Source Control

Why use souce control at all?

Version Control Systems

- Git
- Subversion
- Mercurial

Git

Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Benefits

- Frictionless Context Switching
- Role-based Codelines
- Feature-based Workflow
- Disposable Experimentation

Terminology

Repository: The place where code is stored, the folder for the project.

Terminology

Clone: To create a local working copy of an entire repository.

Terminology

Branch: A finite, isolated development workspace. A repository can have an arbitrary number of branches.

Terminology

Checkout: To make a copy of a branch or batch of commits on your local machine.

Terminology

Head: The current state of your current branch, the place where you're doing your work.

Terminology

Commit: A single point of history for a branch or repository.

Terminology

Stage: The area where changes are placed prior to forming a commit.

Terminology

Fetch: Viewing the latest changes from a remote repository without merging them into your local working copy.

Terminology

Merge: To combine the contents of one branch into another.

Terminology

Pull: The combination of fetch and merge.

Terminology

Push: To send commits to the remote repository.

Terminology

Diff: The difference in changes between two commits or branches.

Terminology

Conflict: An irreconcilable diff.

Terminology

Tag: Used to mark a particular commit in a repository. Often used to mark releases.

Terminology

Upstream: The primary branch of the original repository. Branches made off of it are called downstream.

Terminology

Rebase: To apply a series of changes from one branch to a different base, resetting the head of that branch.

Terminology

Fork: Create an identical, but separate repository

Terminology

Pull Request: Proposed changes to a repository.

First Time Git Setup

```
$ git config --global user.name "Firstname Lastname"
$ git config --global user.email "firstnamelastname@example.com"
```

```
$ git config --global core.editor "atom --wait"
```

```
$ git config --list
$ git config user.name
```

How to Use Git: Existing Repository

```
$ git clone https://github.com/uagc-it-readiness/git-lecture-example.git
$ git status
$ git log
$ git remote
```

Exercise 1

How to Use Git: New Repository

```
$ git init
$ git remote add origin <url of remote repository>
$ git status
$ git remote
```

Exercise 2

Branch

```
$ git branch
$ git branch feature/bb-math
$ git checkout feature/bb-math
```

Stage

```
$ git add math.js
$ git add '*.js'
```

```
$ git reset math.js
```

Commit

```
$ git commit -m 'add math file'
```

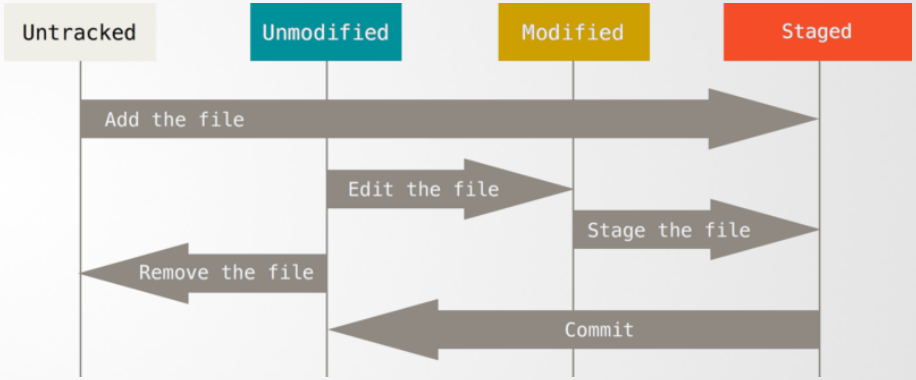
Push

```
$ git push origin feature/bb-math
```

Exercise 3

Git File States

The different stages a file can be in, and the actions from how a file gets from one state to another.



Exercise 4

Fetch

```
$ git fetch origin
```

Pull

```
$ git pull
```

Merge

```
$ git merge feature/bb-merge-no-conflict
```

Merge Conflicts

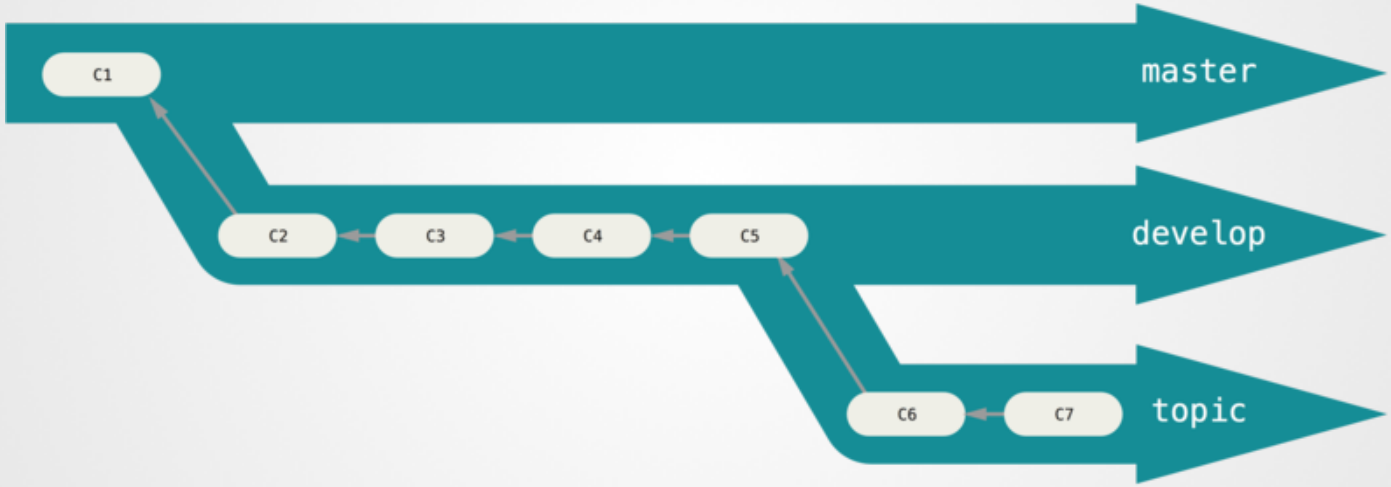
```
$git mergetool
$git commit -m 'merge branch name'
```

Exercise 5

Delete

```
$ git branch -d bb-math
```

Typical Branch Structure



Exercise 6

Pull Requests

Git Ignore

```
$touch .gitignore
$git rm --cached nameoffile
```

SSH Keys

```
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
$ eval "$(ssh-agent -s)"
$ ssh-add ~/.ssh/id_rsa
$ pbcopy < ~ /.ssh/id_rsa.pub
$ ssh -T git@github.com
```

ONLY DO THIS ON COMPUTERS YOU OWN