**1. basic Git commands**

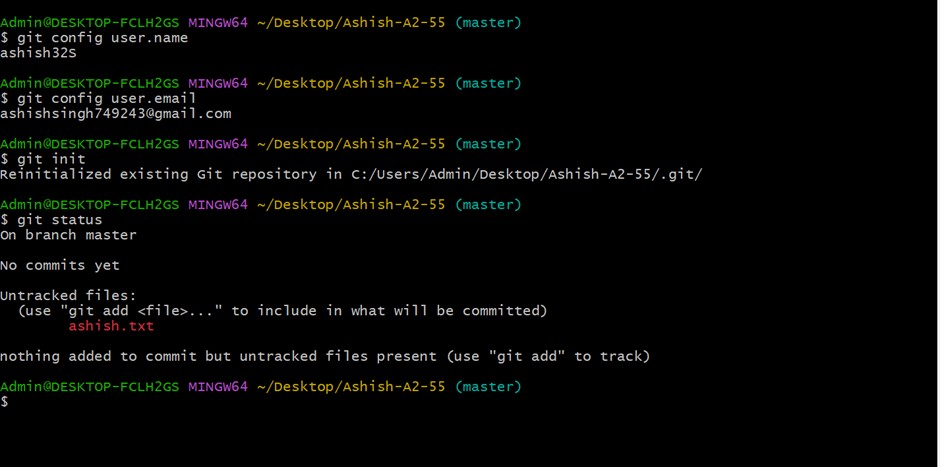
# Setup & Configuration

* git config --global user.name "Your Name" – Set your Git username.
* git config --global user.email "your.email@example.com" – Set

your Git email.

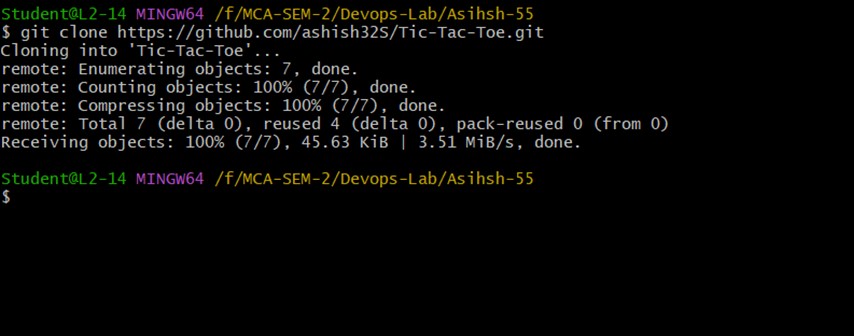
# Initialize a Repository

● git init – Initialize a new Git repository in the current directory.



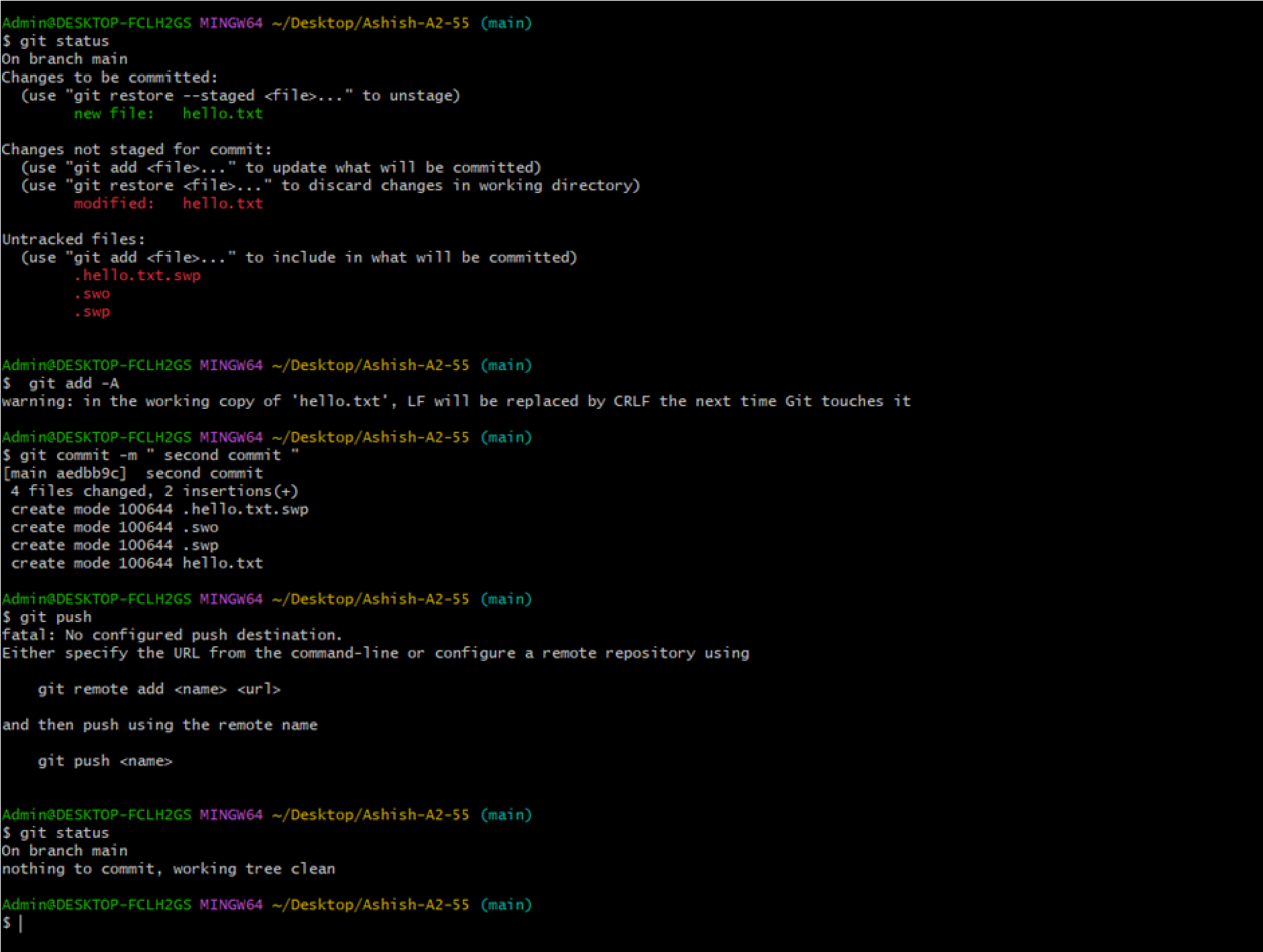
# Clone a Repository

● git clone <repository\_url> – Clone an existing repository from a remote source.



# Working with Files

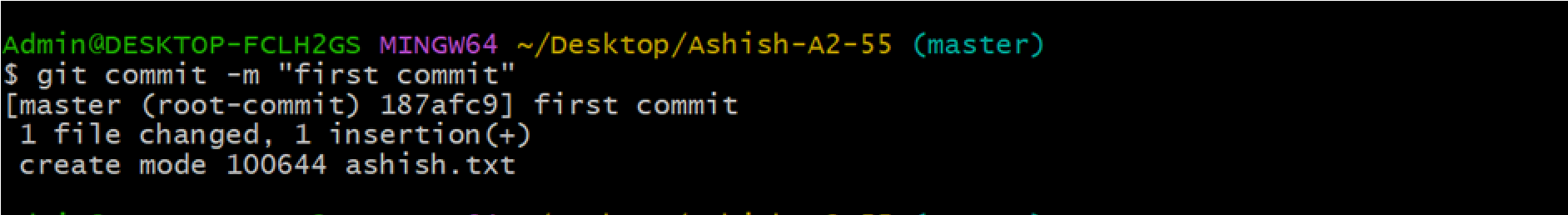
* git status – Check the status of files (staged, unstaged, untracked).
* git add <file> – Add a specific file to the staging area.
* git add . – Add all changes to the staging area.
* git rm <file> – Remove a file from the repository and working directory.



# Committing Changes

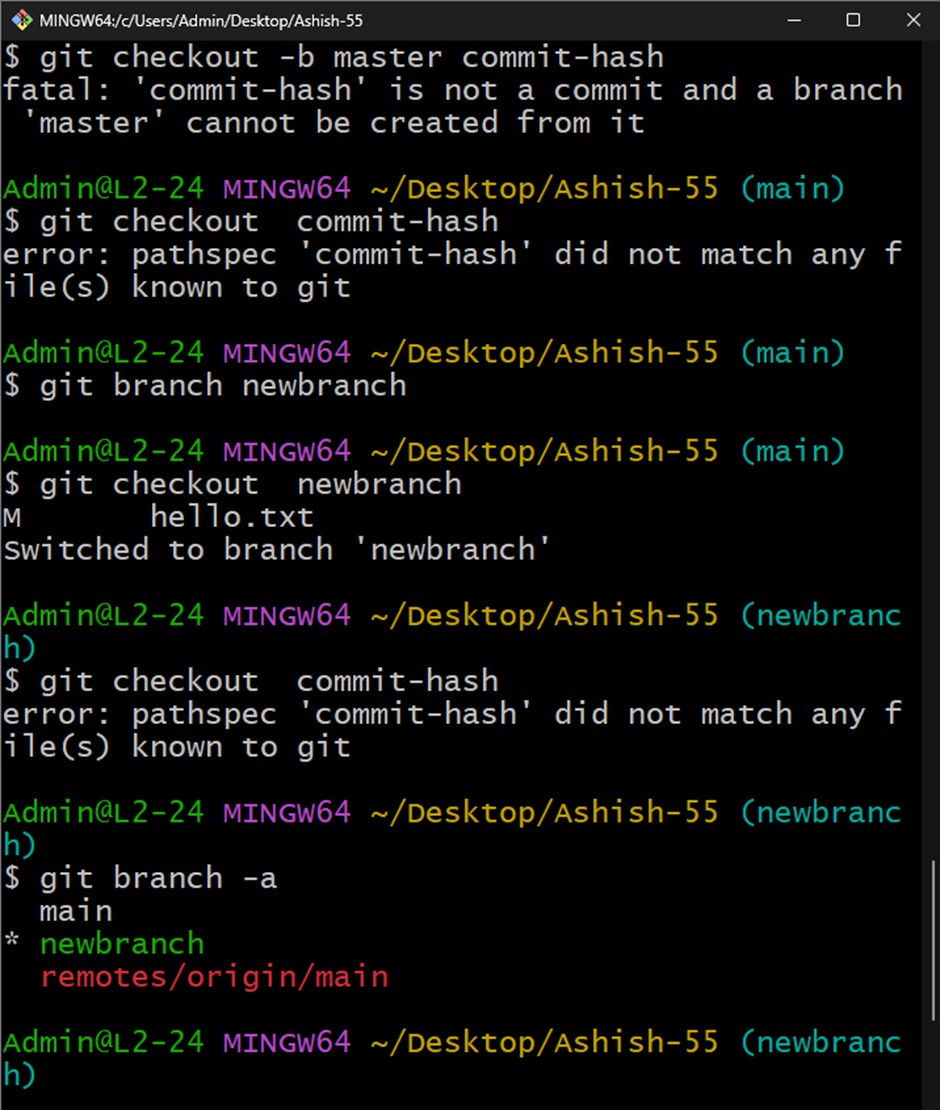
* git commit -m "Commit message" – Commit changes with a message.
* git commit -am "Commit message" – Add and commit changes in one step

