MASTER GIT in 60 Seconds

From Beginner To Advance

INTRODUCTION

What is Git?

Git is a distributed version control system that helps manage code changes efficiently and collaborate with others on software projects.

Installation

- Windows/Mac/Linux - Download Git
- Verify Installation

```
git --version
```

Basic Concepts

- Repository (Repo): Directory tracked by Git
- Commit: A snapshot of your code
- Branch: Independent line of development
- Clone: Copy of a repository from remote to local

STEP-BY-STEP GUIDE

Initialize Repository

```
git init
```

Add Files

```
git add < file_name>
git add. # Add all files in current directory
```

Commit Changes

```
git commit -m "initial commit"
```

Check Status

```
git status
```

View Commit History

```
git log
```

Set Git Identity

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

WORKING WITH REMOTE REPOSITORIES (GITHUB, GITLAB)

Clone Remote Repository

```
git clone <repository_URL>
```

Push Changes

```
git push origin <branch_name>
```

Pull Updates

```
git pull origin <branch_name>
```

BRANCHING & MERGING

Create New Branch

```
git branch <branch_name>
```

Switch Branches

```
git checkout <branch_name>
```

Create and Switch

```
git checkout -b <branch_name>
```

Merges Branch

```
git merge <branch_name>
```

Delete Branch

```
git branch -d <branch_name>
```

RESOLVING CONFLICTS

- If merging causes conflicts, Git will indicate it.
- Resolve manually, then -

```
git add <resolved_file>
git commit -m "Resolved conflict"
```

UNDOING CHANGES

Undo Last Commit(keep changes)

```
git reset --soft HEAD~1
```

Undo Last Commit(discard changes)

```
git reset --hardHEAD~1
```

Discard unstaged changes

```
git checkout -- <file_name>
```

ADVANCED GIT USAGE

Stashing Changes : Temporarily store changes without committing.

```
git stash
git stash pop
```

Rebase : Rewrite commit history

```
git checkout feature_branch git rebase main
```

Cherry-Pick : Apply a specific commit from one branch to another

```
git cherry-pick <commit_hash>
```

Tagging : Create tags to mark versions

```
git tag -a v1.0 -m "version 1.0 release" git push --tags
```

Amend Commit : Modify last commit message or content

```
git commit --amend -m "New commit message"
```

CLEANING REPOSITORY

Remove untracked files

```
git clean -f
```

Remove untracked directories too

```
git clean -fd
```

BEST PRACTICES

- Write meaningful commit messages.
- Commit often.
- Keep branches small and focused.
- Regularly sync branches with main.

EXPERT COMMANDS

Interactive Rebase : Squash, commits, reorder or edit history

```
git rebase -i HEAD~3
```

Submodules : Include external repositories within your repo

```
git submodule add <repo_url>
git submodule update --init --recursive
```

Bisect (Finding Bugs) : Identify the commit causing a bug

```
git bisect start
git bisect bad
git bisect good <good_commit_hash>
```

Aliases : Create shortcuts for commands

```
git config --global alias.st "status"
```

Hooks : Run scripts automatically at specific points

- pre-commit
- commit-msg
- pre-push

(Create custom scripts in .git/hooks/directory)