

GIT & GITHUB



What is **Git**?



A "Version Control System".

Git is used to save and backup work, share your code and collaborate on projects.

Does this **Look Familiar?**

Any term/school paper you've ever worked on:

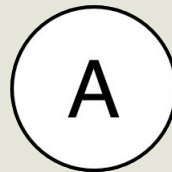
- `term_paper.docx`
- `term_paper2.docx`
- `term_paper2_with_footnotes.docx`
- `final_term_paper.docx`
- `final_term_paper_draft2.docx`
- `term_paper_for_submission.docx`
- `term_paper_for_submission_for_real_this_time.docx`

A Version Control System such as Git alleviates the above nightmare.

Snapshots in Time

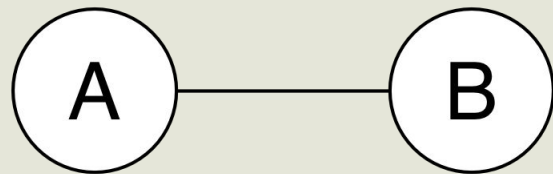
Git uses “commits” to represent each successive version of a file or files.

Commits are the Git equivalent of “Save As...”



Snapshots in Time

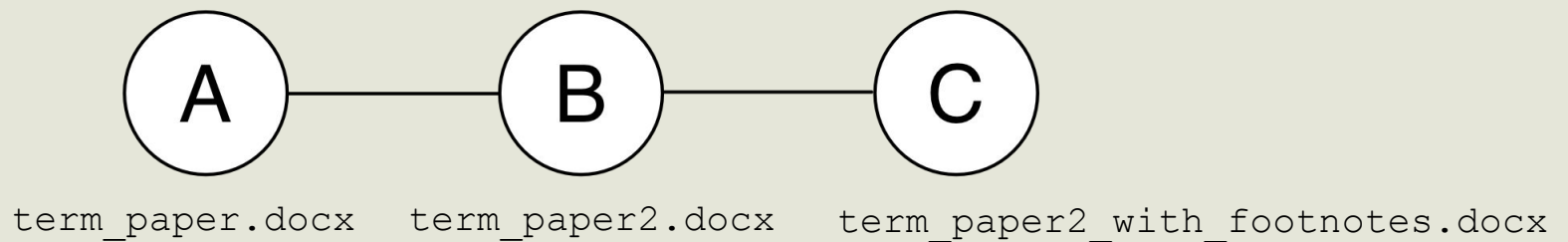
Each successive version creates a new snapshot on the timeline of the project.



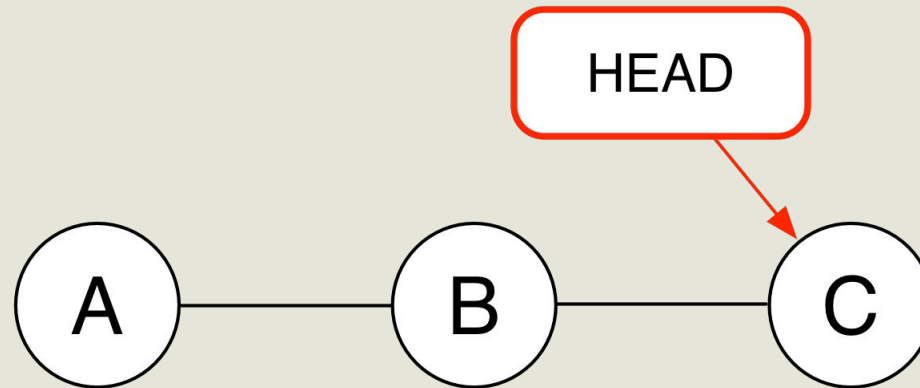
Snapshots in Time

As we continue to update and revise the files in our project Git helps us keep track of where we are and where we've been.

Think of each snapshot, or “commit” as each new version of the paper you were saving. However, instead of making a full copy of every file, it only keeps track of the actual differences between each version, making it very efficient and fast.



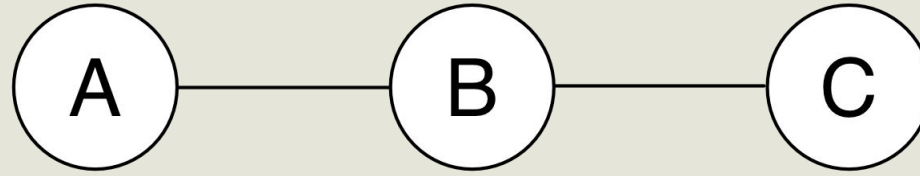
Keeping Track



Each snapshot, or “commit,” can also have a label that points to it.

One of these is HEAD, which always points to the place in the timeline that you are currently looking at. You can think of HEAD as being “You Are Here.”

A Summary of **Git**



A Git repository is a set of points in time, with history showing where you've been.

Each point has a name (here A, B, C) that uniquely identifies it, called a hash.

Each commit also has a user-generated message which describes its purpose.

The path from one point to the previous is represented by the difference between the two points.

Enter GitHub

Remotes serve as a way of sharing work with other developers.

GitHub has emerged as a premier location for such sharing.

It provides you with a common location that anyone can access.

In addition, it provides a number of useful tools for managing work that is being shared among a dispersed group of people.

Git is designed to be a 'distributed' system, where you can share code between any connected computers.

GitHub gives you a centralized 'canonical' repo that a team can access for the latest contributions from across the team.

SO WHAT?

HOMEWORK ASSIGNMENTS

GROUP PROJECTS



Wrapping Up

Git and Github together allow for:

- Back up
- Sharing
- Collaboration

ANY
QUESTIONS?

WORKING WITH REPOSITORIES



Working with Remotes

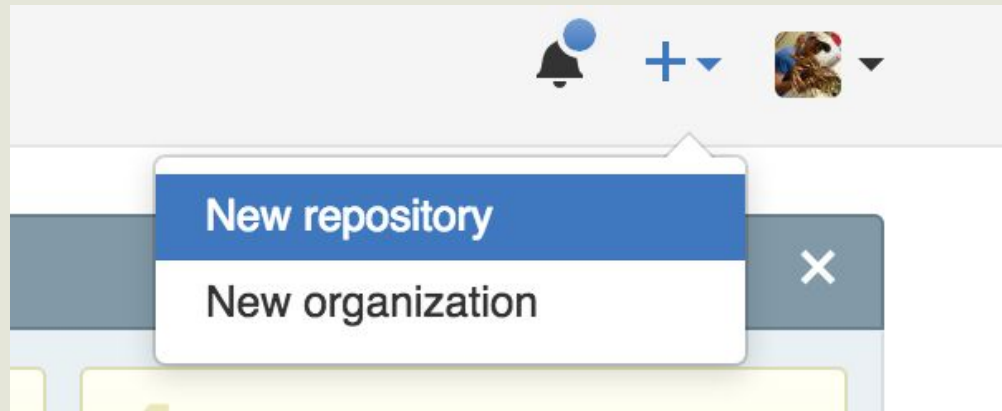
Since Git is a distributed versioning system, there is no central repository that serves as the one to rule them all.

Instead, you work with local repositories, and remotes that they are connected to.

A remote is a repository located on external servers in the cloud.

Remotes are important because they enable your work to be saved in the cloud so that if disaster strikes your local computer your project is not lost.

Creating a Repository in GitHub

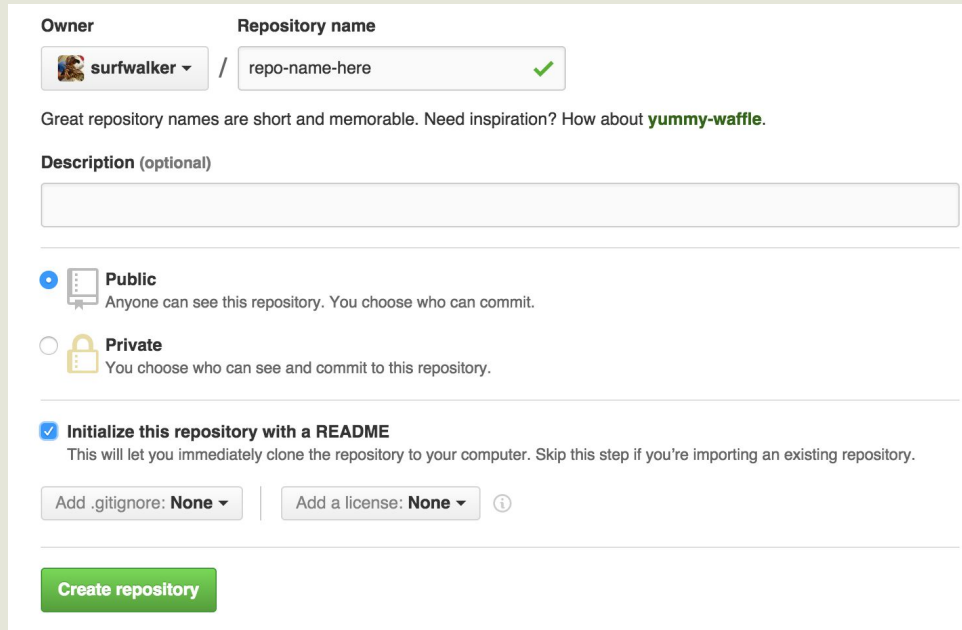


At the top right side of the window, look for your name and avatar.

Next to it you'll find a small + sign, click that.

From the menu that opens, select *New repository*.

Creating a Repository in GitHub



The screenshot shows the GitHub repository creation interface. At the top, there are two fields: 'Owner' with a dropdown menu showing 'surfwalker' and a 'Repository name' field containing 'repo-name-here' with a green checkmark. Below these fields is a text prompt: 'Great repository names are short and memorable. Need inspiration? How about **yummy-waffle**.' Underneath is a 'Description (optional)' text area. The next section has two radio button options: 'Public' (selected) with a subtext 'Anyone can see this repository. You choose who can commit.' and 'Private' with a subtext 'You choose who can see and commit to this repository.' Below the radio buttons is a checked checkbox for 'Initialize this repository with a README' and its subtext: 'This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.' At the bottom of this section are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None', followed by an information icon. A green 'Create repository' button is at the very bottom.

Owner: surfwalker / Repository name: repo-name-here ✓

Great repository names are short and memorable. Need inspiration? How about **yummy-waffle**.

Description (optional)

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

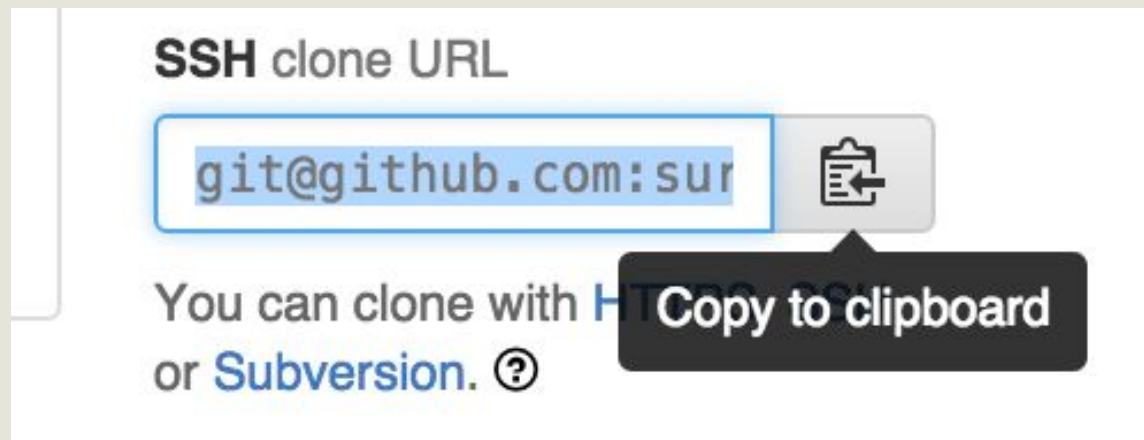
☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

Create repository

Set up your repository fill in *Repository name*, check the *Initialize this repository with a README* option and click *Create repository*.

Creating a Repository in GitHub



On the bottom right hand side of your screen make sure the option says *SSH clone URL* then copy the URL using either the *Copy to clipboard* button or your keyboard shortcut.

Navigate to your terminal.

GIT COMMAND: **clone**

The `clone` command creates a local copy of a remote repository (or repo).

It takes a URL as an argument. This URL is the pointer to the remote repo.

```
$ git clone git@github.com:surfwalker/repo-name-here.git
```

This will create a directory with the repo's name containing all of the repo's directories and files.

You can now copy or move any files you want to be a part of this project into the cloned directory just created.

Push To Your Remote

Now that you have files in your local repo you then need to commit (take a snapshot) those changes before “pushing” (uploading) them to your remote repo in the cloud (Github).

After navigating into your local repo directory, run the following commands in sequence:

```
$ git status
```

Git will show you what files on your computer have changed since you last made a commit.

```
$ git add <file_name_with_extension>
```

This tells Git that you’re going to want to take a snapshot of this file soon. It’s like telling your files so say, “CHEESE” for the upcoming photo.

```
$ git commit -m 'Your message goes here'
```

`Commit` tells Git to take the snapshot. The `-m` bit tells it that you want to save a message with that snapshot (think of it like a caption).

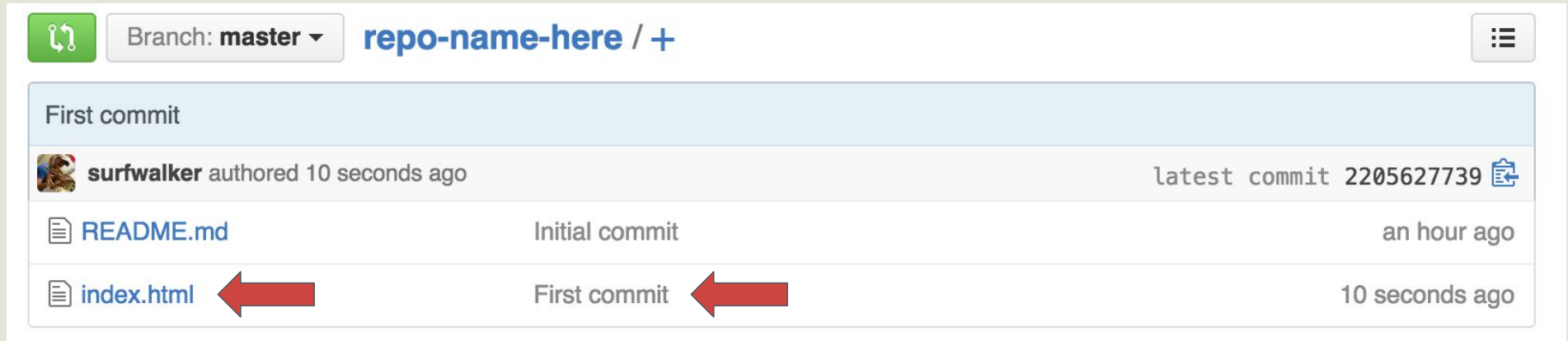
GIT COMMAND: **push**

You can now push the current version of all of your project's files to GitHub for safe keeping.

```
$ git push origin master
Counting objects: 6, done.
...
To git@github.com:surfwalker/repo-name-here.git
* [new branch]      master -> master
```

The `push` command sends your code to GitHub, and will make your GitHub repo have the same files, with the same changes, as the commit you just made.

Verify it on GitHub



The screenshot shows a GitHub repository interface. At the top, there's a green button with a fork icon, a dropdown menu showing 'Branch: master', and the repository name 'repo-name-here' followed by a plus sign. On the right, there's a hamburger menu icon. Below this, a light blue header bar says 'First commit'. The main content area shows a commit by 'surfwalker' (with a profile picture) 'authored 10 seconds ago'. To the right, it says 'latest commit 2205627739' with a clipboard icon. Below the commit info, there's a table of files. The first row shows 'README.md' as the 'Initial commit' made 'an hour ago'. The second row shows 'index.html' as the 'First commit' made '10 seconds ago'. Two large red arrows point to the 'index.html' file and its commit information.

File	Commit Message	Time
README.md	Initial commit	an hour ago
index.html	First commit	10 seconds ago

In your browser on GitHub you will see the file(s) that you pushed as well as the commit message.

Git Commands

New terminology learned in this lesson:

git clone

Copies a remote repository and all its files to your local machine in a new directory.

git push

Pushes all changes from your local repository to the named remote.

ANY
QUESTIONS?