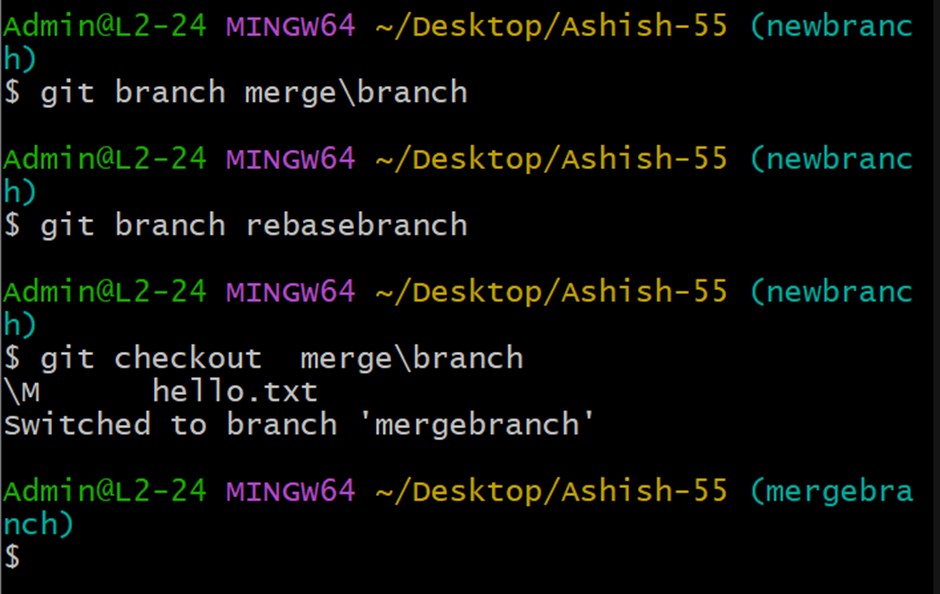
(only tracked files).



# Branching & Merging

* git branch – List branches.
* git branch <branch\_name> – Create a new branch.
* git checkout <branch\_name> – Switch to another branch.
* git checkout -b <branch\_name> – Create and switch to a new branch.
* git merge <branch\_name> – Merge a branch into the current branch.

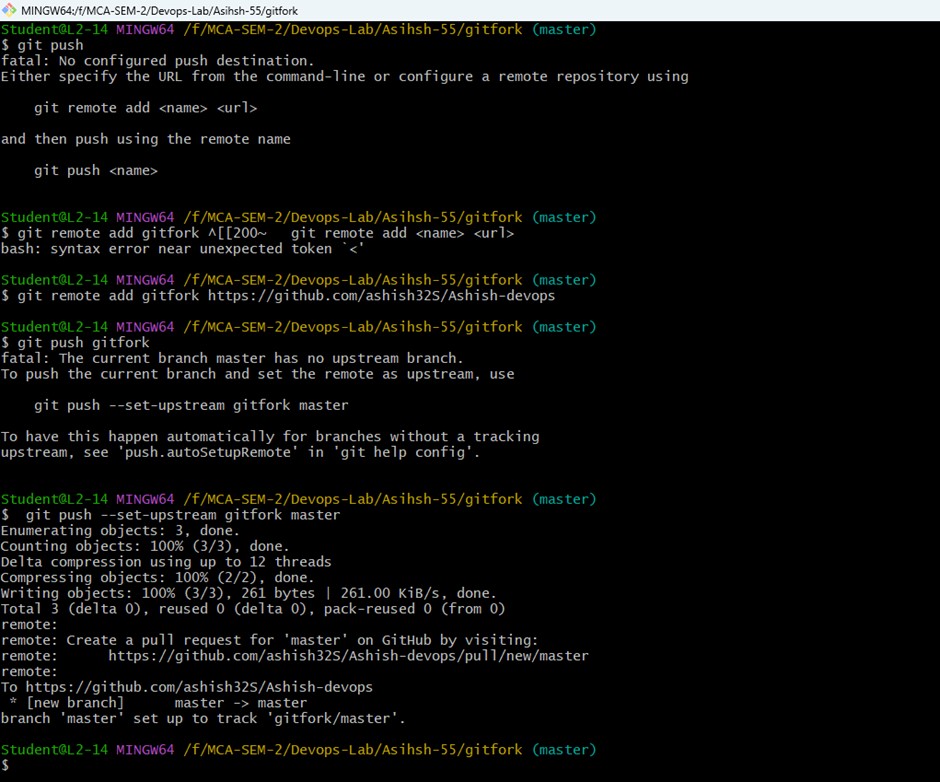




# Working with Remote Repositories

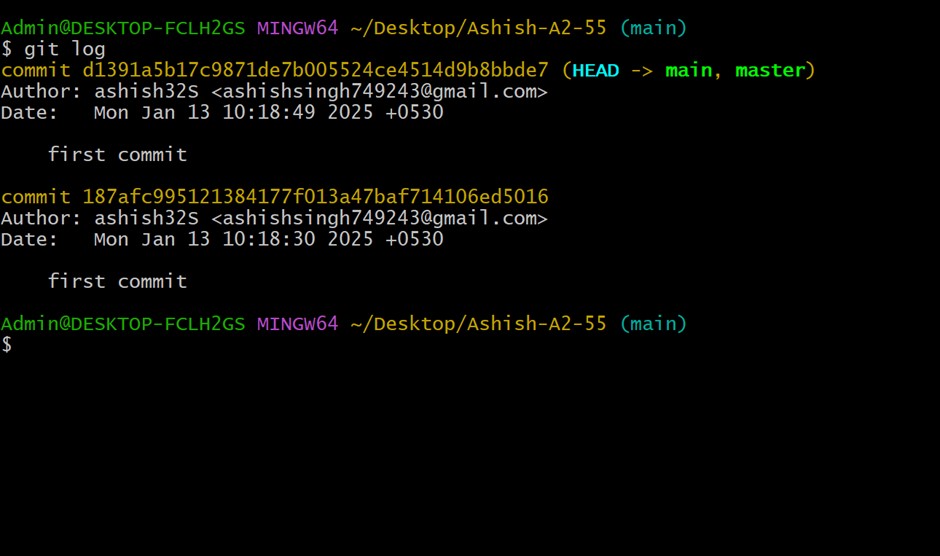
* git remote add origin <repository\_url> – Add a remote repository.
* git remote -v – View remote repositories.
* git push -u origin <branch\_name> – Push a branch to a remote repository.
* git pull origin <branch\_name> – Pull the latest changes from the remote branch.

.



# Viewing History

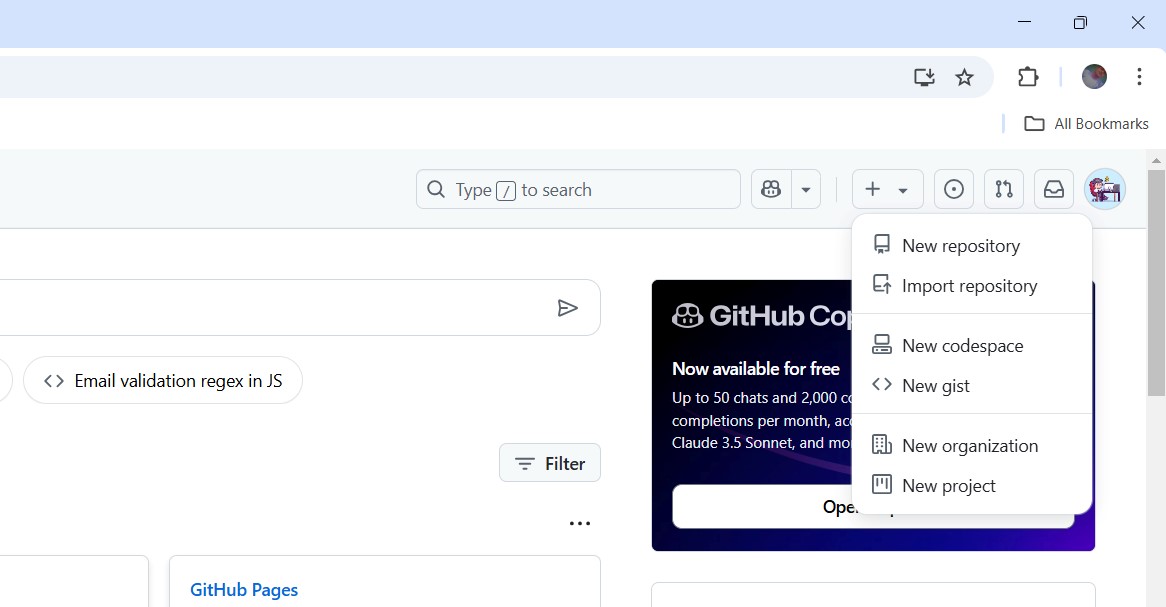
● git log – Show commit history.



**2. creating and forking repositories on GitHub, along with applying branching, merging, and rebasing concepts in Git.**

# 1. Create a Repository on GitHub

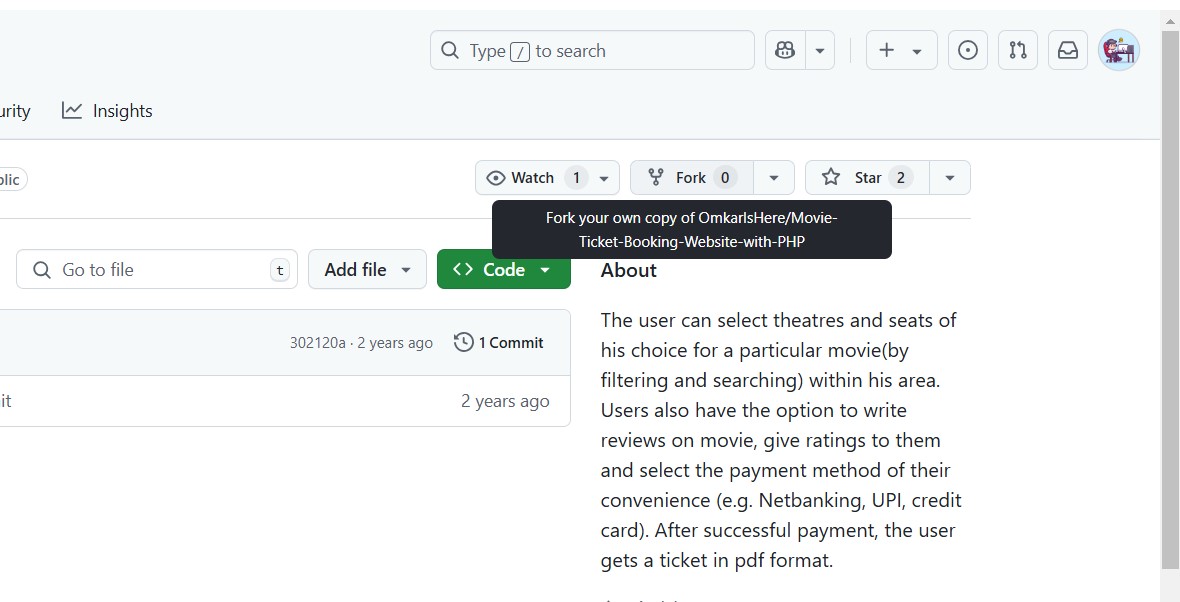
1. Go to [GitHub](https://github.com/) and log in.
2. Click on **"+"** (top-right corner) → **"New repository"**.
3. Provide a **repository name** (e.g., my-project).
4. Choose **public/private** visibility.
5. (Optional) Add a README, .gitignore, and license.
6. Click **Create repository**.



# 2. Fork a Repository

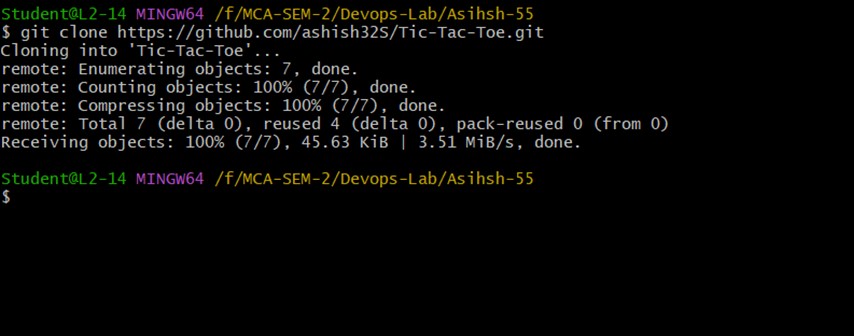
A fork creates a copy of someone else's repository under your GitHub account.

1. Go to the repository you want to fork.
2. Click the **Fork** button (top-right).
3. Choose where to fork the repository (your GitHub account).
4. GitHub creates a **copy** under your profile.



💡 **To clone the forked repo locally**:

git clone https://github.com/YOUR\_USERNAME/FORKED\_REPO.git cd FORKED\_REPO



# 3. Work with Branching

Branches allow you to work on features without affecting the main branch.

**Create a new branch:**

git checkout -b feature-branch

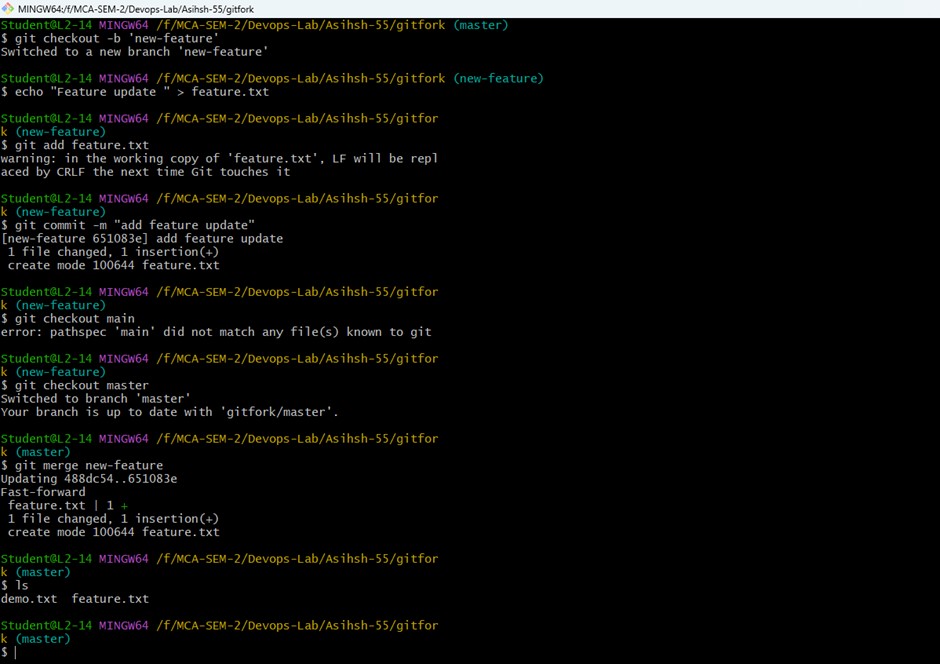
OR

git branch feature-branch git checkout feature-branch

**List all branches:**

git branch

**Push a branch to GitHub:** git push origin feature-branch



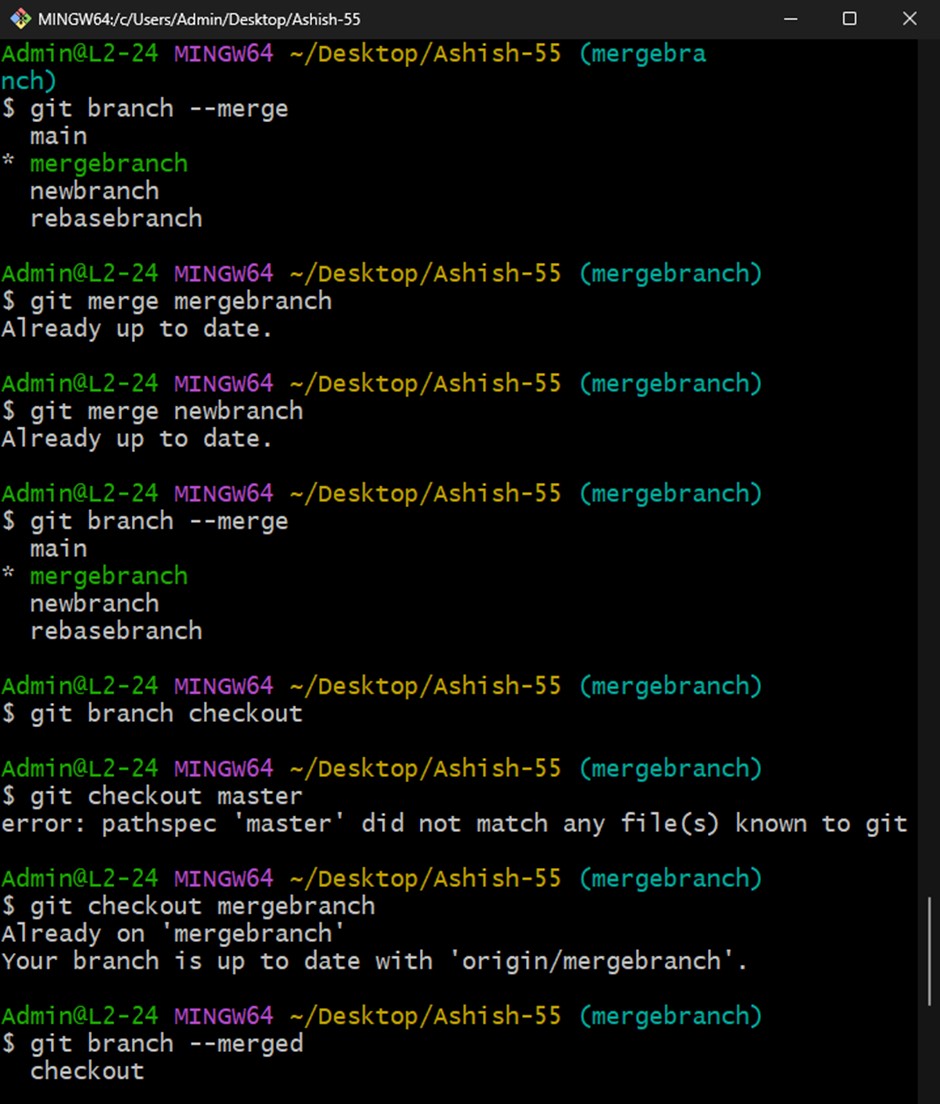
# 4. Merge Branches

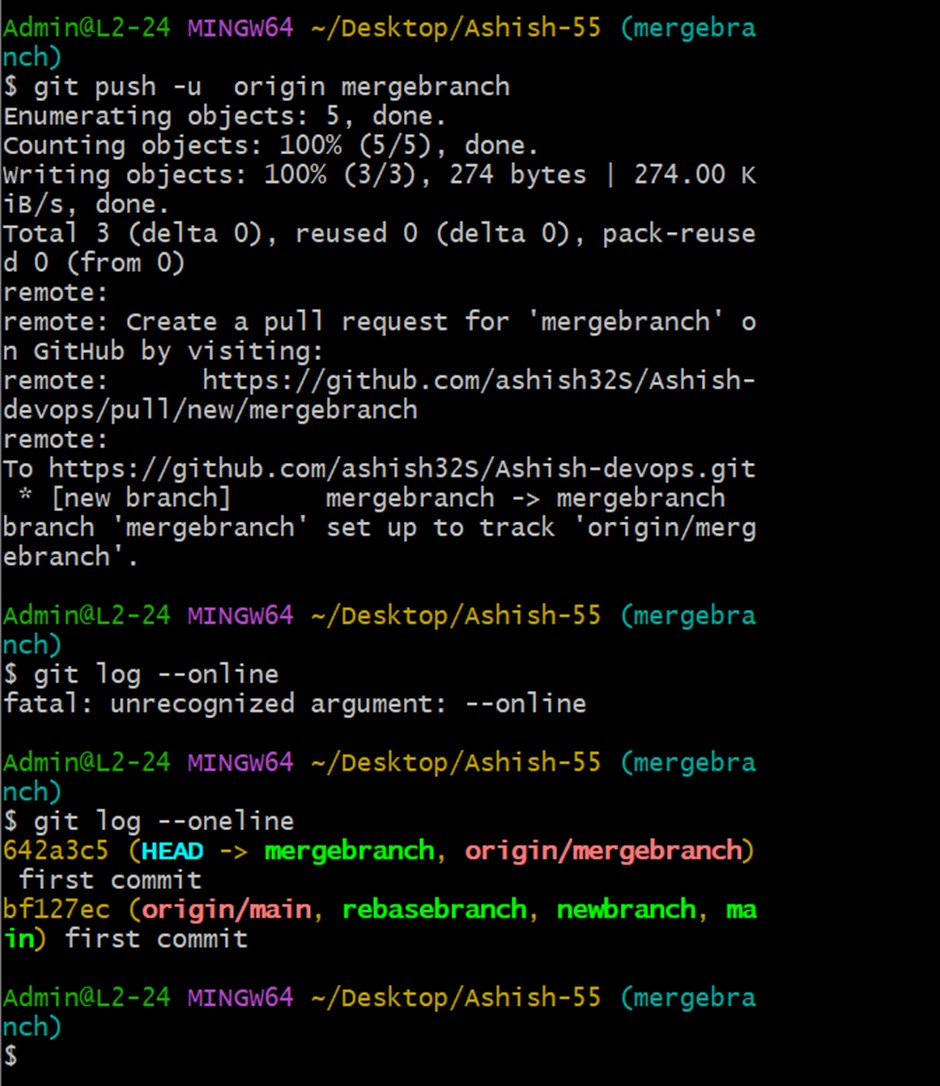
Merging combines changes from one branch into another.

**Merge using GitHub:**

1. Push changes to GitHub (git push).
2. Open a **Pull Request** in GitHub.
3. Click **Merge** when approved.

**Merge using Git (Fast-forward merge):** git checkout main git merge feature-branch git push origin main





# 5. Rebase (Alternative to Merge)

Rebasing keeps the commit history linear.

**Rebase onto the latest main branch:**

git checkout feature-branch git rebase main

**Rebase with conflicts resolution:**

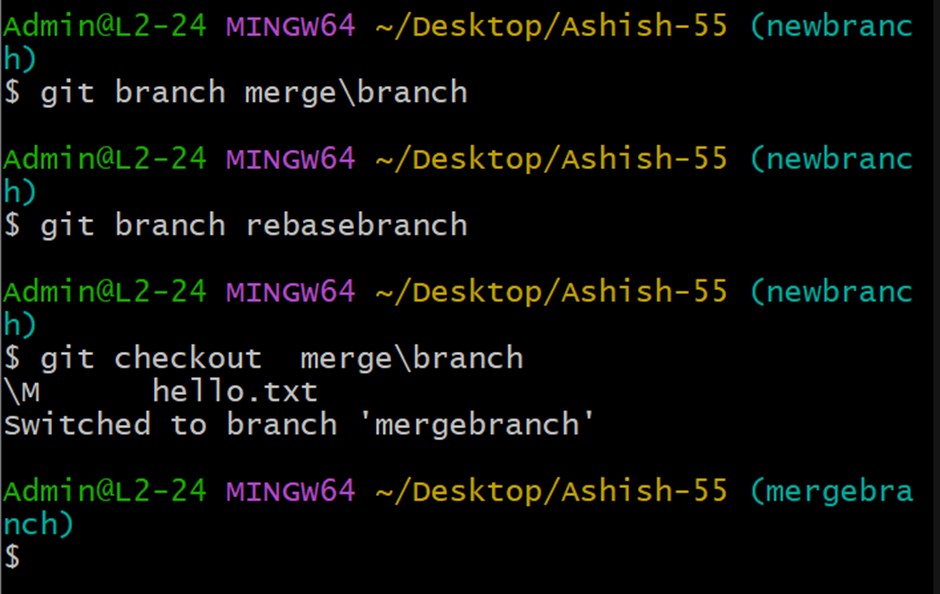
1. Git will pause if there are conflicts.
2. Fix the conflicting files manually.

Use: git add .

git rebase --continue

3.

Finally, push changes: git push --force origin feature-branch



# 6. Sync Fork with Original Repository

To keep your fork updated with the original repository:

git remote add upstream https://github.com/ORIGINAL\_OWNER/REPO\_NAME.git git fetch upstream git checkout main git merge upstream/main git push origin main

**3. Using Git for Collaboration**

**Basic Collaborative Workflow:**

1. **Clone the Repository:**
   1. Start by cloning the remote repository to your local machine:

|  |  |
| --- | --- |
| Bash |  |
| git clone <repository\_url> | |

1. **Create a Branch:**
   1. Create a new branch for your feature or bug fix:

|  |  |
| --- | --- |
| Bash |  |
| git checkout -b feature-name | |

1. **Make Changes and Commit:**
   1. Make your changes to the files.

○ Stage the changes:

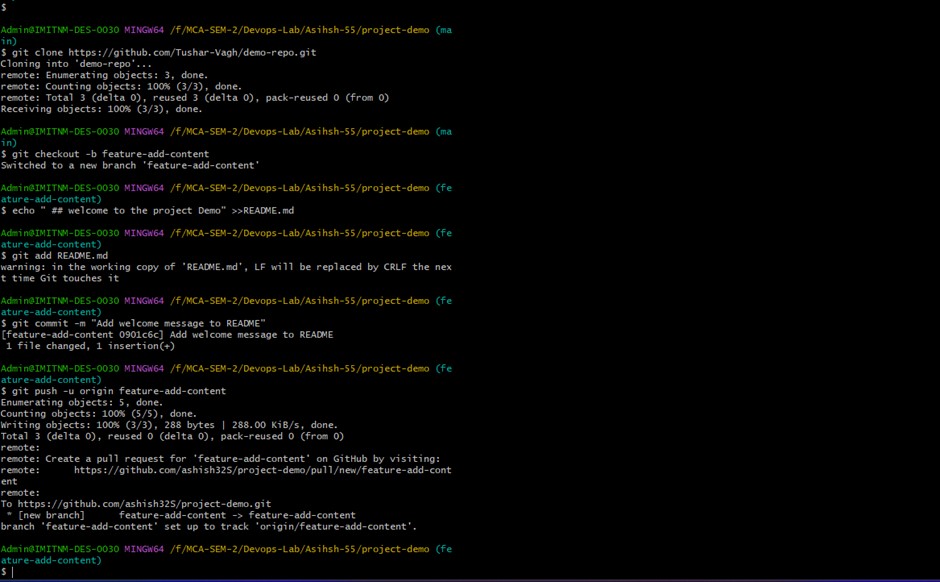
|  |  |
| --- | --- |
| Bash |  |
| git add . | |

○ Commit the changes with a descriptive message:

|  |  |
| --- | --- |
| Bash |  |
| git commit -m "Add feature-name" | |

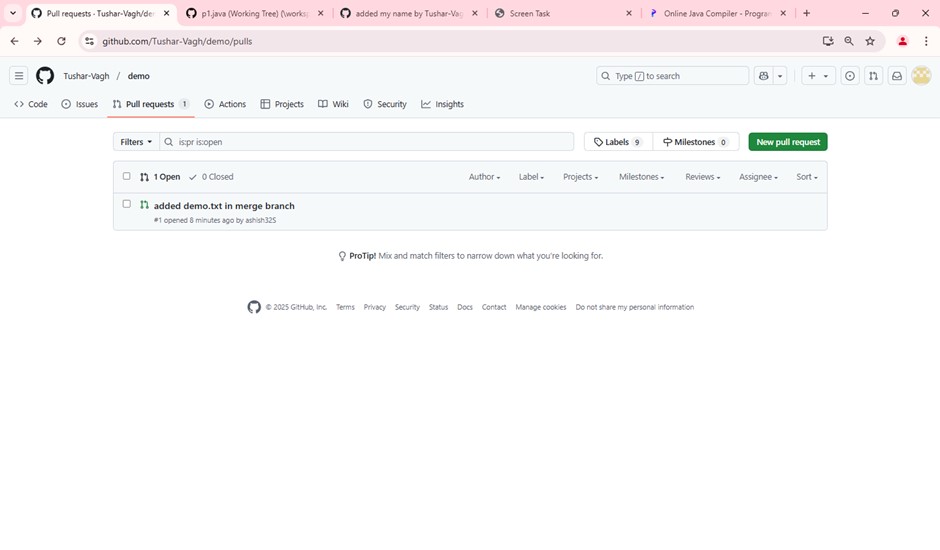
1. **Push Your Branch:**
   1. Push your branch to the remote repository:

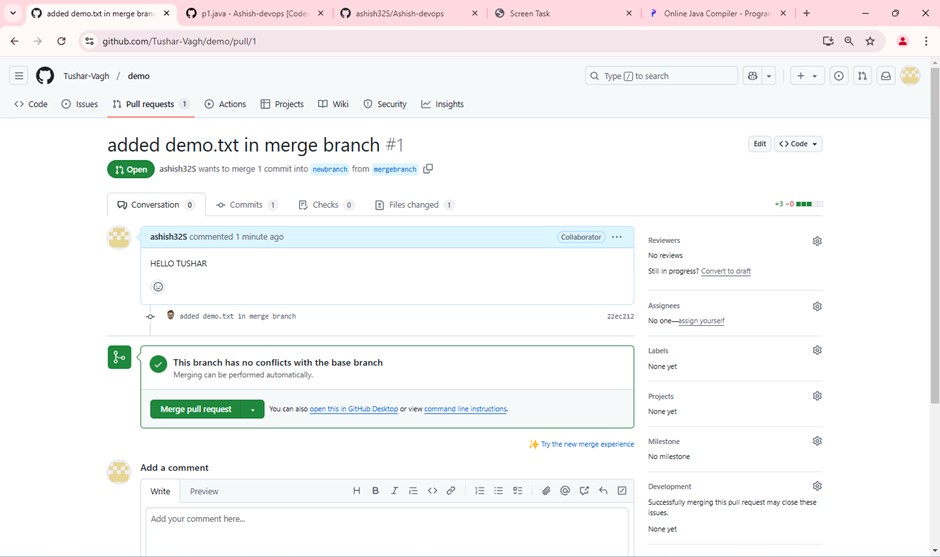
|  |  |
| --- | --- |
| Bash |  |
| git push origin feature-name | |



1. **Create a Pull Request:**
   1. Go to the remote repository (e.g., on GitHub) and create a pull request from your branch to the main branch (often main or develop).

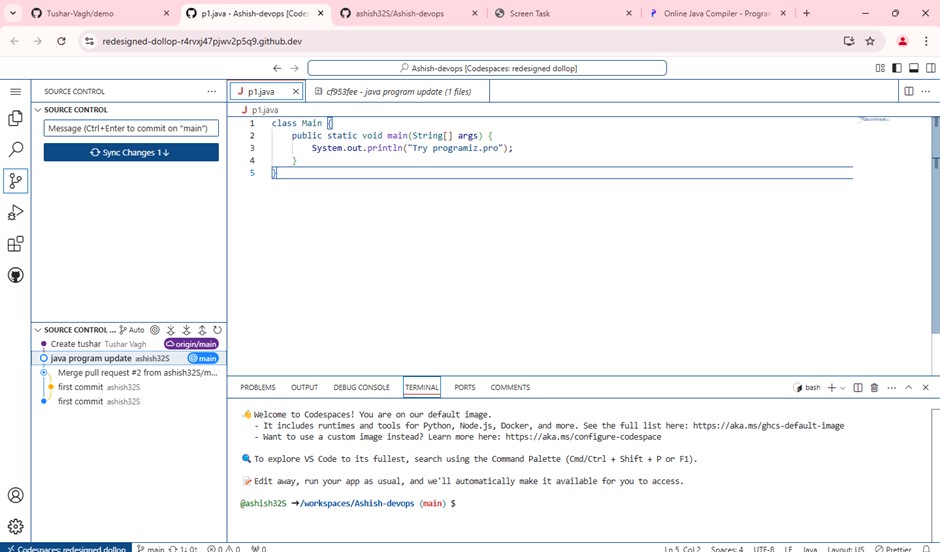
○ Describe your changes and request a review from your team members.





1. **Code Review:**
   1. Team members review your code, provide feedback, and suggest changes.

○ Make the necessary changes and push them to your branch. The pull request will automatically update.



**4. Collaborating and Cloning Using GitHub**

# 1. Cloning a Repository

Cloning allows you to create a local copy of a GitHub repository so you can work on it.

## Clone a GitHub Repository

1. Go to the GitHub repository page.
2. Click on the **"Code"** button.
3. Copy the repository URL (HTTPS or SSH).

Open a terminal and run:

git clone https://github.com/USERNAME/REPO\_NAME.git

4.

Navigate into the project directory:

cd REPO\_NAME

# 2. Collaborating on a GitHub Repository

To work on a project together, team members must be able to **push changes**, **create branches**, and **submit pull requests**.

## Add Collaborators

1. Go to the repository on GitHub.
2. Click on **Settings** → **Manage Access**.
3. Click **Invite Collaborator** and add GitHub usernames.
4. Collaborators will receive an invitation to accept.

# 3. Working with Branches for Collaboration

Each team member should work on a separate branch instead of making changes directly to main.

**Create a New Branch**

git checkout -b feature-branch

**Push a Branch to GitHub**

git push origin feature-branch

**View All Branches**

git branch -a

**Switch to Another Branch**

git checkout branch-name

# 4. Pull Requests (PRs) for Collaboration

A **Pull Request (PR)** allows team members to review and discuss changes before merging.

## Submit a Pull Request

Push your branch to GitHub: git push origin feature-branch

1.

1. Go to **GitHub** → **Repository**.
2. Click **Pull Requests** → **New Pull Request**.
3. Select the base branch (main) and compare branch (feature-branch).
4. Add a **title**, **description**, and request reviewers.
5. Click **Create Pull Request**.
6. Team members review and approve or request changes.
7. Click **Merge Pull Request** when ready.

# 5. Keeping Your Local Repository Updated

To avoid conflicts, regularly update your local repository with the latest changes.

**Pull Latest Changes**

git pull origin main

## Fetch and Rebase for a Clean History

git fetch origin git rebase origin/main

## Merge Latest Changes

git checkout main git merge feature-branch git push origin main

# 6. Forking and Syncing a Repository

If you don’t have write access to a repository, you can **fork** it and submit **pull requests**.

## Fork a Repository

1. Go to the repository on GitHub.
2. Click **Fork** (top-right corner).

Clone your fork:

git clone https://github.com/YOUR\_USERNAME/FORKED\_REPO.git