Git Lifecycle

Git Lifecycle

Following are the lifecycle stages of files in Git

Working Directory

Staging Area

Commit

Git Lifecycle

Working Directory

Staging Area

Commit

- The place where your project resides in your local disk
- This project may or may not be tracked by git
- In either case, the directory is called the working directory
- The project can be tracked by git, by using the command *git init*
- By doing *git init*, it automatically creates a hidden .git folder

Git Lifecycle

Working Directory

Staging Area

Commit

- Once we are in the working directory, we have to specify which files are to be tracked by git
- We do not specify all files to be tracked in git, because some files could be temporary data which is being generated while execution
- To add files in the staging area, we use the command *git add*

Git Lifecycle

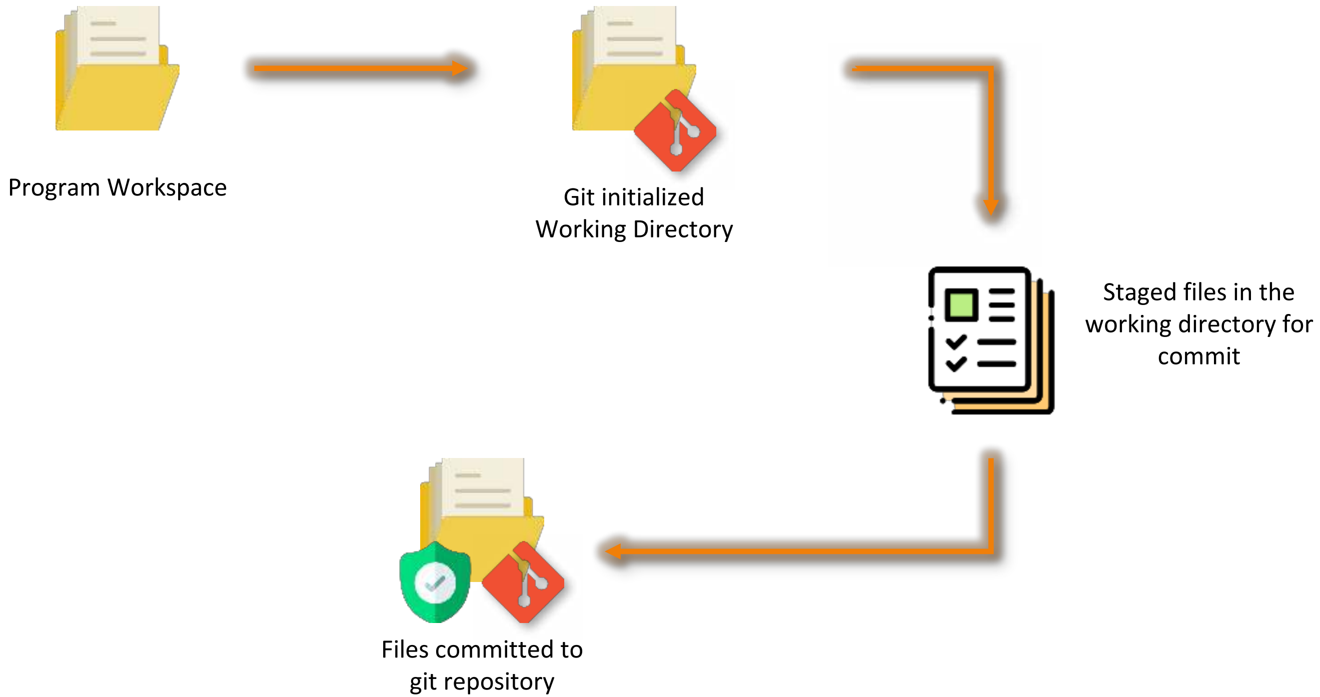
Working Directory

Staging Area

Commit

- Once the files are selected and are ready in the staging area, they can now be saved in repository
- Saving a file in the repository of git is known as doing a commit
- When we commit a repository in git, the commit is identified by a commit id
- The command for initializing this process is *git commit –m “message”*

Git Lifecycle



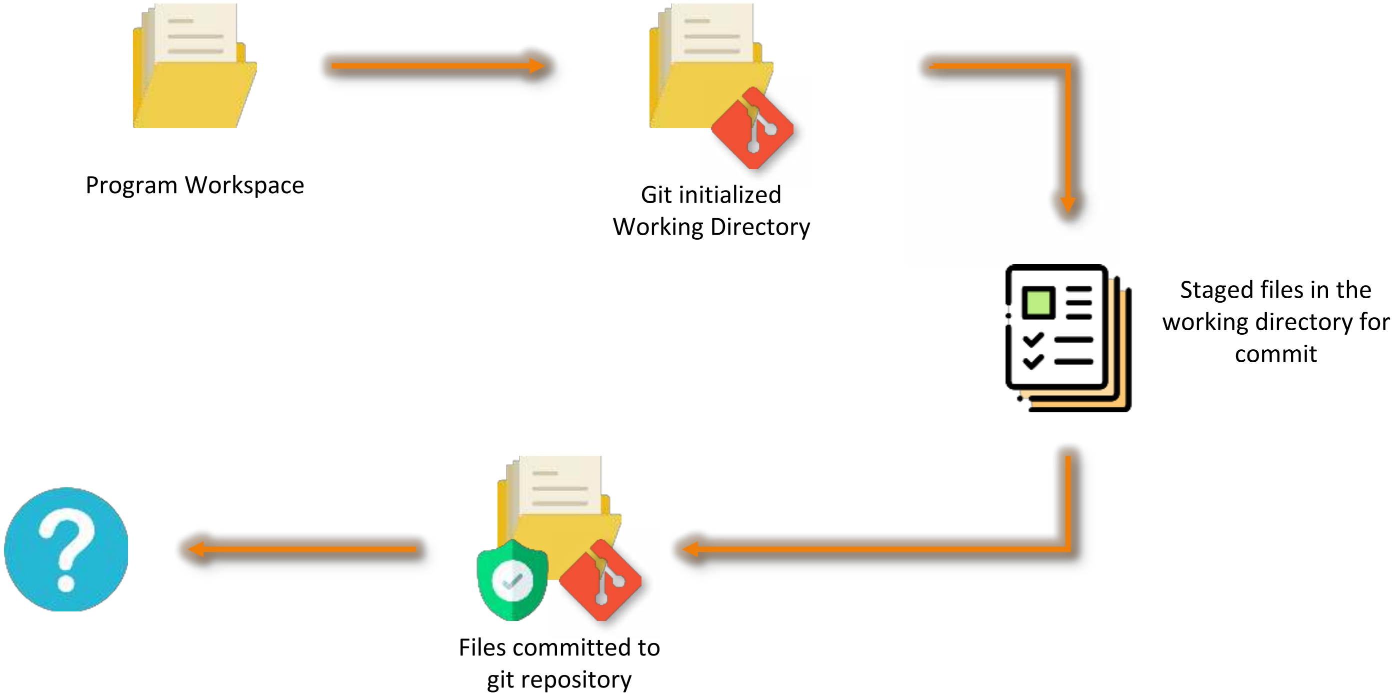
Program Workspace

Git initialized
Working Directory

Staged files in the
working directory for
commit

Files committed to
git repository

Git Lifecycle



Program Workspace

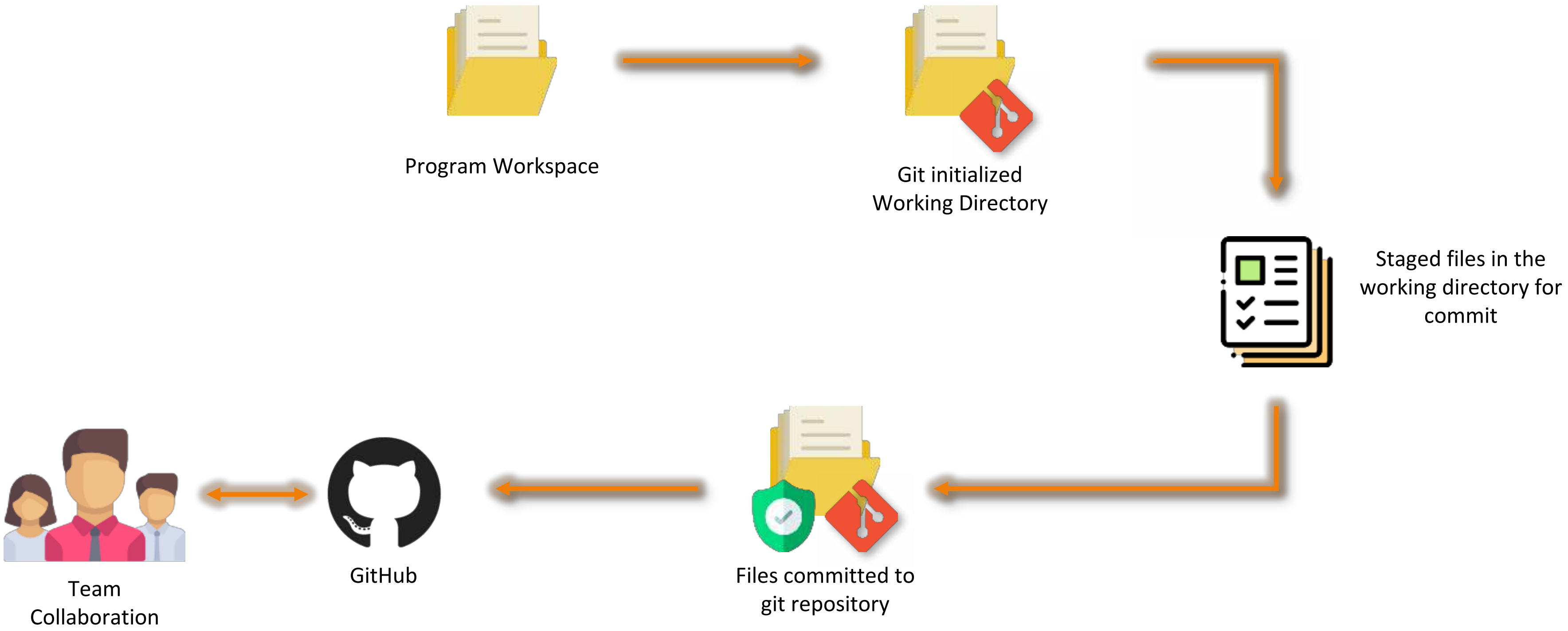
Git initialized Working Directory

