Git/GitHub Guide

Biostatistics for Social Impact

Adam Peterson and Jonathan Skaza

University of Michigan

Git vs. GitHub

Git: version control system

- tracks changes to content
- provides mechanisms for sharing with collaborators

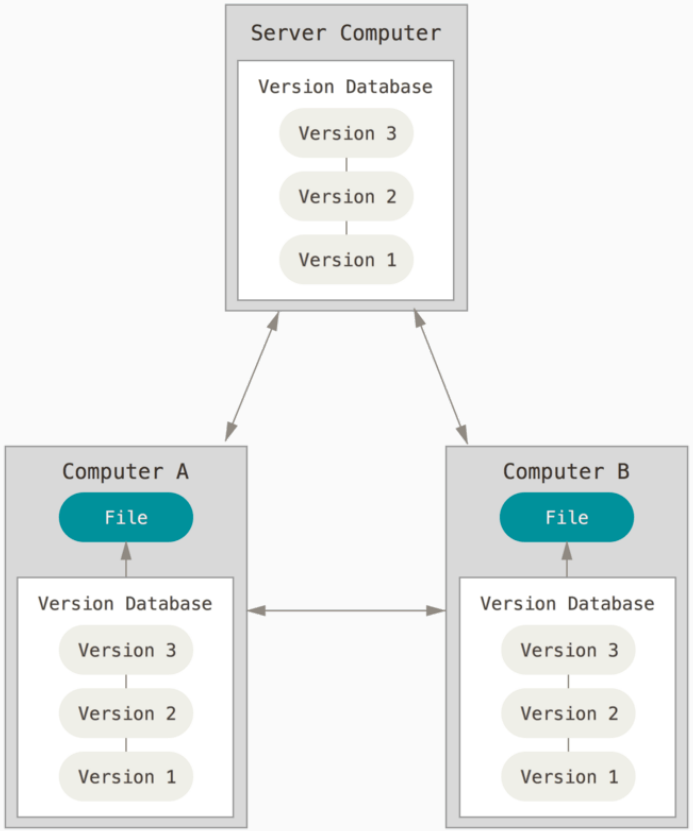
GitHub: company that provides Git repository hosting

- provides additional value to Git users
- user-friendly interface

GitHub is dependent on Git, but not vice versa!

Advantages of Git and GitHub

Version control, Merge collaborators' changes, Open source



Getting Started

1. Create GitHub account
2. Download and install Git
3. Set up Git with username and email
4. Authenticate with GitHub from Git

The process is explained well on GitHub

Basic Usage

Add

```
$ git add modified_file.R
```

```
$ git add -A  # stages All
```

```
$ git add . # stages new and modified, without deleted
```

```
$ git add -u # stages modified and deleted, without new
```

Commit

```
$ git commit -m "type a message about changes"
```

Push

```
$ git push
```

Pull

```
$ git pull
```

Creating Repos and Contributing Code

From Scratch

1. Create a new directory
2. `cd` into the new directory
3. Type `git init`
4. Add code
5. `git add`
6. `git commit`

From an Existing Project

1. `cd` into the project directory
2. Type `git init`
3. `git add`
4. `git commit`

.gitignore

- Files which you do not want to track in Git should be indicated in a `.gitignore` file
- May have a `.gitignore` file in each subdirectory or a global `.gitignore` file

```
.Rhistory
.RData
```

Branching and Merging

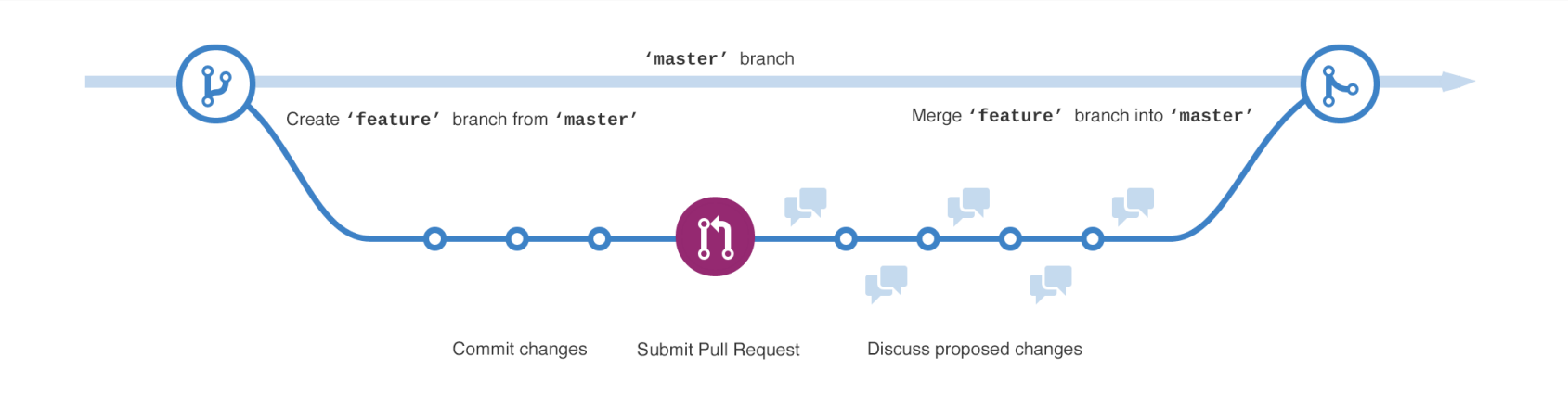
- Once you've made a commit, you can always roll back to it
- However, to really explore a new feature, you might want to "branch" your project
- `cd` into your GitHub directory

```
$ git branch new_feature
```

```
$ git checkout new_feature
```

Checking out, merging a branch

- "Checking out" a branch means that any code you commit will now be placed into this path *separate* from your other work



- Eventually, you'll likely want to re-merge this back with your other work

```
$ git checkout master
```

```
$ git merge new_feature
```

Best Practices for the B4SI Org Site

- Separate repository for each individual; make your own
- Keep your Master branch completely functional; this code always works
- Work on your new functions, objects, etc. in a different branch
- Merge new branches when they pass their unit tests
- Commit whenever you've written code you don't want to lose

Rules for Reproducibility

- Organize Data & Code
- Code Everything
- Use Relative Paths
- Automate Your Pipeline
- Use Functions to Reduce Repetition
- Use Version Control

Adapted from Broman (2016)

Additional Resources

- Karl Broman's Git/GitHub guide
- "Try Git" online tutorial
- For more, see "Learn git in 20 minutes"