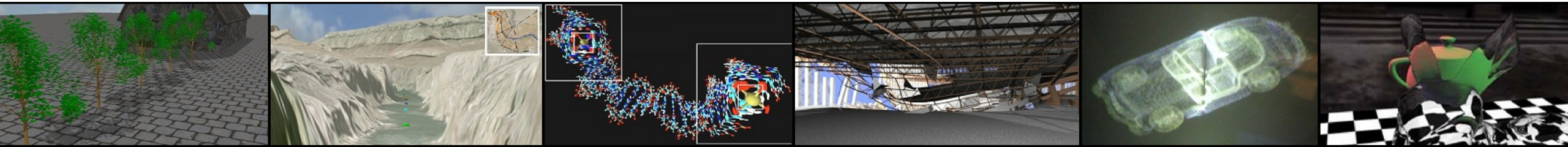
CIS 4930/6930-002
DATA VISUALIZATION

Introduction to Git

Paul Rosen
Assistant Professor
University of South Florida

(slide acknowledgments: http://excess.org/article/2008/07/ogre-git-tutorial/)



GIT IS A DISTRIBUTED VERSION-CONTROL SYSTEM

Terminology: In git-speak, a "version" is called a "commit."

Git keeps track of the history of your commits, so you can go back and look at earlier versions, or just give up on the current version and go back some earlier version.

Can be used to implement a variety of software configuration management models and workflows

GIT IS A DISTRIBUTED VERSION-CONTROL SYSTEM

You keep your files in a *repository* on your local machine.

You synchronize your repository with a remote repository on a server (in our case, GitHub).

- You protect your code from system crashes by synchronizing with the server.
- If you move from one machine to another, you can pick up the changes by synchronizing with the server.
- If you work on a team, other people's uploads can be synchronized using the server.

GIT TOOLS

A collection of many tools

- Very flexible

You can do anything the model permits

- Including shooting yourself in the foot

Need to understand the underlying model

Groups of Git commands

Setup and branch management

- init, checkout, branch, **clone**

Modify

- **add**, delete, rename, **commit**

Get information

- **status**, diff, log

Create reference points

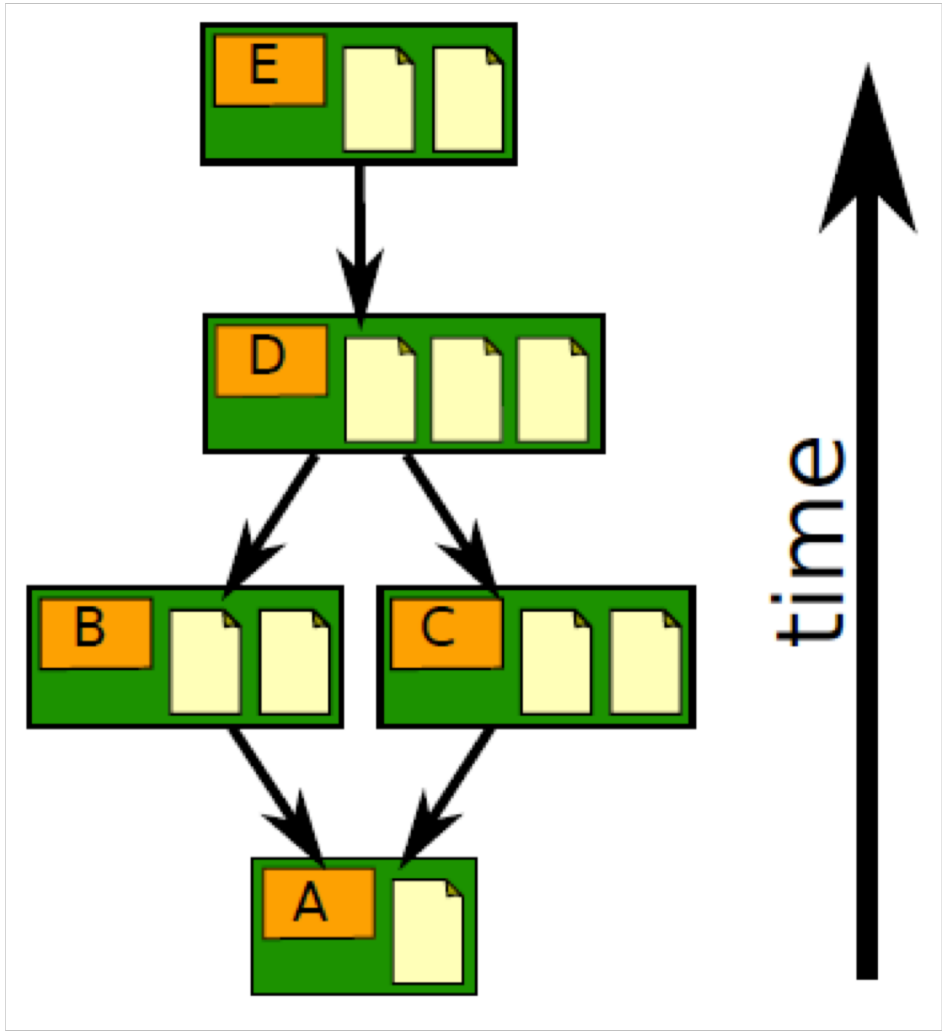
- tag, branch

Repository Contains

files & directories

commits

ancestry relationships



http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

ANCESTRY GRAPH FEATURES

form a directed acyclic graph (DAG)

Commits

Snapshots of file status

Tags

identify versions of interest
including "releases"

Branches

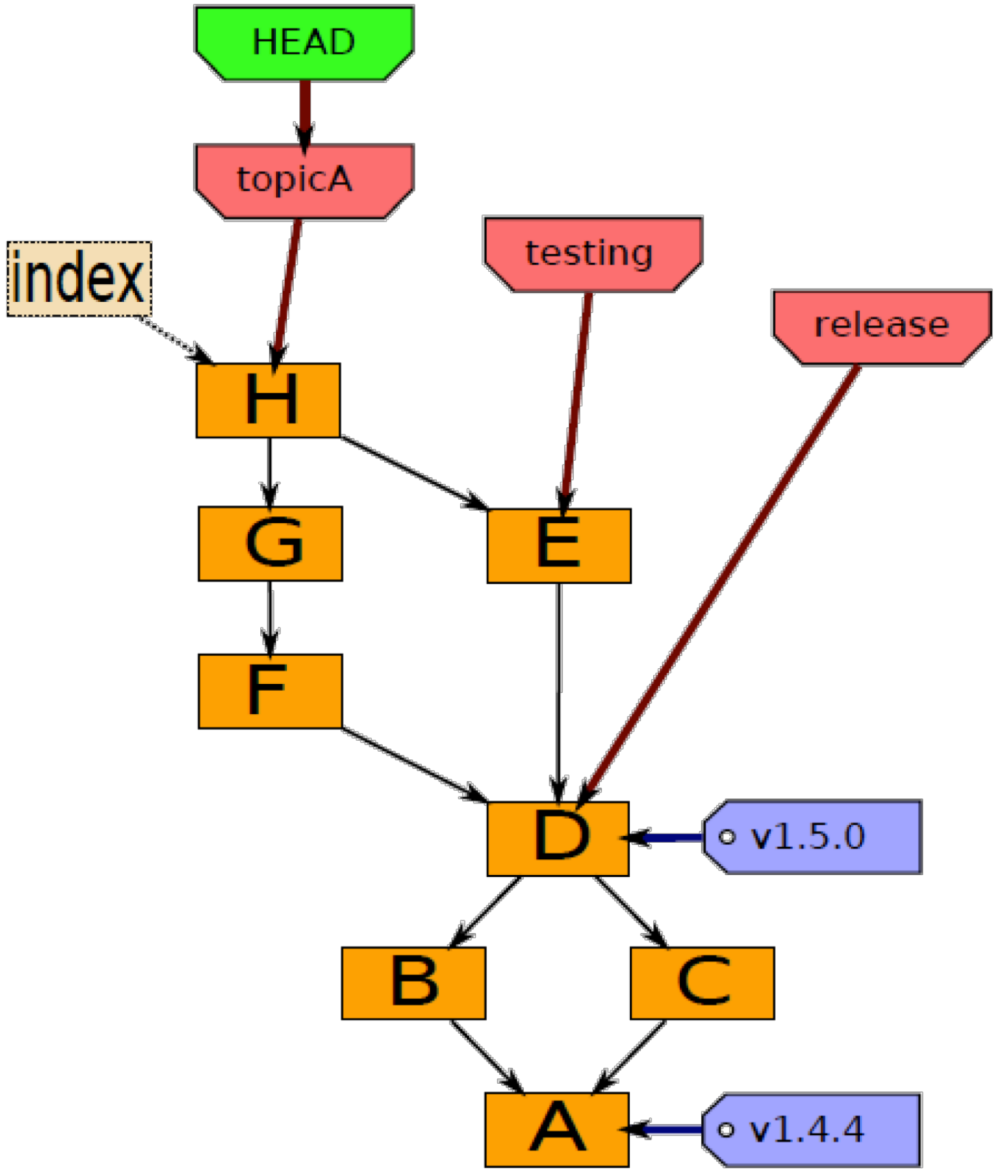
divergent path for source code modification

HEAD

is current checkout
usually points to a branch

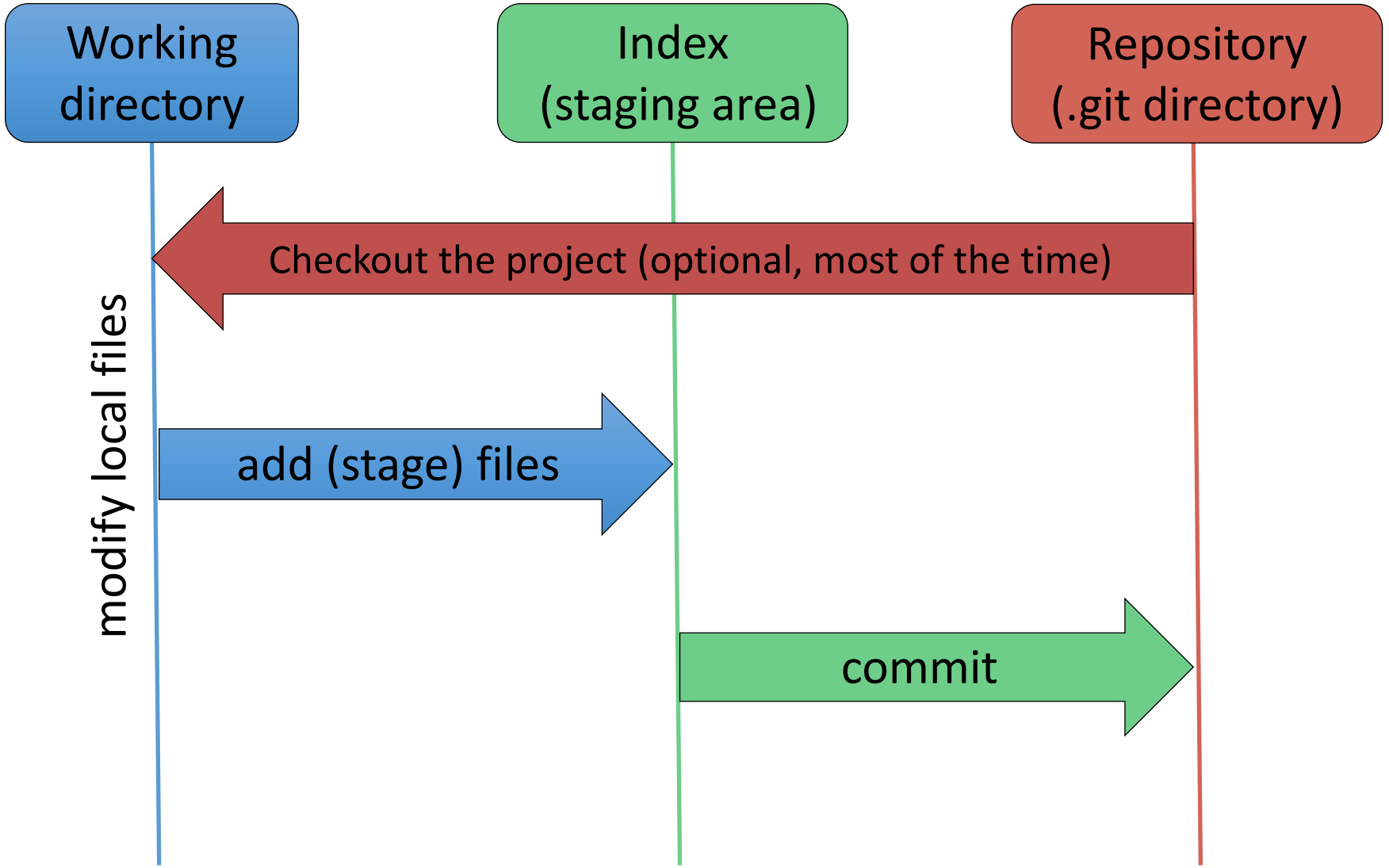
Index

"staging area"
what is to be committed

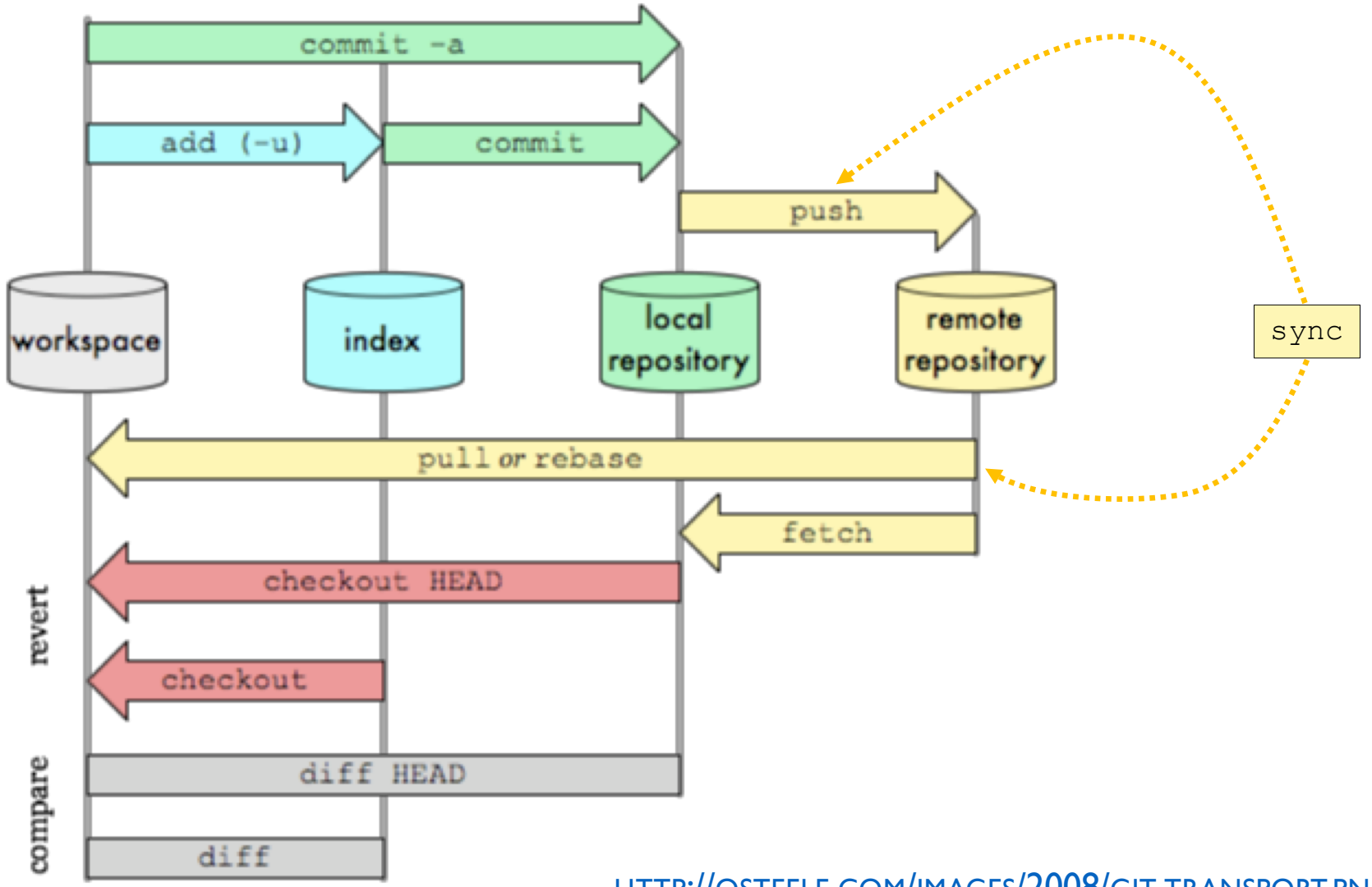


HTTP://EDGYU.EXCESS.ORG/GIT-TUTORIAL/2008-07-09/INTRO-TO-GIT.PDF

Local Operations



GIT TRANSPORT COMMANDS



HTTP://OSTEELE.COM/IMAGES/2008/GIT-TRANSPORT.PNG

Git Software

Windows

Git command line tools – https://git-scm.com/download/win

Git GUI – https://tortoisegit.org/ (also requires download of command line tools)