Staged files in the working directory for commit

Files committed to git repository

How do we collaborate with the team?

Git Lifecycle

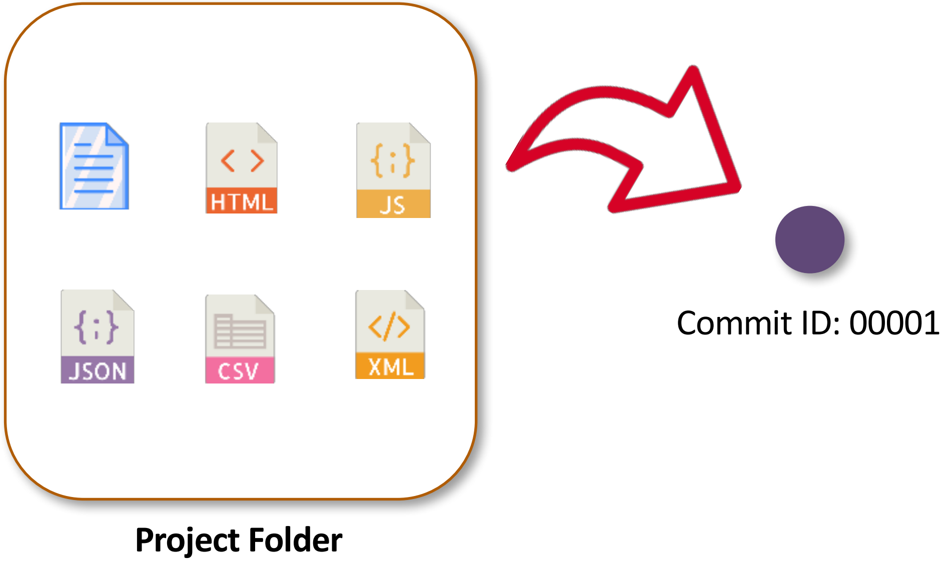


Program Workspace
Git initialized Working Directory
Staged files in the working directory for commit
Files committed to git repository
GitHub
Team Collaboration

