



**BGGN 213**

**Hands-on Lab Session**

**Class 07**

**Barry Grant**

**UC San Diego**

<http://thegrantlab.org/bggn213>

Last class!

function()

# Last class revisited...

- Write a function `grade()` to determine an overall grade from a vector of student homework assignment scores dropping the lowest single assignment score.

```
# student 1  
c(100, 100, 100, 100, 100, 100, 100, 90)  
  
# student 2  
c(100, NA, 90, 90, 90, 90, 97, 80)  
  
# now grade all students in an example class  
url <- "https://tinyurl.com/gradeinput"
```

# Last class revisited...

- Write a function `grade()` to determine an overall grade from a vector of student homework assignment scores dropping the lowest single assignment score.

```
grade <- function(x) {  
  x <- as.numeric(x)  
  x[ is.na(x) ] = 0  
  mean(x[ -which.min(x) ] )  
}
```

# Last class revisited...


- Write a function `grade()` to determine an overall grade from a vector of student homework assignment scores **dropping the lowest single assignment score**.

```
grade <- function(x) {  
  x <- as.numeric(x)  
  x[ is.na(x) ] = 0  
  mean(x[ -which.min(x) ] )  
}
```

# Last class revisited...

- Write a function `grade()` to determine an overall grade from a vector of student homework assignment scores **dropping the lowest single assignment score**.

```
grade <- function(x) {  
  x <- as.numeric(x)  
  x[ is.na(x) ] = 0  
  mean(x[ -which.min(x) ] )  
}
```



# Do this now...

- Write a function **grade2()** to determine an overall grade from a vector of student homework assignment scores **OPTIONALLY** dropping the lowest single assignment score.

```
grade <- function(x) {  
  x <- as.numeric(x)  
  x[ is.na(x) ] = 0  
  mean(x[ -which.min(x) ] )  
}
```

# Last class revisited...

```
grade2 <- function(x, drop.lowest=TRUE) {  
  x <- as.numeric(x)  
  x[ is.na(x) ] = 0  
  if(drop.lowest) {  
    mean(x[ -which.min(x) ])  
  } else {  
    mean(x)  
  }  
}
```



And some function homework.....

**Complete Q6. In last days lab supplement**

Your Homework!

```
library(bio3d)
s1 <- read.pdb("4AKE") # kinase with drug
s2 <- read.pdb("1AKE") # kinase no drug
s3 <- read.pdb("1E4Y") # kinase with drug

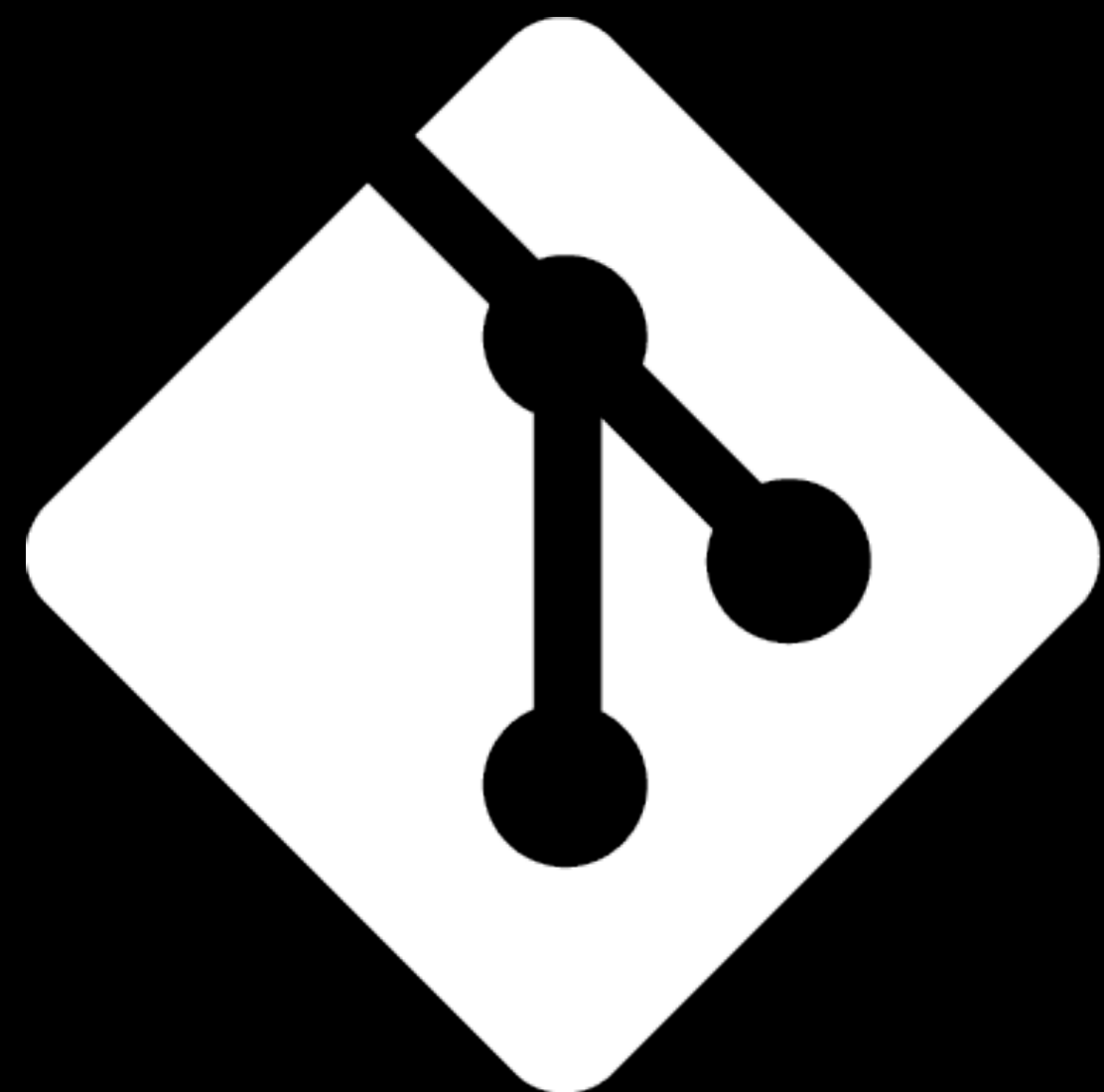
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")

s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b

plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
plotb3(s2.b, sse=s2.chainA, typ="l", ylab="Bfactor")
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor")
```

# Suggested steps for writing your functions

1. Start with a simple problem and get a working snippet of code
2. Rewrite to use temporary variables (e.g. x, y, df, m etc.)
3. Rewrite for clarity and to reduce calculation duplication
4. Turn into an initial function with clear useful names
5. Test on small well defined input and (subsets of) real input
6. Report on potential problem by failing early and loudly!
7. Refine and polish



**git**

# What is Git?

(1) An unpleasant or contemptible person. Often incompetent, annoying, senile, elderly or childish in character.



(2) A modern distributed version control system with an emphasis on speed and data integrity.



# What is Git?

(1) An unpleasant or contemptible person. Often incompetent, annoying, senile, elderly or childish in character.



(2) A modern distributed version control system with an emphasis on speed and data integrity.



# Version Control

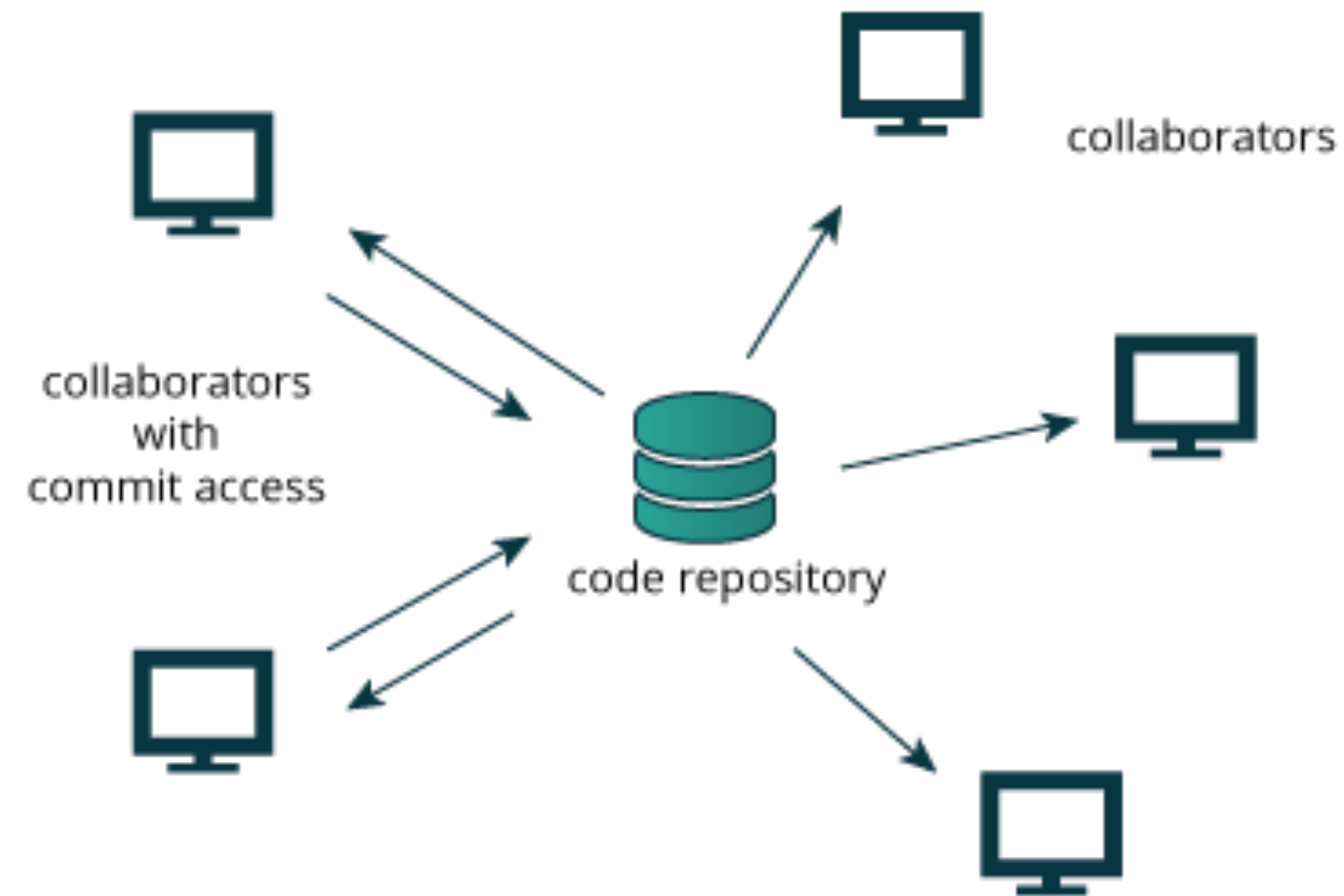
Version control systems (VCS) record changes to a file or set of files over time so that you can recall specific versions later.

Client-server	Free/open-source	CVS (1986, 1990 in C) · CVSNT (1998) · QVCS Enterprise (1998) · Subversion (2000)
	Proprietary	Software Change Manager (1970s) · Panvalet (1970s) · Endeavor (1980s) · Dimensions CM (1980s) · DSEE (1984) · Synergy (1990) · ClearCase (1992) · CMVC (1994) · Visual SourceSafe (1994) · Perforce (1995) · StarTeam (1995) · Integrity (2001) · Surround SCM (2002) · AccuRev SCM (2002) · SourceAnywhere (2003) · Vault (2003) · Team Foundation Server (2005) · Team Concert (2008)
Distributed	Free/open-source	GNU arch (2001) · Darcs (2002) · DCVS (2002) · ArX (2003) · Monotone (2003) · SVK (2003) · Codeville (2005) · Bazaar (2005) · Git (2005) · Mercurial (2005) · Fossil (2007) · Veracity (2010)
	Proprietary	TeamWare (1990s?) · Code Co-op (1997) · BitKeeper (1998) · Plastic SCM (2006)

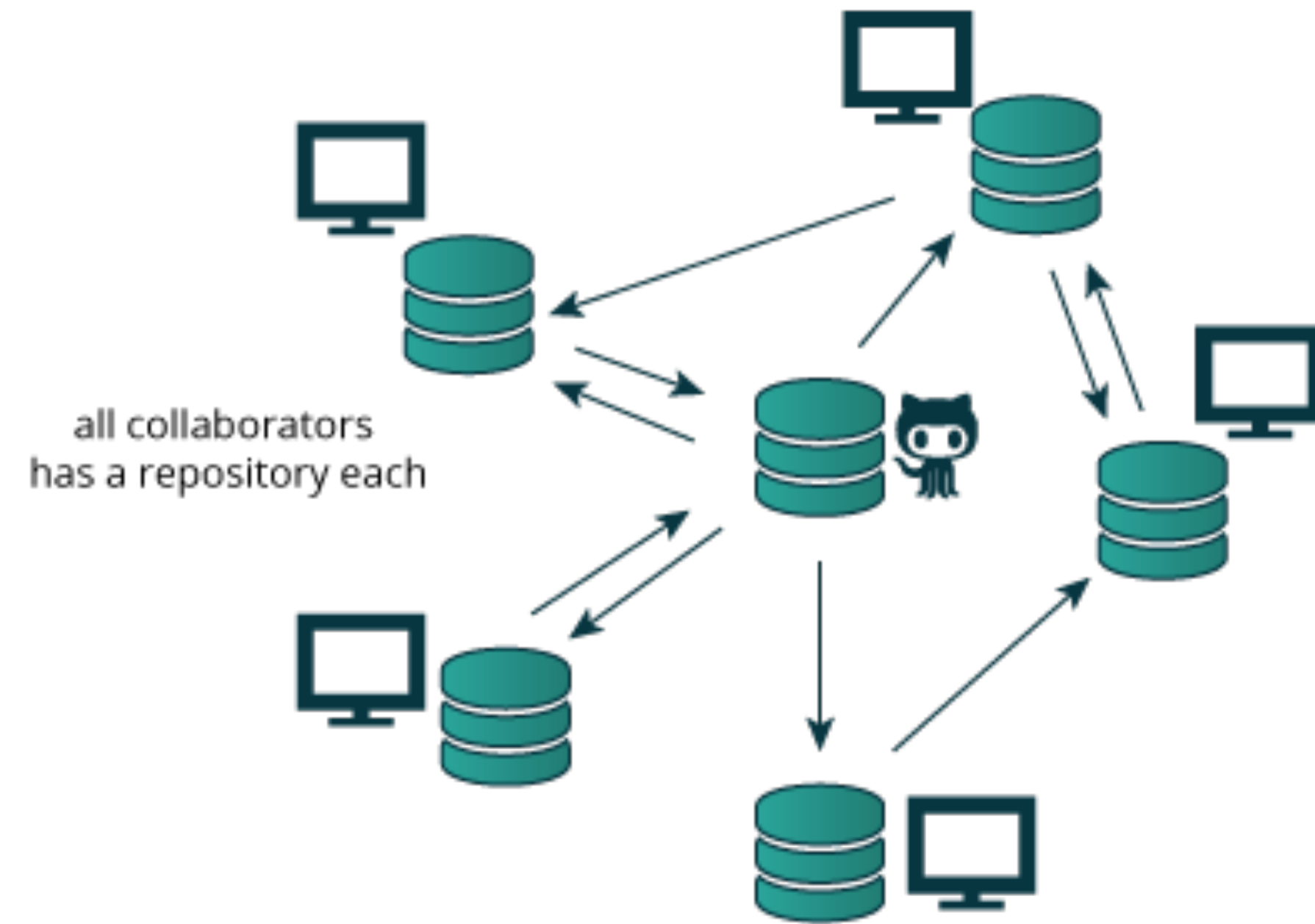
There are many VCS available, see:

[https://en.wikipedia.org/wiki/Revision\\_control](https://en.wikipedia.org/wiki/Revision_control)

# Client-Server vs Distributed VCS



Client-server approach



Distributed approach

**Distributed** version control systems (DCVS) allows multiple people to work on a given project without requiring them to share a common network.



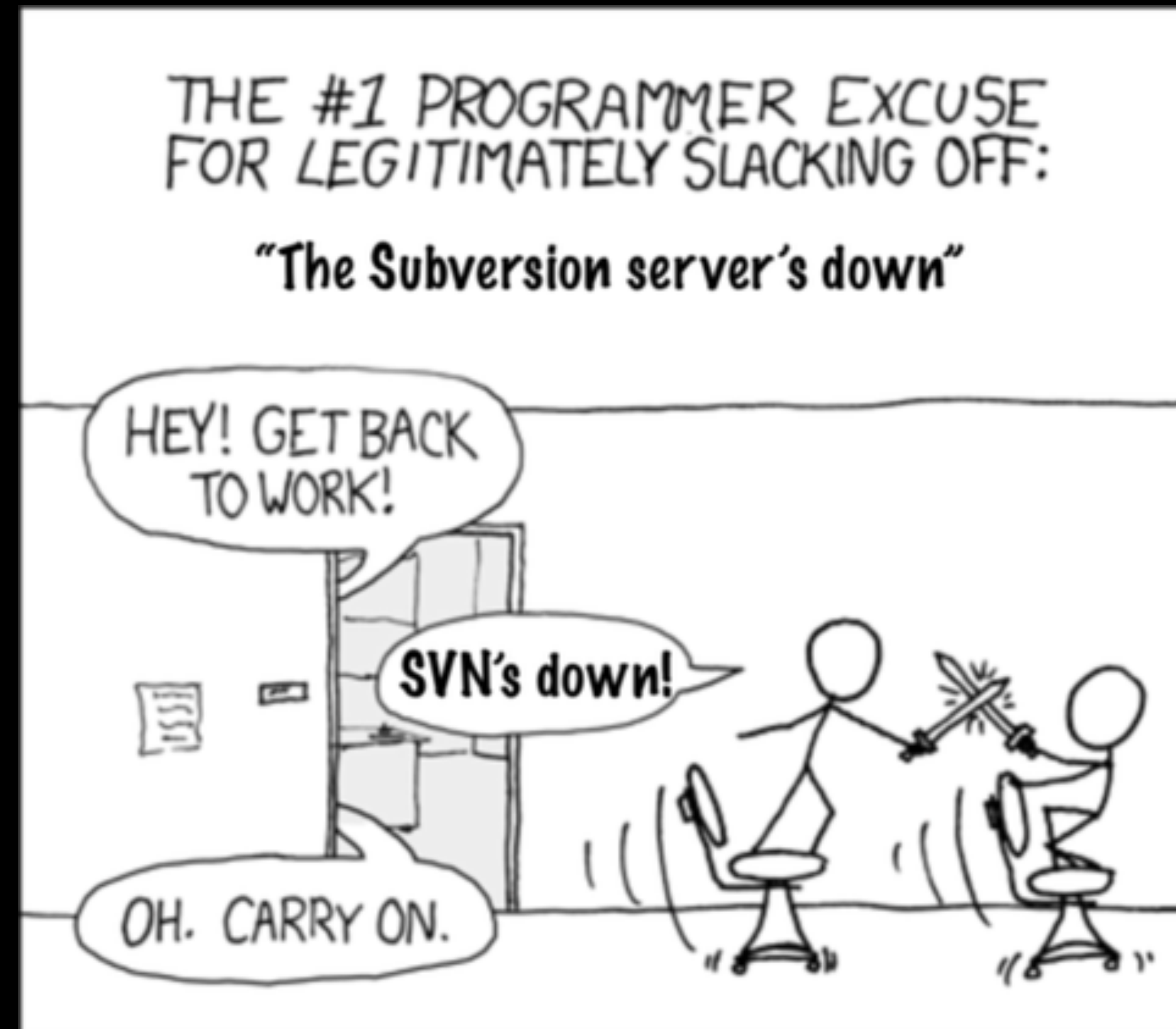
THE #1 PROGRAMMER EXCUSE  
FOR LEGITIMATELY SLACKING OFF:

**"The Subversion server's down"**



<http://tinyurl.com/distributed-advantages>

# Git is now the most popular free VCS!



## Git offers:

- Speed
- Backups
- Off-line access
- Small footprint
- Simplicity\*
- Social coding

<http://tinyurl.com/distributed-advantages>

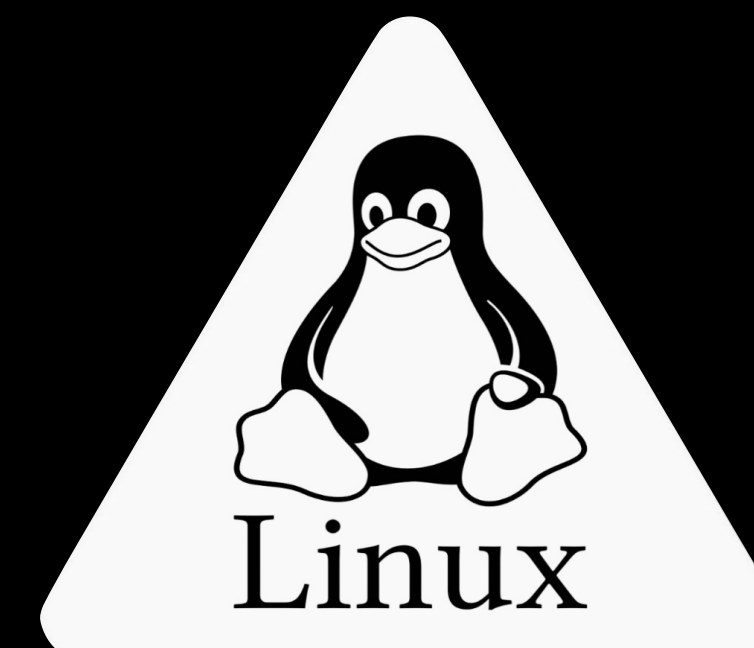
# Where did Git come from?

Written initially by Linus Torvalds to support Linux kernel and OS development.

Meant to be distributed, fast and more natural.

Capable of handling large projects.

Now the most popular free VCS!





**Why use Git?**

Q. Would you write your lab book in pencil, then erase and overwrite it every day with new content?

Q. Would you write your lab book in pencil, then erase and overwrite it every day with new content?

Version control is the lab notebook of the digital world: it's what professionals use to keep track of what they've done and to collaborate with others.

# Why use Git?

- Provides 'snapshots' of your project during development and provides a full record of project history.
- Allows you to easily reproduce and rollback to past versions of analysis and compare differences. (N.B. Helps fix software regression bugs!)
- Keeps track of changes to code you use from others such as fixed bugs & new features
- Provides a mechanism for sharing, updating and collaborating (like a social network)
- Helps keep your work and software organized and available



# Obtaining Git

**Note:** You hopefully already have git installed!  
To check open the “Terminal” tab in RStudio and type:

① `which git`

② `git --version`

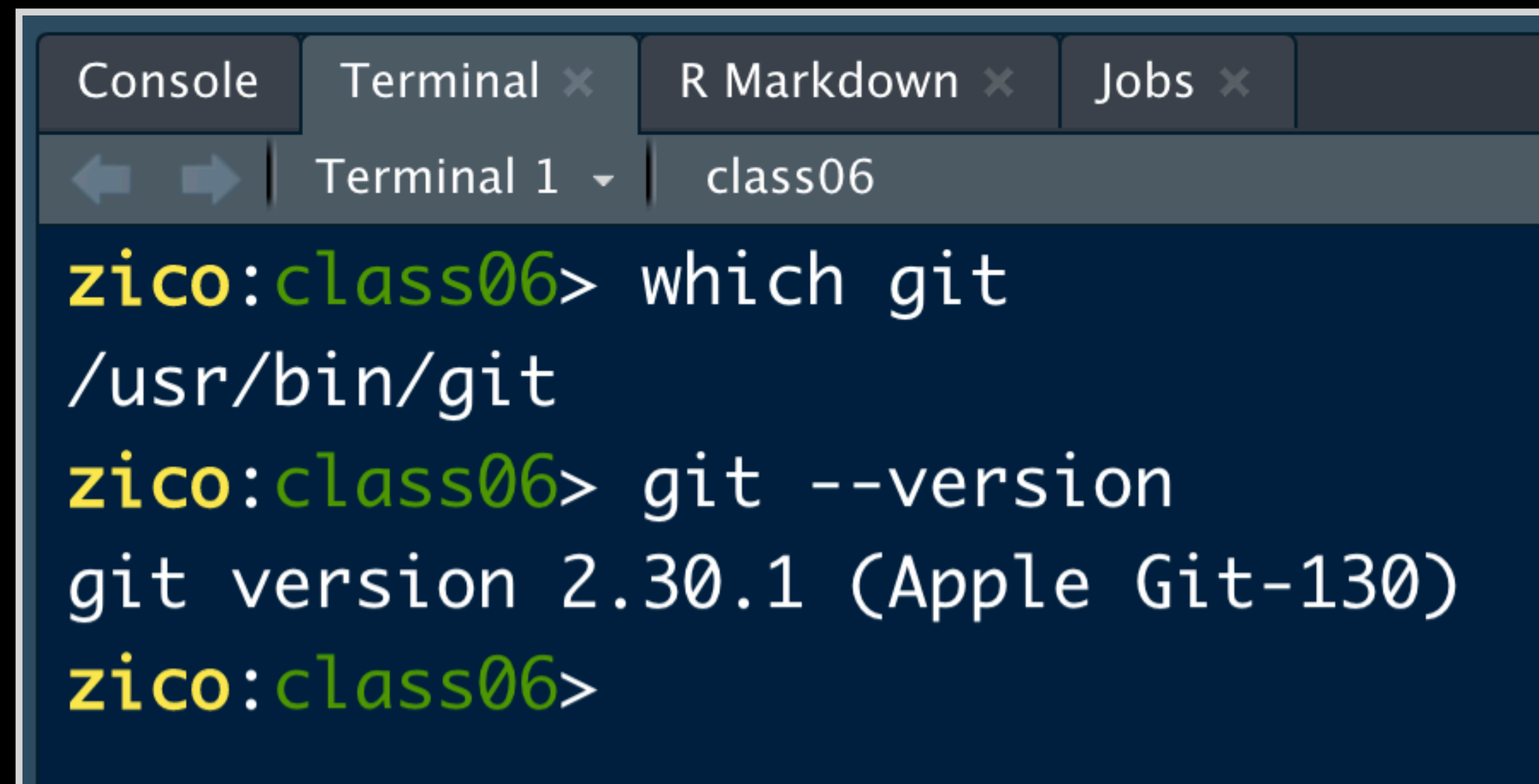
# Obtaining Git

**Note:** You might already have git installed  
To check open the “Terminal” tab in RStudio and type:

① **which git**

② **git --version**

# Obtaining Git



```
Console Terminal x R Markdown x Jobs x  
← → Terminal 1 class06  
zico:class06> which git  
/usr/bin/git  
zico:class06> git --version  
git version 2.30.1 (Apple Git-130)  
zico:class06>
```

**Note:** You might already have git installed  
To check open the “Terminal” tab in RStudio and type:

① **which git**

② **git --version**

# Obtaining Git

## Windows Only (if you have problems)

If the “**which git**” command did not work, try:

**where git**

If this works see next slide. If not then you need to install **GitBash**, instructions here:

Class [Computer Setup Page](#)

## Mac Only (if you have problems)

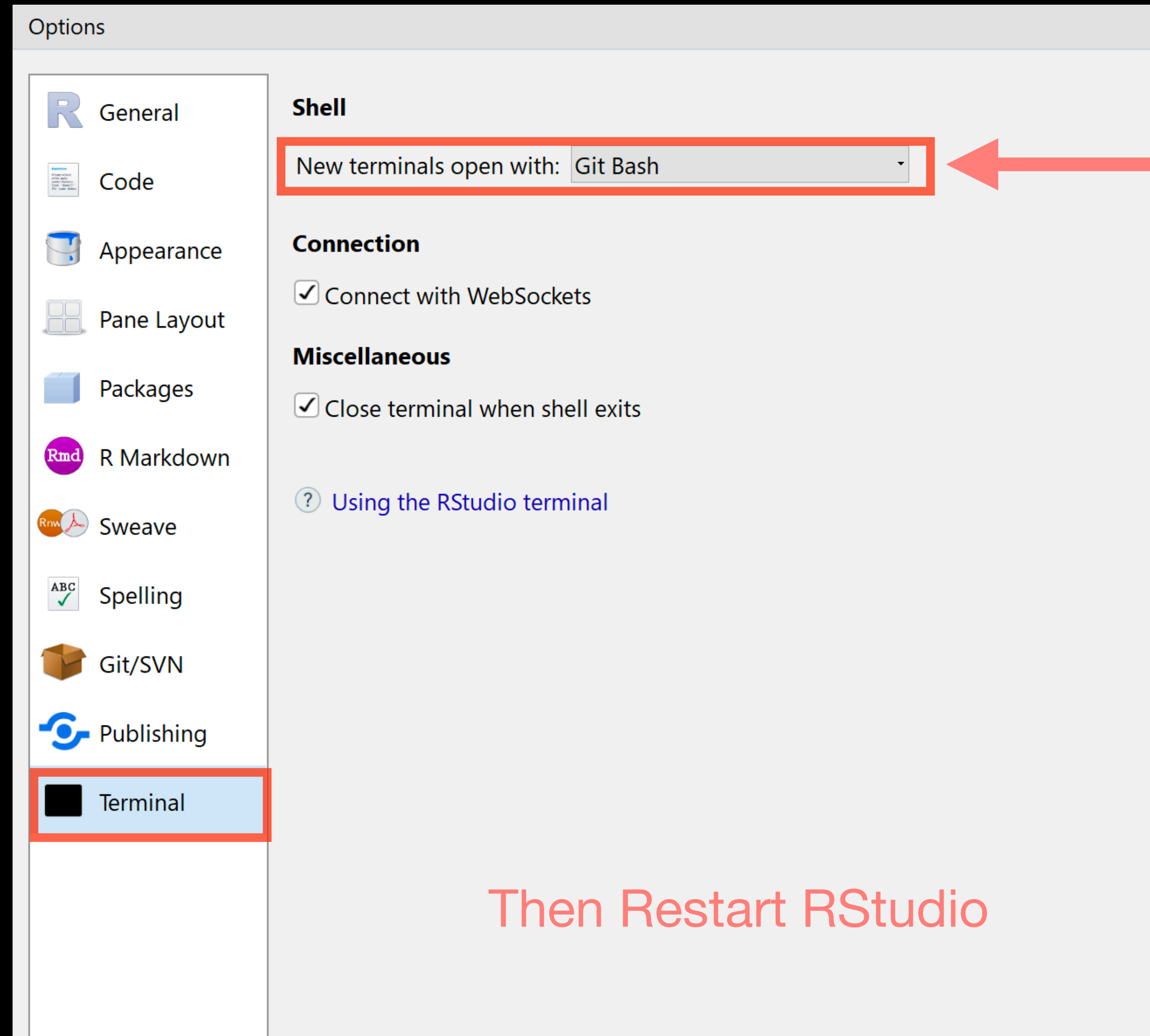
If the “**which git**” command did not work, you may need to install select developer tools.

In your Terminal type:

**xcode-select --install**

# On a PC Only!

**Go to:** RStudio > Tools > Global Options > Terminal



**Make sure  
Git Bash is  
selected!**

Then Restart RStudio

Do it Yourself!

**Note:** You might already have git installed  
To check open the “Terminal” tab in RStudio and type:

- 1 **which git**
- 2 **git --version**

# Installing Git

**Windows (if you have problems)**

Follow the GitBash instructions here:

[Class Computer Setup Page](#)

**Mac (if you have problems)**

In the **Terminal** instal select developer tools

**xcode-select --install**

# Configuring Git

Do it Yourself!

# Configuring Git

(RStudio Terminal Tab)  
(...or *RStudio* > *Tools* > *Shell*)

*# First tell Git who you are*

```
> git config --global user.name "Barry Grant"  
> git config --global user.email "bjgrant@ucsd.edu"
```



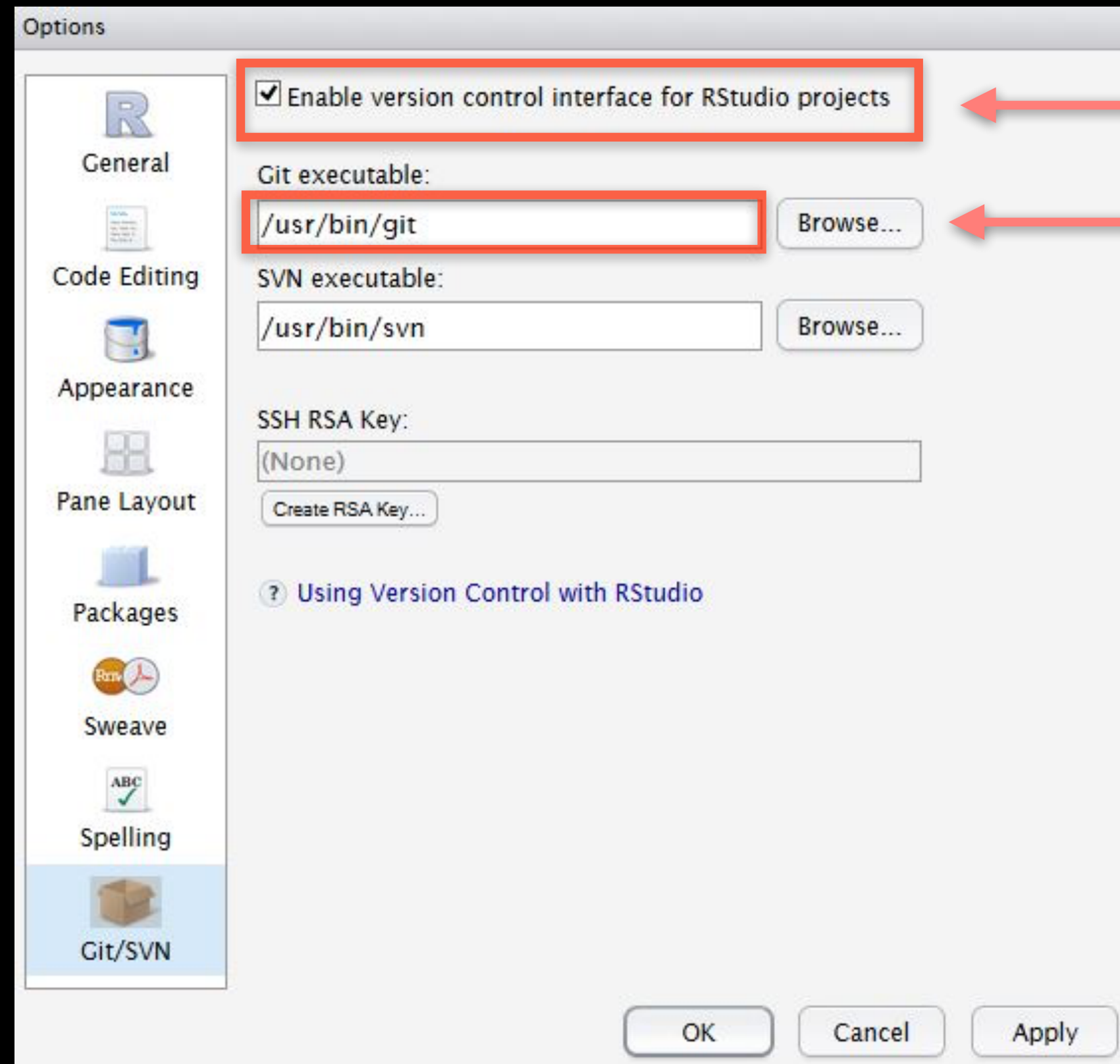
# Configuring RStudio

# For Mac & Linux

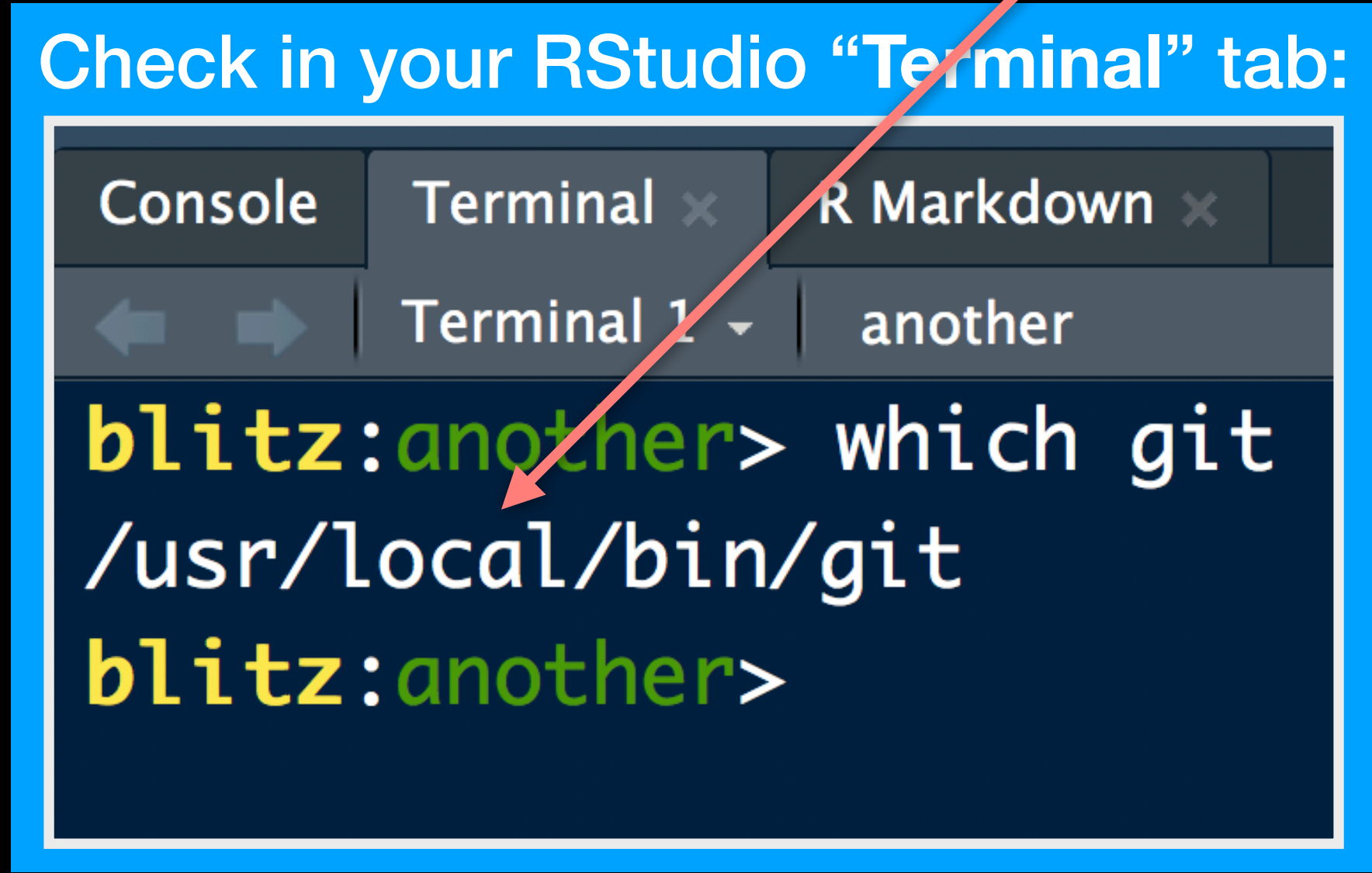
(PC on next slide)

Do it Yourself!

**Go to:** RStudio > Tools > Global Options > Git/SVN



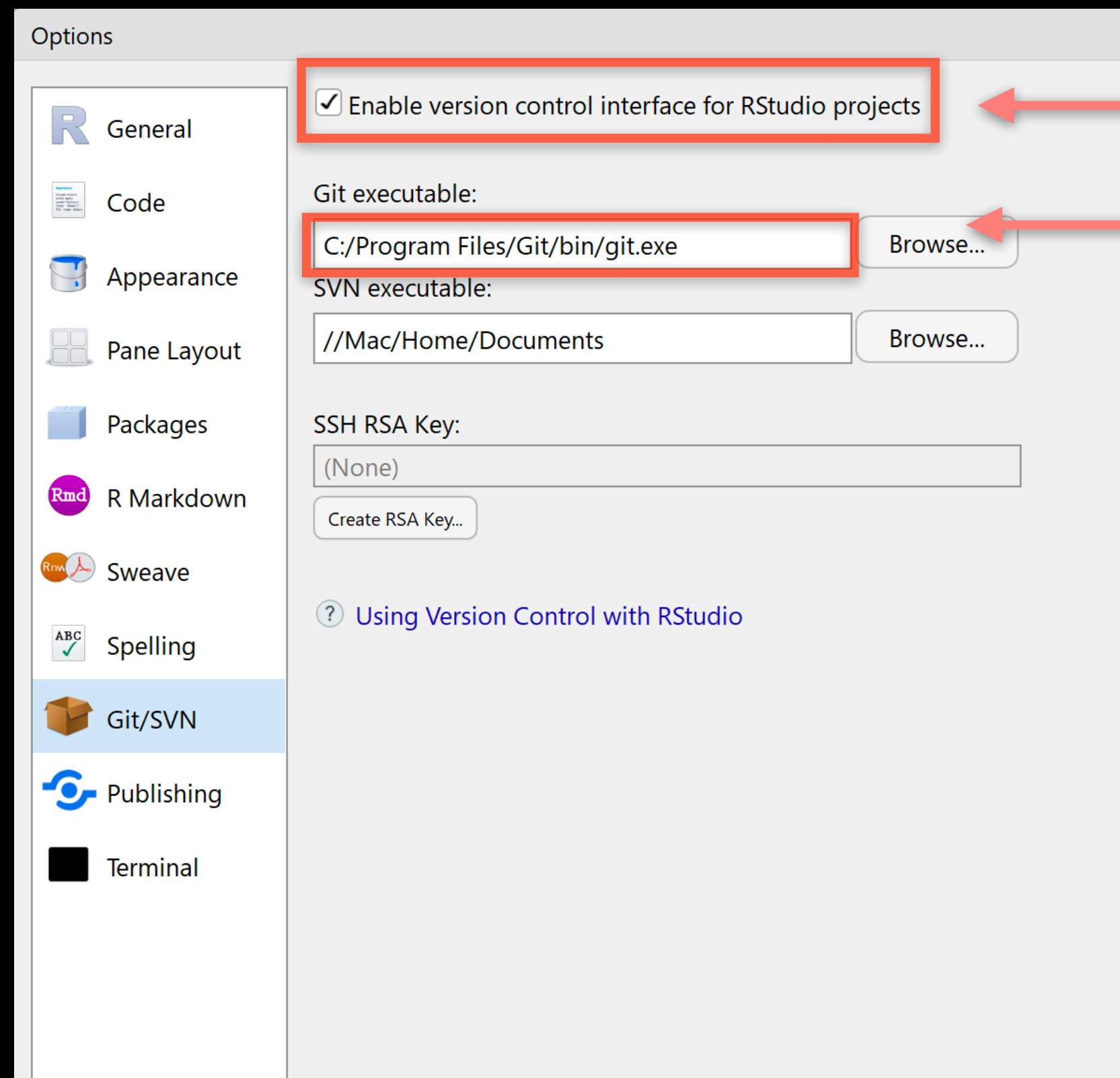
- 1 Make sure this is **ticked!**
- 2 Make sure this is **correct!**



Do it Yourself!

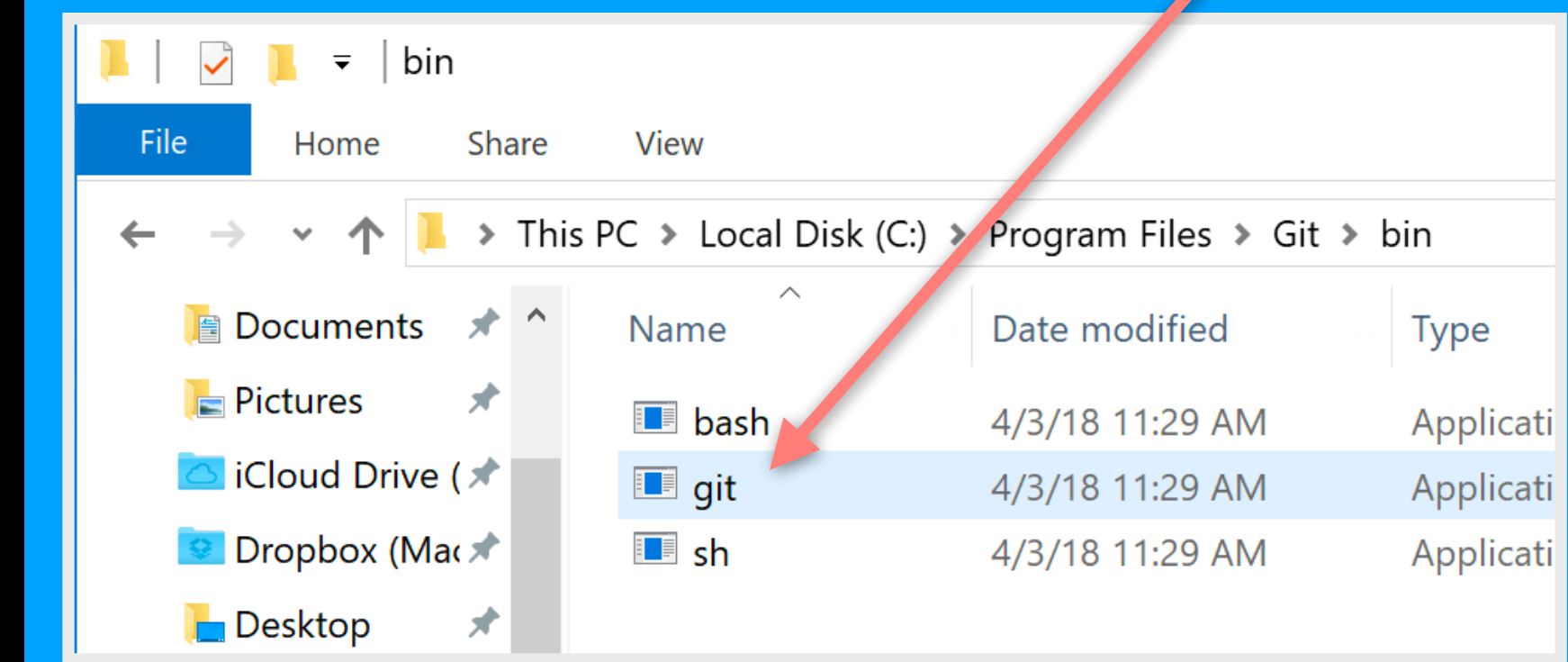
# On a PC!

**Go to:** RStudio > Tools > Global Options > Git/SVN



- 1 Make sure this is **ticked!**
- 2 This is the PATH for **PC!**

Check in your Windows File Explorer:



**Restart RStudio!**

# Using Git

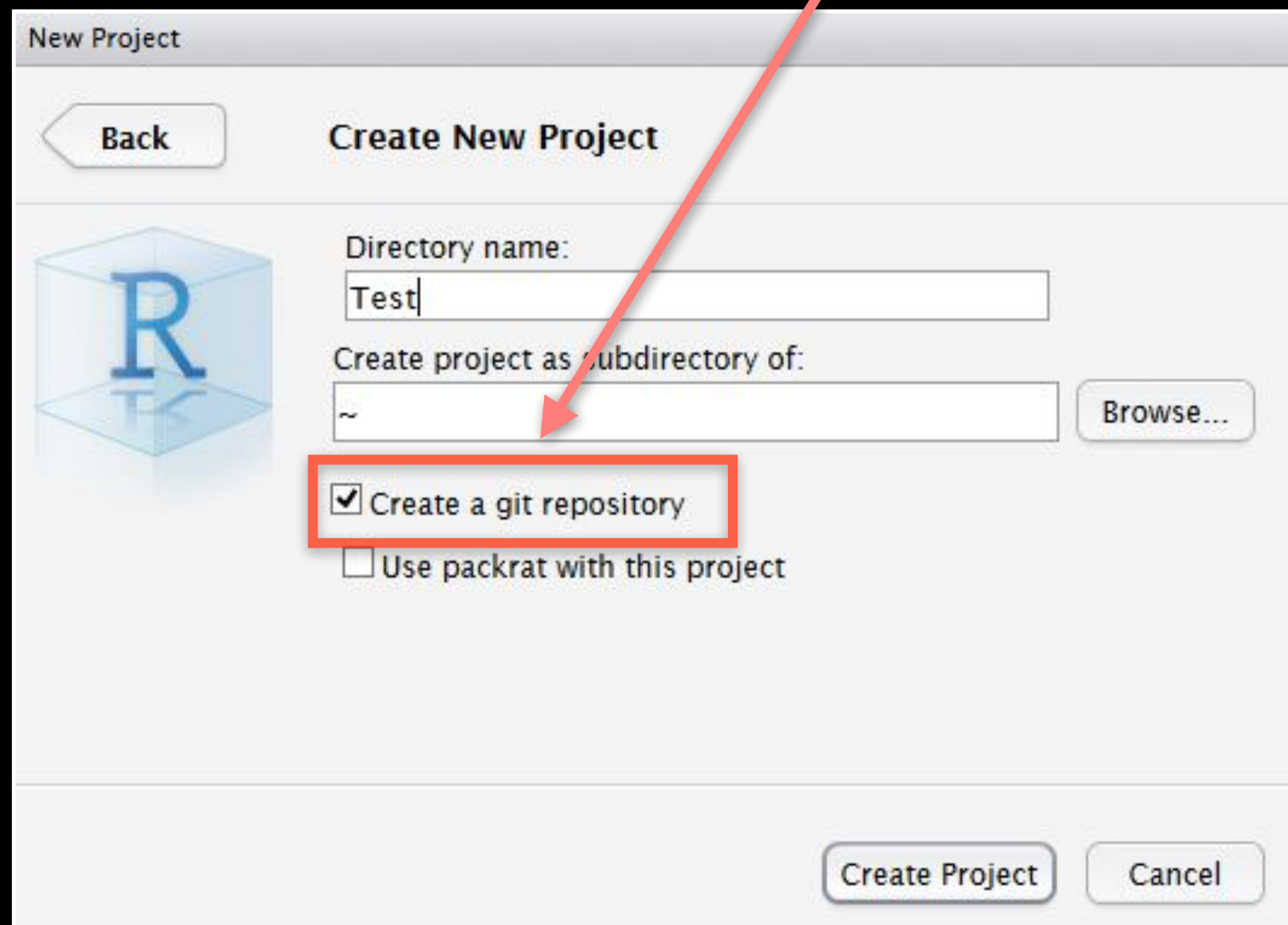
# Using Git

1. Initiate a Git repository.
2. Edit content (i.e. change some files).
3. Store a 'snapshot' of the current file state.\*

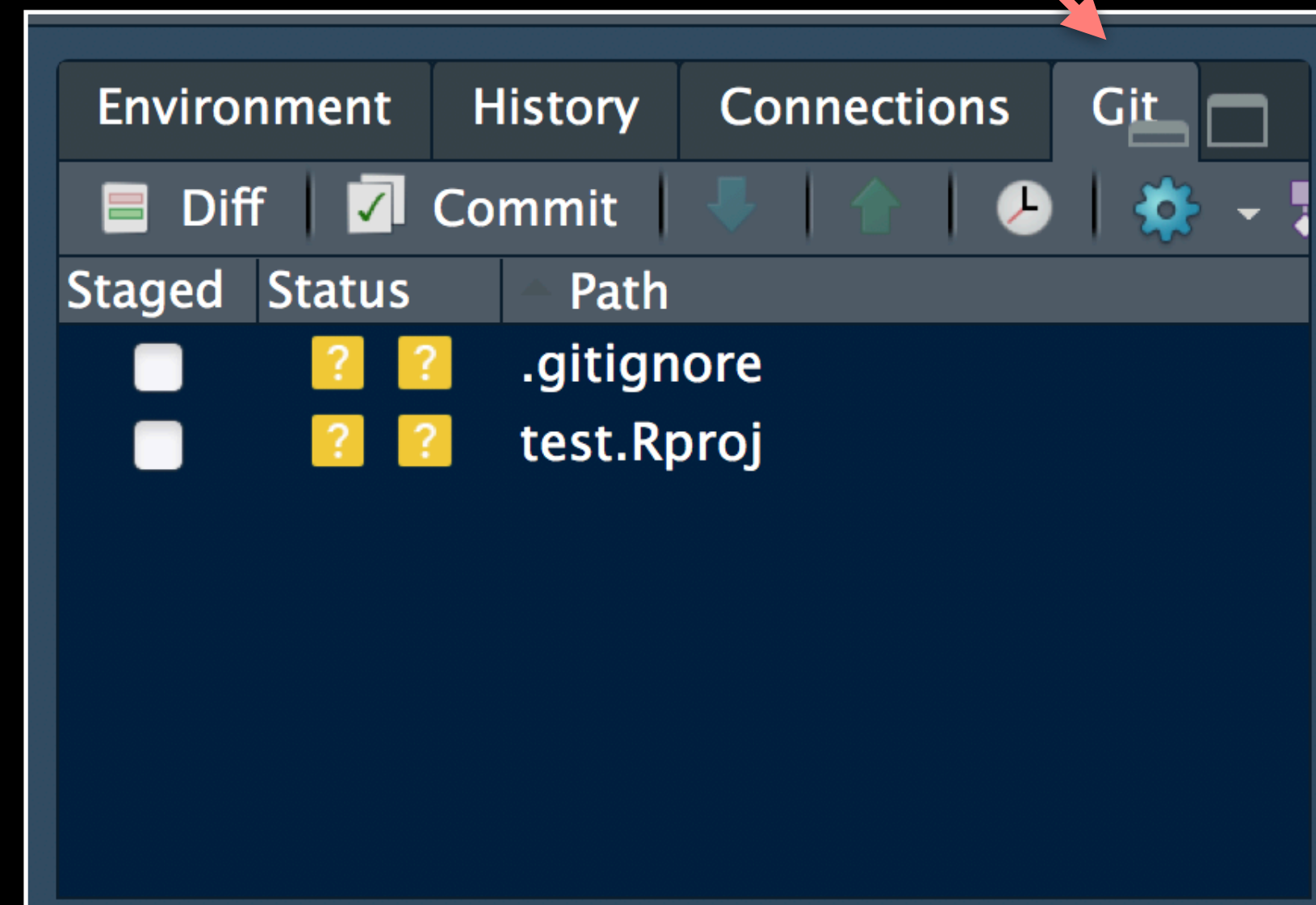
Do it Yourself!

# Create a new **Test** RStudio project

**1** New option to create a Git repository...



**2** New Git tab...



**Check if new Git options appear in RStudio?**

# Using Git in RStudio

1. **Initiate** a git repository for an RStudio Project
2. Do your work and edit content as normal
3. Store a 'snapshot' of the current file state
  - (a) Periodically **add** important files to git "Staging Area"
  - (b) **Commit** changes to your "git repository"

Rinse and repeat....

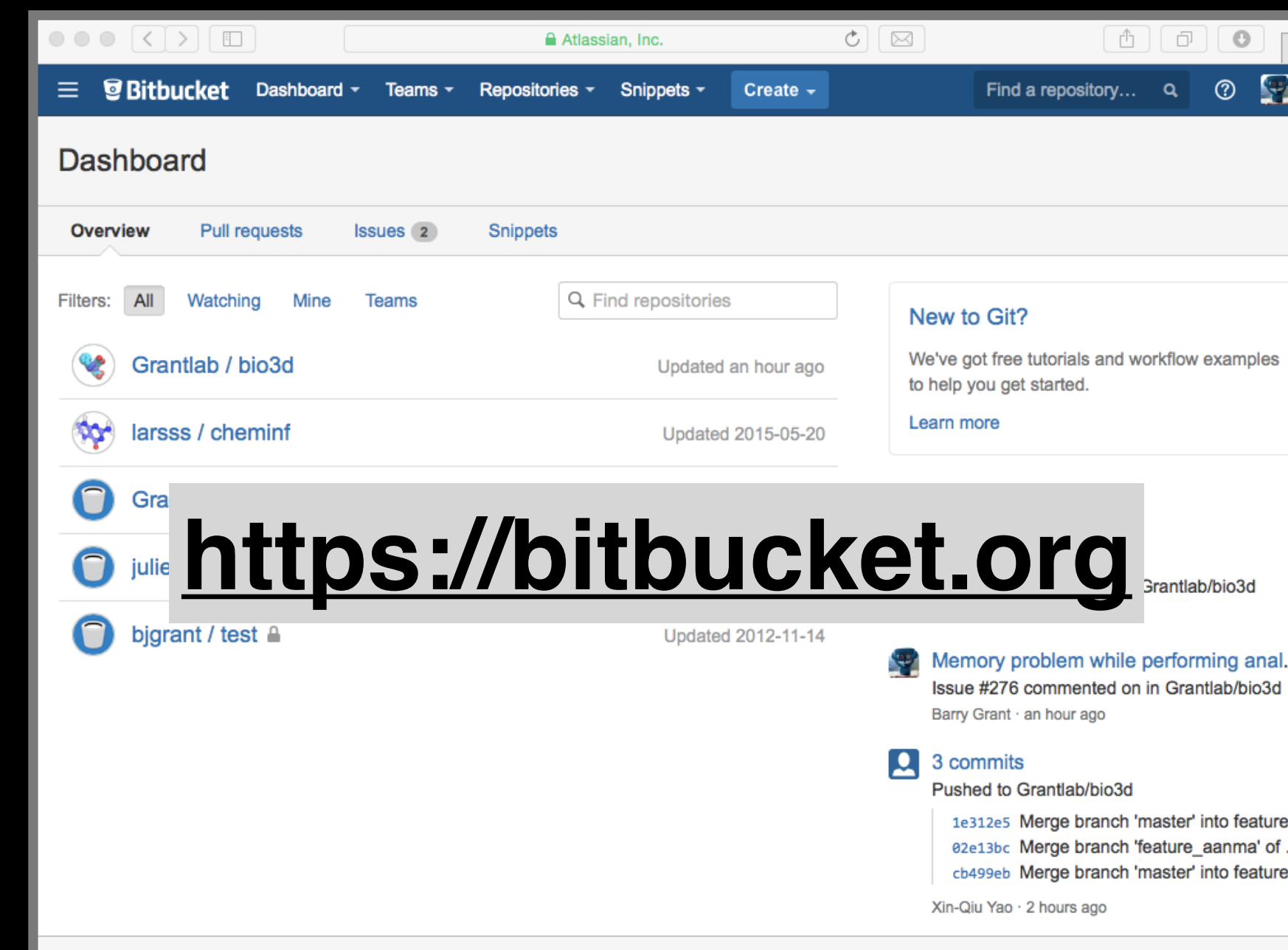
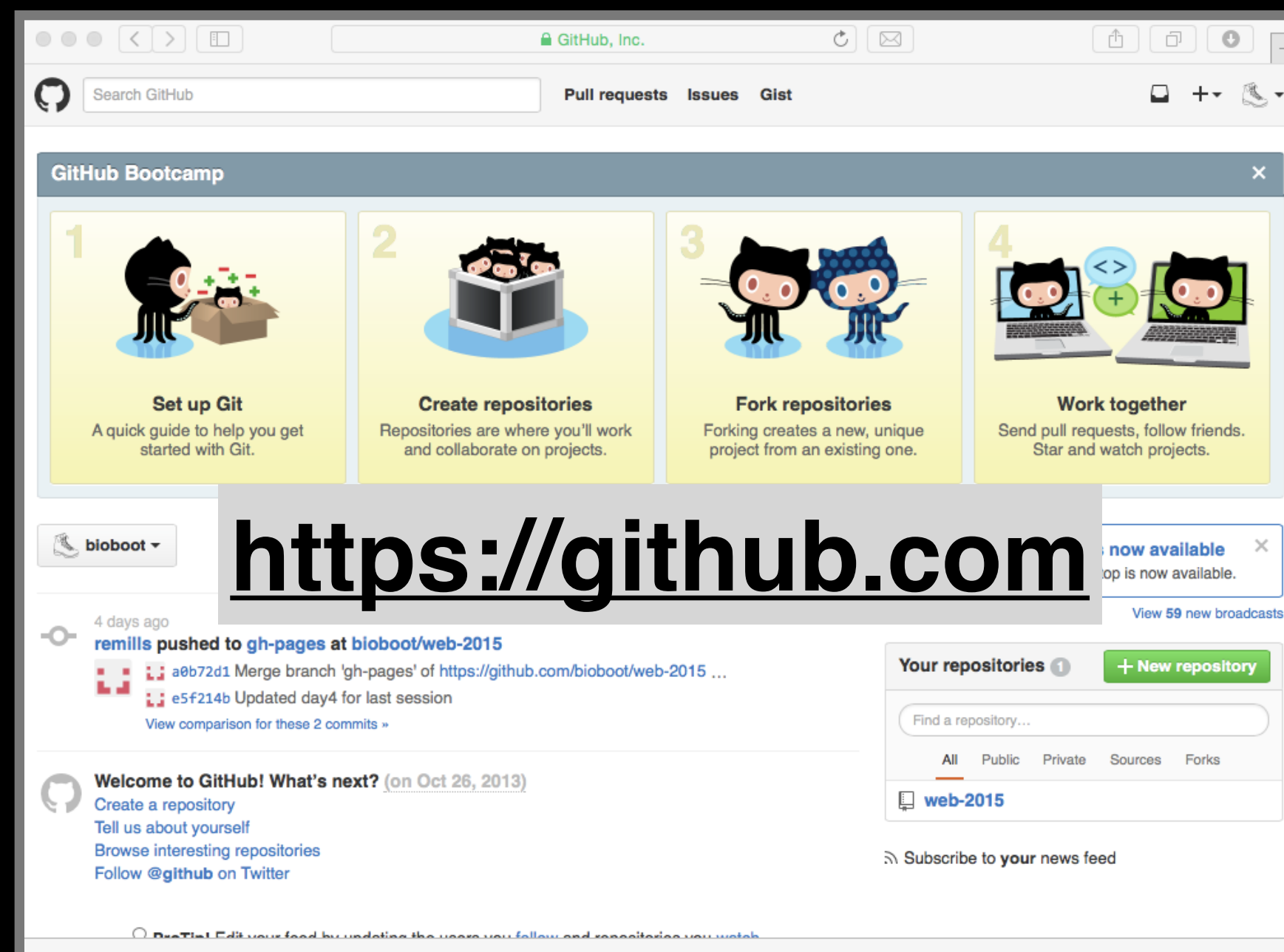


*Follow along!*

**Demo:**

# GitHub & Bitbucket

**GitHub** and **Bitbucket** are two popular hosting services for Git repositories. These services allow you to share your projects and collaborate with others using both **'public'** and **'private'** repositories\*.



Nikkei 17893.73 0.49%

Hang Seng 21404.96 0.72%

U.S. 10 Yr -0/32 Yield 2.074%

Crude Oil 39.17 -0.36%

Yen 119.16 0.26%

EXPAND

# THE WALL STREET JOURNAL.

Subscribe Now | Sign In  
**\$12 FOR 12 WEEKS**

Home World U.S. Politics Economy Business **Tech** Markets Opinion Arts Life Real Estate



Workers Get New Tools for Airing Their Gripes



Cell Carriers Battle for Wi-Fi Airwaves



Snapchat Names ex-Mattel Exec Vollerero Its Finance Chief



**YOU ARE READING A PREVIEW OF A PAID ARTICLE. [SUBSCRIBE NOW](#) TO GET MORE GREAT CONTENT.**



3234



433



TECH

## GitHub Raises \$250 Million at \$2 Billion Valuation

Capital raise puts company's total funding at \$350 million



### Analytics

How does your organization's talent measure up to its technology?

Read the MIT Sloan report



www.bbc.com/news/technology-44351214


Home Gmail Gcal Bitbucket GitHub BIMM143\_F18 BGGN213\_S18 BIMM-194 GDocs Disqus Blink News Atmosphere Galaxy + MMTF

**BBC** Sign in News Sport Weather Shop Reel Travel More Search

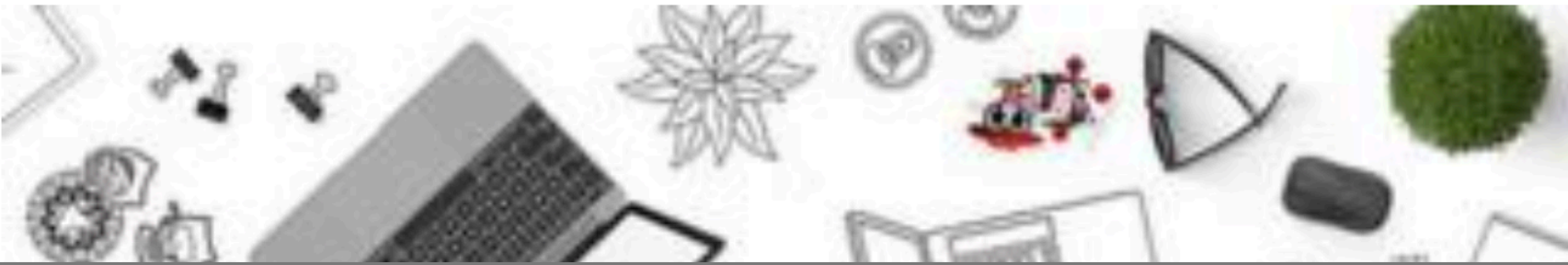
**NEWS**

Home Video World US & Canada UK Business Tech Science Stories Entertainment & Arts Health

# Microsoft buys Github code-sharing site for \$7.5bn

 **Dave Lee**  
North America technology reporter

🕒 4 June 2018 | 📧 Share



## Top Stories

**Gangster 'Whitey' Bulger killed in prison**  
Bulger was severely beaten by one or more inmates shortly after arriving at the prison, sources say.  
🕒 30 October 2018

**Synagogue shooting victims' funerals start**  
🕒 30 October 2018

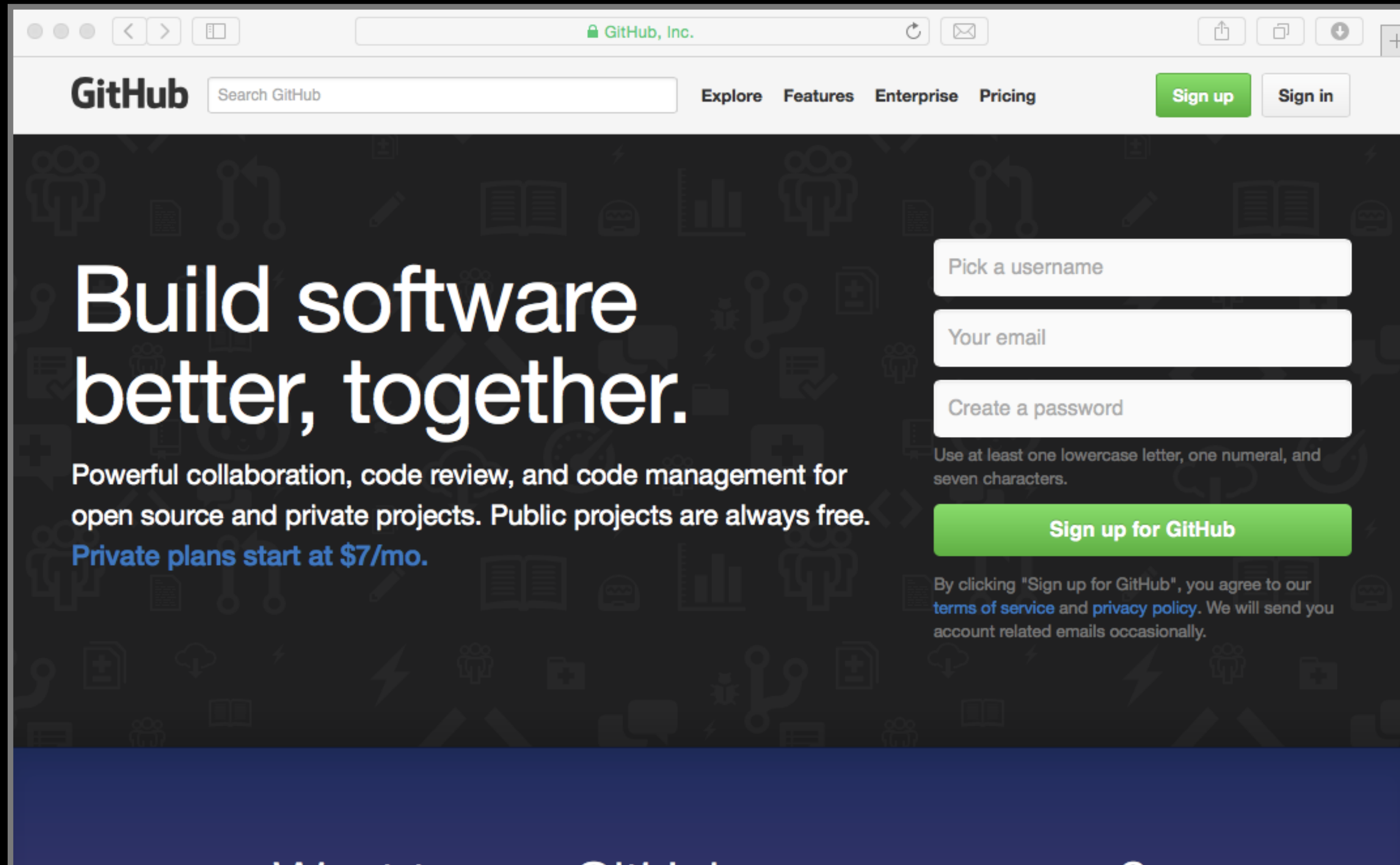
**What do American voters care about?**  
🕒 30 October 2018

# What is the big deal?

- At the simplest level GitHub and Bitbucket offer **backup** of your projects history and a centralized mechanism for **sharing** with others by putting **your Git repo online**.
  - GitHub in particular is often referred to as the “nerds FaceBook and LinkedIn combined”.
- At their core both services **offer a new paradigm for open collaborative project development**, particularly for software.
  - In essence they allow anybody to contribute to any public project and get acknowledgment.

# First sign up for a GitHub account

<https://github.com>

A screenshot of the GitHub website's sign-up page. The browser's address bar shows "GitHub, Inc." and the page title is "GitHub". The navigation bar includes "Search GitHub", "Explore", "Features", "Enterprise", "Pricing", "Sign up", and "Sign in". The main content area features the slogan "Build software better, together." and a description of GitHub's services. On the right, there is a sign-up form with three input fields: "Pick a username", "Your email", and "Create a password". Below the password field is a note: "Use at least one lowercase letter, one numeral, and seven characters." A green "Sign up for GitHub" button is positioned below the form. At the bottom of the form, there is a disclaimer: "By clicking 'Sign up for GitHub', you agree to our terms of service and privacy policy. We will send you account related emails occasionally." The background of the page is dark with a pattern of faint icons related to software development.

GitHub, Inc.

GitHub Search GitHub Explore Features Enterprise Pricing Sign up Sign in

## Build software better, together.

Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free. Private plans start at \$7/mo.

Pick a username

Your email

Create a password

Use at least one lowercase letter, one numeral, and seven characters.

**Sign up for GitHub**

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally.

# Pick the **FREE** plan!

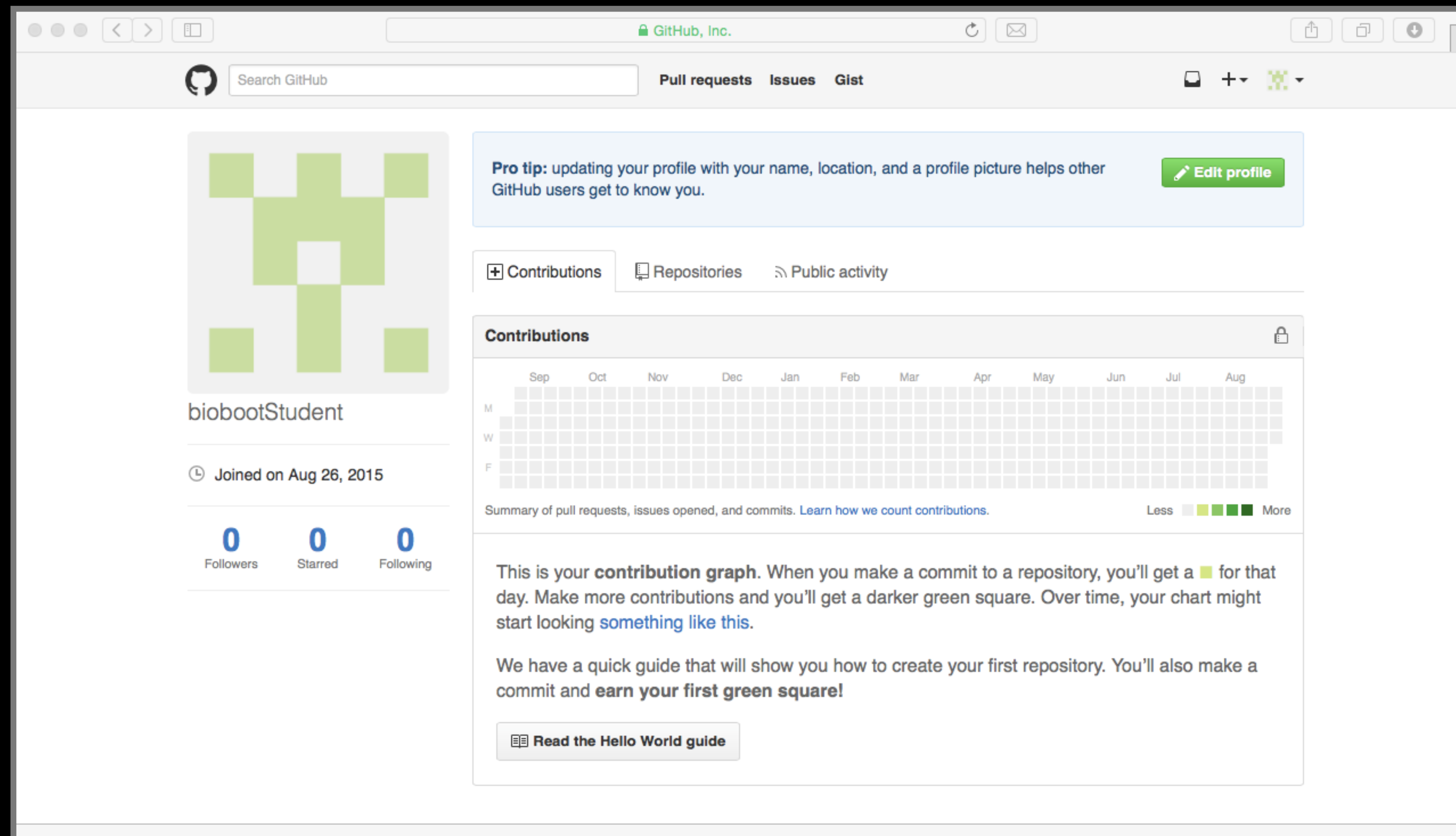
The screenshot shows the GitHub account setup process. The browser address bar displays 'GitHub, Inc.'. The navigation bar includes a search bar, 'Pull requests', 'Issues', and 'Gist'. The main heading is 'Welcome to GitHub' with a sub-heading 'You've taken your first step into a larger world, @biobootStudent.'. A progress bar shows three steps: 'Completed: Set up a personal account', 'Step 2: Choose your plan', and 'Step 3: Go to your dashboard'. Below this is a table titled 'Choose your personal plan' with columns for Plan, Cost, and Private repositories. The 'Free' plan is highlighted in blue and its 'Chosen' button is circled in red. To the right, a box titled 'Each plan includes:' lists features like 'Unlimited collaborators', 'Unlimited public repositories', 'Free setup', 'HTTPS Protection', 'Email support', and 'Wikis, Issues, Pages, & more'. At the bottom, a disclaimer states that charges are in US Dollars and prices are estimates based on current exchange rates.

Plan	Cost	Private repositories	
Large	\$50/month	50	Choose
Medium	\$22/month	20	Choose
Small	\$12/month	10	Choose
Micro	\$7/month	5	Choose
Free	\$0/month	0	Chosen

Charges to your account will be made in **US Dollars**. Converted prices are provided as a convenience and are only an *estimate* based on *current* exchange rates. Local prices will change as the exchange rate fluctuates.  
Don't worry, you can cancel or upgrade at any time.

# Your GitHub homepage

Check your email for verification request



The screenshot shows a web browser window displaying the GitHub homepage for a user named 'biobootStudent'. The browser's address bar shows 'GitHub, Inc.' and the page title is 'GitHub, Inc.'. The navigation bar includes a search box, 'Pull requests', 'Issues', and 'Gist' links. The main content area features a profile picture placeholder (a green grid pattern), the username 'biobootStudent', and a 'Joined on Aug 26, 2015' timestamp. Below this, there are three statistics: '0 Followers', '0 Starred', and '0 Following'. A 'Pro tip' banner suggests updating the profile. The 'Contributions' section is active, showing a grid of squares representing contributions from September to August. A legend indicates that the number of squares represents the number of pull requests, issues, or commits. A 'Read the Hello World guide' button is located at the bottom of the contributions section.

GitHub, Inc.

Search GitHub Pull requests Issues Gist

Pro tip: updating your profile with your name, location, and a profile picture helps other GitHub users get to know you. [Edit profile](#)

Contributions Repositories Public activity

Contributions

Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug

M

W

F

Summary of pull requests, issues opened, and commits. [Learn how we count contributions.](#) Less More

This is your **contribution graph**. When you make a commit to a repository, you'll get a ■ for that day. Make more contributions and you'll get a darker green square. Over time, your chart might start looking [something like this](#).

We have a quick guide that will show you how to create your first repository. You'll also make a commit and **earn your first green square!**

[Read the Hello World guide](#)



# Connecting RStudio to GitHub

Create a **Personal Access Token** (PAT) on GitHub

See **section 4** of lab worksheet

# Skip the hello-world tutorial

<https://guides.github.com/activities/hello-world/>

GitHub, Inc.

Home Gmail GCal WolverineAccess 2delicious 2CiteULike 2Papers UMProxy + Gscholar Plex It! ToRead SCALI Bioinf525\_Video Bio3P... App index of /

Search GitHub Pull requests Issues Gist

Your email was verified.

## Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Let's get started!](#)

biobootStudent

### Welcome to GitHub! What's next? (3 hours ago)

- [Create a repository](#)
- [Tell us about yourself](#)
- [Browse interesting repositories](#)
- [Follow @github on Twitter](#)

**ProTip!** Edit your feed by updating the users you [follow](#) and repositories you [watch](#).

**Your repositories** 0 [+ New repository](#)

You don't have any repositories yet!  
[Create your first repository](#) or [learn more about Git and GitHub](#).

**ProTip!** [Feline cephalopod adhesives](#) are great for decorating portable computation devices.

[Subscribe to your news feed](#)

# Name your repo

## bggn213

Home Email Calendar Dropbox GitHub News Bivim45\_316 BGN213\_316 Disqus Bivim194 Atmosphere Bink GDocs Galaxy

Goog Computer Se... rstudio\_test/... kebabs pack... Institute of B... bioconducto... BIMM-143, L... Happ Create a Ne...

Search GitHub Pull requests Issues Marketplace Explore

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: bioboot Repository name: **bggn213** ✓

Great repository names are short and memorable. Need inspiration? How about [cuddly-invention](#).

Description (optional)

**Public**  
Anyone can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

**Add a README**  **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

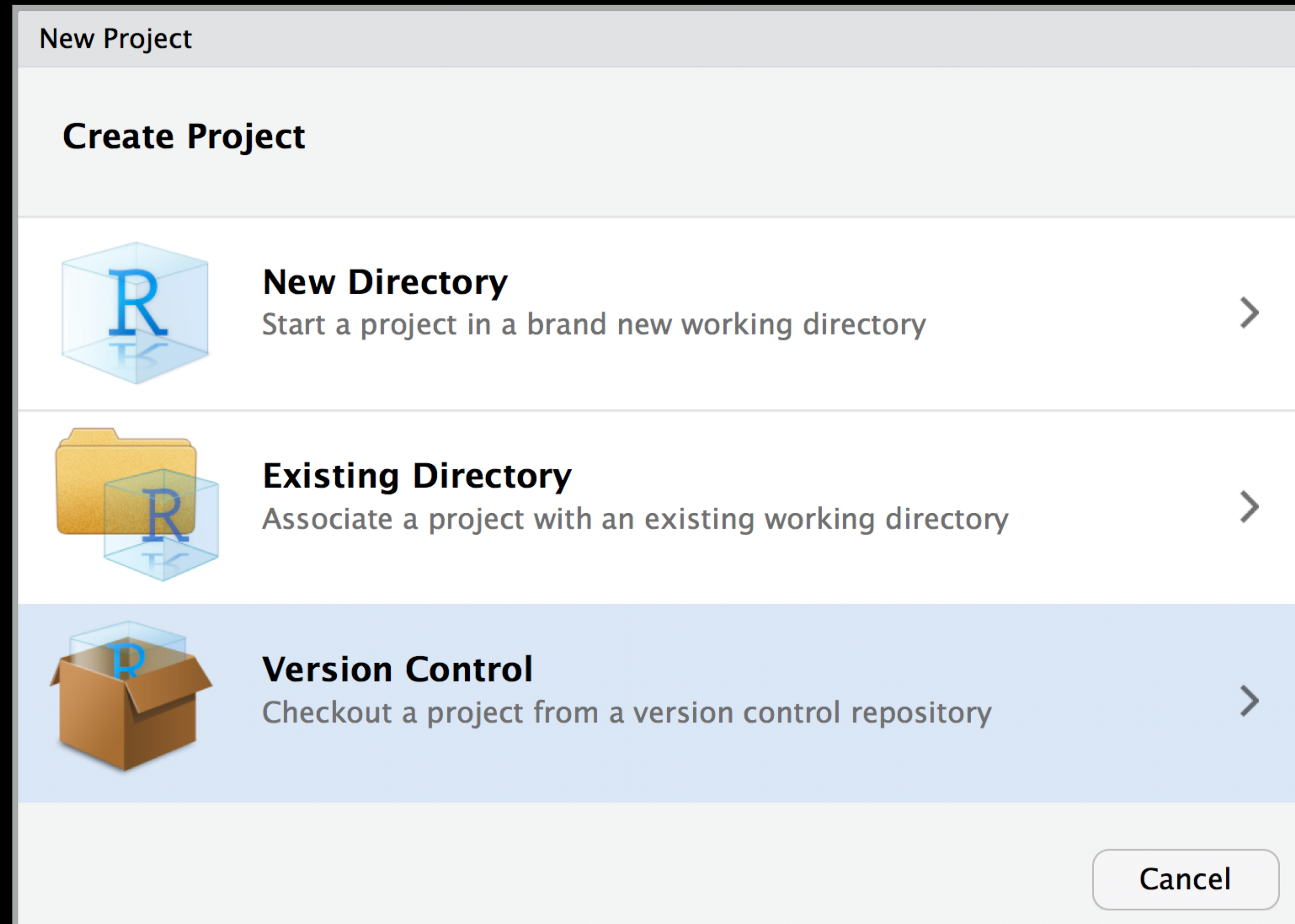
**Create repository** **Create**

# Copy the “Clone” HTTPS link

This screenshot shows the GitHub interface for the repository `bioboot / bimm143`. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The `Clone or download` button is highlighted in green. The dropdown menu is open, showing the `Clone with HTTPS` option, which is highlighted in red. The URL `https://github.com/bioboot/bimm143.git` is also highlighted in red. The repository content shows a file named `README.md` with the text `bggn213`.

GitHub repository page for `bioboot / bimm143`. The page shows the repository name, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The `Clone or download` button is highlighted in green, and the dropdown menu is open, showing the `Clone with HTTPS` option, which is highlighted in red. The URL `https://github.com/bioboot/bimm143.git` is also highlighted in red. The repository content shows a file named `README.md` with the text `bggn213`.


# RStudio > New Project > Version Control



# RStudio > New Project > Version Control

New Project

[Back](#) **Clone Git Repository**



Repository URL:

Project directory name:

Create project as subdirectory of:  
 [Browse...](#)

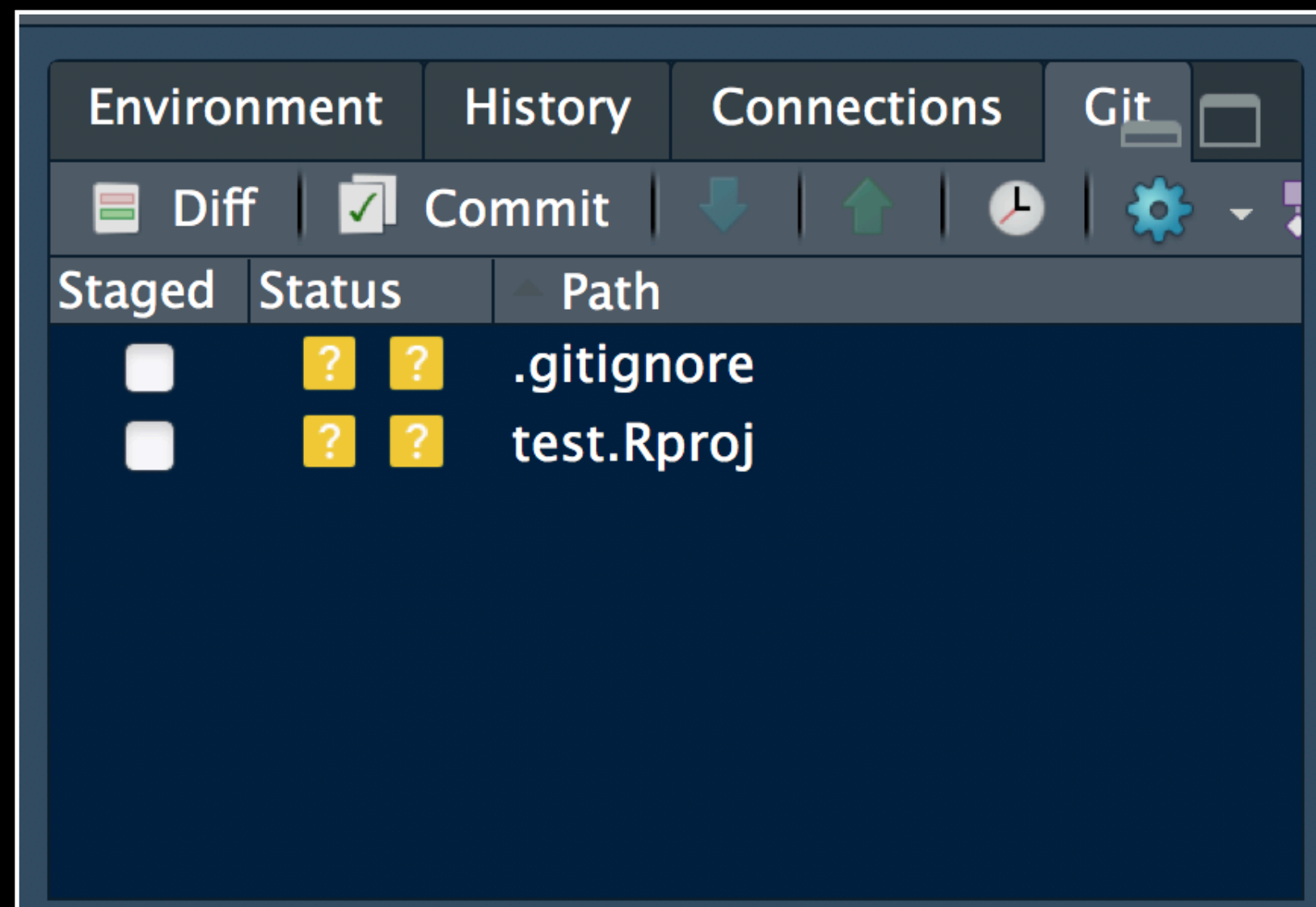
[GitHub Paste](#)

Open in new session

[Create Project](#) [Cancel](#)

# Demo of *editing*, *adding* *committing* and *pushing*

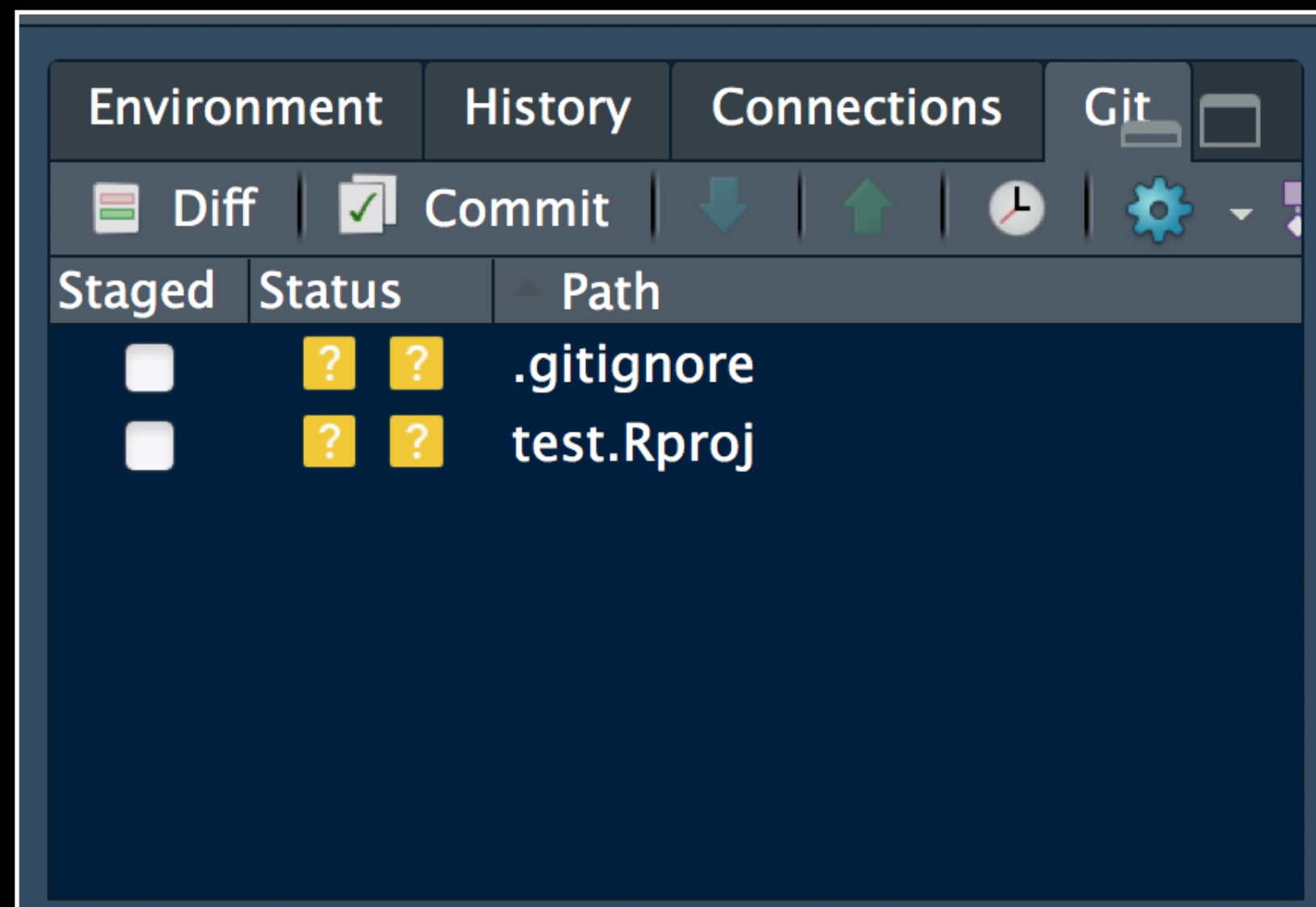
**Check if new Git tab  
Appears in RStudio?**



**Now experiment editing the  
README.md file in RStudio  
and adding, committing and  
pushing changes to GitHub  
via this tab**

# Demo of *editing, adding committing and pushing*

**Check if new Git tab  
Appears in RStudio?**



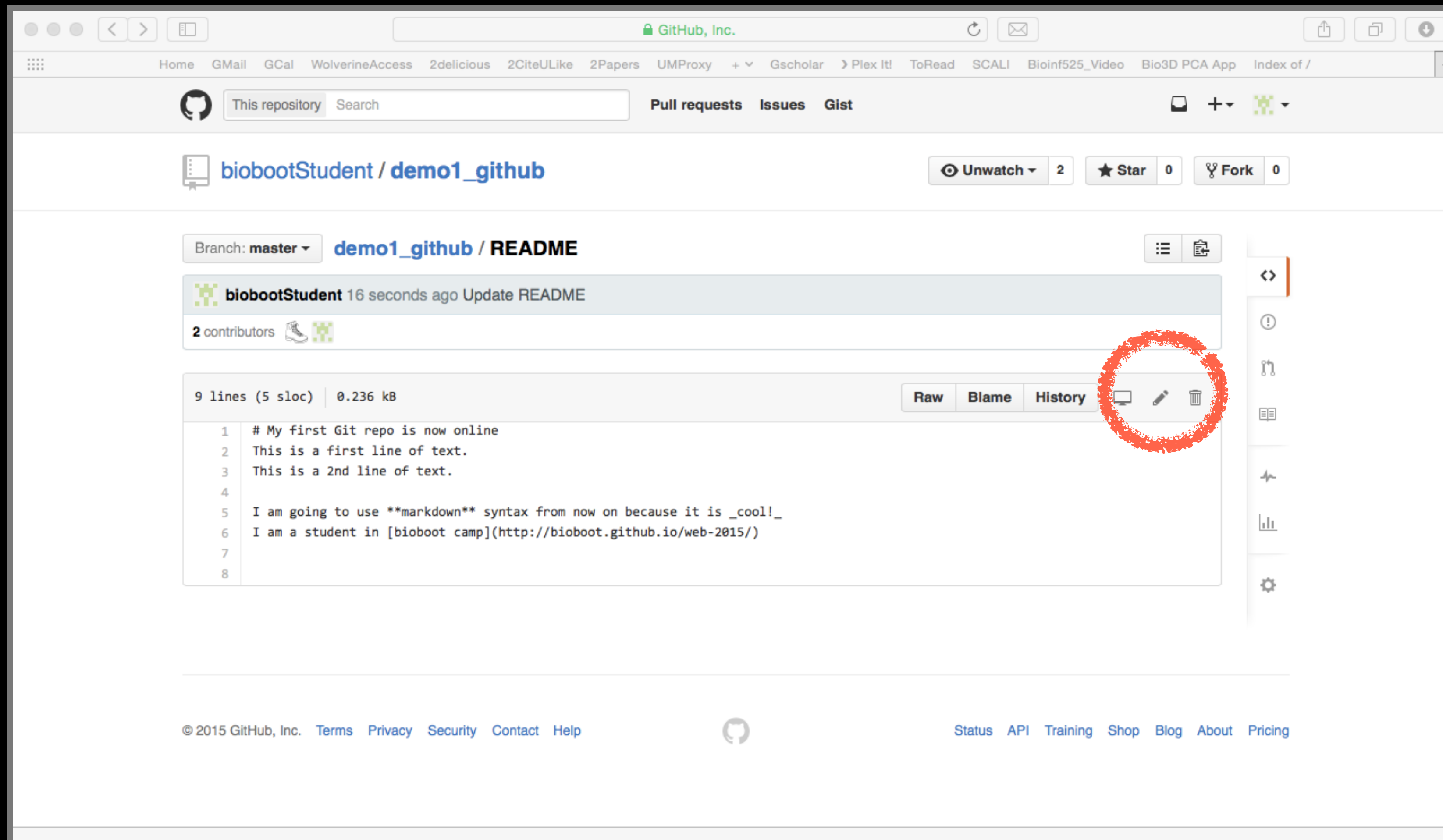
**Now experiment editing the  
README.md file in RStudio  
and adding, committing and  
pushing changes to GitHub  
via this tab**

**When you are ready copy your  
different class directories/projects  
to this new GitHub tracked folder**



# Side-note: How to edit online

Specifically lets add some Markdown content



The screenshot shows a web browser window displaying a GitHub repository page for 'biobootStudent / demo1\_github'. The repository is on the 'master' branch. The README file is shown with 9 lines of text. The edit icon (a pencil) is circled in red, indicating the option to edit the file online. The README content is as follows:

```
1 # My first Git repo is now online
2 This is a first line of text.
3 This is a 2nd line of text.
4
5 I am going to use markdown syntax from now on because it is _cool!
6 I am a student in [bioboot camp](http://bioboot.github.io/web-2015/)
7
8
```

At the bottom of the page, there is a footer with copyright information and navigation links.

# Summary

- Git is a popular 'distributed' version control system that is lightweight and free
- GitHub and BitBucket are popular hosting services for git repositories that have changed the way people contribute to open source projects
- Introduced basic git and GitHub usage within RStudio and encouraged you to adopt these 'best practices' for your future projects.

# Learning Resources

- **Set up Git**. If you will be using Git mostly or entirely via **GitHub**, look at these how-tos.  
< <https://help.github.com/categories/bootcamp/> >
- **Getting Git Right**. Excellent **Bitbucket** git tutorials  
< <https://www.atlassian.com/git/> >
- **Pro Git**. A complete, book-length guide and reference to Git, by Scott Chacon and Ben Straub.  
< <http://git-scm.com/book/en/v2> >
- **StackOverflow**. Excellent programming and developer Q&A.  
< <http://stackoverflow.com/questions/tagged/git> >

# Learning git can be painful!

However in practice it is not nearly as crazy-making as the alternatives:

- Documents as email attachments
- Hair-raising ZIP archives containing file salad
- Am I working with the most recent data?
- Archaeological “digs” on old email threads and uncertainty about how/if certain changes have been made or issues solved

Finally Please remember that **GitHub**  
and **BitBucket** are **PUBLIC** and that  
you should cultivate your professional  
and scholarly profile with intention!



**BGGN 213**

**Hands-on Lab Session**

**Class 07**

**Barry Grant**

**UC San Diego**

<http://thegrantlab.org/bggn213>

# Reference Slides

# Using Command Line Git

1. Initiate a Git repository.
2. Edit content (i.e. change some files).
3. Store a 'snapshot' of the current file state.\*



**Initiate** a Git repository

# Initiate a Git repository

```
> cd ~/Desktop
> mkdir git_class # Make a new directory
> cd git_class # Change to this directory
> git init # Our first Git command!
> ls -a # what happened?
```

# Side-Note: The `.git/` directory

- Git created a 'hidden' `.git/` directory inside your current working directory.
- You can use the `'ls -a'` command to list (*i.e.* see) this directory and its contents.
- This is where Git stores all its goodies - **this is Git!**
- You should not need to edit the contents of the `.git` directory for now but do feel free to poke around.

# Important Git commands

```
> git status      # report on content changes
```

```
> git add <filename>    # stage/track a file
```

```
> git commit -m "message"  # snapshot
```

# Important Git commands

```
> git status      # report on content changes
```

```
> git add <filename>  # stage/track a file
```

```
> git commit -m "message"  # snapshot
```



*You will use these three commands again and again in your Git workflow!*

# Git TRACKS your directory content

- To get a report of changes (since last commit) use:  
> **git status**

- You tell Git which files to track with:

> **git add <filename>**

This adds files to a so called **STAGING AREA** (akin to a “shopping cart” before purchasing).

- You tell Git when to take an historical **SNAPSHOT** of your staged files (*i.e.* record their current state) with:  
> **git commit -m ‘Your message about changes’**

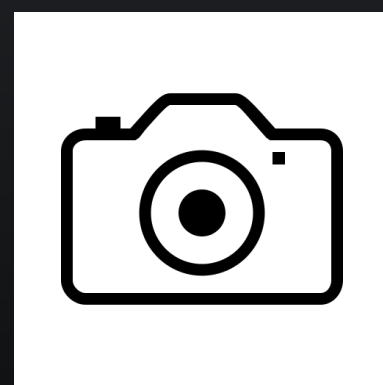
# Example Git workflow



Eva creates a README text file  
(this starts as untracked)



Adds file to STAGING AREA\*  
(tracked and ready to take a snapshot)



Commit changes\*  
(records snapshot of staged files!)

Hands on example!

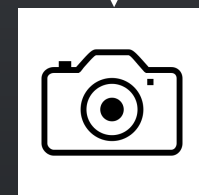
# Example Git workflow



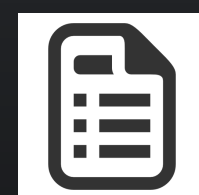
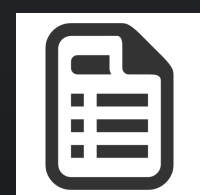
Eva creates a README text file



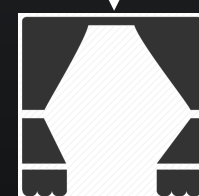
Adds file to STAGING AREA\*



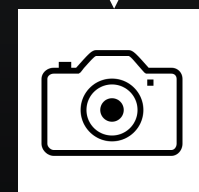
Commit changes\*



Eva modifies README and adds a ToDo text file



Adds both to STAGING AREA\*



Commit changes\*



# 1. Eva creates a README file

```
> # cd ~/Desktop/git_class
> # git init

> echo "This is a first line of text." > README
> git status      # Report on changes

# On branch master
#
# Initial commit
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
#  README
#
# nothing added to commit but untracked files present (use "git add" to track)
```

## 2. Adds to 'staging area'

```
> git add README      # Add README file to staging area
> git status          # Report on changes
```

```
# On branch master
```

```
#
```

```
# Initial commit
```

```
#
```

```
# Changes to be committed:
```

```
# (use "git rm --cached <file>..." to unstage)
```

```
#
```

```
#   new file:   README
```

```
#
```

# 3. Commit changes

```
> git commit -m "Create a README file" # Take snapshot  
# [master (root-commit) 8676840] Create a README file  
# 1 file changed, 1 insertion(+)  
# create mode 100644 README
```

```
> git status # Report on changes  
# On branch master  
# nothing to commit, working directory clean
```

## 4. Eva modifies README file and adds a ToDo file

```
> echo "This is a 2nd line of text." >> README
```

```
> echo "Learn git basics" >> ToDo
```

```
> git status      # Report on changes
```

```
# On branch master
```

```
#
```

```
# Changes not staged for commit:
```

```
# (use "git add <file>..." to update what will be committed)
```

```
# (use "git checkout -- <file>..." to discard changes in working directory)
```

```
#
```

```
#    modified:   README
```

```
#
```

```
# Untracked files:
```

```
# (use "git add <file>..." to include in what will be committed)
```

```
#
```

```
#    ToDo
```

```
#
```

```
# no changes added to commit (use "git add" and/or "git commit -a")
```

## 5. Adds both files to 'staging area'

```
> git add README ToDo # Add both files to 'staging area'
```

```
> git status # Report on changes
```

```
# On branch master
```

```
# Changes to be committed:
```

```
# (use "git reset HEAD <file>..." to unstage)
```

```
#
```

```
#   modified:   README
```

```
#   new file:   ToDo
```

```
#
```

## 6. Commits changes

```
> git commit -m "Add ToDo and modify README"
```

```
# [master 7b679fa] Add ToDo and modify README
```

```
# 2 files changed, 2 insertions(+)
```



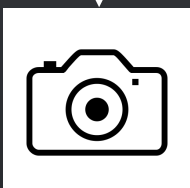
```
# create mode 100644 ToDo
```



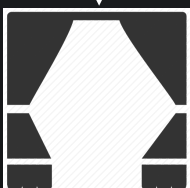
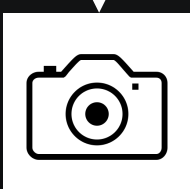
```
> git status
```

```
# On branch master
```

```
# nothing to commit, working directory clean
```

# Example Git workflow

1.  Eva creates a README text file
2.  Adds file to STAGING AREA\*
3.  Commit changes\*

4.   Eva modifies README and adds a ToDo text file
5.  Adds both to STAGING AREA\*
6.  Commit changes\*

...But, how do we see the history of our project changes?

# git log: Timeline history of snapshots (*i.e.* commits)

> **git log**

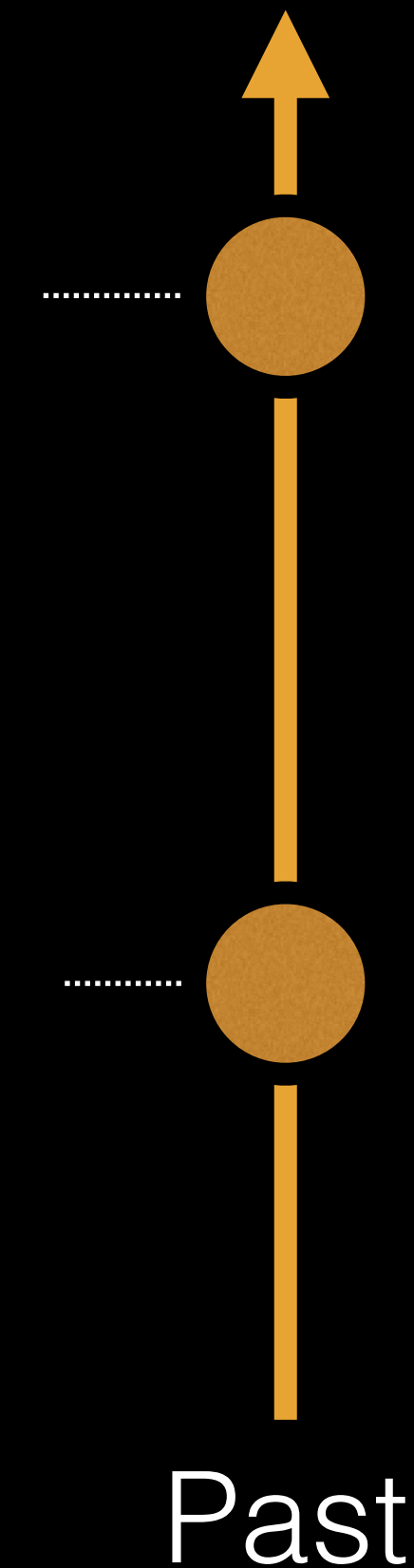
```
# commit 7b679fa747e8640918fcaad7e4c3f9c70c87b170
# Author: Barry Grant <bjgrant@umich.edu>
# Date: Thu Jul 30 11:43:40 2015 -0400
#
#   Add ToDo and finished README
#
# commit 86768401610770ae32e2fd4faee07d1d5c68619c
# Author: Barry Grant <bjgrant@umich.edu>
# Date: Thu Jul 30 11:26:40 2015 -0400
#
#   Create a README file
#
```



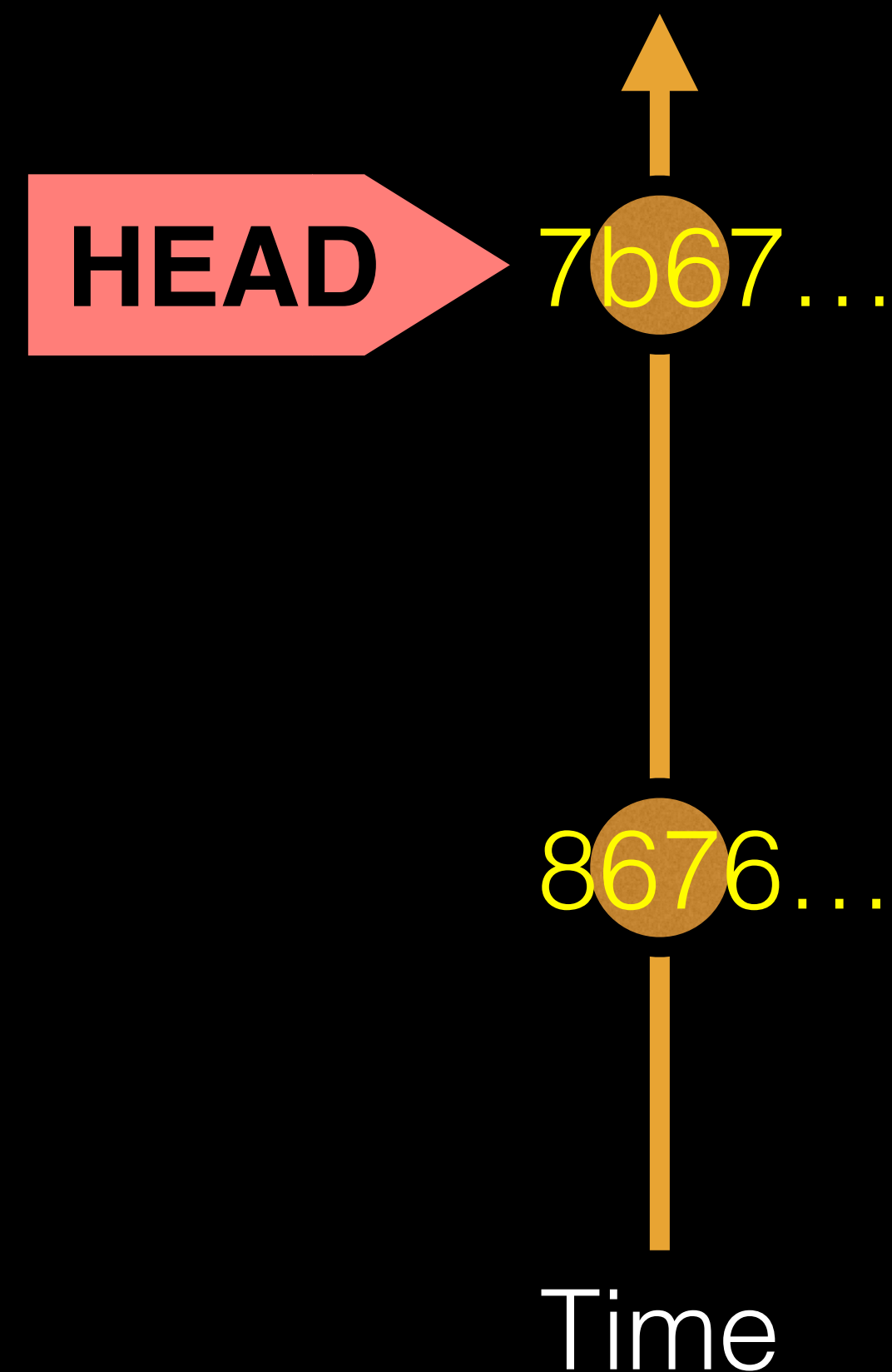
# git log: Timeline history of snapshots (*i.e.* commits)

> **git log**

```
# commit 7b679fa747e8640918fcaad7e4c3f9c70c87b170
# Author: Barry Grant <bjgrant@umich.edu>
# Date: Thu Jul 30 11:43:40 2015 -0400
#
#   Add ToDo and finished README
#
# commit 86768401610770ae32e2fd4faee07d1d5c68619c
# Author: Barry Grant <bjgrant@umich.edu>
# Date: Thu Jul 30 11:26:40 2015 -0400
#
#   Create a README file
#
```



# Side-Note: Git history is akin to a graph

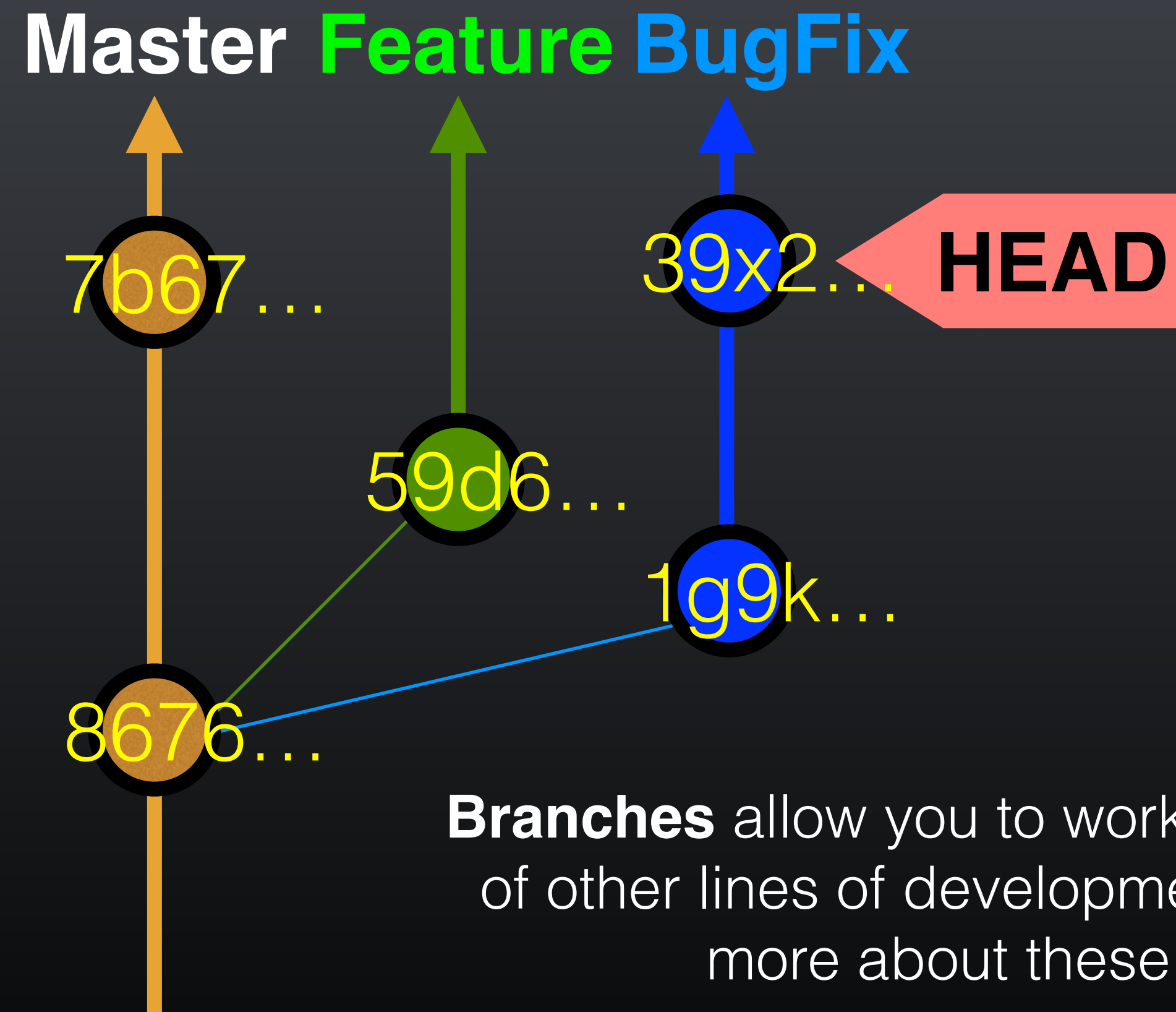


Nodes are **commits** labeled by their unique '**commit ID**'.

(This is a CHECKSUM of the commits author, time, commit msg, commit content and previous commit ID).

**HEAD** is a reference (or '**pointer**') to the currently checked out commit (typically the most recent commit).

Projects can have complicated graphs due to **branching**



**Branches** allow you to work independently of other lines of development we will talk more about these later!

## Key Points:

You explicitly and iteratively tell git what files to track (“**git add**”) and snapshot (“**git commit**”).

Git keeps an historical log (“**git log**”) of the content changes (and your comments on these changes) at each past commit.

It is good practice to regularly check the status of your working directory, staging arena repo (“**git status**”)

Break

# Summary of key Git commands:

> **git status** # Get a status report of changes since last commit

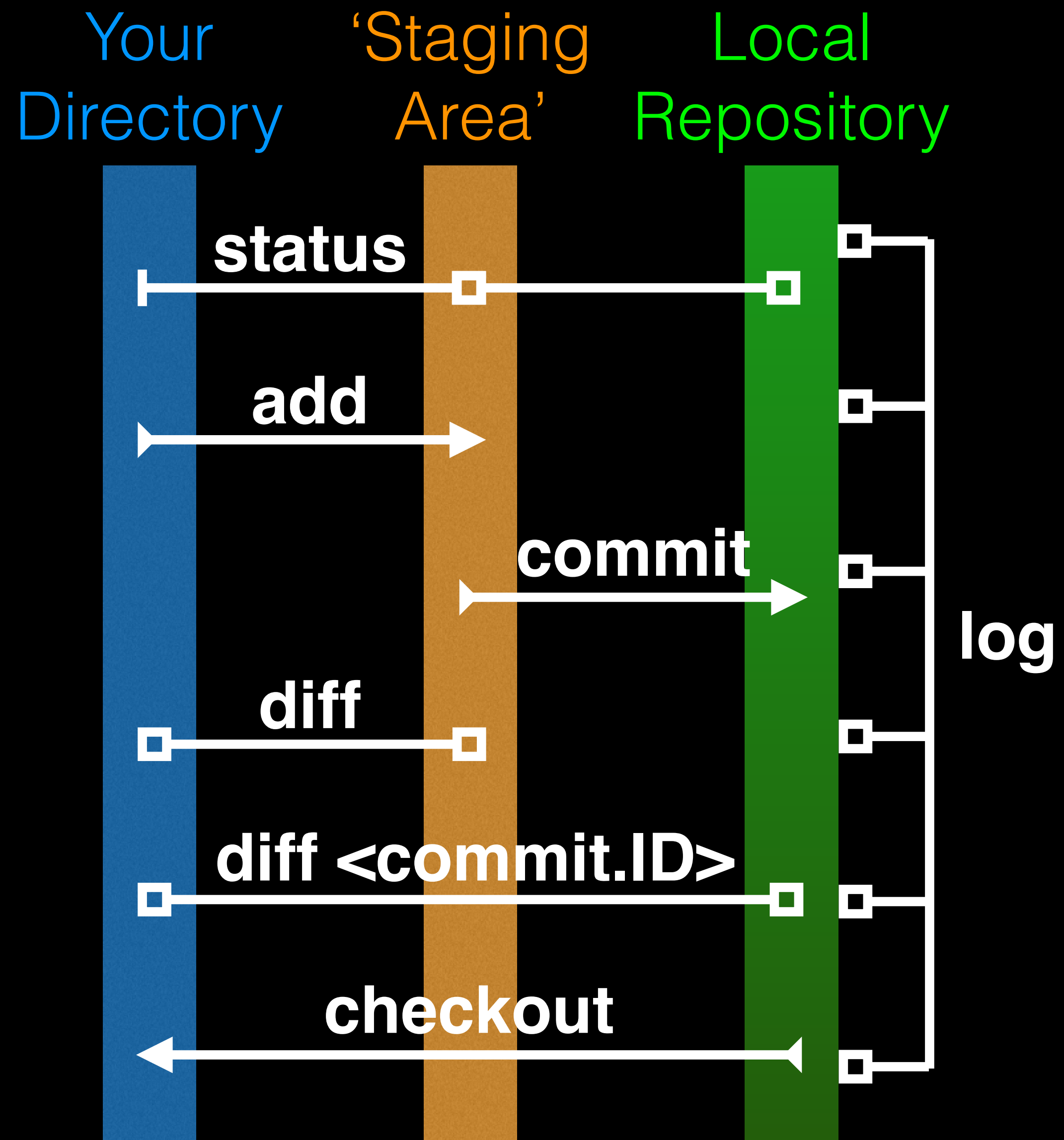
> **git add <filename>** # Tell Git which files to track/stage

> **git commit -m 'Your message'** # Take a content snapshot!

> **git log** # Review your commit history

> **git diff <commit.ID> <commit.ID>** # Inspect content differences

> **git checkout <commit.ID>** # Navigate through the commit history



# git diff: Show changes between commits

```
> git diff 8676 7b67
```

```
# diff --git a/README b/README
# index 73bc85a..67bd82c 100644
# --- a/README
# +++ b/README
# @@ -1,2 @@
# This is a first line of text.
# +This is a 2nd line of text.

# diff --git a/ToDo b/ToDo
# new file mode 100644
# index 0000000..14fbd56
# --- /dev/null
# +++ b/ToDo
# @@ -0,0 +1 @@
# +Learn git basics
```





# git diff: Show changes between commits

```
> git diff 7b67 8676
```

```
# diff --git a/README b/README
# index 67bd82c..73bc85a 100644
# --- a/README
# +++ b/README
# @@ -1,2 +1 @@
# This is a first line of text.
# -This is a 2nd line of text.

# diff --git a/ToDo b/ToDo
# deleted file mode 100644
# index 14fbd56..0000000
# --- a/ToDo
# +++ /dev/null
# @@ 1 +0,0 @@
# -Learn git basics
```

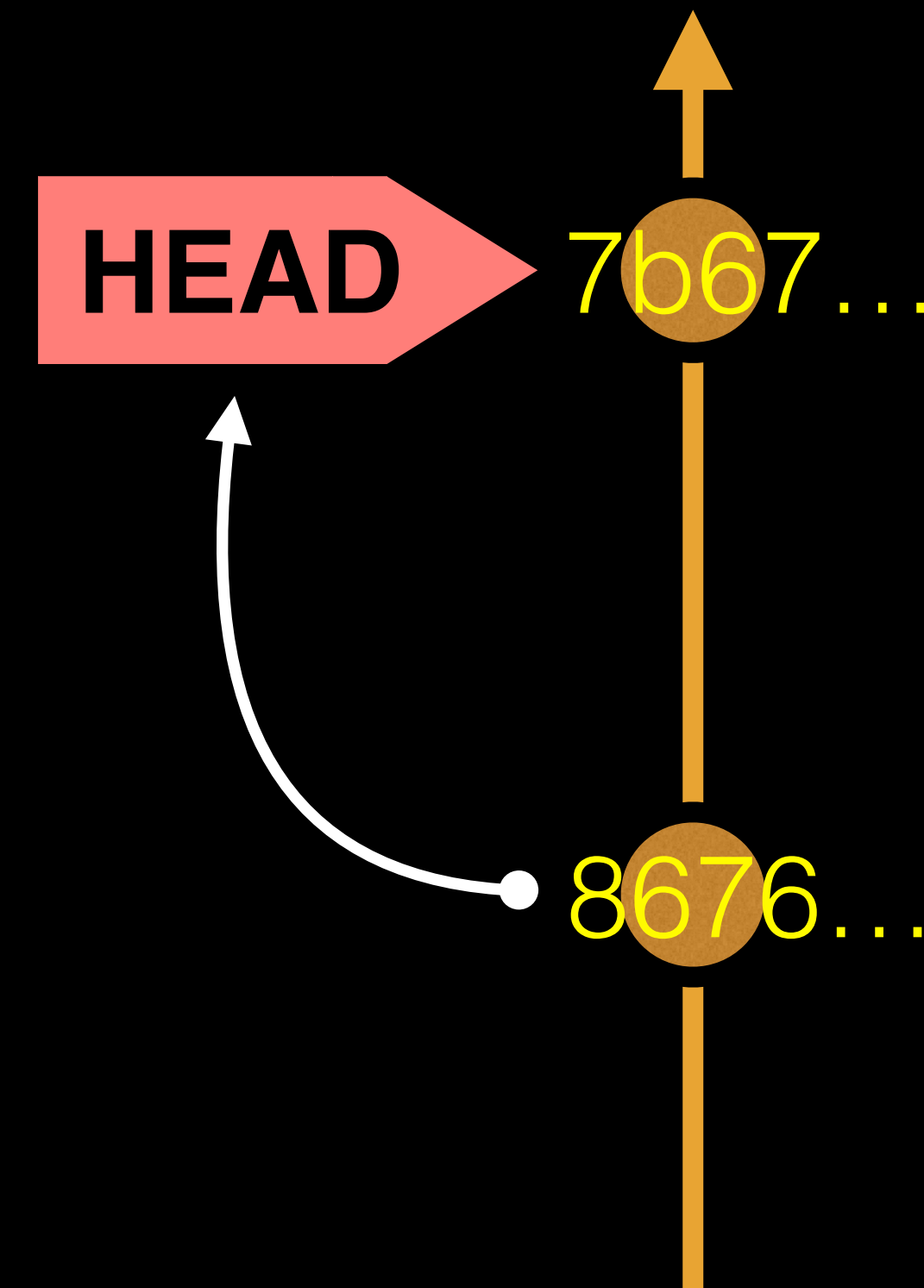


# git diff: Show changes between commits

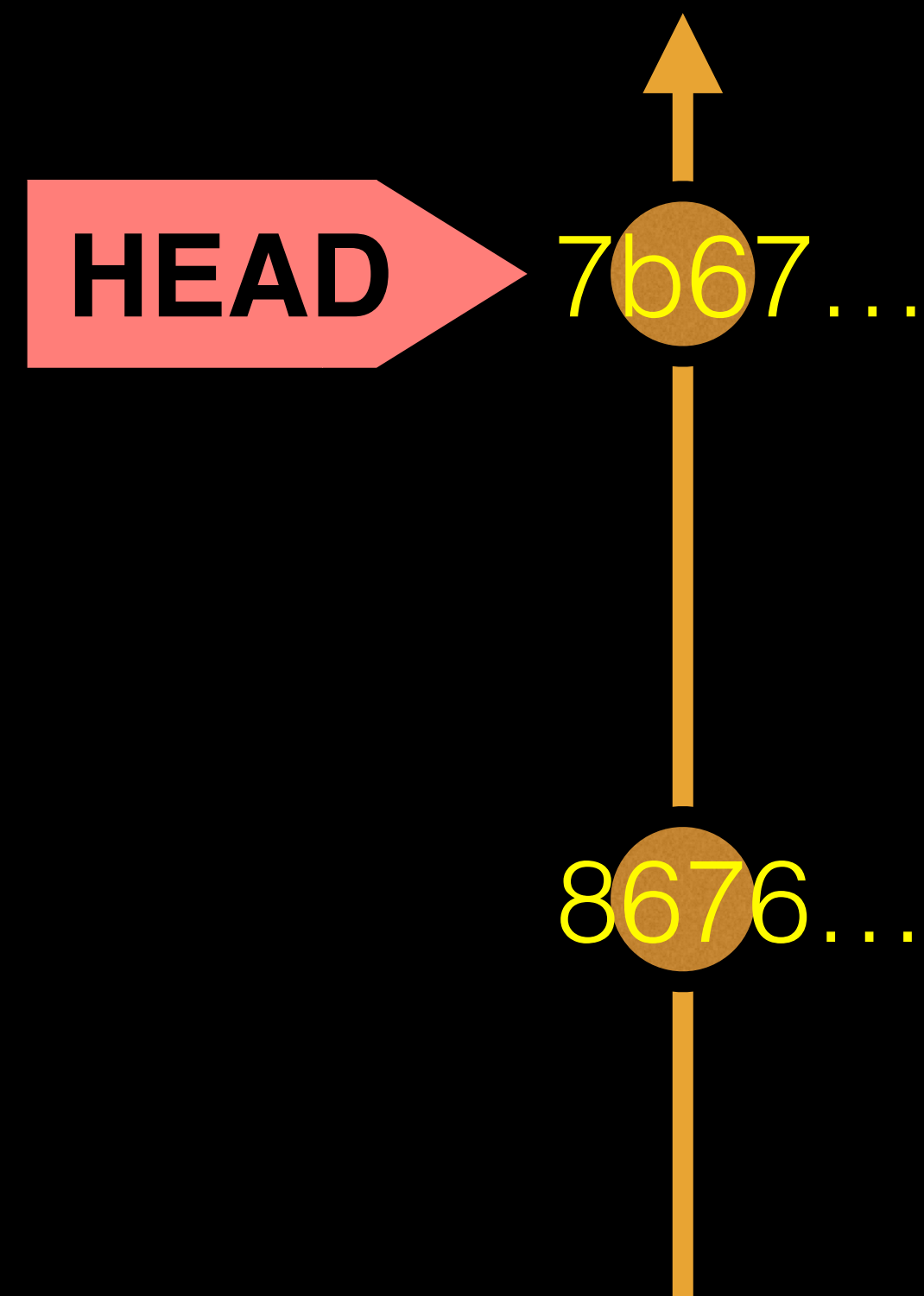
> **git diff 8676** **## Difference to current HEAD position!**

```
# diff --git a/README b/README
# index 73bc85a..67bd82c 100644
# --- a/README
# +++ b/README
# @@ -1 +1,2 @@
# This is a first line of text.
# +This is a 2nd line of text.

# diff --git a/ToDo b/ToDo
# new file mode 100644
# index 0000000..14fbd56
# --- /dev/null
# +++ b/ToDo
# @@ -0,0 +1 @@
# +Learn git basics
```



# HEAD advances automatically with each new commit



To move **HEAD** (back or forward) on the Git graph (and retrieve the associated snapshot content) we can use the command:

```
> git checkout <commit.ID>
```

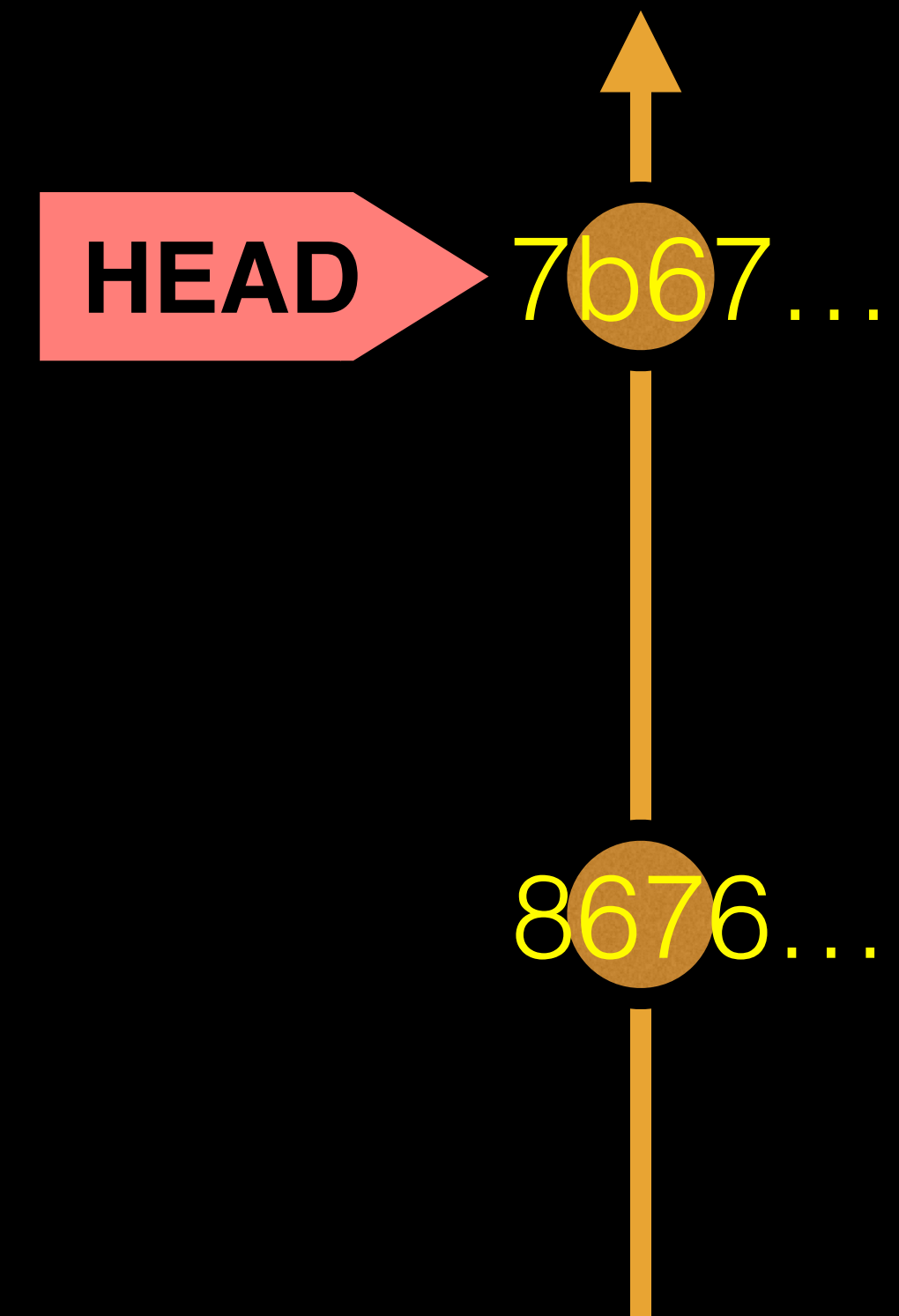
# git checkout: Moves HEAD

> **more README**

This is a first line of text.  
This is a 2nd line of text.

> **git log --oneline**

# 7b679fa Add ToDo and finished README  
# 8676840 Create a README file



# git checkout: Moves HEAD (e.g. back in time)

> **more README**

This is a first line of text.  
This is a 2nd line of text.

> **git log --oneline**

# 7b679fa Add ToDo and finished README  
# 8676840 Create a README file

> **git checkout 86768**

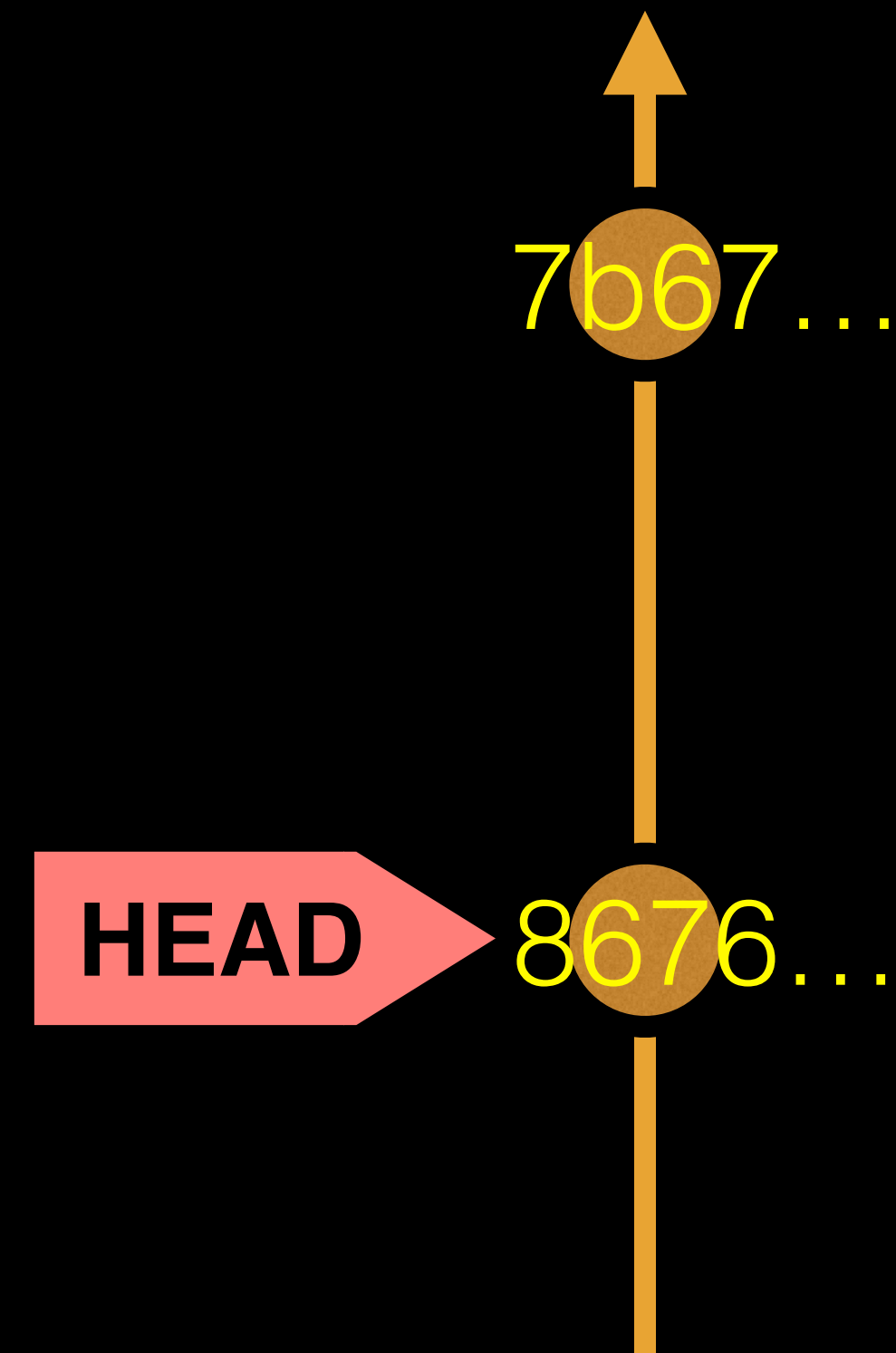
# You are in 'detached HEAD' state...<cut>...  
# HEAD is now at 8676840... Create a README file

> **more README**

This is a first line of text.

> **git log --oneline**

# 8676840 Create a README file



# git checkout: Moves HEAD (e.g. back to the future!)

> **git checkout master**

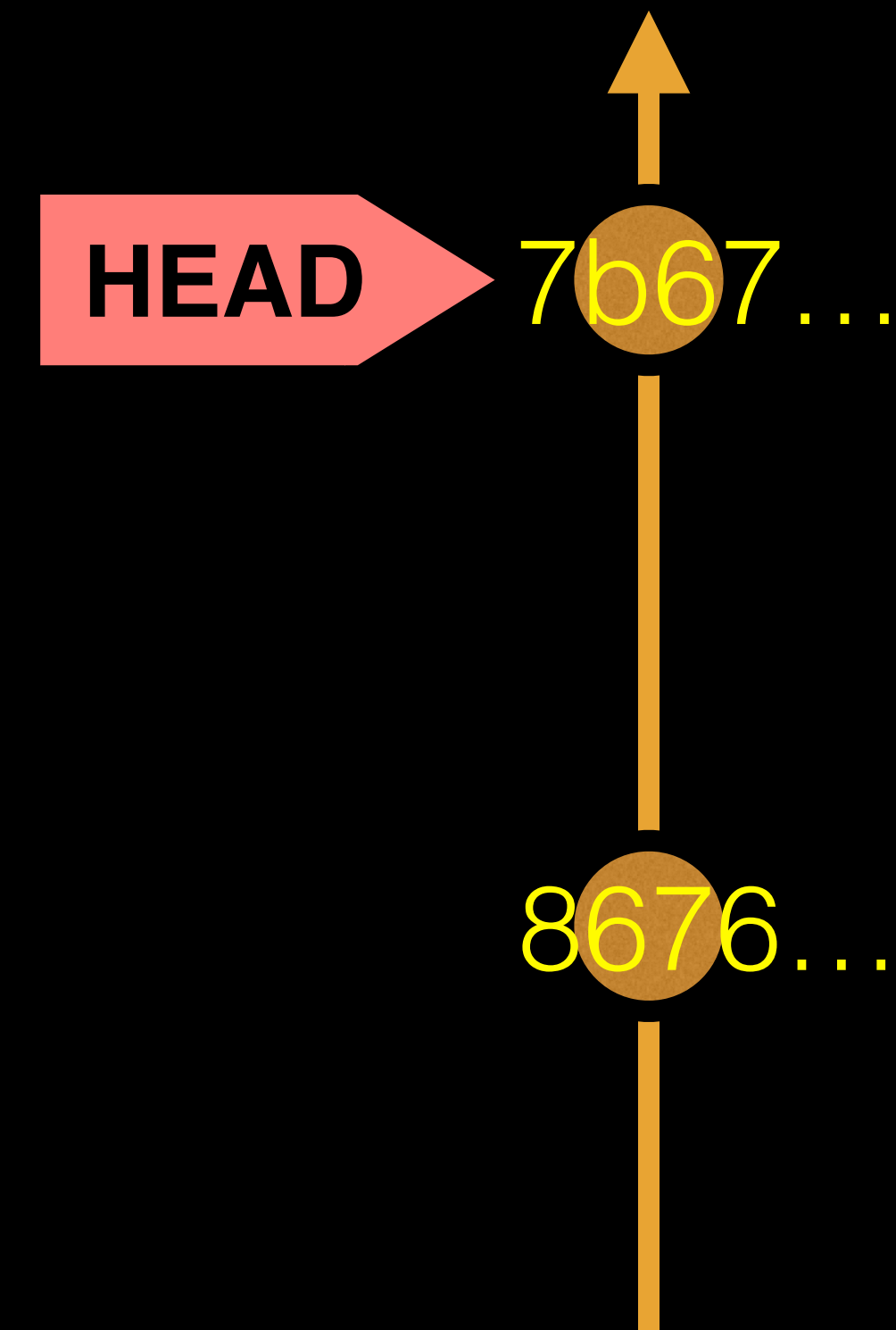
```
# Previous HEAD position was 8676840... Create a README file  
# Switched to branch 'master'
```

> **git log --oneline**

```
# 7b679fa Add ToDo and finished README  
# 8676840 Create a README file
```

> **more README**

```
This is a first line of text.  
This is a 2nd line of text.
```



## Side-Note: There are two\* main ways to use **git checkout**

- Checking out a **commit** makes the entire working directory match that commit. This can be used to view an old state of your project.

```
> git checkout <commit.ID>
```

- Checking out a **specific file** lets you see an old version of that particular file, leaving the rest of your working directory untouched.

```
> git checkout <commit.ID> <filename>
```

# You can discard revisions with **git revert**

- The **git revert** command undoes a committed snapshot.
- But, instead of removing the commit from the project history, it figures out how to **undo the changes** introduced by the commit and **appends a new commit** with the resulting content.

```
> git revert <commit.ID>
```

- This prevents Git from losing history!



# Removing untracked files with **git clean**

- The **git clean** command removes untracked files from your working directory.
- Like an ordinary **rm** command, **git clean** is not undoable, so make sure you really want to delete the untracked files before you run it.
  - > `git clean -n` # dry run display of files to be 'cleaned'
  - > `git clean -f` # remove untracked files

# GUIs

**Tower** (Mac only)

**GitHub\_Desktop** (Mac, Windows)

**SourceTree** (Mac, Windows)

**SmartGit** (Linux)

**RStudio**

<https://git-scm.com/downloads/guis>