Version Control and Reproducible Research with GitHub

Tad Dallas

December 2013

What is GitHub?

Code sharing, publishing and development service for collaborative projects

Why use it?

- Version control
- Open collaboration with other scientists
- Creepily watch what other people are working on!

What is GitHub?

Code sharing, publishing and development service for collaborative projects

Why use it?

- Version control
- Open collaboration with other scientists
- Creepily watch what other people are working on!

Under the hood

- Git is the version control language that GitHub is the GUI for
- Created by Linus Torvalds, a central developer of the Linux OS
- Command-line, but really straight-forward

...but why can't I just use Dropbox?

Git is different

- Actual version control
- Forking
- Open collaboration, open science
- No limit

...but why can't I just use Subversion?

Git is not that different

- A bit more popular (if that means anything)
- Git can talk to Subversion though

A couple quick definitions

- **Repository**: Storage space where your projects reside

A couple quick definitions

- **Repository**: Storage space where your projects reside
- **Commit**: Takes 'snapshot' of your repository, so you can log a new change, or revert to a previous state (Common command)

A couple quick definitions

- **Repository**: Storage space where your projects reside
- **Commit**: Takes 'snapshot' of your repository, so you can log a new change, or revert to a previous state (Common command)
- **Branch**: Think of a folder within a repository, but cooler. More on this later

A couple quick definitions

- **Repository**: Storage space where your projects reside
- **Commit**: Takes 'snapshot' of your repository, so you can log a new change, or revert to a previous state (Common command)
- **Branch**: Think of a folder within a repository, but cooler. More on this later
- **Fork**: What it sounds like. You're taking someone's project and making a copy of it for your own use (either to collaborate and to merge later or to use as a template for a different project)

A couple quick definitions

- **Repository**: Storage space where your projects reside
- **Commit**: Takes 'snapshot' of your repository, so you can log a new change, or revert to a previous state (Common command)
- **Branch**: Think of a folder within a repository, but cooler. More on this later
- **Fork**: What it sounds like. You're taking someone's project and making a copy of it for your own use (either to collaborate and to merge later or to use as a template for a different project)
- **Push**: The act of updating your project files (you will "push" your **commits**)

A couple quick definitions

- **Repository**: Storage space where your projects reside
- **Commit**: Takes 'snapshot' of your repository, so you can log a new change, or revert to a previous state (Common command)
- **Branch**: Think of a folder within a repository, but cooler. More on this later
- **Fork**: What it sounds like. You're taking someone's project and making a copy of it for your own use (either to collaborate and to merge later or to use as a template for a different project)
- **Push**: The act of updating your project files (you will "push" your **commits**)
- **Pull**: Gets commits from a repository to your machine

A couple quick definitions

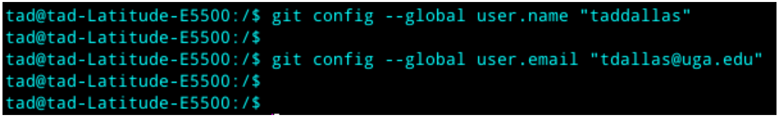
- **Repository**: Storage space where your projects reside
- **Commit**: Takes 'snapshot' of your repository, so you can log a new change, or revert to a previous state (Common command)
- **Branch**: Think of a folder within a repository, but cooler. More on this later
- **Fork**: What it sounds like. You're taking someone's project and making a copy of it for your own use (either to collaborate and to merge later or to use as a template for a different project)
- **Push**: The act of updating your project files (you will "push" your **commits**)
- **Pull**: Gets commits from a repository to your machine
- **Fetch**: A better version of **pull** that doesn't merge **commits**

How to begin

1. Set up an account (go to https://github.com/) and download Git (http://git-scm.com/downloads)
2. Open a terminal window

How to begin

1. Set up an account (go to https://github.com/) and download Git (http://git-scm.com/downloads)
2. Open a terminal window



```
tad@tad-Latitude-E5500:/$ git config --global user.name "taddallas"
tad@tad-Latitude-E5500:/$
tad@tad-Latitude-E5500:/$ git config --global user.email "tdallas@uga.edu"
tad@tad-Latitude-E5500:/$
tad@tad-Latitude-E5500:/$
```

Okay. Now we have Git on our machines and GitHub accounts.

The GitHub framework

Getting your files on GitHub

- Do work in a local directory
- Create a Git repo in this local directory
- **Add** and **commit** your files ('put stones in the catapult')
- **Push** your files to Github ('catapult those stones')

Make your directory

Sets up local directory

> *$ cd folder you want your directory in*
> *$ mkdir directory name*
> *$ cd your project directory*

Initializes git in that directory

> *$ git init*

Do some stuff in the directory!

Committing changes

From within your local directory

```
$ git remote add origin
https://github.com/yourname/yourproject.git
$ git commit -a -m "message associated with your commit"
```

Committing changes

From within your local directory

> *$ git remote add origin https://github.com/yourname/yourproject.git*
> *$ git commit -a -m “message associated with your commit”*

Only need to do the **git remote add** command once. You can check to see what your remote locations are by typing

> *git remote -v*

from within your local directory.

Push it!

```
$ git push origin master
```

General framework for edits thereafter

1. Edit your files locally
2. $ git commit -a -m “message about this commit”
3. $ git push origin master

How to collaborate using GitHub

Up to this point, it's been a solitary experience of **making** and **pushing**

Methods of collaboration

Two ways:

- **'Fork and Pull' model** : better
- **'Shared Repository' model** : easier

How to collaborate using GitHub

Up to this point, it's been a solitary experience of **making** and **pushing**

Methods of collaboration

Two ways:

- **'Fork and Pull' model** : better
- **'Shared Repository' model** : easier

Pulling files from GitHub

- cd to local repository
- $ git remote -v   outputs the .git repos you can push/pull to/from. Use $ git remote add 'http://github.com/name/project.git' if necessary
- $ git pull .   fetches and merges files

Cloning...like forking, but sneakier

$ git clone git://github.com/somename/someproject.git someproject

#This initializes a new local directory on your machine in a folder called 'someproject'

Some useful commands

Check status :

```
$ git status
```

Some useful commands

Check status :

```
$ git status
```

See your remote locations :

```
$ git remote -v
```

Some useful commands

Check status :

```
$ git status
```

See your remote locations :

```
$ git remote -v
```

View commit history :

```
$ git log
```

Some useful commands

Check status :

```
$ git status
```

See your remote locations :

```
$ git remote -v
```

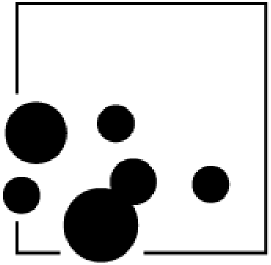
View commit history :

```
$ git log
```

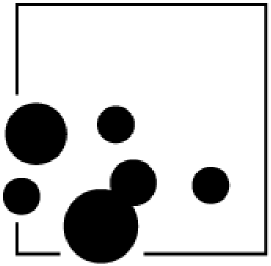
Revert to previous version since last commit:

```
$ git checkout – filename
```

Questions?



Questions?



Now let's look at the user interface of GitHub and play around a bit