Once the files are committed, they can be pushed to a remote repository such as GitHub

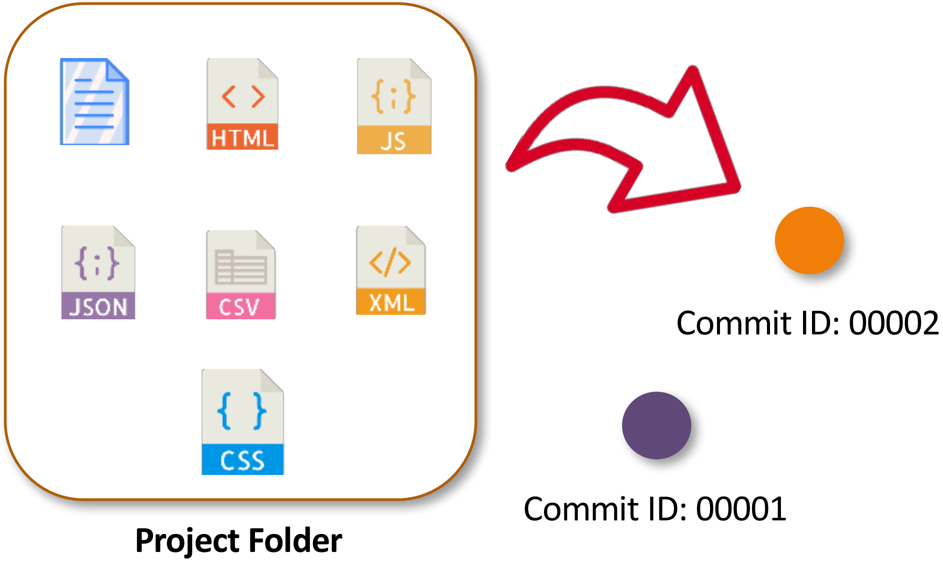
How does Git work?

Any project which is saved on git, is saved using a commit. The commit is identified using a commit ID.



How does Git work?

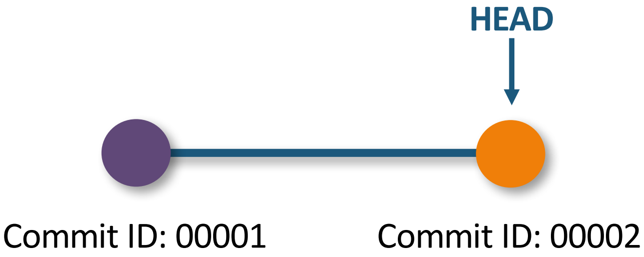
When we edit the project or add any new functionality, the new code is again committed to git, a new commit ID is assigned to this modified project. The older code is stored by git, and will be accessible by it’s assigned Commit ID



How does Git work?

All these commits are bound to a **branch.** Any new commits made will be added to this branch. A branch always points to the latest commit. The pointer to the latest commit is known as **HEAD**

Project Folder



HEAD

Commit ID: 00001

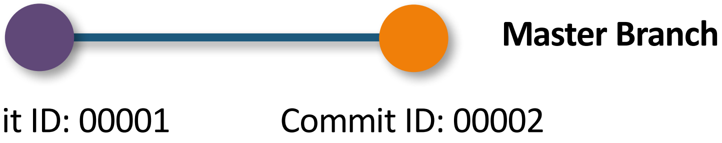
Commit ID: 00002

How does Git work?

The default branch in a git repository is called the Master Branch

HTML
JS
JSON
CSV
XML
CSS

Project Folder



Master Branch

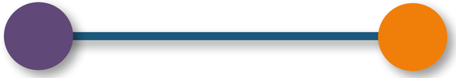
Commit ID: 00001

Commit ID: 00002

How does Git work?

The default branch in a git repository is called the Master Branch

Project Folder



Master Branch

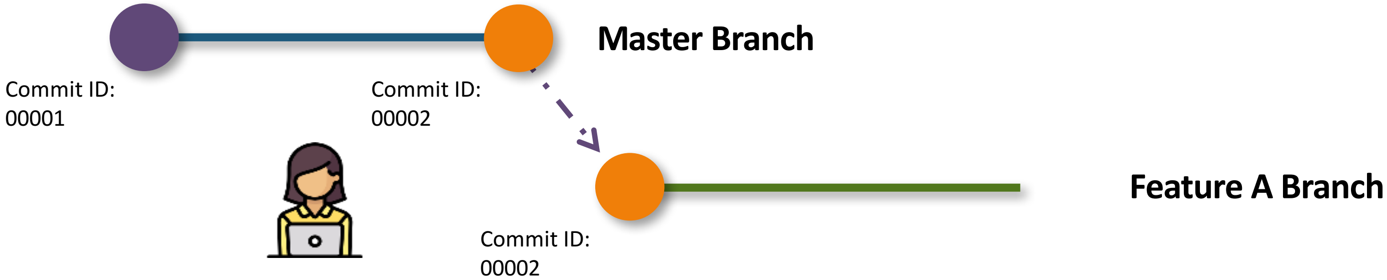
Commit ID: 00001

Commit ID: 00002

But, why do we need a branch?

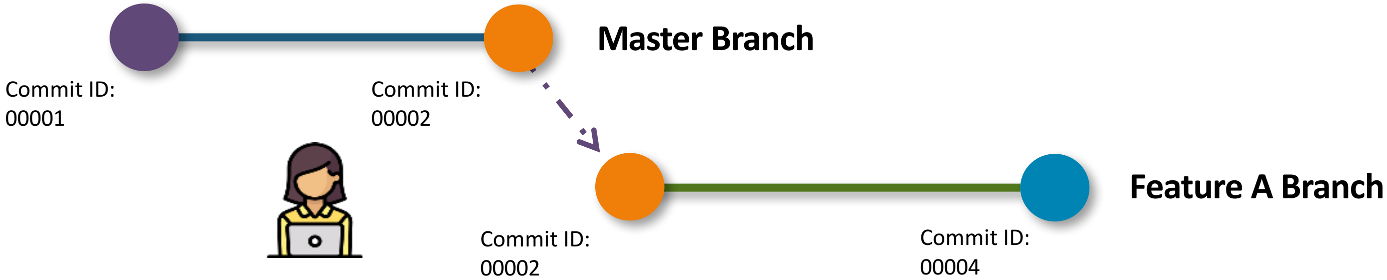
How does Git work?

Say, a developer has been assigned enhance this code by adding Feature A. The code is assigned to this developer in a separate branch “Feature A”. This is done, so that master contains only the code which is finished, finalized and is on production



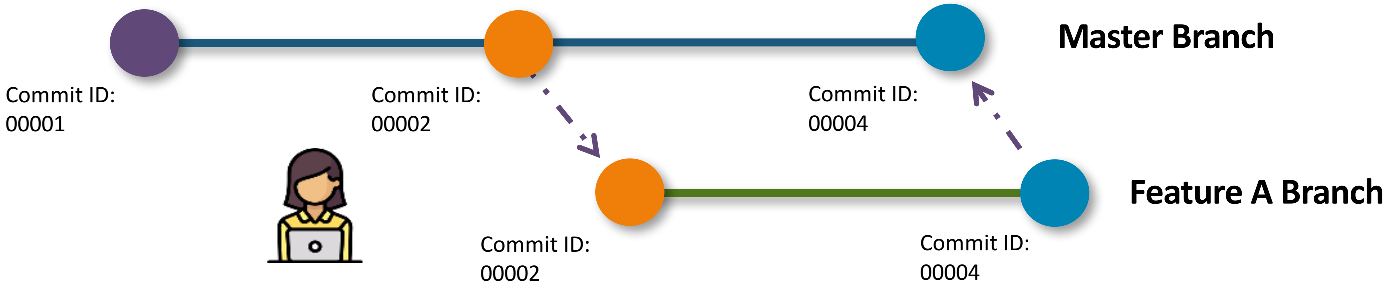
How does Git work?

Therefore, no matter how many commits are made by this developer on Feature A branch, it will not affect the Master Branch.



How does Git work?

Once the code is finished, tested and ready we can merge the Feature A branch, with the master branch and now the code is available on the production servers as well



☎ India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

✉ support@intellipaat.com

24X7 Chat with our Course Advisor