

# Git & GitHub Roadmap

---

## 1. Basics of Version Control Systems (VCS)

**Goal:** Understand *why* version control is important.

- **Topics to Cover:**

- What is Version Control?
  - Benefits of Version Control (tracking changes, collaboration, rollback)
  - Centralized vs. Distributed Version Control Systems
  - Introduction to Git (a distributed VCS)
- 

## 2. Setting Up Git

**Goal:** Install Git and configure basic settings.

- **Steps:**

1. **Install Git:**

- [Download Git](#) and install it on your machine.

**Initial Configuration:**

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
git config --global init.defaultBranch main # Set 'main' as default branch
```

2. **Check Configurations:**

```
git config --list
```

---

## 3. Basic Git Commands

**Goal:** Learn how to create repositories, track files, and commit changes.

- **Topics & Commands:**

**Initialize a Git Repository:**

```
git init
```

**Check the Repository Status:**

```
git status
```

**Adding Files to Staging Area:**

```
git add <filename>      # Add a specific file
```

```
git add .                # Add all files
```

**Committing Changes:**

```
git commit -m "Your commit message"
```

**Viewing Commit History:**

```
git log
```

```
git log --oneline      # Compact view
```

---

## 4. Understanding the Git Workflow

**Goal:** Grasp the basic Git workflow: *Working Directory* → *Staging Area* → *Repository*.

- **Topics:**

- Working Directory vs. Staging Area vs. Local Repository
- The Lifecycle of a File in Git
- Modifying, Staging, and Committing

**Visualize this for students:**

[Working Directory] --(git add)--> [Staging Area] --(git commit)--> [Local Repository]

---

## 5. Working with GitHub

**Goal:** Learn how to link local repositories to remote repositories on GitHub.

- **Steps:**
  1. **Create a GitHub Account:** [Sign up here](#).
  2. **Create a Remote Repository on GitHub:**
    - Go to GitHub → New Repository → Name it → Create.

### **Link Local Repo to GitHub:**

```
git remote add origin https://github.com/yourusername/your-repo.git  
git push -u origin main
```

### **Pushing Changes:**

```
git push
```

---

## 6. Cloning and Pulling Repositories

**Goal:** Learn how to work with existing repositories.

- **Commands:**

### **Cloning a Repository:**

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

### **Pulling Changes from Remote:**

```
git pull origin main
```

---

## 7. Branching and Merging (Important for Teaching!)

**Goal:** Understand how to create branches and merge them. This is essential for collaborative development.

- **Topics & Commands:**

**Create a New Branch:**

`git branch new-feature`

**Switch Between Branches:**

`git checkout new-feature`

# or in one step:

`git checkout -b new-feature`

**Merge Branches:**

`git checkout main`

`git merge new-feature`

**Delete a Branch (after merging):**

`git branch -d new-feature`

---

## 8. Handling Merge Conflicts

**Goal:** Learn to handle conflicts when merging branches.

---

## 9. Working with .gitignore

**Goal:** Learn to ignore files that shouldn't be tracked by Git.

---

## 10. Undoing Changes

**Goal:** Learn how to undo mistakes and revert to previous versions.

### Undo Changes in Working Directory:

`git checkout -- <file>` # Discard changes in a file

### Unstage Files:

`git reset HEAD <file>` # Remove from staging area

### Amend the Last Commit:

`git commit --amend`

### Revert a Commit (without rewriting history):

`git revert <commit-hash>`

---

## 11. Collaborating on GitHub (Fork, Pull Requests, Issues)

**Goal:** Learn to contribute to open-source projects.

- **Topics:**

1. **Forking Repositories:**

- Fork a repo on GitHub to create a copy in your account.

2. **Creating Pull Requests:**

- Push your changes to your forked repo and create a pull request to the original repository.

3. **Using Issues:**

- Report bugs or request features using GitHub Issues.
- 

## 12. Advanced Git Concepts

**Goal:** Dive into more advanced workflows.

**Rebasing:**

git rebase main

**Stashing (Saving Work Temporarily):**

git stash

git stash apply

**Tagging:**

git tag v1.0

git push origin v1.0

**Cherry-picking (Apply specific commits):**

git cherry-pick <commit-hash>

---

## 13. Best Practices

**Goal:** Develop good habits when using Git.

- **Best Practices:**

- Write clear and concise commit messages.
  - Commit small, logical changes frequently.
  - Use branches for features, bug fixes, and experiments.
  - Always pull the latest changes before starting new work.
- 

## 14. Tools & Resources

- **Graphical Interfaces:**

- **GitHub Desktop** (for beginners)
- **GitKraken, Sourcetree**

### **Visualizing Git History:**

`git log --graph --oneline --all`

- - **Learning Platforms:**
    - [Git Documentation](#)
    - [GitHub Learning Lab](#)
    - [Learn Git Branching \(Interactive\)](#)
-