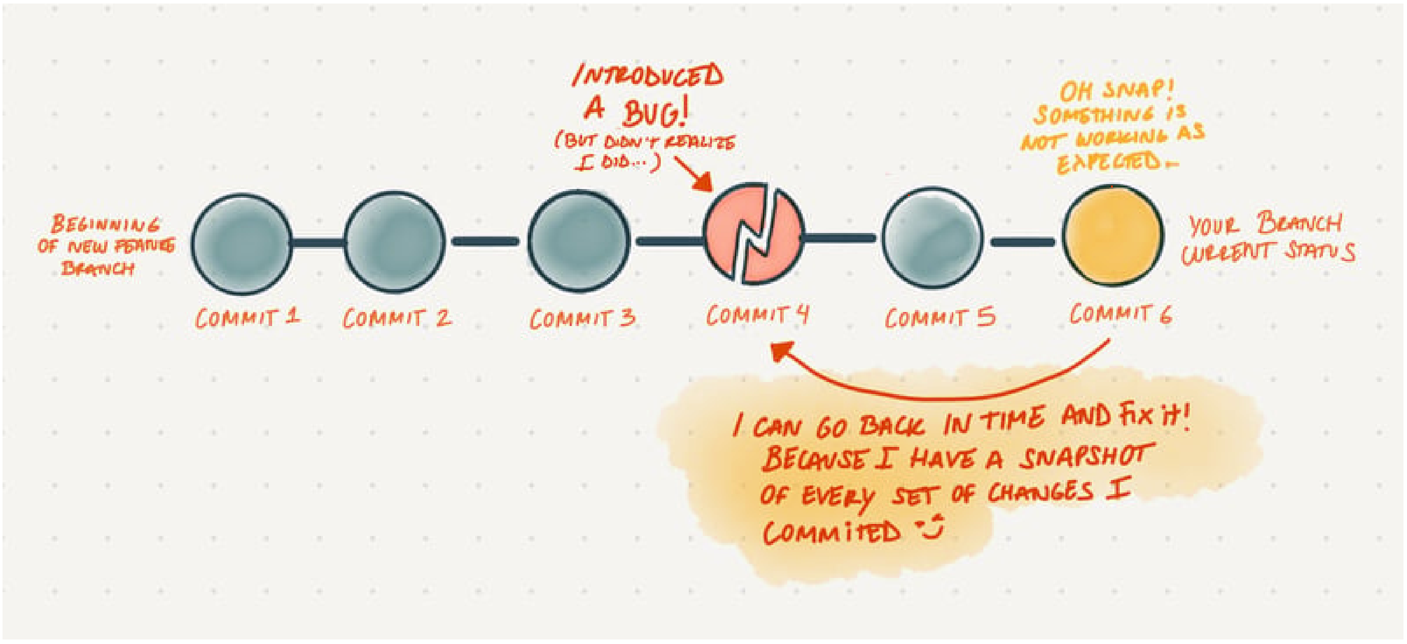
What is Git?

Git is a Version Control System Software. A version control keeps track of every change in your code and it allows you to go back in time when something goes wrong.


INTRODUCED A BUG! (BUT DIDN'T REALIZE I DID...)
OH SNAP! SOMETHING IS NOT WORKING AS EXPECTED
BEGINNING OF NEW FEATURE BRANCH
YOUR BRANCH CURRENT STATUS
COMMIT 1
COMMIT 2
COMMIT 3
COMMIT 4
COMMIT 5
COMMIT 6
I CAN GO BACK IN TIME AND FIX IT! BECAUSE I HAVE A SNAPSHOT OF EVERY SET OF CHANGES I COMMITED

Why Use Git?

- Prevents overwriting changes when multiple people work on the same file.
- Keeps a history of changes, making it easy to debug issues.
- Facilitates collaboration with branches and merging.

Git File Status

1. Untracked

- Files that are not yet being tracked by Git.
- **Example:** A new file created in the working directory.
- **Command to Track:**

```
git add <file_name>
```

or

```
git add .
```

to add all files in the directory.

2. Tracked

- Files that are being tracked by Git, can have three states:
- **Unmodified:** Files that have not been changed since the last commit.
- **Modified:** Files that have been changed since the last commit, this file is needed to be staged with `git add` command again.
- **Staged:** Files that have been added to the staging area and are ready to be committed.

3. Commited

- Files that have been saved to the Git database.
- Command to commit:

  ```
  git commit -m "Commit message"
  ```

4. Ignored

- Files that are set to be ignored/not tracked by Git.
- Can be set in a `.gitignore` file, for example if there is a file named `secret.txt` that you don't want to track, you can add it to the `.gitignore` file.

  ```
  secret.txt
  ```

  This way `secret.txt` will not be staged or committed.

Common Git Commands Cycle

Command	Description
`git add <file_name>`	Add a file to the staging area
`git add .`	Add all files to the staging area
`git commit -m "Commit message"`	Commit the staged files
`git push origin <branch_name>`	Push the committed changes to a remote repository

Git Branch

A branch is a separate line of development, just like a `multiverse`. It allows you to work on a feature without affecting the main codebase. You can create a new branch with the following command:

```
git branch <branch_name>
```

to show all branches and the current branch:

```
git branch
```

to switch to a branch:

```
git checkout <branch_name>
```

Git Merge

Merging is the process of combining changes from different branches. To merge a branch into your current branch you can do:

```
git merge <branch_name>
```

Git Conflict

A conflict is a situation where two branches have made changes to the same line in a file. If that happen, git didn't know which change to keep, so it will ask you to resolve the conflict manually. You can resolve the conflict by editing the file and removing the conflict markers `<<<<<<<`, `=======`, `>>>>>>>`.

example of a conflict:

```
<<<<<<< HEAD
This is the change in the current branch
=======
This is the change in the branch you are merging
>>>>>>> branch_name
```

Revision #3
Created 23 November 2024 05:59:06 by CH
Updated 24 November 2024 04:07:47 by CH