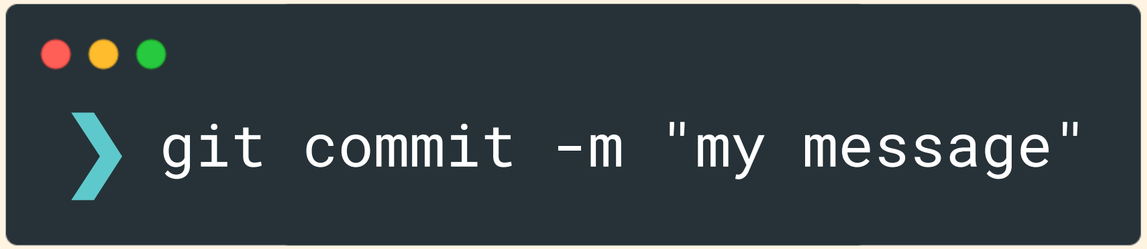
More About Committing

& Other Stuff

Git Commit

The -m flag allows us to pass in an inline commit message, rather than launching a text editor.

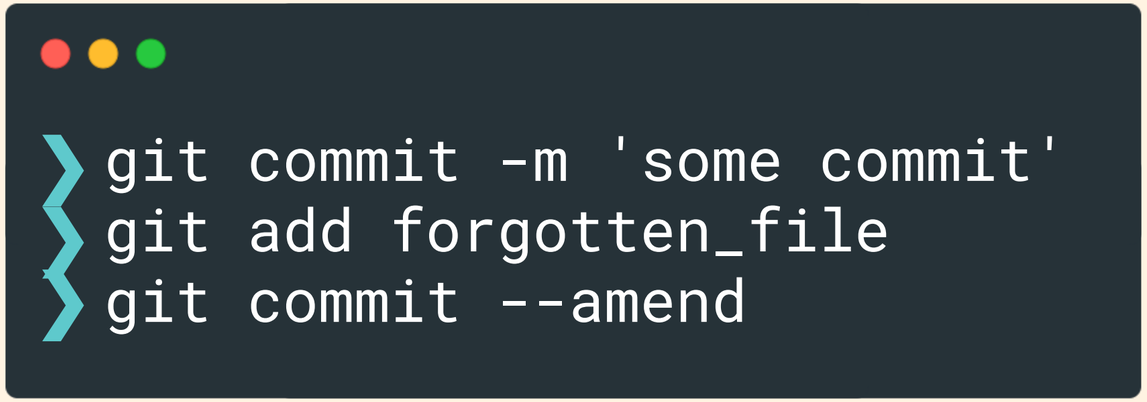
We'll learn more about writing good commit messages later on.



Amending Commits

Suppose you just made a commit and then realized you forgot to include a file! Or, maybe you made a typo in the commit message that you want to correct.

Rather than making a brand new separate commit, you can "redo" the previous commit using the --amend option



Some Basic Guidelines

- Commit early and often

- Make commits atomic (group similar changes together, don't commit a million things at once)

- Write meaningful but concise commit messages

The Git Docs

The Git Docs


The Git Docs

Writing Commit Messages

Present or past tense? Does it really matter?

Present-Tense Imperative Style??

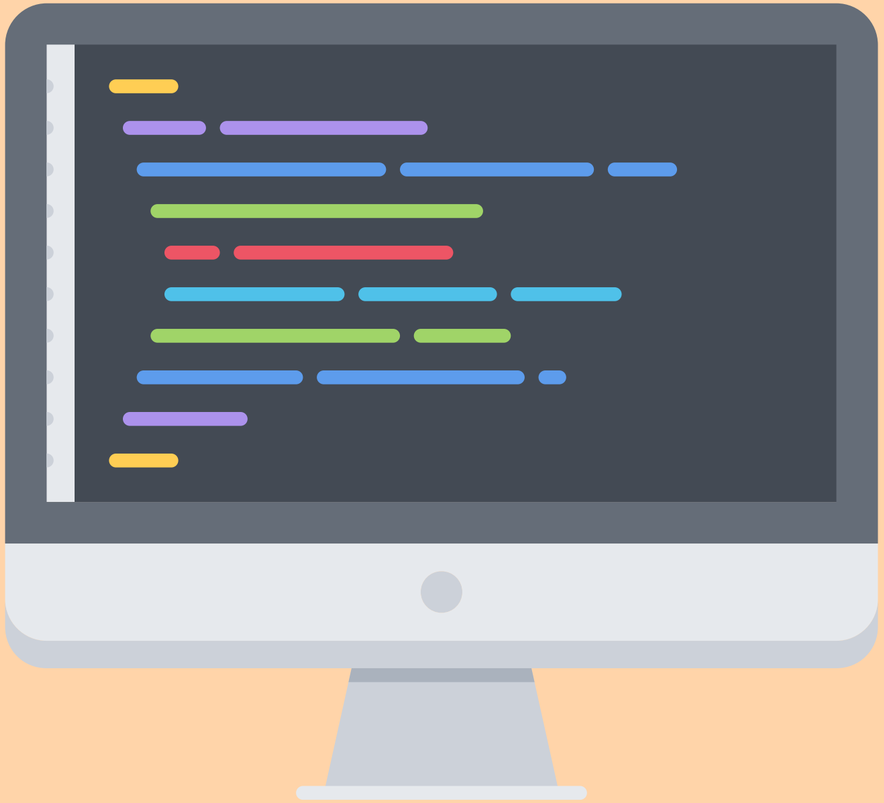
From the Git docs:

> Describe your changes in imperative mood, e.g. "make xyzzy do frotz" instead of "[This patch] makes xyzzy do frotz" or "I changed xyzzy to do frotz", as if you are giving orders to the codebase to change its behavior.

You do NOT have to follow this pattern

Though the Git docs suggest using present-tense imperative messages, many developers prefer to use past-tense messages. All that matters is consistency, especially when working on a team with many people making commits

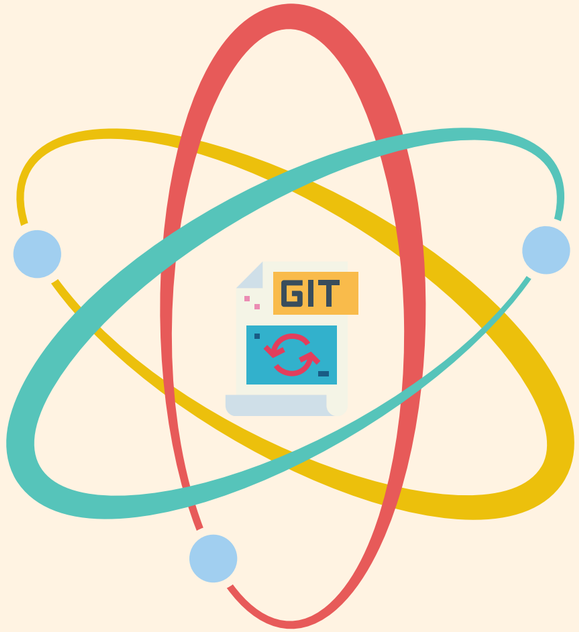
Configuring Your Default Editor



Atomic Commits

When possible, a commit should encompass a single feature, change, or fix. In other words, try to **keep each commit focused on a single thing**.

This makes it much easier to undo or rollback changes later on. It also makes your code or project easier to review.



Ignoring Files

We can tell Git which files and directories to ignore in a given repository, using a **.gitignore** file. This is useful for files you know you NEVER want to commit, including:

- Secrets, API keys, credentials, etc.
- Operating System files (.DS_Store on Mac)
- Log files
- Dependencies & packages

.gitignore

Create a file called .gitignore in the root of a repository. Inside the file, we can write patterns to tell Git which files & folders to ignore:

- .DS_Store will ignore files named .DS_Store
- folderName/ will ignore an entire directory
- *.log will ignore any files with the .log extension