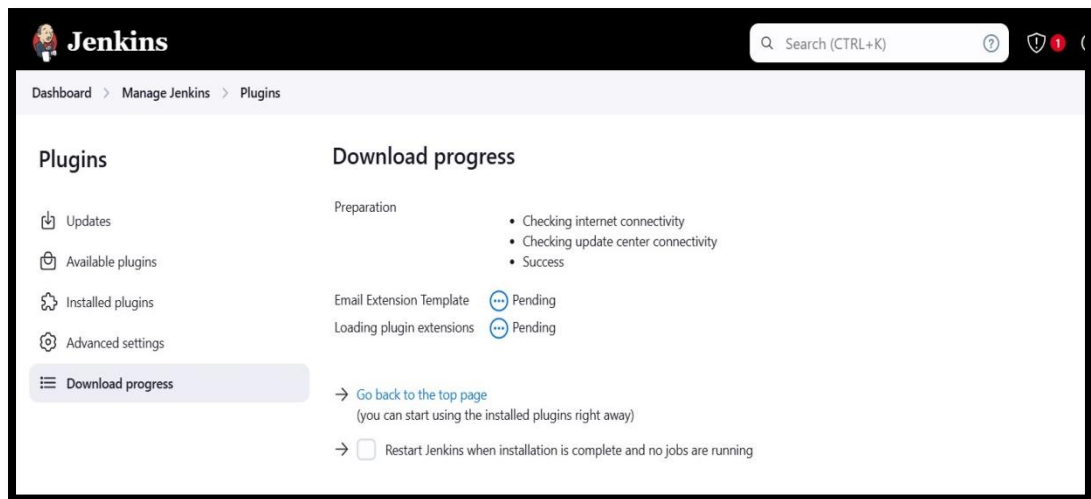
- Configure Jenkins to send Slack/email alerts on pipeline failures.

To configure Jenkins to send email notifications for pipeline failures,

Follow these steps:

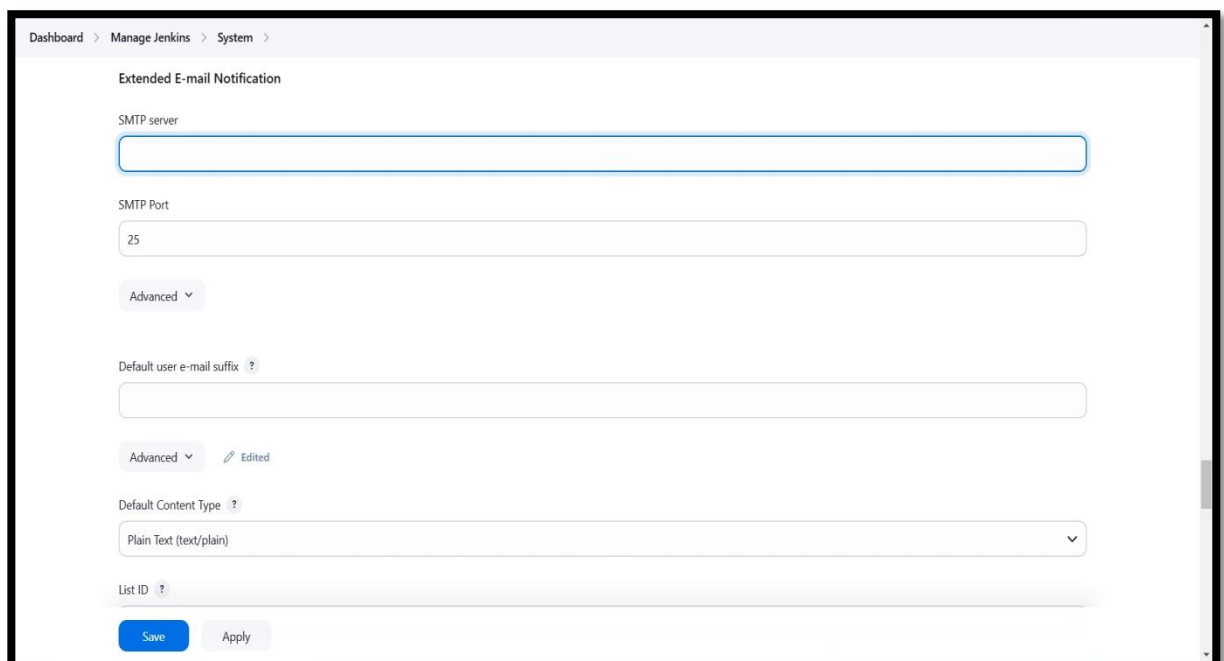
Step 1: Install Email Extension Plugin

1. Go to the **Jenkins Dashboard**.
2. Navigate to **Manage Jenkins > Manage Plugins**.
3. Under the **Available** tab, search for **Email Extension Plugin** (or verify it's installed under **Installed**).
4. Install the plugin and restart Jenkins if necessary.



