Git Basics

1. What is Git?

Git is a **distributed version control system** used to track changes in code and collaborate with others. It allows developers to manage and maintain multiple versions of their code effectively.

- **Why use Git?**
  - Tracks changes in files over time.
  - Enables collaboration among developers.
  - Allows you to roll back to previous versions of code.

Useful Link:

Git Documentation

---

2. Initializing a Git Repository

To start using Git in a project, you need to initialize a repository:

bash
CopyEdit
```
git init
```

This creates a hidden `.git` folder that stores the project's version history.

---

3. Tracking Changes with Git

- Use `git add <file>` to stage changes.
- Use `git add .` to stage all changes in the current directory.

Commit the staged changes with:
bash
CopyEdit
```
git commit -m "Commit message"
```

-
- **Staging vs. Committing:**
  - Staging prepares changes for a commit.
  - Committing saves a snapshot of the changes to the repository.

---

4. Checking the Status of Your Repository

The `git status` command shows:

- The current branch name.
- Staged and unstaged changes.
- Untracked files.

---

5. Branches in Git

Branches are used to work on different features or fixes without affecting the main project.

Creating a new branch:
bash
CopyEdit
`git branch <branch-name>`

-

Switching branches:
bash
CopyEdit
`git switch <branch-name>`

-

Merging branches:
To merge changes from one branch into another:
bash
CopyEdit
`git merge <branch-name>`

-

Useful Link:

Git Branching

---

6. Viewing History

Use `git log` to see the commit history. You'll get details about:

- Commit hashes.

- Author information.
- Commit messages.

---

7. Ignoring Files

The `.gitignore` file specifies files and directories that Git should ignore. For example, to ignore all `.log` files:

bash
CopyEdit
```
*.log
```

Useful Link:

.gitignore Documentation

---

8. Undoing Changes

To undo the last commit but keep the changes:
bash
CopyEdit
```
git reset --soft HEAD~1
```

-

To remove a file from tracking without deleting it:
bash
CopyEdit
```
git rm --cached <file>
```

-

---

9. Pulling Changes

The `git pull` command updates your local branch with changes from the remote repository. It performs two actions:

- Fetches changes.
- Merges them into the current branch.

GitHub Basics

1. What is GitHub?

GitHub is a **web-based platform** for hosting and collaborating on Git repositories. It provides tools for:

- Sharing code.
- Managing projects.
- Reviewing code changes.

Useful Link:

GitHub Documentation

---

2. Repositories

A repository (repo) is a storage space for your project. It contains all your project files and version history.

---

3. Forking a Repository

Forking creates a copy of another user's repository in your account. This allows you to work on the project independently.

---

4. Cloning a Repository

Cloning creates a local copy of a GitHub repository on your computer. Use the command:

bash
CopyEdit

```
git clone <repository-URL>
```

Useful Link:

Cloning Repositories

---

5. Pushing Changes

To send your local changes to a remote repository:

bash
CopyEdit

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

---

6. Pull Requests

A pull request is a way to propose changes to a repository. It allows maintainers to review and discuss your changes before merging them.

Steps to Create a Pull Request:

1. Push your changes to a branch on GitHub.
2. Go to the repository on GitHub.
3. Click on "New Pull Request."

Useful Link:

GitHub Pull Requests

---

7. GitHub Pages

GitHub Pages is a feature that lets you host static websites directly from a GitHub repository.

Steps to Enable GitHub Pages:

1. Go to the repository settings.
2. Select a branch and directory for the site.
3. Visit the provided URL to access your site.

Useful Link:

GitHub Pages

---

8. GitHub Actions

GitHub Actions automates tasks like running tests, building code, or deploying projects.

Key Features:

- Continuous Integration/Continuous Deployment (CI/CD).
- Custom workflows.

Useful Link:

GitHub Actions Documentation

---

9. Issues in GitHub

GitHub Issues is a tool for tracking tasks, bugs, or feature requests.

How to Create an Issue:

1. Go to the "Issues" tab in a repository.
2. Click "New Issue."
3. Add a title, description, and assign labels.

Recommended Resources for Further Learning

1. Git Basics: Beginner’s Guide by GitHub
2. Pro Git Book (Free Online)
3. Interactive Git Tutorial