MAC

Install xcode and the command-line tools

https://developer.apple.com/xcode/

http://railsapps.github.io/xcode-command-line-tools.html

Linux

git should already be installed. If not, use the appropriate package manager (e.g. apt or yum) to install it.

Getting Started

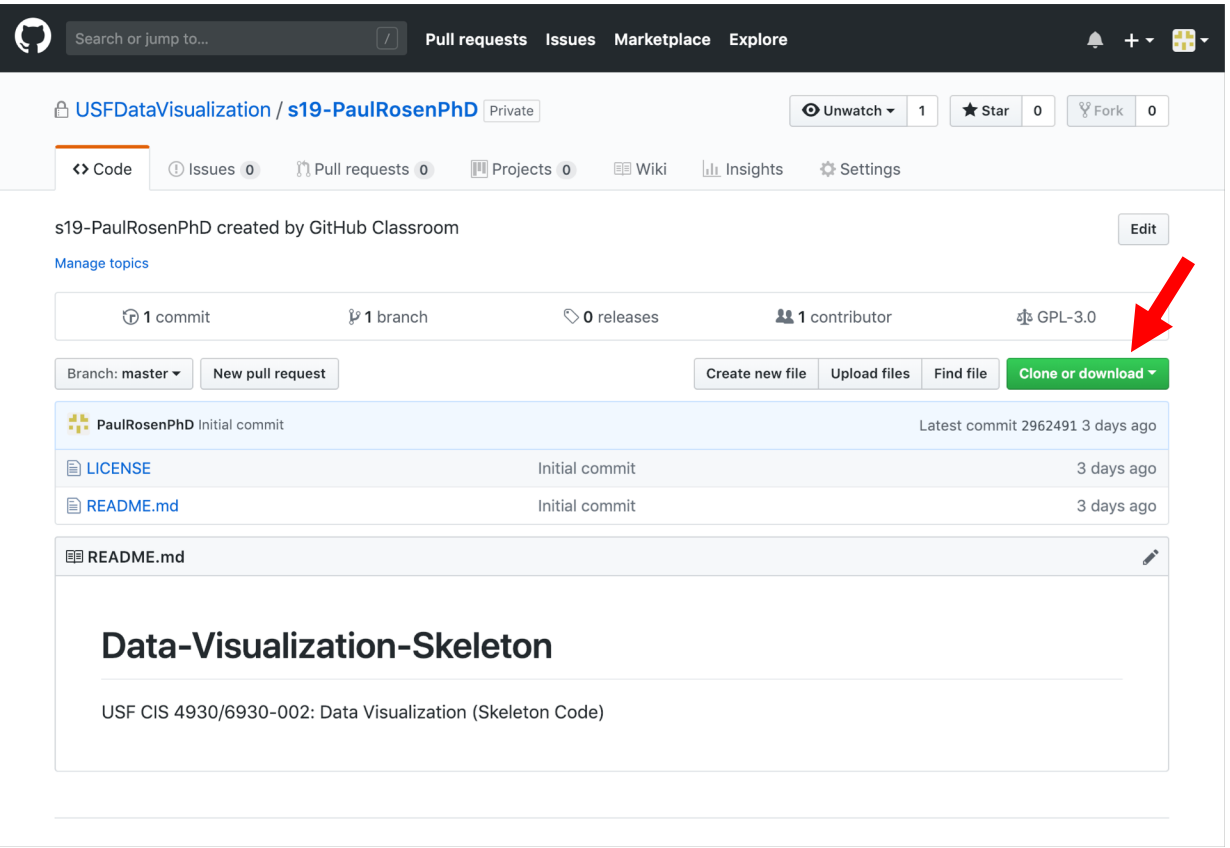
Create a github account, if you don't already have one (https://github.com/)

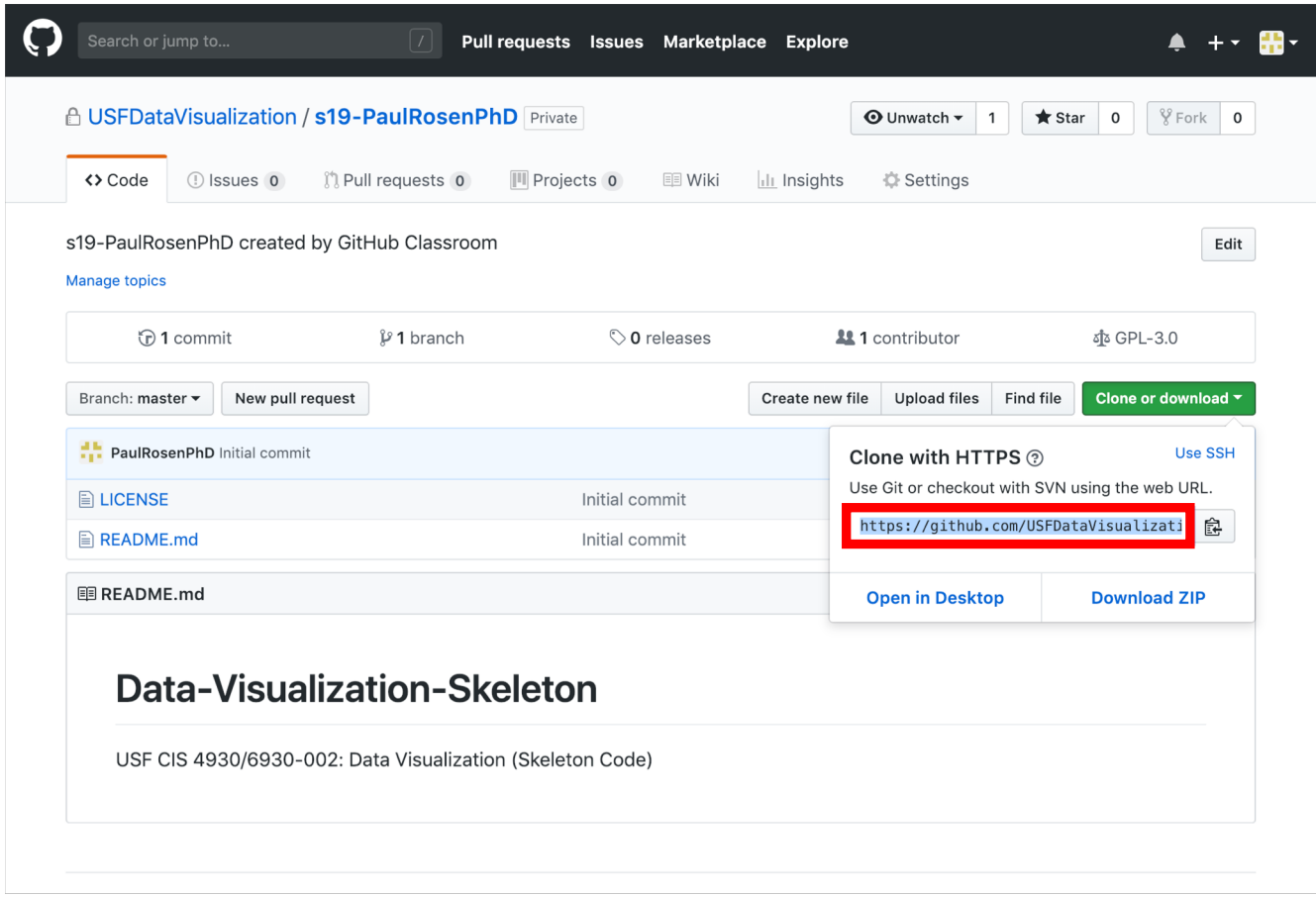
GitHub Education account is optional (https://education.github.com/discount_requests/new)

Visit <https://classroom.github.com/a/0JcYQJsF> to setup your repository

Once the repository is created (this can take a few minutes) determine the remote path and pick a local directory for code.

Finding Remote Path

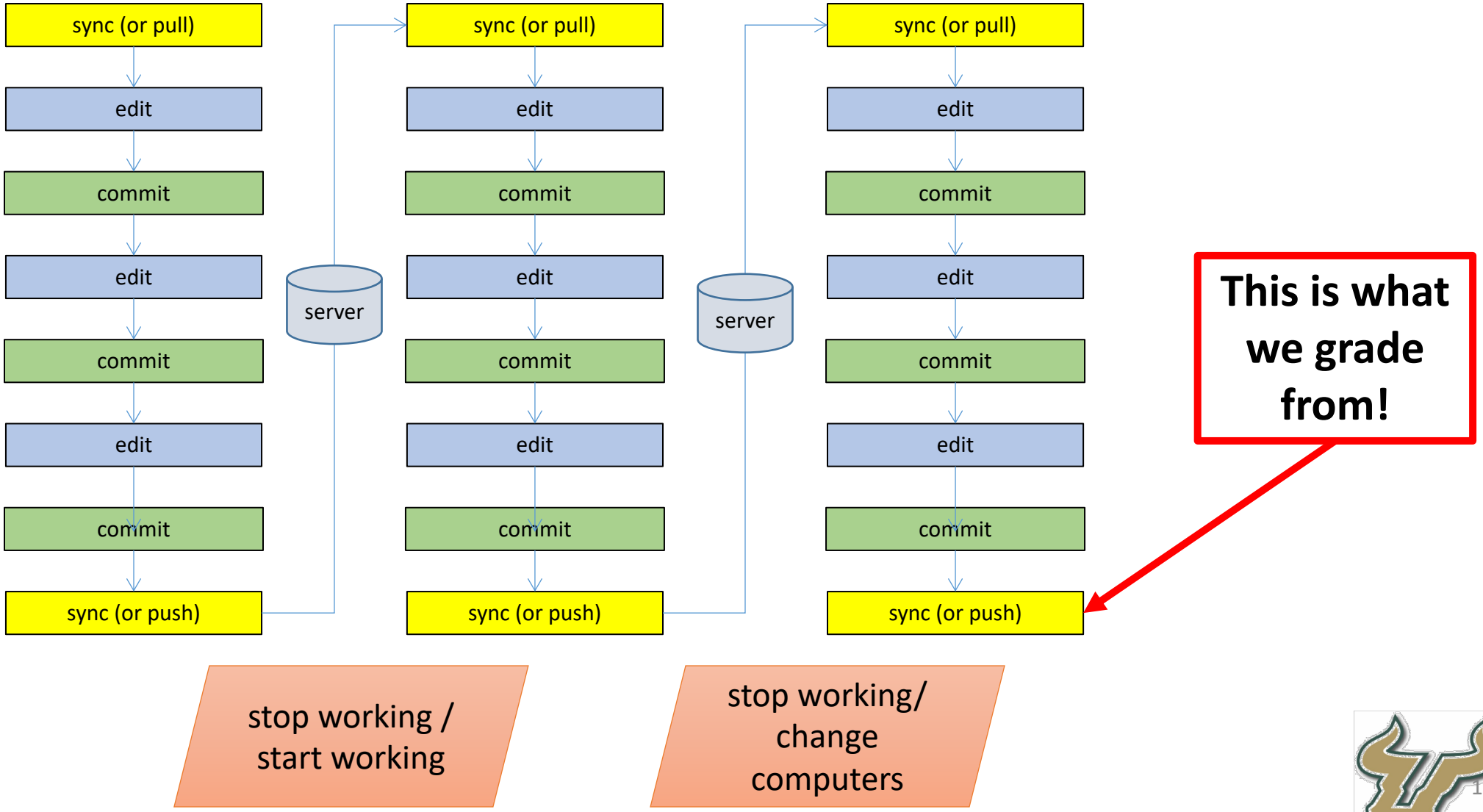




SAMPLE SESSION COMMANDS

```
> git clone <remote_path> <local_directory>
> cd <local_directory>
> git pull
> touch newfile.txt
> git add newfile.txt
> git commit –m “added a new file”
> git push
```

SUGGESTED WORKFLOW



References

http://book.git-scm.com/index.html

http://excess.org/article/2008/07/ogre-git-tutorial/

http://www-cs-students.stanford.edu/~blynn/gitmagic/

http://progit.org/book/

http://www.geekherocomic.com/2009/01/26/who-needs-git/

Many YouTube videos