

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>

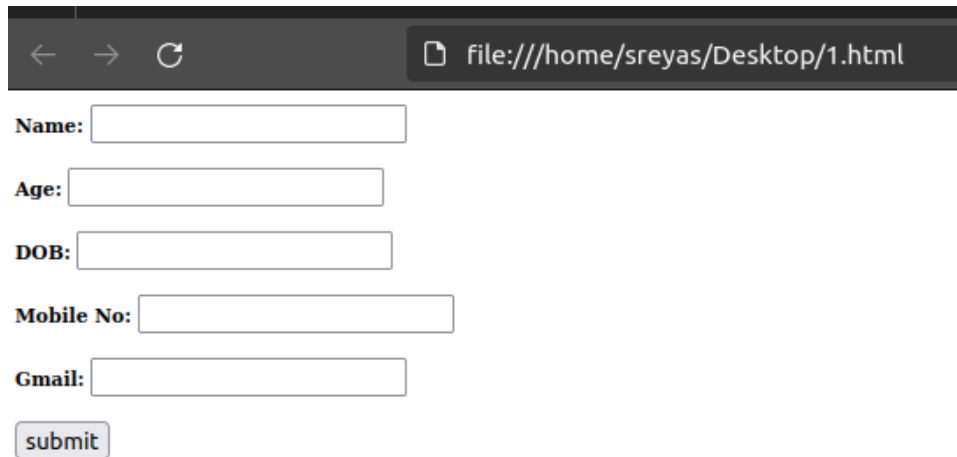
<label="gmail">Gmail:</label>
<input type="text" name="Gmail"><br><br>

<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

A screenshot of a web browser window. The address bar shows the file path: file:///home/sreyas/Desktop/1.html. The browser has navigation buttons (back, forward, refresh). Below the address bar is a registration form with the following fields: "Name:" with a text input, "Age:" with a text input, "DOB:" with a text input, "Mobile No:" with a text input, and "Gmail:" with a text input. At the bottom of the form is a "submit" button.

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 in git repository , so it will show 1.html in red colour, which indicates that our file 1.html is not transferred to git repository, 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)

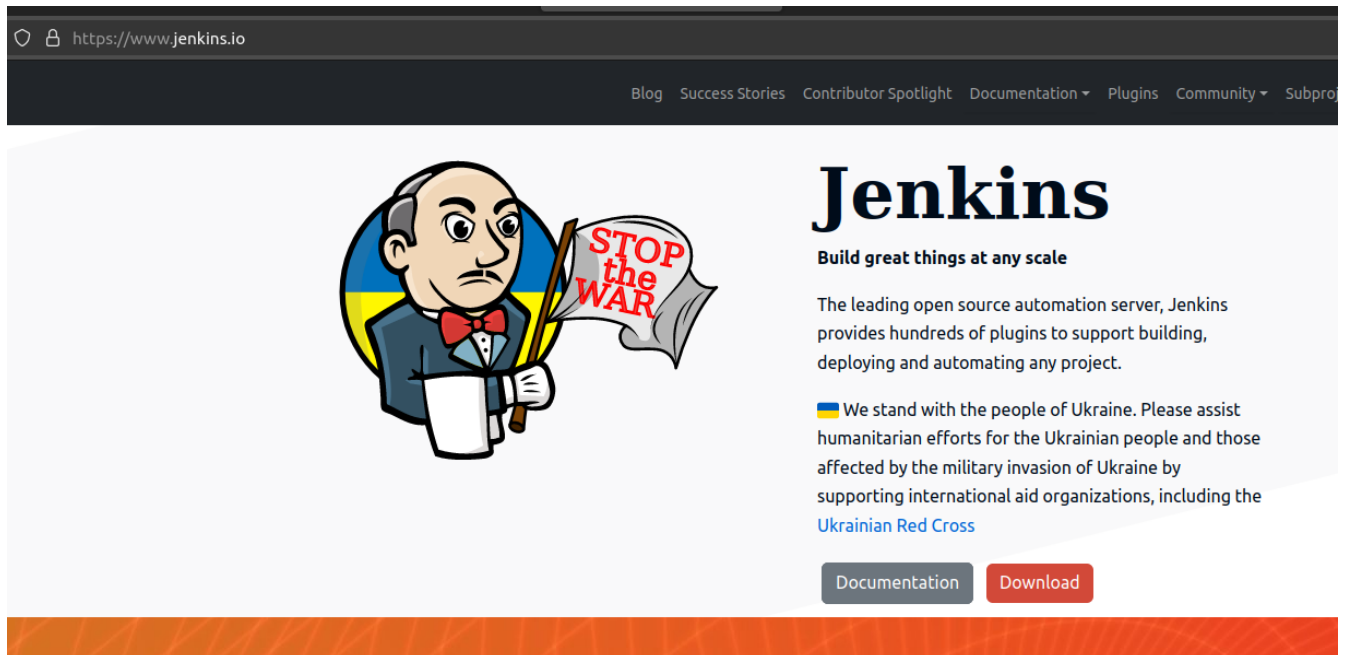
now you file is successfully transferred to git hub webiste fo git repository

