> Sunbeam

Git File Lifecycle (Stages)

In Git, **every file moves through well-defined stages** from its creation on a developer's machine to being stored on a remote repository such as GitHub. Understanding these stages is **fundamental** to using Git correctly and confidently.

Overview: The Four Main Stages

```
Working Directory → Staging Area → Local Repository → Remote Repository
```

Each stage represents a different level of tracking and permanence for your files.

---

1. Working Directory (Untracked / Modified)

What is the Working Directory?

- The actual **project folder** on your system
- Where developers **create, edit, or delete files**
- This is the default state when you open a project

File States in the Working Directory

1. Untracked

- A **new file** created in the project
- Git is **not tracking** this file yet
- Git has no record of this file in its history

Example:

```
index.html → untracked
```

---

2. Modified

- File is already **tracked by Git**
- Content has been **changed after the last commit**

Example:

```
style.css → modified
```

Commands Used

```
git status
```

This command shows:

- Untracked files
- Modified files
- Staged files

Key Point

> Changes in the working directory are **not saved in Git history** until they are staged and committed.

---

2. Staging Area (After `git add`)

What is the Staging Area?

- An **intermediate area** between working directory and local repository
- Used to **select which changes** will go into the next commit
- Also called the **Index**

What `git add` Does

```
git add file_name
```

or

```
git add .
```

- Moves selected changes from **Working Directory** → **Staging Area**
- Does **NOT** create a commit
- Only marks files as **ready to be committed**

File State

- **Staged**

Example

```
git add index.html
```

Current state:

```
index.html → staged
style.css → still modified
```

Why the Staging Area is Important

- Allows **partial commits**
- Helps create **clean and meaningful commit history**
- Widely used in **professional and team-based projects**

---

3. Local Repository (After `git commit`)

What is the Local Repository?

- Stored inside the hidden **.git** folder
- Contains the **complete commit history**
- Fully functional **without internet**

What `git commit` Does

```
git commit -m "Initial commit"
```

- Takes a **snapshot of staged files**
- Saves them permanently in the local repository
- Generates the following metadata:
  - Commit ID (hash)
  - Author name and email
  - Timestamp
  - Commit message

File State

- **Committed**

Important Rule

> Only **staged files** are committed. Unstaged or modified files are ignored during commit.

---

4. Remote Repository (After `git push`)

What is a Remote Repository?

- Repository hosted on platforms like:
  - GitHub
  - GitLab
  - Bitbucket
- Used for **sharing, collaboration, and backup**

What `git push` Does

```
git push origin main
```

- Sends local commits to the remote repository
- Makes the code available to:
  - Team members
  - CI/CD pipelines
  - Code review systems

File State

- **Published**

Complete Flow with Commands

```
# Working Directory
vim index.html

# Check current file status
git status

# Stage changes
git add index.html

# Commit to local repository
git commit -m "Add homepage structure"

# Push commits to GitHub
git push origin main
```

Visual Summary (Quick Revision)

```
Edit File
   ↓
Working Directory (Untracked / Modified)
   ↓ git add
Staging Area (Staged)
   ↓ git commit
Local Repository (Committed)
   ↓ git push
Remote Repository (GitHub)
```
