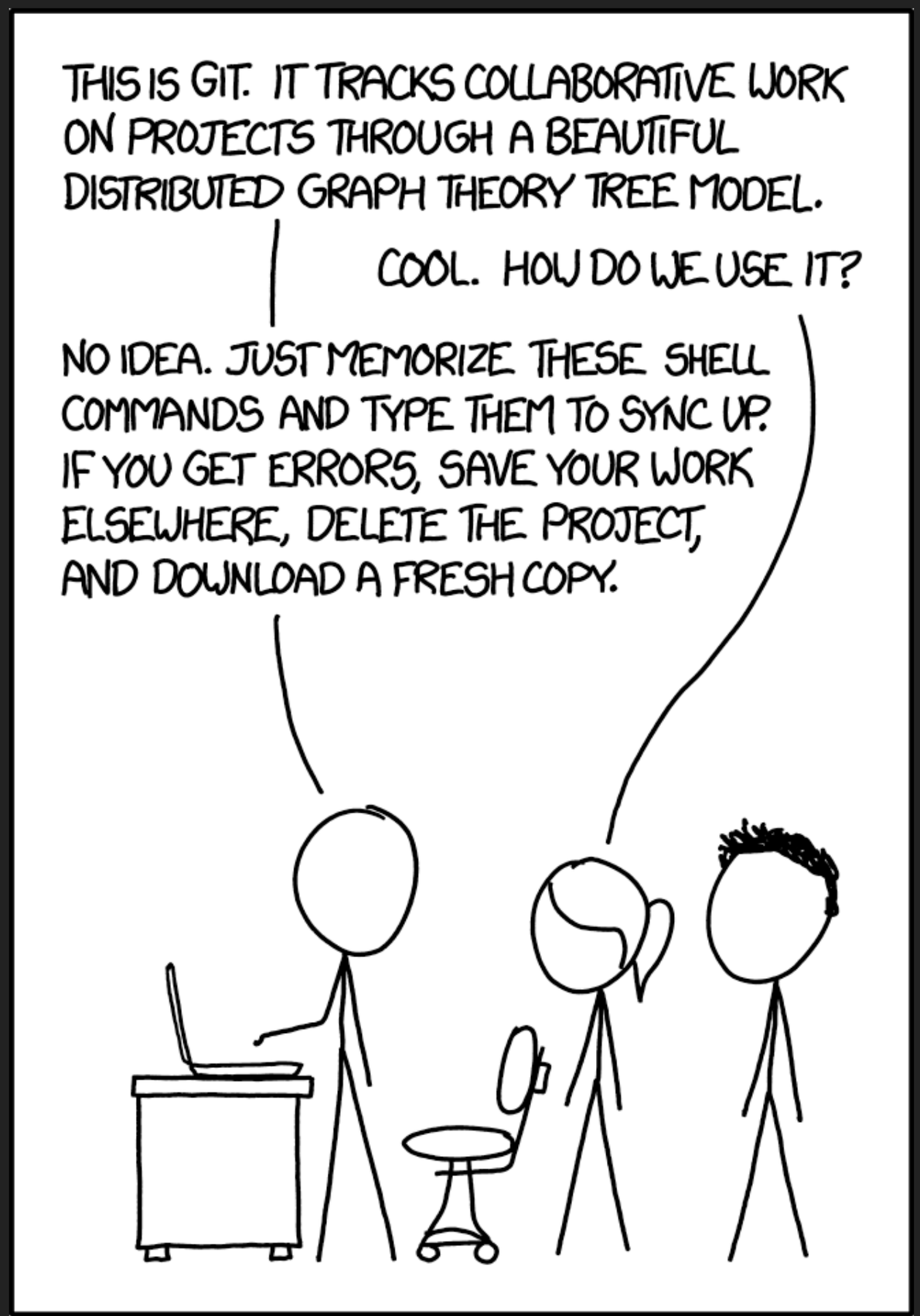
# GIT GUD: AN INTRODUCTION TO GIT

By: Jordan Hewko && Cooper Wallace

# A FOREWORD:

- This presentation is not all encompassing, this is the basics
- We will be using git on the command line, ignoring GUI. The command line isn't scary!
- Strongly recommend that you create SSH Keys for using Git. Creation of them is outside of the scope of this presentation.



