Before we begin...

- Videos On!

Welcome

Agenda

- Terminal
- Git
- GitHub

Review

Terminal

What is the Terminal?

- A way to manipulate and interact with your computer
- It's entirely text-based
  - As opposed to the regular way we interact with our computers (which is called WIMP - Windows, Icons, Menus, Pointers)
- It's also known as the Command Line

Why do we use the Terminal?

- It's very fast
- It's automatable and flexible
- No interruptions
- Sometimes it is the only way:
  - Command Line Interaction (CLI) tools
  - Web Servers
- It gives us access to tools we wouldn't have access to otherwise

How do we use the Terminal?

- We will open up a Terminal application:
  - *Windows*: Git for Windows, Cygwin, WSL etc.
  - *Mac*: iTerm2, Terminal, Hyper etc.
- Once there, we interact with a shell (we will use the **Bash Shell**)
  - This, more or less, sends commands directly to the kernel (think of this as the hardware)

What is the Bash Shell?

- Bash is a regular program on your computer
- It was created to take commands from you
    - We talk to it using the Bash Shell Language

Are there other shells?

There are lots of them!

- Bash (Bourne Again Shell)
- C Shell
- Z Shell (zsh)
- Korn Shell
- Debian's Almquist Shell (dash)
- Plus many more

What can you do with it?

Anything!

- Perform day to day tasks (creating files, folders etc.)
- Run programs
- Edit and convert files
- Create backups
- Interact with databases
- Create servers
- Downloading, compiling and running programs
- Plus anything that you can think of!

How do we work with it?

Interactively

By opening up a REPL (like Git for Windows or iTerm2)

Non-Interactively

By running scripts

Let's see it!

Some Utilities

- Who am I?
- Where am I?
- Show me everything that is running
- Show me the manual
- Opening files

```
# Where am I?
pwd

# List all files in the current folder
ls

# Show all the running processes
top # CTRL + C to exit
```

Arguments

Just as we can in JavaScript, we can pass information to these commands

```
# Move into the desktop directory
cd Desktop
```

```
# Create a file
touch index.html

# Make a directory
mkdir new-folder

# Remove a file or a directory
rm file.txt

# Copy a file or a directory
cp file.text file-copy.txt
```

Flags

We can also give a command a bit more detail about how we would like it to run

```
rm -r FolderToBeDeleted

# `-r` stands for recursively delete
# all files and folders within FolderToBeDeleted
```

```bash
# Show everything within the current folder with details
ls -l

# Show everything including hidden files
ls -a

# Open `index.html` in Google Chrome
open index.html -a "Google Chrome"
```

```
# Clear your screen
clear

# Print the contents of a file
cat index.html

# Show the manual for a command
man ls
# Type `:q` to exit!
```

Resources

- Watch these Codecademy videos
- Track down the Terminal City Murderer
- Some other useful links:
  - 40 terminal tricks and tips
  - 25 useful find examples
  - Terminal cheat sheet

Git

History of Git

- Made in 2005 by Linux Torvalds
- Before that, he made the Linux Kernel
  - Here is a Ted talk
  - Here is his GitHub
  - Here is the source code for Git

Warning!

Git and GitHub can be tough to get used to and they take a long time to get comfortable with

Most explanations quickly get very technical, so focus on the concepts (as always)

What does Git do?

A Version Control System (VCS)

It takes snapshots of our projects and it gives us a project-wide undo button

A Collaboration Tool

It merges differences in our code for us

A Local Development Tool

It supports non-linear development as well

What does Git do?

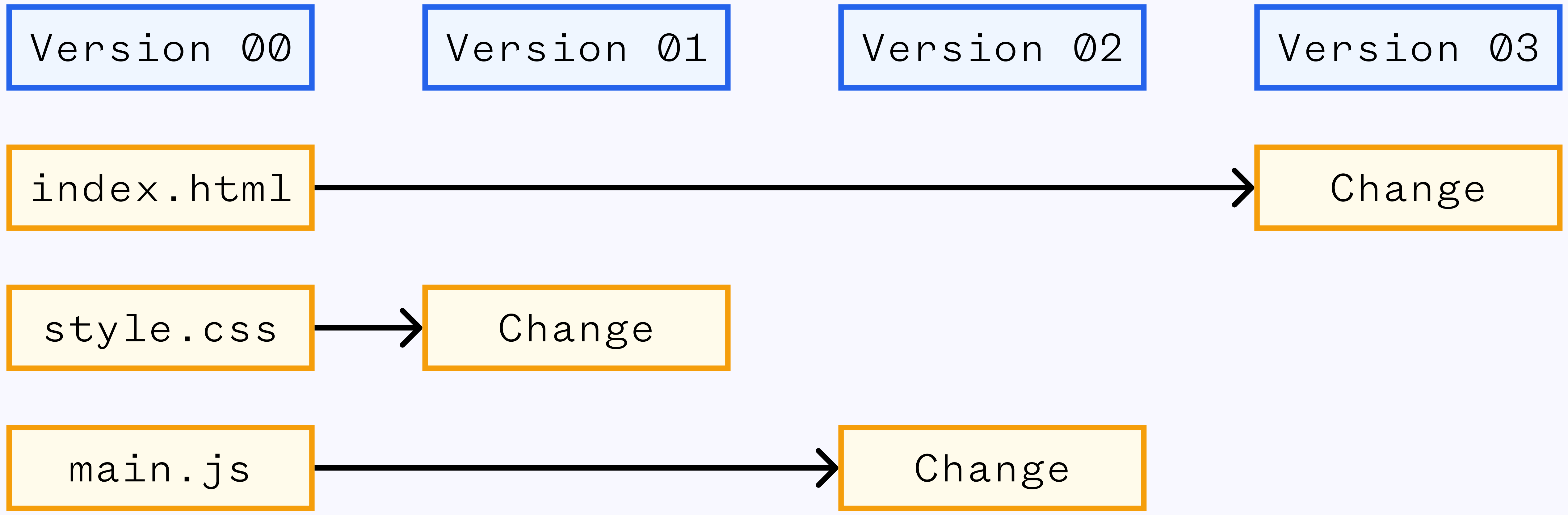
It's a tool for modern-day teamwork, for people who are working asynchronously on a shared body of work.

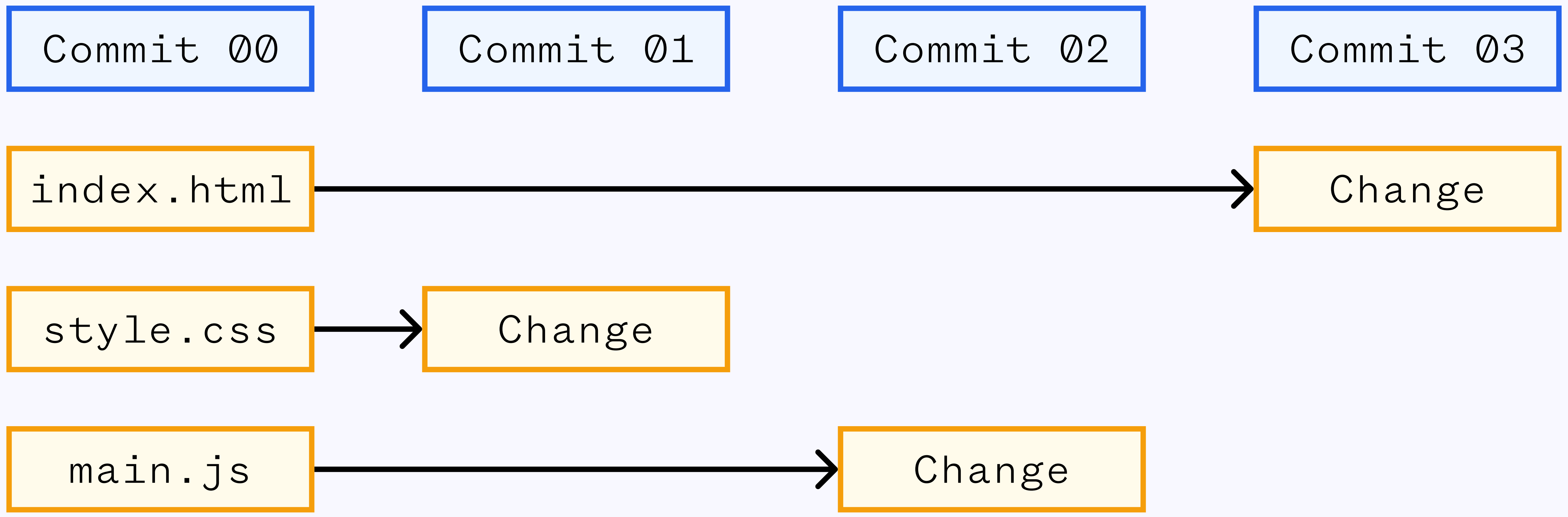
It saves us from moving floppy disks around, or saving lots of copies of the one file.

The more people there are on a project, the more likely you are to use it (though I use it all the time, for everything)

Why use Git?

- **You make a change and want to take it back?**
  - Git can undo it
- **You want to find where things went wrong?**
  - Git will show you
- **You want try adding a new, risky feature?**
  - Git can protect you
- **You want to work with a bunch of people?**
  - Git will make that easier


Version 00
Version 01
Version 02
Version 03
index.html
Change
style.css
Change
main.js
Change


Commit 00
Commit 01
Commit 02
Commit 03
index.html
Change
style.css
Change
main.js
Change

Terminology

Repository

A project

Add

Tell Git to pay attention to specific files and folders

Terminology

Commit

Tell Git to take a snapshot of the current *added* files

Origin

A place where your code is stored

Terminology

Push

Put all of the code online (normally using GitHub)

Branch

A version of your project

Terminology

Clone

When you copy a project (normally from GitHub) to your computer

Fork

Your copy of someone else's repository

Terminology

Merge Conflict

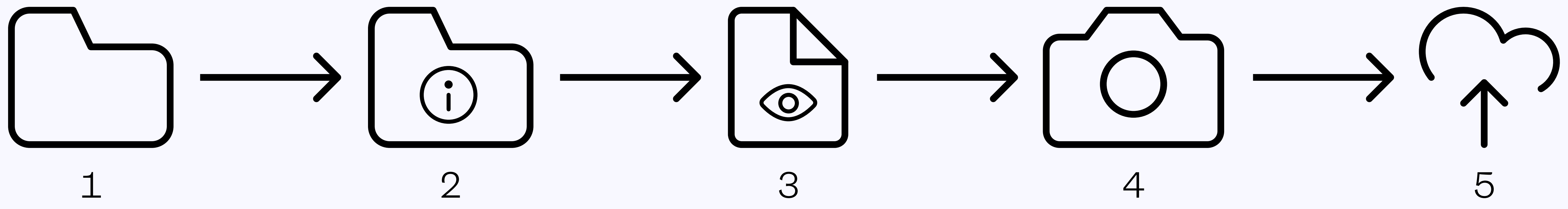
This is what happens when code can't be automatically merged. You have to decide which parts you want, and which parts you don't want

Pull Request

When you request to have a project include your code

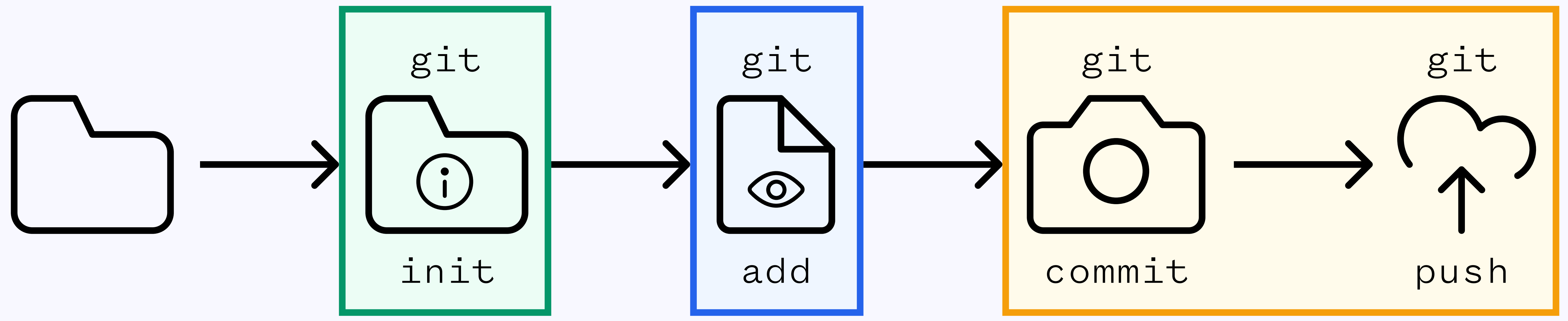
Git Process

1. Create your folder and your files within it
2. Turn that folder into something that Git can watch
3. Tell Git which files to pay attention to
4. Take a snapshot of all of the files at a certain time
5. Upload them (normally to GitHub)


1
2
3
4
5

Git Process

1. Create your folder and files
2. Turn that folder into a Git repository (using **`git init`**)
3. Tell Git which files to pay attention to (using **`git add`**)
4. Create a new version of your project by taking a snapshot of the current state (using **`git commit`**)
5. Upload them (using **`git push`**)


git
init
git
add
git
commit
git
push

When do I do all of this?

You use **`git init`** only once at the start of the project.

You use **`git add`** whenever there are new, important files to tell Git about.

You use **`git commit`** and **`git push`** whenever there have been significant changes.

How do we use Git?

- The Command Line
- Through Applications
  - GitHub for Desktop
  - SourceTree
  - Text Editor extensions
  - GitKraken
  - Plus many more...

Let's install the `code` tool

In VS Code:

- Open up the "Command Palette"
  - View -> Command Palette
- Search for "Install code"
  - Hit enter
- Restart your Terminal app
  - Test it! By running `code .`

Let's set up our Git

```
git config --global user.email "YOUR GITHUB EMAIL"

git config --global user.name "YOUR GITHUB NAME"

git config --global color.ui "auto"

git config --global core.editor "code --wait"
```

This is just a once off, but we need to run these commands one at a time in the terminal. Make sure you add your GitHub name and email in the quotes!

Some useful Git commands

Create a Git repository

```
git init
```

Show the status

```
git status
```

Make Git watch files

```
git add README.md

# Add everything in the current directory
git add .
```

Take a snapshot of all added files

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

Show the previous commits

```
git log # Press :q to exit
```

Resources

- Atlassian: Learn Git
- Official GitHub Git Tutorial
- Codecademy
- Git & GitHub for Poets
- CodeSchool
- Git For Humans

GitHub

What is GitHub?

- It is a website that hosts Git repositories
- Helps with collaboration
- It is a Graphical User Interface (GUI)
- Helps us perform common tasks
- The Dropbox or Google Drive for code

Why do we use it?

- To share our code with other computers
- For collaboration (Pull Requests, Forks etc.)
- It acts as a portfolio
- To visualise what is going on
- As a project management tool (Projects)
- An error reporting system (Issues)
- Documentation (Wiki)
- Free Hosting (GitHub Pages)
- It is the Industry Standard

How do we use it?

Once you have a local Git repository...

- Create a repository on GitHub
- We need to tell Git where the code should be stored
  - **`git remote add origin URL`**
- We need to push (or upload) all of the code
  - **`git push origin main`**

How do we use it?

Once you have a local Git repository...

- We need to pull (or download) all of the code
  - **`git pull origin master`**
- We can also clone a repository
  - **`git clone URL`**

Personal Access Tokens

- Generate a Personal Access Token by following these steps
  - Make sure you copy it! You'll need it in the next step
- When you use **`git push`**, use this Personal Access Token as the password!

A Typical Upload Workflow

```
git add .

git commit -m "Made changes"

git push origin main
```

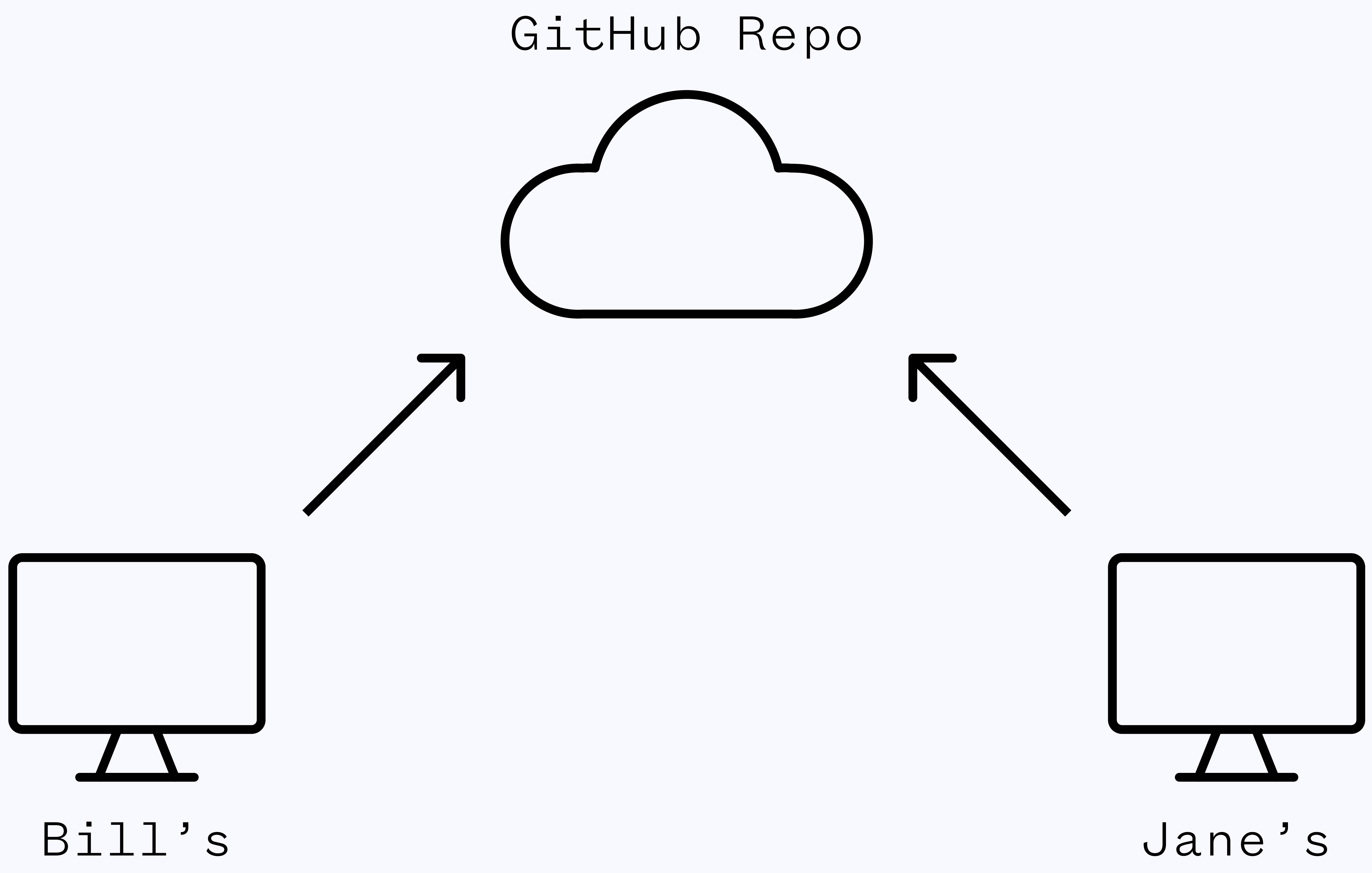
Collaboration Approaches

A Shared GitHub Repository

There is one shared GitHub repository that everyone contributes to

Someone makes the GitHub repository, then everyone else clones it to their computer

Someone makes their changes, then they **push** them up to GitHub. After that, everyone else **pull**s down the changes

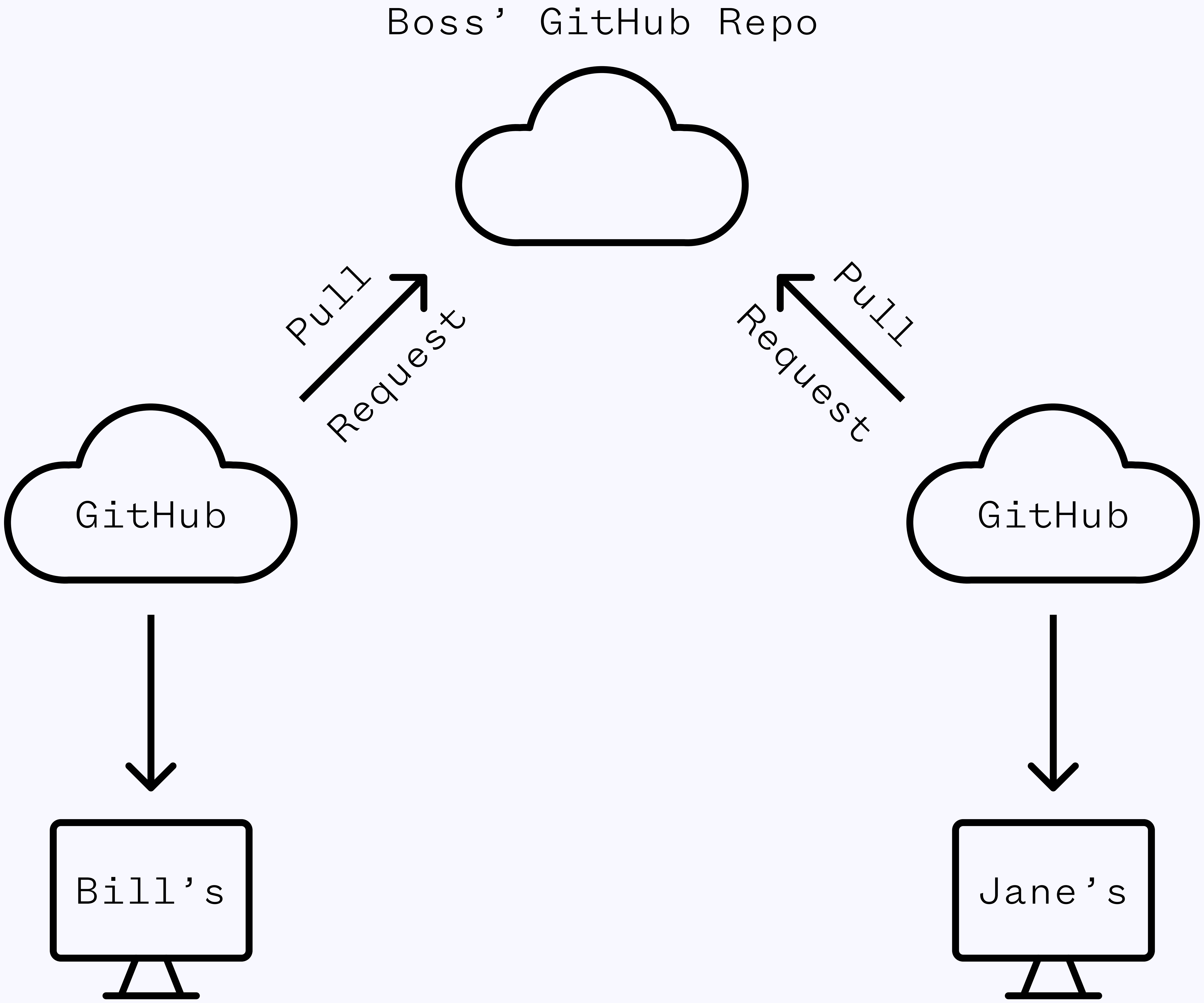

GitHub Repo
Bill's
Jane's

Pull Requests

Everyone has their own GitHub repository that represents their version of the project, but the boss also has a GitHub repository

When someone makes changes that they want to make available, they make a pull request to the boss's GitHub repository

The boss acts as a gatekeeper and can then merge that pull request, or reject it


Boss' GitHub Repo
Pull Request
Pull Request
GitHub
GitHub
Bill's
Jane's

That's all for tonight!

Homework

- Find the Terminal City Murderer
- Add your homework to Git and GitHub!

Homework: Extra

- Read about Parcel
  - Make sure you look at the Web App Guide (not the library one)

What's Next?

- Node
- NPM
- Creating Web Servers using HTTP Server
- AJAX
- APIs

Thank You!