Created by Linus Torvalds in 2005

Git and Github: Essentials

Git is a free and open-source distributed version control system designed to track changes, manage project history, and support collaboration in software development. It allows developers to work independently using branches, merge changes seamlessly, and revert to previous versions when needed. Git powers platforms like GitHub, GitLab, and Bitbucket, making it essential for modern development workflows.

Setup and Config

git init

Initialize a new Git repository in a local directory,

creates a .git folder to track changes.

git config

The `git config` command lets you personalize your Git setup by defining global or local settings, such as your name, email, and preferred editor.

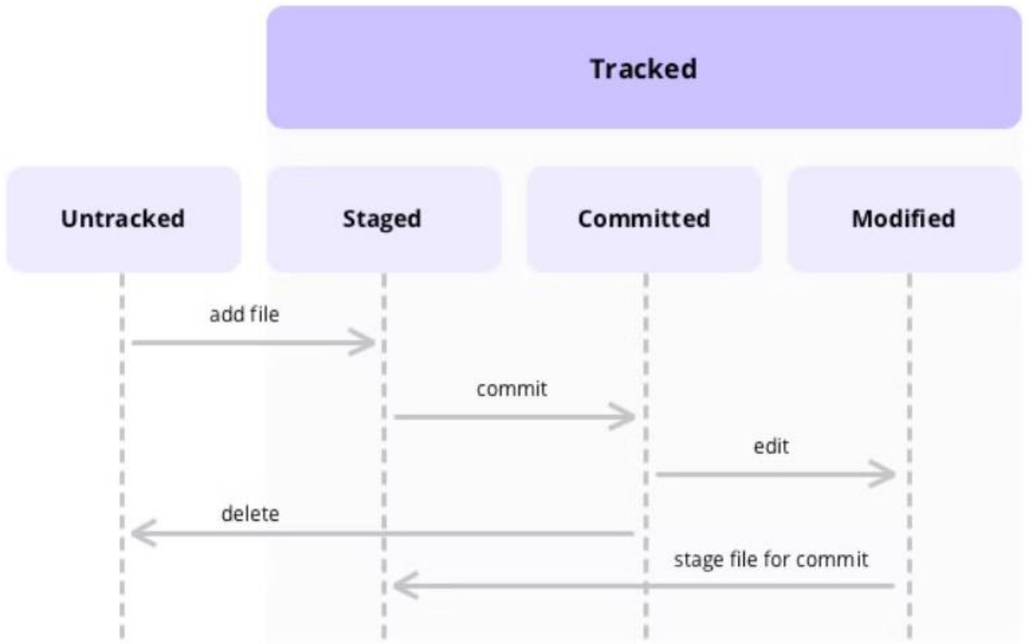
```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
git config --global core.editor "code --wait"
```

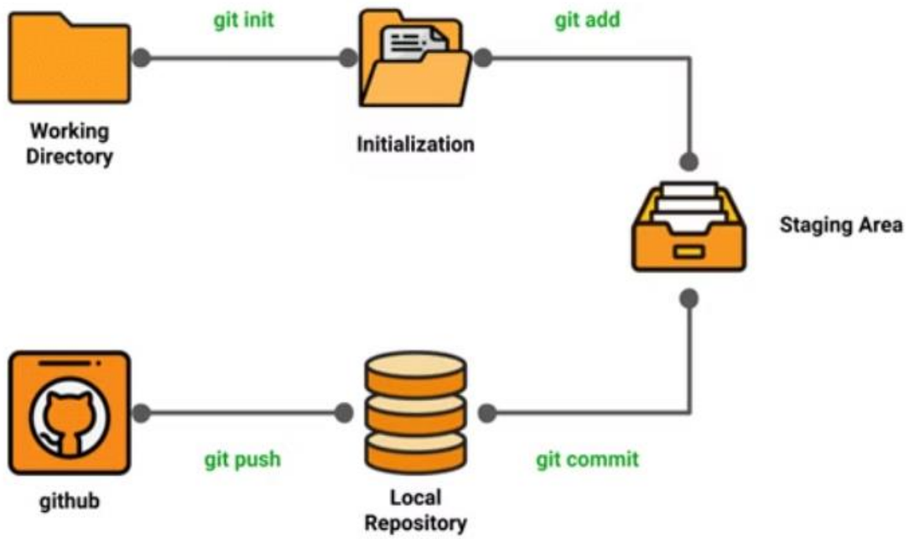
git help

The `git help` command provides access to comprehensive Git documentation

FILE STATUS LIFECYCLE

git status





FILE STATUS LIFECYCLE

Untracked

Files that are not being tracked by Git.

Staged

Files that have been added to the staging area.

Committed

Files that have been saved in the Git repository with a commit.

Modified

Files that have been changed after the last commit.

```
Untracked → Staged

          git add <file>

Staged → Committed

          git commit -m "message"

Modified → Staged

          git add <file>

Staged → Modified || Staged → Untracked

          git reset <file>
```

Git Cloning

The `git clone` command is used to create a local copy of a remote repository. It copies the entire repository, including its history, branches, and files, onto your local machine.

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

Cloning a Repository with a Custom Directory Name

  git clone <repository-url> <directory-name>

shallow clone

  git clone --depth <number-of-recent-commits> <repository-url>
```

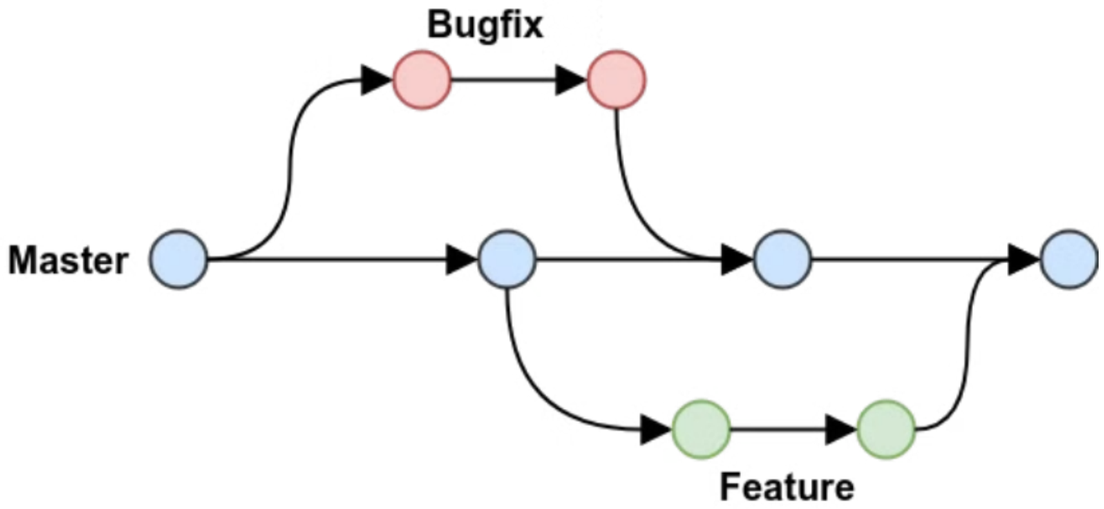
git Remote

The `git remote` command manages the set of repositories ("remotes") whose branches you track. It allows you to add, rename, remove, and configure remotes, as well as fetch their branches and update URLs.

```
git remote add origin <-link->
git remote -v
git remote remove <name>
git remote rename <oldname> <newname>
```

Branching

branching allows you to create and manage multiple lines of development in a project. Branches are lightweight, making it easy to switch between different versions of your code.



```
[check branches]                    =>  git branch

[Create a new branch]               =>  git branch <branch-name>
                                    =>  git checkout -b <branch-name>

[Switch to an existing branch]      =>  git checkout <branch-name>
                                    =>  git switch   <branch-name>

[Rename a branch]                   =>  git branch -m <newname>
                                    =>  git branch -m <oldname> <newname>

[Delete a branch]                   =>  git branch -d <branch-name>
```

Merging

Merging in Git is the process of combining changes from one branch into another.

```
git merge branchname
```

git diff

`git diff` shows changes between commits, the staging area, and branches

```
git diff --staged
git diff HEAD
git diff commit1 commit2
```

merge conflicts

An event that takes place when git is unable to automatically resolve differences in code between two commits

Git Push

The git push command uploads local changes to a remote repository . It sends the committed changes in local branch to the corresponding branch in the remote repository.

```
git push -u <remote> <branch>
```

git Pull

Git pull is used to **fetch** changes from a remote repository and automatically **merge** them into your local branch. It's a combination of fetch and merge .

```
git pull <remote> <branch>
```

Pull request

It tells about changes pushed to a branch in a repository on github

Forking

Forking is a process where you create a personal rough copy of someone else's repository

Fixing mistakes

```
staged -> unstaged
  => git reset <filename>
  => git restore --staged <filename>


Modified → Unmodified
  =>git restore <filename>

last commits
  =>git reset HEAD~1
  =>git checkout HEAD~2 (look at 2 commit prior)
  =>git checkout commitid


  => git checkout branchname
  => git reflog
```

Git stash

Git stash temporarily saves changes in your working directory. It's useful when you need to switch branches without committing

```
Stash changes                   ->  git stash
List stashes                    ->  git stash list
Apply the latest stash          ->  git stash apply
Apply the ith stash             ->  git stash apply "stash@{i}"
                                ->  git stash pop

Clear all stashes               ->  git stash clear
```