**1. What is Git?**

* Git is a version control system that tracks changes to files and helps manage collaborative work on projects.
* It is installed on your local computer and operates independently of online platforms.

**2. What is GitHub?**

* GitHub is a platform for hosting Git repositories online.
* It facilitates collaboration, sharing, and managing Git repositories.

**3. Core Git Concepts**

**a) Repository (Repo)**

* A repository is a directory that contains your project files and a hidden folder (.git) where Git stores all the history and metadata about your project.
* Example of initializing a repository:

git init

**b) Working Directory**

* The working directory is the folder on your computer where you make changes to your project files.

**c) Staging Area**

* The staging area is a space where changes are prepared for a commit.
* When you add files to the staging area, you’re telling Git which changes should be included in the next snapshot.
* Command to add files to the staging area:

git add <file>

**d) Commit**

* A commit is a snapshot of the changes in the staging area.
* It creates a record in the repository’s history.
* Command to commit changes:

git commit -m "Describe changes"

**e) Branch**

* A branch is a separate line of development within the repository.
* By default, Git creates a main branch, but you can create additional branches for features or experiments.
* Commands:
  + Create a new branch:

git branch <branch-name>

* + Switch to a branch:

git checkout <branch-name>

**4. Basic Workflow**

1. Make changes in your working directory.
2. Add changes to the staging area:

git add <file>

1. Commit the changes to the repository:

git commit -m "Your message"

1. Push the changes to a remote repository on GitHub:

git push origin <branch-name>

**5. Key GitHub Concepts**

**a) Clone**

* Copy a repository from GitHub to your local computer.
* Command:

git clone <repository-url>

**b) Pull**

* Fetch and merge the latest changes from the remote repository into your local branch.
* Command:

git pull <remote-name> <branch-name>

**c) Push**

* Upload your local commits to a branch in the remote repository.
* Command:

git push <remote-name> <branch-name>

**d) Pull Request**

* A pull request is a way to propose changes from one branch to another, often used for code review and collaboration.
* It can be created directly on GitHub.

**6. Merge Conflicts**

* A merge conflict occurs when Git cannot automatically resolve changes between branches.
* To resolve conflicts:
  1. Open the conflicting file and edit it to resolve the conflict.
  2. Stage the resolved file:

git add <file>

* 1. Commit the resolution:

git commit

**7. Revert**

* **Maintains History**: Yes.
* **What it Does**: Creates a new commit that reverses the changes made by a specific commit.
* **Use Case**: When you want to undo changes but preserve a clear history of what happened.
* **Command**:

git revert <commit-hash>

* Example: If commit abc123 added a bug, reverting it creates a new commit that negates the changes introduced by abc123.

**8. Reset**

* **Maintains History**: Depends on how it is used.
* **What it Does**: Moves the branch pointer to a specific commit and optionally changes the working directory and staging area.
* **Options**:
  + --soft: Keeps changes in the staging area.
  + --mixed (default): Keeps changes in the working directory but unstages them.
  + --hard: Deletes changes completely.

**Command Examples:**

* To unstage changes but keep the files:

git reset <commit-hash>

* To delete changes completely:

git reset --hard <commit-hash>

**Basic GitHub/Git Operations:**

**1. Status**

* Used to check the current state of your Git repository.

**command**

git status

**2. Authentication**

* Set up your user details for Git.

**command**

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

git config --global user.name "YourName"

**3. Clone**

* **Definition**: Copy a repository from GitHub to your local machine.
* **Command**:

git clone <repository-url>

* **Use Case**: Start working on an existing project hosted on GitHub.

**4. Branch**

* **Definition**: A branch is a separate line of development, allowing parallel work without affecting the main codebase.
* **Commands**:
  + Create a new branch:

git branch <branch-name>

* + Switch to a branch:

git checkout <branch-name>

* + Create and switch:

git checkout -b <branch-name>

* + List all branches:

git branch

* **Use Case**: Work on features, fixes, or experiments without disturbing the main branch.

**5. Push**

* **Definition**: Upload local changes to a remote repository.
* **Command**:

git push <remote-name> <branch-name>

* **Use Case**: Share your changes with collaborators or update the remote repository.

**6. Pull**

* **Definition**: Fetch changes from a remote repository and merge them into your local branch.
* **Command**:

git pull <remote-name> <branch-name>

* **Use Case**: Keep your local branch up-to-date with the remote branch.

**7. Pull Request**

* **Definition**: A request to merge changes from one branch into another, typically used in collaborative workflows.
* **How to Create on GitHub**:
  1. Push your branch to GitHub.
  2. Go to the repository on GitHub.
  3. Click **"Pull Request"** and follow the prompts to merge your branch.
* **Use Case**: Code review, discussion, and collaboration before changes are merged into the main branch.

**8. Merge**

* **Definition**: Combine the changes from one branch into another.
* **Command**:

git merge <branch-name>

* **Use Case**: Finalize development or feature integration into the main codebase.

**9. Merge Conflict**

* **Definition**: Occurs when changes in two branches conflict and cannot be merged automatically.
* **Resolution**:
  + Open the conflicting files.
  + Edit the conflicts marked by Git (e.g., <<<<<<<, =======, >>>>>>>).
  + After resolving conflicts, stage and commit:

git add <file>

git commit

* **Use Case**: Happens when two people edit the same part of a file or when simultaneous changes affect the same area.

**10. Revert**

* **Definition**: Undo a specific commit by creating a new commit that reverses its changes.
* **Command**:

git revert <commit-hash>

* **Use Case**: Roll back changes without rewriting history.

**Practical Workflow Example:**

1. **Clone** the repository:

git clone https://github.com/user/repo.git

cd repo

1. Create and **switch to a new branch**:

git checkout -b feature-branch

1. Make changes, then **stage and commit** them:

git add .

git commit -m "Add feature"

1. **Push** the branch:

git push origin feature-branch

1. Create a **pull request** on GitHub to merge feature-branch into main.
2. If there’s a **merge conflict**, resolve it locally and push the fixes.

GitHub Collaboration Guide

# 1. Accessing the Repository

* **Invite Collaborators:**  
  Go to your GitHub repo, click **Settings** > **Collaborators and teams**, and add your teammates by their GitHub usernames or email addresses.
* **Access the Repository Link:**  
  Share the repository URL with your teammates.

# 2. Cloning the Repository

Teammates need to clone the repository to their local machines:

git clone <repository-url>

Example:

git clone https://github.com/your-username/your-repo.git

# 3. Pulling the Latest Changes

Once cloned, they should pull the latest changes to ensure their local copy is up-to-date:

git pull origin main

# 4. Creating a New Branch

Encourage teammates to create feature branches for their changes:

git checkout -b feature/branch-name

This creates and switches to a new branch named feature/branch-name.

# 5. Making Changes

After creating a branch, they can make changes to the code. Once done, they need to stage and commit the changes:

git add .

git commit -m "Descriptive commit message"

# 6. Pushing the Branch

Push the newly created branch to the repository:

git push origin feature/branch-name

# 7. Creating a Pull Request (PR)

* Go to the GitHub repository page.
* Click **Pull Requests** > **New Pull Request**.
* Select their branch (e.g., feature/branch-name) and compare it with main.
* Add a description and assign reviewers if required.
* Submit the PR for review.

# 8. Reviewing and Merging PRs

* Teammates can review PRs, suggest changes, or approve them.
* After approval, the PR can be merged into the main branch by clicking **Merge Pull Request**.

# 9. Updating Local Branches

After merging, teammates should pull the latest changes to keep their local branches updated:

git pull origin main

# Common Git Commands Summary

| **Action** | **Command** |
| --- | --- |
| Clone a repository | git clone <repository-url> |
| Pull latest changes | git pull origin main |
| Create a new branch | git checkout -b branch-name |
| Switch branches | git checkout branch-name |
| Stage changes | git add . |
| Commit changes | git commit -m "message" |
| Push changes | git push origin branch-name |
| Update main branch locally | git pull origin main |