Step 2: Configure Jenkins Email Notification Settings

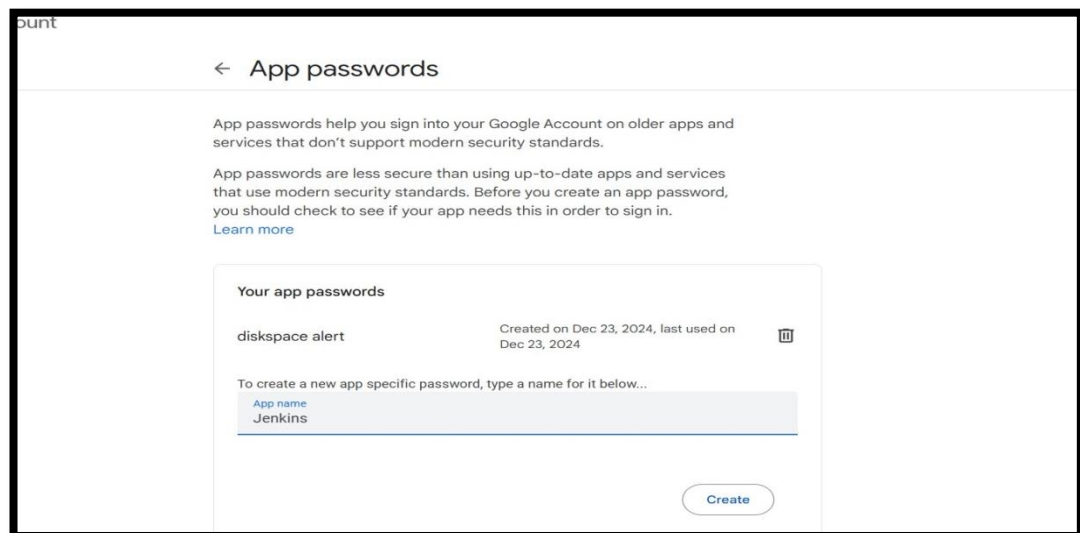
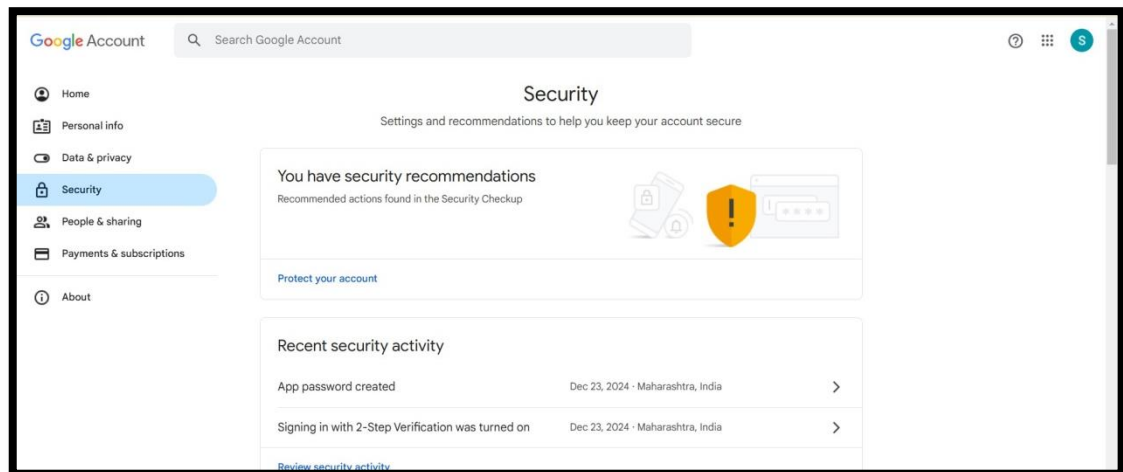
1. Navigate to **Manage Jenkins > Configure System**.
2. Scroll down to the **Extended E-mail Notification** section.



3. Fill in the following details based on your email provider:

- **SMTP Server:** e.g., **smtp.gmail.com** for Gmail.
- **Use SMTP Authentication:** Check this box.