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.

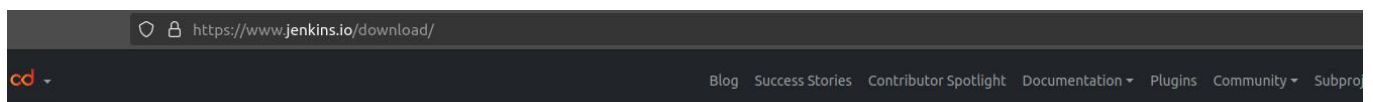
Experiment-4 : Jenkins installation and setup, explore the Environment

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



The screenshot shows the Jenkins homepage at <https://www.jenkins.io>. The header includes links for Blog, Success Stories, Contributor Spotlight, Documentation, Plugins, Community, and Subprojects. The main content features the Jenkins logo (a cartoon man with a bow tie) and a banner that reads "STOP the WAR". To the right, the text says "Jenkins Build great things at any scale" and describes it as the leading open source automation server. A statement of support for Ukraine is also present, along with buttons for "Documentation" and "Download".

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



The screenshot shows the Jenkins download page at <https://www.jenkins.io/download/>. The header is identical to the homepage. The main content area is titled "Jenkins download and deployment" and provides information about the two release lines: Stable (LTS) and Regular (Weekly).

Jenkins download and deployment

The Jenkins project produces two release lines: Stable (LTS) and regular (Weekly). Depending on your organization's needs, one may be preferred over the other. See the links below for recommendations about the release lines.

Stable (LTS)

Long-Term Support (LTS) release baselines are chosen every 12 weeks from the stream of regular releases. Every 4 weeks we release stable releases which include bug and security fix backports. [Learn more...](#)

[Changelog](#) | [Upgrade Guide](#) | [Past Releases](#)

Regular releases (Weekly)

This release line delivers bug fixes and new features rapidly to use them. It is generally delivered on a weekly cadence. [Learn more...](#)

[Changelog](#) | [Past Releases](#)

Downloading Jenkins

Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

1. Before downloading, please take a moment to review the [Hardware and Software requirements](#) section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section of the User Handbook.
4. You may also want to verify the package you downloaded. [Learn more about verifying Jenkins downloads.](#)

Download Jenkins 2.440.2 LTS for:

Generic Java package (.war)

SHA-256: 8126628e9e2f8ee2f807d489ec0a6e37fc9f5d5ba84fa8f3718e7f3e2a27312e 

Docker

Kubernetes

Ubuntu/Debian

Download Jenkins 2.453 for:

Generic Java package (.war)

