Roll Number: C24079

MCAL23 DevOps Lab INDEX				
Sr.No	**Title**	**CO**	**Date**	**Sign**
1	Demonstrate basic Git commands	CO2		
2	Create and fork repositories in Git Hub. Apply branch, merge and rebase concepts.	CO2		
3	Demonstrate Git for Collabration	CO2		
4	Demonstrate Collabrating and cloning using Git	CO2		
5	Using GitLab Web IDE	CO2		
6	Demonstrate CI/CD Workflow in GitLab using .py, .bash, .java file	CO2		
7	Demonstrate setting Jenkins CI/CD pipeline.	CO3		
8	Demonstrate Setting up of a CI/CD pipeline to build and deploy a web application to a local HTTP server	CO3		
9	Demonstrate basic Docker commands	CO3		
10	Develop a simple containerized application using Docker	CO3		
11	Demonstrate add-on ansible commands	CO4		
12	Demonstrate Ansible Playbooks	CO4		

Practical 1
AIM: Demonstrate basic Git commands

1. Set-up Git (One-time setup)
    - **git config –global user.name "sam"** -- Set your name for commits.
    - **git config –global user.email "shramikapatne@gmail.com"** -- Set your email for commits.

```
hp@DESKTOP-RL6G3J5 MINGW64 ~
$ git config --global user.name "sam"

hp@DESKTOP-RL6G3J5 MINGW64 ~
$ git config --global user.email "shramikapatne@gmail.com"
```

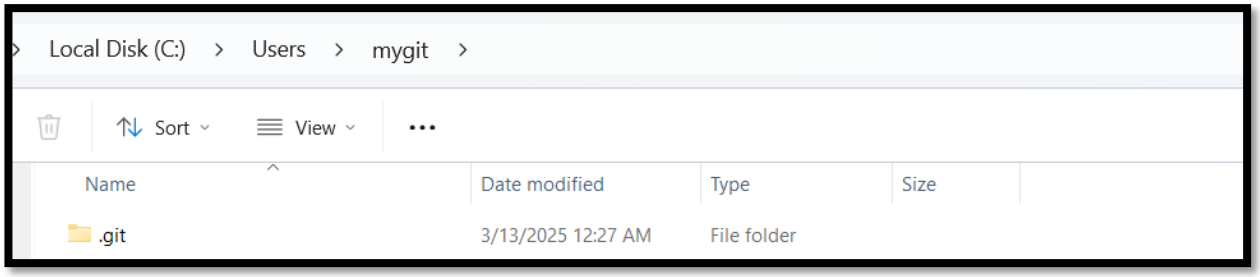
```
hp@DESKTOP-RL6G3J5 MINGW64 ~
$ git config --global user.email
shramikapatne@gmail.com

hp@DESKTOP-RL6G3J5 MINGW64 ~
$ git config --global user.name
sam
```

2. Start a new project

- **git init :-** Initialize a new repository in your project folder.
- **git status:-** Check the status of files in the working directory (untracked, modified, etc.).
- **git add . :-** Stage all changes (new, modified, or deleted files).
- **git commit -m "Initial Commit" :-** Commit the staged files with a message describing the changes

```
hp@DESKTOP-RL6G3J5 MINGW64 ~ (master)
$ cd "C:/Users/newgit"
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit
$ git init
Initialized empty Git repository in C:/Users/newgit/.git/
```




```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git commit -m "Initial Commit"
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        myfile.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git add .

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git commit -m "Initial Commit"
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
```

3.Work on a New Feature

- **git branch <branch_name> :-** Create a new branch for a feature (e.g., feature-1).
- **git checkout <branch_name>** :- Switch to the new branch.

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git branch new-feature

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git checkout new-feature
Switched to branch 'new-feature'
```

4. Make Changes and Commit them

- **git add <file>:-** Stage specific files for commit.
- **git commit -m "Add new feature" :-** Commit changes with a descriptive message.
- **git log :-** View the history of commits in the current branch.

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (new-feature)
$ git add myfile.txt

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (new-feature)
$ git status
On branch new-feature
nothing to commit, working tree clean

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (new-feature)
$ git commit -m "Added New Feature"
On branch new-feature
nothing to commit, working tree clean

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (new-feature)
$ git log
commit 502f58640a499a58a2f04ededf762e2a0991c9c0 (HEAD -> new-feature, master)
Author: sam <shramikapatne@gmail.com>
Date:   Tue May 20 16:34:48 2025 +0530

    Initial Commit
```

5. Merge changes back into main branch.

- **git checkout main** :- Switch back to the main branch
- **git merge <branch_name>** :- Merge the changes from the feature branch into the main branch.

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (new-feature)
$ git checkout master
M       myfile.txt
Switched to branch 'master'

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git merge new-feature
Already up to date.
```

6.Set up a remote repository

- **git remote add origin <repository-url> :-** Link your local repository to a remote repository (e.g., on GitHub).
- **git push -u origin main**

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git remote add origin "https://github.com/Shramikapatne20/myrepo_sam79.git"

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git push -u origin master
To https://github.com/Shramikapatne20/myrepo_sam79.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Shramikapatne20/myrepo_sam79.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git pull origin master --rebase
error: cannot pull with rebase: You have unstaged changes.
error: Please commit or stash them.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git add .

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git commit  -m "save local changes before rebase"
[master 6c75a0d] save local changes before rebase
 1 file changed, 1 insertion(+), 1 deletion(-)

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git pull origin master --rebase
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 190 bytes | 31.00 KiB/s, done.
From https://github.com/Shramikapatne20/myrepo_sam79
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master
Successfully rebased and updated refs/heads/master.
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 552 bytes | 552.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Shramikapatne20/myrepo_sam79.git
   202bd0e..8c490e8  master -> master
branch 'master' set up to track 'origin/master'.
```

7.Collaborate with a team

- **git pull origin main :-** Fetch and merge changes from the remote main branch to your local branch.
- **git branch:-** List all branches to see if new ones were created by collaborators.
- **git fetch:-** Download updates from the remote without merging them.

- **git diff** :- Compare changes between your working directory and the staging area, or between commits.
- **git push** :- Push your committed changes to the remote branch

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git pull origin master
From https://github.com/Shramikapatne20/myrepo_sam79
 * branch            master     -> FETCH_HEAD
Already up to date.
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git branch
* master
  new-feature

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git fetch
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 32 (delta 0), reused 12 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (32/32), 8.12 KiB | 143.00 KiB/s, done.
From https://github.com/Shramikapatne20/myrepo_sam79
 * [new branch]      main       -> origin/main
 * [new branch]      myb        -> origin/myb

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git diff

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git push
Everything up-to-date
```

Practical 2

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

Step 1: Initial Setup

1. **Create a Git repository (if not already created)**

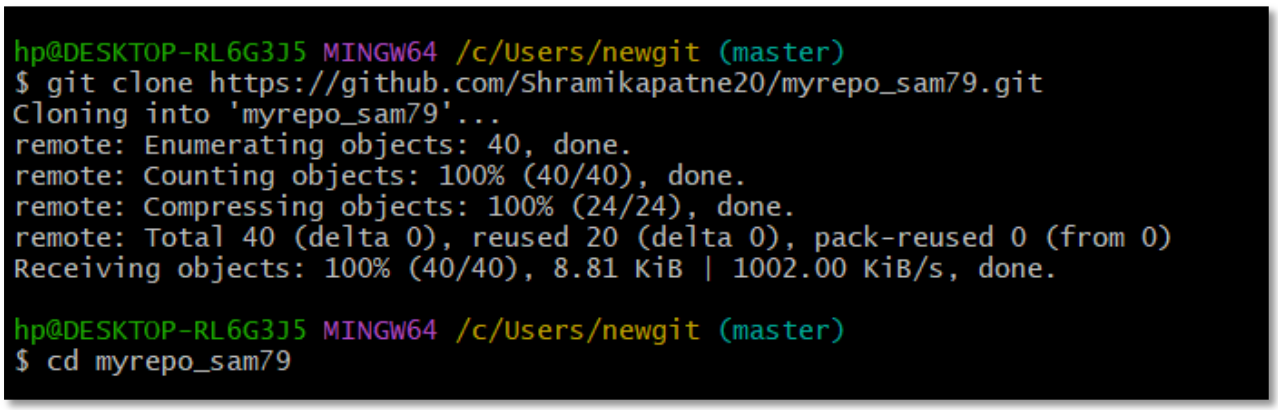
   If you don't have a repository yet, you can create one by running:

   ```
   git init
   ```

2. **Clone an existing repository (if you're working on an existing project)**

   If you're working with an existing remote repository, you can clone it by running:

   ```
   git clone <repository-url>
   cd <repository-name>
   ```



Step 2: Working with Branches

1. **Check the current branch** By default, Git starts with a branch named `main` or `master`. To see which branch you are currently on, use:

   ```
   git branch
   ```

   The current branch will be marked with an asterisk (*).

2. **Create a new branch** To create a new branch, use:

   ```
   git branch <branch-name>
   ```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (main)
$ git branch new_branch
```

3. **Switch to the new branch** To start working on the new branch, use:

   ```
   git checkout <branch-name>
   ```

   You can also combine the creation and switch into one command:

   ```
   git checkout -b <branch-name>
   ```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (main)
$ git checkout new_branch
Switched to branch 'new_branch'

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git checkout -b branch1
Switched to a new branch 'branch1'
```

4. **View all branches** To see all branches in your repository:

   ```
   git branch
   ```

   The current branch will be marked with an asterisk (*).

Step 3: Make Changes in the Branch

1. **Make some changes in the code**

   Now that you're on your new branch, make some changes to your files (e.g., modify code, add new features, etc.).

2. **Stage the changes** After making changes, you need to add them to the staging area:

   ```
   bash
   git add <file-name>  # Add a specific file
   git add .            # Add all files (recommended if you want to stage
   everything)
   ```

‘

3. **Commit the changes** After staging, commit the changes to your branch:

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ git add myfile.txt
fatal: pathspec 'myfile.txt' did not match any files

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ git add myfile.txt
fatal: pathspec 'myfile.txt' did not match any files

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ ls
hello.txt  new.txt  test.txt

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ git add hello.txt

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ git add .

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ git status
On branch branch1
nothing to commit, working tree clean

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ git commit -m "decsription of changes"
On branch branch1
nothing to commit, working tree clean
```

```
git commit -m "Description of changes"
```

git

Step 4: Merge the Branch into Main

1. **Switch to the `main` branch (or the branch you want to merge into)** Before merging, switch back to the `main` branch:

```
git checkout main
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (branch1)
$ git checkout master
branch 'master' set up to track 'origin/master'.
Switched to a new branch 'master'

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git pull origin
Already up to date.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git merge branch1
fatal: refusing to merge unrelated histories

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git init
Reinitialized existing Git repository in C:/Users/newgit/myrepo_sam79/.git/

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git merge branch1
fatal: refusing to merge unrelated histories

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git branch
  branch1
  main
* master
  new_branch

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git merge new_branch
fatal: refusing to merge unrelated histories
```

2. **Pull the latest changes** Ensure your `main` branch is up to date with the remote repository:

   ```
   git pull origin main
   ```

3. **Merge the feature branch into `main`** Now merge your branch into `main`:

   ```
   git merge <branch-name>
   ```

   - If there are no conflicts, Git will automatically complete the merge and add a merge commit.
   - If there are conflicts, Git will notify you, and you'll need to resolve them manually.

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git merge branch1 --allow-unrelated-histories
Auto-merging hello.txt
CONFLICT (add/add): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then commit the result.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master|MERGING)
$ git add hello.txt
```

Step 5: Resolving Merge Conflicts (If Any)

1. **Check for conflicts** If Git encounters conflicts during the merge, it will pause and mark the conflicted files.
2. **Open the conflicted files** Conflicted sections will be marked with:

```
<<<<<<< HEAD
(changes from `main` branch)
=======
(changes from `<branch-name>`)
>>>>>>> <branch-name>
```

3. **Resolve the conflicts** Edit the file to keep the changes you want and remove the conflict markers (<<<<<<<, =======, >>>>>>>).
4. **Mark the conflicts as resolved** After resolving conflicts, stage the files as resolved:

```
git add <resolved-file>
```

5. **Complete the merge** Once all conflicts are resolved, commit the merge:

```
git commit
```

Git will automatically create a merge commit if you didn't need to resolve conflicts manually

Step 6: Push Changes to Remote

1. **Push the changes to the remote repository** After merging, push the changes to the remote repository:

```
git push origin main
```

This updates the remote repository with the changes from your merge.

Step 7: Clean Up (Optional)

1. **Delete the branch after merging (optional)** After merging, you can delete your feature branch if you no longer need it:

```
git branch -d <branch-name>  # Deletes the local branch
```

2. **Delete the remote branch (optional)** If you want to delete the branch on the remote as well, use:

```
git push origin --delete <branch-name>
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master|MERGING)
$ git commit -m "Merged branch1 into master allowing unrelated histories"
[master f756b20] Merged branch1 into master allowing unrelated histories

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git push origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 401 bytes | 401.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0 (from 0)
To https://github.com/Shramikapatne20/myrepo_sam79.git
   8c490e8..f756b20  master -> master

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git branch -d branch1
Deleted branch branch1 (was 23cf05c).

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ ^[[200~git push origin --delete branch1
bash: $'\E[200~git': command not found
```

Step 8: Regular Maintenance

1. **Sync your local repository with the remote regularly** To avoid conflicts, it's good practice to frequently pull changes from the main branch into your working branch:

```
git checkout <branch-name>    # Switch to your feature branch
git pull origin main          # Pull the latest changes from main
```

2. **Stay organized**
   - Use descriptive branch names (e.g., `feature/auth`, `bugfix/login`).
   - Regularly merge back into `main` to keep your changes synchronized.

Practical 3
AIM: Demonstrate Git for Collaboration

Set Up Git and GitHub

Before you start collaborating or cloning repositories, make sure you have the following set up:

1. **Install Git**: If you haven't already, download and install Git on your machine.
2. **Create a GitHub Account**: Go to GitHub and create an account if you don't have one.
3. **Configure Git**: Set up your Git configuration with your name and email.

git config --global user.name "Your Name"
git config --global user.email youremail@example.com

Step 2: Clone a GitHub Repository

Cloning a repository allows you to create a copy of a project on your local machine, enabling you to work on it.

1. **Find a Repository to Clone**: Visit the repository page on GitHub (e.g., https://github.com/username/repository) and click the green **Code** button.
2. **Copy the Clone URL**: In the popup, choose either **HTTPS** or **SSH** and copy the URL. If you're using HTTPS, it will look like `https://github.com/username/repository.git`.
3. **Clone the Repository Locally**:

   Open a terminal on your computer and navigate to the directory where you want to clone the repository. Then run:

   ```
   git clone https://github.com/username/repository.git
   ```

- Replace `https://github.com/username/repository.git` with the URL you copied.

This will create a local copy of the repository on your machine.

- **Navigate to the Repository Folder**:

cd repository # Navigate into the cloned directory

```
hp@DESKTOP-RL6G3J5 MINGW64 ~ (master)
$ git config --global user.name "sam"

hp@DESKTOP-RL6G3J5 MINGW64 ~ (master)
$ git config --global user.email "shramikapatne@gmail.com"

hp@DESKTOP-RL6G3J5 MINGW64 ~ (master)
$ git config --global user.email
shramikapatne@gmail.com
```

```
hp@DESKTOP-RL6G3J5 MINGW64 ~ (master)
$ cd "C:\Users\newgit"

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git clone "https://github.com/Shramikapatne20/myrepo_sam79.git"
fatal: destination path 'myrepo_sam79' already exists and is not an empty direct
ory.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ cd myrepo_sam79
```

Step 3: Work on the Project Locally

Once you've cloned the repository, you can start making changes to the code.

1. **Create a New Branch**: Before making changes, it's recommended to create a new branch. This ensures your changes don't interfere with the main codebase until you're ready to merge.

```
git checkout -b feature/your-feature  # Create and switch to a new
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git checkout new_branch
Switched to branch 'new_branch'

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git add .

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git commit -m "Added some text"
[new_branch 749e265] Added some text
 1 file changed, 1 insertion(+)
```

1. **Make Changes**: Edit files as needed using your preferred editor or IDE.
2. **Stage Changes**: After making changes, you need to stage them before committing.

   ```
   git add .  # Stages all modified files
   ```

3. **Commit Changes**: Once changes are staged, commit them to your local branch.

   ```
   git commit -m "Add feature X"
   ```

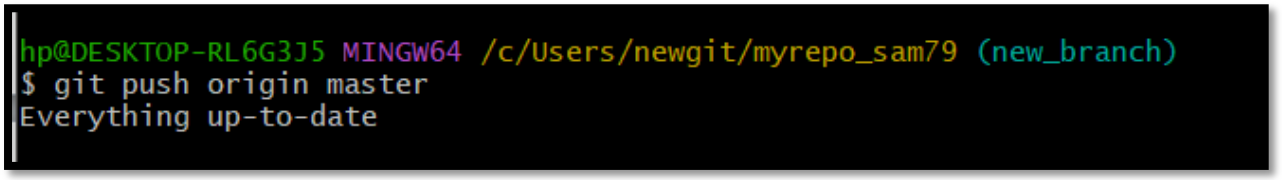
Step 4: Push Changes to GitHub

Once you've committed your changes locally, you need to push them to your GitHub repository.

1. **Push Your Changes** to the remote repository:

```
git push origin feature/your-feature  # Push the feature branch to GitHub
```

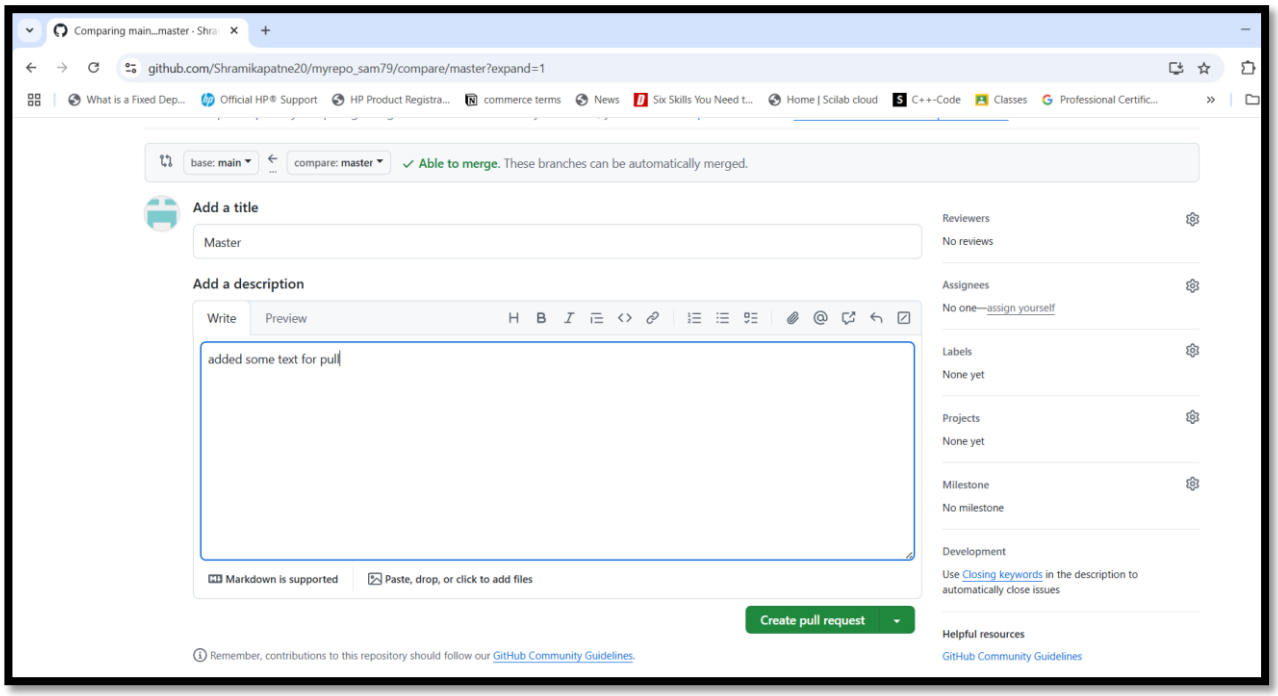
This uploads your local changes to your GitHub repository.



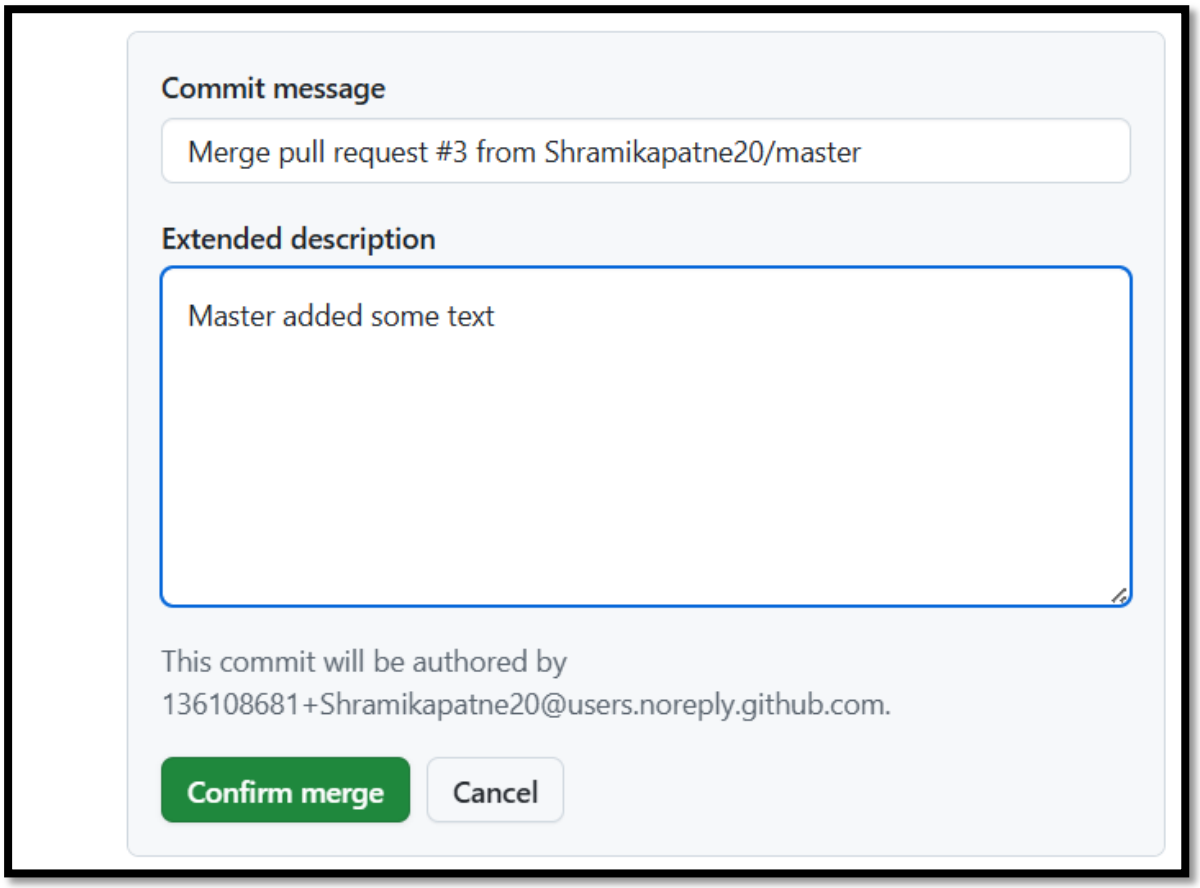
Step 5: Create a Pull Request (PR)

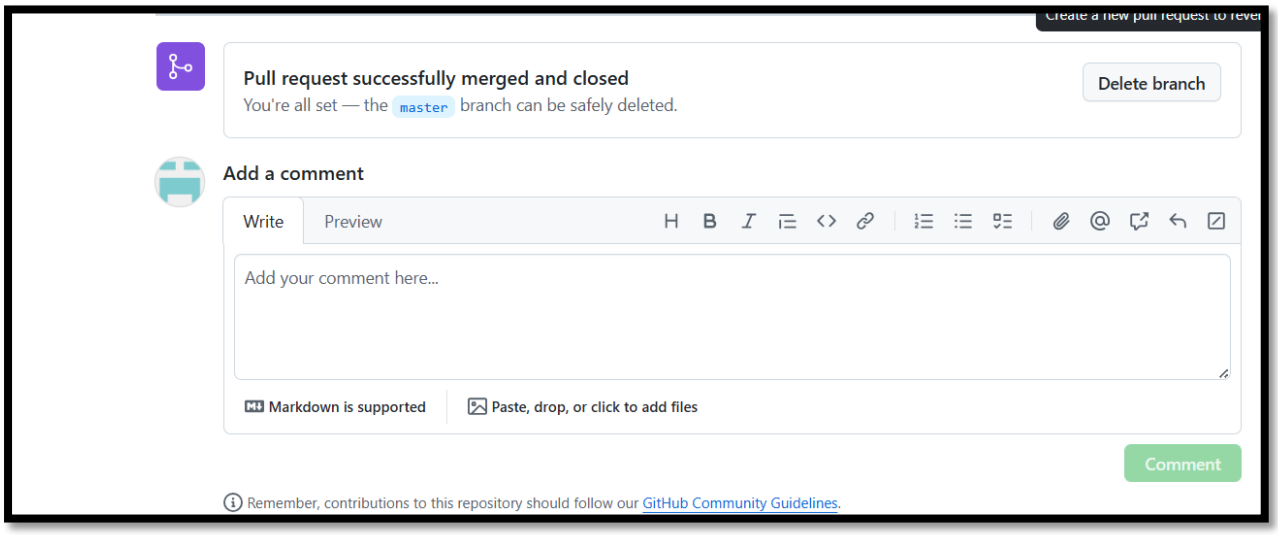
To contribute your changes back to the original repository, you'll need to open a pull request (PR).

1. **Go to the GitHub Repository**: Visit the repository where you want to make the changes.
2. **Create a Pull Request**: On GitHub, you'll see an option to compare your branch (e.g., `feature/your-feature`) with the main branch of the repository (e.g., `main`). Click **New Pull Request**.
3. **Fill Out the Pull Request Form**:
   - Add a **title** and **description** explaining the changes you've made.
   - Review your changes.
   - Click **Create Pull Request**.



4. **Code Review**: The repository maintainer (or other collaborators) will review your changes. They might ask for changes or approve the PR.
5. **Merge the Pull Request**: Once your changes are approved, the maintainer will merge your changes into the main branch of the project.





Step 6: Sync Your Fork (If Working on a Forked Repo)

If you are working on a forked repository and want to keep your fork in sync with the original repository:

1. **Add the Original Repository as a Remote**: This allows you to fetch updates from the original repository.

```
git remote add upstream https://github.com/owner/original-repository.git
```

Replace `https://github.com/owner/original-repository.git` with the original repository's URL.

2. **Fetch the Latest Changes from the Original Repository**:

```
git fetch upstream  # Fetch the changes from the original repo
```

3. **Merge the Latest Changes into Your Local Branch**:

```
git checkout main  # Switch to your main branch
git merge upstream/main  # Merge the latest changes from the original repo
```

4. **Push the Changes to Your Fork**:

```
git push origin main  # Push the updated main branch to your fork
```

Step 7: Pull Latest Changes from the Original Repository

To keep your local repository up-to-date with the remote repository on GitHub, you can pull the latest changes.

1. **Switch to Your Local Main Branch**:

```
git checkout main
```

2. **Pull Latest Changes from GitHub**:

```
git pull origin main  # Pull the latest changes from the remote repository
```

Step 8: Collaborate with Other Developers

When collaborating with other developers on GitHub, you'll typically follow these best practices:

1. **Regularly Pull Latest Changes**: To ensure you're not working on outdated code, frequently pull the latest changes from the main branch (especially before you start working on new features or bug fixes).
2. **Create Feature Branches**: Always create a new branch for each feature or bug fix. This avoids conflicts and keeps the history clean.
3. **Review Pull Requests**: If you're reviewing someone else's PR, ensure you provide feedback and approve it once you're satisfied.
4. **Resolve Merge Conflicts**: If two developers edit the same part of a file, a merge conflict will occur when merging. Resolve these conflicts manually by editing the files and then committing the changes.

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git remote add upstream https://github.com/snehalparab27/demo.git

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git fetch upstream
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 24 (delta 1), reused 5 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (24/24), 8.28 KiB | 184.00 KiB/s, done.
From https://github.com/snehalparab27/demo
 * [new branch]      branch1    -> upstream/branch1
 * [new branch]      main       -> upstream/main

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git push origin master
Everything up-to-date

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (main)
$ git pull origin master
From https://github.com/Shramikapatne20/myrepo_sam79
 * branch            master     -> FETCH_HEAD
Updating 23cf05c..f756b20
Fast-forward
 hello.txt  | 2 +-
 myfile.txt | 1 +
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 myfile.txt
```

Practical 4
AIM: Demonstrate Collaborating and cloning using Git

Set Up Git and GitHub

Before you start collaborating or cloning repositories, make sure you have the following set up:

4. **Install Git**: If you haven't already, download and install Git on your machine.
5. **Create a GitHub Account**: Go to GitHub and create an account if you don't have one.
6. **Configure Git**: Set up your Git configuration with your name and email.

git config --global user.name "Your Name"
git config --global user.email youremail@example.com

Step 2: Clone a GitHub Repository

Cloning a repository allows you to create a copy of a project on your local machine, enabling you to work on it.

4. **Find a Repository to Clone**: Visit the repository page on GitHub (e.g., https://github.com/username/repository) and click the green **Code** button.
5. **Copy the Clone URL**: In the popup, choose either **HTTPS** or **SSH** and copy the URL. If you're using HTTPS, it will look like `https://github.com/username/repository.git`.
6. **Clone the Repository Locally**:

   Open a terminal on your computer and navigate to the directory where you want to clone the repository. Then run:

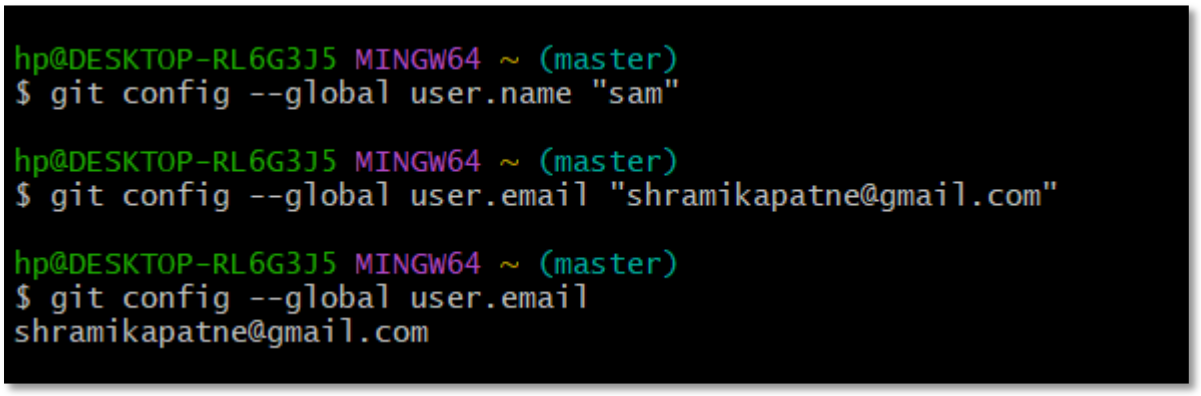
   ```
   git clone https://github.com/username/repository.git
   ```

- Replace `https://github.com/username/repository.git` with the URL you copied.

This will create a local copy of the repository on your machine.

- **Navigate to the Repository Folder**:

cd repository # Navigate into the cloned directory



```
hp@DESKTOP-RL6G3J5 MINGW64 ~ (master)
$ cd "C:\Users\newgit"

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ git clone "https://github.com/Shramikapatne20/myrepo_sam79.git"
fatal: destination path 'myrepo_sam79' already exists and is not an empty direct
ory.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit (master)
$ cd myrepo_sam79
```

Step 3: Work on the Project Locally

Once you've cloned the repository, you can start making changes to the code.

2. **Create a New Branch**: Before making changes, it's recommended to create a new branch. This ensures your changes don't interfere with the main codebase until you're ready to merge.

```
git checkout -b feature/your-feature  # Create and switch to a new
```

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git checkout new_branch
Switched to branch 'new_branch'

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git add .

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git commit -m "Added some text"
[new_branch 749e265] Added some text
 1 file changed, 1 insertion(+)
```

4. **Make Changes**: Edit files as needed using your preferred editor or IDE.
5. **Stage Changes**: After making changes, you need to stage them before committing.

   ```
   git add .  # Stages all modified files
   ```

6. **Commit Changes**: Once changes are staged, commit them to your local branch.

   ```
   git commit -m "Add feature X"
   ```

Step 4: Push Changes to GitHub

Once you've committed your changes locally, you need to push them to your GitHub repository.

Push Your Changes to the remote repository:

```
git push origin feature/your-feature  # Push the feature branch to GitHub
```

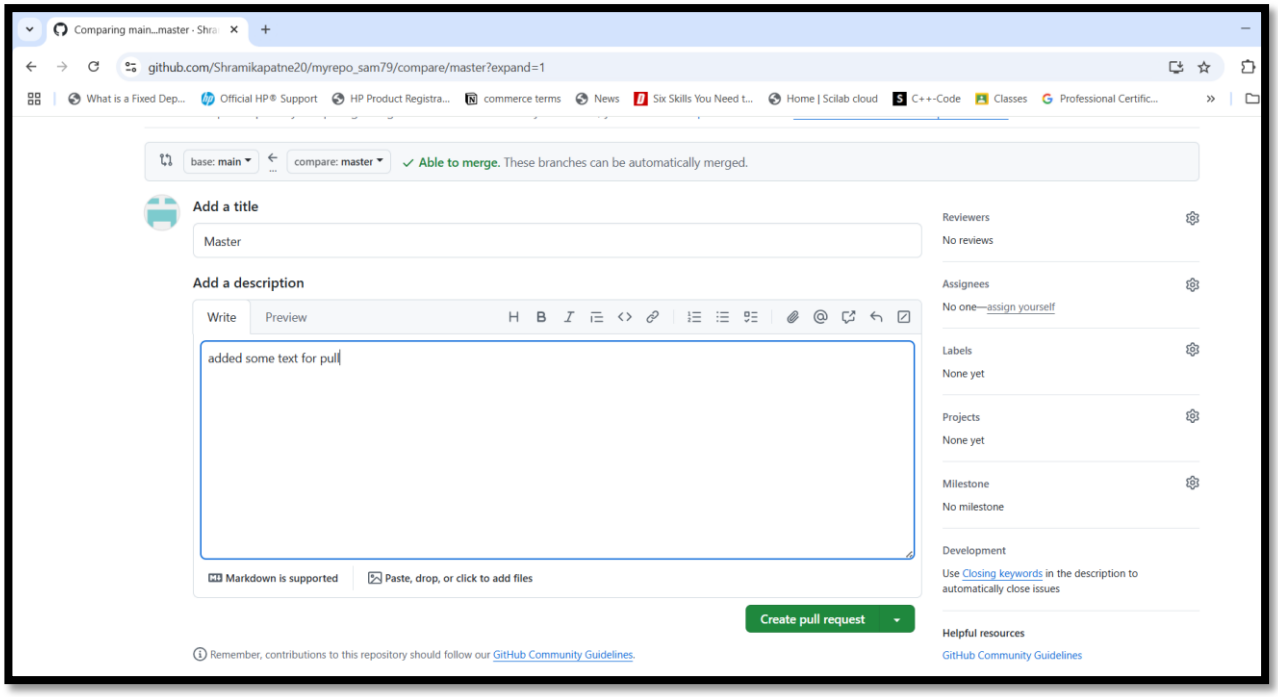
This uploads your local changes to your GitHub repository.

