# Git, GitHub, and Jupyter

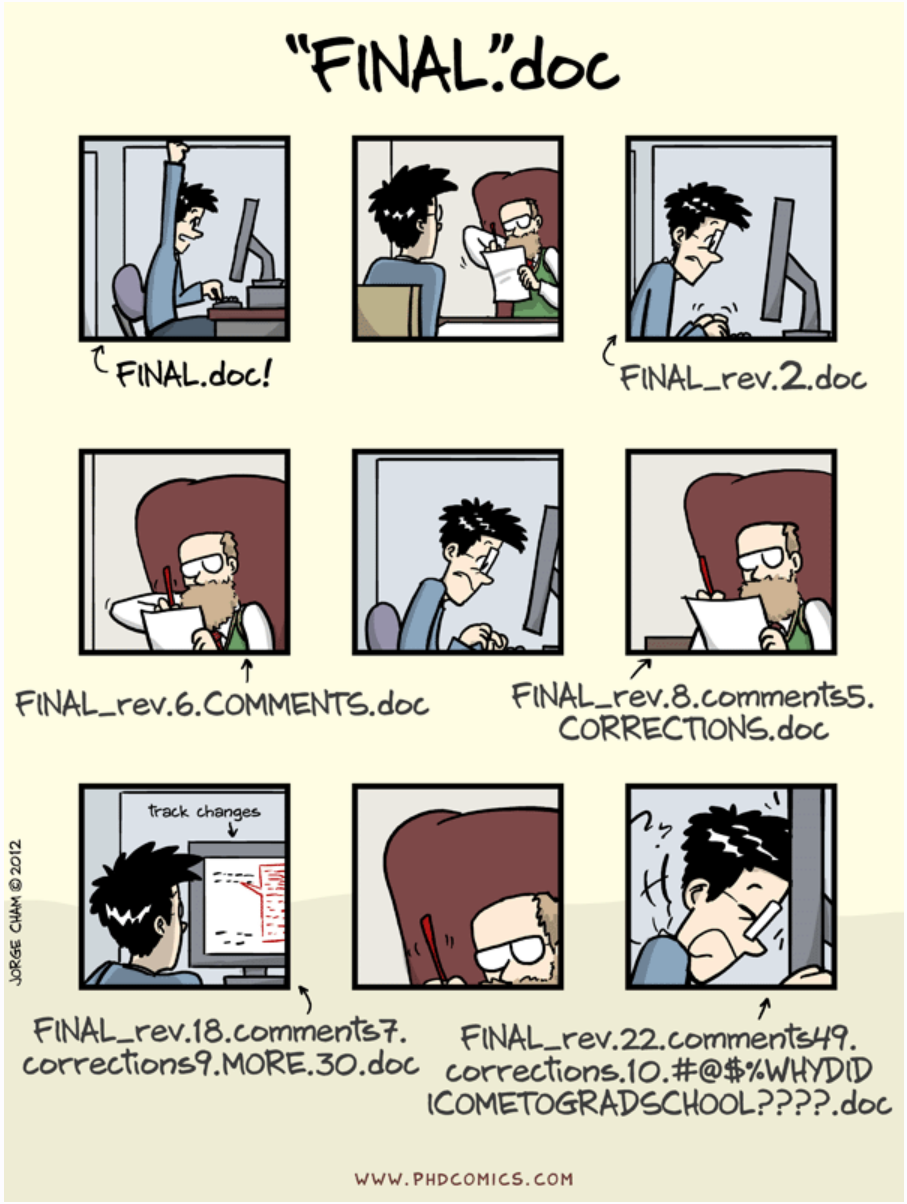
## Lecture Notes

Ty Janoski
City College of New York, CUNY

# Learning objectives:

- Understand **version control** and its role in collaborative software development.
- Learn to effectively use **Jupyter notebooks** for data analysis and collaborative documentation.
- Develop skills to set up and manage **collaborative workflows** with Git/GitHub and Jupyter for group projects.

# An introduction to version control


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

“Version control is the practice of tracking and managing changes to software code.”

# Git

- Git: version control system
  - tracks changes to code over time
  - enables recall of specific versions
- Makes it easier for developers to work together
  - Lets users create *branches* to work independently and then *merge* the changes back into the main code
  - Each developer has a local copy of the code

# Git basics

1. Add file(s) to a staging area.

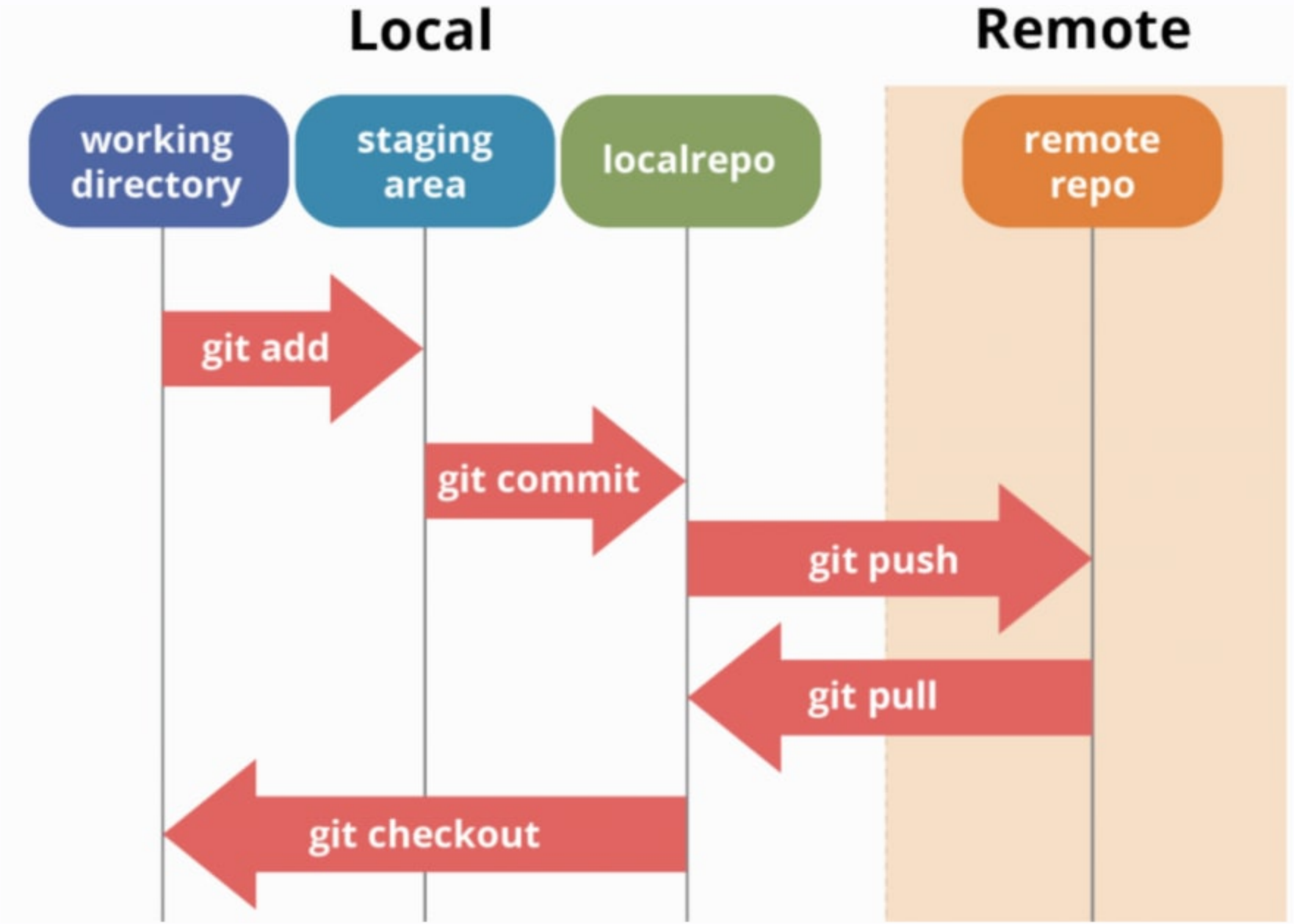
   ```
   git add somefile.py
   ```

2. Save changes to a *local* repository.

   ```
   git commit -m "Made some awesome changes!"
   ```

3. (Opt.) Upload changes to *remote* repository.

   ```
   git push
   ```


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout

# Git is helpful on its own

## But it really shines when used with GitHub!

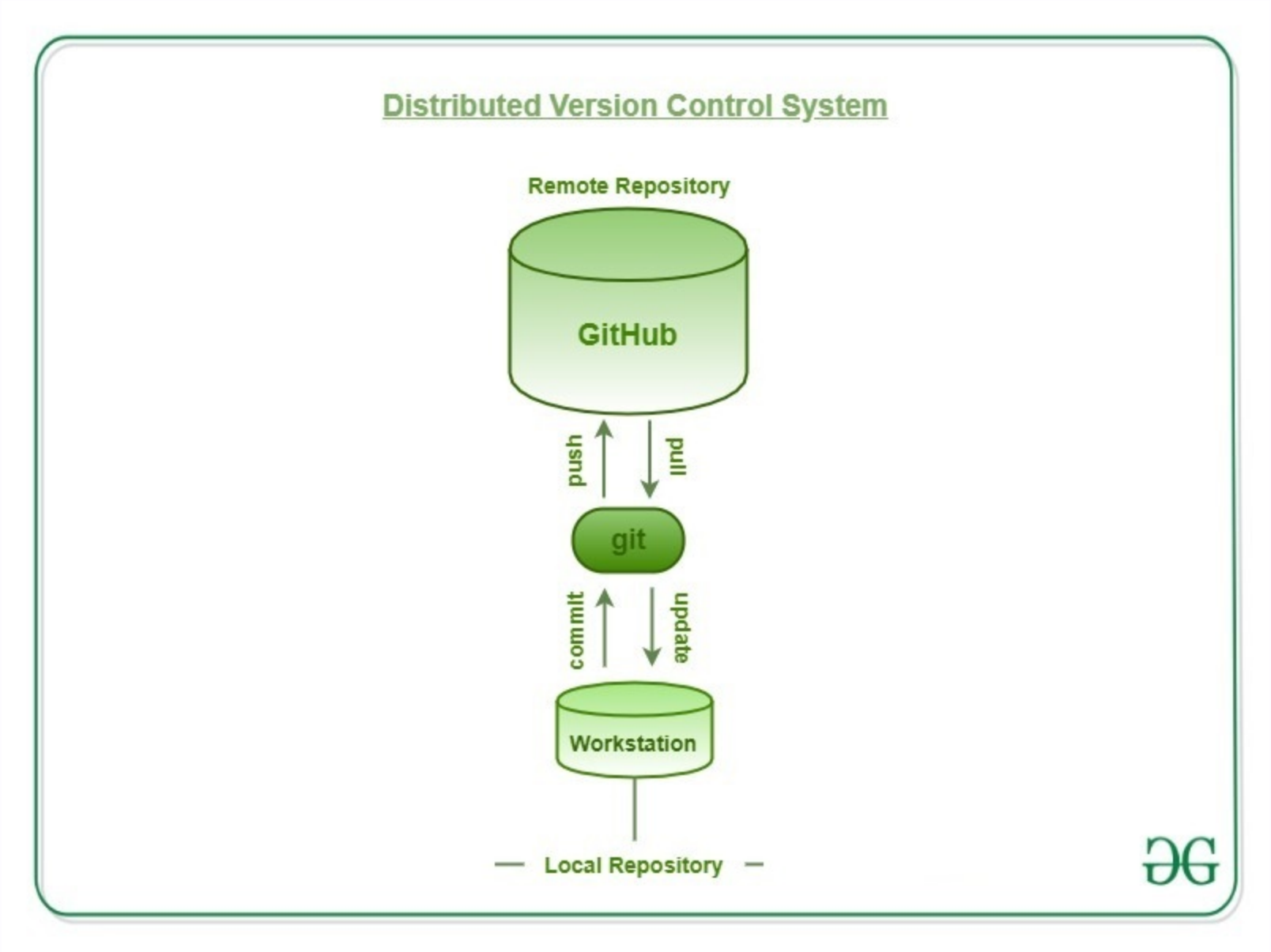
# GitHub

# GitHub

GitHub is a website where you can host your code & files in databases called "repositories."

I don't fully understand how GitHub gets money but they offer tons of services for free.

# Typical GitHub workflow



# We can go the other way, too.

- `git clone github.com/somerepository.git`
  - Creates a copy of an entire remote repository
- `git pull`
  - "Pulls" the latest changes and commits from the remote repository to your local working copy

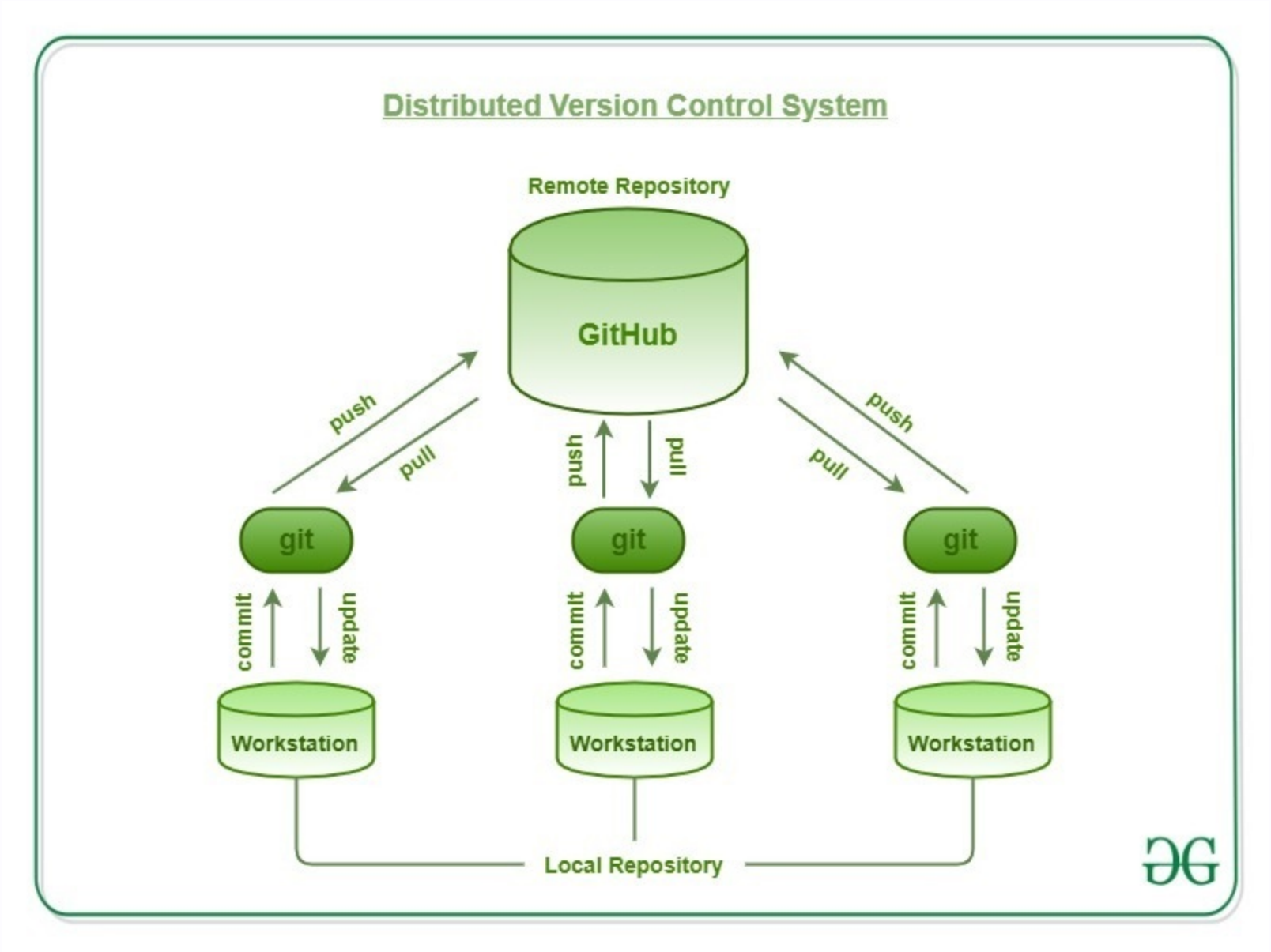
Typically, you only `clone` once but `pull` many times.

**So far, the workflows we've described are good for people working by themselves, but in this class and in the future, you will be working on a team.**

# Branching

In Git/GitHub, branching creates **a parallel version of the code** to work on new features or fixes *without affecting the main codebase*. It allows developers to isolate their work, make changes, and then merge those changes back into the main code when they are ready.

# Branching


Distributed Version Control System
Remote Repository
GitHub
push
pull
push
pull
push
pull
git
git
git
commit
update
commit
update
commit
update
Workstation
Workstation
Workstation
Local Repository

# Branching workflow

- Create and switch to branch

  ```
  git checkout -b new-branch
  ```

- Add changes and commit

  ```
  git add ...
  git commit -m "..."
  ```

- Push branch to GitHub

  ```
  git push -u origin new-branch
  ```
  *

* after the first push, you can just use `git push`

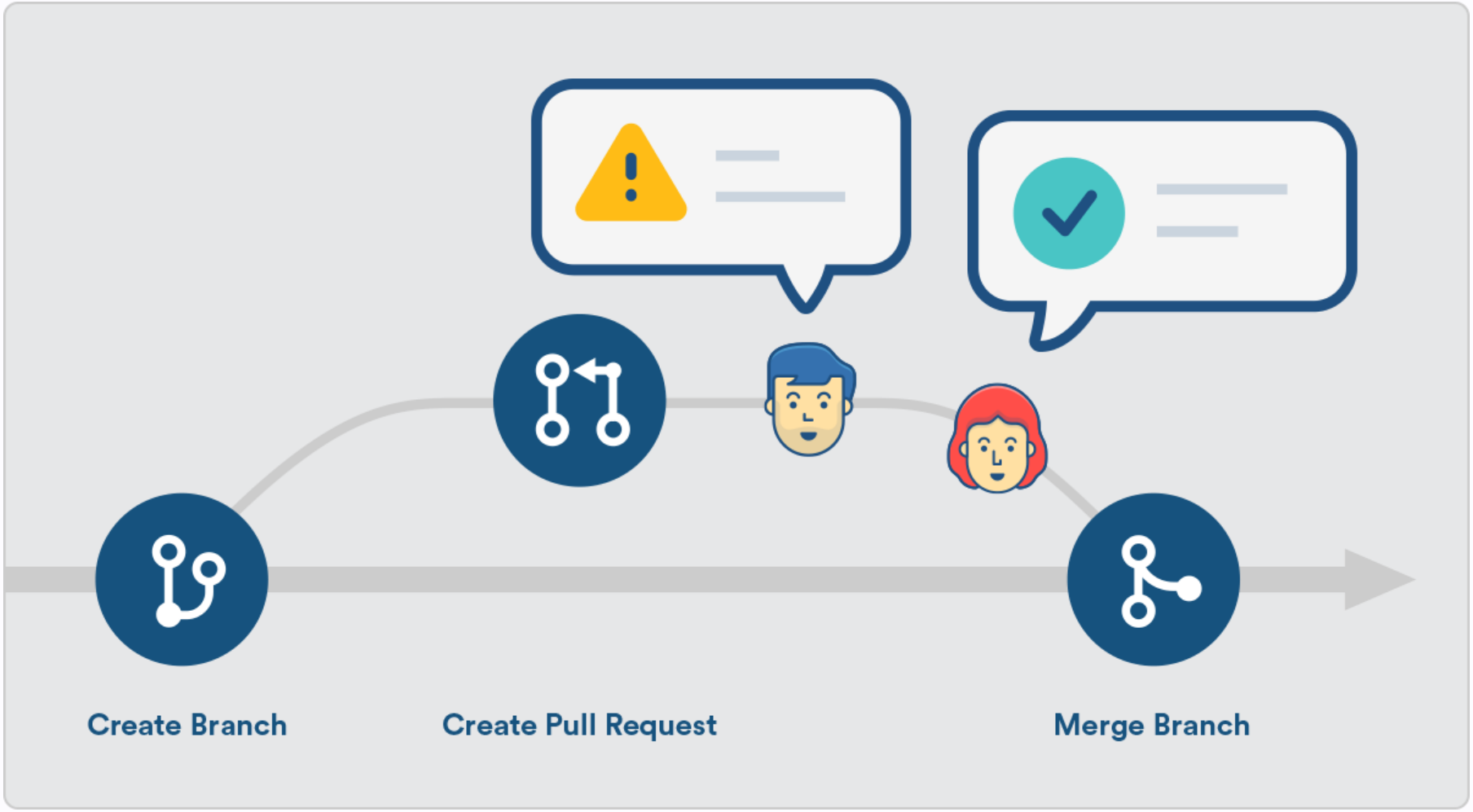
# If you want to merge changes from a branch to the main code...

- Start a **pull request**
- Review & merge

# Pull requests

A pull request is a way to propose changes from a branch in a repository to the main code. It allows developers to notify others about the changes they've made, discuss modifications, and eventually merge the changes into the main code.

# Pull requests



# Other helpful git tools

- `git status` = check status of your files
- `git branch` = list branches
- `git rm` = delete a file
- `git stash` = temporarily store your changes
- `git checkout <branchname>` = switch branches
- `git stash pop` = apply changes to current branch

Version control with Git and GitHub can be confusing. You don't need to understand everything right now to develop a collaborative workflow for this class 😄.

# Resources/References

- Learn git branching
- Git/GitHub Lecture by Madicken Munk
- GitHub Docs
- The Git book