DevOps list of experiments

1. Write code for a simple user registration form for an event.
2. Explore Git and GitHub commands
3. Practice Source code management on GitHub. Experiment with the source code written in exercise 1
4. Jenkins installation and setup, explore the environment
5. Demonstrate continuous integration and development using Jenkins.
6. Explore Docker commands for content management.
7. Develop a simple containerized application using Docker
8. Integrate Kubernetes and Docker
9. Automate the process of running containerized application developed in exercise 7 using Kubernetes
10. Install and Explore Selenium for automated testing
11. Write a simple program in JavaScript and perform testing using Selenium
12. Develop test cases for the above containerized application using selenium

Experiment - 1.Write code for a simple user registration form for an event.

Program Objectives:

The objective of this program is to create a simple user registration form for an event. The form will collect the user's name, age, date of birth (DOB), mobile number, and Gmail address. Upon submission, the form will process the entered data and possibly store it in a database or perform other operations as needed.

Description:

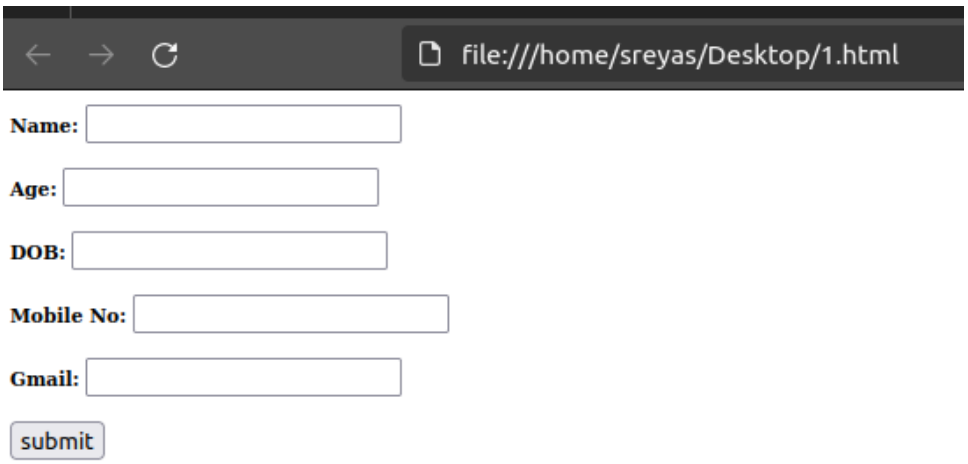
This HTML code creates a basic registration form with fields for the user to input their name, age, DOB, mobile number, and Gmail address. The form also includes a submit button for the user to submit their registration details.

Code:

```
<html>
<head>
<title>registration form</title>
</head>
<body>
<h6>
<label="name">Name:</label>
<input type="text" name="name"><br><br>

<label="age">Age:</label>
<input type="text" name="Age"><br><br>

<label="DOB">DOB:</label>
<input type="text" name="DOB"><br><br>

<label="mobile no">Mobile No:</label>
<input type="text" name="mobile no"><br><br>

<label="gmail">Gmail:</label>
<input type="text" name="Gmail"><br><br>

<input type="submit" value="submit"><br><br>

</h6>
</body>
</html>
```

Output:

A rendered HTML page will display the registration form as described above, allowing users to input their details and submit the form



Experiment-2
Explore Git and GitHub commands

Program Objectives: The objective of this experiment is to familiarize yourself with essential Git and GitHub commands commonly used in day-to-day development workflows. Through this experiment, you will learn how to initialize a Git repository, track changes, create branches, collaborate with remote repositories on GitHub, and manage version history.

Description: In this experiment, you will explore fundamental Git and GitHub commands and their functionalities. These commands will help you understand version control concepts and enable you to efficiently manage your project's source code.

List of Git and GitHub Commands:

1. **git init**: Creates a new empty Git repository in the current directory.
2. **git status**: Shows the current status of your files in the Git repository, indicating which files are staged, modified, or untracked.
3. **git add**: Stages changes (new files, modifications, deletions) in the current directory.

- Example:
- **git add .** - stages all files in the current directory.
- **git add file.txt** - stages file.txt in the Git directory.

4. **git commit**: Saves staged changes to the local repository, creating a new commit with a snapshot of the changes along with a commit message to describe the changes.

- Example: git commit -m "my first file for website design"

5. **git branch -M main**: Renames the current branch to "main".
6. **git remote add origin <URL>**: Links your local Git repository with a remote repository hosted on a service like GitHub.

- Example: git remote add origin https://github.com/NAGENDRASAICH/myproject.git

7. **git push -u origin main**: Pushes local commits from the main branch of your local repository to the remote repository named "origin" on a branch also named "main".
8. **git branch**: Lists all branches in your local repository and highlights the current branch.
9. **git checkout -b <branch_name>**: Creates a new branch.

- Example: git checkout -b sreyasnewproject

10. **git reset**: Undoes a commit.

- Example: git reset HEAD file.txt - unstages changes for a specific file named file.txt.

11. **git fetch**: Downloads changes from a remote repository into the local repository.
12. **git merge**: Merges changes from one branch into another branch.
13. **git clone**: Clones a remote repository into a new directory on your local machine.

- Example: git clone https://github.com/example/example.git

14. **git log**: Displays a history of commits in the repository.
15. **git pull**: Fetches and downloads content from a remote repository and immediately updates the local repository to match that content.

Program Outcomes: Upon completion of this experiment, you will be able to:

- Initialize a Git repository and track changes.
- Create branches and manage version history effectively.
- Collaborate with remote repositories on platforms like GitHub.
  Understand the importance of version control in software development workflows.

Experiment-3.

Practice Source code management on GitHub. Experiment with the source code written in exercise 1.

Program Objectives: The objective of this experiment is to demonstrate the process of managing source code using Git and GitHub. You will learn how to initialize a local Git repository, add and commit changes to it, and then push those changes to a remote repository on GitHub. Through this hands-on demonstration, you will understand the workflow involved in source code management and collaboration using version control systems.

Description: In this experiment, you will practice source code management using Git and GitHub. You will start by initializing a local Git repository in your project directory, adding your source code files to the repository, and committing changes with descriptive commit messages. Then, you will create a remote repository on GitHub and link it with your local repository. Finally, you will push your local commits to the remote repository on GitHub, making your source code accessible to collaborators and enabling seamless collaboration and version control.

Steps:

Step-1: Place your file in any folder

for example my file is 1.html . So , i will place that file in a folder named “projects” in my Desktop location

step-2: Open terminal in the same location where your file (1.html) is located and type following commands:

git init

it will initialize new empty Git repository in the current directory where, 1.html is located

git status

it Shows the current status of your files in the Git repository

initially, my file 1.html is not stored it git repository , so it will show 1.html it red colour, which indicates that our file 1.html is not transferred to git repositroy, now we need to move our file to git repository

git add .

It will move all files which are present in my folder to git repository

git status

now, status will display “1.html” in green colour, which indicates that our file “1.html” is transferred to git repository

“this is how we transfer files to git repository”

now other task is to transfer files from git to github repository:

step-1:

open git hub website

step 2:

generate token and save it any location in your system
open github website -> settings -> Developer settings -> personal access tokens -> Token (classic)

->generate new token -> give any name to your token and accept all checkboxes > generate token
now save that token no in text editor for future use
generally we need token to trasfer files from git to github website

step-3:

now create repository with any name

for example : here, i will give my repository name as “my repositroy:

step-4:

now we need to create branch and move 1.html which is located in git repository to git hub website
so run these commands:

git commit -m “myfirstcode”
git branch -M master
git remote add origin https://github.com/NAGENDRASAICH/my-repository.git
git push -u origin master

