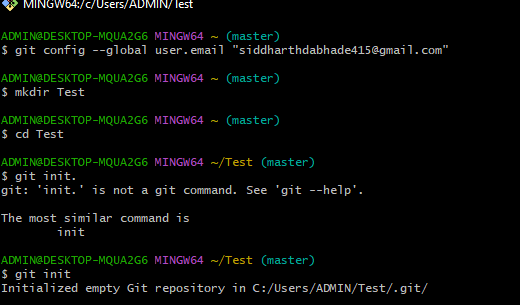
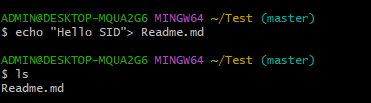
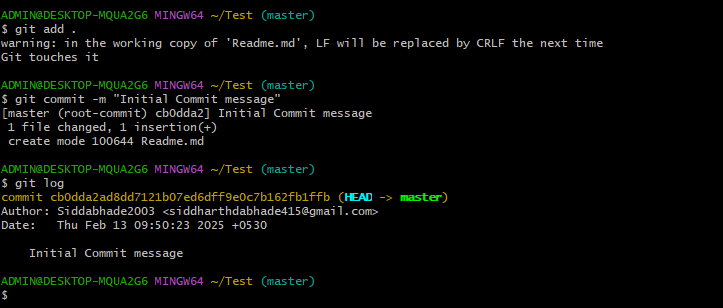
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Name of the Practical** | | **CO** | **Signature** |
|  | | **Devops Practical Index** | | |
| 1. | Basic Git commands | | CO1 |  |
| 2. | Create and fork repositories in GitHub. Apply branch, merge, rebase concepts. | | CO2 |
| 3. | Using Git for Collaboration | | CO2 |
| 4. | Collaborating and Cloning using GitHub | | CO2 |
| 5. | Using GitLab Web IDE | | CO2 |
| 6. | Performing merge requests using GitLab | | CO2 |
| 7. | Workflow management in GitLab | | CO2 |
| 8. | Demonstrate Continuous Integration and development using Jenkins | | CO3 |
| 9. | Explore docker commands for content management | | CO3 |
| 10. | Develop a simple containerized application using Docker | | CO3 |
| 11. | Ad-hoc Ansible commands | | CO4 |  |
| 12 | Using Ansible playbooks | | CO4 |  |