# WHAT IS GIT?

- “Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people” - Wikipedia
- Written by Linus Torvalds in 2005.
- To test the stability of Git, he committed 1 million lines of the linux kernel. Which it could handle.

# WHY SOURCE CONTROL?

Bonus Points:

- Source control (Git) and familiarity with unix command line
- Experienc

Some Of Our Tools

Some experience using Python and/or Django, Javascript and jQuery would definitely be a plus. In our experience, Git is usually the largest stumbling block, so students comfortable with Git (or able to quickly get comfortable with Git) will likely have an easier time developing.

ractices.

-Strong u

-Strong k

-Strong u

-Experien

Qualifications

:

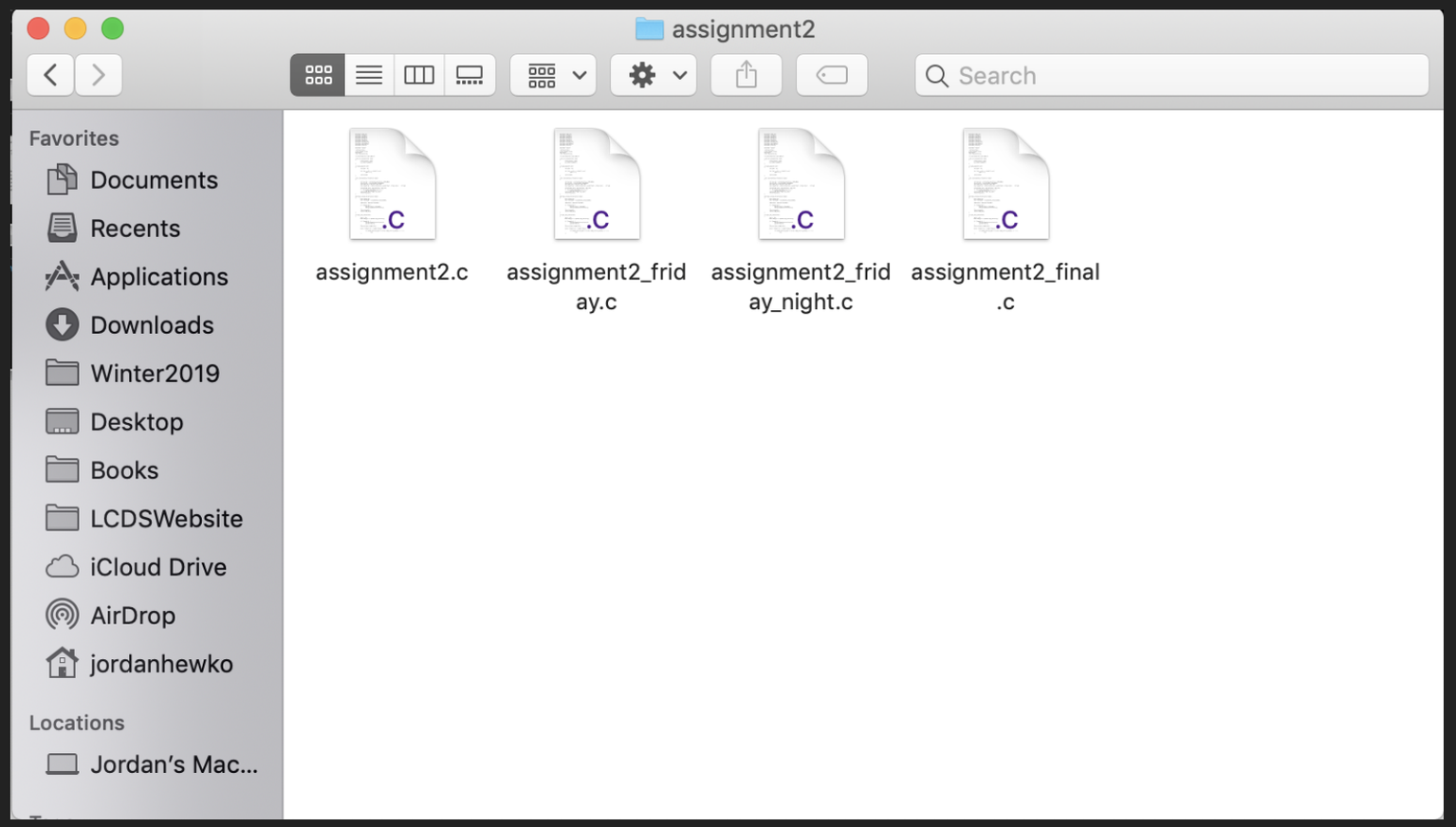
- Strong interpersonal skills and positive attitude
- Commercial experience in web application development, software development and app Development
- Object oriented software analysis and design
- Strong written and verbal communication skills
- Working knowledge of code repositories (Git, Mercurial)
- Self-motivated and ability to work with little oversight
- Experience in unit testing (any

