Unit 2

•••

Git & GitHub

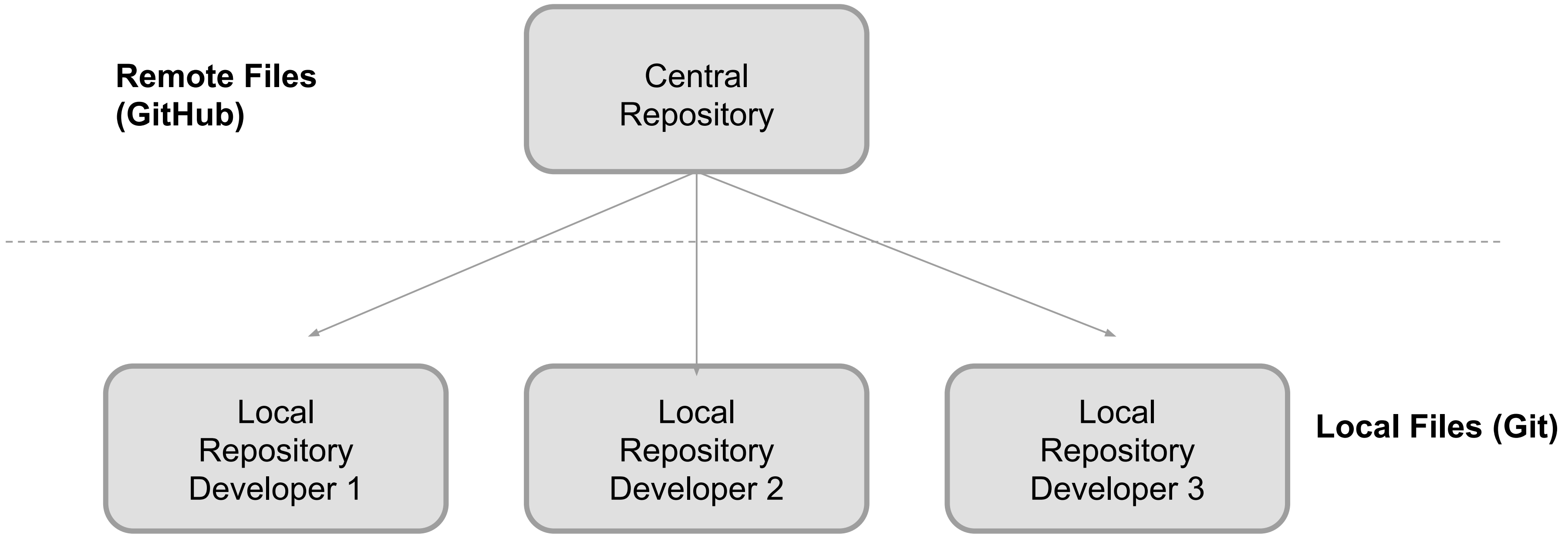
Objectives

In this lecture, we’ll talk about version control in your projects, and how to work with GitHub.

Version Control

As JavaScript developers, you’ll commonly work with Git and GitHub.

Simple Version Control Workflow


Remote Files (GitHub)
Central Repository
Local Repository Developer 1
Local Repository Developer 2
Local Repository Developer 3
Local Files (Git)

What is Version Control?

- Version control allows you to take “**snapshots**” of your project. If you make a mistake, you can backup to the last snapshot.
- Version control provides safety in allowing you to archive and **revert working code**.
- Version control is a key tool in enabling **teams** to work together collaboratively on code.

Types of Version Control

There are many types of version control software:

- **Git**
- SVN
- Mercurial
- And so on…

Here at Codesmith, and most likely in your daily life moving forward, you’ll work with Git.

Git Terminology

- **Repository** (source folder)
  - When you create Git repository (or clone one), you end up with a project folder with source files, called a repository.
- **.git** folder
  - The head references denote changes and are stored in an invisible folder with the name .git.
- **Snapshots** saved as commits
  - Commits are saved as a **linked list**, where each commit is a node with a reference back to the previous commit.
- **.gitignore**
  - Used to exclude certain files from git tracking

File System Commands

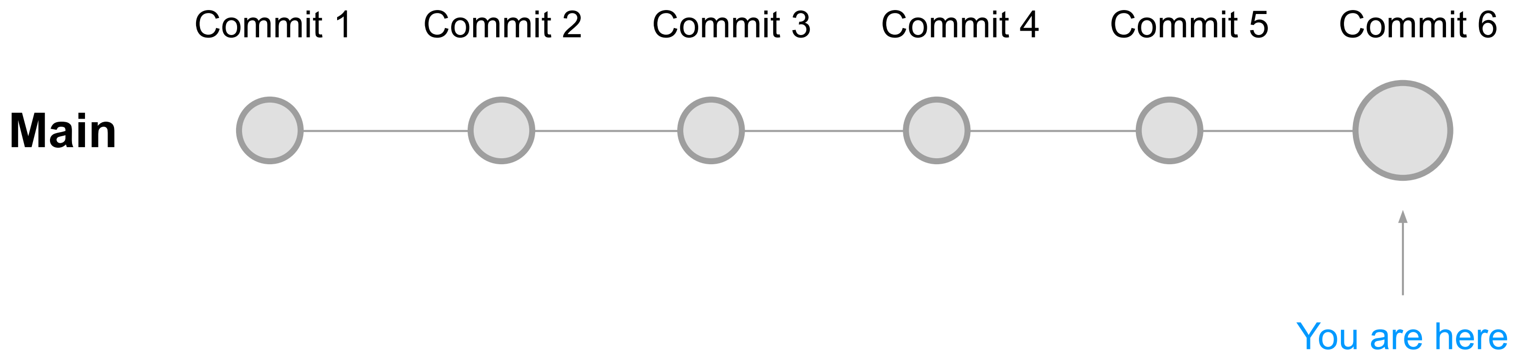
1. **mkdir**: creates a directory
2. **cd**: changes directory
3. **ls**: prints files in directory
4. **ls -a**: prints all files including hidden files
5. **touch** baz.txt: creates an empty text file
6. **echo** ‘hi’ >> foo.txt: creates a file with text in it
7. **cp** foo.txt bar.txt: creates a new file by copying a file

Git Commands

1. **git init**: initializes a folder as a Git repository
2. **git status**: prints status of files in repository
3. **git add**: adds a file to version control
4. **git commit**: commits changes
5. **git log**: logs recent operations
6. **git checkout**: creates or switches to branch
7. **git merge**: merges one branch with another

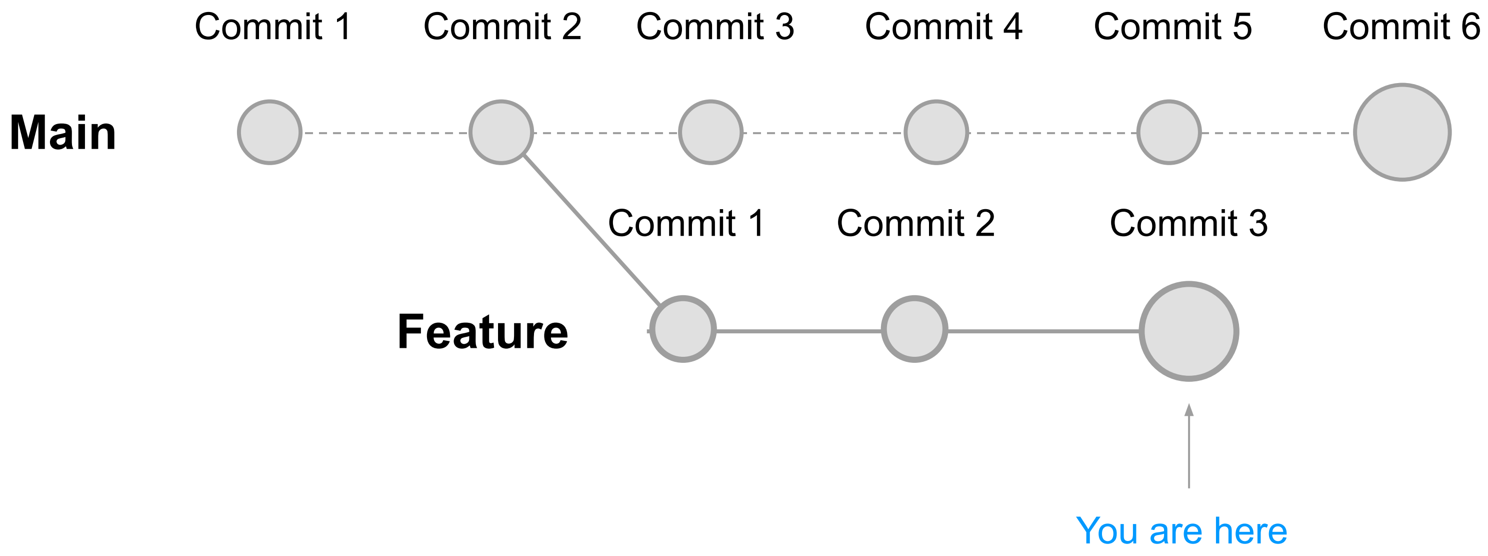
Demo: Basic Git

Nodes on the Master Branch



Demo: Resetting to Prior Commits

Creating Feature Branches



Demo: Branches

Merge Conflicts

If other people make non-overlapping changes with the master branch, there will be NO MERGE CONFLICT, and you'll simply pull the changes into your files when you merge.

If other people edit the same lines of code as your changes, then a MERGE CONFLICT appears denoted in the code.

FIX: you delete the code that should be removed, leaving only the intended code.

Demo: Merge Conflict

GitHub

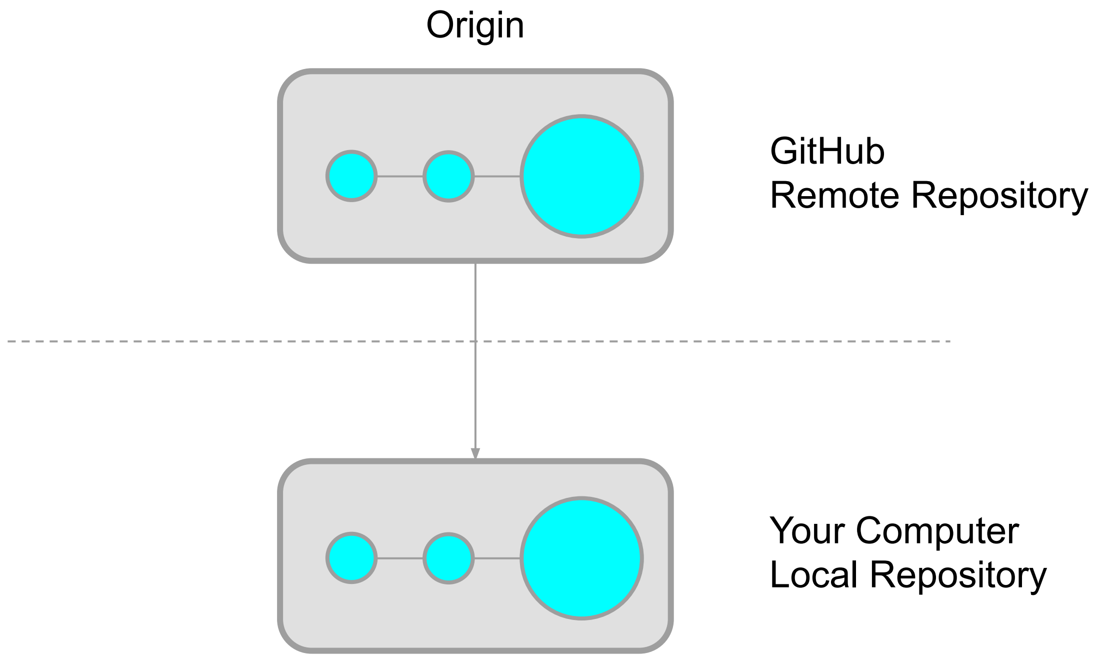
Share code and collaborate using GitHub.

GitHub

GitHub is a hosting platform for Git repositories online.

a. It’s the most popular, but there are other options:
    i. **BitBucket**
    ii. **GitLab**
b. GitHub enables you to share code and collaborate with teams.

Creating Copies



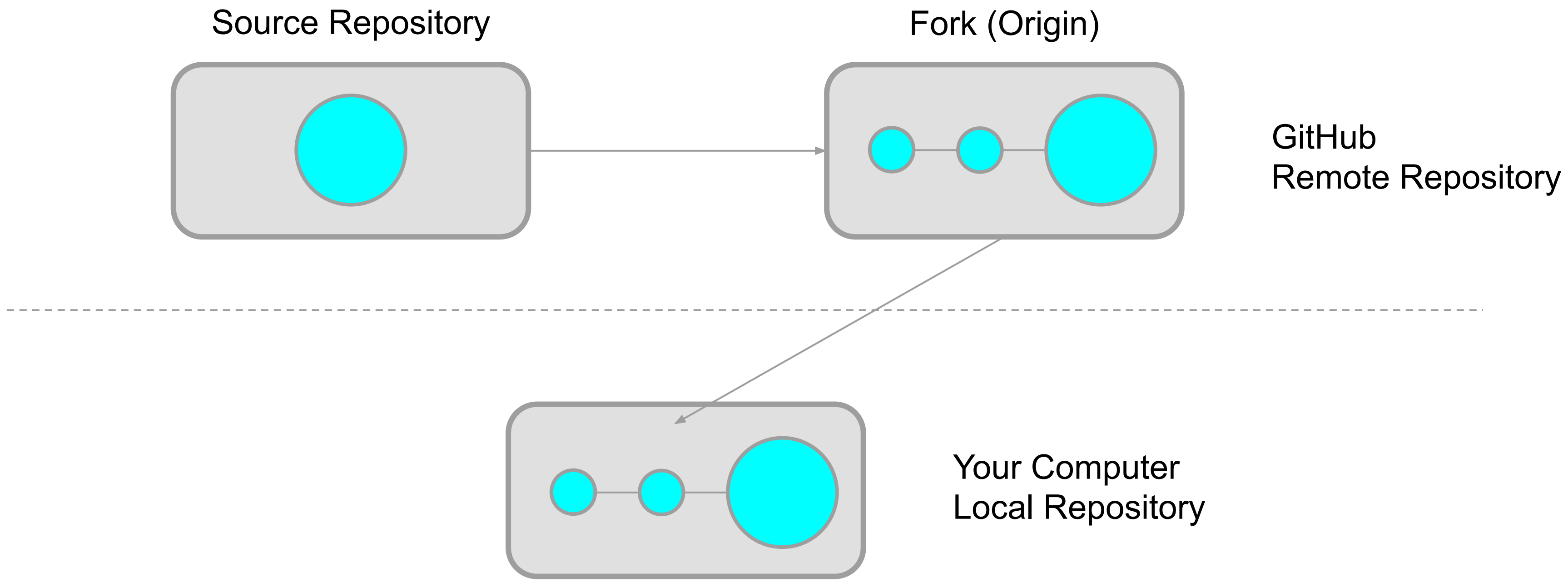
```
git clone https://github.com/yourid/reponame/
```

GitHub Workflows

The GitHub workflow is similar to the local Git workflow, except that you have a new collaborative layer called **PULL REQUESTS**.

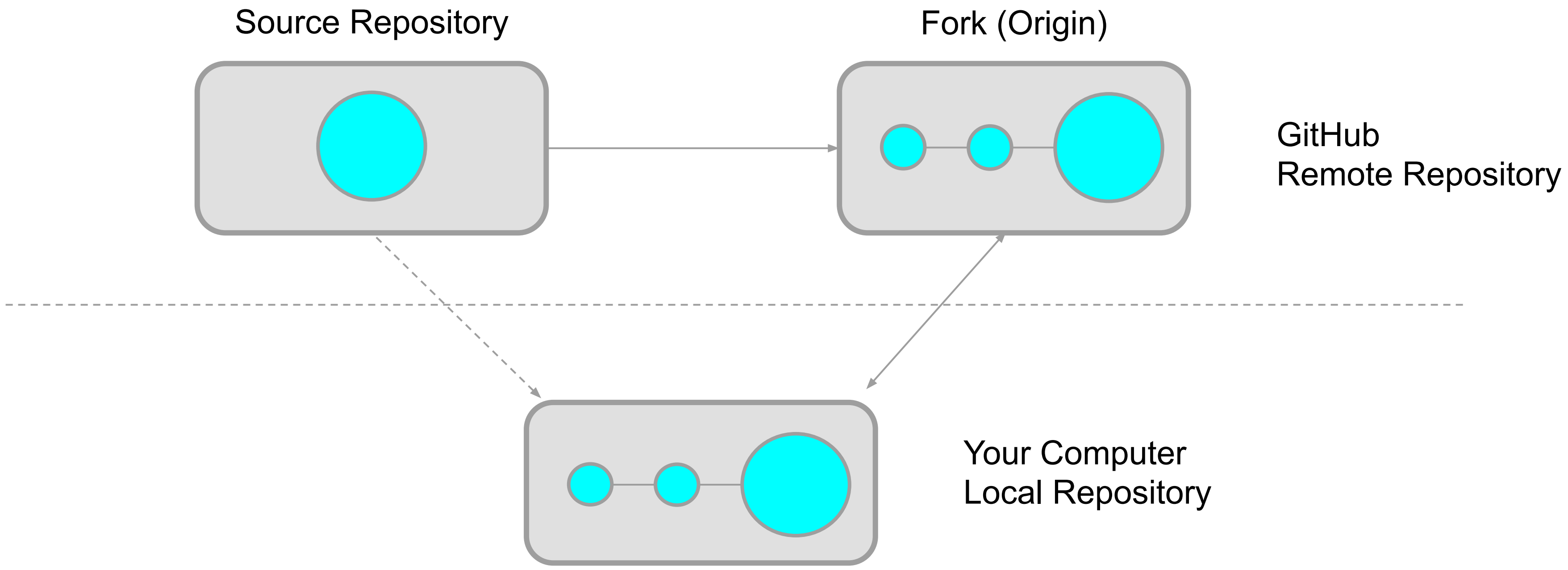
a. The idea here is that you'll work with group review before you merge changes back into the master branch. This is a safer way to go, and generally considered a best practice while working in team environments.
b. **PULL**: Downloads changes from the remote repo and affects local changes (FETCH and MERGE).
c. **PUSH**: Uploads local branch commits to GitHub.

Forking



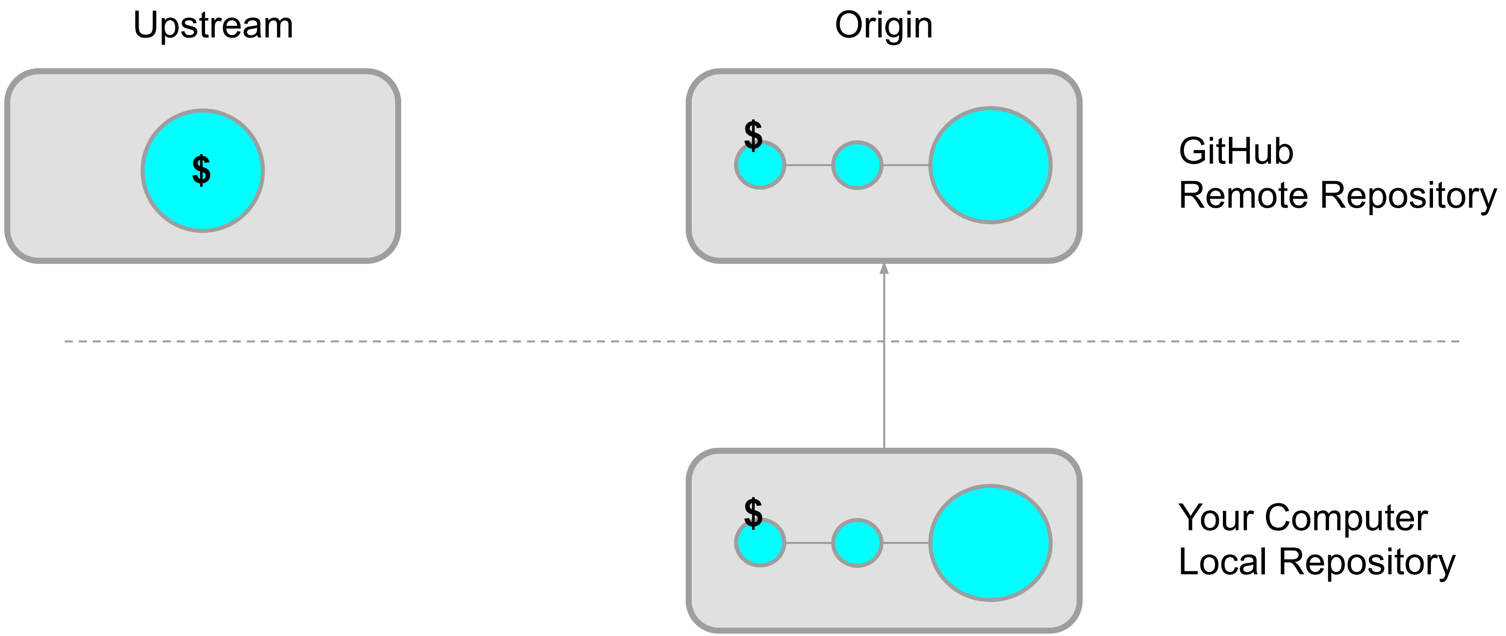
```
git clone https://github.com/yourid/repo/
```

Multiple Remotes



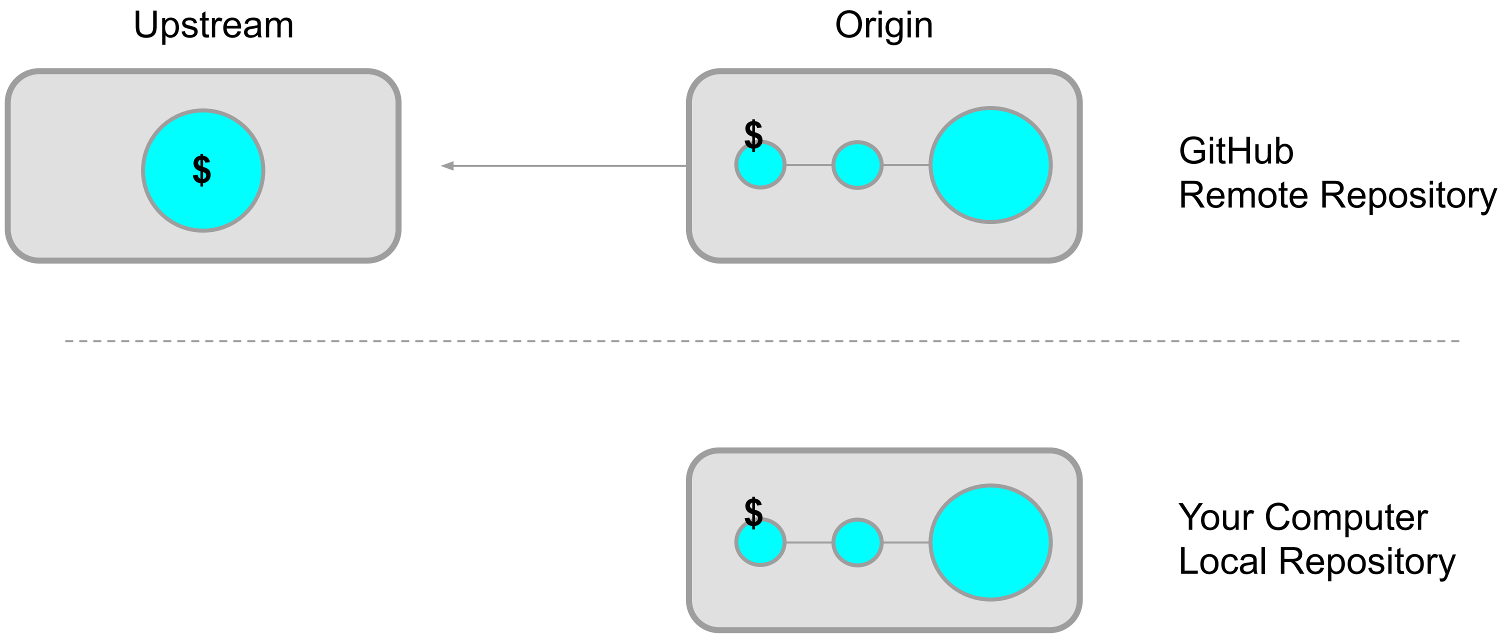
```
git remote add upstream <url>
git pull upstream master
```

Push to Origin



```
git push origin master
```

Pull Request (Merge to Master)



Submit Pull Request on GitHub

Demo: GitHub

Conclusion

- Expand your knowledge of working with the file system, and various Git commands on the Terminal. Take a look at the resources on the following slides to dive in further.
- Also, spend some more time exploring collaborative workflows on GitHub.

Further Reading

Workflows

- Understanding the GitHub Flow (GitHub)
- Comparing Git Workflows (Atlassian)
- Branching Workflows (Git)
- Pro Git (Git)

Cheat Sheets

- Git Cheat Sheet (GitHub)
- Linux Command-Line (Computerworld.com)

Interactive tools

- Learn Git Branching (Git)