SHA-256: a782f364fd2817427bc97911e8648a62cba9d9e267893c2c5e5136b43605d3ca 

Docker

Ubuntu/Debian

Red Hat/Fedora/Alma/Rocky/CentOS

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

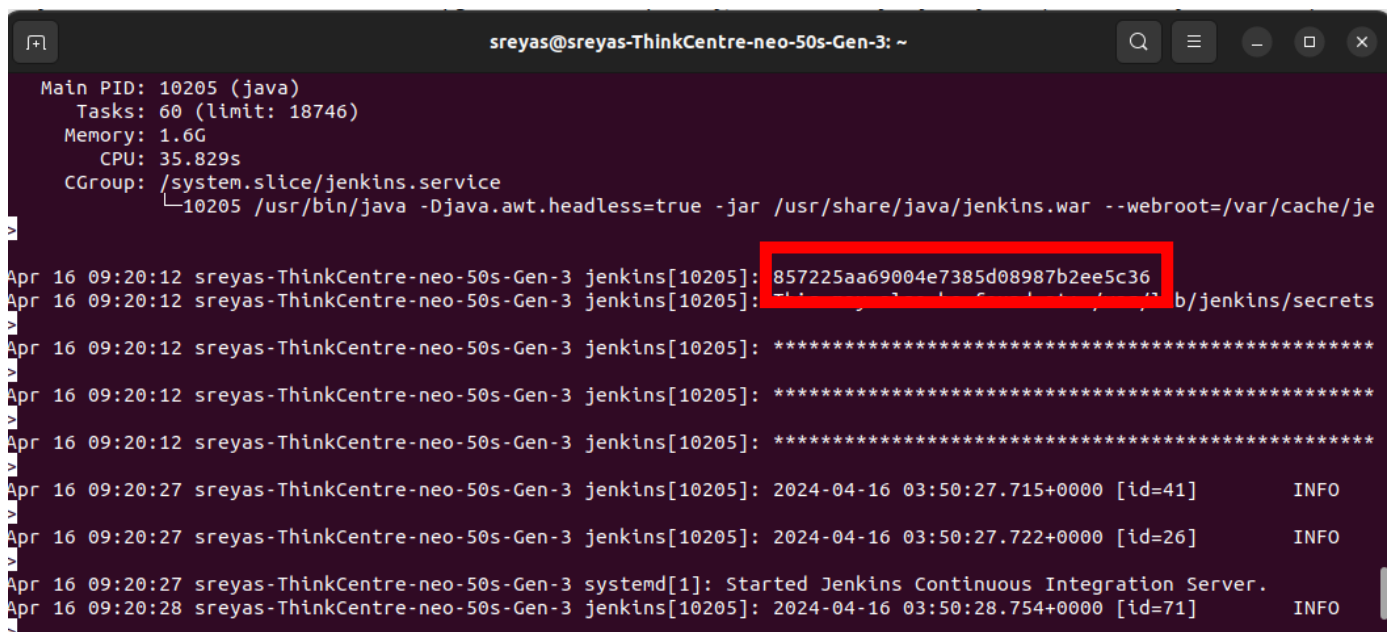
So, after running all the above commands , Jenkins will be installed in your system

To start jenkins not run below command:

sudo systemctl start Jenkins

To see Jenkins is active or not run below command:

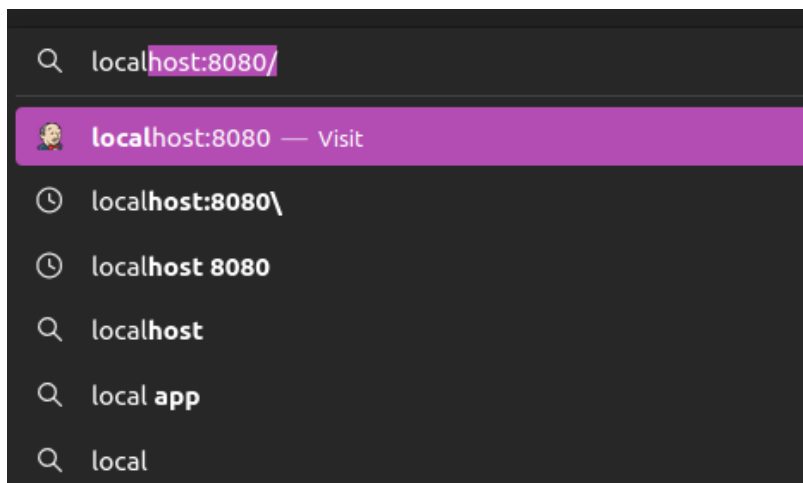
sudo systemctl status Jenkins

A terminal window titled 'sreyas@sreyas-ThinkCentre-neo-50s-Gen-3: ~' showing the status of the Jenkins service. The output of 'sudo systemctl status Jenkins' is displayed, showing the main PID as 10205 (java) and the tasks as 60 (limit: 18746). The memory usage is 1.6G and CPU usage is 35.829s. The CGroup is /system.slice/jenkins.service. The logs show the Jenkins service starting successfully on April 16, 2024, at 09:20:12. The password for Jenkins is highlighted in a red box: 857225aa69004e7385d08987b2ee5c36. The logs also show the Jenkins service starting successfully on April 16, 2024, at 09:20:27. The logs show the Jenkins service starting successfully on April 16, 2024, at 09:20:28. The logs show the Jenkins service starting successfully on April 16, 2024, at 09:20:28.

```
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3: ~
Main PID: 10205 (java)
Tasks: 60 (limit: 18746)
Memory: 1.6G
CPU: 35.829s
CGroup: /system.slice/jenkins.service
└─10205 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Apr 16 09:20:12 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: 857225aa69004e7385d08987b2ee5c36
Apr 16 09:20:12 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Apr 16 09:20:12 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: *****
Apr 16 09:20:12 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: *****
Apr 16 09:20:12 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: *****
Apr 16 09:20:27 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: 2024-04-16 03:50:27.715+0000 [id=41] INFO
Apr 16 09:20:27 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: 2024-04-16 03:50:27.722+0000 [id=26] INFO
Apr 16 09:20:27 sreyas-ThinkCentre-neo-50s-Gen-3 systemd[1]: Started Jenkins Continuous Integration Server.
Apr 16 09:20:28 sreyas-ThinkCentre-neo-50s-Gen-3 jenkins[10205]: 2024-04-16 03:50:28.754+0000 [id=71] INFO
```

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



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

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.

Select – install suggested plugins

Getting Started

Getting Started

<input checked="" type="checkbox"/> Folders	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input type="checkbox"/> Build Timeout	<input type="checkbox"/> Credentials Binding	<div>** Ionicons API Folders OWASP Markup Formatter</div>
<input type="checkbox"/> Timestampers	<input type="checkbox"/> Workspace Cleanup	<input type="checkbox"/> Ant	<input type="checkbox"/> Gradle	
<input type="checkbox"/> Pipeline	<input type="checkbox"/> GitHub Branch Source	<input type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="checkbox"/> Pipeline: Stage View	
<input type="checkbox"/> Git	<input type="checkbox"/> SSH Build Agents	<input type="checkbox"/> Matrix Authorization Strategy	<input type="checkbox"/> PAM Authentication	
<input type="checkbox"/> LDAP	<input type="checkbox"/> Email Extension	<input type="checkbox"/> Mailer	<input type="checkbox"/> Dark Theme	

Create First Admin User

Username

admin

Password

Confirm password

Full name

admin

E-mail address

aiml3b2023@gmail.com|

aiml3b2023@gmail.com

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

Instance Configuration

Jenkins URL:

http://localhost:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

Jenkins 2.440.2

This is your Jenkins home page

The screenshot shows the Jenkins web interface in a browser. The address bar shows 'localhost:8080'. The page has a dark header with the Jenkins logo, a search bar, and user information 'admin'. The main content area is divided into a left sidebar and a main panel. The sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two 'Idle' executors). The main panel has a 'Welcome to Jenkins!' message, a brief description of the page's purpose, and a 'Start building your software project' section with buttons for 'Create a job', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'.