```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git push origin master
Everything up-to-date
```

Step 5: Create a Pull Request (PR)

To contribute your changes back to the original repository, you'll need to open a pull request (PR).

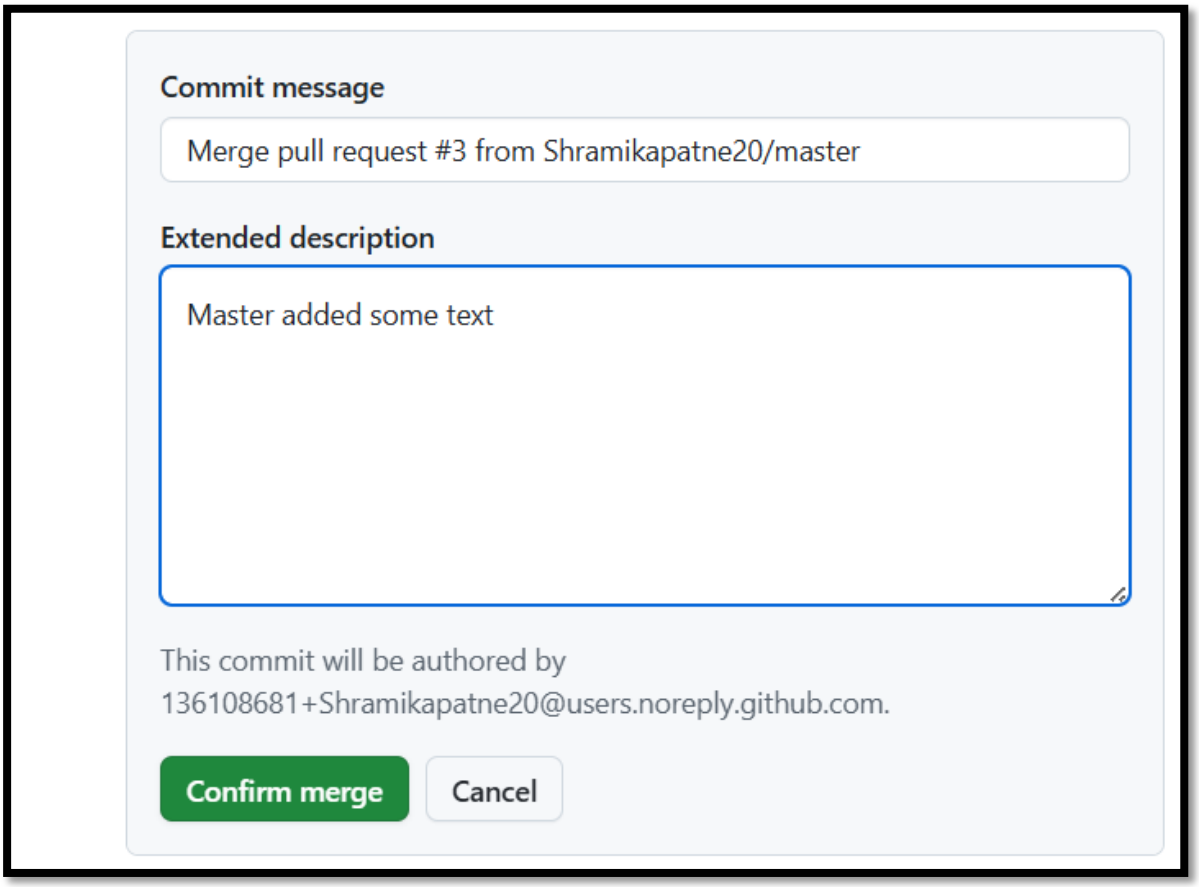
6. **Go to the GitHub Repository**: Visit the repository where you want to make the changes.
7. **Create a Pull Request**: On GitHub, you'll see an option to compare your branch (e.g., `feature/your-feature`) with the main branch of the repository (e.g., `main`). Click **New Pull Request**.
8. **Fill Out the Pull Request Form**:
   - Add a **title** and **description** explaining the changes you've made.

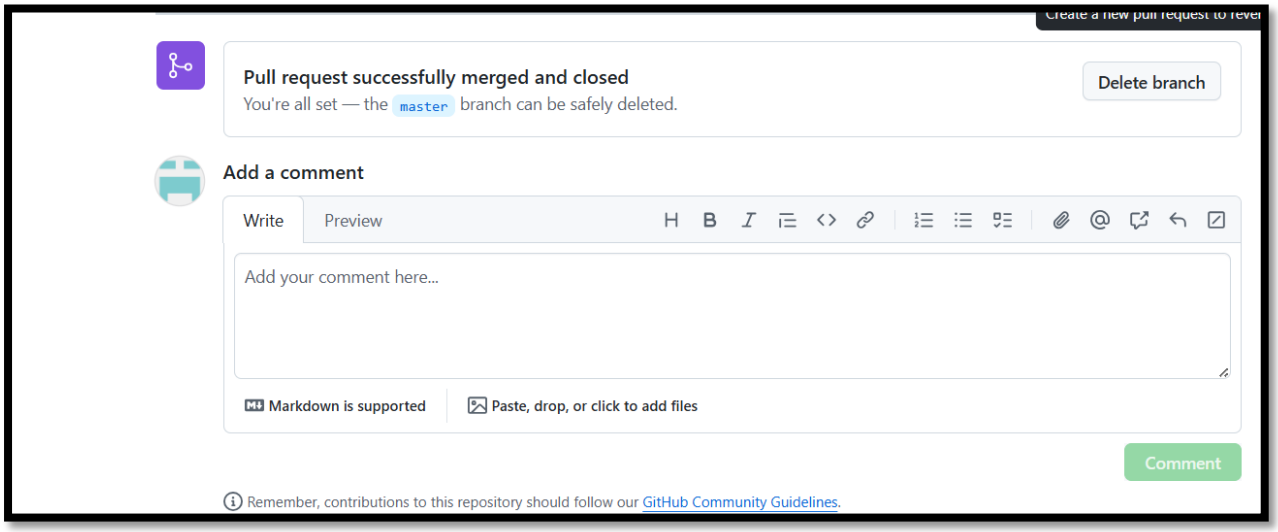


   - Review your changes.
   - Click **Create Pull Request**.

9. **Code Review**: The repository maintainer (or other collaborators) will review your changes. They might ask for changes or approve the PR.

10. **Merge the Pull Request**: Once your changes are approved, the maintainer will merge your changes into the main branch of the project.





Step 6: Sync Your Fork (If Working on a Forked Repo)

If you are working on a forked repository and want to keep your fork in sync with the original repository:

5. **Add the Original Repository as a Remote**: This allows you to fetch updates from the original repository.

   ```
   git remote add upstream https://github.com/owner/original-repository.git
   ```

   Replace `https://github.com/owner/original-repository.git` with the original repository's URL.

6. **Fetch the Latest Changes from the Original Repository**:

```
git fetch upstream  # Fetch the changes from the original repo
```

7. **Merge the Latest Changes into Your Local Branch**:

```
git checkout main  # Switch to your main branch
git merge upstream/main  # Merge the latest changes from the original
repo
```

8. **Push the Changes to Your Fork**:

```
git push origin main  # Push the updated main branch to your fork
```

Step 7: Pull Latest Changes from the Original Repository

To keep your local repository up-to-date with the remote repository on GitHub, you can pull the latest changes.

3. **Switch to Your Local Main Branch**:

```
git checkout main
```

4. **Pull Latest Changes from GitHub**:

```
git pull origin main  # Pull the latest changes from the remote
repository
```

Step 8: Collaborate with Other Developers

When collaborating with other developers on GitHub, you'll typically follow these best practices:

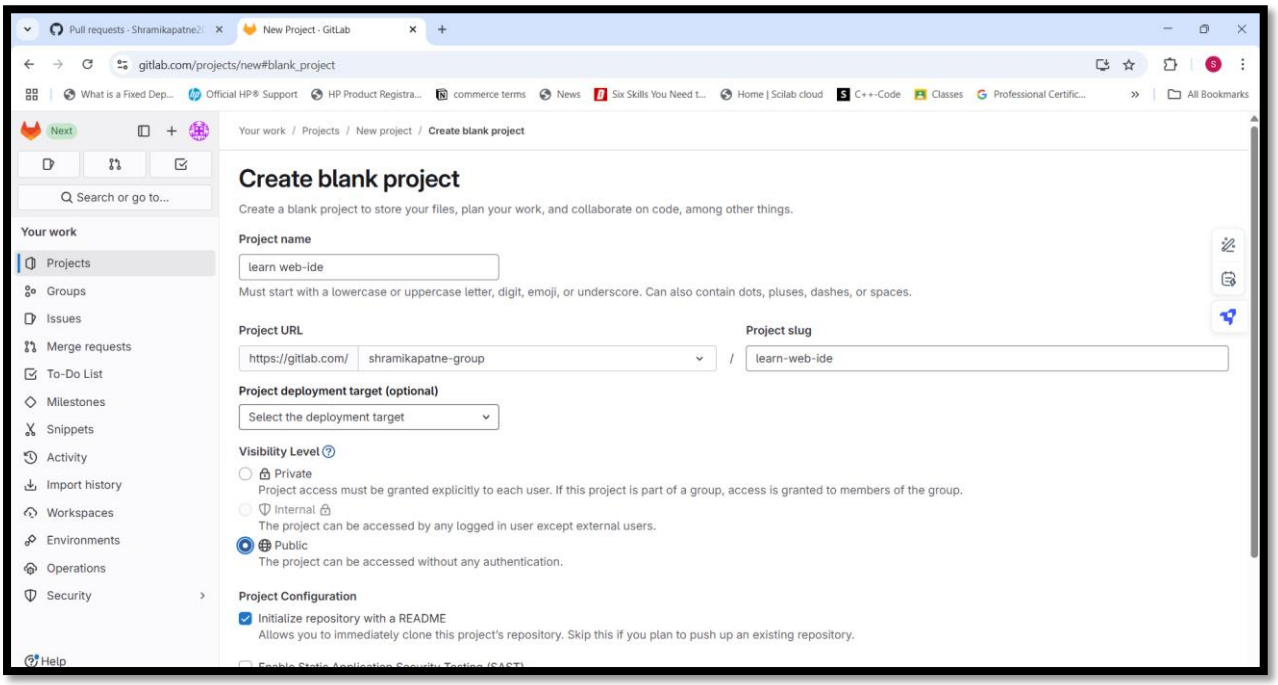
5. **Regularly Pull Latest Changes**: To ensure you're not working on outdated code, frequently pull the latest changes from the main branch (especially before you start working on new features or bug fixes).
6. **Create Feature Branches**: Always create a new branch for each feature or bug fix. This avoids conflicts and keeps the history clean.
7. **Review Pull Requests**: If you're reviewing someone else's PR, ensure you provide feedback and approve it once you're satisfied.
8. **Resolve Merge Conflicts**: If two developers edit the same part of a file, a merge conflict will occur when merging. Resolve these conflicts manually by editing the files and then committing the changes.

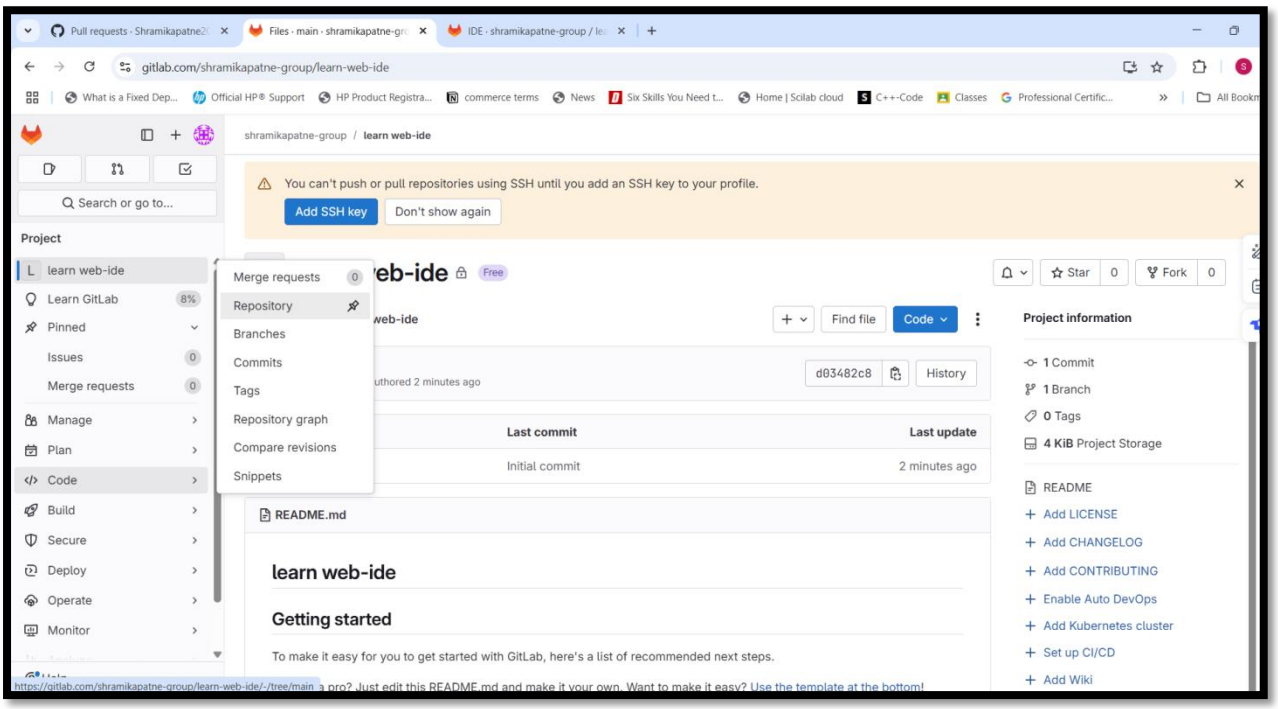
```
hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git remote add upstream https://github.com/snehalparab27/demo.git

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git fetch upstream
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 24 (delta 1), reused 5 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (24/24), 8.28 KiB | 184.00 KiB/s, done.
From https://github.com/snehalparab27/demo
 * [new branch]      branch1    -> upstream/branch1
 * [new branch]      main       -> upstream/main

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (new_branch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git push origin master
Everything up-to-date

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (master)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