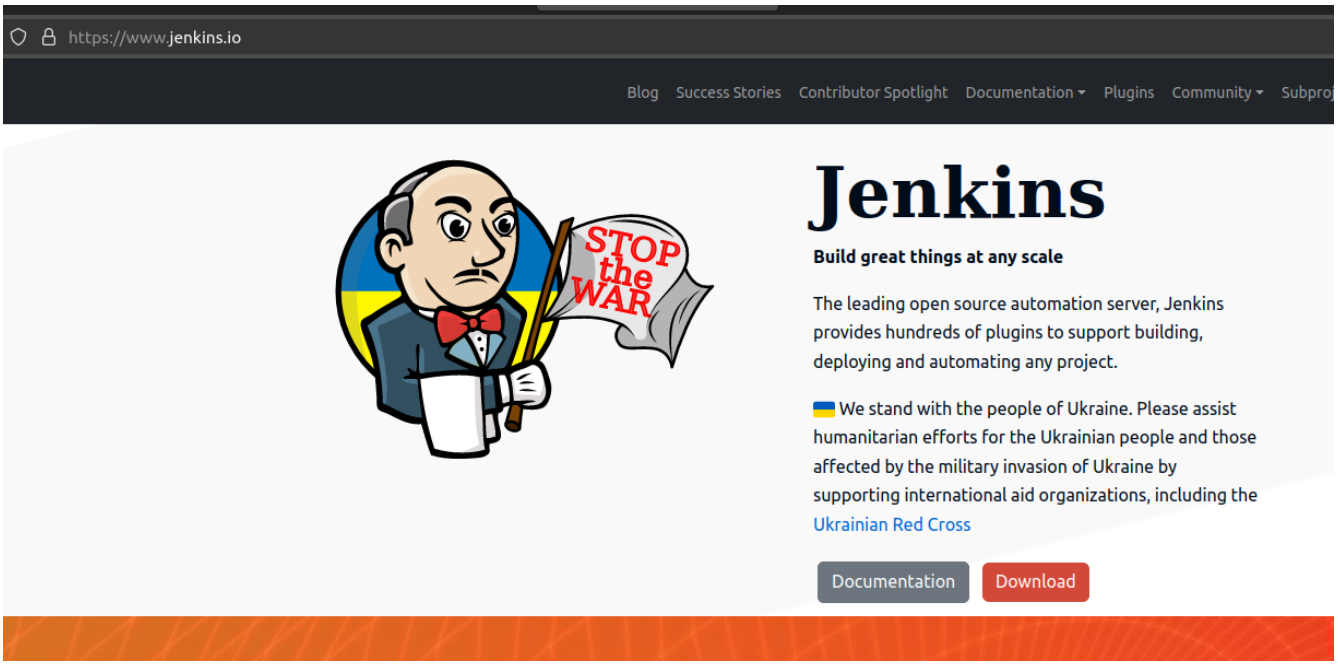
enter user name and password(token no)
now you file is successfully transferred to git hub webiste fo git repository

Program Outcomes: Upon completion of this experiment, you will be able to:

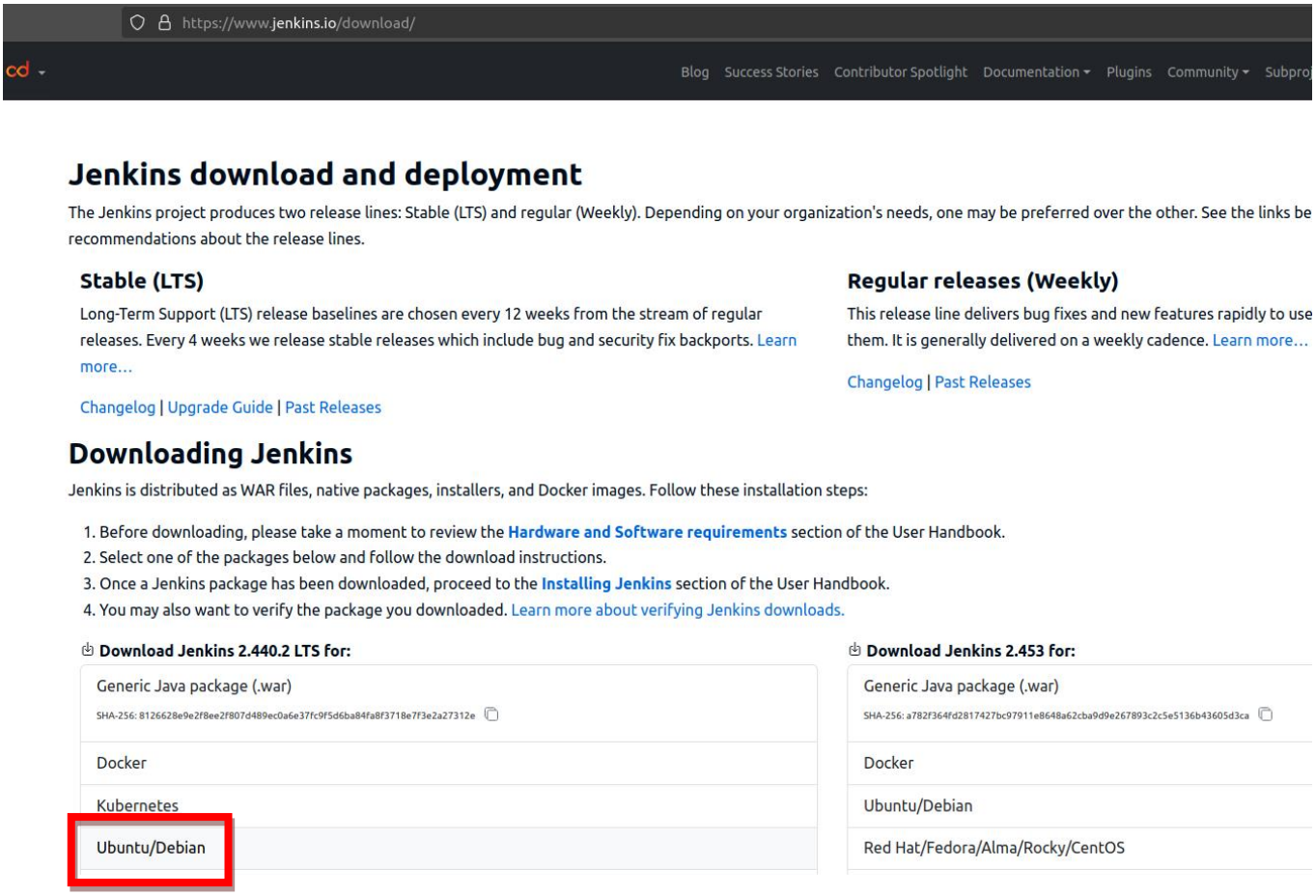
1. Initialize a local Git repository for your project.
2. Add source code files to the local repository and commit changes with meaningful commit messages.
3. Create a remote repository on GitHub to host your project.
4. Link your local Git repository with the remote repository on GitHub.
5. Push your local commits to the remote repository on GitHub, enabling collaboration and version control.
6. Understand the importance of source code management and version control in software development projects.

Experiment-4 : Jenkins installation and setup, explore the Environment

Step-1: Visit official website : https://www.jenkins.io



As I am using ubantu OS – I will select Unantu/Debian



Jenkins Debian Packages

This is the Debian package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system (for the Weekly Release Line):

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

Then add a Jenkins apt repository entry:

```
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
```

Update your local package index, then finally install Jenkins:

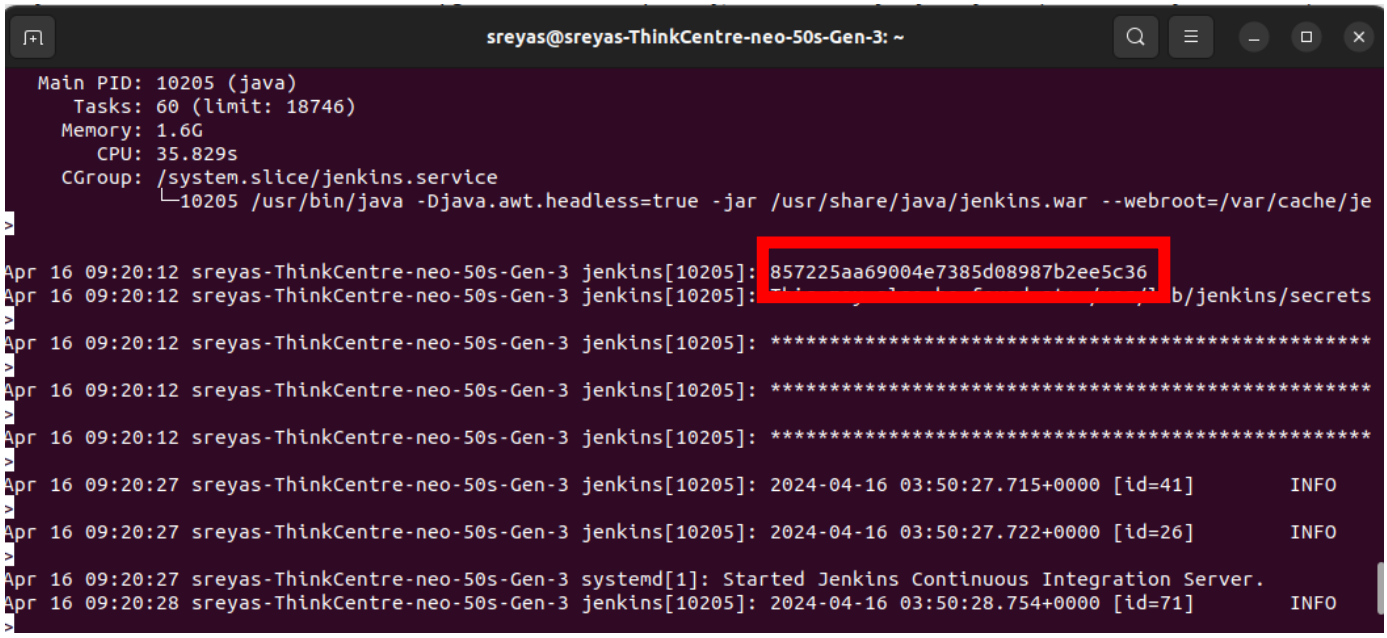
```
sudo apt-get update
sudo apt-get install fontconfig openjdk-17-jre
sudo apt-get install jenkins
```

The apt packages were signed using this key:

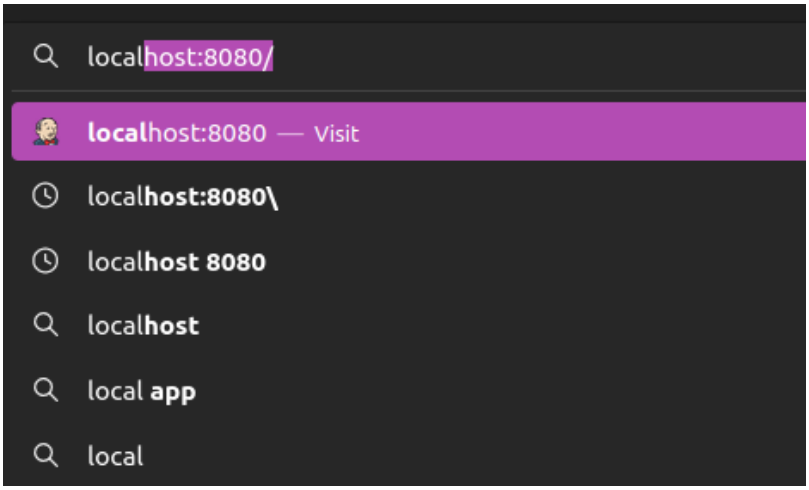
```
pub   rsa4096 2023-03-27 [SC] [expires: 2026-03-26]
      63667EE74BBA1F0A08A698725BA31D57EF5975CA
uid                      Jenkins Project
sub   rsa4096 2023-03-27 [E] [expires: 2026-03-26]
```

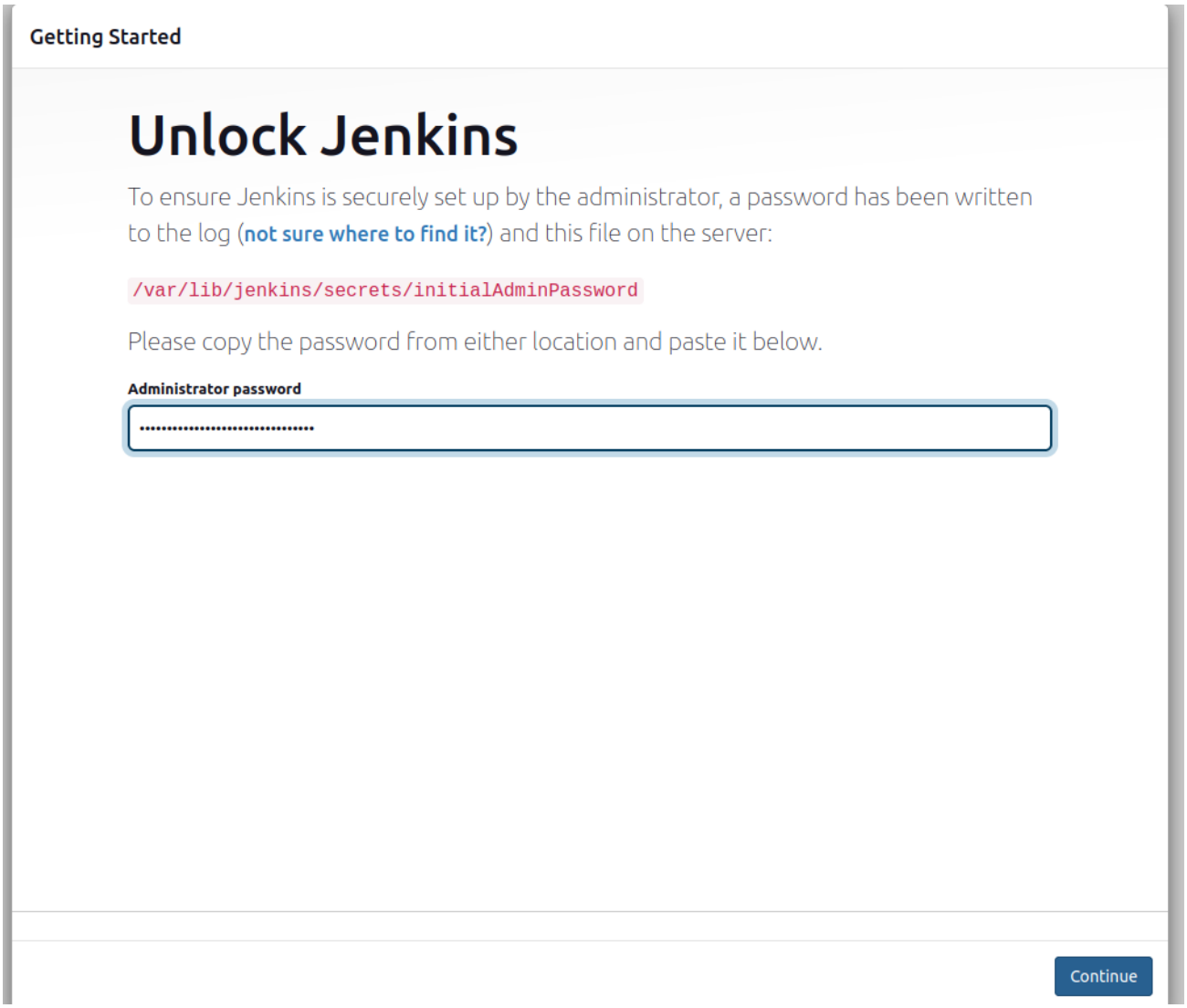
You will need to explicitly install a supported Java runtime environment (JRE), either from your distribution (as described above) or another Java vendor (e.g., Adoptium).

Now run all these commands in your terminal
So, after running all the above commands , Jenkins will be installed in your system
<u>To start jenkins not run below command:</u>
sudo systemctl start Jenkins
<u>To see Jenkins is active or not run below command:</u>
sudo systemctl status Jenkins

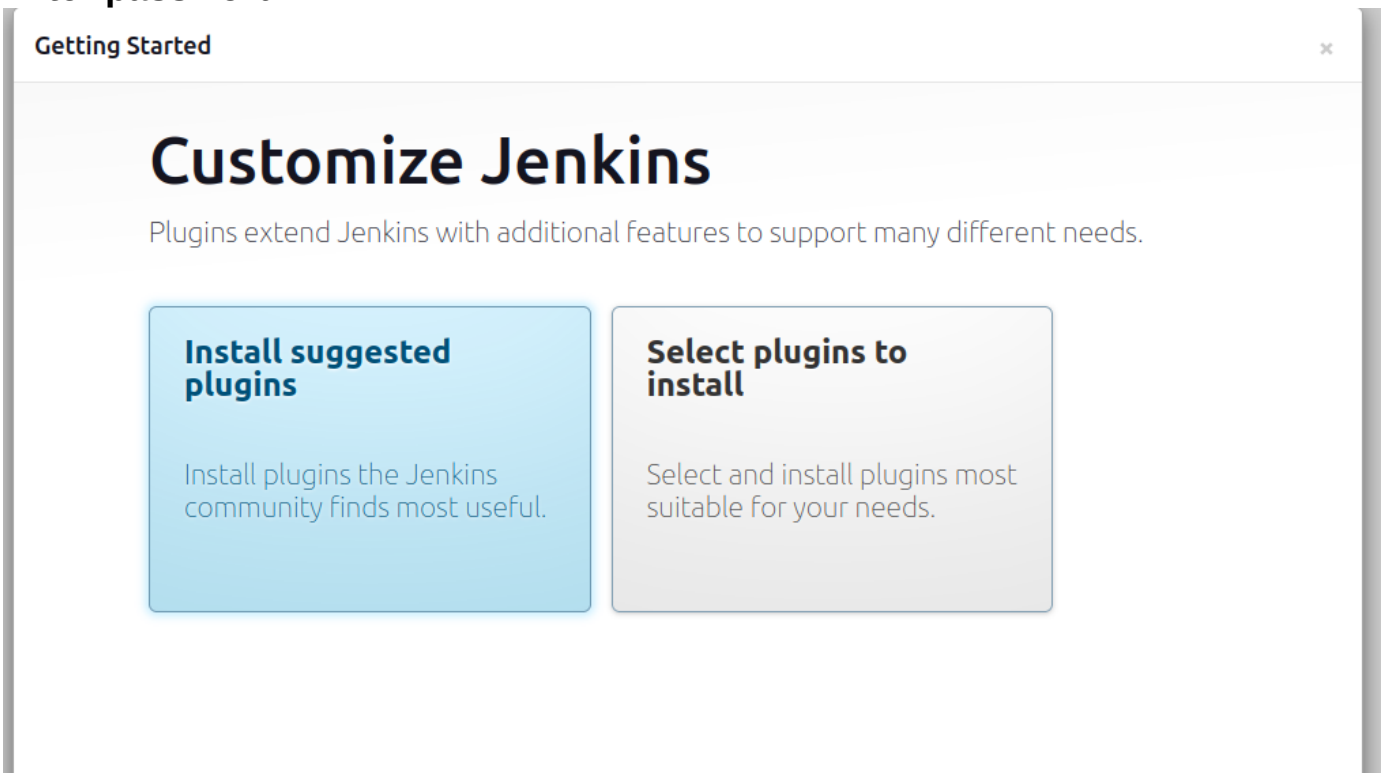


Now above one is the password to open Jenkins , now minimize terminal and open any web browser and type https://localhost:8080 to open Jenkins

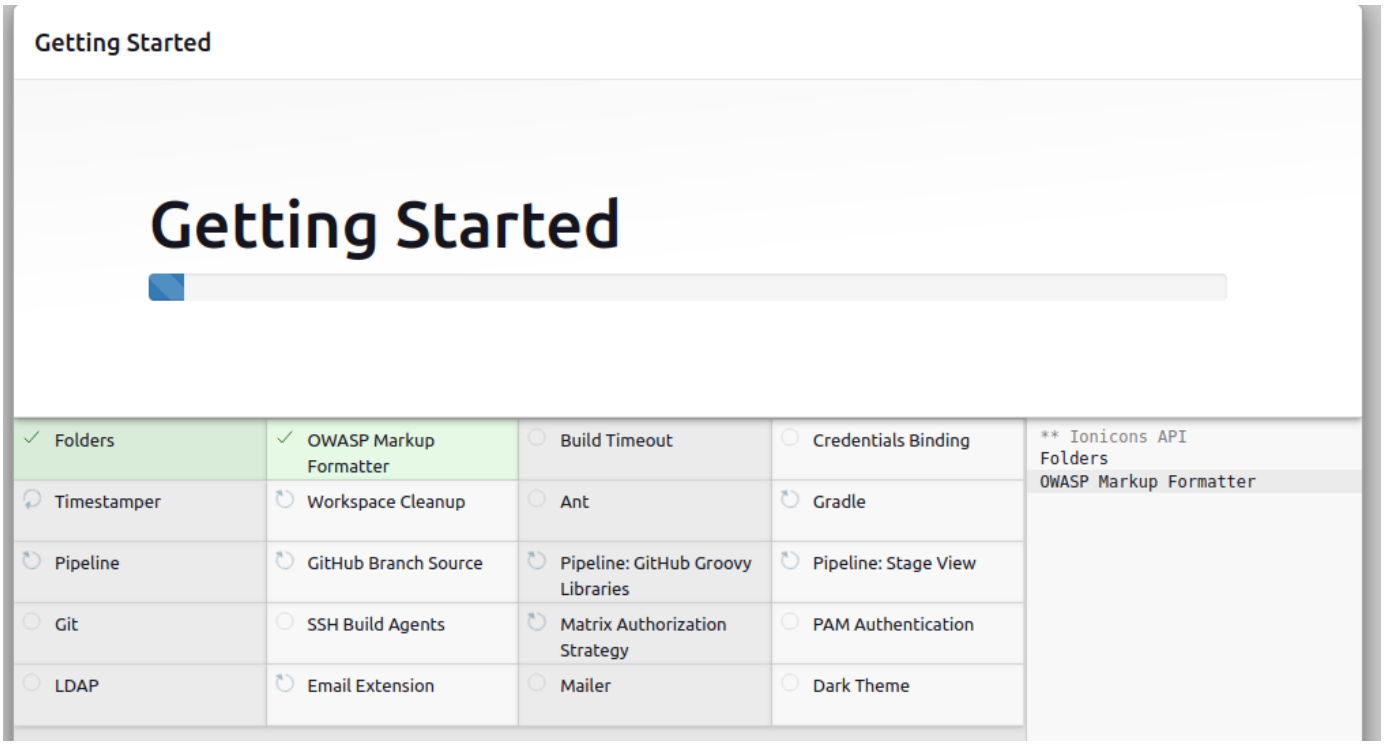

localhost:8080/
localhost:8080 — Visit
localhost:8080\
localhost 8080
localhost
local app
local

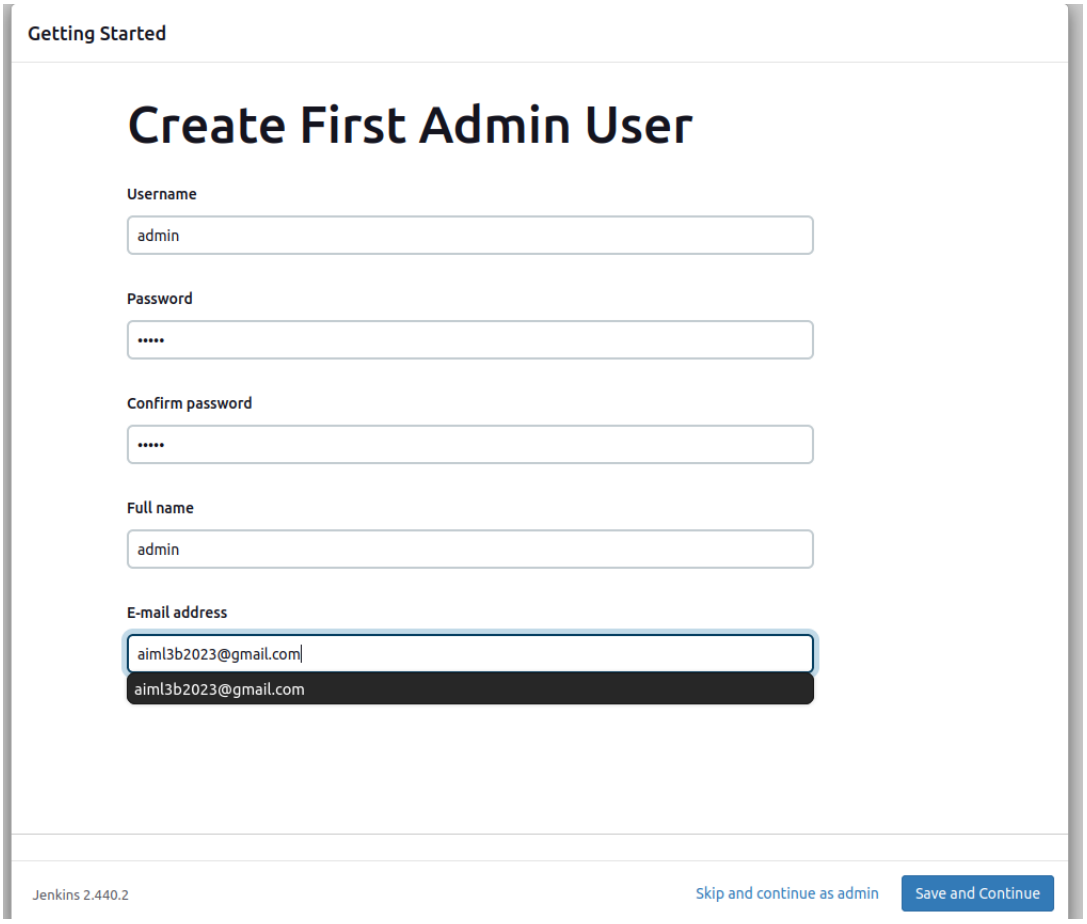

Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/lib/jenkins/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password
Continue

Enter password

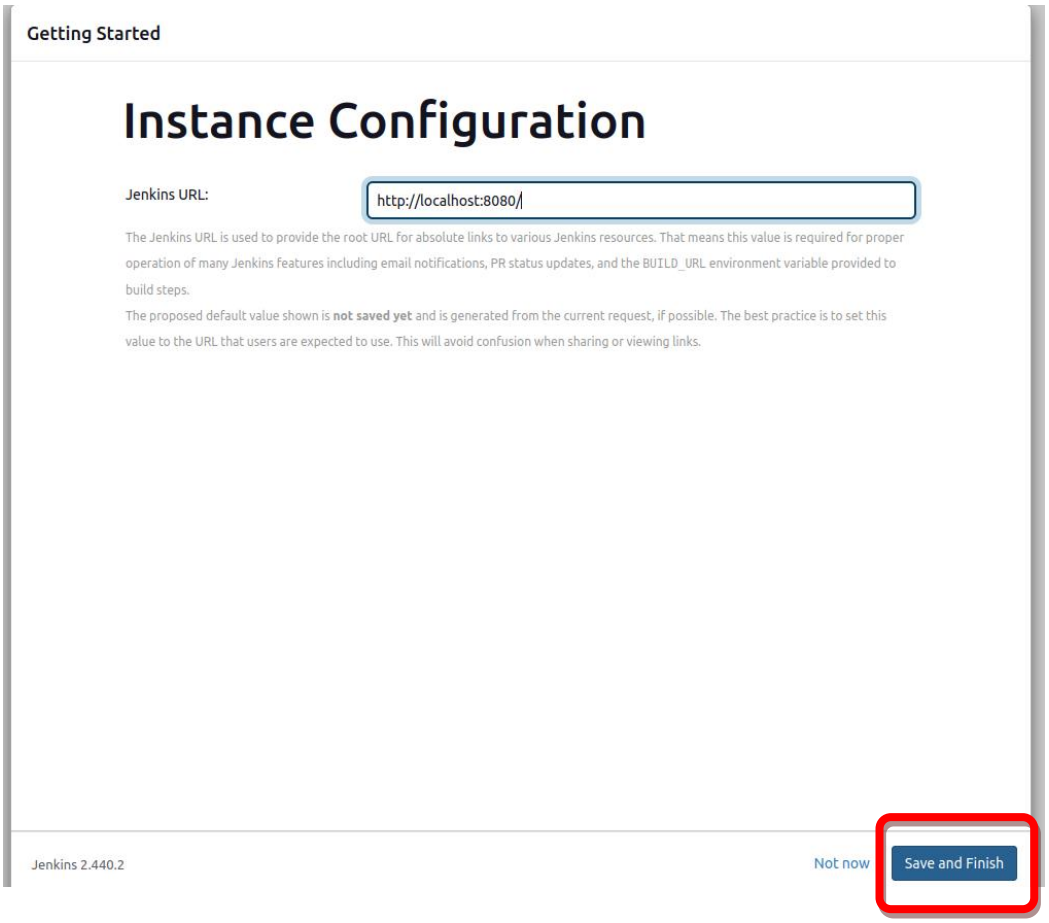


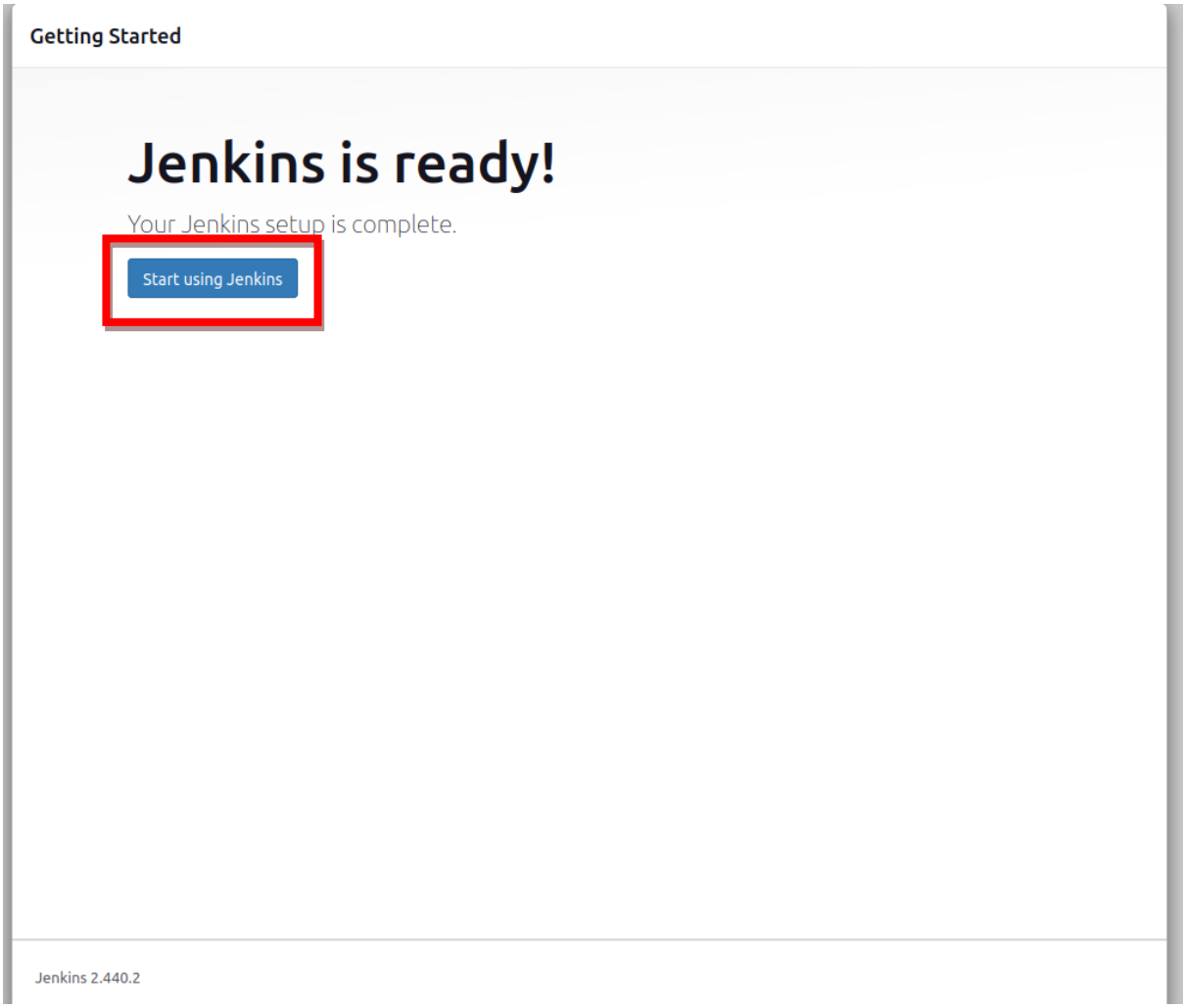
Select – install suggested plugins



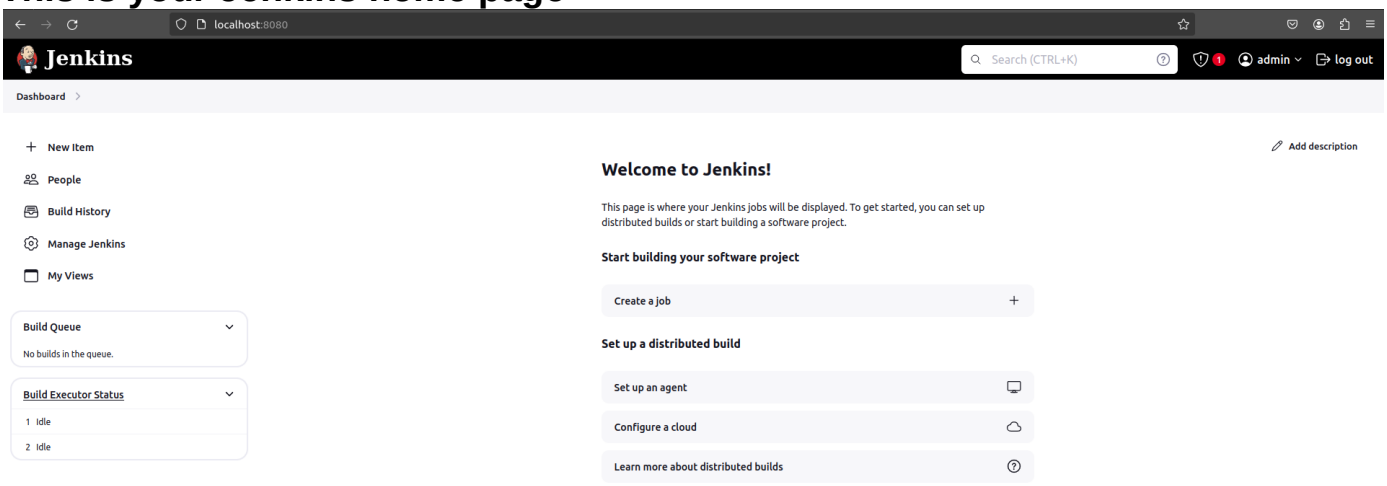


You can give user name And password, remember username and password because you need to enter it , when ever you open jekins. by default I am giving user name as admin and password as admin





This is your Jenkins home page



<u>If you want to uninstall Jenkins , you can use below commands:</u>

sudo service jenkins stop
sudo systemctl stop jenkins
sudo apt-get remove jenkins
sudo rm -rf /var/lib/jenkins
sudo rm -rf /var/lib/jenkins
sudo userdel jenkins

Experiment-5:
Demonstrate continuous integration and development using Jenkins

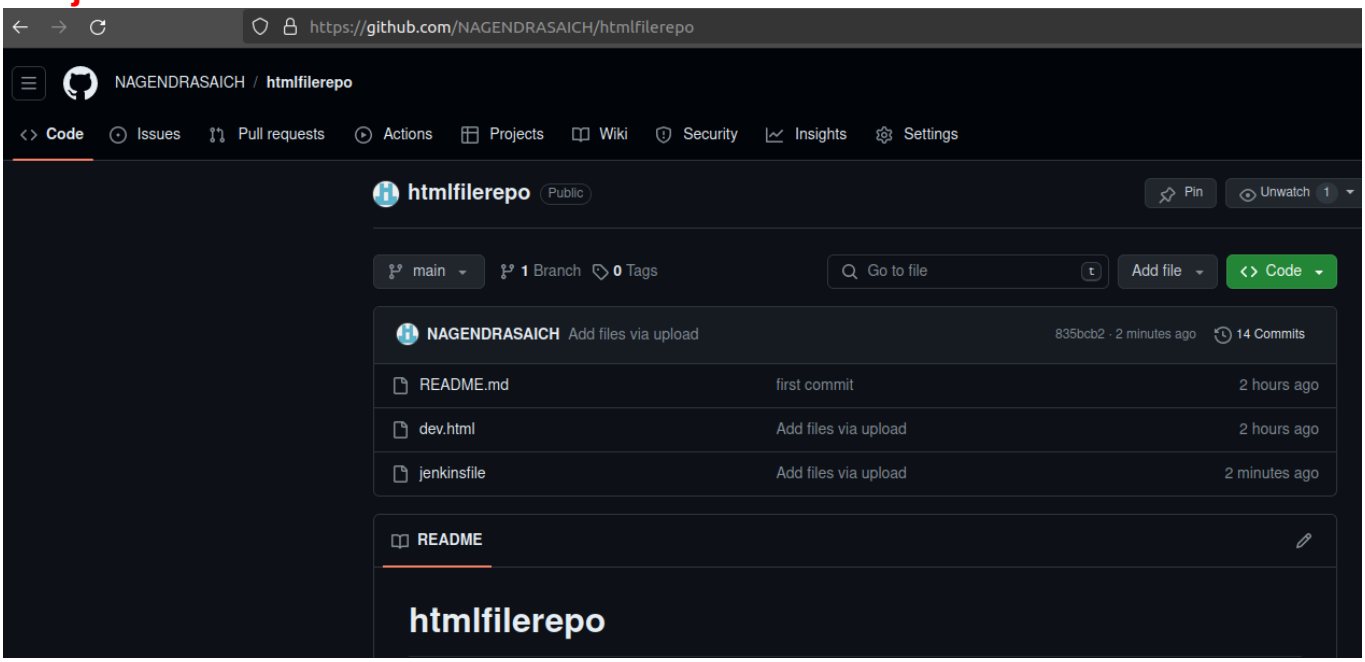
Objective: The objective of this experiment is to demonstrate the implementation of continuous integration and development using Jenkins. Specifically, the goal is to create a Jenkins pipeline that automates the build, test, and artifact generation process for a sample web application hosted on GitHub.

Description:

1. **Setup Jenkins:** Install and configure Jenkins on a server or local machine.
2. **Install Required Plugins:** Install necessary plugins in Jenkins such as Git, GitHub, and any other plugins required for the pipeline.
3. **Create Pipeline Job:** Create a new pipeline job in Jenkins.
4. **Configure Pipeline**: Configure the pipeline to fetch the source code from a GitHub repository.
5. **Define Stages:** Define stages in the pipeline for building, testing, and artifact generation.
6. **Build Stage:** In the build stage, compile the source code and package it into a deployable artifact.
7. **Test Stage:** Execute automated tests on the compiled code to ensure its quality and functionality.
8. **Artifact Generation Stage:** Generate a .rar artifact containing the necessary files, such as dev.html, from the build.

Steps:

Your github repository must contain two files ., one is dev.html and next file is – jenkinsfile



Jenkinsfile code:

```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                // Checkout the repository
                git branch: 'main', credentialsId: 'GitHubCreds', url: 'https://github.com/NAGENDRASAICH/htmlfilerepo.git'
```

```
            }
        }

        stage('Build and Test') {
            steps {
                // Here you can perform any build and test operations
                // For simplicity, let's just print a success message
                echo 'Build and Test successful!'
            }
        }

        stage('Generate Artifact') {
            steps {
                // Create the artifacts directory if it doesn't exist
                sh 'mkdir -p artifacts'
                // Copy the dev.html file to the artifacts directory
                sh 'cp dev.html ./artifacts/'
            }
        }
    }

    post {
        success {
            // Archive the generated artifact
            archiveArtifacts artifacts: 'artifacts/*.html', onlyIfSuccessful: true
            // Print a success message if the pipeline completes successfully
            echo 'Pipeline completed successfully!'
        }
    }
}
```

Dev.html code:

```
<html>
    <head>
        <title>registration form</title>
    </head>
    <body>
    <h6>
    <label="name">Name:</label>
    <input type="text" name="name"><br><br>
   <label="age">Age:</label>
   <input type="text" name="Age"><br><br>
   <label="DOB">DOB:</label>
   <input type="text" name="DOB"><br><br>
  <label="mobile no">Mobile No:</label>
  <input type="text" name="mobile no"><br><br>
  <label="gmail">Gmail:</label>
```

```
    <input type="text" name="Gmail"><br><br>
    <input type="submit" value="submit"><br><br>
    </h6>
    </body>
</html>
```

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

my project

» Required field

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 folders.

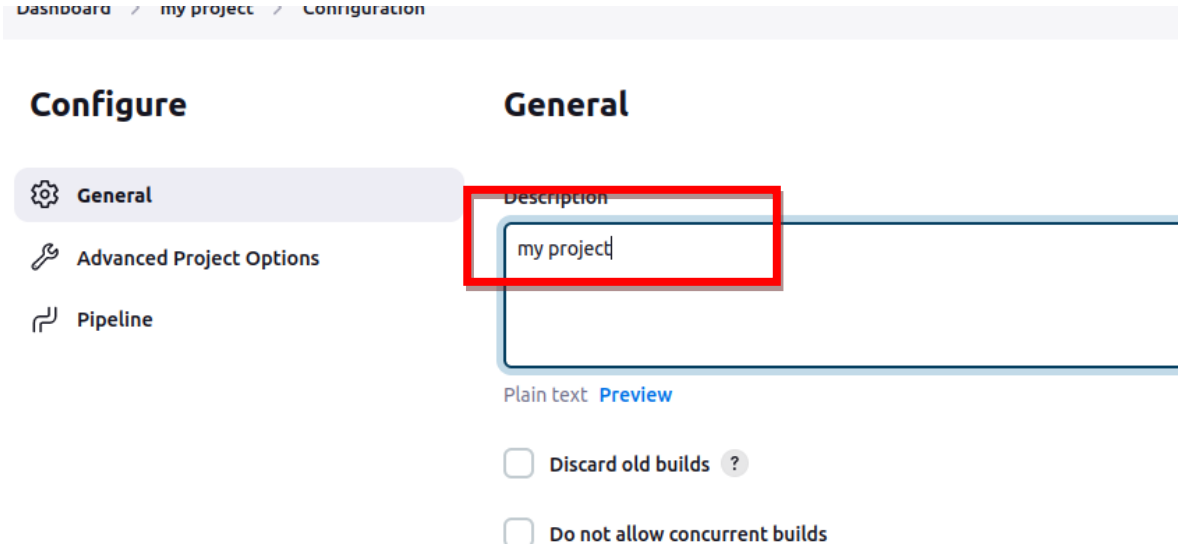
Multibranch Pipeline

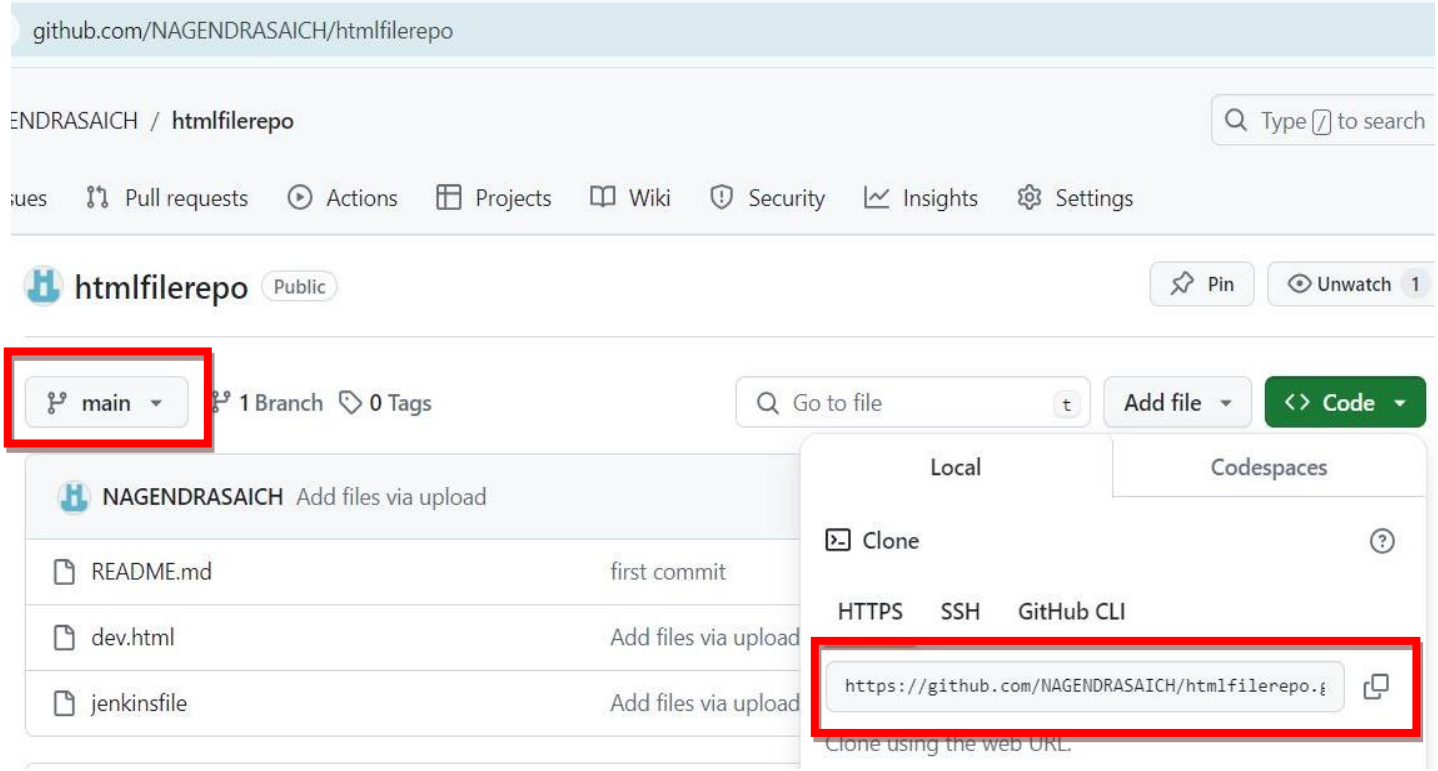
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

OK





As you see these two files are stored in **main branch**

Now, Copy this URL and paste it in jenkins

igure

eral

anced Project Options

eline

Advanced Project Options

Advanced

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/NAGENDRASAICH/htmlfilerepo.git

Credentials ?

- none -

+ Add

Advanced

Add Repository

Save Apply

localhost:8080/job/my project/configure

Dashboard > my project > Configuration

Configure

General

Advanced Project Options

Pipeline

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

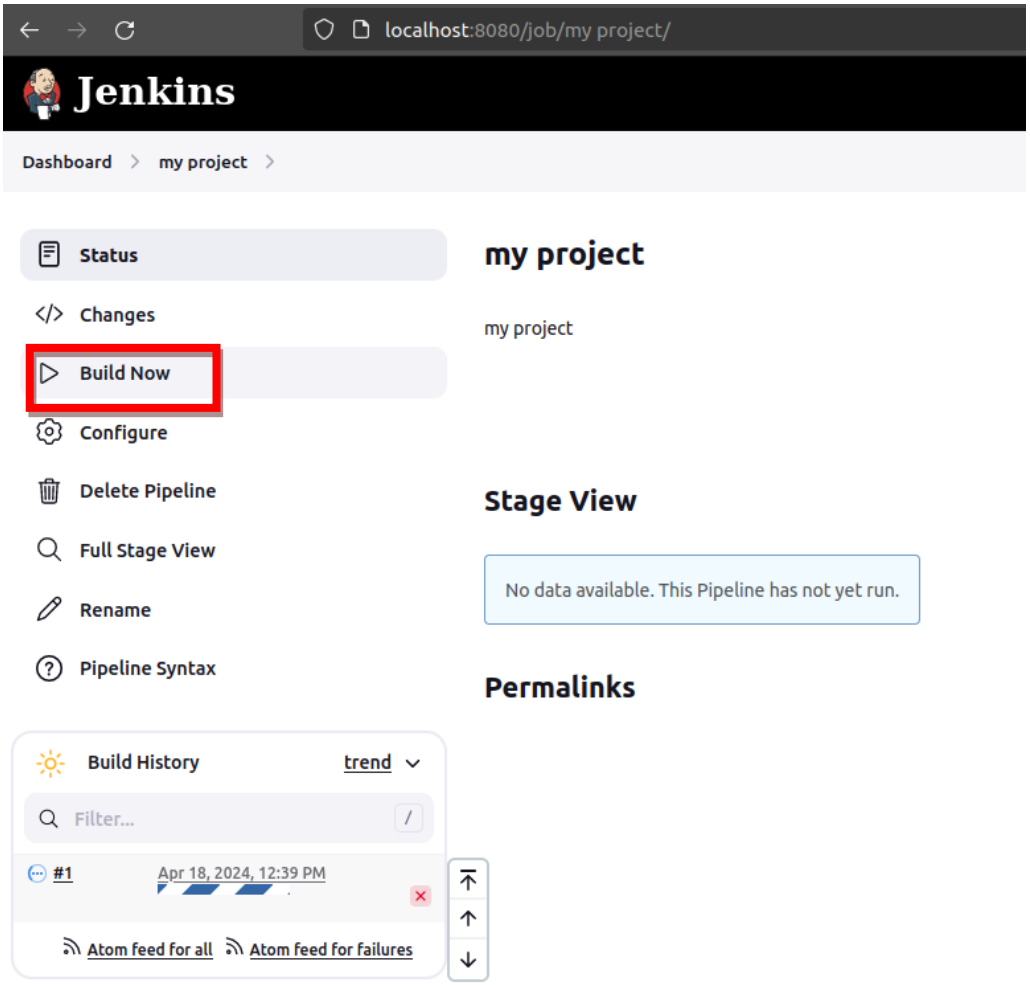
Script Path ?

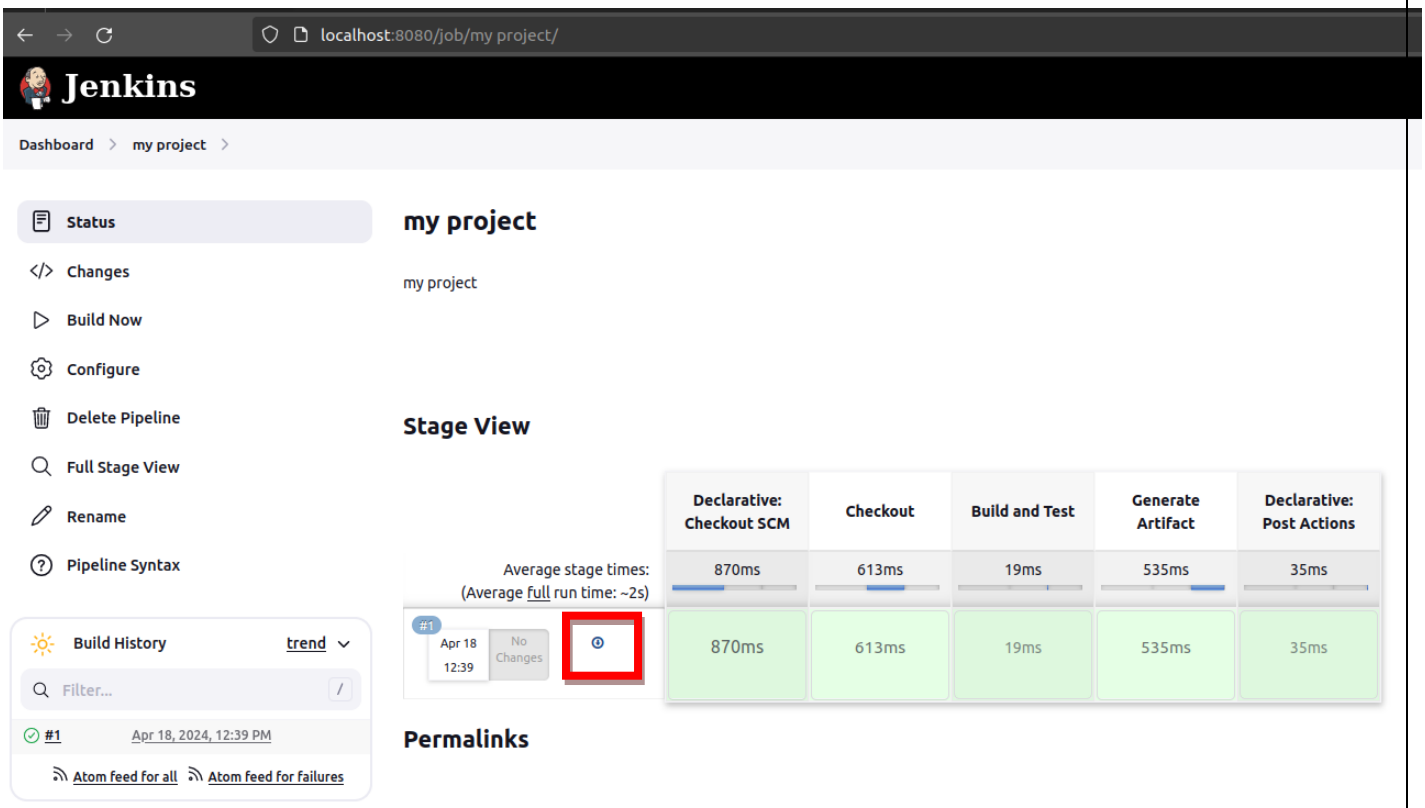
jenkinsfile

Lightweight checkout ?

Pipeline Syntax

Save Apply





After completion of all stages, artefact will be generated and , by clicking on download button you can download an artefact

Program Outcomes:

1. **Automated Build Process**: Successfully automate the build process of the web application using Jenkins.
2. **Automated Testing:** Implement automated tests to verify the functionality and quality of the code.
3. **Artifact Generation:** Generate a .rar artifact containing the dev.html file as per the pipeline configuration.

Experiment-6:

Explore Docker commands for content management

Program Objectives:

Familiarization with Docker Commands: Understand the basic Docker commands used for managing containers and images.

Practical Experience: Gain hands-on experience by executing Docker commands to manage containers and images.

Content Management: Learn how to handle Docker containers and images effectively to manage content within Dockerized environments.

Program Description:

The program aims to introduce learners to fundamental Docker commands necessary for content management within Docker containers. It covers essential commands such as **docker run, docker ps, docker stop, docker rm, docker images, docker pull, docker rmi, docker-compose up, docker-compose down, docker exec, and docker inspect.**

Participants will learn how to:

Start and stop Docker containers.
Remove containers and images when they are no longer needed.
List existing containers and images.
Pull Docker images from registries.
Use docker-compose for managing multi-container applications.
Execute commands inside running containers.
Inspect detailed information about containers and images.

Commands:

1. **docker run**: Start a new container from a Docker image.

   Example: **docker run hello-world**

2. **docker ps**: List the containers that are currently running.

   Example: **docker ps**

3. **docker stop**: Stop a running container.

   Example: **docker stop <container_id>**

4. **docker rm**: Remove a container.

   Example: **docker rm <container_id>**

5. **docker images**: List all Docker images downloaded or created on your system.

   Example: **docker images**

6. **docker pull**: Download a Docker image from a registry.

   Example: **docker pull nginx:latest**

7. **docker rmi**: Remove a Docker image.

   Example: **docker rmi <image_id>**

8. **docker-compose up**: Start services defined in a docker-compose.yml file.

   Example: **docker-compose up**

9. **docker-compose down**: Stop and remove containers, networks, and volumes defined in a docker-compose.yml file.

   Example: **docker-compose down**

10. **docker exec**: Run a command inside a running container.

    Example: **docker exec -it <container_id> bash**

11. **docker images**: List all Docker images stored locally.

    Example: **docker images**

12. **docker start**: Start a stopped container.

    Example: **docker start <container_name or container_id>**

13. **docker stop**: Stop a running container.

    Example: **docker stop <container_name or container_id>**

14. **docker rm**: Remove a container.

    Example: **docker rm <container_name or container_id>**

15. **docker rmi**: Remove a Docker image.

    Example: **docker rmi <image_name or image_id>**

16. **docker inspect**: Display detailed information on one or more containers or images.

    Example: **docker inspect <container_name or image_name>**

Command to uninstall docker:

sudo systemctl stop docker && sudo apt purge -y docker-ce docker-ce-cli containerd.io && sudo rm -rf /etc/docker /var/lib/docker && sudo groupdel docker && sudo rm /etc/apt/sources.list.d/docker.list && sudo apt update && sudo apt autoremove –y

Command to install docker:

sudo apt install docker.io

Experiment-7:

Develop a simple containerized application using Docker

Program Objectives:

1. **Introduction to Containerization:** Understand the concept of containerization and its benefits in application development and deployment.
2. **Familiarization with Docker:** Gain familiarity with Docker, a popular containerization platform, and its basic usage.
3. **Creating Docker Images:** Learn how to create Docker images containing application files and dependencies.
4. **Running Docker Containers:** Understand the process of running Docker containers from created images.
5. **Basic Application Deployment:** Gain hands-on experience in deploying a simple application within a Docker container.

Program Description:

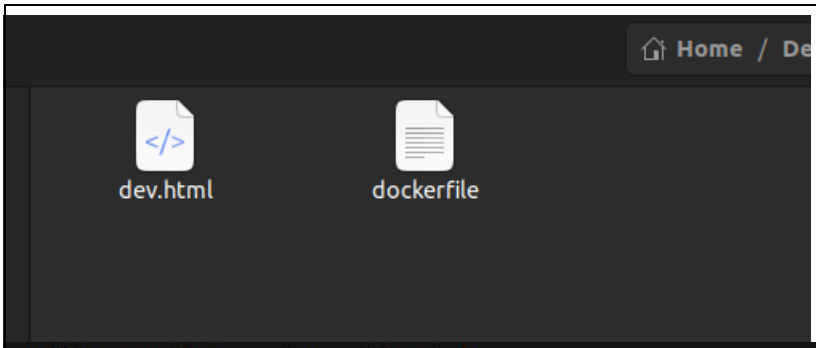
In this program, you will learn how to containerize a simple application using Docker. The application consists of a single HTML file (1.html). You will follow these steps:

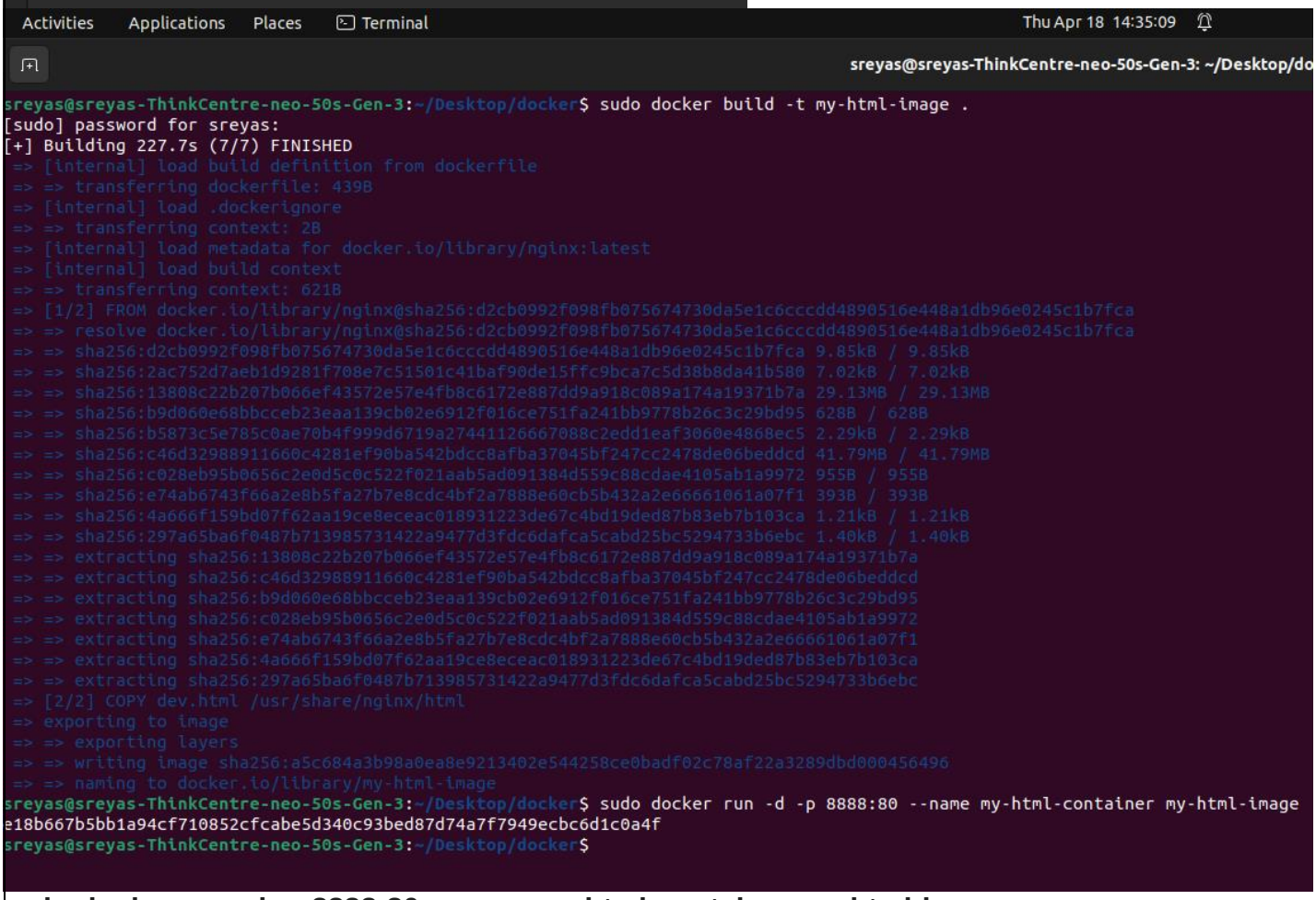
1. **Create HTML File:** Begin by creating a simple HTML file (1.html) with some content. This file will serve as the content of your containerized application.
2. **Write Dockerfile:** Create a Dockerfile, a text file that contains instructions for building a Docker image. In the Dockerfile, specify the base image, copy the HTML file into the image, and define any necessary configurations.
3. **Build Docker Image:** Use the Docker CLI to build a Docker image based on the Dockerfile you created. This process involves executing the docker build command with appropriate parameters.
6. **Run Docker Container:** Once the Docker image is built successfully, use the docker run command to instantiate a Docker container from the created image. Specify any necessary options such as port mappings or volume mounts.
7. **Access Application:** Access the running application by opening a web browser and navigating to the appropriate URL, typically http://localhost:<port> if port mapping was configured.

Steps:

```
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3:~/Desktop/docker$ sudo service docker start
[sudo] password for sreyas:
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3:~/Desktop/docker$ sudo systemctl start docker
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3:~/Desktop/docker$ nano Dockerfile
```

```
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3: ~/Desktop/docker
  GNU nano 6.2                                                    dockerfile *
# Use a base image, for example, Nginx-Nginx is a web server software that serves web pages
FROM nginx

# Copy your HTML file into the Docker container
COPY dev.html /usr/share/nginx/html

#  Port 80 is the default port for HTTP traffic,specifying EXPOSE 80 means that the container will listen for incoming network connections on port 80.
EXPOSE 80
#press ctrl+x and then pressy - to save this file
```





sudo docker run -d -p 8888:80 --name my-html-container my-html-image