If you want to uninstall Jenkins , you can use below commands:

```
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

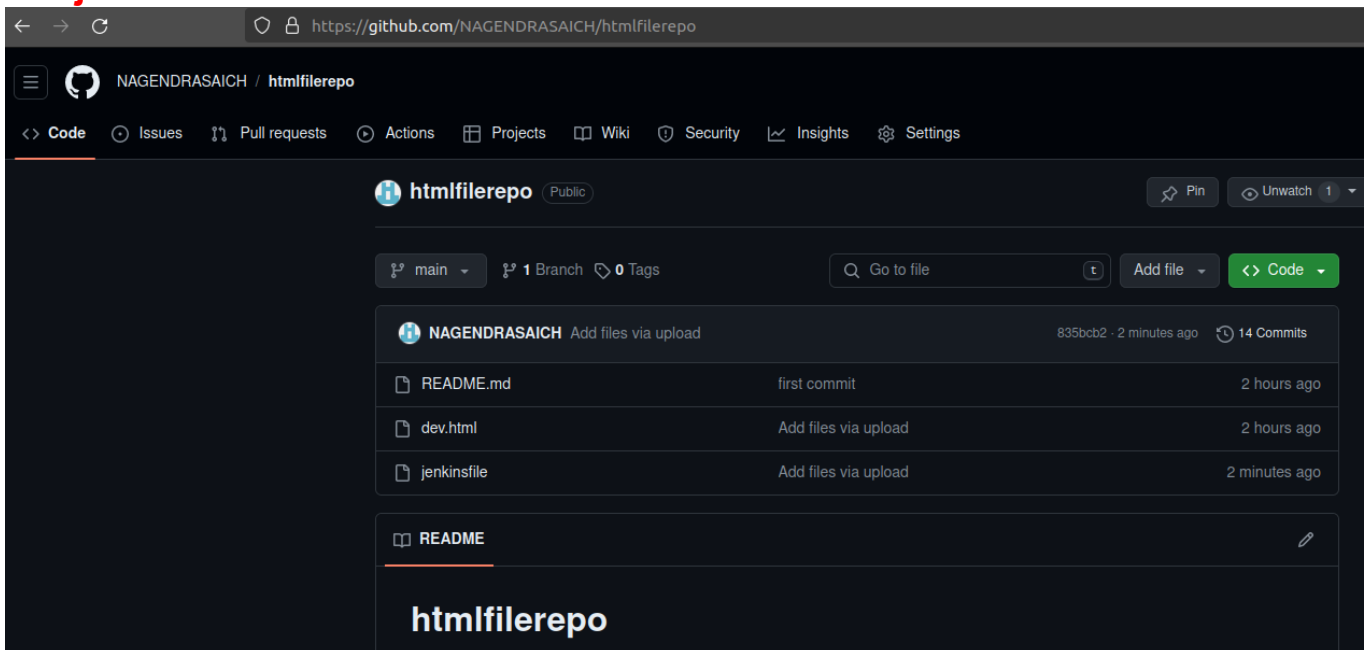
Objective: The objective of this experiment is to demonstrate the implementation of continuous integration and development using Jenkins. Specifically, the goal is to create a Jenkins pipeline that automates the build, test, and artifact generation process for a sample web application hosted on GitHub.

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

