Useful GIT Commands

Commit Changes

- Commit the staged snapshot. This will launch a text editor prompting you for a commit message. After you've entered a message, save the file and close the editor to create the actual commit.

```
git commit
```

- Commit a snapshot of all changes in the working directory. This only includes modifications to tracked files (those that have been added with git add at some point in their history).

```
git commit -a
```

Commit Changes

- A shortcut command that immediately creates a commit with a passed commit message. By default, git commit will open up the locally configured text editor, and prompt for a commit message to be entered. Passing the -m option will forgo the text editor prompt in-favor of an inline message.

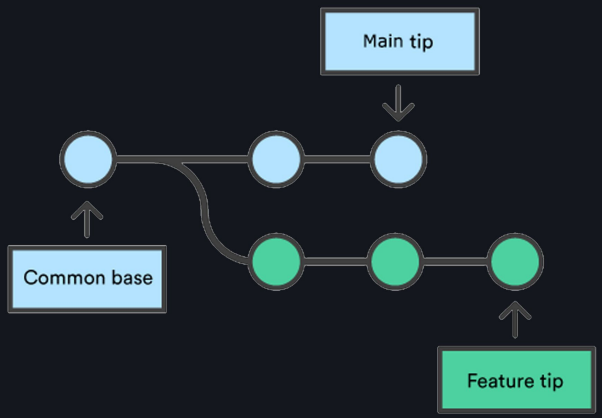
```
git commit -m "commit message"
```

- A power user shortcut command that combines the -a and -m options. This combination immediately creates a commit of all the staged changes and takes an inline commit message.

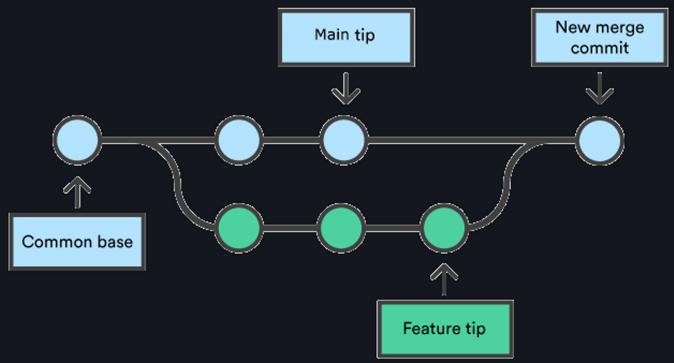
```
git commit -am "commit message"
```

Merge Branches

Say we have a new branch feature that is based off the main branch. We now want to merge this feature branch into main.



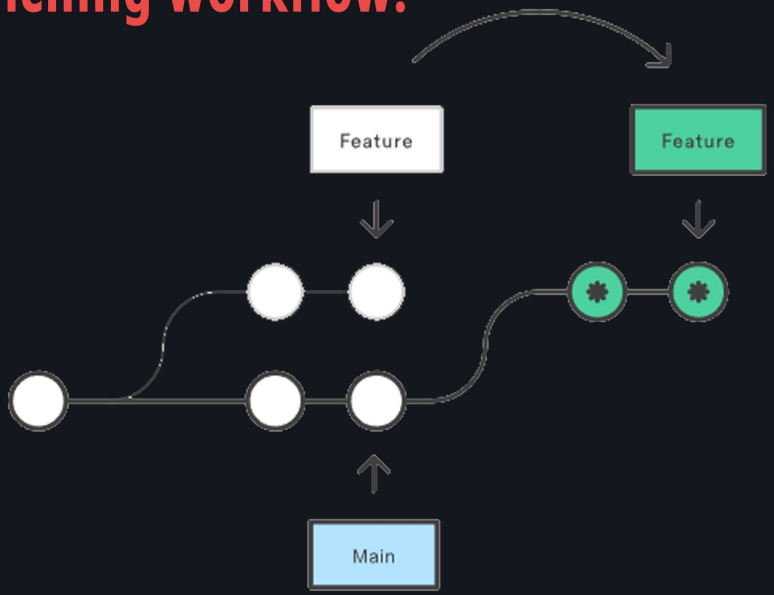
Invoking this command will merge the specified branch feature into the current branch, we'll assume main. Git will determine the merge algorithm automatically.




```
git checkout main
git merge new-feature
git branch -d new-feature
```

Rebase

Rebasing is the process of moving or combining a sequence of commits to a new base commit. Rebasing is most useful and easily visualized in the context of a feature branching workflow.




```
git rebase <base>
```

DID YOU FIND THIS POST USEFUL?

Let me know in the Comments below!