N

-Unit framework)

# NO REALLY, IT WILL MAKE YOUR LIFE EASIER

- Track your progress
- Fix errors
- Makes group projects easier



# SET USER INFORMATION

- The first time you use git, You'll need to setup your user

```
git config --global user.name "Joe"
git config --global user.email "Joe@email.ca"
```

- You can change your options per repo by dropping --global

# CREATING A NEW REPOSITORY

- Create a new local repository with `git init`
- Connect it to a remote repository (ie on Github) with the command `git remote add <repo name> <url>`
- Note: This repository is only for this folder!

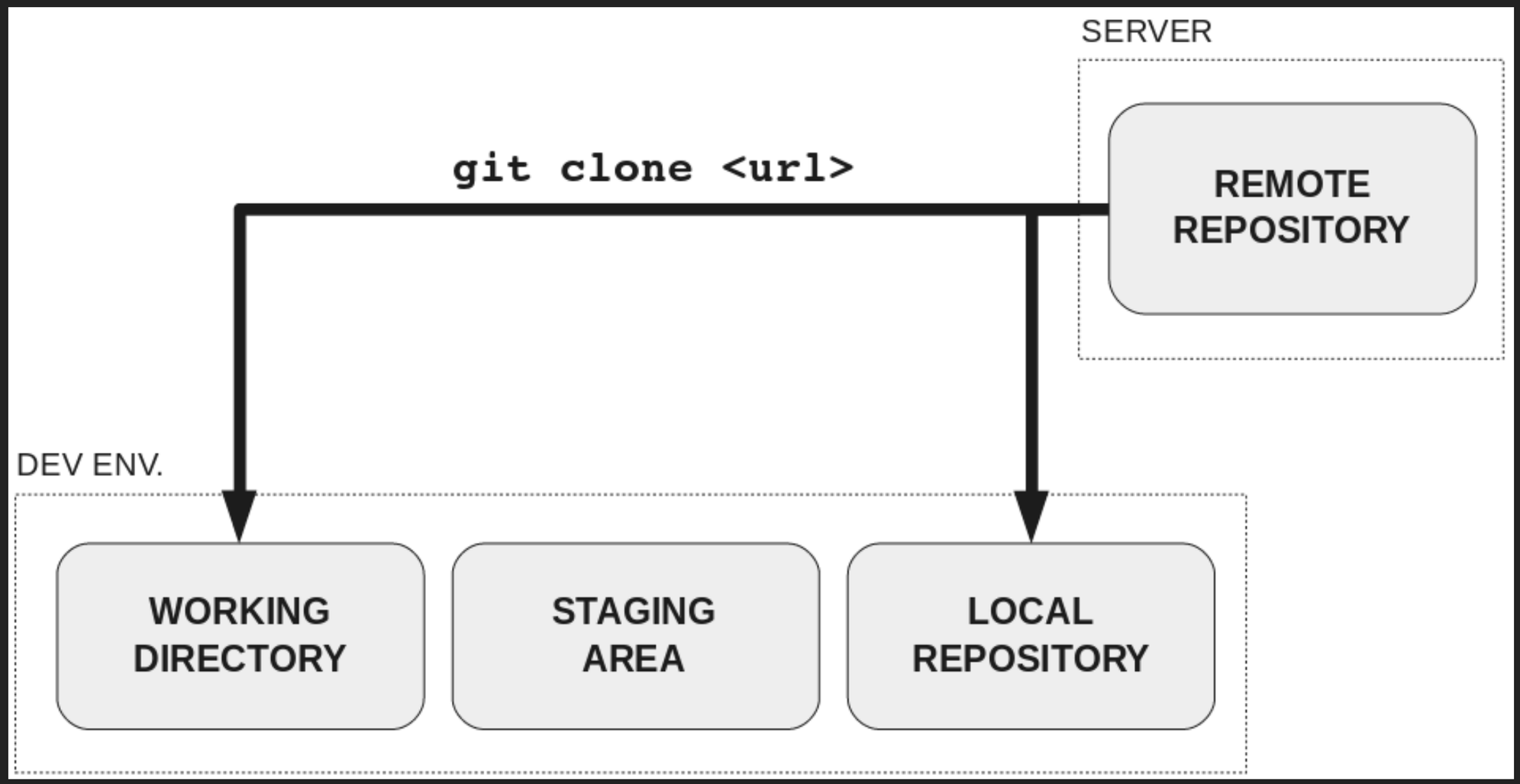
# CLONING

- To receive the in your working directory, you want to clone them onto your machine
- The command `git clone <url>` will do this

# FORKING A PROJECT

- To receive “a copy” of a public repo, you fork it
  - This will create a copy of the repository under your name
- After forking the project, clone the project to edit the files

# PROJECT STRUCTURE



# COMMITTING A FILE

- When you add a file, it is staged
- Staged files need to be committed
- Check your committed files using the command `git status`
- Use `git diff` to see what changes you made

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   myfind.c

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .DS_Store
        makefile
        tree.c
        tree.h
```

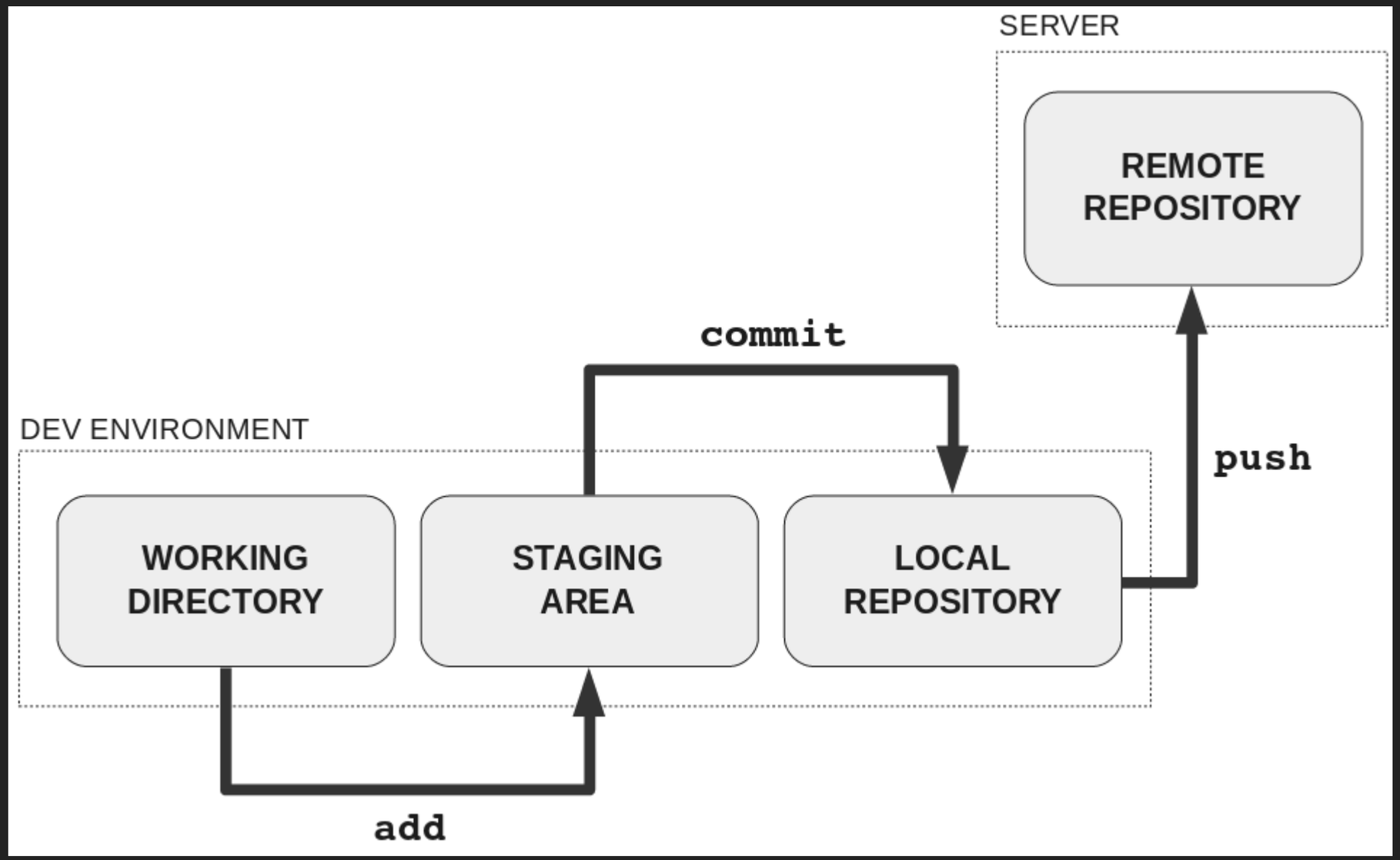
# COMMIT MESSAGES


https://xkcd.com/1296/

- It’s important to write proper and informative messages for your commitments.
- Help your teammates and your future self.
- Useful Guide: https://chris.beams.io/posts/git-commit/

# PUSHING YOUR CHANGES

- Push changes with `git push -u origin master`
- Origin: the name of the remote repository
- Master: your current branch
- The command “-u” sets the upstream. Your upstream is where you “send” the code.
  - Doesn’t need to be on github. It can be any repository


SERVER
REMOTE REPOSITORY
commit
DEV ENVIRONMENT
push
WORKING DIRECTORY
STAGING AREA
LOCAL REPOSITORY
add

# BRANCHES

- Create a branch with `git <branch name>`
- Note: Creating a new branch will not switch you to it
- Using `git checkout -b [branch]` will both create and switch to the branch
- Change to the new branch with `git checkout <branch>`

# RESTORING FILES

- Some times, you may need to restore a deleted file, or roll a file back to its last version.
- To do this, run `git checkout [commit] <filename>`

# VIEWING A PREVIOUS COMMIT

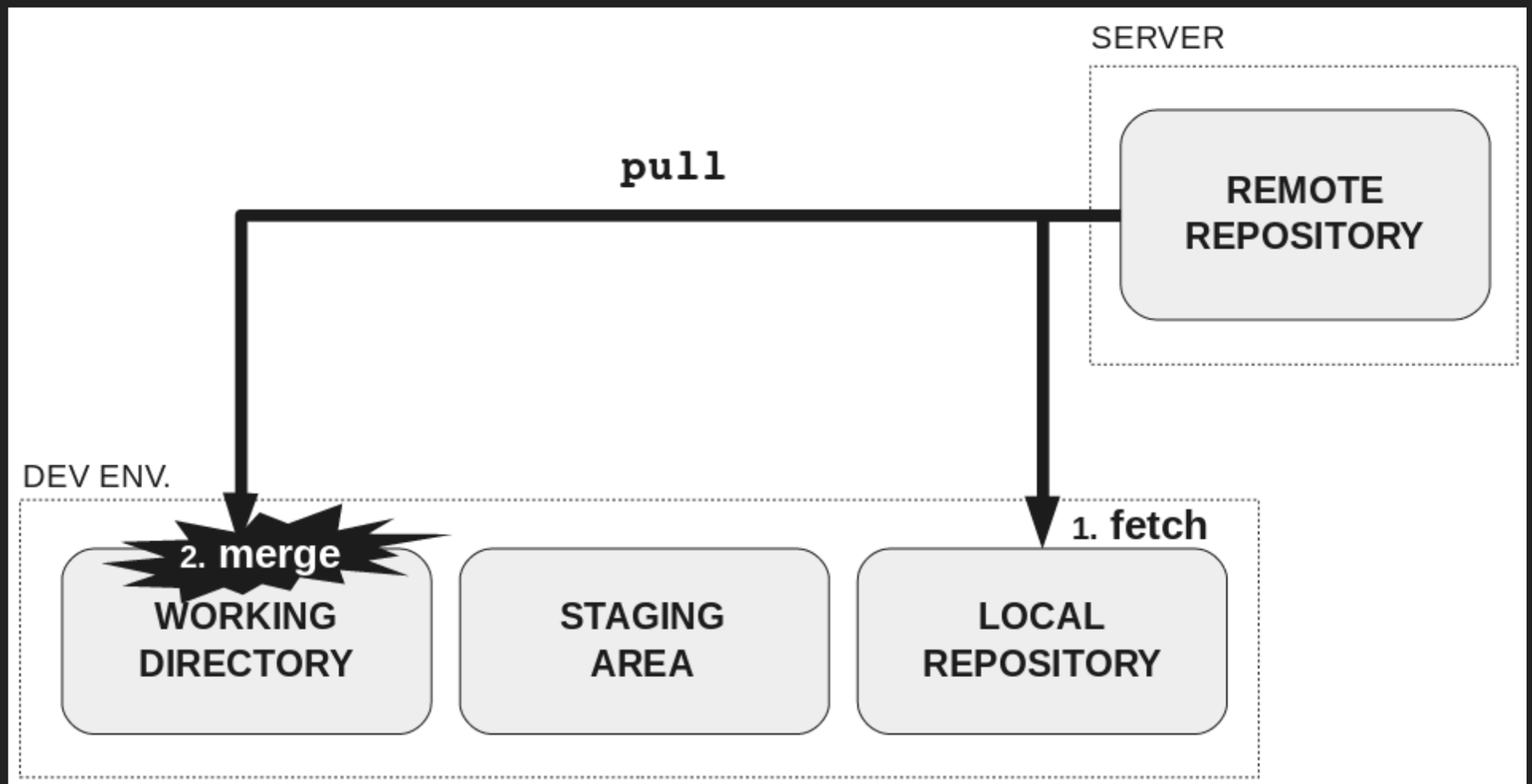
- To view a log of the commits run the command `git log`
- To see how the file changed in the last commit, run `git show <First few character of Hex>`
- This way you can find and shame the person who broke your build
- For run `git blame <filename>` for a detailed view

# LOOKING AT CHANGES

- It can be useful when you're working on a project to see differences between commits.
- View unstaged Changes with `git diff`
- View Staged Changes with `git diff —staged`
  - Useful when writing commit messages
- Just like log, you can view the differences of a specific commit with `git diff <Commit Hash>`

# TO PULL IN YOUR CHANGES

- To get the changes from master, use the following commands:
  - `git fetch`
  - `git merge FETCH HEAD`
- You can also run the shorthand `git pull`


SERVER
REMOTE REPOSITORY
pull
DEV ENV.
1. fetch
2. merge
WORKING DIRECTORY
STAGING AREA
LOCAL REPOSITORY

# THIS IS WHERE THE FUN BEGINS

- This is where things start to get complicated.
- You will create numerous conflicts

# Merge conflicts...

- Merge conflicts occur when two people try to commit changes to the same line in a file
- They can be frustrating at first, but are easily fixed once you know what you are doing

# IGNORING FILES

- As you work on your project, you'll eventually have files that you don't want to commit.
  - Such as binary files
- Creating a `.gitignore` file in your directory will make git ignore these files.

```
#ignore all .pyc files
*.pyc
myprogram
```

# KEEP SECRETS A SECRET

- You NEVER want to commit files that contain “Secrets” to your repo.
  - It does happen
  - There have been numerous cases of people pushing keys to public Github Repos
- If you do commit a secret:
  - Invalidate your secrets
  - Reroll all of the secrets committed to be safe
  - Never use the old secret again

# In Conclusion

- What we've talked about isn't all that Git is capable of, but it's what you need to know to get using it
- The best way to learn Git is to use it, and practice!
- Understanding the tools that you use is very important, and can make your life a lot easier!

# Special thanks to

- Cam Macdonell - Lab 1 395, Winter 2017
- https://rachelcarmena.github.io/2018/12/12/how-to-teach-git.html
- https://guides.github.com/activities/forking/