PRACTCIAL NO 1

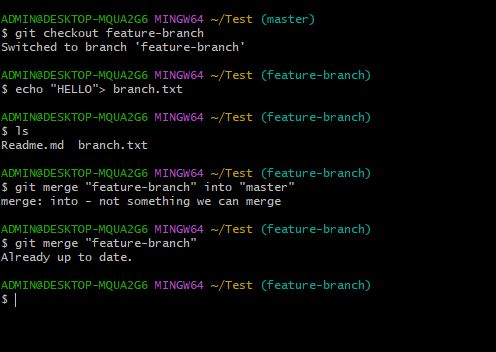
AIM: Basic Git commands

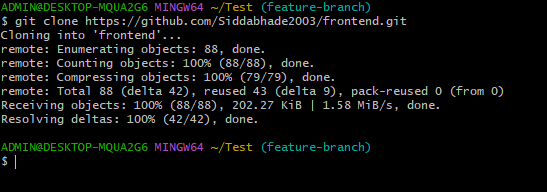






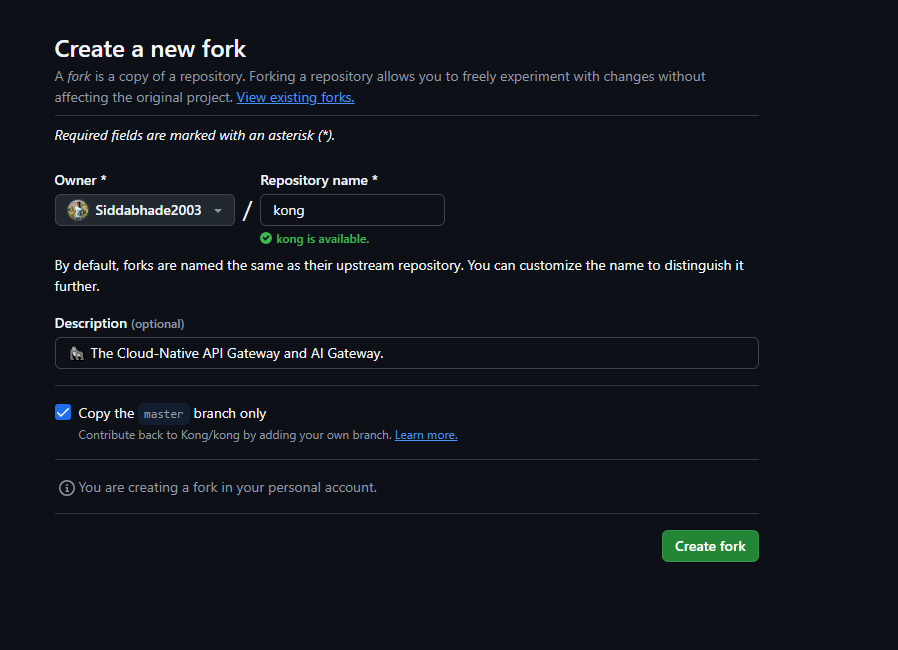


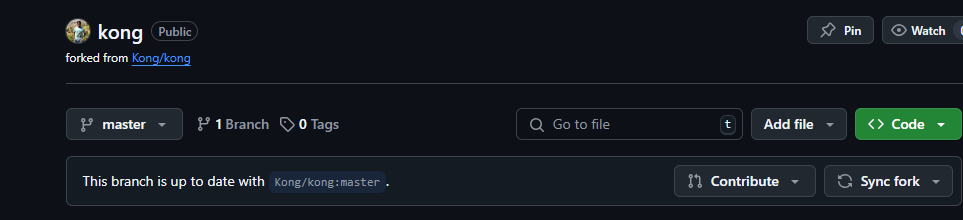


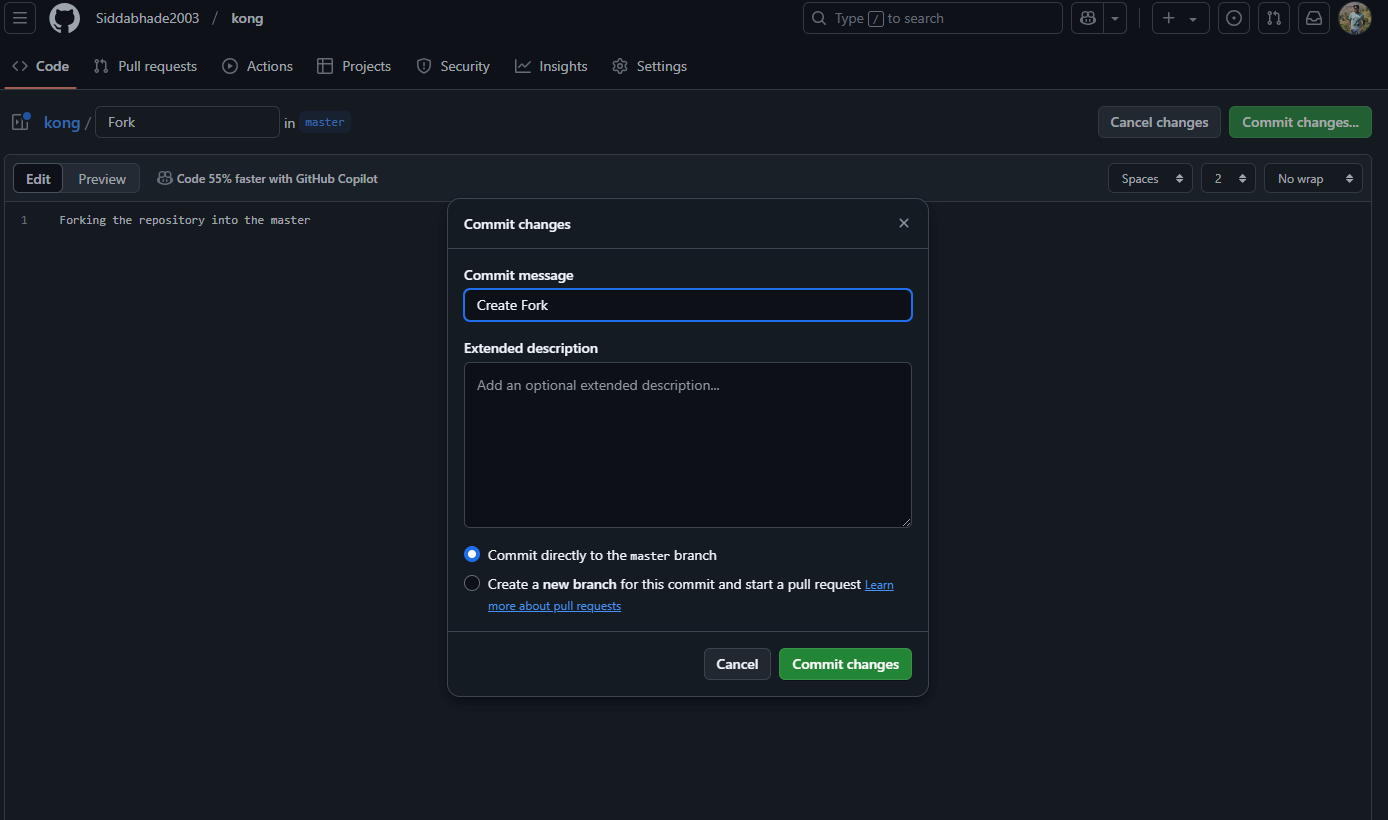


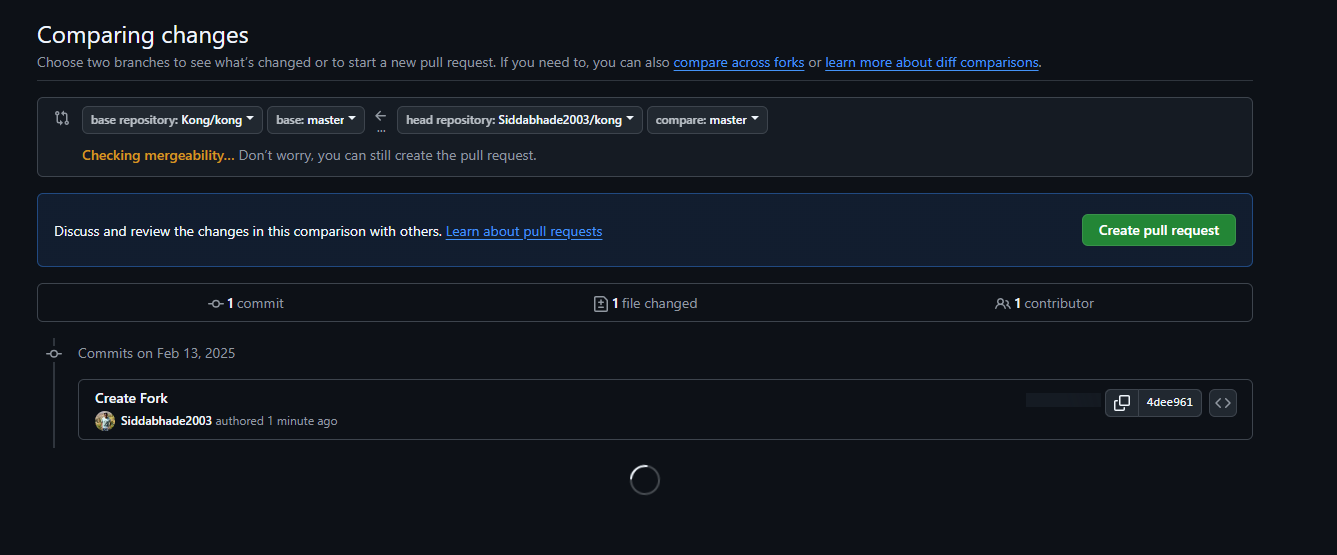
PRACTICAL NO 2

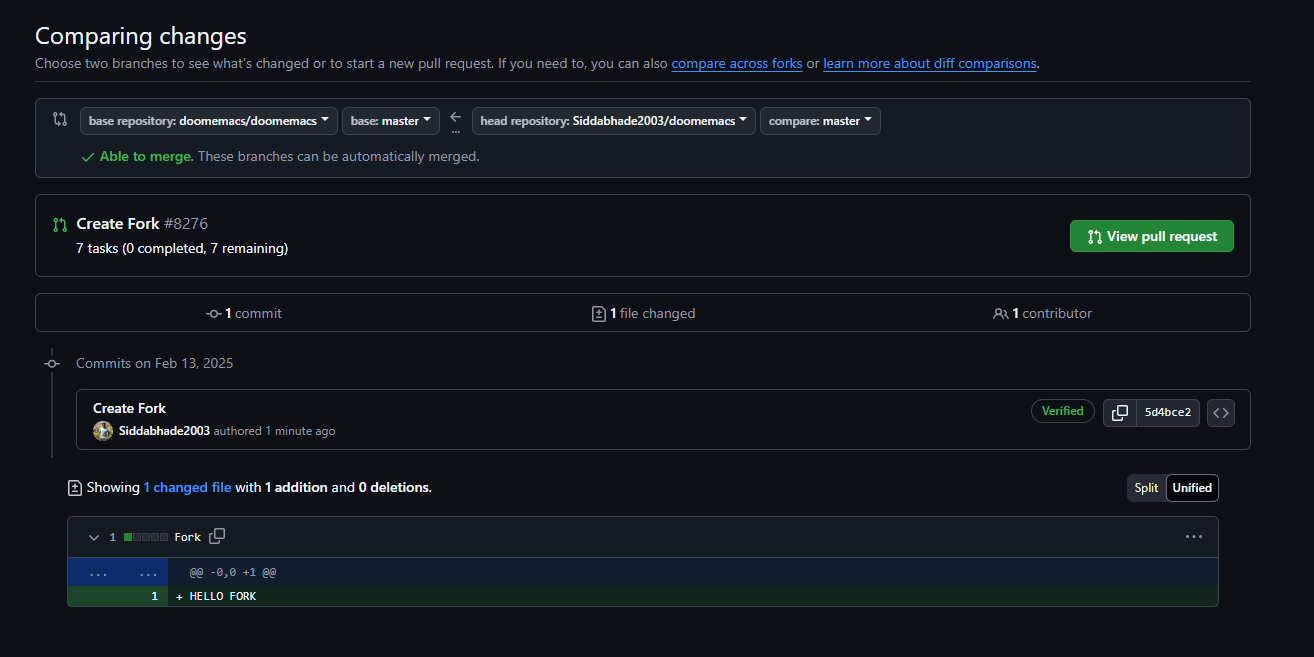
AIM: Create and fork repositories in GitHub. Apply branch, merge, rebase concepts.

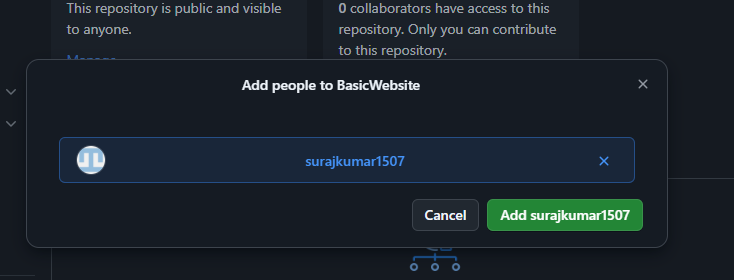






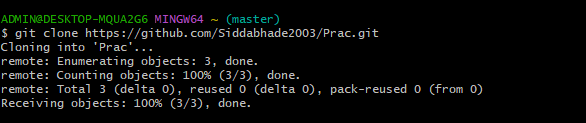


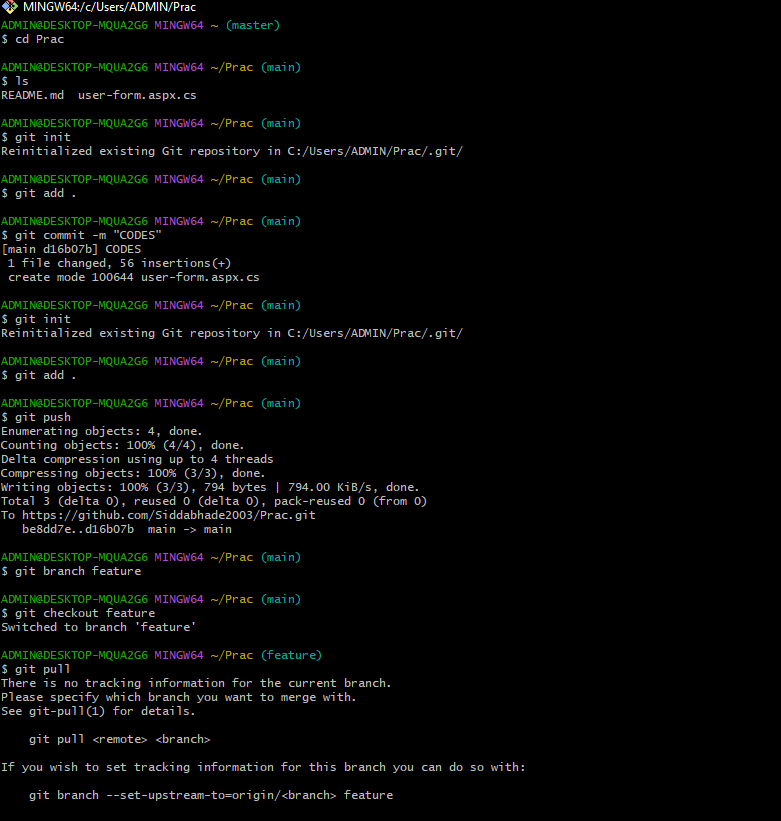


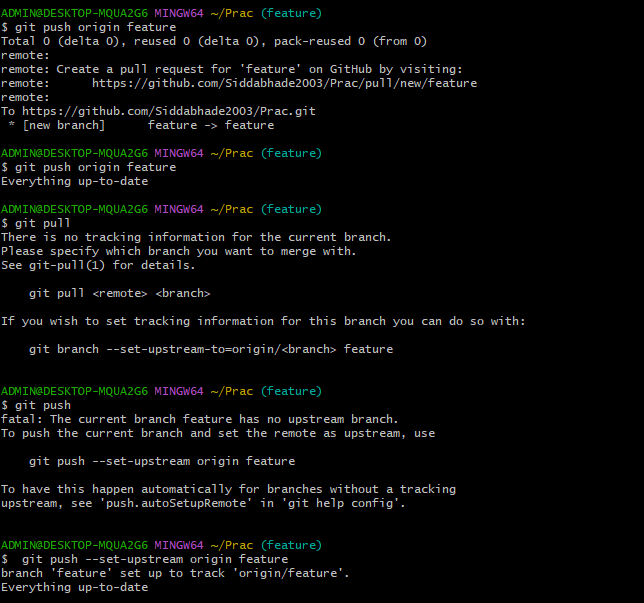


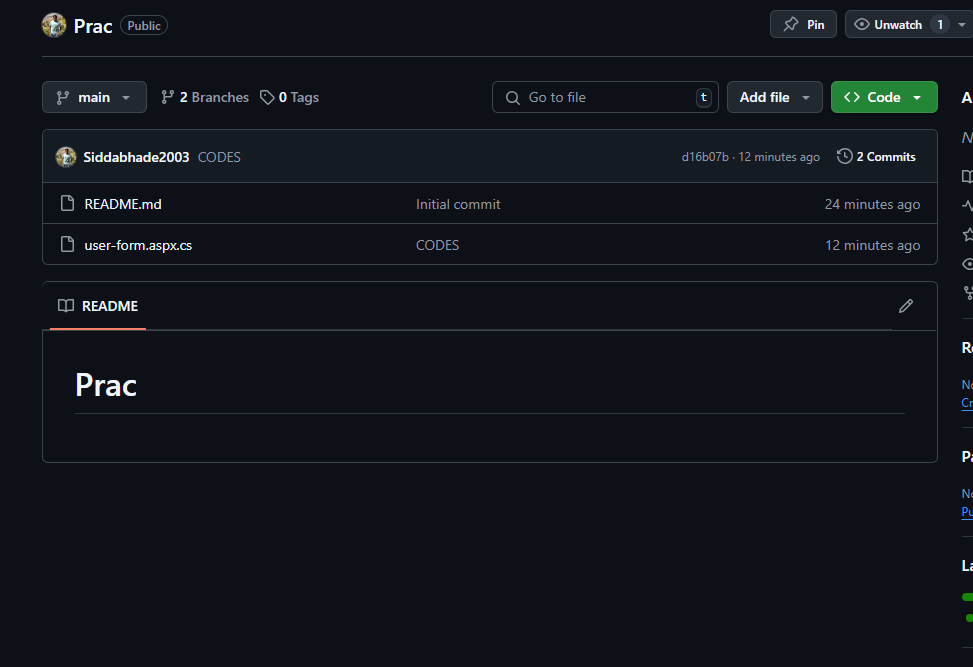
PRACTICAL NO 4

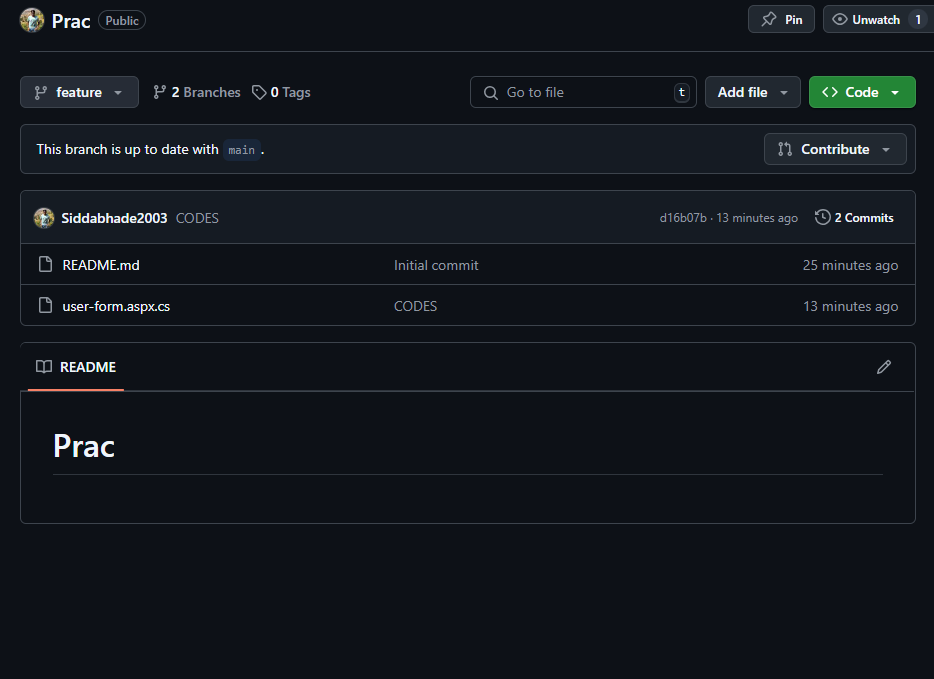
AIM: Collaborating and Cloning using GitHub











PRACTICAL NO 5

AIM: Using GitLab Web IDE

**🔹 What is GitLab Web IDE?**

GitLab Web IDE is an **online code editor** where you can **edit, create, and commit files** directly in your GitLab repository **without installing anything** on your computer.

**Step 1: Open GitLab and Your Repository**

1. **Go to GitLab** → <https://gitlab.com/>
2. **Login** to your GitLab account.
3. Click on the **"Projects"** tab.
4. Select the **repository (project)** where you want to make changes.

**Step 2: Open Web IDE**

1. Inside your repository, **look for the "Web IDE" button** (top-right of the page).
2. **Click on "Web IDE"** → It will open an online code editor inside your browser.

**Step 3: Edit an Existing File**

1. From the **left panel**, click on a file (e.g., index.html).
2. **Make some changes** in the file.  
   Example (if it’s an HTML file, edit like this):
3. <h1>Hello from GitLab Web IDE!</h1>
4. **Click on the checkmark ✅ (Save) icon** on the top-left to save the changes.

**Step 4: Create a New File (Optional)**

1. Click on the **“+” button** in the left panel.
2. Select **“New file”**.
3. Type the **file name** (e.g., style.css).
4. Add content to the file:
5. body {
6. background-color: lightblue;
7. }
8. **Click Save ✅**.

**Step 5: Commit (Save) Your Changes**

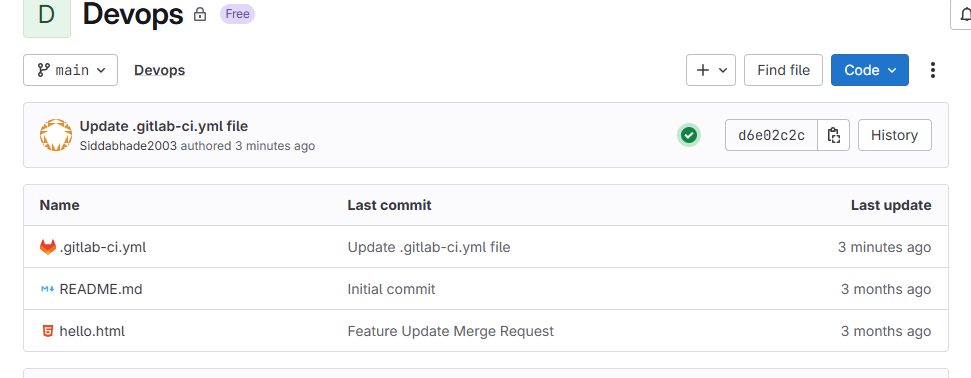
1. Click the **"Commit..." button** on the left side.
2. **Type a commit message** (Example: "Updated HTML and added CSS").
3. Click **"Commit"**.
4. **Your changes are now saved in the repository! 🎉**

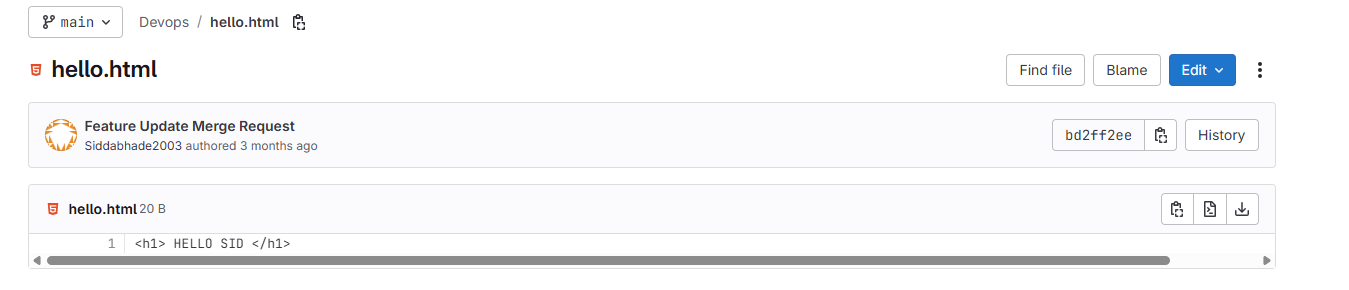
**Step 6: Verify Your Changes**

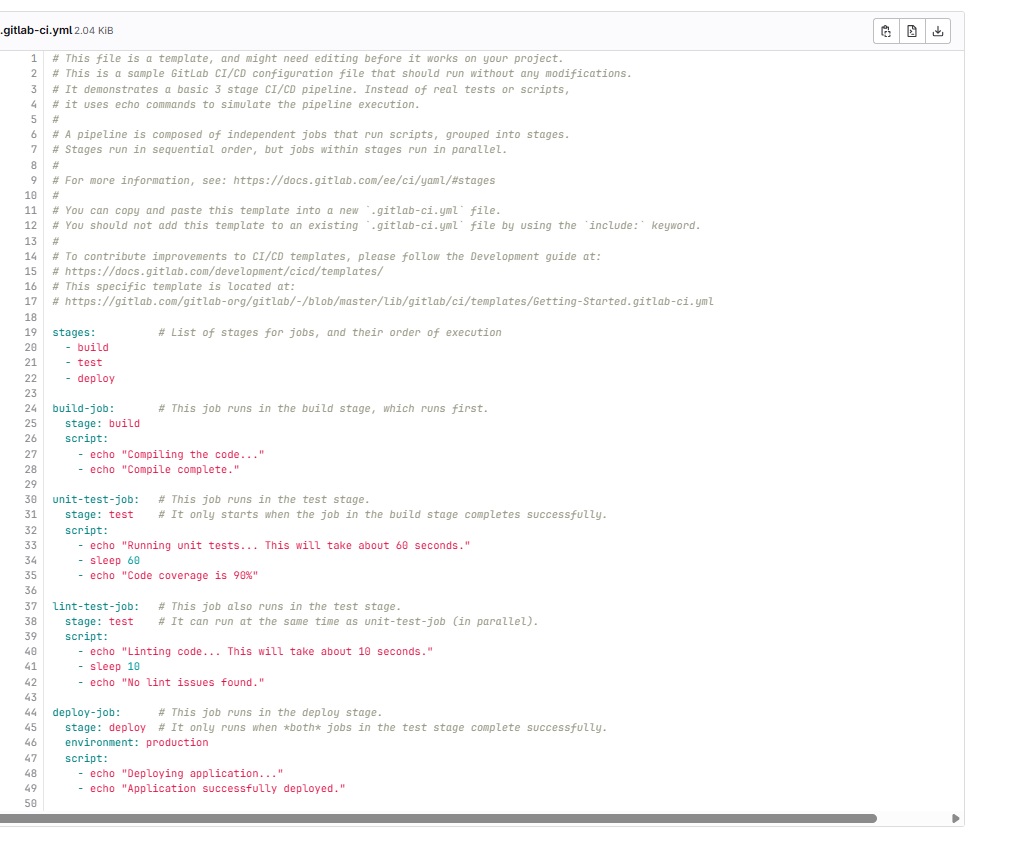
1. Go back to your **repository home page**.
2. Open the file you modified and check if the changes are saved.

**Step 7: Push Changes (If Required)**

If you are working on a branch, you might need to **push changes to the main repository**:

1. Click on the **"Commit"** section.
2. Click **"Push to repository"**.
3. **Done!** Your changes are now in the main project. 🎉
4. 





PRACTICAL NO 6

AIM: Performing merge requests using GitLab

**🔹 What is a Merge Request in GitLab?**

A **Merge Request (MR)** in GitLab allows you to **merge changes from one branch** into another (usually main). It is used in team collaboration to review and approve code before merging it.

**Step 1: Open Your GitLab Project**

1. Go to <https://gitlab.com/>.
2. **Login** to your GitLab account.
3. Click on the **"Projects"** tab.
4. Select your **repository (project)**.

**Step 2: Create a New Branch**

1. Click on **"Repository" → "Branches"** in the left menu.
2. Click **"New Branch"**.
3. Type a **branch name** (Example: feature-update).
4. Click **"Create Branch"**.

**Step 3: Edit a File in Your New Branch**

1. In your **new branch**, open any file (or create a new one).
2. Make some changes.  
   Example: If editing an index.html file, modify it like this:
3. <h1>Welcome to GitLab!</h1>
4. <p>This is my new feature update.</p>
5. Click **Save ✅**.

**Step 4: Commit Your Changes**

1. Click **"Commit..."**.
2. Type a **commit message** (Example: "Added new feature update").
3. Click **"Commit"**.

**Step 5: Create a Merge Request**

1. Go to **"Merge Requests"** in the left menu.
2. Click **"New merge request"**.
3. Select:
   * **Source branch**: feature-update (your new branch).
   * **Target branch**: main (where you want to merge).
4. Click **"Compare branches and continue"**.
5. Add a **title** (Example: "Feature Update Merge Request").
6. Click **"Create merge request"**.

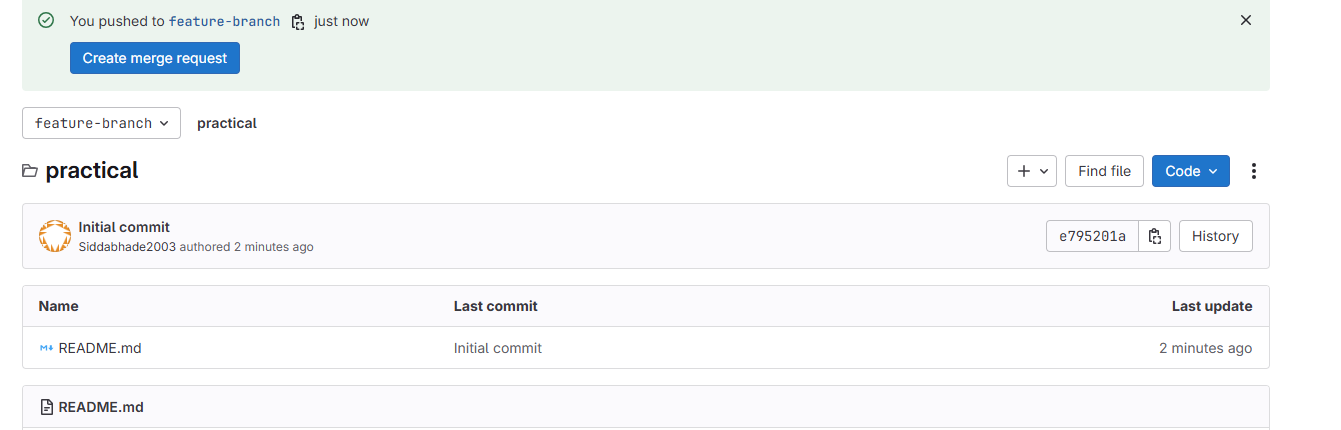
**Step 6: Review and Merge**

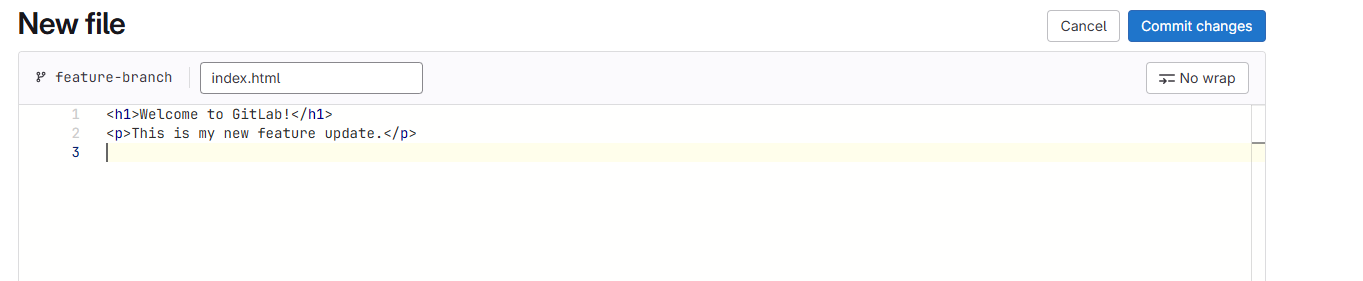
1. Your team (or you) can **review the changes**.
2. If everything looks good, click **"Merge"**.
3. Your changes are now **merged into the main branch! 🎉**

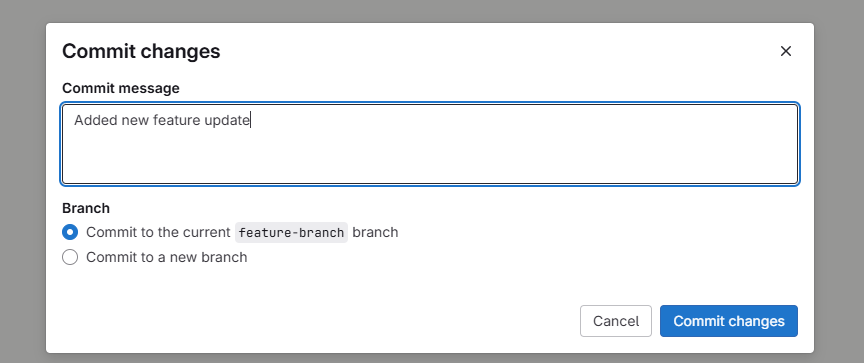
**Step 7: Delete the Branch (Optional)**

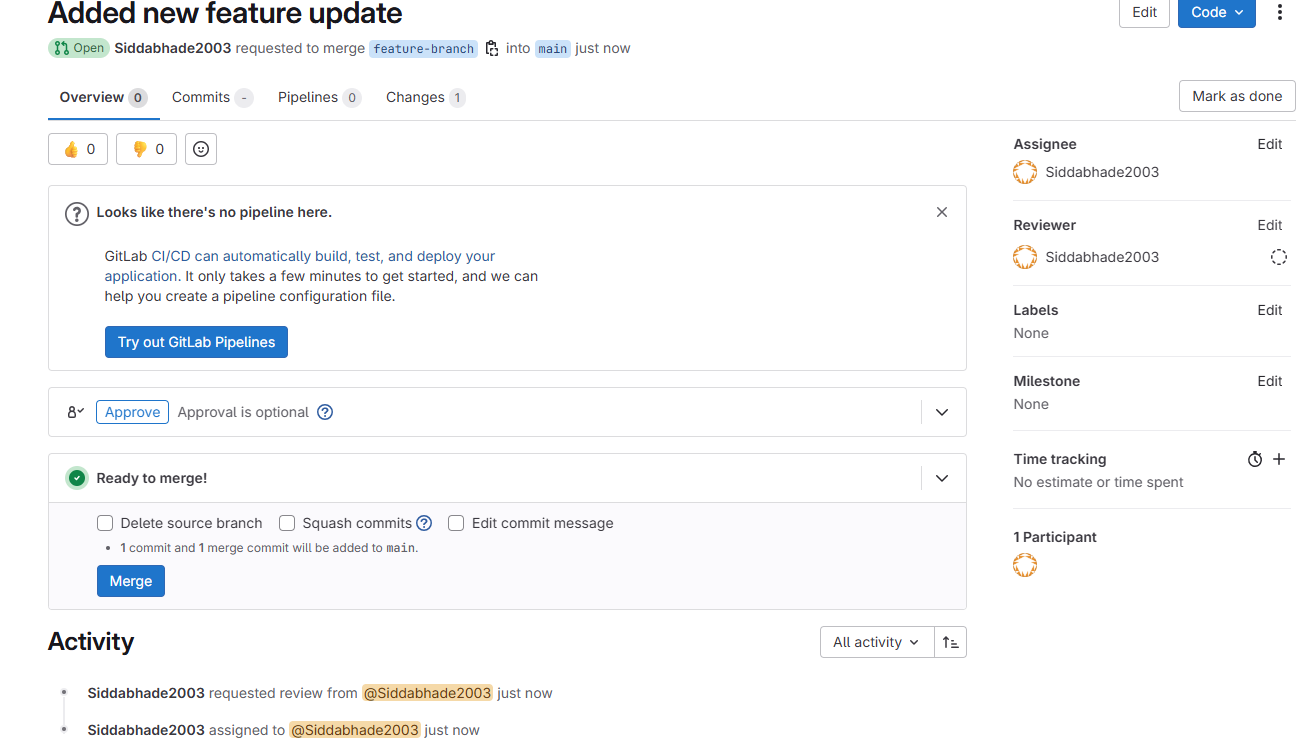
After merging, you can delete the feature-update branch:

1. Go to **"Repository" → "Branches"**.
2. Click **"Delete branch"** next to feature-update.



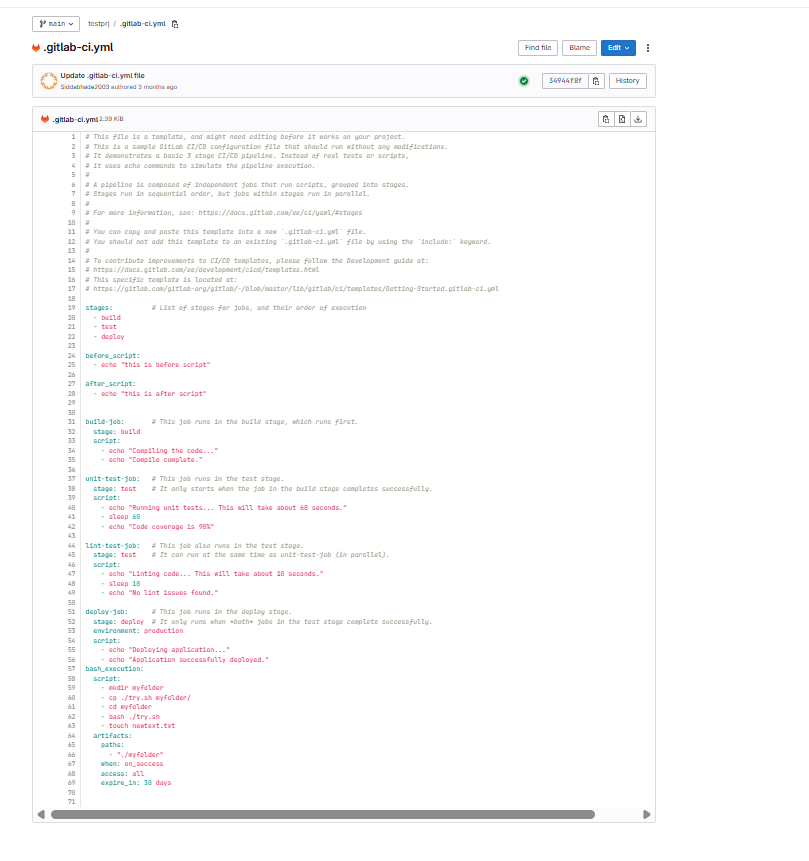


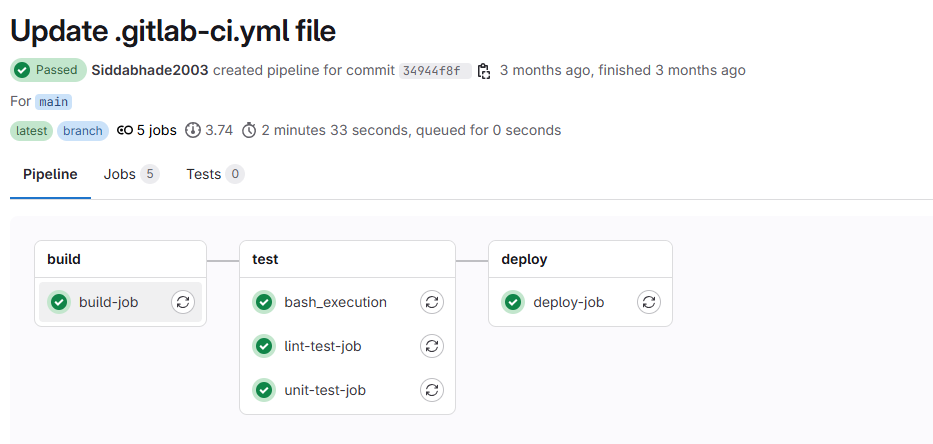


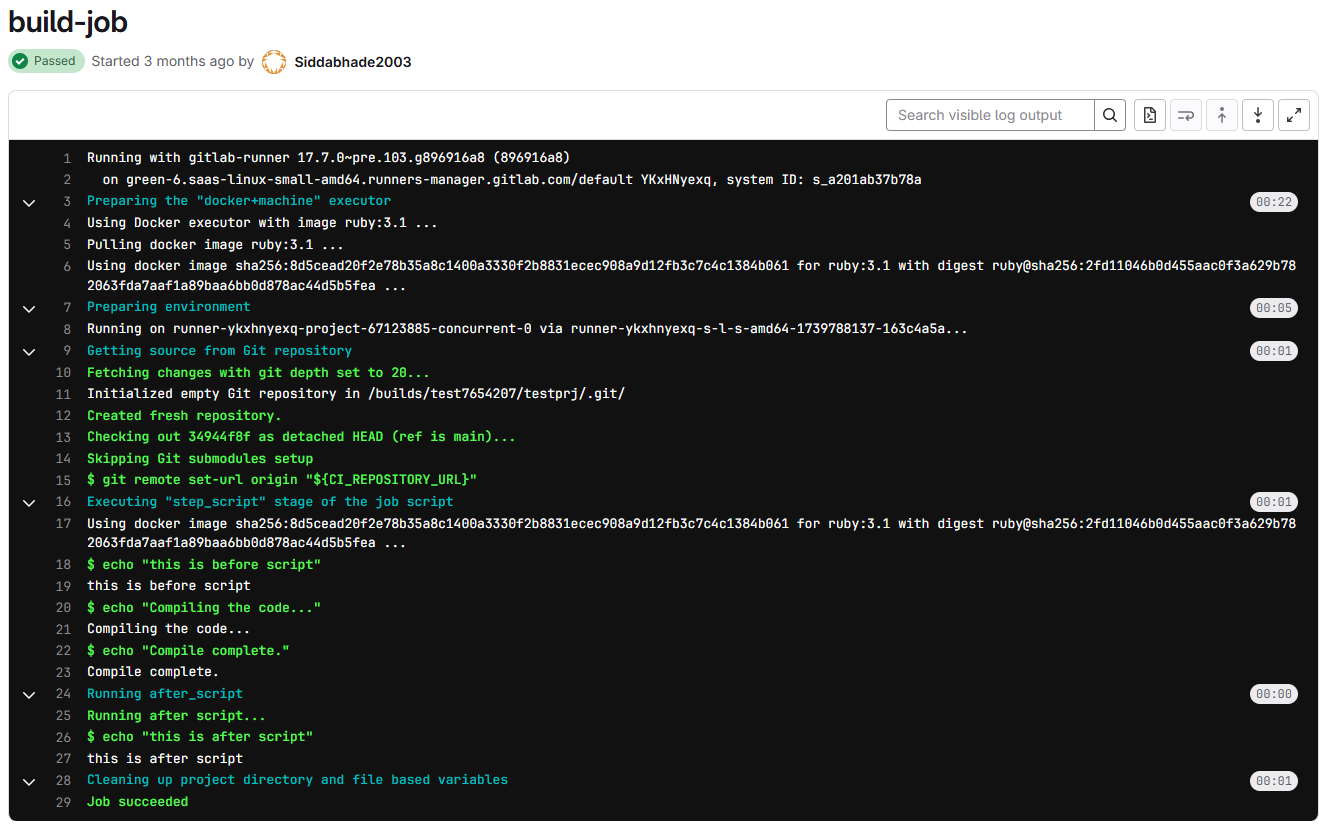


PRACTICAL NO 7

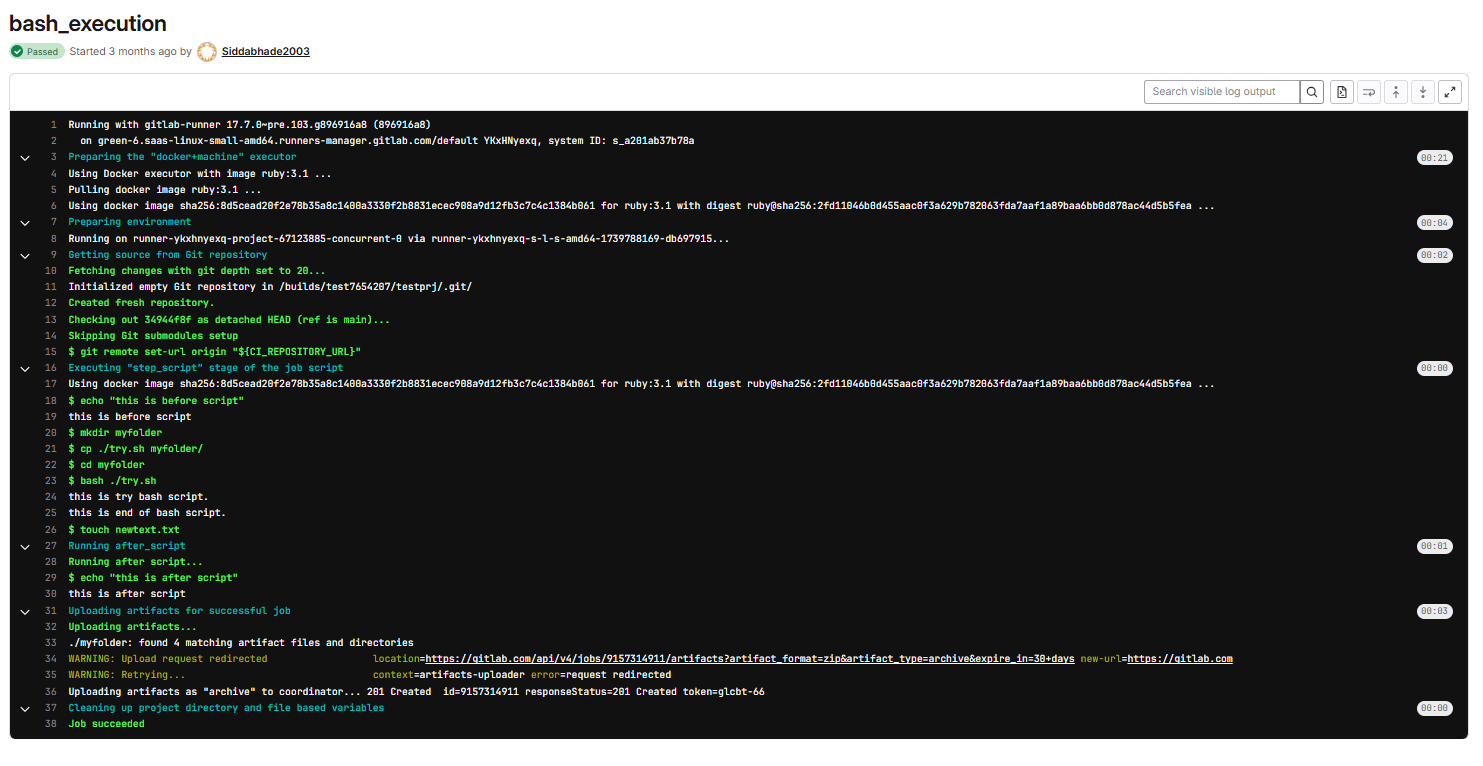
AIM: Workflow management in GitLab





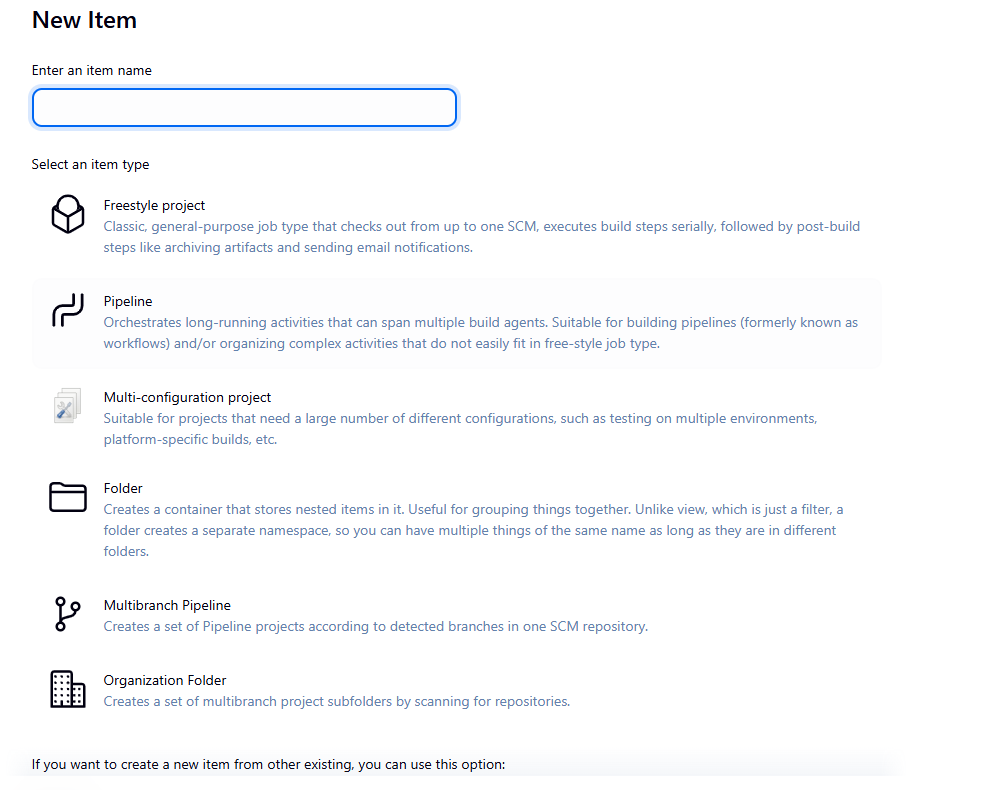






PRACTICAL NO 8

AIM: Demonstrate Continuous Integration and development using Jenkins







**Demonstrate a simple CI/CD process using Jenkins for a sample Java or Python application:**

Create a GitHub repository with a sample application (e.g., a “Hello World”

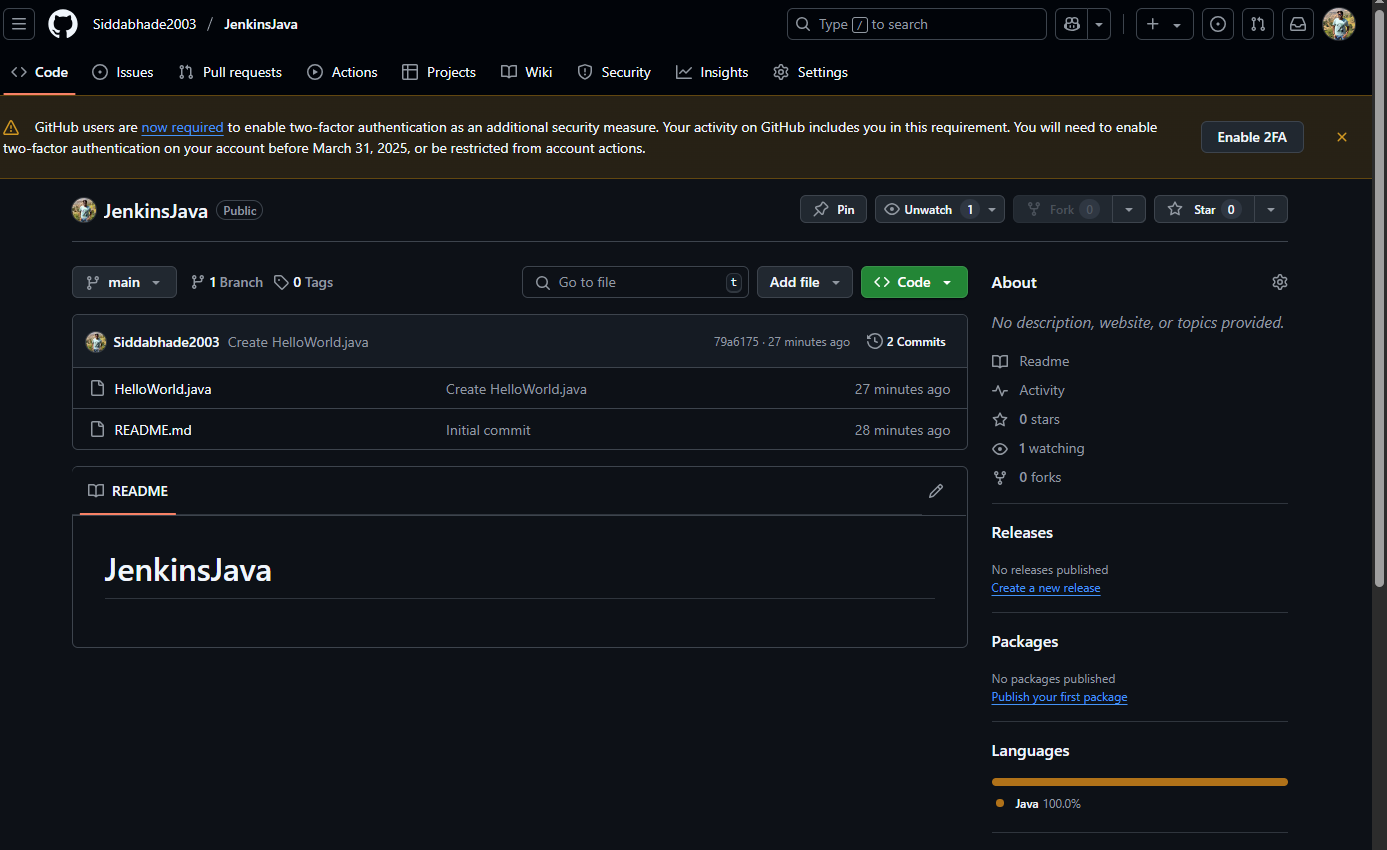
program).

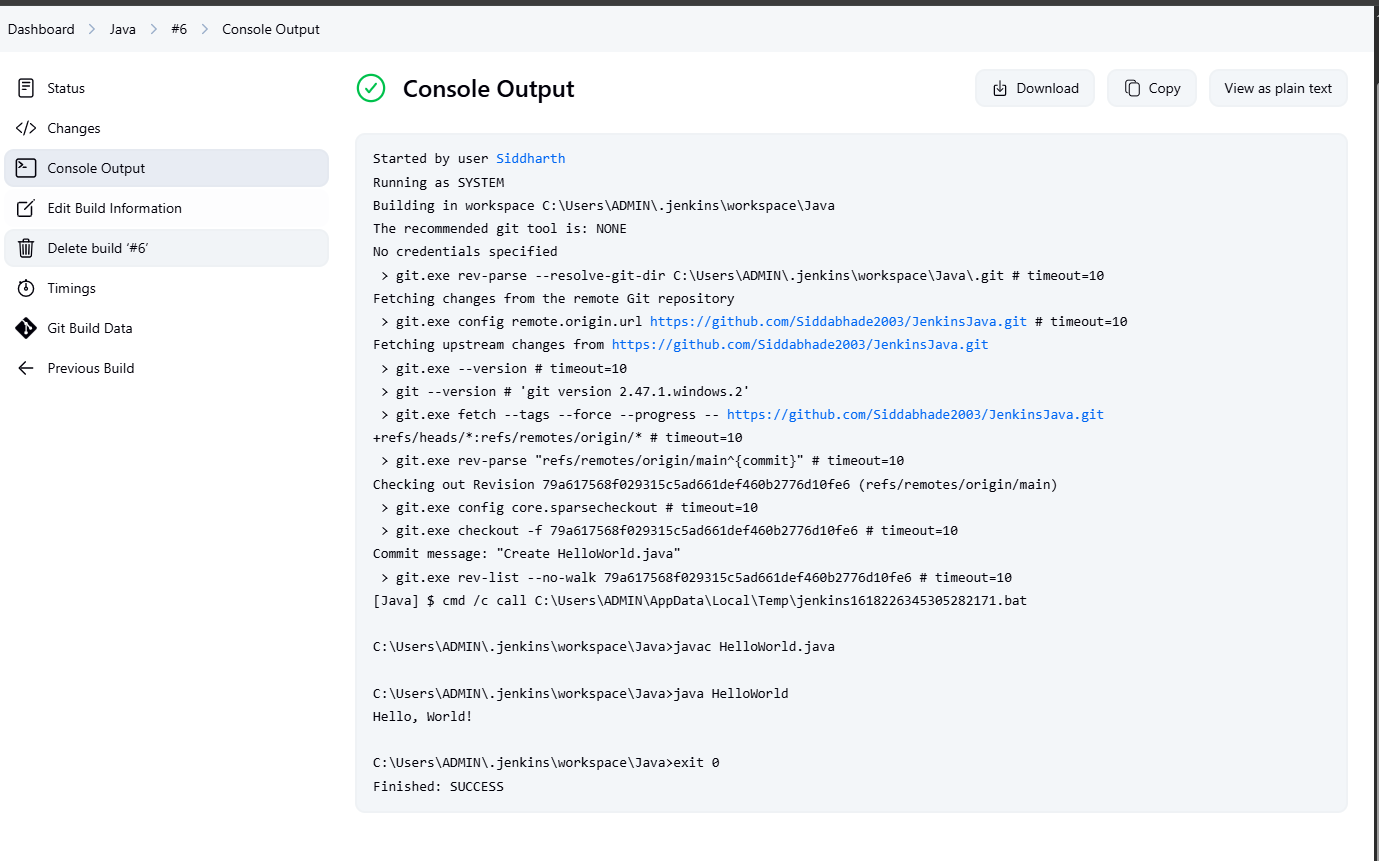
Configure a Jenkins freestyle job to pull the project from GitHub.

Set up build steps (compilation or script execution).

Trigger a build manually or automatically after a commit.

Display and explain build logs and output artifacts.





PRACTICAL NO 9

AIM: Explore docker commands for content management

**🪟 WINDOWS-ONLY GUIDE: Docker for Content Management (Beginner-Friendly)**

**🟢 Step 1: What You Need**

* A Windows 10/11 PC (preferably 64-bit)
* Internet connection
* Basic file navigation skills (nothing fancy!)

**🧰 Step 2: Install Docker Desktop on Windows**

1. Go to: <https://www.docker.com/products/docker-desktop>
2. Click the **Windows Download** button.
3. Once downloaded, **double-click the installer** and follow the steps:
   * Click *Yes* when it asks for permission.
   * Leave settings as default.
4. After installation, **restart your computer** if prompted.
5. Open **Docker Desktop** (you’ll see a blue whale 🐳 icon in your system tray).
6. Wait until it says **Docker is running**.

**✅ Step 3: Test Docker is Working**

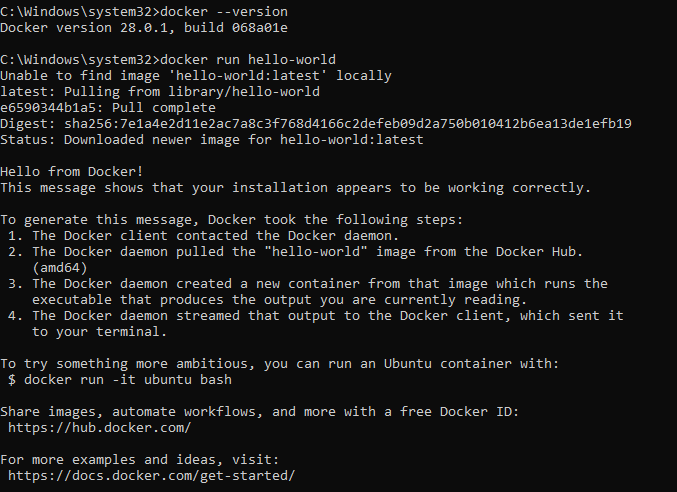
Open **Command Prompt** (type cmd in the Start menu) and run:

docker --version

Then try:

docker run hello-world

If you see a success message, Docker is installed properly. 🎉

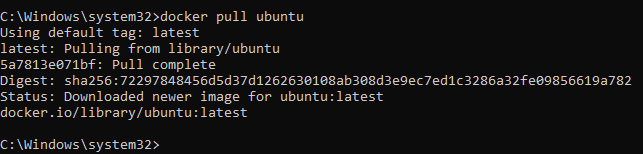


**📥 Step 4: Download a Linux Container Image**

Still in Command Prompt, type:

docker pull ubuntu

This downloads a small Linux system image called **Ubuntu** – like a mini computer inside your PC.



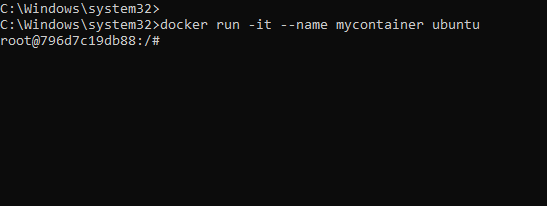
**🚀 Step 5: Start a Container**

Now, run your Ubuntu container:

docker run -it --name mycontainer ubuntu

You’ll now see a Linux-style terminal prompt like:

root@randomID:/#

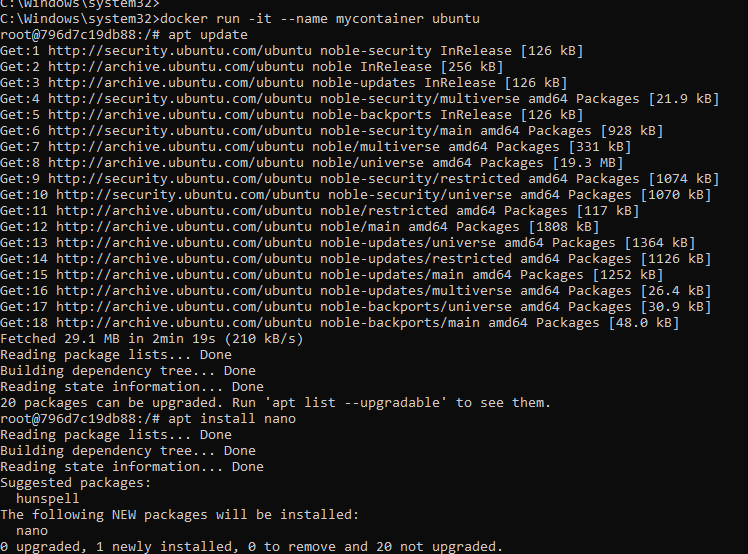


You're inside your container!

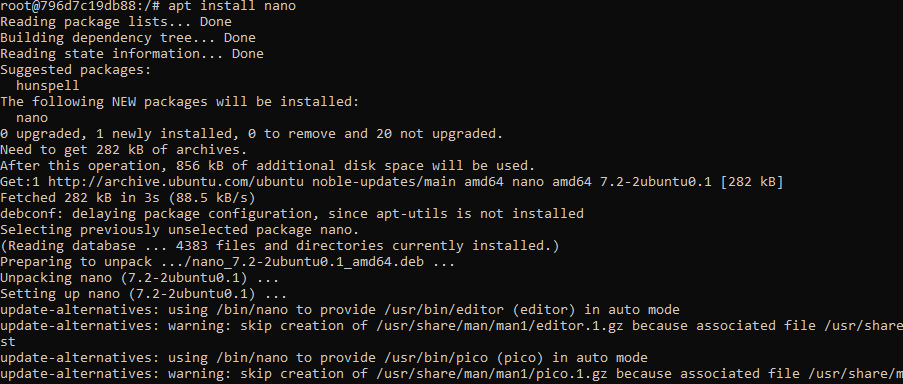
**📁 Step 6: Create and Manage Files**

Do basic stuff like:

apt update

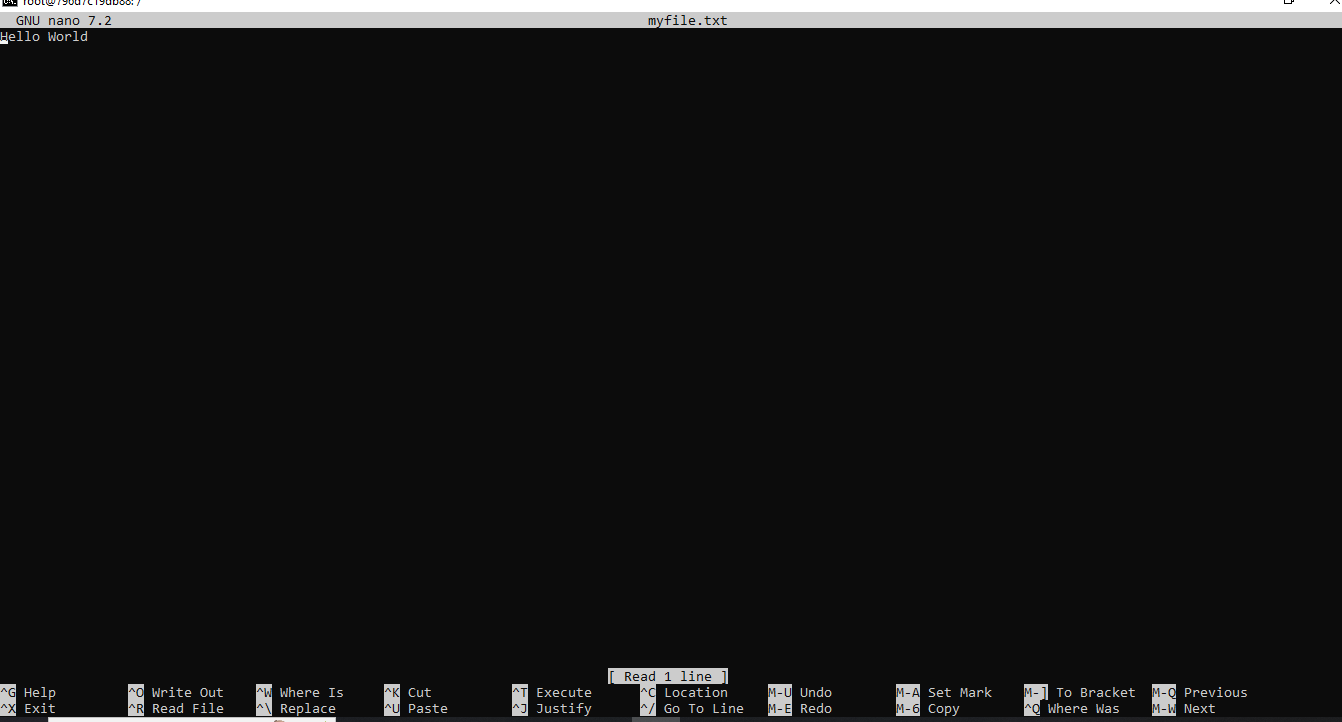


apt install nano



nano myfile.txt   # Write something and save it





Other commands:

mkdir myfolder



mv myfile.txt myfolder/



**🔁 Step 7: Move Files Between Windows and Container**

Open a second Command Prompt (outside the container) and try:

📤 **Copy from Windows to Container**:

docker cp C:\Users\YourName\Desktop\notes.txt mycontainer:/root/



📥 **Copy from Container to Windows**:

docker cp mycontainer:/root/myfile.txt C:\Users\YourName\Desktop\



✅ You just moved content in and out of your Docker container!

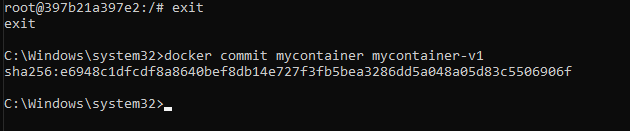
**💾 Step 8: Save Your Container**

Exit the container:

exit

Then save your changes as a new image:

docker commit mycontainer mycontainer-v1



This creates a reusable image with all your files.

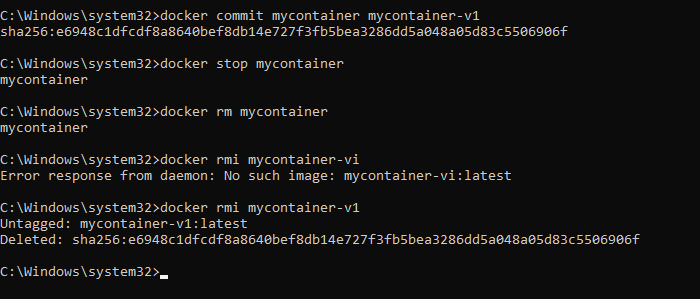
**🧹 Step 9: Cleanup (Optional)**

To stop and delete things:

docker stop mycontainer

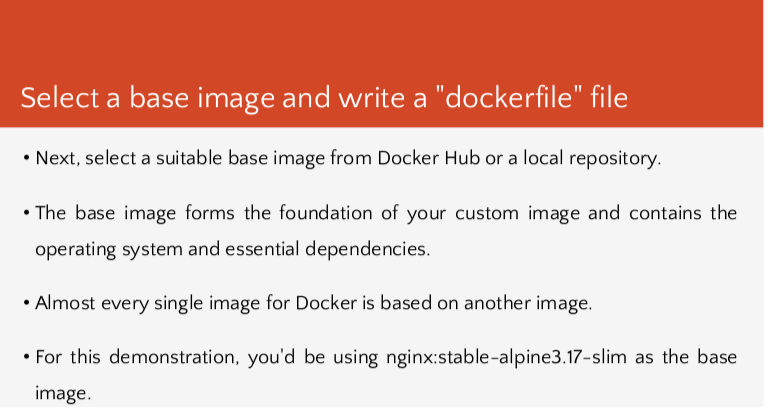
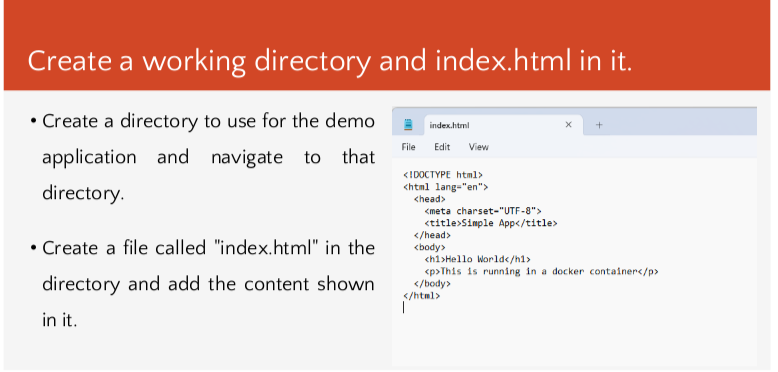
docker rm mycontainer

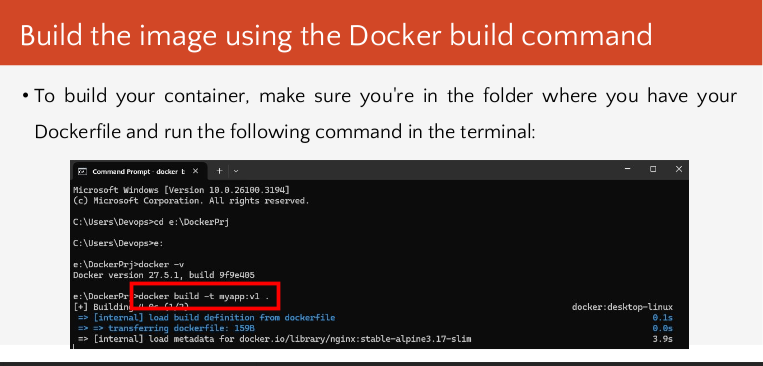
docker rmi mycontainer-v1

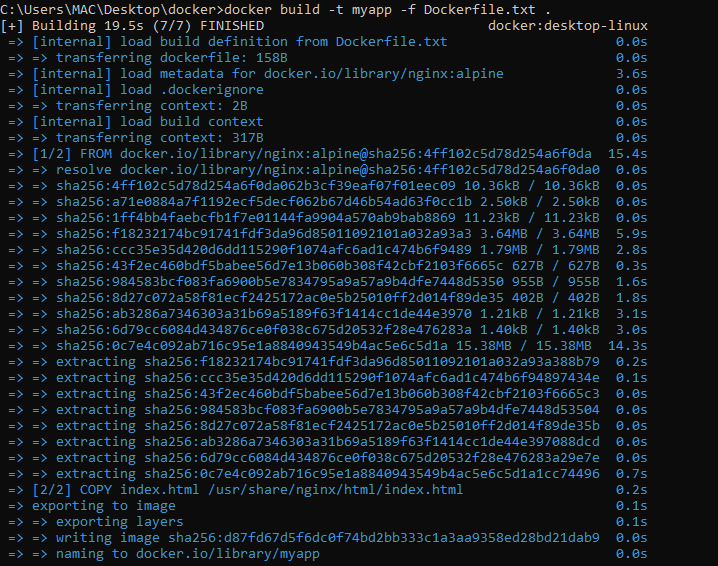


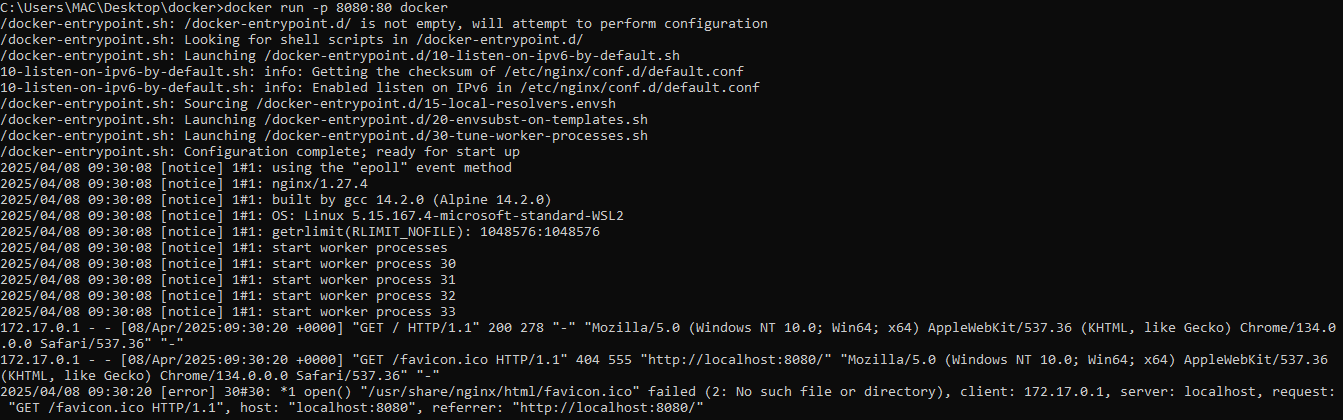
PRACTICAL NO 10

AIM: Develop a simple containerized application using Docker



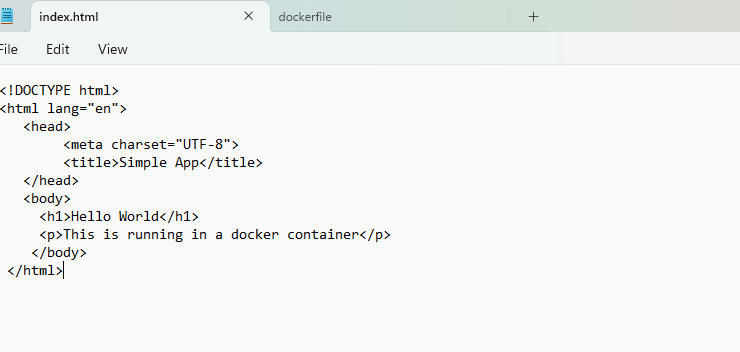
1. Save file as “Dockerfile” all files
2. By default the error is generated while saving the dockerfile so we need to rename the docker file by using a command:
3. Dir – enter ( all files will be visible)
4. To change docker file.txt rename as rename dockerfile.txt dockerfile ( press enter)
5. 



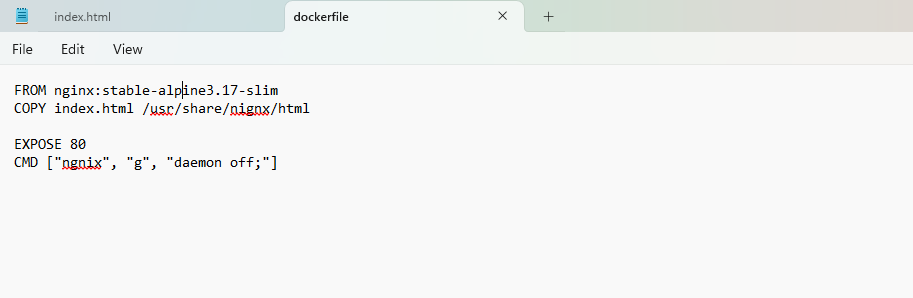


Build, deploy and manage web application on Docker Engine

Create a working directory and index.html in it.

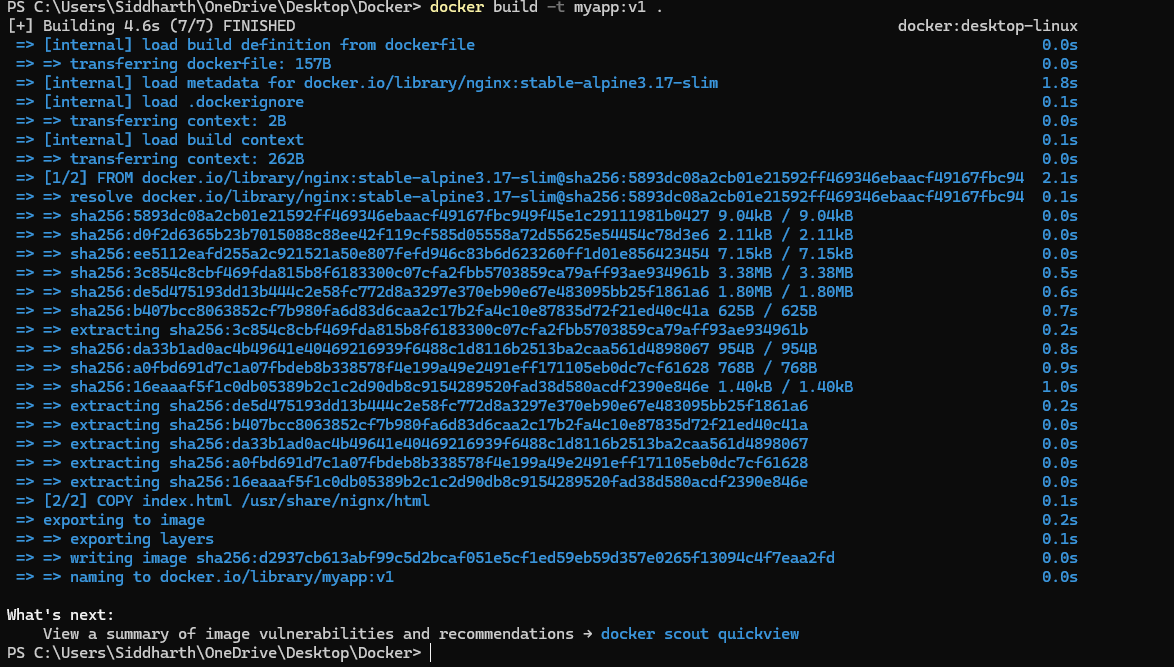


Select a base image and write a "dockerfile" file

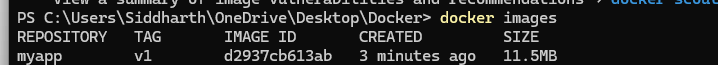


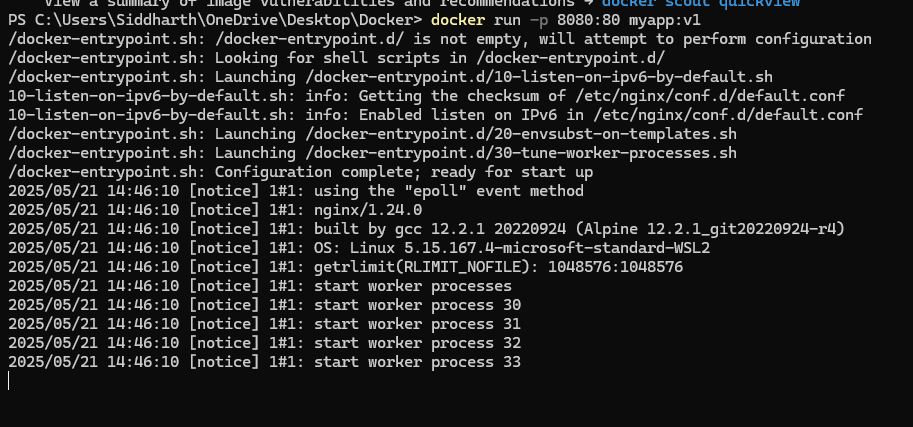
Build the image using the Docker build command