- **User Name:** Your email address, e.g., your-email@gmail.com.
 - **Password:** Your email's application-specific password or token (if required by your email provider).
4. For that, create an app password for email: Go to Google account > security > search for app password > give name “Jenkins” or anything > copy password and paste.



5. **SMTP Port:** 465 (for Gmail or any email provider using TLS).

- **Use SSL:** Uncheck (TLS will be used instead).
- **Reply-To Address:** The email address for replies, e.g., your-email@gmail.com.
- **Charset:** Leave as UTF-8.

6. Click **Test Configuration by Sending Test E-mail** to verify the settings.
 - Enter a recipient email address and ensure the test email is received.



Dashboard > Manage Jenkins > System >

Advanced [⌵] Edited

☒ Use SMTP Authentication [?]

User Name

shraddhachaudhari1730@gmail.com

⚠ For security when using authentication it is recommended to enable either TLS or SSL

Password

☒ Use SSL [?]

☐ Use TLS

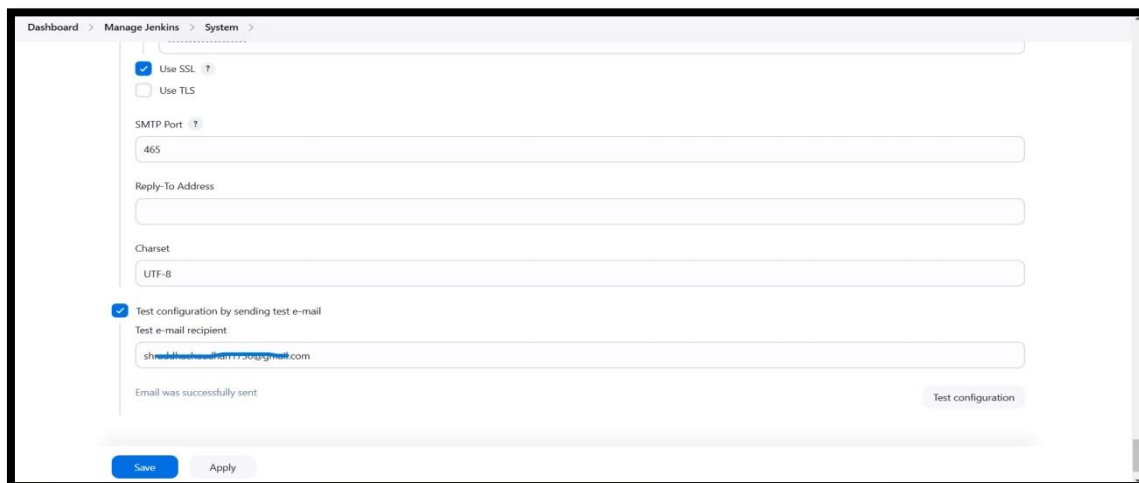
SMTP Port [?]

465

Reply-To Address

Charset

Save Apply



Dashboard > Manage Jenkins > System >

☒ Use SSL [?]

☐ Use TLS

SMTP Port [?]

465

Reply-To Address

Charset

UTF-8

☒ Test configuration by sending test e-mail

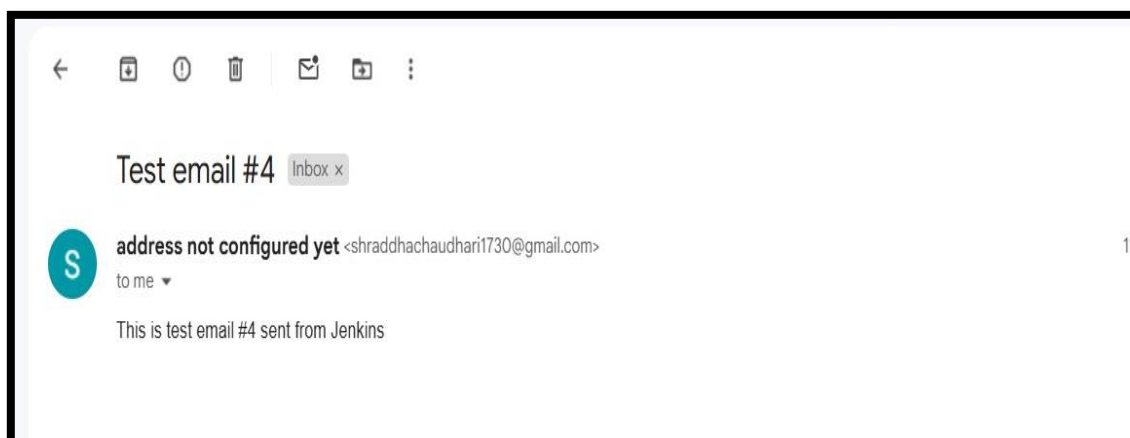
Test e-mail recipient:

shraddhachaudhari1730@gmail.com

Email was successfully sent

Test configuration

Save Apply



This is test email notification. Means email integration correctly.

7. Step 3: Modify the Jenkinsfile for Notifications

To modify the Jenkinsfile so that it intentionally fails (for testing the notification system), we can introduce an artificial error in one of the pipeline stages. Here's an updated Jenkinsfile with the intentional failure included:



```

pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'shaani/ci-cd-pipeline-image'
        STAGING_ENV = 'staging-container'
    }
    stages {
        stage('Checkout Code') {
            steps {
                echo 'Checking out code from GitHub...'
                git credentialsId: 'github-credentials', branch: 'main', url: 'https://github.com/shaanicha/CI-CD-pipeline.git'
            }
        }
        stage('Install Dependencies') {
            steps {
                echo 'Installing Python dependencies...'
                script {
                    sh 'python3 -m venv venv'
                    sh '. venv/bin/activate && pip install -r requirements.txt'
                }
            }
        }
        stage('Run Unit Tests') {
            steps {
                echo 'Running unit tests...'
                script {
                    sh '. venv/bin/activate && pytest tests/'
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                echo 'Building Docker image...'
                sh 'docker build -t $DOCKER_IMAGE .'
            }
        }
        stage('Push Docker Image') {

```

```

        stage('Push Docker Image') {
            steps {
                echo 'Pushing Docker image to Docker Hub...'
                withDockerRegistry(credentialsId: 'docker-credentials', url: '') {
                    sh 'docker push $DOCKER_IMAGE'
                }
            }
        }
        stage('Deploy to Staging') {
            steps {
                echo 'Deploying Docker container to staging environment...'
                sh """
                docker stop $STAGING_ENV || true
                docker rm $STAGING_ENV || true
                docker run -d --name $STAGING_ENV -p 5000:5000 $DOCKER_IMAGE
                """
            }
        }
        stage('Intentional Failure') {
            steps {
                echo 'Introducing a failure for testing notifications...'
                script {
                    error("This is an intentional failure for testing!")
                }
            }
        }
    }
    post {
        always {
            echo 'Pipeline execution completed.'
        }
        failure {
            echo 'Pipeline failed. Sending email notification...'
            emailext(
                subject: "Jenkins Pipeline Failure: ${env.JOB_NAME} #${env.BUILD_NUMBER}",

```



```

pipeline {
  stages {
    stage('Intentional Failure') {
      steps {
        echo 'Introducing a failure for testing notifications...'
        script {
          error("This is an intentional failure for testing!")
        }
      }
    }
  }
  post {
    always {
      echo 'Pipeline execution completed.'
    }
    failure {
      echo 'Pipeline failed. Sending email notification...'
      emailtext(
        subject: "Jenkins Pipeline Failure: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
        body: """
          <p>The pipeline has failed during execution.</p>
          <p>Details:</p>
          <ul>
            <li><b>Job:</b> ${env.JOB_NAME}</li>
            <li><b>Build Number:</b> ${env.BUILD_NUMBER}</li>
            <li><b>URL:</b> <a href="${env.BUILD_URL}">${env.BUILD_URL}</a></li>
          </ul>
          """,
        recipientProviders: [[class: 'DevelopersRecipientProvider']],
        to: 'your-recipient@example.com',
        mimeType: 'text/html'
      )
    }
  }
}

```

11. Changes Made:

1. Added an **Intentional Failure** Stage:

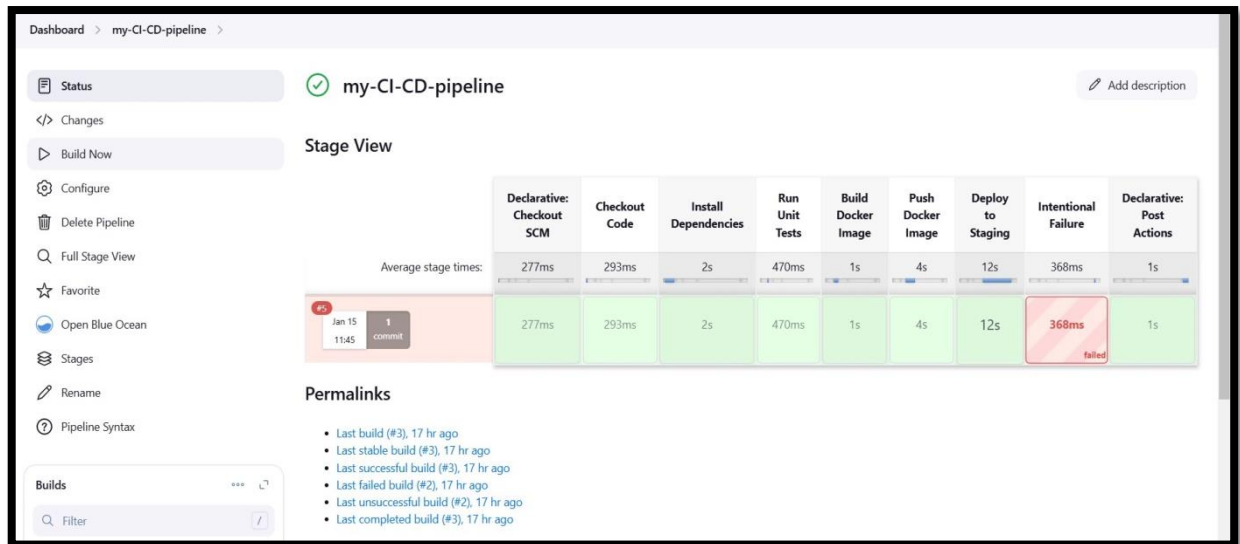
- This stage includes a deliberate `error()` step to make the pipeline fail for testing purposes.

2. Post-Failure Notification:

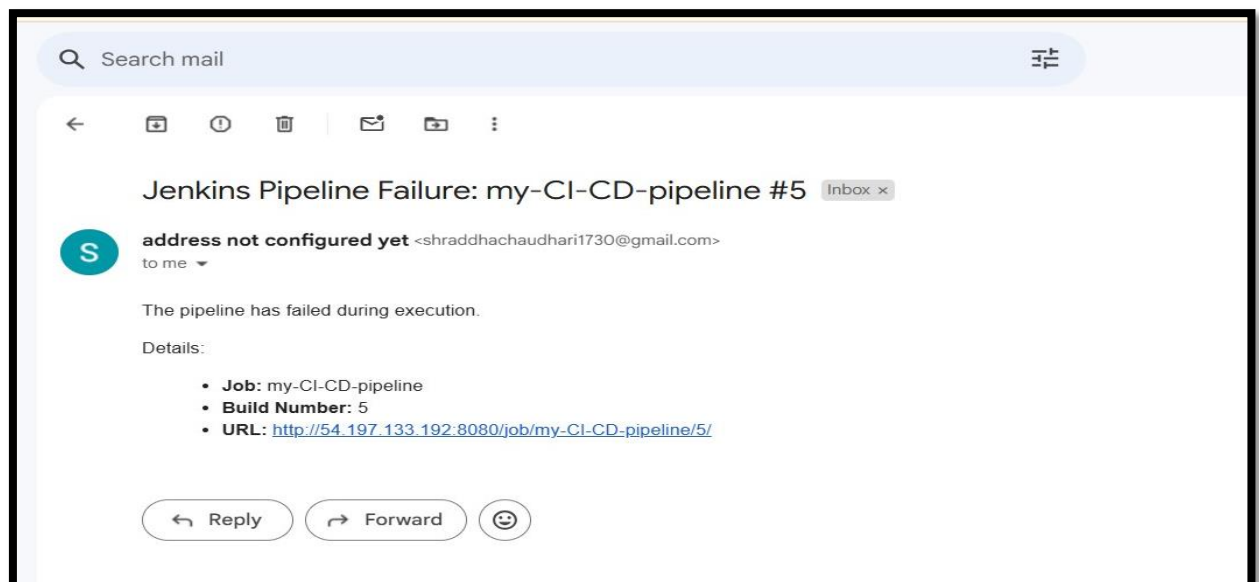
- The `failure` block in the `post` section ensures an email is sent when the pipeline fails.

12. Steps to Test:

1. Replace `your-recipient@example.com` with the email address where notifications should be sent.
2. Commit the updated Jenkinsfile to your repository.
3. Trigger a new build in Jenkins.



4. The pipeline will fail at the **Intentional Failure** stage, and you should receive an email notification with failure details.



Configure Jenkins to send automated notifications on pipeline failures. Email alerts will be used to promptly inform the team about errors, ensuring timely resolution and smooth development workflows.

Note: All files related to project structure provided in above mentioned GitHub link.