hp@DESKTOP-RL6G3J5 MINGW64 /c/Users/newgit/myrepo_sam79 (main)
$ git pull origin master
From https://github.com/Shramikapatne20/myrepo_sam79
 * branch            master     -> FETCH_HEAD
Updating 23cf05c..f756b20
Fast-forward
 hello.txt  | 2 +-
 myfile.txt | 1 +
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 myfile.txt
```

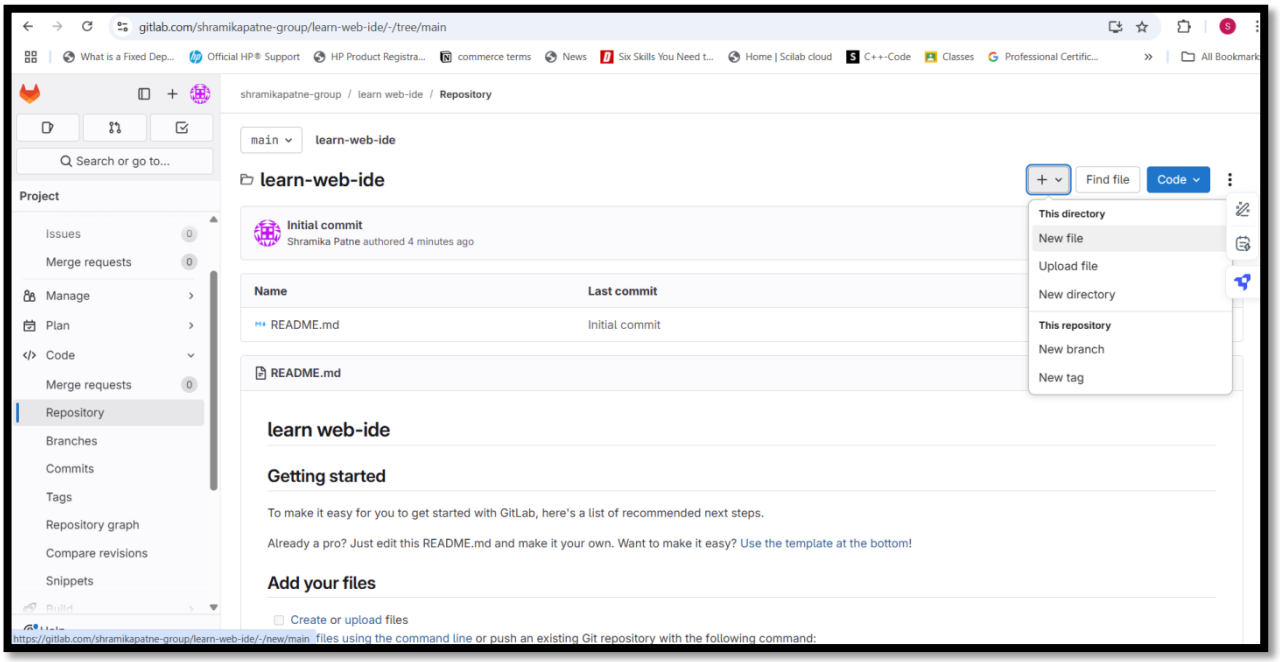
Practical 5
AIM: Using GitLab Web IDE

Steps:

1. Sign up at https://gitlab.com
2. Create a project.
3. Click on Web IDE in your repository.

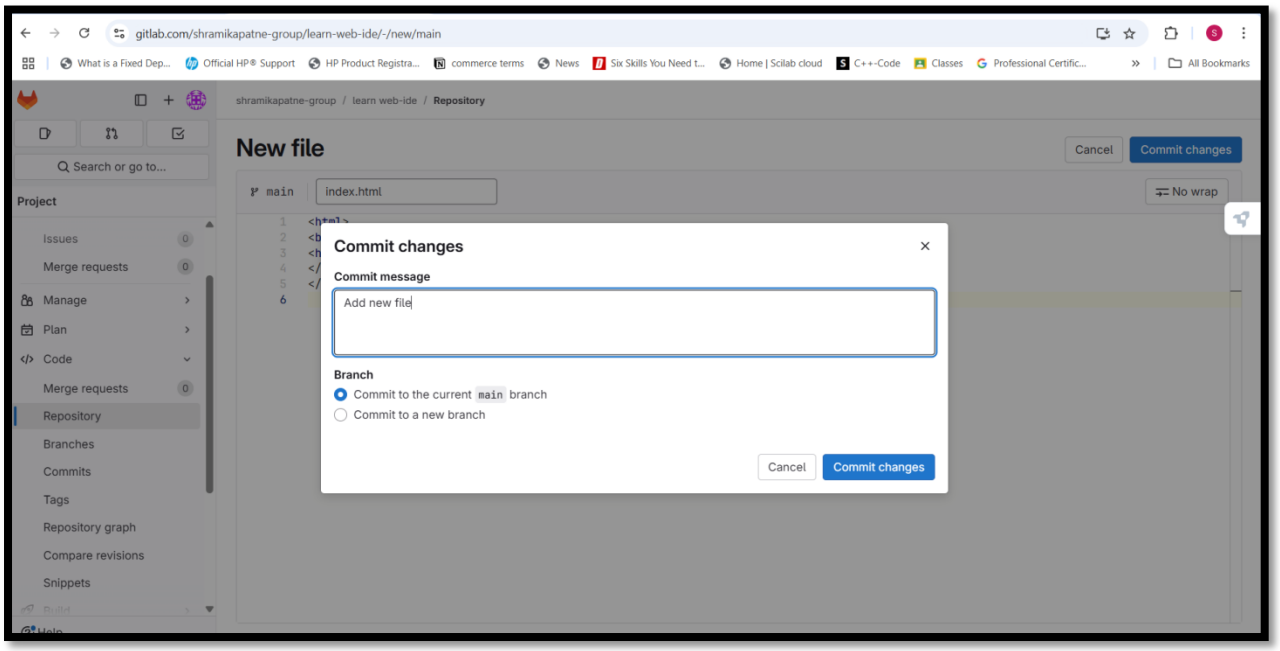




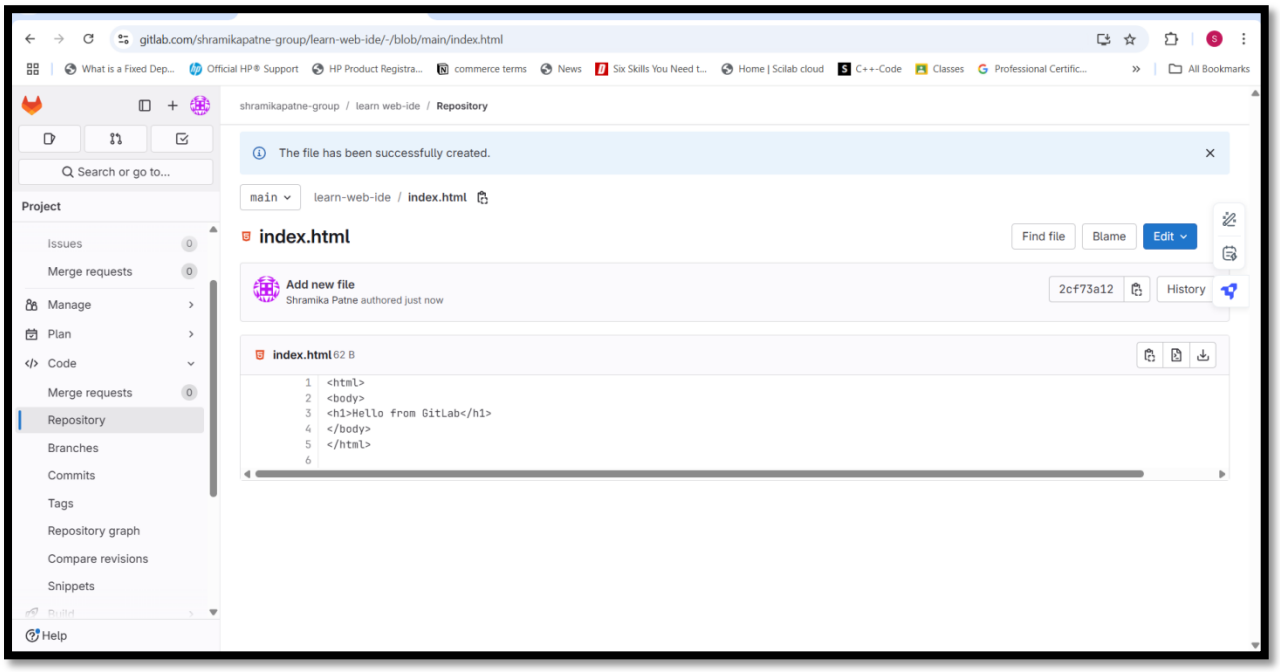


4. Create a file (index.html):

```
<html>
 <body>
   <h1>Hello from GitLab</h1>
 </body>
</html>
```



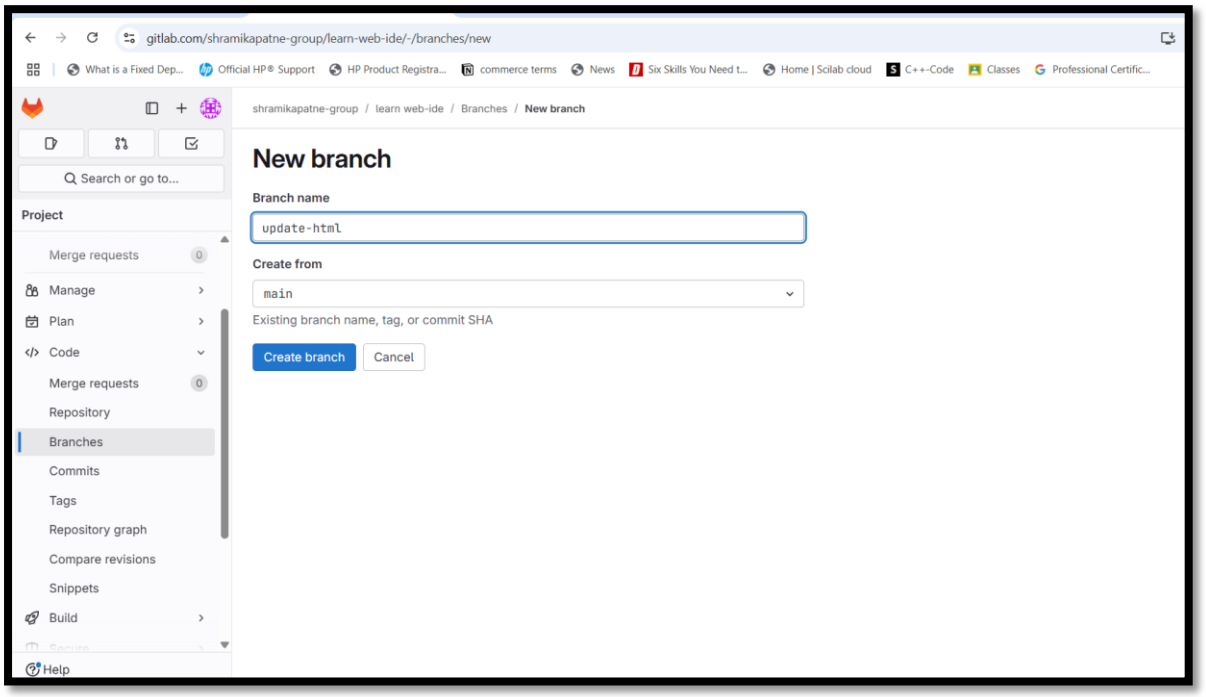
4. Click Commit and push changes.



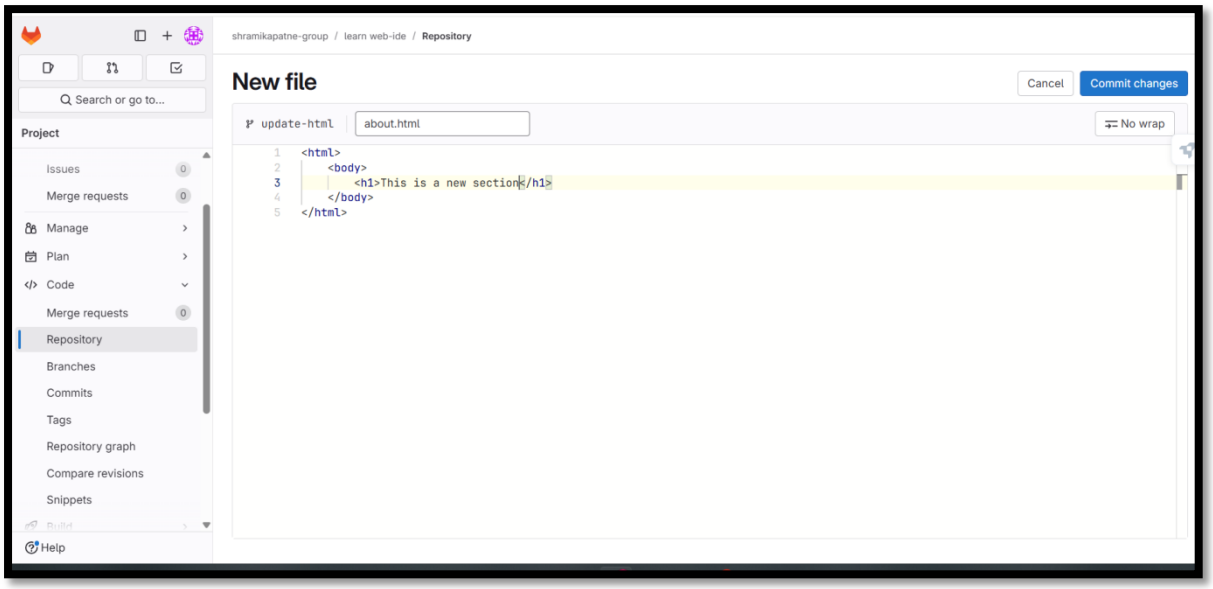
Practical 6

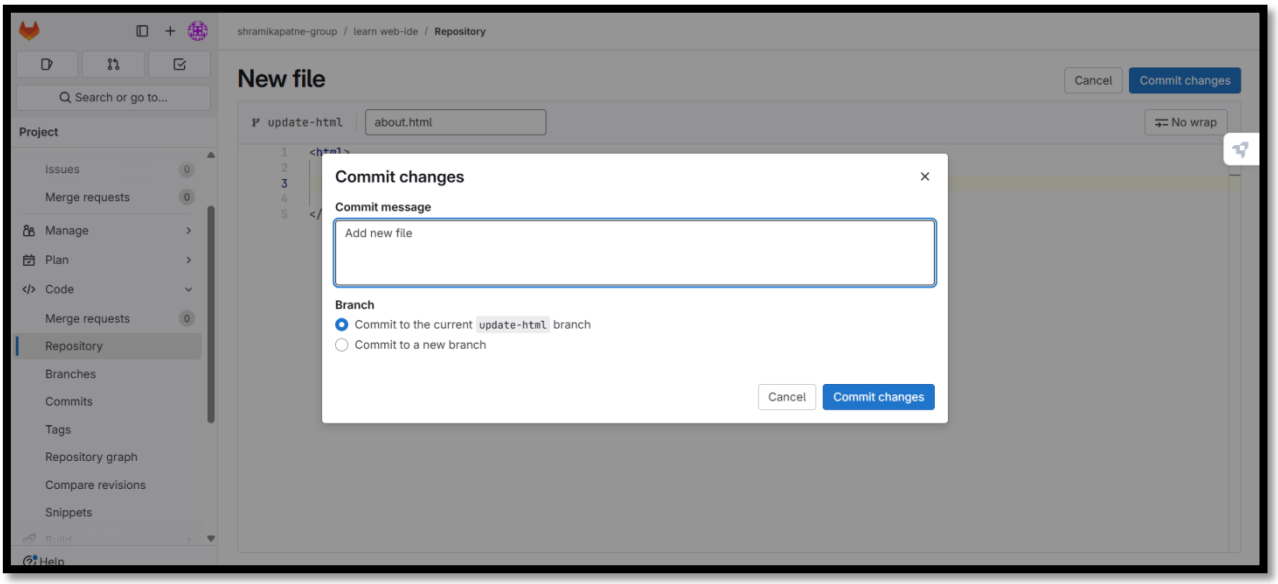
AIM: Performing Merge request using GitLab

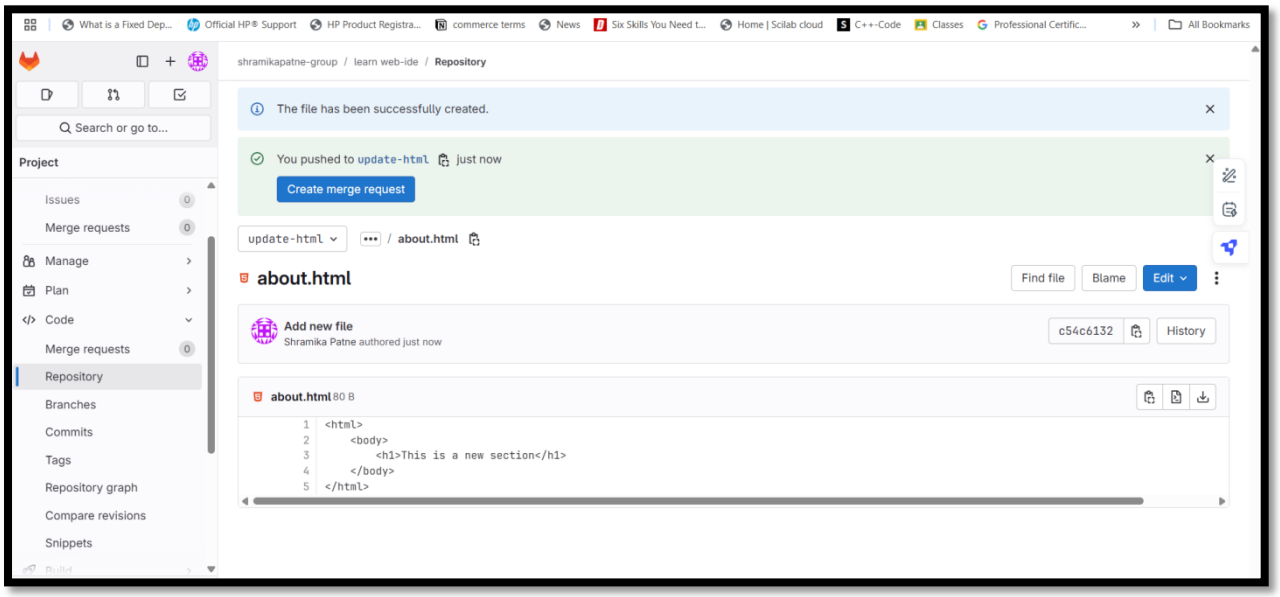
1. Create a new branch in Web IDE.



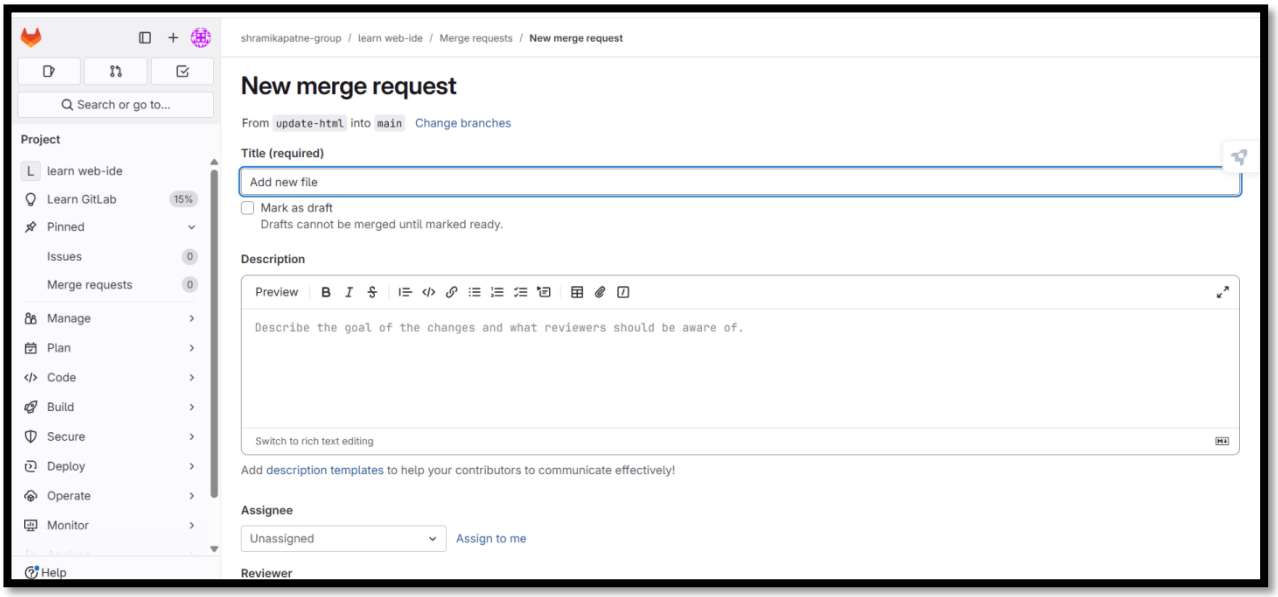
2. Add/edit a file and commit.

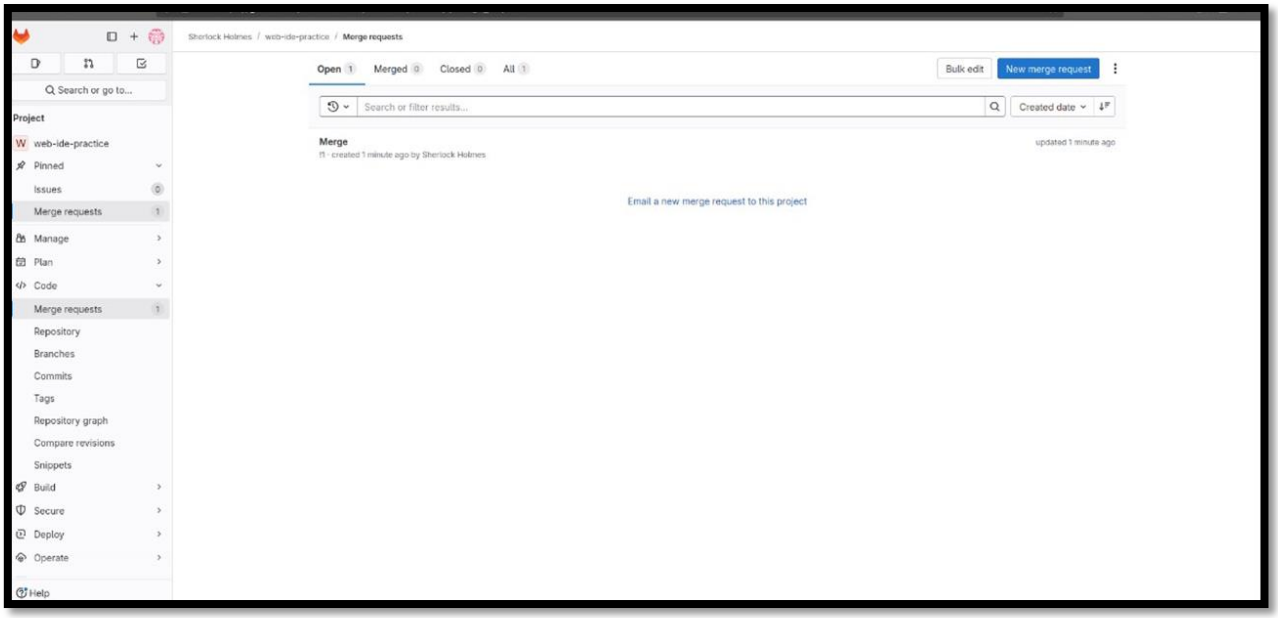



shramikapatne-group / learn web-ide / Repository
New file
Cancel
Commit changes
update-html
about.html
No wrap
Commit changes
Commit message
Add new file
Branch
Commit to the current update-html branch
Commit to a new branch
Cancel
Commit changes
Project
Issues
Merge requests
Manage
Plan
Code
Merge requests
Repository
Branches
Commits
Tags
Repository graph
Compare revisions
Snippets

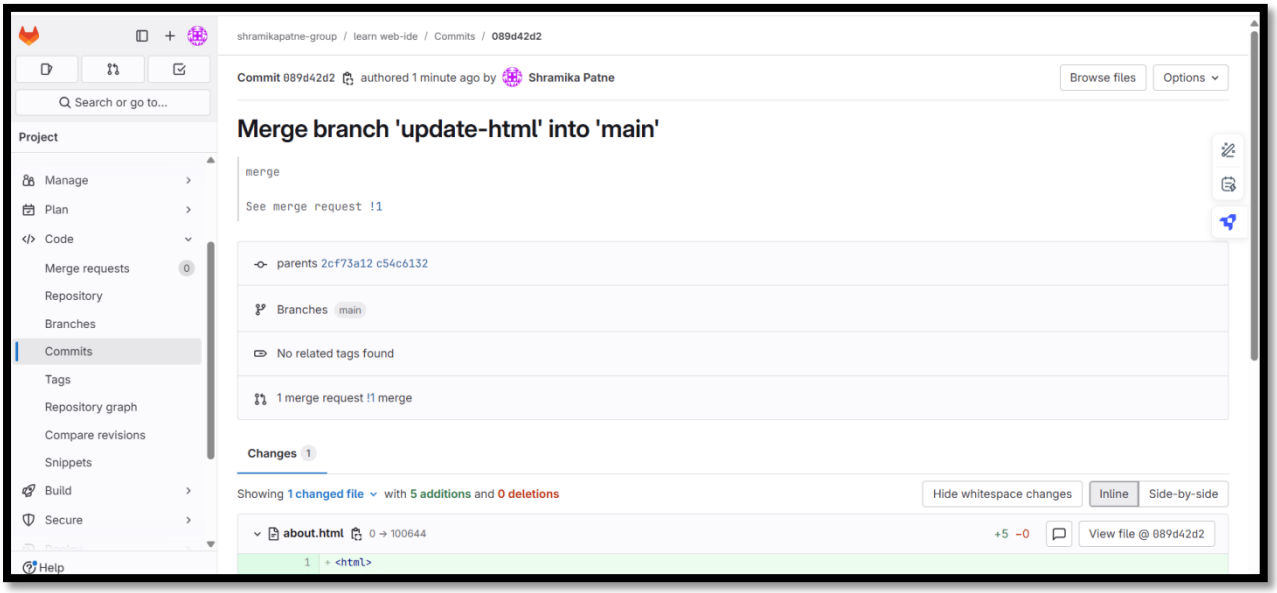

shramikapatne-group / learn web-ide / Repository
The file has been successfully created.
You pushed to update-html just now
Create merge request
update-html / about.html
about.html
Find file
Blame
Edit
Add new file
Shramika Patne authored just now
c54c6132
History
about.html 80 B
<html>
<body>
<h1>This is a new section</h1>
</body>
</html>
Project
Issues
Merge requests
Manage
Plan
Code
Merge requests
Repository
Branches
Commits
Tags
Repository graph
Compare revisions
Snippets

3. Click on Merge Requests > New Merge Request.





4. Select source and target branches.
5. Submit and merge after review.

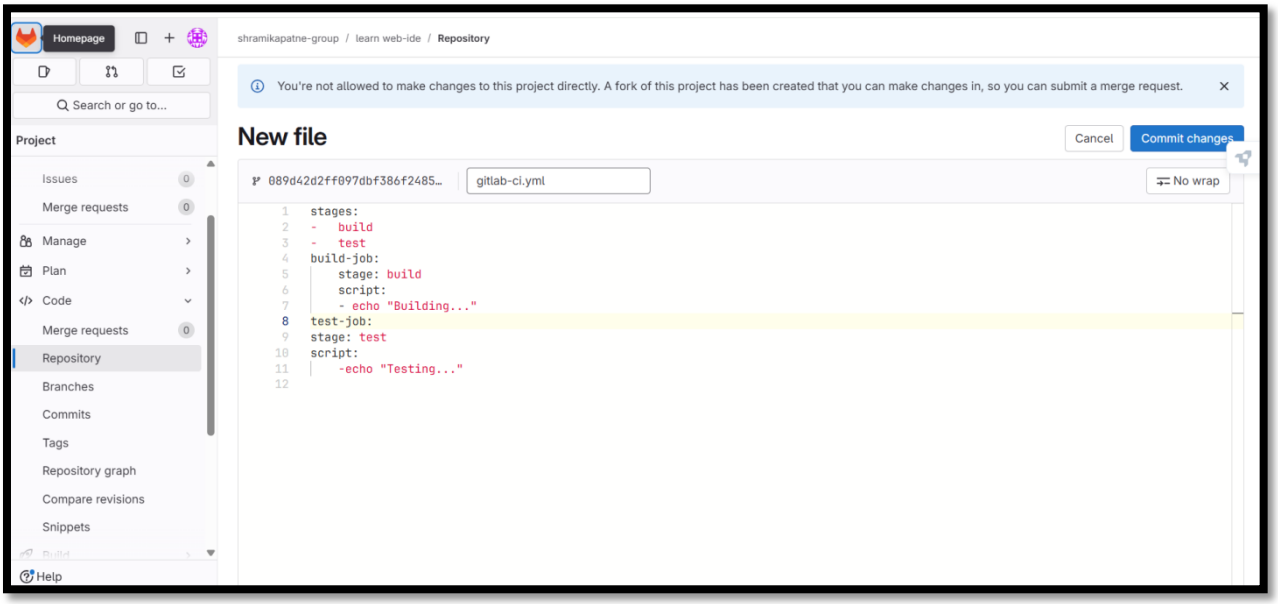


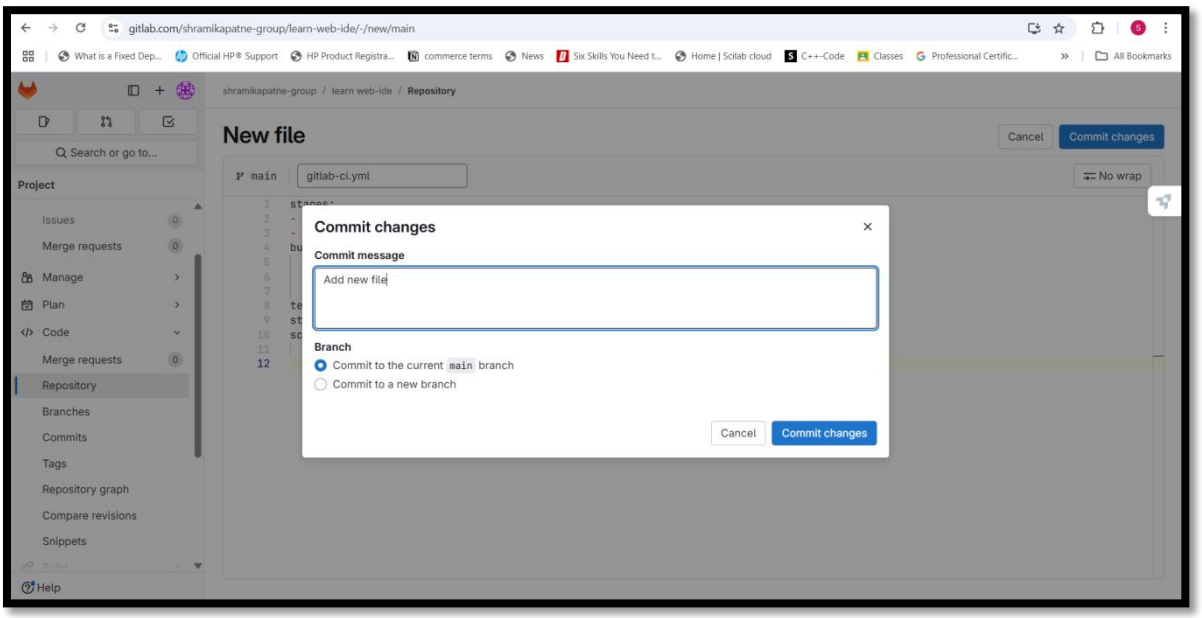
Practical 7
Aim: Workflow Management in GitLab

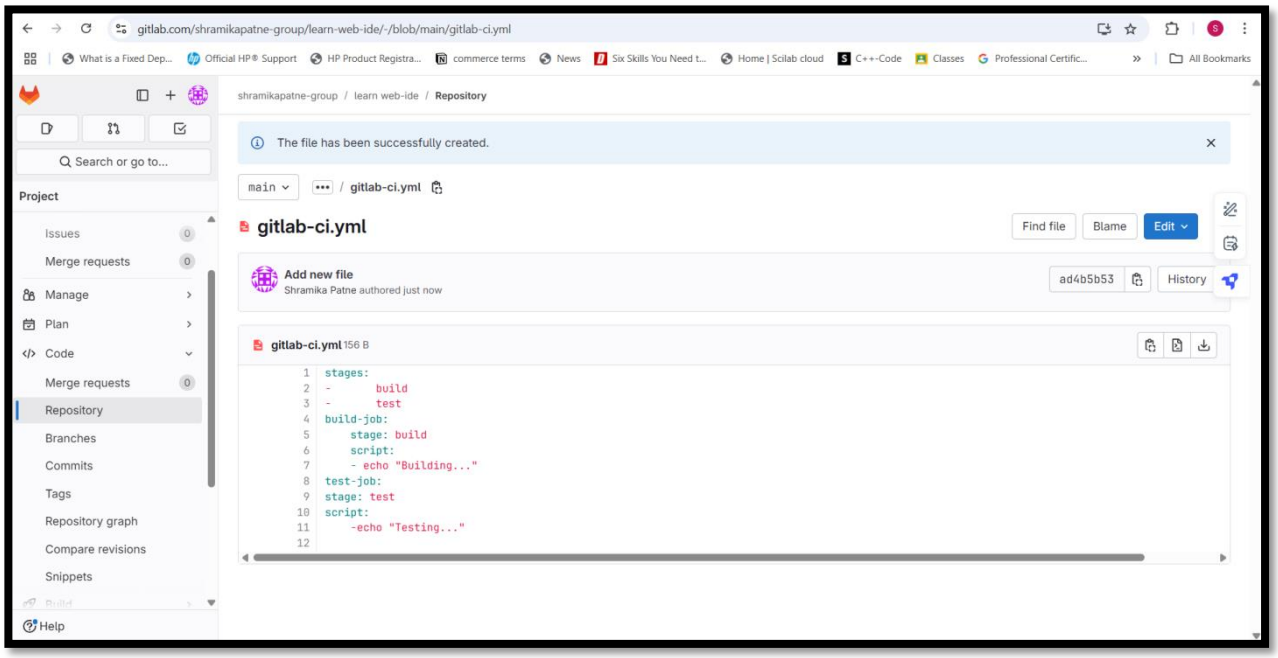
Steps:

1. In your repo, create .gitlab-ci.yml:
   stages:
   - build
   - test
   build-job:
   stage:
   build
   script:
   - echo "Building..." test-job:
   stage: test
   script:
   - echo "Testing..."

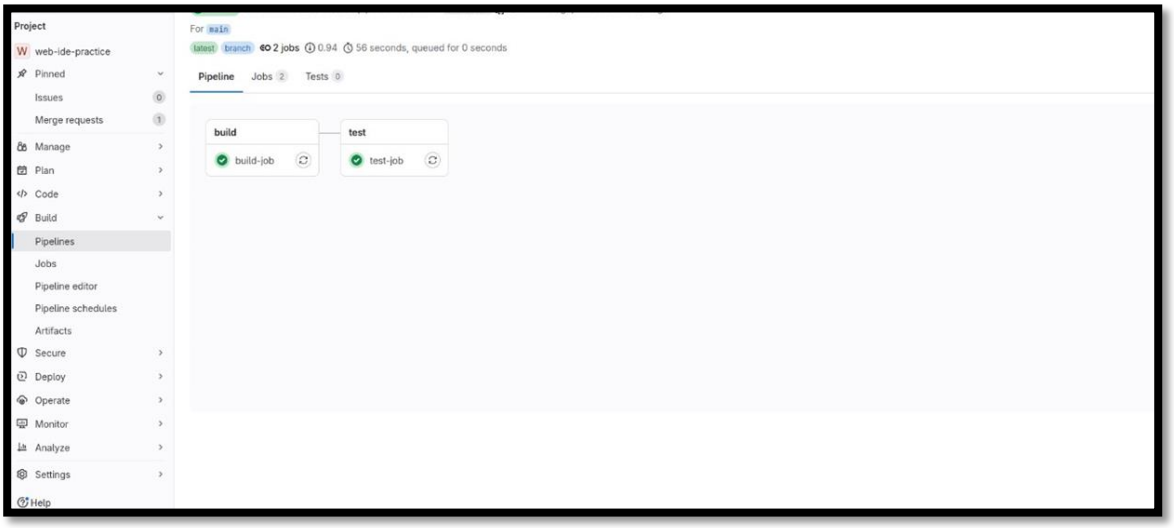
2. Commit and push.

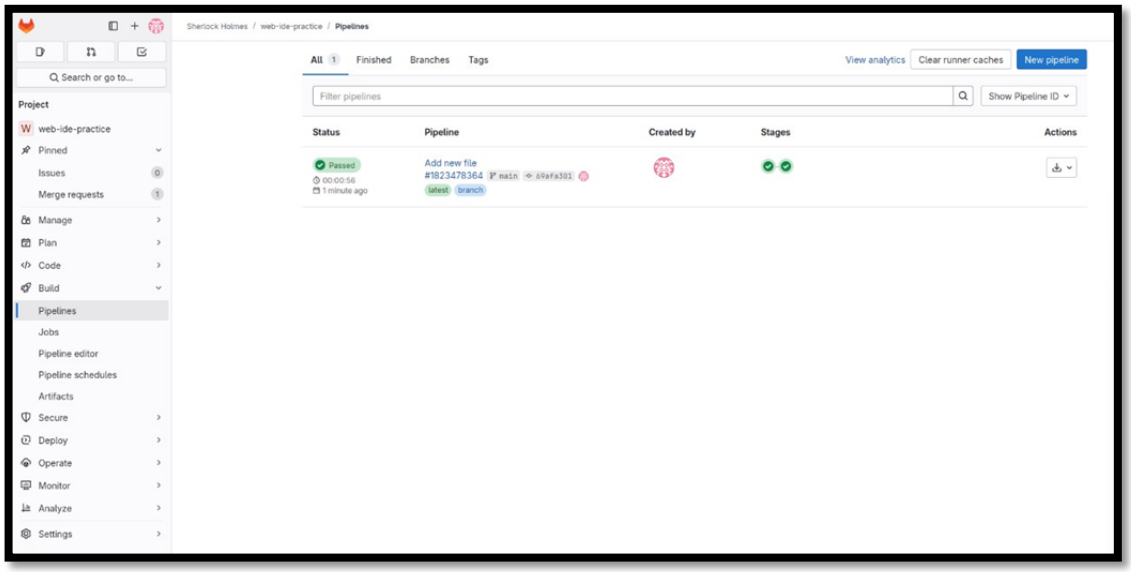






3. Go to CI/CD > Pipelines and view the build/test stages.


Project
web-ide-practice
Pinned
Issues
Merge requests
Manage
Plan
Code
Build
Pipelines
Jobs
Pipeline editor
Pipeline schedules
Artifacts
Secure
Deploy
Operate
Monitor
Analyze
Settings
Help
For main
latest branch 2 jobs 0.94 56 seconds, queued for 0 seconds
Pipeline Jobs 2 Tests 0
build
build-job
test
test-job

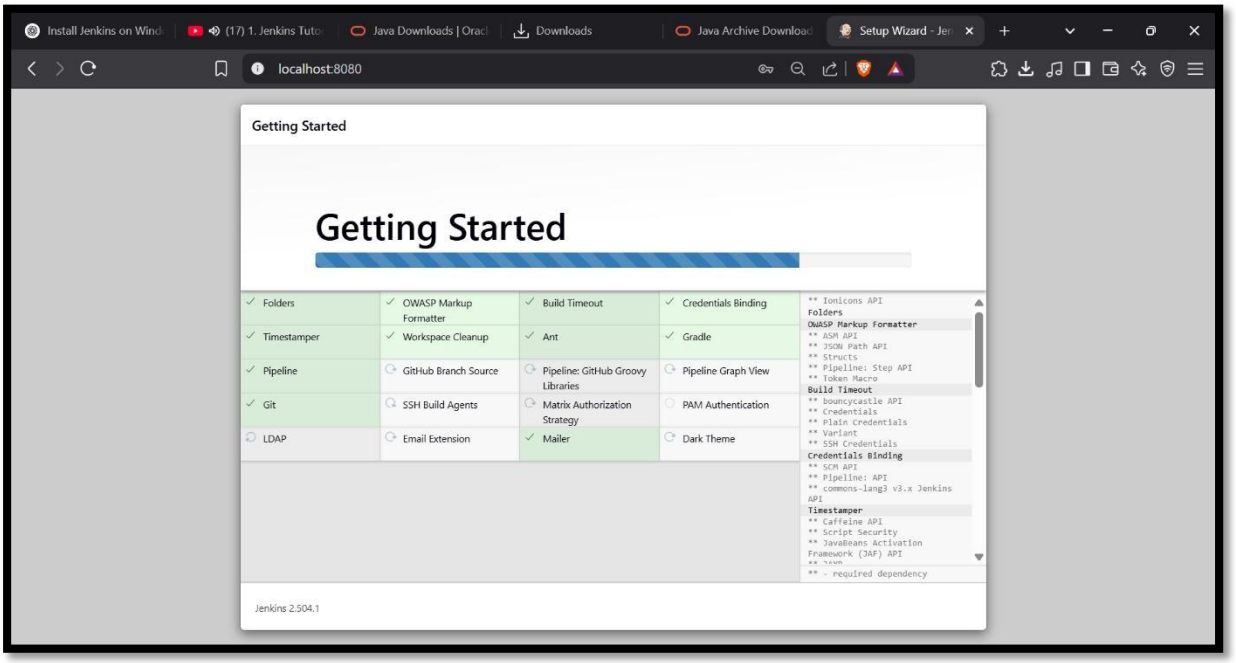

Sherlock Holmes / web-ide-practice / Pipelines
All 1 Finished Branches Tags
View analytics Clear runner caches New pipeline
Filter pipelines
Show Pipeline ID
Status Pipeline Created by Stages Actions
Passed
00:00:56
1 minute ago
Add new file
#1823478364 main 69afa301
latest branch
Search or go to...
Project
web-ide-practice
Pinned
Issues
Merge requests
Manage
Plan
Code
Build
Pipelines
Jobs
Pipeline editor
Pipeline schedules
Artifacts
Secure
Deploy
Operate
Monitor
Analyze
Settings

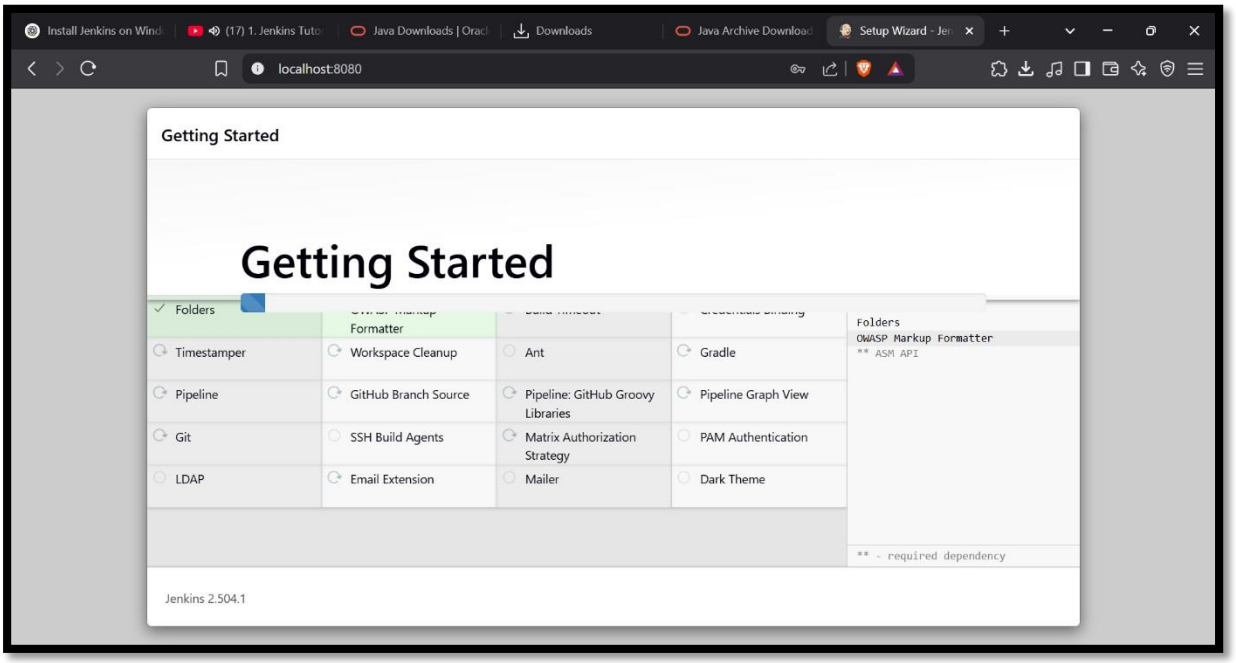
Practical 8

AIM: Demonstrate continuous integration and development using Jenkins

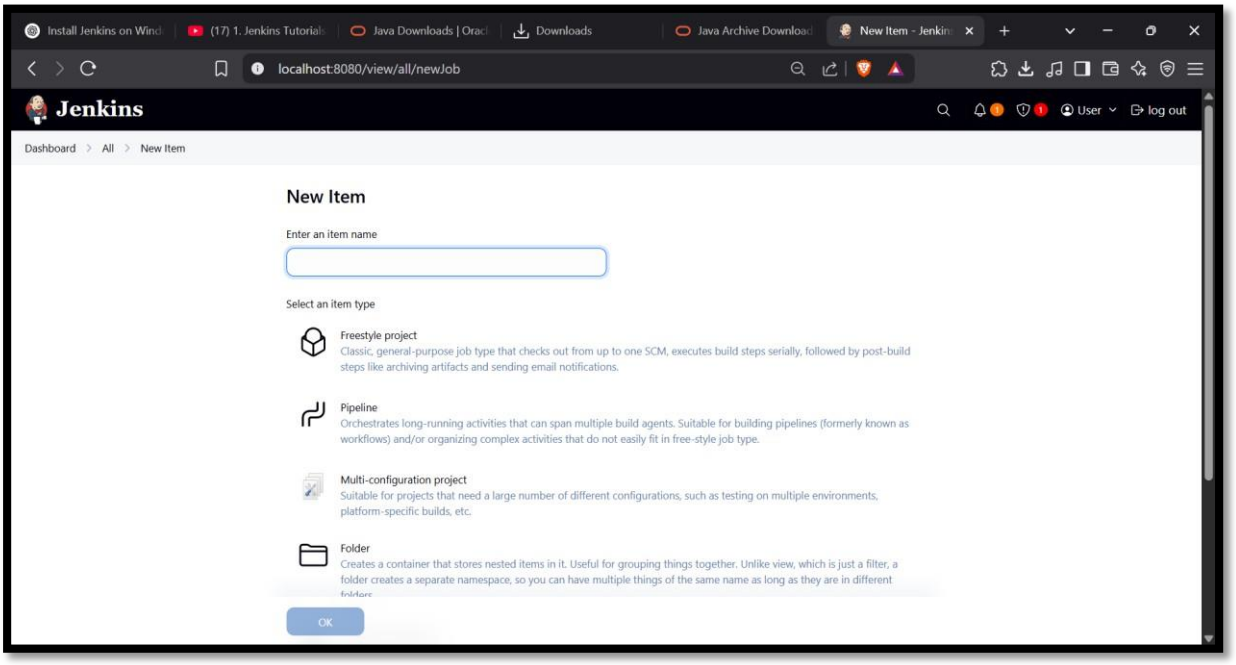
Steps

1. Install Jenkins (visit https://www.jenkins.io)
2. Run Jenkins: http://localhost:8080

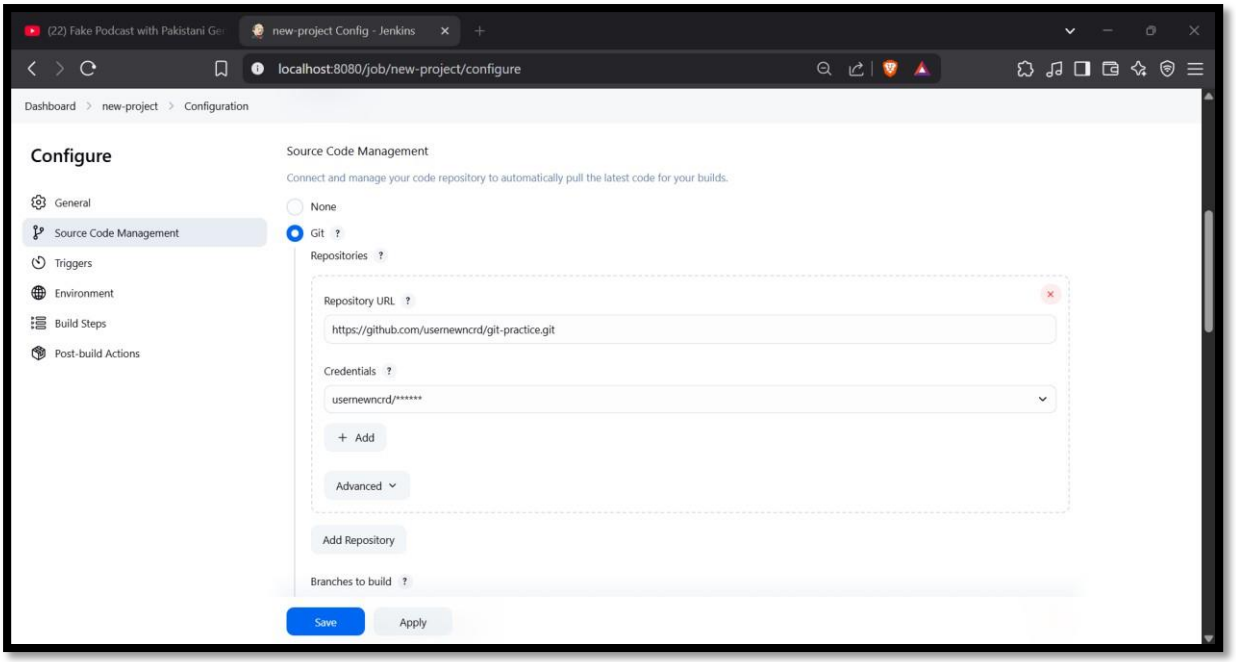




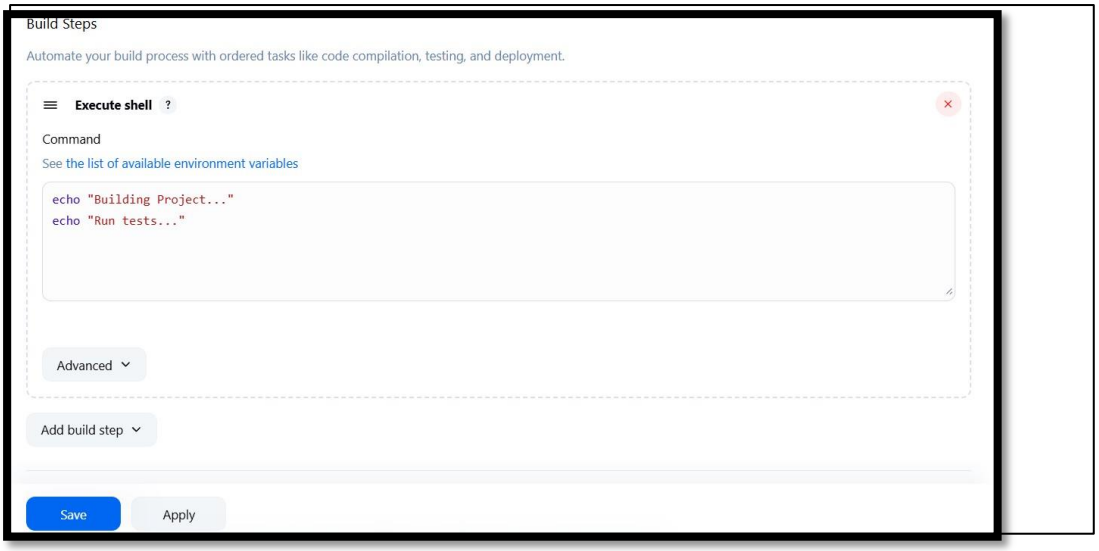
1. Create new Freestyle Project: CI-Demo



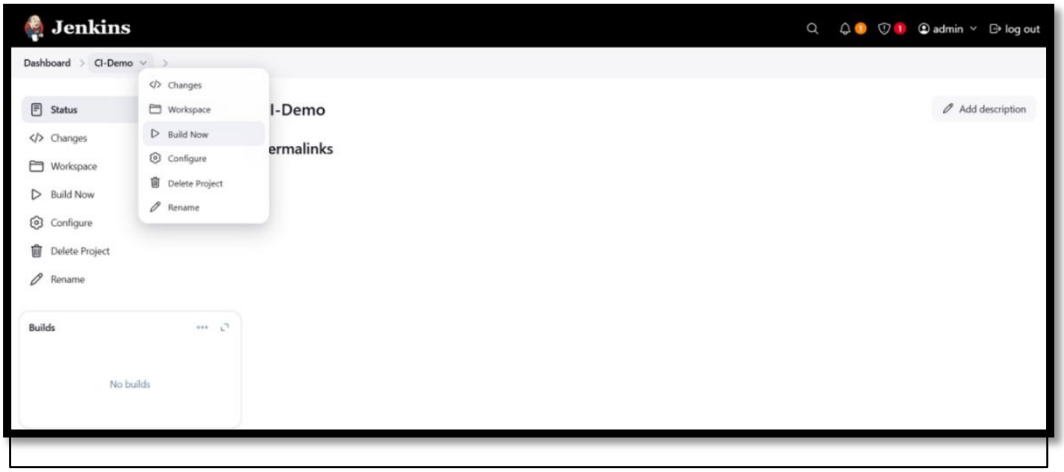
2. Under Source Code Management, choose Git and enter your repo URL.



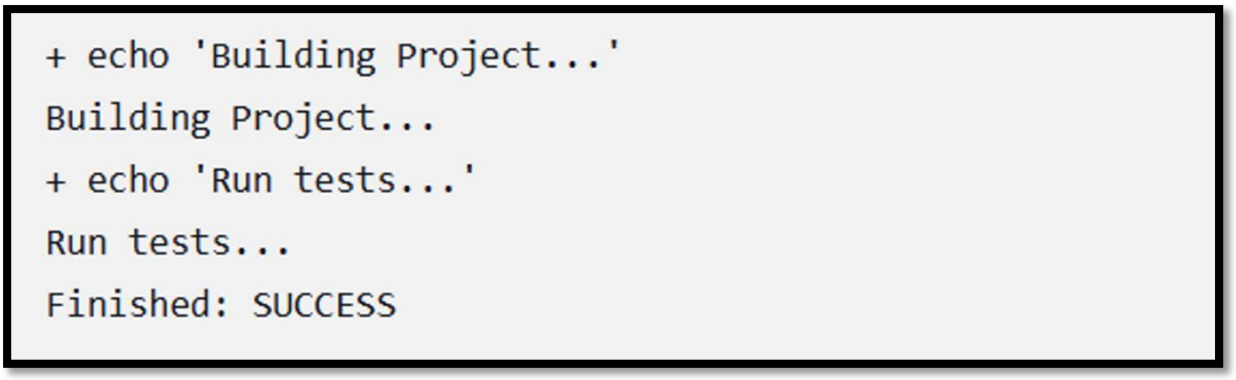
3. Add Build Step > Execute Shell: echo "Building Project..."
   echo "Run tests..."



4. Save and click Build Now.



5. Check output in Console Output.



Practical 9
AIM: Demonstrate basic Docker commands

1. Check Docker version docker --version

```
C:\Users\hp>docker --version
Docker version 28.1.1, build 4eba377
```

2. Pull a Docker image from Docker Hub docker pull nginx

```
C:\Users\hp>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
254e724d7786: Pull complete
d38f2ef2d6f2: Pull complete
40a6e9f4e456: Pull complete
913115292750: Pull complete
3e544d53ce49: Pull complete
4f21ed9ac0c0: Pull complete
d3dc5ec71e9d: Pull complete
Digest: sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

3. List all Docker images docker images

```
C:\Users\hp>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    c15da6c91de8   4 weeks ago   279MB
```

4. Run a container from an image

   docker run -d -p 8080:80 --name mynginx nginx

   This will run the Nginx container and map port 80 (inside the container) to port 8080 (on your host).

```
C:\Users\hp>docker run -d -p 8080:80 --name mynginx nginx
be119c94991613d02f223222f9c1bd038694ab48648d3c1dc38ab8a076e77697
```

5. List all running containers docker ps

```
C:\Users\hp>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                  NAMES
be119c949916   nginx     "/docker-entrypoint.…"   54 seconds ago   Up 54 seconds   0.0.0.0:8080->80/tcp   mynginx
```

6. Copy content from host to container
   docker cp index.html mynginx:/usr/share/nginx/html/
   Replace index.html with your actual file. This copies a file into the running container.

```
C:\Users\hp>docker cp index.html mynginx:/usr/share/nginx/html/
CreateFile C:\Users\hp\index.html: The system cannot find the file specified.
```

7 . Create and use Docker volume for persistent content
docker volume create mydata
   docker run -d -p 8081:80 --name nginx_vol -v mydata:/usr/share/nginx/html nginx
   Now any data added to the /usr/share/nginx/html inside the container will persist even if the container is removed.

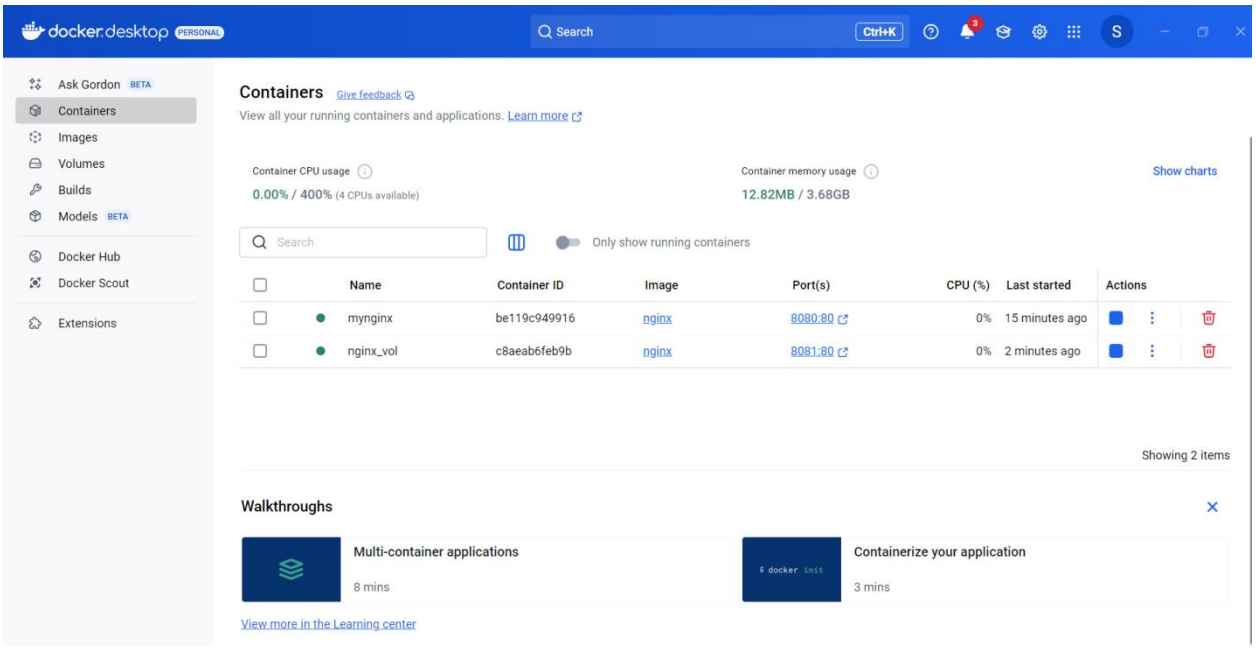
```
C:\Users\hp>docker run -d -p 8081:80 --name nginx_vol -v mydata:/usr/share/nginx/html nginx
c8aeab6feb9b69c022e844c4e7e0998bfa40089e9c3d74f64655d4d8c9f99041
```

```
C:\Users\hp>docker volume create mydata
mydata
```

8 List Docker
volumes docker
volume ls

```
C:\Users\hp>docker volume ls
DRIVER    VOLUME NAME
local     mydata
```

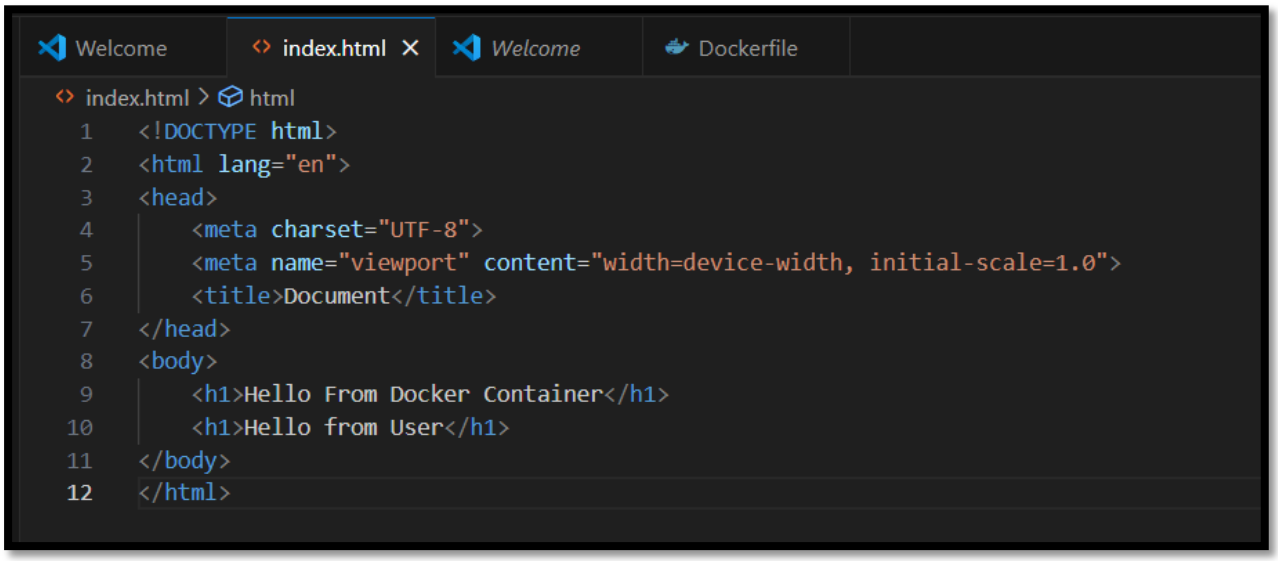
9. Remove a container
   docker rm -f mynginx
   Remove an image
   docker rmi nginx



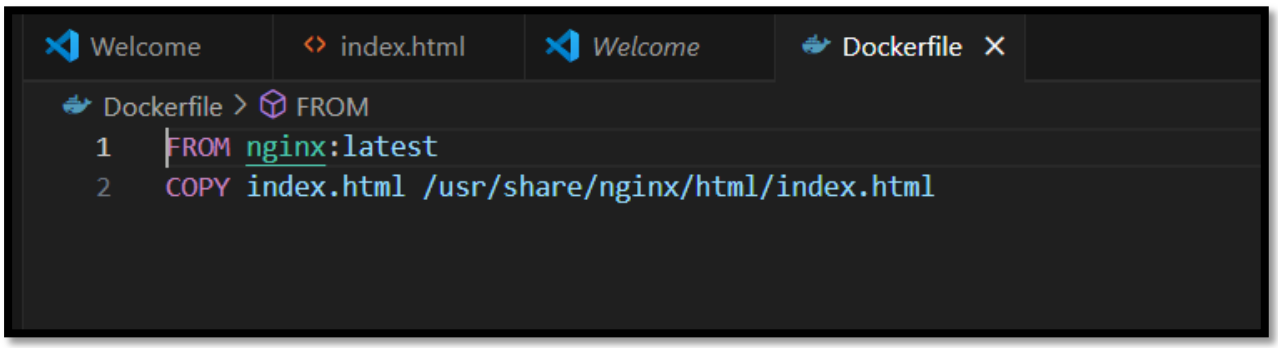
Practical 10
AIM: Develop a simple containerized application using Docker

Develop a Simple Containerized Application using Docker

1. Index.html



2. Dockerfile :-



3. docker build -t my-docker-webapp .

```
PS C:\Users\myapp> code Dockerfile
PS C:\Users\myapp> docker build -t my-docker-webapp .
[+] Building 2.6s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 104B
 => [internal] load metadata for docker.io/library/nginx:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 320B
 => [1/2] FROM docker.io/library/nginx:latest@sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66
 => => resolve docker.io/library/nginx:latest@sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66
 => [auth] library/nginx:pull token for registry-1.docker.io
 => [2/2] COPY index.html /usr/share/nginx/html/index.html
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:0931fdcbc45611ccb959befd0840ebe19bdae276a8549fe234f5e5a371c4be98
 => => exporting config sha256:af297af0407d6db437d84ba0b2862f7884945673ff9f3684cbebfe6a3d81a67b
 => => exporting attestation manifest sha256:42bd5183c690afd8cb7a835d3dbeefc32ce51f47f1e500f223ea7011c57852a5
 => => exporting manifest list sha256:e52dfa33cc70e51ba0dbe044257e359b09f6e579db77e7061a13610f776cddc1
 => => naming to docker.io/library/my-docker-webapp:latest
 => => unpacking to docker.io/library/my-docker-webapp:latest
```

4. docker run -d -p 8080:80 --name webapp-container my-docker-webapp

```
PS C:\Users\myapp> docker run -d -p 8080:80 --name webapp-container my-docker-webapp
6eef9ba52dcde9c4d40a54043711f3b21db18984ad7aa017f60311481b678660
```

5. docker ps

```
PS C:\Users\myapp> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS             PORTS                    NAMES
c8aeab6feb9b   nginx     "/docker-entrypoint.…"   56 minutes ago      Up 56 minutes      0.0.0.0:8081->80/tcp     nginx_vol
be119c949916   nginx     "/docker-entrypoint.…"   About an hour ago   Up About an hour   0.0.0.0:8080->80/tcp     mynginx
```

```
PS C:\Users\myapp> docker stop webapp-container
webapp-container
```

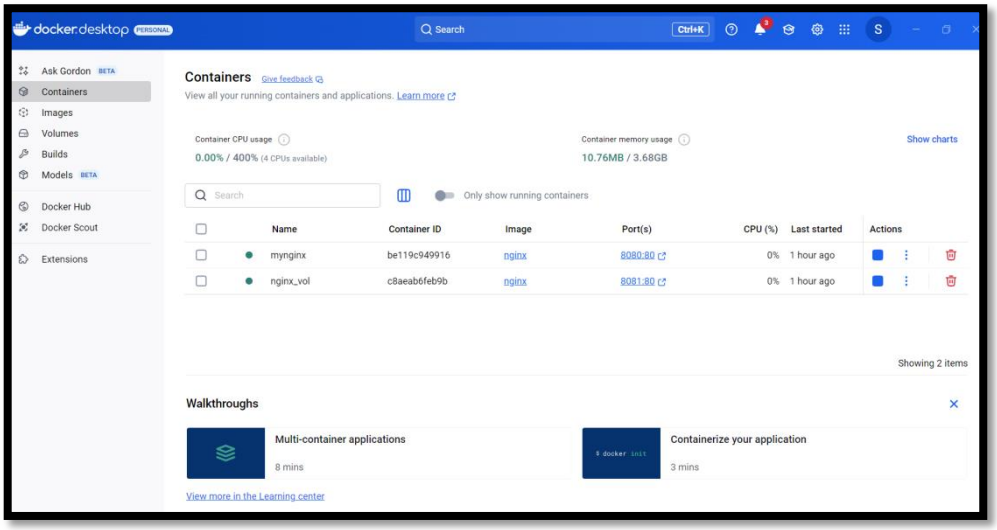
6. docker stop webapp-container

```
PS C:\Users\myapp> docker rmi my-docker-webapp
Untagged: my-docker-webapp:latest
```

7. docker rm webapp-container

8. docker rmi my-docker-webapp

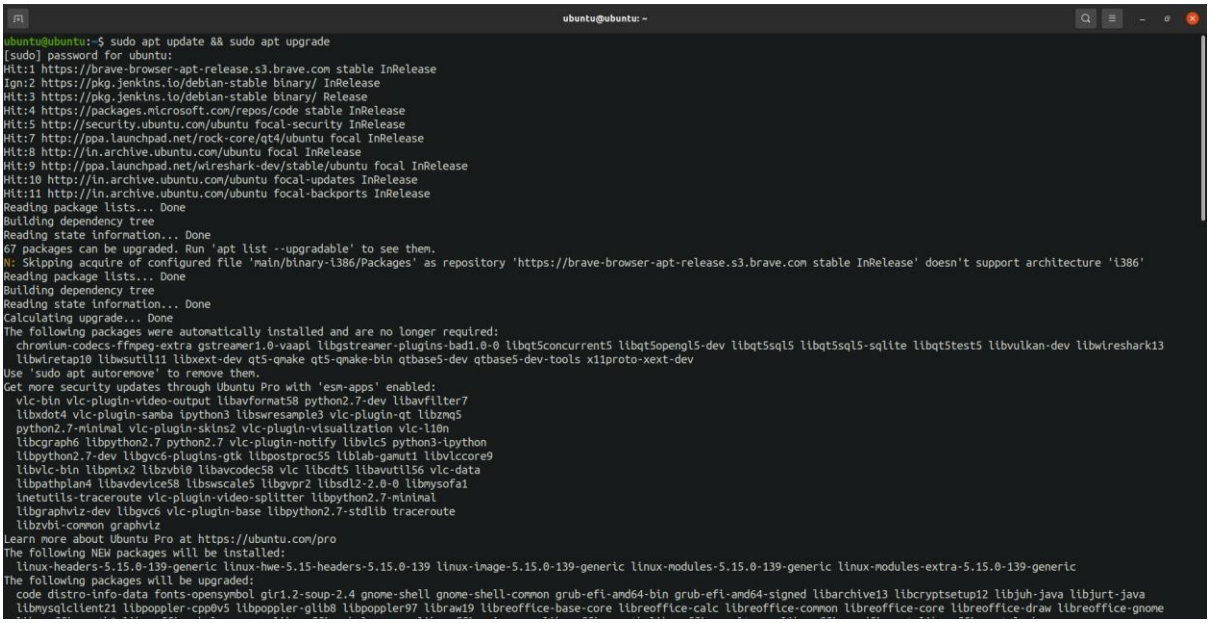
```
PS C:\Users\myapp> docker rmi my-docker-webapp
Untagged: my-docker-webapp:latest
Deleted: sha256:e52dfa33cc70e51ba0dbe044257e359b09f6e579db77e7061a13610f776cddc1
```


docker desktop
Search
Ctrl+K
Ask Gordon BETA
Containers
Images
Volumes
Builds
Models BETA
Docker Hub
Docker Scout
Extensions
Containers
Give feedback
View all your running containers and applications. Learn more
Container CPU usage
0.00% / 400% (4 CPUs available)
Container memory usage
10.76MB / 3.68GB
Show charts
Search
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Last started
Actions
mynginx
be119c949916
nginx
8080:80
0%
1 hour ago
nginx_vol
c8aeab6feb9b
nginx
8081:80
0%
1 hour ago
Showing 2 items
Walkthroughs
Multi-container applications
8 mins
Containerize your application
3 mins
View more in the Learning center

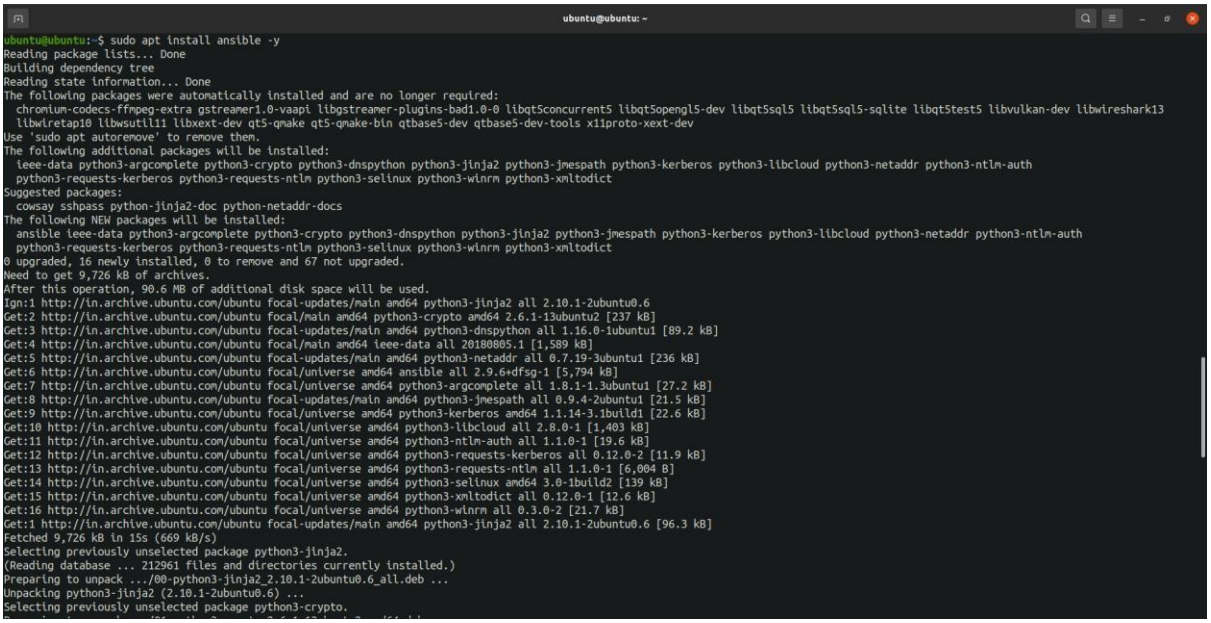

localhost:8080
Hello from Docker Container
Hello From User

Practical 11
AIM: Demonstrate add-on ansible commands

Step 1: Update your VM



Step 2: Install Ansible



Step 3: Check version:


ubuntu@ubuntu:~$ ansible --version
ansible 2.9.6
config file = /etc/ansible/ansible.cfg
configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
ansible python module location = /usr/lib/python3/dist-packages/ansible
executable location = /usr/bin/ansible
python version = 3.8.10 (default, Mar 18 2025, 20:04:55) [GCC 9.4.0]
ubuntu@ubuntu:~$

```
ubuntu@ubuntu:~$ nano host.ini
ubuntu@ubuntu:~$
```

1. Ping the remote host
   ansible local -i host.ini -m ping

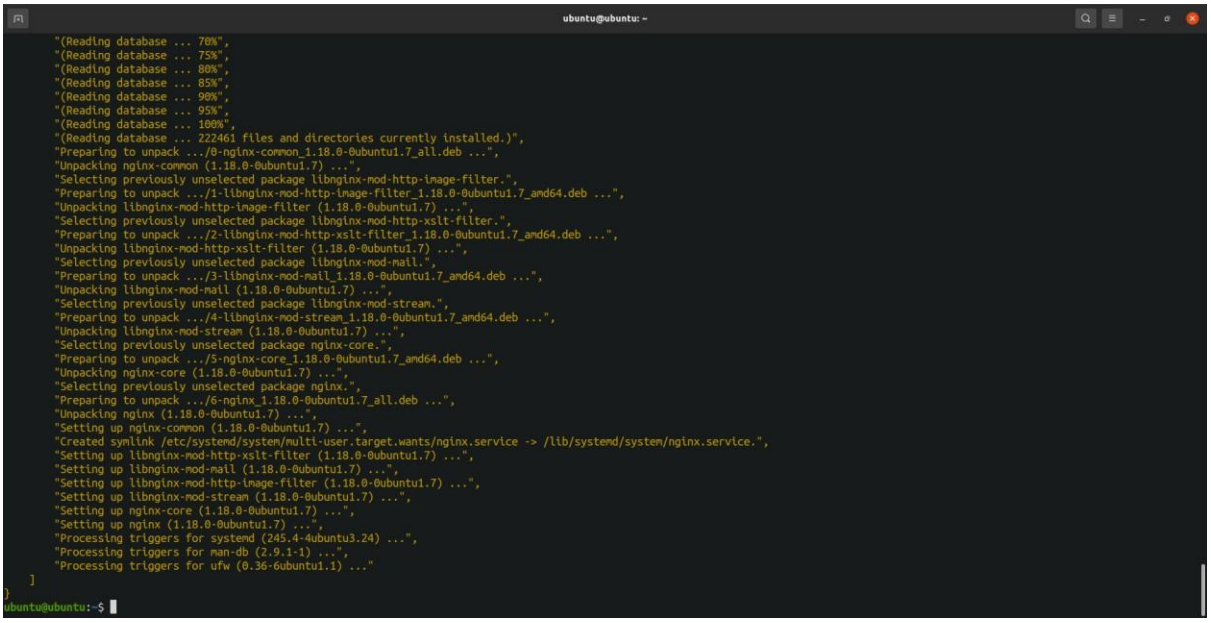
```
ubuntu@ubuntu:~$ ansible local -i host.ini -m ping
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host localhost should use /usr/bin/python3, but is using /usr/bin/python for backward compatibility with prior Ansible releases. A future
Ansible release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information. This feature will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
ubuntu@ubuntu:~$
```

2. Check uptime
   ansible local -i host.ini -a "uptime"

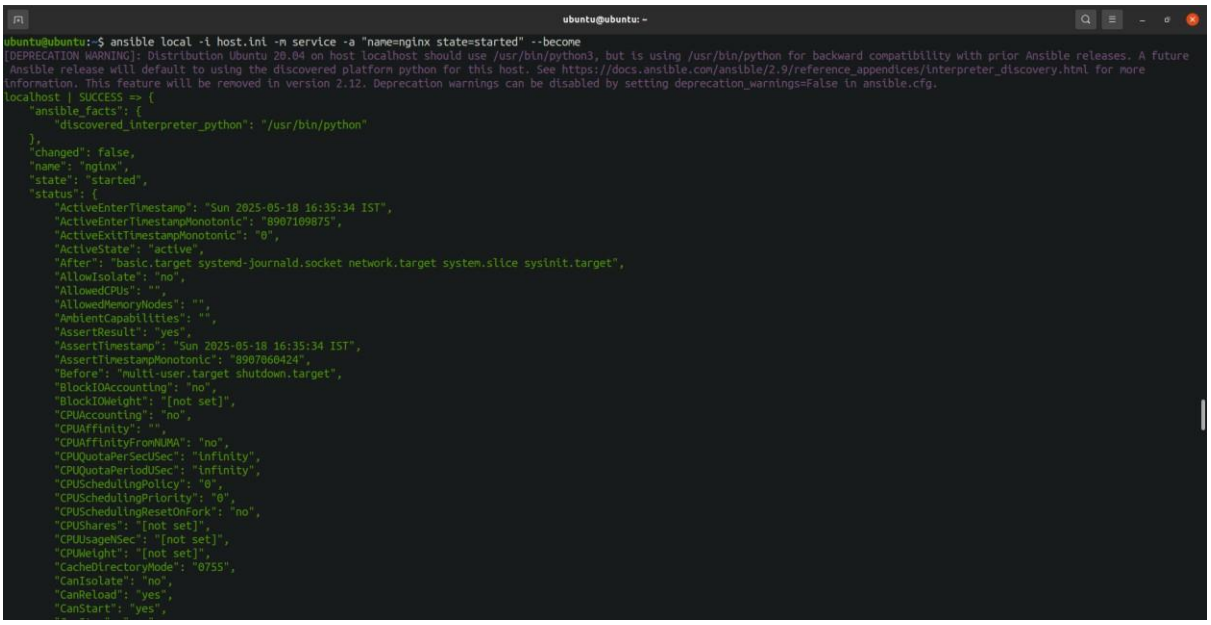
```
ubuntu@ubuntu:~$ ansible local -i host.ini -a "uptime"
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host localhost should use /usr/bin/python3, but is using /usr/bin/python for backward compatibility with prior Ansible releases. A future
Ansible release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information. This feature will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
localhost | CHANGED | rc=0 >>
 16:31:16 up  2:49,  1 user,  load average: 1.08, 0.98, 0.90
ubuntu@ubuntu:~$
```

3. Install a package
   ansible local -i host.ini -m apt -a "name=nginx state=present update_cache=yes" –become

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ ansible local -i host.ini -m apt -a "name=nginx state=present update_cache=yes" --become
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host localhost should use /usr/bin/python3, but is using /usr/bin/python for backward compatibility with prior Ansible releases. A future
Ansible release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information. This feature will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
localhost | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "cache_update_time": 1747566323,
    "cache_updated": true,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following packages were automatically installed and are no longer required:\n  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi\n  libgstreamer-plugins-bad1.0-0 libqt5concurrent5 libqt5opengl5-dev libqt5sql5\n  libqt5sql5-sqlite libqt5test5 libvulkan-dev libwireshark13 libwiretap10\n  libwsutil11 libxext-dev qt5-qmake qt5-qmake-bin qtbase5-dev\n  qtbase5-dev-tools x11proto-xext-dev\nUse 'sudo apt autoremove' to remove them.\nThe following additional packages will be installed:\n  libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter\n  libnginx-mod-mail libnginx-mod-stream nginx nginx-common nginx-core\nSuggested packages:\n  fcgiwrap nginx-doc\nThe following NEW packages will be installed:\n  libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter\n  libnginx-mod-mail libnginx-mod-stream nginx nginx-common nginx-core\n0 upgraded, 7 newly installed, 0 to remove and 67 not upgraded.\nNeed to get 605 kB of archives.\nAfter this operation, 2141 kB of additional disk space will be used.\nGet:1 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx-common all 1.18.0-0ubuntu1.7 [37.8 kB]\nGet:2 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-http-image-filter amd64 1.18.0-0ubuntu1.7 [14.8 kB]\nGet:3 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-http-xslt-filter amd64 1.18.0-0ubuntu1.7 [13.1 kB]\nGet:4 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-mail amd64 1.18.0-0ubuntu1.7 [43.0 kB]\nGet:5 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-stream amd64 1.18.0-0ubuntu1.7 [67.3 kB]\nGet:6 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx-core amd64 1.18.0-0ubuntu1.7 [425 kB]\nGet:7 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx all 1.18.0-0ubuntu1.7 [3620 B]\nPreconfiguring packages ...\nFetched 605 kB in 7s (85.6 kB/s)\nSelecting previously unselected package nginx-common.\r\n(Reading database ... \r(Reading database ... 5%\r(Reading database ... 10%\r(Reading database ... 15%\r(Reading database ... 20%\r(Reading database ... 25%\r(Reading database ... 30%\r(Reading database ... 35%\r(Reading database ... 40%\r(Reading database ... 45%\r(Reading database ... 50%\r(Reading database ... 55%\r(Reading database ... 60%\r(Reading database ... 65%\r(Reading database ... 70%\r(Reading database ... 75%\r(Reading database ... 80%\r(Reading database ... 85%\r(Reading database ... 90%\r(Reading database ... 95%\r(Reading database ... 100%\r(Reading database ... 222461 files and directories currently installed.)\r\nPreparing to unpack .../0-nginx-common_1.18.0-0ubuntu1.7_all.deb ...\r\nUnpacking nginx-common (1.18.0-0ubuntu1.7) ...\r\nSelecting previously unselected package libnginx-mod-http-image-filter.\r\nPreparing to unpack .../1-libnginx-mod-http-image-filter_1.18.0-0ubuntu1.7_amd64.deb ...\r\nUnpacking libnginx-mod-http-image-filter (1.18.0-0ubuntu1.7) ...\r\nSelecting previously unselected package libnginx-mod-http-xslt-filter.\r\nPreparing to unpack .../2-libnginx-mod-http-xslt-filter_1.18.0-0ubuntu1.7_amd64.deb ...\r\nUnpacking libnginx-mod-http-xslt-filter (1.18.0-0ubuntu1.7) ...\r\nSelecting previously unselected package libnginx-mod-mail.\r\nPreparing to unpack .../3-libnginx-mod-mail_1.18.0-0ubuntu1.7_amd64.deb ...\r\nUnpacking libnginx-mod-mail (1.18.0-0ubuntu1.7) ...\r\nSelecting previously unselected package libnginx-mod-stream.\r\nPreparing to unpack .../4-libnginx-mod-stream_1.18.0-0ubuntu1.7_amd64.deb ...\r\nUnpacking libnginx-mod-stream (1.18.0-0ubuntu1.7) ...\r\nSelecting previously unselected package nginx-core.\r\nPreparing to unpack .../5-nginx-core_1.18.0-0ubuntu1.7_amd64.deb ...\r\nUnpacking nginx-core (1.18.0-0ubuntu1.7) ...\r\nSelecting previously unselected package nginx.\r\nPreparing to unpack .../6-nginx_1.18.0-0ubuntu1.7_all.deb ...\r\nUnpacking nginx (1.18.0-0ubuntu1.7) ...\r\nSetting up nginx-common (1.18.0-0ubuntu1.7) ...\r\nCreated symlink /etc/systemd/system/multi-user.target.wants/nginx.service -> /lib/systemd/system/nginx.service.\r\nSetting up libnginx-mod-http-xslt-filter (1.18.0-0ubuntu1.7) ...\r\nSetting up libnginx-mod-mail (1.18.0-0ubuntu1.7) ...\r\nSetting up libnginx-mod-http-image-filter (1.18.0-0ubuntu1.7) ...\r\nSetting up libnginx-mod-stream (1.18.0-0ubuntu1.7) ...\r\nSetting up nginx-core (1.18.0-0ubuntu1.7) ...\r\nSetting up nginx (1.18.0-0ubuntu1.7) ...\r\nProcessing triggers for systemd (245.4-4ubuntu3.24) ...\r\nProcessing triggers for man-db (2.9.1-1) ...\r\nProcessing triggers for ufw (0.36-6ubuntu1.1) ...\r\n",
    "stdout_lines": [
        "Reading package lists...",
        "Building dependency tree...",
        "Reading state information...",
```



4. Start a service
   ansible local -i host.ini -m service -a "name=nginx state=started" –become



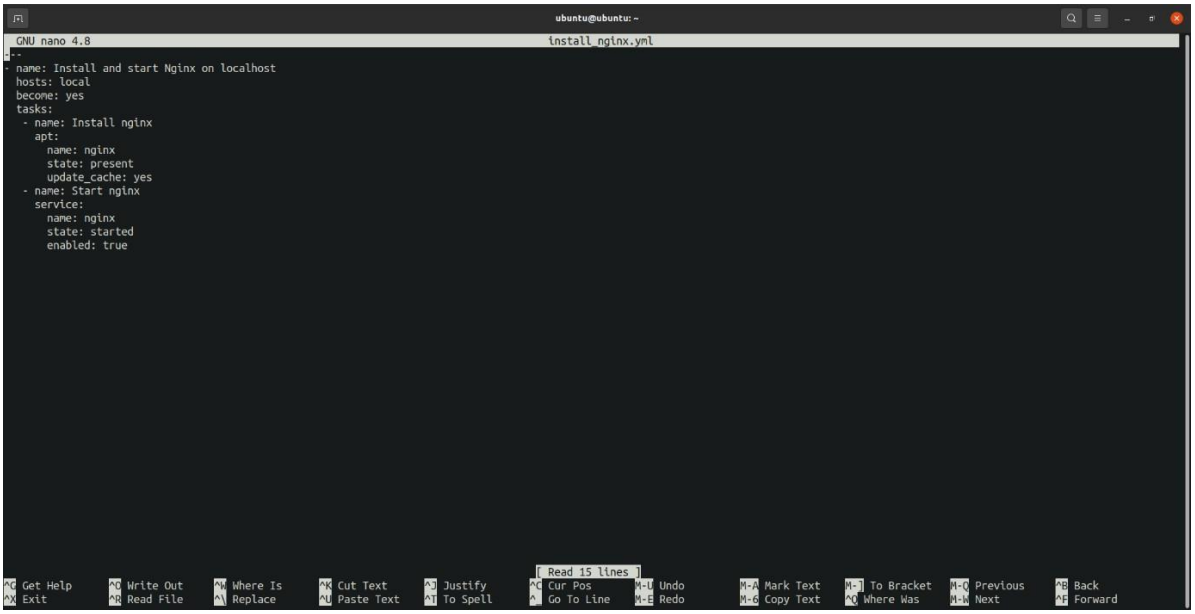
Practical 12
AIM: Demonstrate Ansible Playbooks

Install and Start Nginx

install_nginx.yml:

```
- name: Install and start Nginx on web servers
hosts: webservers
become: true tasks:
- name: Install
  Nginx apt:
  name: nginx
  state: present
  update_cache: yes
- name: Start Nginx
   service:
     name: nginx
     state: started
     enabled: true
```

```
ubuntu@ubuntu:~$ nano install_nginx.yml
```



Run the Playbook:
ansible-playbook -i hosts.ini install_nginx.yml

```
ubuntu@ubuntu:~$ ansible-playbook -i host.ini install_nginx.yml

PLAY [Install and start Nginx on localhost] ************************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************************
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host localhost should use /usr/bin/python3, but is using /usr/bin/python for backward compatibility with prior Ansible releases. A future
 Ansible release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information. This feature will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [localhost]

TASK [Install nginx] ***********************************************************************************************************************************************************************
ok: [localhost]

TASK [Start nginx] *************************************************************************************************************************************************************************
ok: [localhost]

PLAY RECAP *********************************************************************************************************************************************************************************
localhost                  : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```