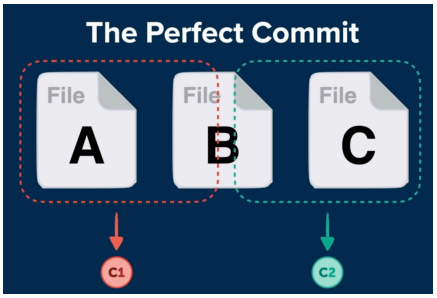
Git for Professionals

1. The Perfect Commit



Include 1 Topic Per Commit

Look for files that have been modified (git status), then:
- **Add related files**: git add <filename>
- **Add related parts from each file**: git add -p <filename>

Include The Perfect Commit Message

Subject: Concise summary of what happened. (*Convention*: Use present tense, not past tense)
Body: More detailed explanation

---

2. Branching Strategy

Long-Running Branches ("main")
- Exist through the complete lifetime of the project.
- NO DIRECT COMMITS. Requires testing and code reviewing before integration.

Short-Lived Branches
- For new features, bug fixes, refactoring and experiments.
- Usually deleted after integration (merge/rebase).

---

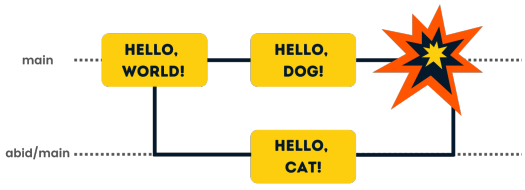
3. Pull Requests

With pull requests, you are inviting other people to review and provide feedback before merging. After a conversation, your change can be merged to one of the branches.

To contribute to a repository that you don't have access to:
1. Create a "**Fork**" of the original repository, where you can make changes.
2. Suggest the changes to be included via a **Pull Request**.

---

4. Merge Conflicts



- **When they occur**: A version of a file has been submitted that is newer than the version of file you have started to base your changes on.
- **How to solve**: Make one or several merge operations. Alternatively, you can always undo your or the other developer's changes. Collaborate to get the problem solved.

- Usually, merges are automatically handled by Git.

---

5. Useful Commands:

- git branch – What your current branch is.
- git checkout <branch_name> – Switch to a different branch.
- git checkout -b <branch_name> – Create a new branch and switch to it.