```

```
<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

Dashboard / my project / Configuration

Configure

General

General

Advanced Project Options

Pipeline

Description

my project

Plain text [Preview](#)

☐ Discard old builds ?

☐ Do not allow concurrent builds

github.com/NAGENDRASAICH/htmlfilerepo

NAGENDRASAICH / htmlfilerepo

Type to search

Issues Pull requests Actions Projects Wiki Security Insights Settings

htmlfilerepo Public

Pin Unwatch 1

main 1 Branch 0 Tags

Go to file t Add file <> Code

Local Codespaces

Clone ?

HTTPS SSH GitHub CLI

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

Clone using the web URL.

NAGENDRASAICH Add files via upload

README.md	first commit
dev.html	Add files via upload
jenkinsfile	Add files via upload

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

Now, Copy this URL and paste it in jenkins

Figure

Advanced Project Options

Advanced ▾

General

Advanced Project Options

Pipeline

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

← → ↺

localhost:8080/job/my project/

 **Jenkins**

Dashboard > my project >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

✎ Rename

❓ Pipeline Syntax

my project

my project

Stage View

No data available. This Pipeline has not yet run.

Permalinks

☀️ Build History

trend ▾

🔍 Filter... /

#1

Apr 18, 2024, 12:39 PM

⌵

📡 Atom feed for all

📡 Atom feed for failures


⬆

⬆

⬆

← → ↺

localhost:8080/job/my project/

 **Jenkins**

Dashboard > my project >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

✎ Rename

❓ Pipeline Syntax

my project

my project

Stage View

Average stage times:
(Average full run time: ~2s)

	Declarative: Checkout SCM	Checkout	Build and Test	Generate Artifact	Declarative: Post Actions
#1	870ms	613ms	19ms	535ms	35ms

Permalinks

☀️ Build History

trend ▾

🔍 Filter... /

#1

Apr 18, 2024, 12:39 PM

⌵

📡 Atom feed for all

📡 Atom feed for failures

⬆

⬆

⬆

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
```

```
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
```

```
Home / De

dev.html  dockerfile

Activities Applications Places Terminal Thu Apr 18 14:35:09
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3: ~/Desktop/do

sreyas@sreyas-ThinkCentre-neo-50s-Gen-3:~/Desktop/docker$ sudo docker build -t my-html-image .
[sudo] password for sreyas:
[+] Building 227.7s (7/7) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 439B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load build context
=> => transferring context: 621B
=> [1/2] FROM docker.io/library/nginx@sha256:d2cb0992f098fb075674730da5e1c6ccdd4890516e448a1db96e0245c1b7fca
=> => resolve docker.io/library/nginx@sha256:d2cb0992f098fb075674730da5e1c6ccdd4890516e448a1db96e0245c1b7fca
=> => sha256:d2cb0992f098fb075674730da5e1c6ccdd4890516e448a1db96e0245c1b7fca 9.85kB / 9.85kB
=> => sha256:2ac752d7aeb1d9281f708e7c51501c41baf90de15ffc9bca7c5d38b8da41b580 7.02kB / 7.02kB
=> => sha256:13808c22b207b066ef43572e57e4fb8c6172e887dd9a918c089a174a19371b7a 29.13MB / 29.13MB
=> => sha256:b9d060e68bbccce23eaa139cb02e6912f016ce751fa241bb9778b26c3c29bd95 628B / 628B
=> => sha256:b5873c5e785c0ae70b4f999d6719a27441126667088c2edd1eaf3060e4868ec5 2.29kB / 2.29kB
=> => sha256:c46d32988911660c4281ef90ba542bdcc8afba37045bf247cc2478de06beddcd 41.79MB / 41.79MB
=> => sha256:c028eb95b0656c2e0d5c0c522f021aab5ad091384d559c88cdae4105ab1a9972 955B / 955B
=> => sha256:e74ab6743f66a2e8b5fa27b7e8cdc4bf2a7888e60cb5b432a2e66661061a07f1 393B / 393B
=> => sha256:4a666f159bd07f62aa19ce8eace018931223de67c4bd19ded87b83eb7b103ca 1.21kB / 1.21kB
=> => sha256:297a65ba6f0487b713985731422a9477d3fdc6dafca5cabd25bc5294733b6ebc 1.40kB / 1.40kB
=> => extracting sha256:13808c22b207b066ef43572e57e4fb8c6172e887dd9a918c089a174a19371b7a
=> => extracting sha256:c46d32988911660c4281ef90ba542bdcc8afba37045bf247cc2478de06beddcd
=> => extracting sha256:b9d060e68bbccce23eaa139cb02e6912f016ce751fa241bb9778b26c3c29bd95
=> => extracting sha256:c028eb95b0656c2e0d5c0c522f021aab5ad091384d559c88cdae4105ab1a9972
=> => extracting sha256:e74ab6743f66a2e8b5fa27b7e8cdc4bf2a7888e60cb5b432a2e66661061a07f1
=> => extracting sha256:4a666f159bd07f62aa19ce8eace018931223de67c4bd19ded87b83eb7b103ca
=> => extracting sha256:297a65ba6f0487b713985731422a9477d3fdc6dafca5cabd25bc5294733b6ebc
=> [2/2] COPY dev.html /usr/share/nginx/html
=> => exporting to image
=> => exporting layers
=> => writing image sha256:a5c684a3b98a0ea8e9213402e544258ce0badf02c78af22a3289dbd000456496
=> => naming to docker.io/library/my-html-image
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3:~/Desktop/docker$ sudo docker run -d -p 8888:80 --name my-html-container my-html-image
e18b667b5bb1a94cf710852cfcabe5d340c93bed87d74a7f7949ecbc6d1c0a4f
sreyas@sreyas-ThinkCentre-neo-50s-Gen-3:~/Desktop/docker$
```

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

Activities Applications Places Firefox Web Browser

registration form × +

← → ↺ localhost:8888/dev.html

Name:

Age:

DOB:

Mobile No:

Gmail: