

# Git and GitHub

---

INFX: Intro to Programming

# Today's Objectives

- Feel comfortable with the **command-line** and **markdown**.
- Use **redirects** and write **shell scripts**
- Understand the purpose of **version control systems**
- Manage code using **git**
- Save code to the cloud using **GitHub**

**RECALL**

# Command-Line

Lets you type **commands** to control your machine.

```
# list all files in the current folder
```

```
ls -a ← Note the -a option!
```

```
# change to a folder
```

```
cd ~/Desktop ← ~ is shortcut for "home" directory
```

```
# make a directory
```

```
mkdir my-folder
```

```
# go back to the *parent* directory
```

```
cd .. ← .. means "parent folder"
```

```
# open a file (in current folder) for viewing
```

```
less README.md
```

← **Use up/down arrows to scroll.  
Type q to quit.**

# Command Reference

Action	Syntax
Copy a file	<code>cp OLD_FILE NEW_FILE</code>
Move a file	<code>mv OLD_FILE NEW_FILE</code>
Delete a file (careful!)	<code>rm FILE</code>
Open a file	<code>open FILE</code> [Mac]; <code>start FILE</code> [Windows]
View file contents on command-line	<code>less FILE</code>
Output file contents	<code>cat FILE</code>
See previous commands executed	<code>history</code>

Action	Syntax
Make your computer speak [Mac]	<code>say "Text to say"</code>
Do the same thing again	<code>!!</code>
Watch Star Wars	<code>telnet towel.blinkenlights.nl</code> (use <code>ctrl-]</code> then <code>quit</code> to exit)

# Visual Studio Code



**(command + space, search for "Visual Studio")**

# Redirects

>

Put output in file instead of display

```
echo "Hello World" > hello.txt
```

>>

Append to end of file

```
echo "Goodbye :)" >> hello.txt
```

<

Take input from file (less common)

```
cat < hello.txt
```

|

Take output and "pipe" (send) to next command

```
cat hello.txt | wc
```

# Shell Script

A list of commands (in order) that we want to run as a bunch; the "script" that our program should follow.

# Make a Shell Script



# Markdown

Markdown is a simple **syntax** for specifying how plain text should be formatted.

```
This is a paragraph in which we'll add
**bold text**, _italicized text_, and `code`
into the middle of a sentence
```

```
# Top Level header
```

```
## Second Level Header
```

```
Here is a normal paragraph
```

- List item 1
- List item 2
- List item 3

```
```
```

```
block of code
across multiple lines
```
```

```
> Here is a block quote
```

This is a paragraph in which we'll add **bold text**, *italicized text*, and `code` into the middle of a sentence

## Top Level header

---

### Second Level Header

---

Here is a normal paragraph

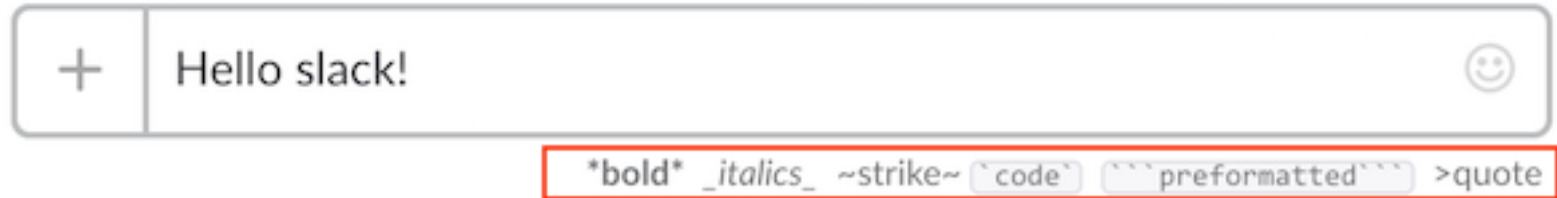
- List item 1
- List item 2
- List item 3

```
block of code
across multiple lines
```

Here is a block quote

## Module 3 exercise-1

# Use Markdown in Slack!



We expect questions to be clearly formatted with

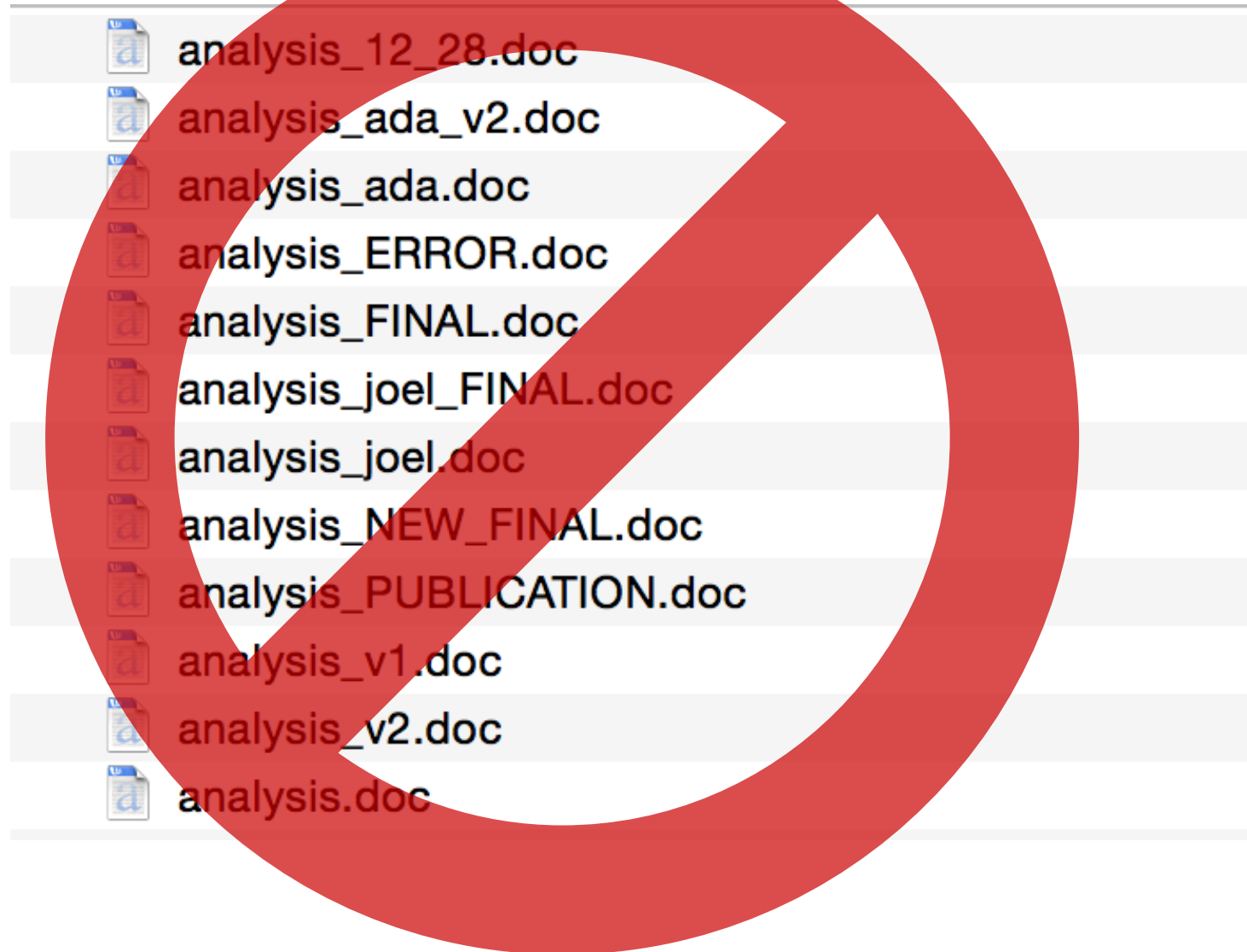
```
```
```

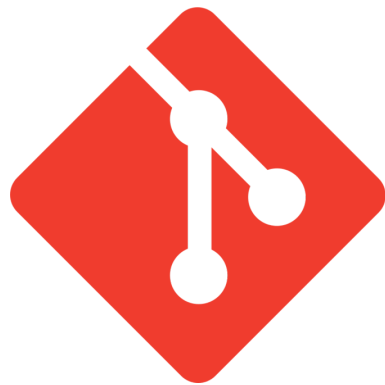
code blocks

```
```
```

or even ``inline code`` when appropriate!

# Version Control





**git**

# Version Control



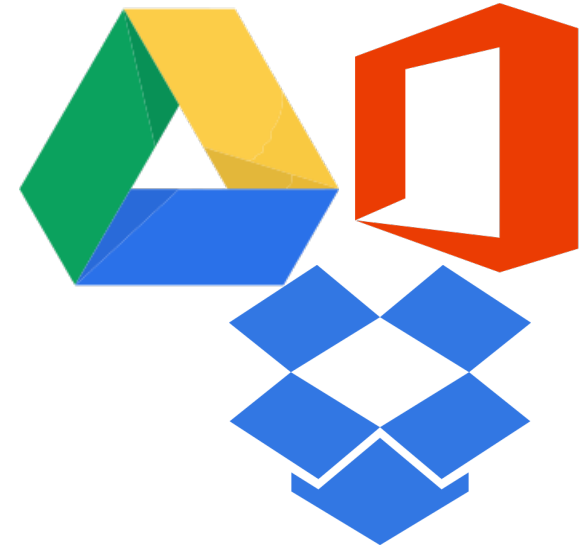
*“A version control system (VCS) is a tool for managing a collection of program code that provides you with three important capabilities: **reversibility**, **concurrency**, and **annotation**.*

*— Eric Raymond*



# git

vs.



- Line-by-line change tracking
- Simultaneous, off-line editing
- Detailed history and "undo" capabilities

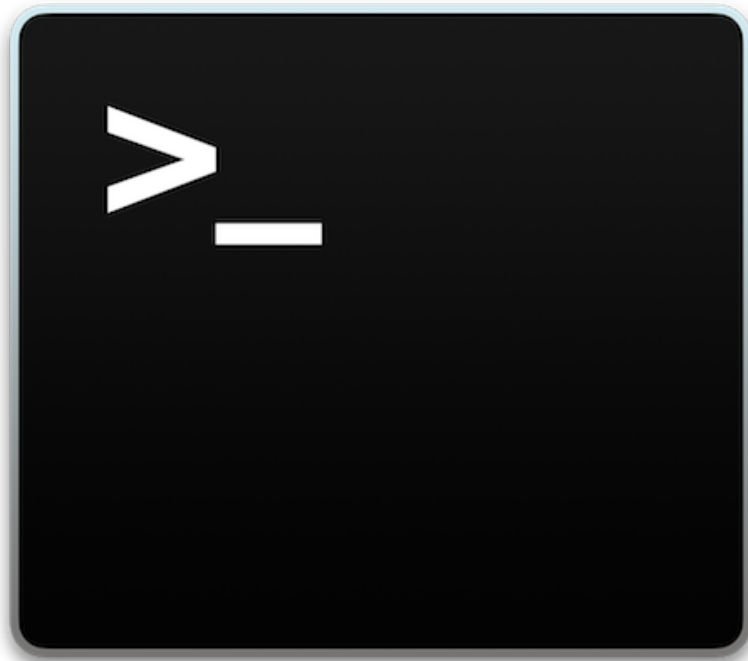


THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.





# Install git

<http://git-scm.com/downloads>

(see [module 1](#))

# Configure git

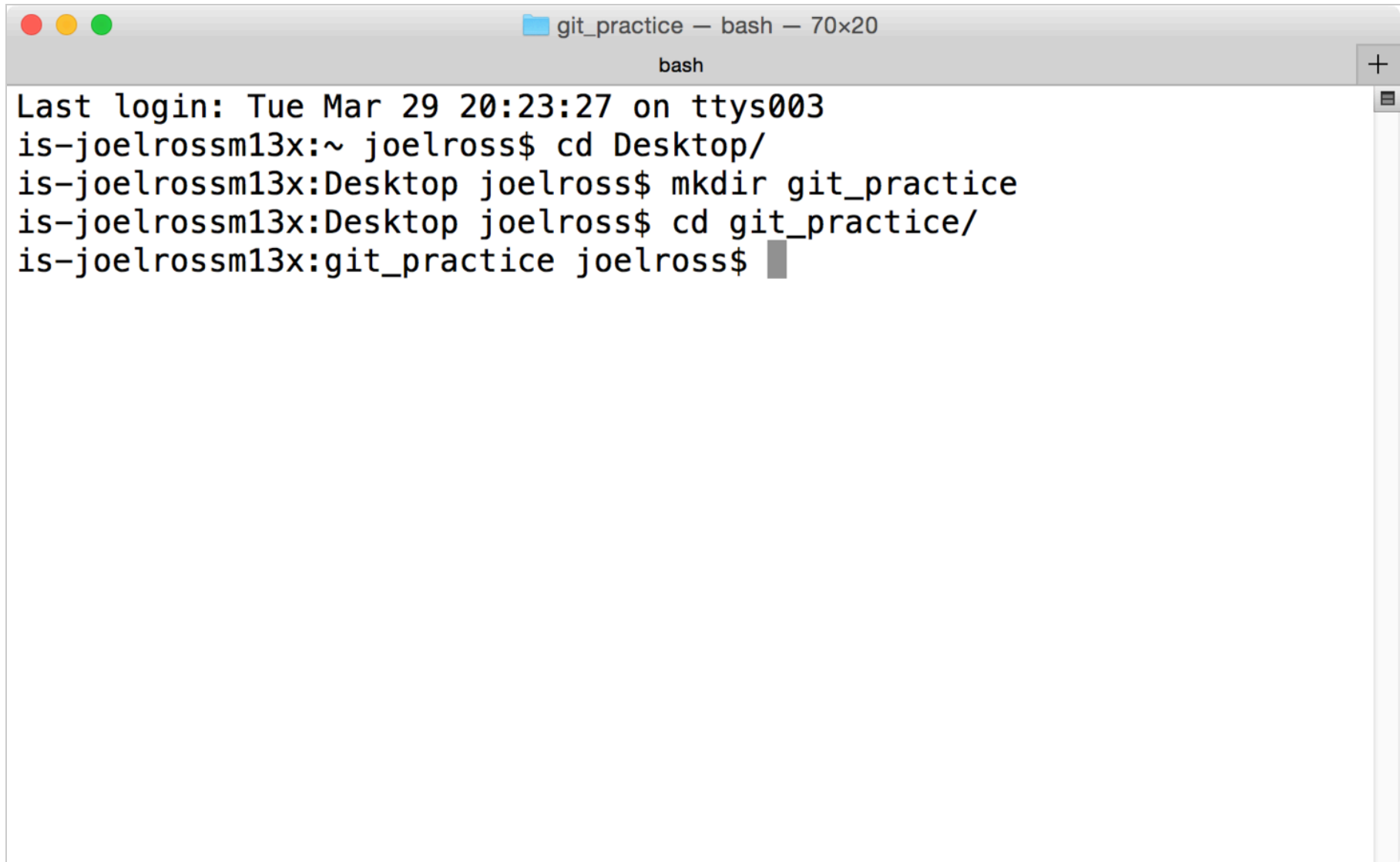
```
#do this just once per machine!
```

email you signed up for  
GitHub with



```
git config --global user.email "your-email-address"
```

```
git config --global user.name "your-full-name"
```



```
git_practice — bash — 70x20
bash
Last login: Tue Mar 29 20:23:27 on ttys003
is-joelrossm13x:~ joelross$ cd Desktop/
is-joelrossm13x:Desktop joelross$ mkdir git_practice
is-joelrossm13x:Desktop joelross$ cd git_practice/
is-joelrossm13x:git_practice joelross$
```

# Repository

(or "repo")

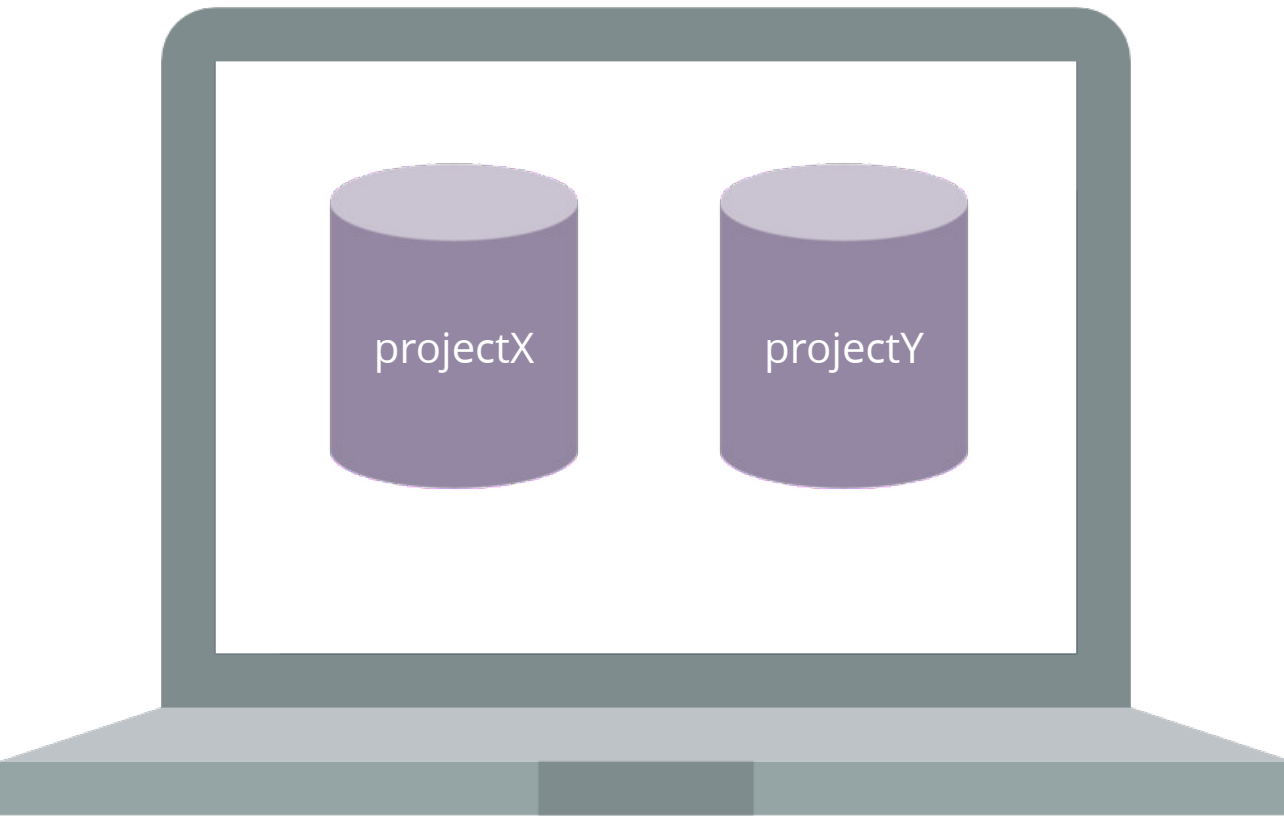


repo/.git

```
# run IN directory of project  
git init
```

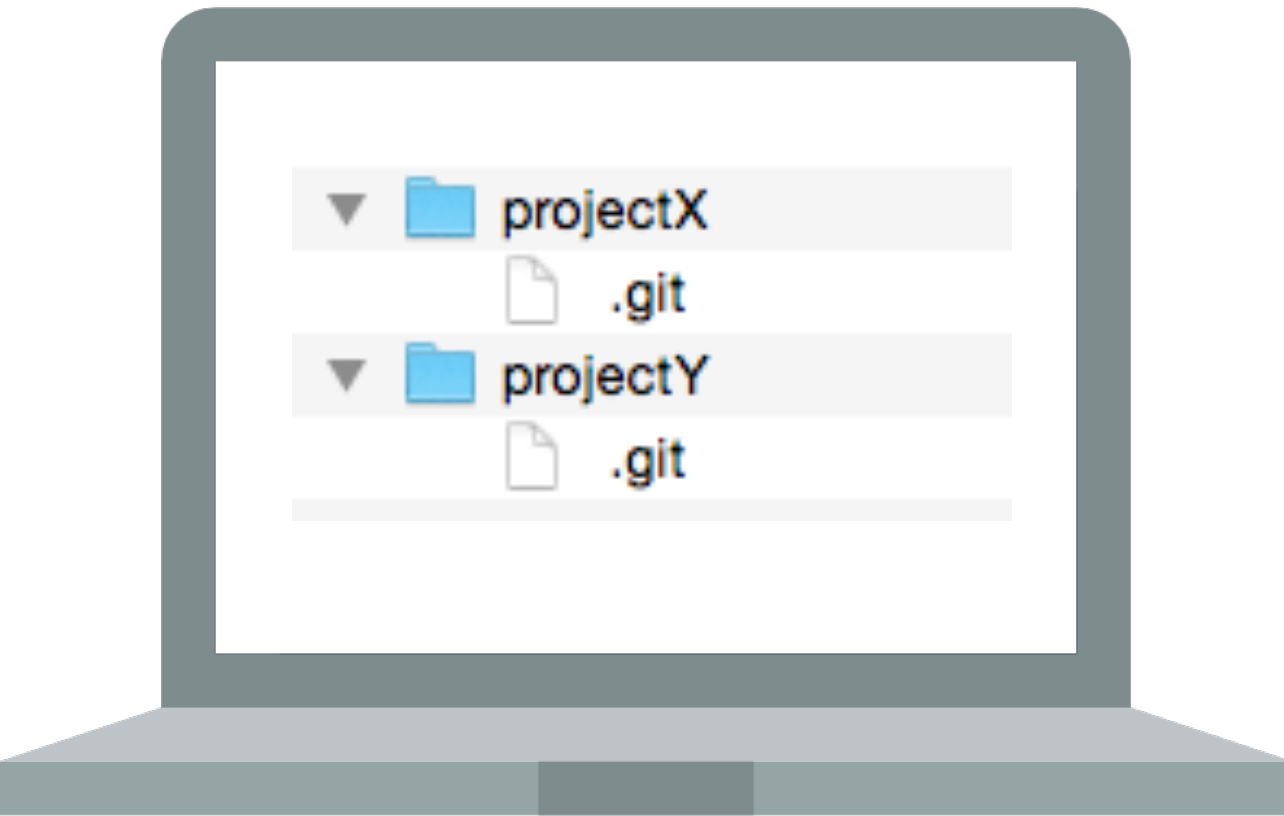
# Repository

(or "repo")



# Repository

(or "repo")





**DO NOT PUT ONE  
REPO INSIDE OF  
ANOTHER!!**



# Git Commands

```
git init
```

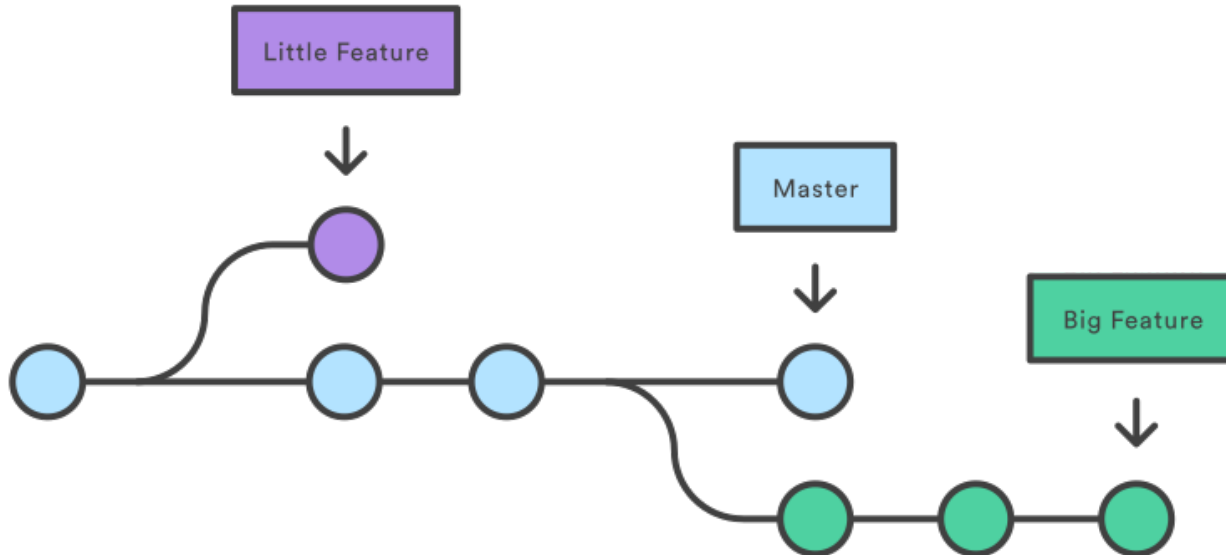
Create new repo in folder

```
git status
```

Check status of repo

# Branches

Branches allow for **non-linear development** and for naming different versions of code in the same repo.



*more on this later in the quarter!*

# Make Some Changes!

Create a new file `places.md` inside the repo directory.

This document should include a Markdown-formatted **list** of **3** places you would like to visit.

**How do we "save" our changes?**

# The Staging Area

Put changes in temporary storage before committing.

```
git add file
```

Add file to staging area

```
git add .
```

Add everything in directory



# Git Commands

```
git init
```

Create new repo in folder

```
git status
```

Check status of repo

```
git add file
```

Add file to staging area

# Committing Changes

Store current snapshot of files in repository!

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

Commit changes

If you forget the **-m** option,  
use **:q** (colon then q) to quit *vi*

# Commit Message

## Etiquette

- Detail what change the commit is making
  - Use **imperative** mood ("Add feature", not "added feature")
    - *"If applied, this commit will {your subject line}"*
- 50 character limit for first line
  - Can add more details after a blank line



# Be Informative!



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

[xkcd.com/1296/](http://xkcd.com/1296/)

# Git Commands

```
git init
```

Create new repo in folder

```
git status
```

Check status of repo

```
git add file
```

Add file to staging area

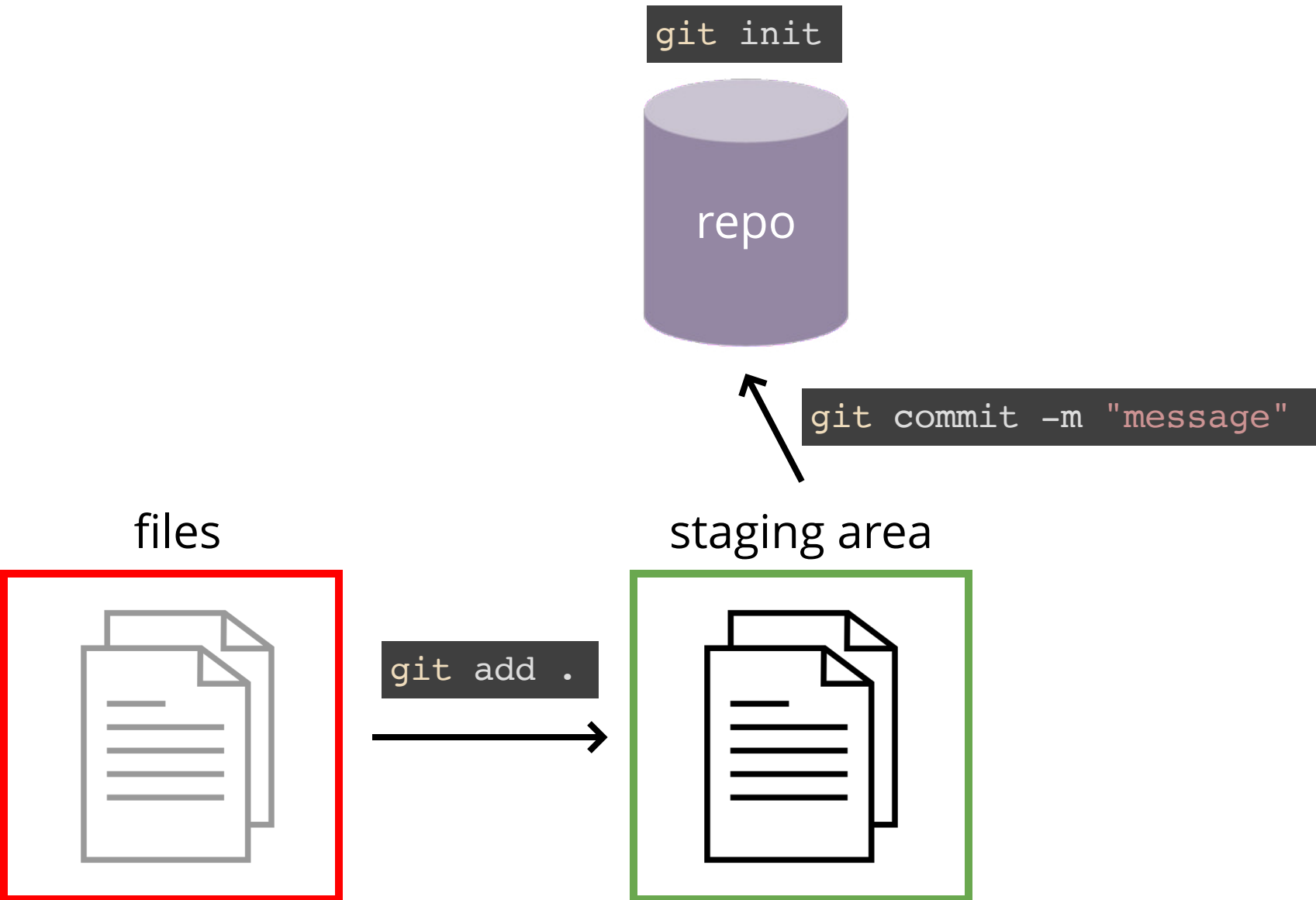
```
git commit -m "message"
```

Commit changes

```
git log [--online]
```

View commit history

# Local Process



# Practice!

1. **Edit** your document to include a *second* list with 2 places you've already visited.
2. **Add** the changes to the staging area.
3. **Commit** the changes to the repository.





**git**

*is to*



**github**  
SOCIAL CODING

*as*



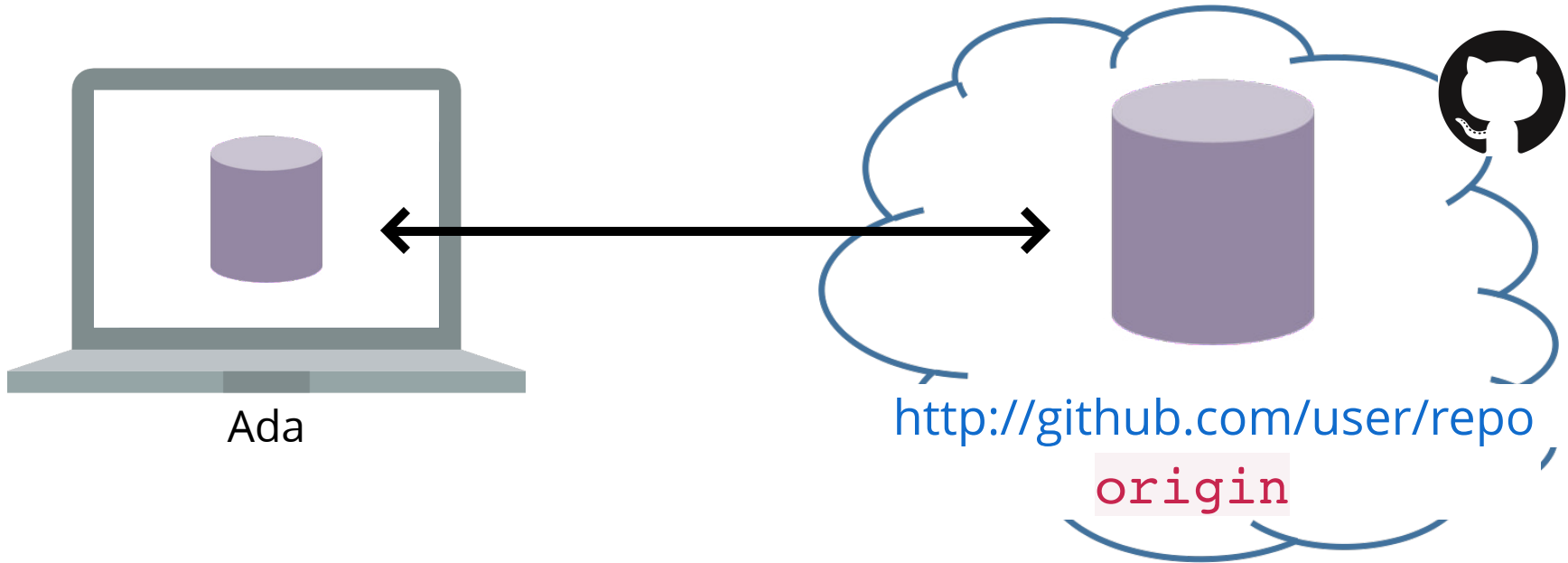
*is to*

**imgur**

# Remotes

A **remote** is a repository *on another machine* that a repository can upload and download code from.

Git gives each remote a name (an "alias") to easily refer to it. By convention, the "primary" remote (where you cloned from) is named `origin`



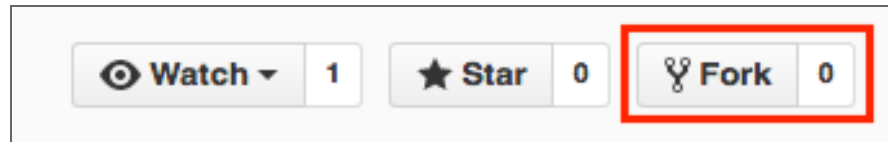
# Forking

Forking creates a **copy** of a repo *on GitHub's computers*.



# Forking

[https://github.com/joelwross/  
github\\_practice](https://github.com/joelwross/github_practice)



# Git Commands II

```
git clone url
```

Copy repo to local machine

*Only need to do this once  
per machine!*

# Ch-ch-changes

1. **Edit** the README to include your name
2. **Add** the changes to the staging area.
3. **Commit** the changes to the repository.

# Push to GitHub

Upload commits to the GitHub cloud repo.

```
git push origin master
```

Upload to the **origin** remote, **master** branch

# Git Commands II

```
git clone url
```

Copy repo to local machine

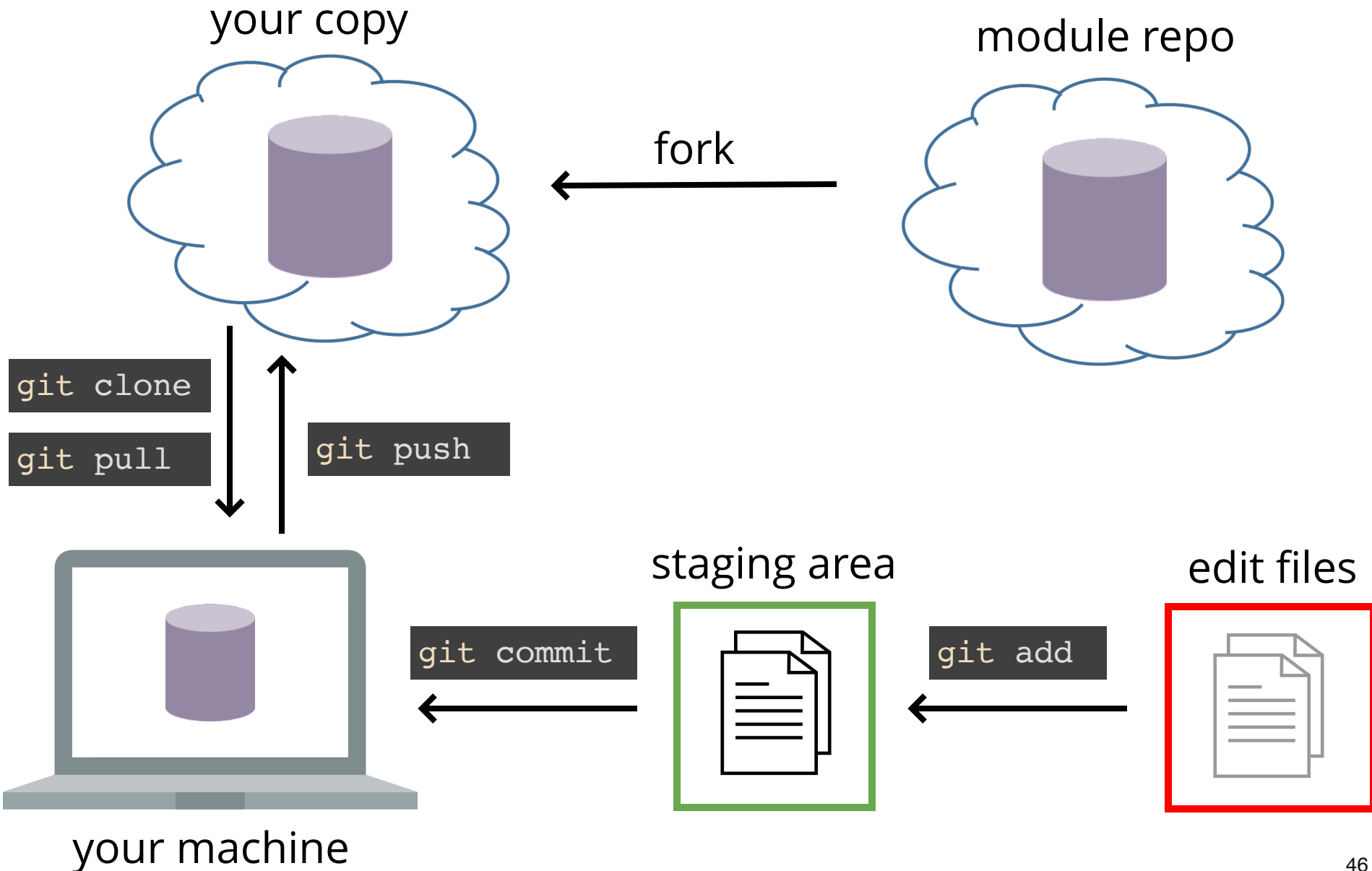
```
git push [origin master]
```

Upload commits

```
git pull
```

Downloads and *merges* commits

# Using GitHub



# GitHub Classroom

Assignments use a tool called GitHub Classroom to automatically create *private* repos for your homework.



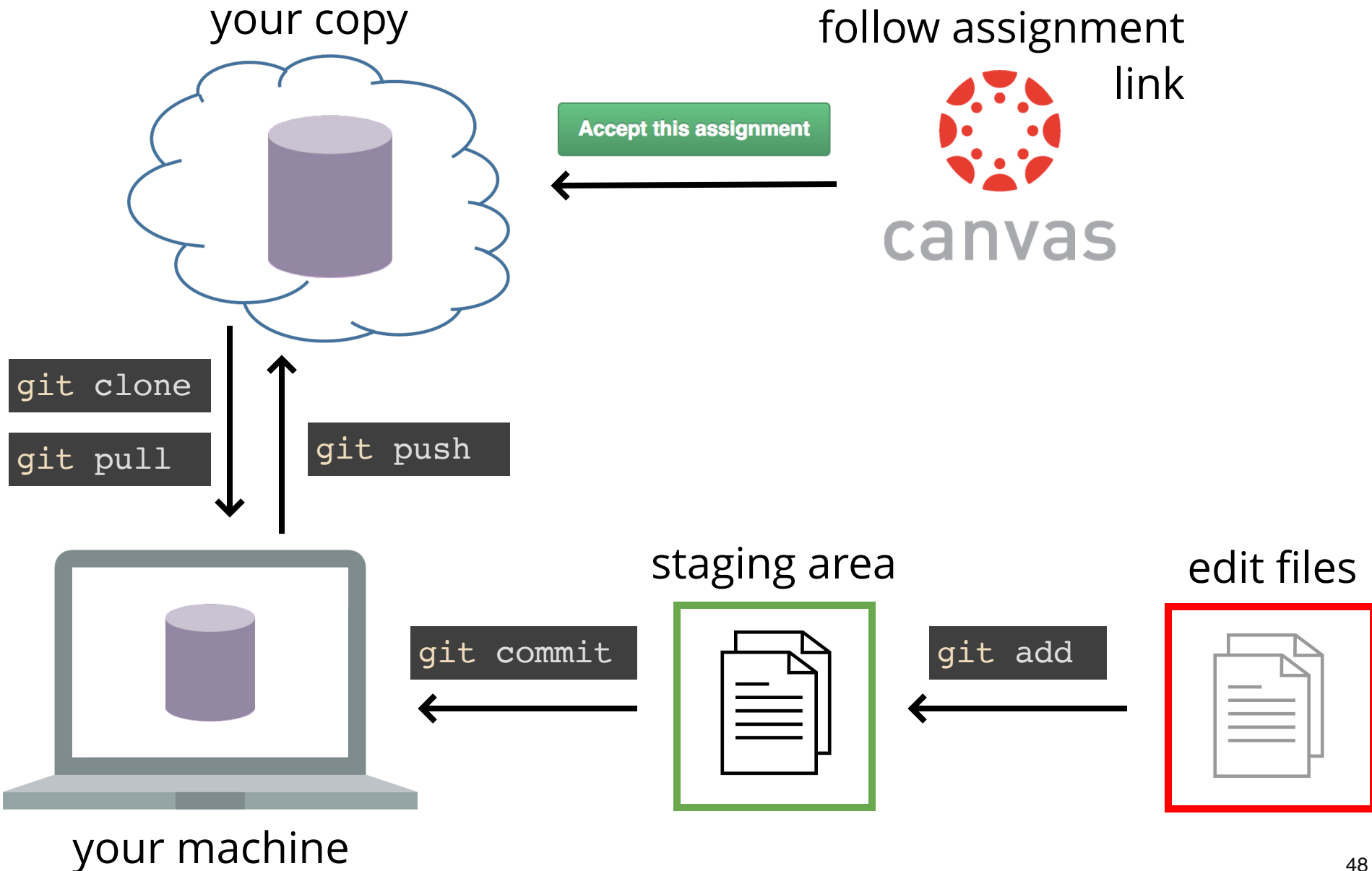
Accept the **a1 - Start with Git** assignment

Accepting this assignment will give you access to the **a1-start-with-git-studenttest-jr** repository in the [@info201-w17](#) organization on GitHub.

Accept this assignment

DO NOT FORK  
ASSIGNMENT REPOS

# Using GitHub (Assignments)





# Questions on Git?

# Version Control



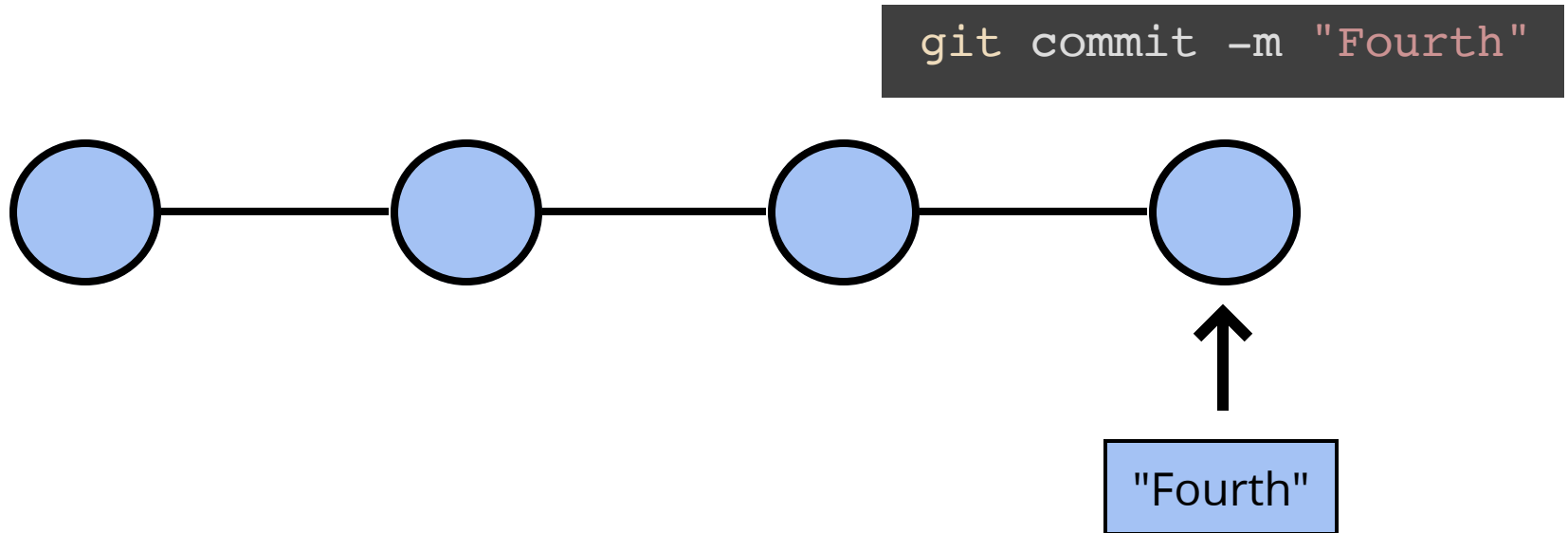
*“A version control system (VCS) is a tool for managing a collection of program code that provides you with three important capabilities: **reversibility**, **concurrency**, and **annotation**.*

??

— Eric Raymond

# Commit History

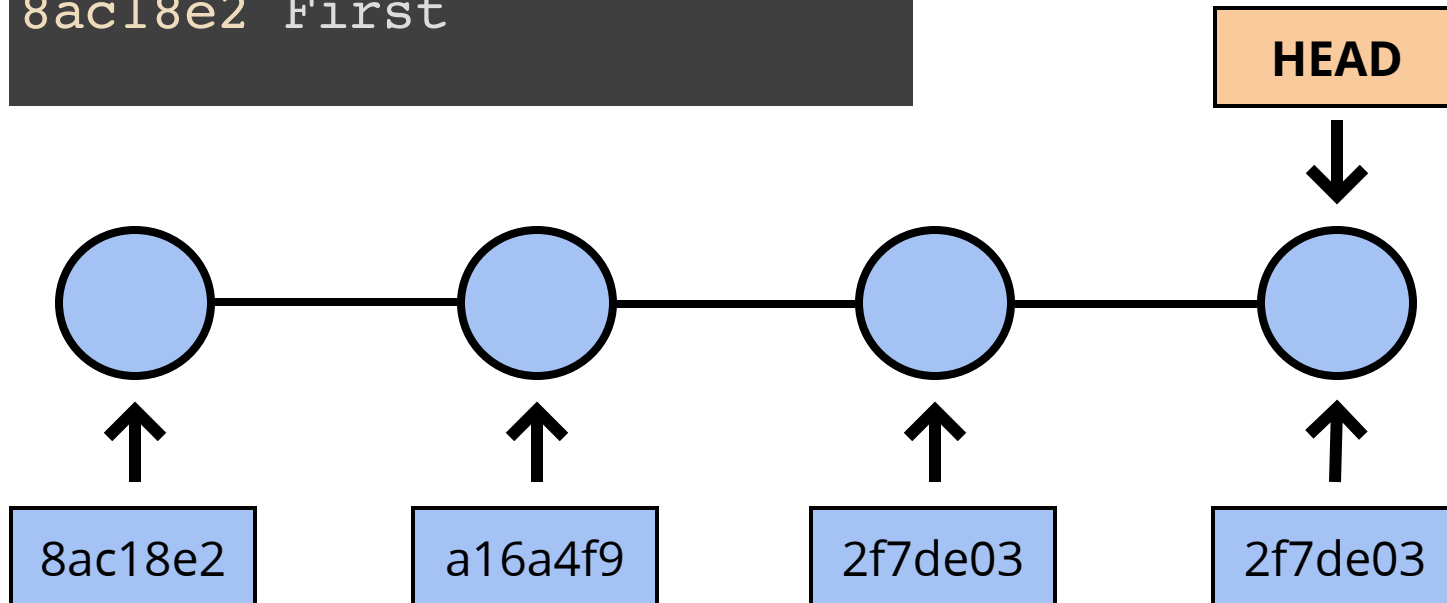
Git stores the **sequence** of commits that have been made.



# Commit IDs

Each commit has a unique **commit id** that refers to it

```
$ git log --oneline
2f7de03 Fourth
8417290 Third
a16a4f9 Second
8ac18e2 First
```



# Undoing Things

```
git checkout [commit] [file]
```

Replace file with version from previous commit

```
git revert [commit]
```

Change files to undo commit and remove the changes it made (adding a new commit, preserving history)

# What we did...

- Practiced with the command-line and markdown
- Saved file versions with git
- Push data to GitHub

# Action Items!

- Be comfortable with **module 4**
- Assignment 1 due Tues night - start it now!

Tuesday: starting with Python!  
(pre-read: **module 5 & 6**)