

# Version control with git for scientists

<https://github.com/mbjoseph/git-intro>

Max Joseph

July 27, 2016

# Discuss

What is your current version control system?

1. How do you manage different file versions?
2. How do you work with collaborators on the same files?
3. How much would your science suffer if your workstation exploded right now? (scale from 1-10)

# What is git

Version control system

- ▶ manage different versions of files
- ▶ collaborate with yourself
- ▶ collaborate with other people

# Why use git

“Always remember your first collaborator is your future self, and your past self doesn’t answer emails”

- ▶ Christie Bahlai

# What is git good for?

- ▶ backup
- ▶ reproducibility
- ▶ collaboration

# What you get

Tour of a git repository

# Overview

1. Git on the command line
2. Git in RStudio
3. Github vs. GitLab vs. Bitbucket for remote mirroring

# Command line git

Make a directory with a file

```
mkdir test  
cd test  
nano sim.R
```

Then write a short simulation, e.g.

```
x <- rnorm(10)  
save(x, file = "x.RData")
```



# Initializing a repository

Prerequisites:

- ▶ git installed (check with `which git`)
- ▶ git configured (check with `git config --list`)

```
git config --global user.name "Vlad Dracula"  
git config --global user.email "vlad@tran.sylvan.ia"  
git config --global color.ui "auto"  
git config --global core.editor "nano"
```

# Initializing a repository

```
git init
```

Notice the `.git/` directory

# Checking repository status

```
git status
```

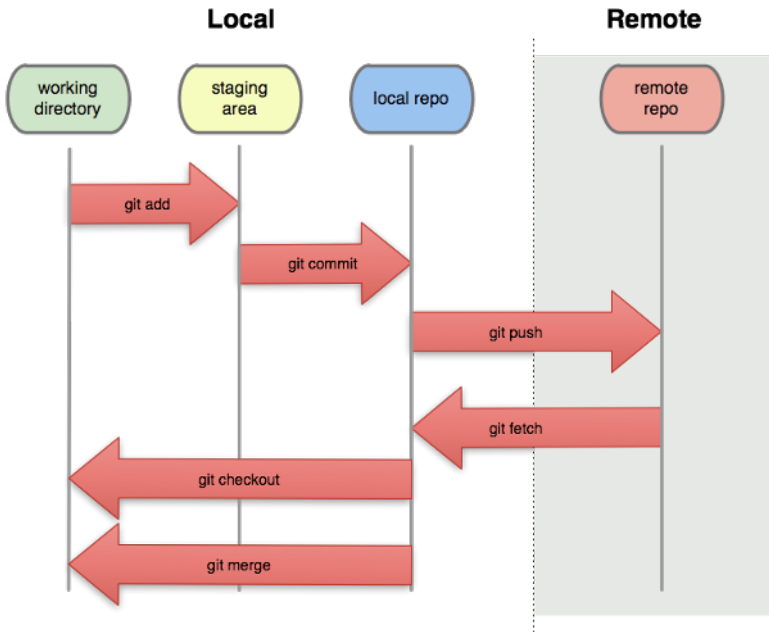
## Adding your file

```
git add your_filename.R
```

or, to add everything

```
git add --all
```

# Your changes are now staged



# Committing

Changes aren't final until they're committed

```
git status
```

# Committing

Once you're sure that you're changes are worth saving  
(THIS WILL GO ON YOUR PERMANENT RECORD)

```
git commit -m 'changed x, y, and z'
```

## Commit messages

- ▶ Describe why and the what “in a nutshell”
- ▶ Note to your future self (and to anyone else who you’re collaborating with)



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.



# What did we do?

```
git status
```

```
git log
```

## Make another change

1. Change file
2. Add changes
3. Commit changes
4. View updated log

## Now, do something really stupid

“Accidentally” introduce some errors to your file

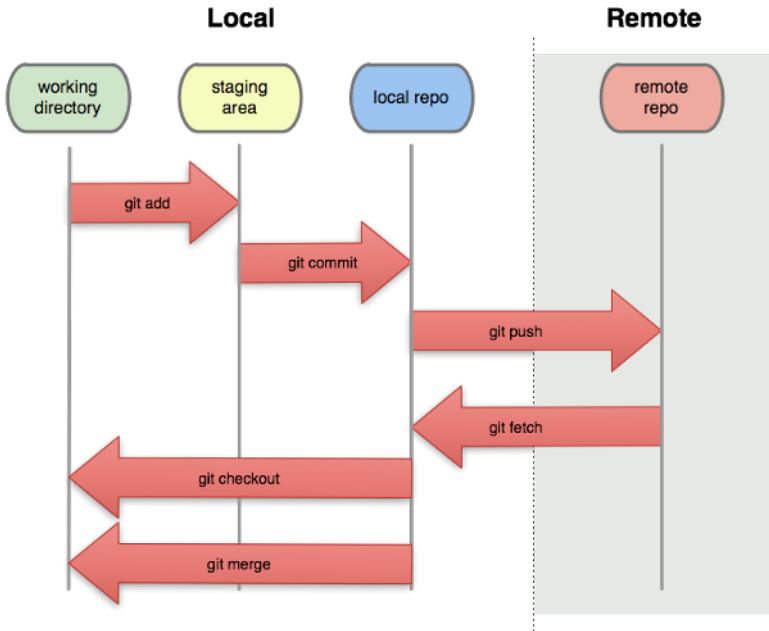
# Woops

Not that this ever happens...

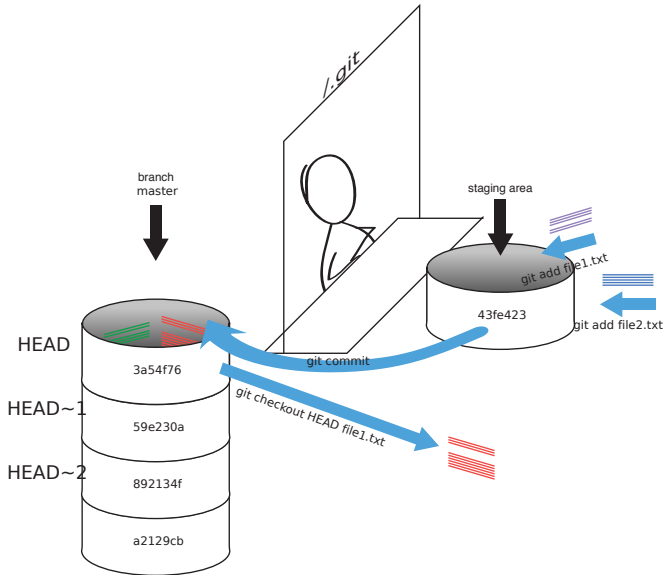
```
git diff
```

```
git checkout HEAD your_file.R
```

# What happened?



# Wait, what does HEAD refer to?



**Figure 4:** Commits  $\approx$  a stack of heads

# What if you really screw up?

**A git choose your own adventure**

<http://sethrobertson.github.io/GitFixUm/fixup.html>

# Mirroring your repository on the internet

## Setting up a “remote”

1. Create repository on Github with no .gitignore, no README, and no license
2. Add that as a remote

```
git remote add origin https://www.github.com/user/test.git
```

How to check: `git remote -v`



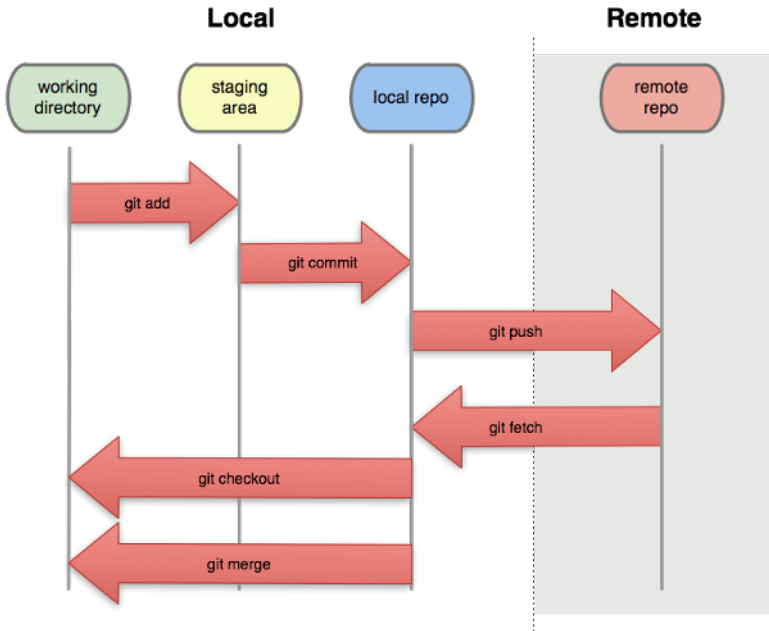
## Once your repository has been linked to remote

Push your changes

```
git push -u origin master
```

Check the remote (Github or Bitbucket) to see new changes

# Overview



# RStudio's interface

Start a new project

Check it out from your remote git repository using ssh or html  
(ssh is better, html may require additional config w/ RStudio)

# Demo of adding, committing, pushing

# Github vs. GitLab vs. Bitbucket

Private repos:

- ▶ free on Bitbucket (w/  $< 6$  collaborators)
- ▶ free on GitLab (unlimited collaborators)
- ▶ not free on Github


# Github vs. GitLab vs. Bitbucket

- ▶ all very similar
- ▶ Popularity & user base (4 vs. ?? vs. 1 million)
- ▶ free vs. pay
- ▶ open source vs. closed source

**You can use all three if you want!**

# Continuing education & advanced stuff

## Pro Git

 **git** --everything-is-local

Search entire site...

**About**  
**Documentation**  
Reference  
Book  
Videos  
External Links  
**Blog**  
**Downloads**  
**Community**

Download this book in [PDF](#), [mobi](#), or [ePub](#) form for free.

This book is translated into [Deutsch](#), [简体中文](#), [正體中文](#), [Français](#), [日本語](#), [Nederlands](#), [Русский](#), [한국어](#), [Português \(Brasil\)](#) and [Čeština](#).


Partial translations available in [Arabic](#), [Español](#), [Indonesian](#).

## Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#). Print versions of the book are available on [Amazon.com](#).

### 1. Getting Started

- 1.1 About Version Control
- 1.2 A Short History of Git
- 1.3 Git Basics
- 1.4 The Command Line
- 1.5 Installing Git
- 1.6 First-Time Git Setup
- 1.7 Getting Help
- 1.8 Summary



2nd Edition (2014)  
[Switch to 1st Edition](#)

**Download Ebook**





-  pdf
-  epub
-  mobi
-  html

Figure 6: