

GIT FUNDAMENTALS

TechRacine Academy

https://techracine.com

https://academy.techracine.com

Table of Contents

1. Introduction to Git 3
2. Why Git Matters 3
3. Understanding Version Control 3
4. Git Repository and Working Areas 3
5. Installing and Configuring Git 4
6. Basic Git Commands 4
7. Git Workflow 4
8. Staging and Committing Changes 4
9. Branches 5
10. Merging Branches 5
11. Remote Repositories 5
12. GitHub and Git 5
13. Cloning and Pulling 6
14. Pushing Changes 6
15. Pull Requests 6
16. Git Status and Git Log 6
17. Undoing Changes 6
18. .gitignore 7
19. Git Best Practices 7
20. Common Git Mistakes 7
21. Git in Team Development 8
22. Real World Usage 8
23. Summary 8
24. Exercises 8

1. Introduction to Git

Git is a distributed version control system used to track changes in source code and other project files.

It allows developers to save different versions of a project and return to earlier versions when needed.

Git is widely used in software development and supports both individual and team-based development.

2. Why Git Matters

Git helps developers track changes clearly and safely.

It makes collaboration easier because multiple developers can work on the same project.

Git also provides a history of changes, making it easier to identify when and why a change was made.

3. Understanding Version Control

Version control is a system for recording changes to files over time.

Git stores project history through commits, allowing developers to compare versions and restore earlier work.

Version control reduces the risk of losing important project changes.

4. Git Repository and Working Areas

A Git repository is the project location where Git stores version history and configuration.

The working directory contains the files currently being edited.

The staging area contains changes selected for the next commit.

The repository stores committed snapshots of the project.

5. Installing and Configuring Git

Git can be installed on Windows, macOS, and Linux.

After installation, developers commonly configure a username and email address.

Example:

```
git config --global user.name "Your Name"

git config --global user.email "you@example.com"
```

6. Basic Git Commands

Common commands include git init, git status, git add, git commit, git log, git branch, git merge, git clone, git pull, and git push.

These commands are used to create repositories, track changes, save versions, work with branches, and synchronize with remote repositories.

7. Git Workflow

A simple Git workflow is: Edit files → Check status → Stage changes → Commit changes → Push to remote repository.

This workflow helps developers keep project history organized.

8. Staging and Committing Changes

The staging area lets developers choose which changes should be included in a commit.

git add stages files, while git commit records the staged changes in the repository.

Example:

```
git add .

git commit -m "Add project files"
```

9. Branches

A branch is an independent line of development.

Branches allow developers to work on features or fixes without directly changing the main branch.

Example:

```
git branch feature-login

git switch feature-login
```

10. Merging Branches

Merging combines changes from one branch into another branch.

Developers should review conflicts carefully when two branches modify the same part of a file.

Example:

```
git merge feature-login
```

11. Remote Repositories

A remote repository is a Git repository stored on another system or service.

Remote repositories make it possible to share code and synchronize work between developers.

12. GitHub and Git

Git is the version control system, while GitHub is a platform that hosts Git repositories and provides collaboration features.

GitHub can be used for repository hosting, pull requests, issue tracking, and team collaboration.

13. Cloning and Pulling

git clone creates a local copy of a remote repository.

git pull downloads new changes from a remote repository and integrates them into the current branch.

Example:

```
git clone <repository-url>

git pull origin main
```

14. Pushing Changes

git push uploads local commits to a remote repository.

Developers normally commit changes locally before pushing them.

Example:

```
git push origin main
```

15. Pull Requests

A pull request is a request to review and merge changes from one branch into another.

Pull requests support code review, discussion, testing, and controlled integration of changes.

16. Git Status and Git Log

git status shows the current state of the working directory and staging area.

git log displays the history of commits.

These commands are useful for understanding what has changed and what has already been saved.

17. Undoing Changes

Git provides several ways to undo or modify changes depending on whether the changes are unstaged, staged, or committed.

Commands such as git restore, git reset, and git revert should be used carefully because they have different effects.

18. .gitignore

The .gitignore file specifies files and folders that Git should not track.

It is commonly used for temporary files, build output, environment files, and dependency folders.

Example:

```
node_modules/
.env
*.log
```

19. Git Best Practices

Use clear and meaningful commit messages.

Commit small, logical changes.

Create branches for separate features or fixes.

Pull recent changes before starting collaborative work.

Review changes before committing and pushing.

20. Common Git Mistakes

Avoid committing sensitive information such as passwords or private keys.

Do not use unclear commit messages.

Do not ignore merge conflicts without reviewing them.

Avoid force-pushing shared branches unless the team has agreed to it.

21. Git in Team Development

Git supports collaborative development by allowing team members to create branches, review changes, merge work, and maintain project history.

Teams often use a shared remote repository and pull requests to coordinate development.

22. Real World Usage

Git is commonly used in web applications, mobile applications, backend systems, open-source projects, DevOps workflows, and enterprise software development.

It is also frequently integrated with CI/CD systems to automatically test and deploy changes.

23. Summary

Git is an essential tool for tracking and managing changes in software projects.

Understanding repositories, staging, commits, branches, remotes, merging, and basic Git commands provides a strong foundation for collaborative development.

Regular practice with Git commands helps developers work more safely and efficiently.

24. Exercises

1. Install Git and configure your username and email.

2. Create a new Git repository for a simple project.

3. Add files and create your first commit.

4. Create a feature branch and make a change.

5. Merge the feature branch into the main branch.

6. Create a remote repository and connect it to your local repository.

7. Push your commits to the remote repository.

8. Clone a repository and inspect its files.

9. Practice git status and git log.

10. Create a .gitignore file and exclude temporary files.