code/.
{craft}

Git Essentials Handbook

BY
KARTHIK KRISHNAN GS

Introduction

In today's world of software development, collaboration and version control are not just optional—they are essential. Whether you are a student writing your first program, a professional developer working in a large team, or an open-source contributor, managing code effectively is the foundation of every successful project.

This book, **"Git Essentials Handbook"**, is designed to guide you through one of the most powerful and widely used version control systems: **Git**. Unlike traditional manuals that dive deep into every technical detail, this handbook focuses on practical knowledge and hands-on usage. It is structured to give you a clear understanding of Git concepts, commands, and workflows that you can immediately apply to your projects.

You will start by learning the basics—such as installing Git, understanding repositories, and tracking changes. From there, you'll gradually explore branching, merging, collaboration with remote repositories, and more. By the end, you will not only be comfortable using Git but will also gain confidence in working on real-world projects with teams, contributing to open source, and managing your own codebases effectively.

The goal of this handbook is simple:

- **Clarity over complexity** – Concepts explained in simple, straightforward language.
- **Practical over theoretical** – Real examples and workflows that you can try yourself.
- **Essentials over overload** – Focus on the core features that matter the most for everyday development.

Whether you are a beginner taking your first steps into version control or an intermediate developer seeking to sharpen your Git skills, this handbook will serve as your reliable companion. Keep it at your desk, carry it with your laptop, and let it be your quick reference whenever you need a Git command at your fingertips.

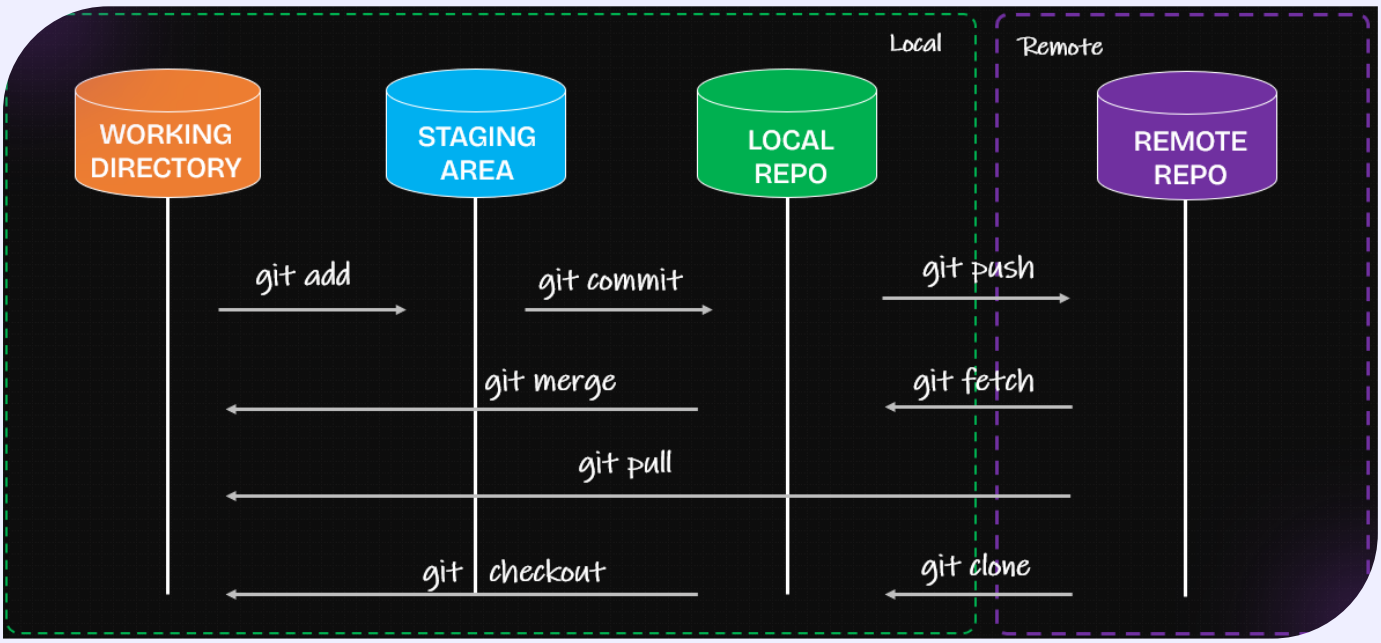
Let’s begin the journey of mastering Essential Git—one command at a time.

Why use git?

Git is a version control system that allows you to track changes in your code and collaborate with others efficiently. It keeps a history of your work, making it easy to revert changes if something goes wrong.

Using Git brings numerous advantages that enhance the development process for both individual developers and teams.

How git works?



Essential Commands

Check git version

```
git --version
```

Displays the currently installed version of Git.

Initialize a git repository

```
git init
```

Initializes a new Git repository. This command creates a new Git repository in the current directory. It sets up the basic files and directories needed to start tracking changes

Clone repository

```
git clone [repository URL]
```

Clones an existing Git repository. This command creates a copy of an existing repository on your local machine

Add file to staging

```
git add [file/direc tory]
```

Adds a file or directory to the staging area. This command prepares the changes for the next commit. It adds the specified file or directory to the index.

TIP : Use ` .` for adding all the files in the directory.

Commit with a message

```
git commit -m “[ commit message]”
```

Creates a new commit with a message describing the changes made. This command creates a new commit with the changes you made to your local repository. The commit message describes the changes made in this commit.

Pull changes

```
git pull
```

Updates the local repository with changes from the remote repository. It pulls the changes from the remote repository and merges them with the local changes.

Push changes

```
git push
```

This command pushes the local changes into the remote repository .

Check status

```
git status
```

Shows the current status of the repository. This command shows the status of the repository and the changes that are currently staged or unstaged.

List branches

```
git branch
```

This command lists all the branches in the current repository. It shows the current branch you're on and highlights it with an asterisk.

Switch a branch

```
git checkout [branch name]
```

This command switches to the specified branch. It updates the working directory to match the contents of the specified branch.

Merge a branch

```
git merge [branch name]
```

This command merges the specified branch into the current branch. It combines the changes from both branches and creates a new commit.

List all commits

```
git log
```

This command shows a list of all commits in the repository. It displays the author, date, and commit message for each commit the current repository.

List remote repositories

```
git remote -v
```

This command lists all the remote repositories associated with the local repository. It shows the URL of each remote repository.

Check difference

```
git diff [file]
```

This command shows the differences between the working directory and the staging area or the repository. It displays the changes made to the specified file.

Fetch

```
git fetch
```

This command downloads the changes made in the remote repository and updates your local repository, but it does not merge the changes with your local branch.

Reset changes

```
git rest [file]
```

This removes the specified file from the staging area, effectively undoing any changes made to the file since the last commit. It does not delete the changes made to a file.

Revert changes

```
git revert [commit]
```

Creates a new commit that undoes the changes made in the specified commit. It does not delete the specified commit, but it creates a new commit that reverts the changes made in that commit.

Final note

Congratulations! 🎉 By reaching this page, you've completed your journey through the **Git Essentials Handbook**. You've learned how to create repositories, track changes, branch, and more.

But remember: **Git is not just a tool, it's a habit.**
The more you use it in your daily workflow, the more natural it will feel. Mistakes will happen—commits may go wrong, branches may clash—but every challenge is an opportunity to sharpen your skills.

Here are a few steps you can take from here:

- **Practice daily** – Use Git in all your projects, big or small.
- **Explore deeper** – Learn advanced topics like rebasing, submodules, and Git hooks.
- **Collaborate** – Join open-source projects on GitHub, GitLab, or Bitbucket.
- **Teach others** – Share your knowledge; explaining Git will make you master it.

As you move forward, remember that Git is more than just commands—it's about **teamwork, efficiency, and confidence** in managing code. Let this handbook remain your quick companion, but never stop experimenting, learning, and growing as a developer.

Thank you for choosing this book as your guide. May every git commit you make bring you closer to becoming the developer you aspire to be.

Keep coding. Keep committing. Keep creating.

— *Karthik Krishnan*

code/.
{craft}