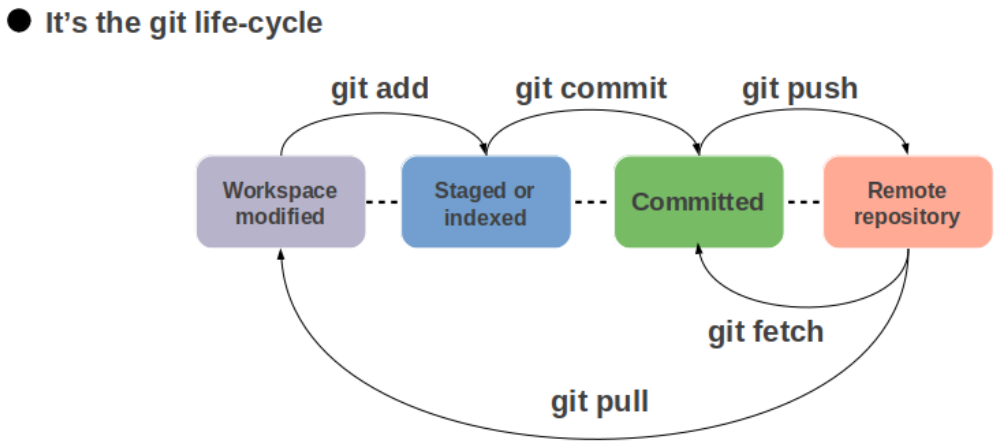
01 Git

April 28, 2022

1 Git Basics: Creating Git Repos and Committing Changes to Version Control

Git is a version control system; when added to a project folder *("initializing a repository")*, it has commands that let you add files to be tracked *("staging a file")*, and save a snapshot of your project in a moment in time *("committing a change")*. You can always ask Git what the current state of that process is *("check git's status")*, and what the history of the project looks like *("check git's log")*.

In this first section, we'll practice the cycle of changing a file, adding it to the list of changes to be committed, and committing it. We'll get practice with doing it in the terminal and in a VSCode-like IDE.



1.1 Terminal Commands

- `git status`: Checks what changes are detected by git, what changes are ready to be committed. Used all the time!
- `git init`: Creates a new git repository. Used only once, at the beginning of a project.
- `git add afile.txt`: Adds a file and its changes to be committed. Done before every commit
  - `git add *`: Shortcut: Adds all files in the project to be committed.
- `git commit -m "my commit message"`: Makes a commit with all added files' changes.
  - `git commit -am "my commit message"`: Shortcut: Adds all and Commits, in one step.
- `git log`: Checks the log of all past commits. **To exit the log, press the `q` key.**
  - `git log -1`: Shortcut, only shows the last one commit.

- `git diff`: Compares the files now to the previous commit, showing any differences.
- `ls -la` **or** `ll` **or** `dir`: Print all the files in the current folder. Command depends on operating system.
- `git config --global user.name "My name"`: Tell git which name to sign your commits with (only need to do this the first time git is installed)
- `git config --global user.email "myemail@email.com@`: Tell git which email to sign your commits with (only need to do this the first time git is installed)

1.2 Exercise

Let's create a git repository and make a few commits, observing how git's status changes along the way. Follow the sequence below, using the commands in the reference above.

Sequence

1: Create a new project, and create a file called README.md.

Check git's status for the repository. Is the readme file being tracked by git yet?

2: Use the `ls -la` (mac, linux) or `ll` (ubuntu) command to list all files and folders, including hidden folders. Make sure you see the readme file in this list. Are there any files/folders besides the readme file in this project folder?

3: Initialize a new git repo in the project folder, then list the files again. Are there anything new here?

Check git's status for the repository. Is the readme file being tracked by git yet?

Current State: At this point, you have initialized a new git repository. There aren't any commits in it yet, and no files are being tracked, but it's a start!

4: Add the readme file to git's tracking, including its data in the next commit.

Check git's status for the repository. Is the readme file being tracked by git yet?

5: Make a commit, using the message "created readme file"

Check git's status for the repository. Is the readme file being tracked by git yet?

Current State: At this point, you've made your first commit! In addition, whenever you type `git status`, any differences git notices in that Readme file will automatically be displayed.

6: Make a change to the readme file, then ask git what the differences are between the current project and the project in the last commit. Do you see your changes in this list?

Check git's status for the repository. Is the readme file's changes being tracked by git yet?

7: Make a commit, using the message "changed readme file"

Check git's status for the repository. Is the readme file's changes being tracked by git yet?

8: Look at git's log, and check the messages. Do you see all of your commits in this list?

Current State: Now you've started to build a history of changes in the file! Every commit will be stored in the log. In the next exercise, we'll use this log to go back to those previous versions.

9: Create a new file called **main.py**, and write in the python code for printing hello world.

10: In your Readme file, add the line "To run the code in this project, type `python main.py`

Check git's status for the repository. Is the main.py file's changes being tracked by git yet?

11: Add both the python file and the readme file to git for the next commit, and check if git's status message shows it is ready to be added.

12: Make a commit, with the message "added main.py"

Current State: At this point, you have multiple files being tracked by git. Changes to multiple files can be simultaneously committed, which is valuable when you make changes that rely on each other!

Make some more commits, this time using the graphical interface. For every command, there should be a button that relates to it–add, commit, etc.

[ ]: