UNIVERSITY of WASHINGTON

DATA 515A

# Version Control I

Bernease Herman[1,2], Joseph Hellerstein[1,2]
[1]eScience Institute
[2]Computer Science

April 7, 2020

# Agenda

1. Confirm sign-in to GitHub, send with UW Net ID and student number to instructors
2. Version Control, Git, and GitHub
3. Hands on practice with Git & GitHub for individuals
4. **On separate video capture:** Turning in homework

# Why use version control?

Compare writing software with writing a manuscript.

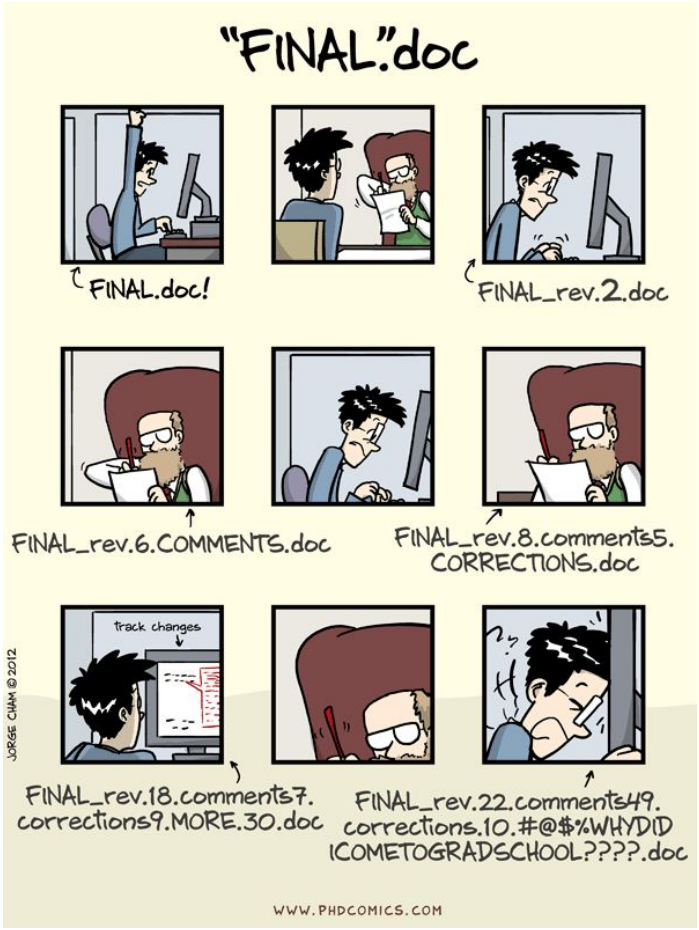
- Use undo to revert to a previous state
- Use track changes when sharing document with advisor

Still, word processors can still be frustrating.

- Intelligently combine changes made concurrently
- Record reasoning (why? for code, what?) for changes
- Efficient use of file storage

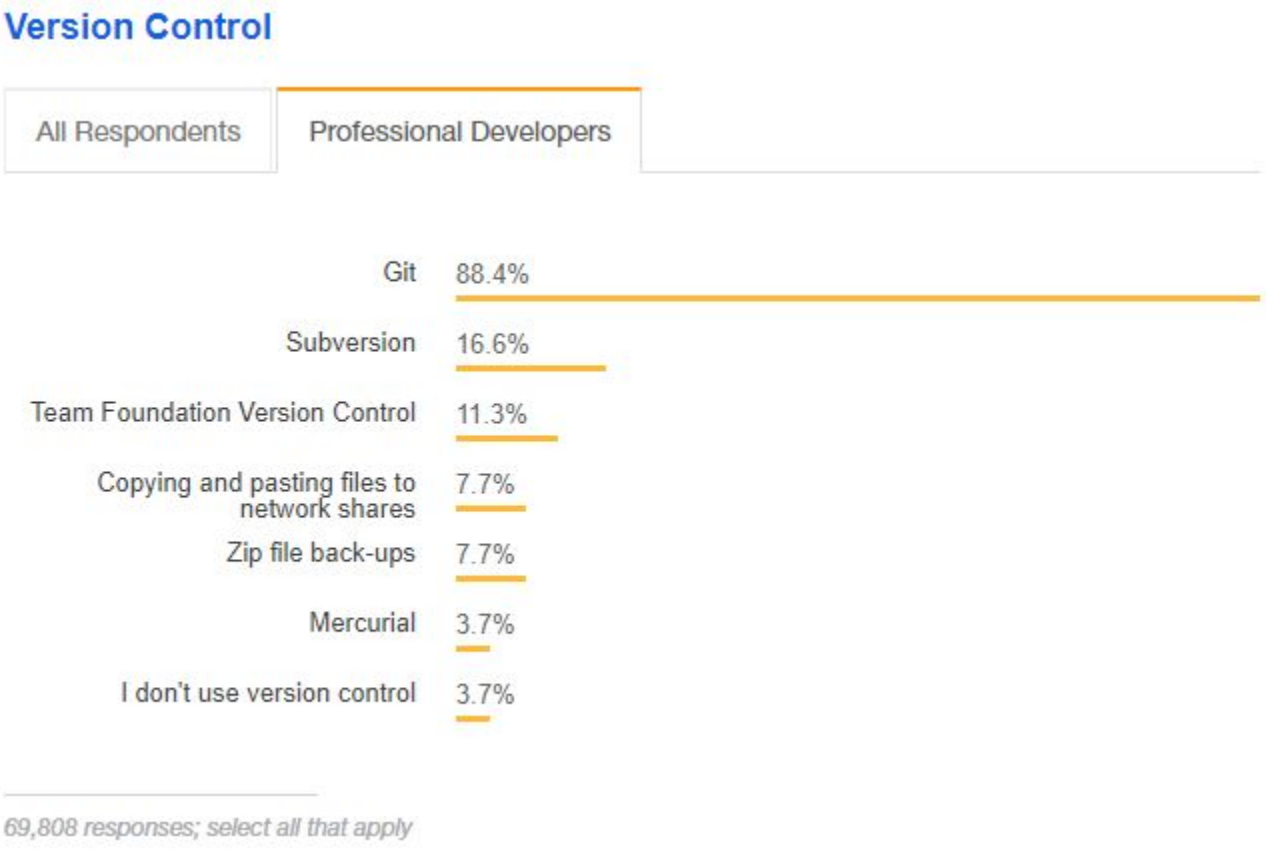
Modern version control systems can address all of these.

# Tracking versions and efficient storage




http://phdcomics.com/comics/archive_print.php?comicid=1531

# Stack Overflow Developer Survey 2018



# Git is a de facto standard for VCS

Benefits (**+++++**)

- **Performance:** operations in Git are optimized and fast
- **Flexibility:** doesn't require use of a particular workflow
- **Security:** protection against untraceable changes
- **Popularity:** employment, available on many platforms
- **Distributed:** can be used offline, no need for server

Downsides (**-**)

- **Distributed:** not ideal for large files, merging changes

# Version control in the cloud, GitHub

A working copy of your repository stored on GitHub's servers connected and accessible via the internet

- Others can download and use your code
- Online repository is suitable as central repository
- Social features, i.e. issue tracker, comments, notifications

Alternatives: Atlassian's BitBucket, GitLab, SourceForge

# Hands on Git / GitHub I



# Introduction To Git
## #(and some GitHub)

## 0. Set up

> git config [options]
> git init
> .gitignore

# Introduction To Git
## #(and some GitHub)

## 0. Set up

> git config [options]
> git init
> .gitignore

## 1. Make Changes

(use your preferred editor and tools.)

# Introduction To Git
## #(and some GitHub)

## 0. Set up

```
>git config [options]
>git init
>.gitignore
```

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

# Introduction To Git
## #(and some GitHub)

## 0. Set up

```
>git config [options]
>git init
>.gitignore
```

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

## 3. Create snapshot

```
>git commit
>git commit
  -m "[msg]
```

# Introduction To Git
## #(and some GitHub)

## 0. Set up

```
>git config [options]
>git init
>.gitignore
```

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

## 3. Create snapshot

```
>git commit
>git commit
  -m "[msg]"
```

# Introduction To Git
*(and some GitHub)

## 0. Set up
> git config [options]
> git init
> .gitignore

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files
> git add
> git add -A
> git rm [path]

## 3. Create snapshot
> git commit
> git commit -m "[msg]"


## 4. Explore
> git status
> git log [options]
> git show [sha1]

(Repeat 1-4 as desired.)

# Introduction To Git
*(and some GitHub)

## 0. Set up
>git config [options]
>git init
>.gitignore

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files
>git add
>git add -A
>git rm [path]

## 3. Create snapshot
>git commit
>git commit -m "[msg]"



## 4. Explore
>git status
>git log [options]
>git show [sha1]

(Repeat 1-4 as desired.)

# Introduction To Git
*(and some GitHub)

## 0. Set up
> git config [options]
> git init
> .gitignore

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files
> git add
> git add -A
> git rm [path]

## 3. Create snapshot
> git commit
> git commit -m "[msg]"


## 4. Explore
> git status
> git log [options]
> git show [sha1]

## (Repeat 1-4 as desired.)

## 5. Add remote
> git remote add [name][url]
> git remote -v

# Introduction To Git
*(and some GitHub)

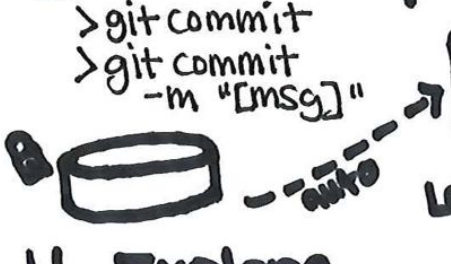
## 0. Set up
> git config [options]
> git init
> .gitignore

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files
> git add
> git add -A
> git rm [path]

## 3. Create snapshot
> git commit
> git commit -m "[msg]"




## 5. Add remote
> git remote add [name][url]
> git remote -v


## 4. Explore
> git status
> git log [options]
> git show [sha1]

## (Repeat 1-4 as desired.)

# Introduction To Git
*(and some GitHub)

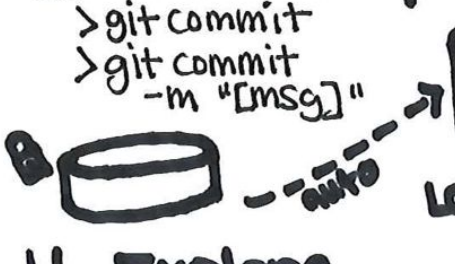
## 0. Set up
>git config [options]
>git init
>.gitignore

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files
>git add
>git add -A
>git rm [path]

## 3. Create snapshot
>git commit
>git commit -m "[msg]"




## 5. Add remote
>git remote add [name][url]
>git remote -v


## 4. Explore
>git status
>git log [options]
>git show [sha1]

## 6. Pull from remote
>git fetch [remote][branch]
>git pull [remote][branch]

## (Repeat 1-4 as desired.)

# Introduction To Git
*(and some GitHub)

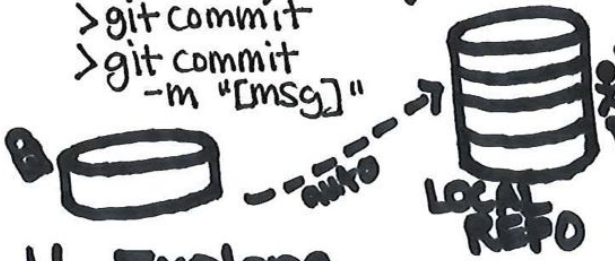
## 0. Set up
>git config [options]
>git init
>.gitignore

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files
>git add
>git add -A
>git rm [path]

## 3. Create snapshot
>git commit
>git commit -m "[msg]"




## 5. Add remote
>git remote add [name][url]
>git remote -v


## 4. Explore
>git status
>git log [options]
>git show [sha1]

## (Repeat 1-4 as desired.)

## 6. Pull from remote
>git fetch [remote][branch]
>git pull [remote][branch]

## 7. Push to remote
>git push [remote][branch]

# Introduction To Git
*(and some GitHub)

## 0. Set up
> git config [options]
> git init
> .gitignore

## 1. Make changes

(use your preferred editor and tools.)

## 2. Stage changed files
> git add
> git add -A
> git rm [path]

## 3. Create snapshot
> git commit
> git commit -m "[msg]"


## 4. Explore
> git status
> git log [options]
> git show [sha1]

## (Repeat 1-4 as desired.)

## 5. Add remote
> git remote add [name] [url]
> git remote -v

## 6. Pull from remote
> git fetch [remote] [branch]
> git pull [remote] [branch]

## 7. Push to remote
> git push [remote] [branch]

## BONUS: Conflicts
TIP: pull before commit to minimize conflicts!
> git merge

# Questions on version control?

(Note that some topics will be covered in later VCS sessions)

UNIVERSITY of WASHINGTON

# Submitting homework via Github Classroom

Bernease Herman[1,2], Joseph Hellerstein[1,2]
[1]eScience Institute
[2]Computer Science

April 7, 2020

# We'll need your GitHub login.

(Our TA, Sam, will send around a survey to collect these later.)

https://classroom.github.com