

Git tooty  
(fart joke?)

Silas Tittes

16 September, 2016

# Outline

- ▶ I don't git git, why should I do this?

# Outline

- ▶ I don't git git, why should I do this?
- ▶ Working with a remote repository – GitHub

# Outline

- ▶ I don't `git git`, why should I do this?
- ▶ Working with a remote repository – GitHub
- ▶ Working with a local repository – `git`

# Outline

- ▶ I don't `git git`, why should I do this?
- ▶ Working with a remote repository – GitHub
- ▶ Working with a local repository – `git`
- ▶ Version control *basics* – reviewing file histories

# Outline

- ▶ I don't `git git`, why should I do this?
- ▶ Working with a remote repository – GitHub
- ▶ Working with a local repository – `git`
- ▶ Version control *basics* – reviewing file histories
- ▶ Where to go from here

“I don't git it, why should I do this?”

- ▶ Version control – because you *will* screw something up.



# “I don't git it, why should I do this?”

- ▶ Version control – because you *will* screw something up.
- ▶ Version control – because uncontrolled version proliferation is *the devil*.





# “I don't git it, why should I do this?”

- ▶ Version control – because you *will* screw something up.
- ▶ Version control – because uncontrolled version proliferation is *the devil*.
- ▶ Collaboration – because *everything* is better together.



# “I don't git it, why should I do this?”

- ▶ Version control – because you *will* screw something up.
- ▶ Version control – because uncontrolled version proliferation is *the devil*.
- ▶ Collaboration – because *everything* is better together.
- ▶ Remote backup – because your computer *will* die.



# “I don't git it, why should I do this?”

- ▶ Version control – because you *will* screw something up.
- ▶ Version control – because uncontrolled version proliferation is *the devil*.
- ▶ Collaboration – because *everything* is better together.
- ▶ Remote backup – because your computer *will* die.
- ▶ Public portfolio – because you *should* demonstrate your awesomeness to the world.



# “I don't git it, why should I do this?”

- ▶ Version control – because you *will* screw something up.
- ▶ Version control – because uncontrolled version proliferation is *the devil*.
- ▶ Collaboration – because *everything* is better together.
- ▶ Remote backup – because your computer *will* die.
- ▶ Public portfolio – because you *should* demonstrate your awesomeness to the world.
- ▶ Encourages reproducibility *and* creativity.



## Things I use git and GitHub for:

- ▶ My blog – [silas.tittes.github.io](https://silas.tittes.github.io)

## Things I use git and GitHub for:

- ▶ My blog – [silas.tittes.github.io](http://silas.tittes.github.io)
- ▶ All the code I care about (some private on BitBucket)

## Things I use git and GitHub for:

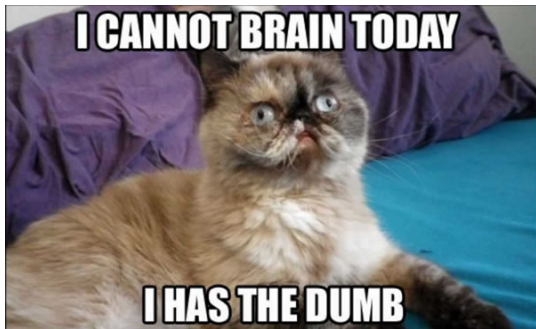
- ▶ My blog – [silas.tittes.github.io](http://silas.tittes.github.io)
- ▶ All the code I care about (some private on BitBucket)
- ▶ Teaching materials and teaching diary (not private)

## Things I use git and GitHub for:

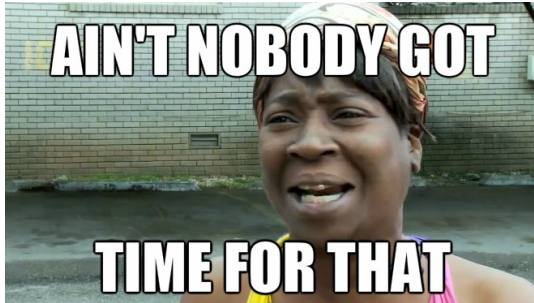
- ▶ My blog – [silas.tittes.github.io](http://silas.tittes.github.io)
- ▶ All the code I care about (some private on BitBucket)
- ▶ Teaching materials and teaching diary (not private)
- ▶ This presentation



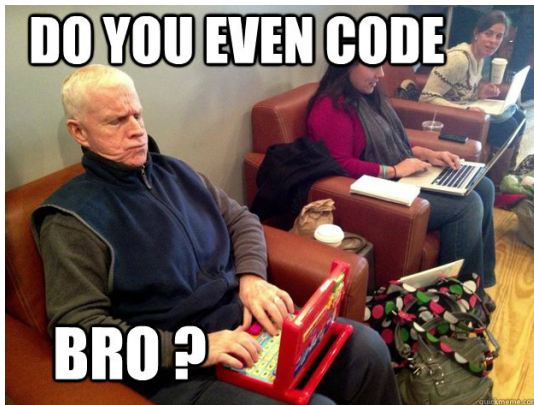
“What if people look at my code and think I’m dumb?”



“What if people look at my code and think I’m dumb?”



“I’m not really a coder”



# You are, but even if you're "not"

"GitHub is used to manage the collaborative development of recipes, musical scores, books, fonts, legal documents, lessons and tutorials, and data sets"

## GitHub for the rest of us



*Credit: Flickr/Nick Guaranto*

Git made it possible for programmers to coordinate distributed work across teams -- now GitHub makes it possible for everyone else



By **Jon Udell** | Follow  
InfoWorld | Feb 24, 2015

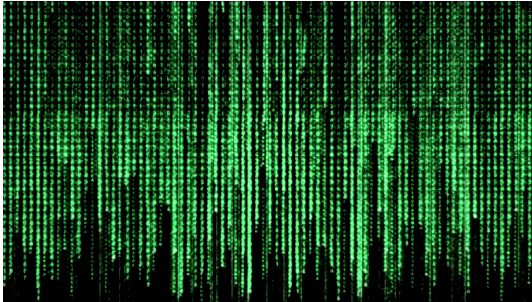
Working with a remote repository – GitHub (or  
BitBucket)

Let's make a GitHub log-in and repository!



Working with a local repository – git

Enter the matrix





## Clone your new remote repo

```
git clone https://github.com/octocat/lovecatz.git
```

Now make some changes to your README.md on your computer, or create a new file. Perhaps `hello.cpp` if you're feeling real crazy.

## Git used to it – the basics

You've made some local repo edits to some code, documentation, draft manuscript, data, or cookie recipe. What now?

*#See what's changed*

`git status`

*#prepare the content for the next commit -- "staging"*

`git add --all` *# --all, or specify files*

*#"The Rub" -- Record changes to the repository*

*#This is what creates version control*

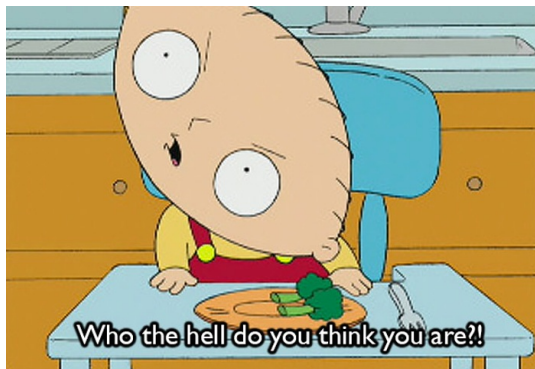
`git commit -m "short discription of code changes"`

*#sync local and remote repos on master branch*

`git push -u origin master`

# Error!

```
git config --global user.name "Octocat69"  
git config --global user.email "Octocat69@bbqchikn.com"
```



## Back on track

*#sync local and remote repos*

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

*#Up to date?*

```
git status
```

Check out remote GitHub repo to see updates.

## Practice the “add, commit, push” cycle 4X



add, commit, push

add, commit, push

add, commit, push

add, commit, push

## Version control basics – reviewing file histories

*#git overview of commits to repo*

`git log` *#for whole repo*

*#git overview of commits for one file*

`git log <filename>`

*#review int-th previous version of file*

`git show HEAD~<int>:<filename>`

Explore previous commits, or testing ideas your unsure about – branch first!

*#create a new branch*

```
git checkout -b branch_name
```

*#short term cache of current repo*

```
git stash
```

*#check which branch you're on*

```
git branch #prints asterisk next to current branch
```

*#go back to old commit by hash value*

*#see git log for hash values*

```
git checkout <commit number>
```

# What now?

1. If you were just curious, but like the most recent commits:

```
#go back to most recent commit  
git stash apply
```



# What now?

2. If you screwed up bad and need to go back to previous version  
check out oh shit, git!

```
#revert to int-th previous commit  
git reset --hard HEAD~<int>  
#add commit to remote repo  
git push origin master
```

# What now?

## 3. Or remove branch entirely

```
#remove branch name -- caution  
git branch -d branch_name
```

# What now?

## 4. Push your new branch to remote repo

```
git add --all # --all, or specify files
git commit -m "short discription of code changes"
git push -u origin branch_name
```

# What now?

## 5. Merge with master branch,

```
#go back to master branch
```

```
git checkout master
```

```
#merge branch with master branch
```

```
git merge branch_name #read messages carefully!
```

## What now?

6. Not your repo? Forking and submitting pull requests. But – too much too soon.

# Conclusions

- ▶ `git` and GitHub are valuable modern tools even for non-programmers – learn how to use them!

# Conclusions

- ▶ `git` and GitHub are valuable modern tools even for non-programmers – learn how to use them!
- ▶ You can now make a solo repo and implement add, commit, push cycles for version control and remote storage.

# Conclusions

- ▶ `git` and GitHub are valuable modern tools even for non-programmers – learn how to use them!
- ▶ You can now make a solo repo and implement add, commit, push cycles for version control and remote storage.
- ▶ Hopefully you have a sense of the more advanced version control options available to you, even if you aren't comfortable using them yet.



## Where to go from here

practice lots, read documentation, practice more

Octocat is watching



git silly