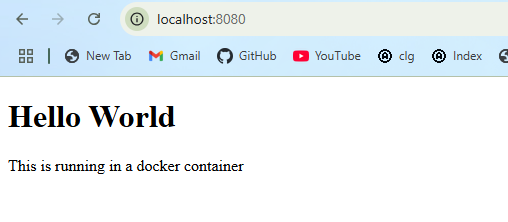


Verify and Run the Docker image





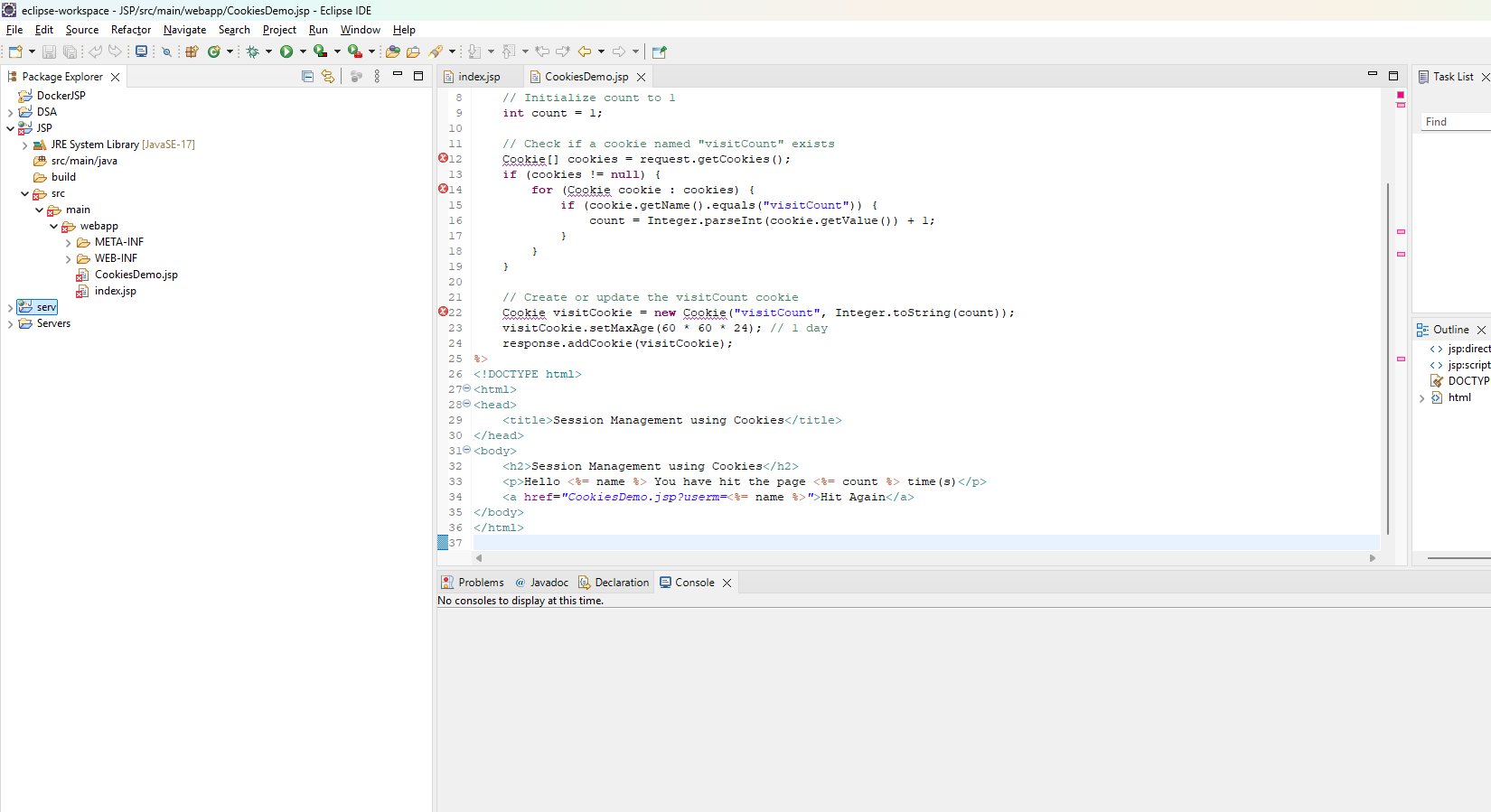
Access the application



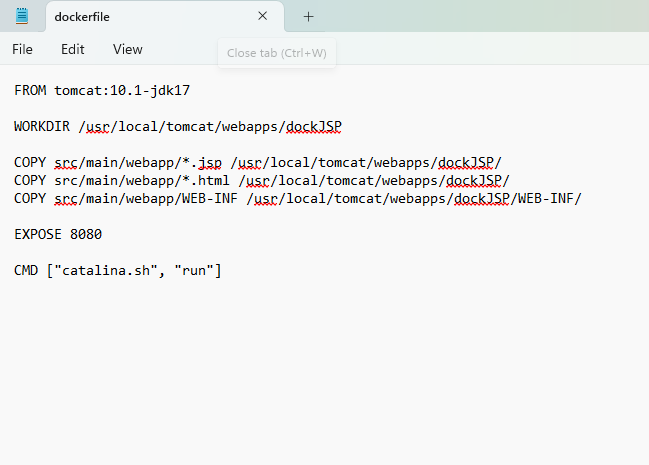
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Dealing with JSP application

Creating JSP Project

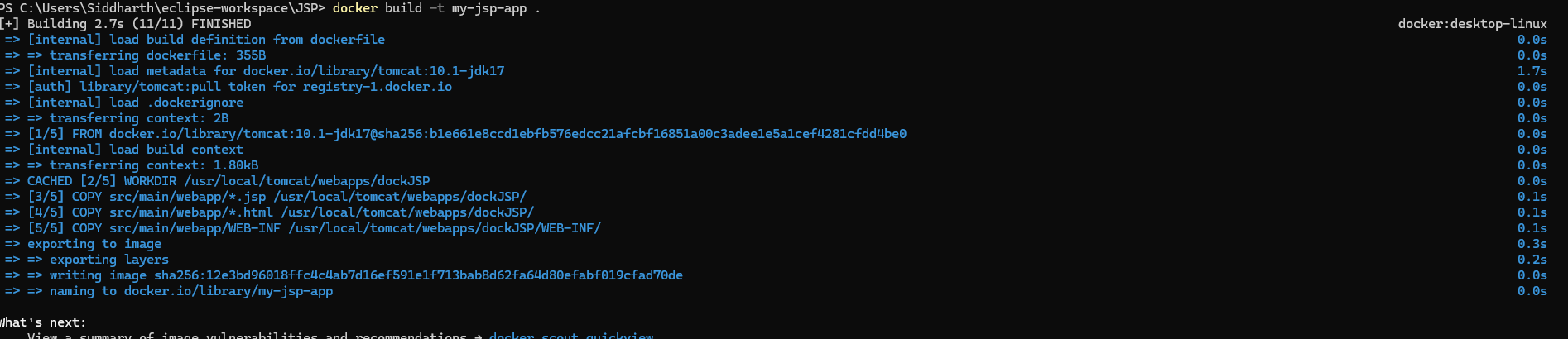


Creating dockerfile

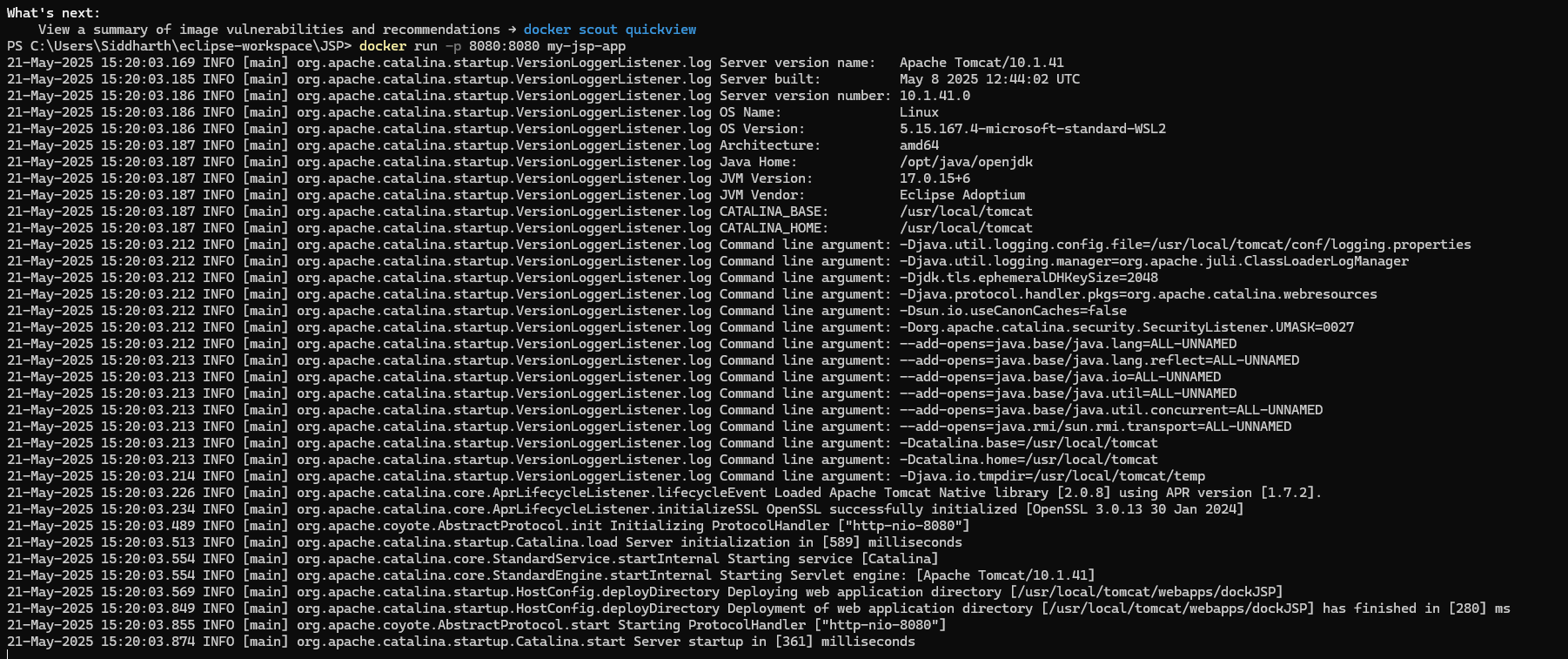


Install Tomact from Web

Build the image using the Docker build command



Verify and Run the Docker image



Access the application

