Extra 3 classes on Git and Github

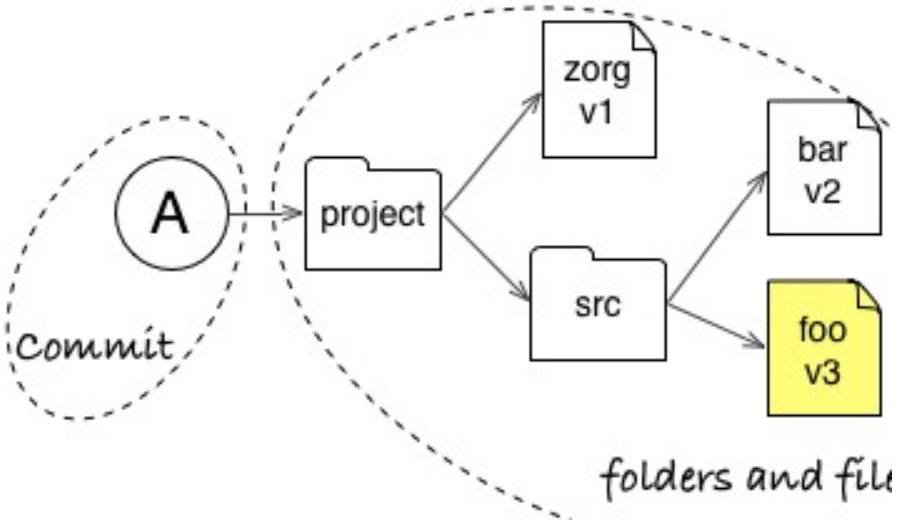
Module 7_extra

Core concepts in Git

Module 7-2

Core Concepts

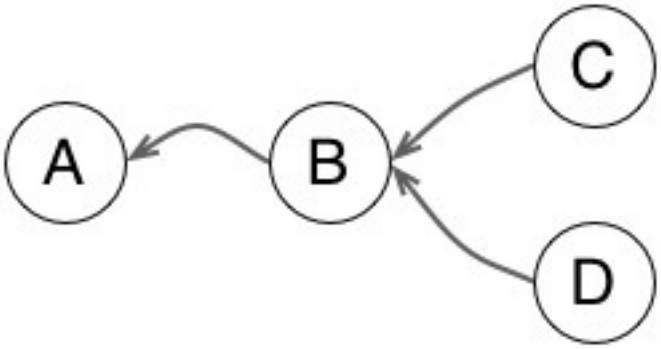
Git stores snapshots
(commits) of your repository



http://sogilis.com/blog/demystifying-git-concepts-to-understand/

Core Concepts

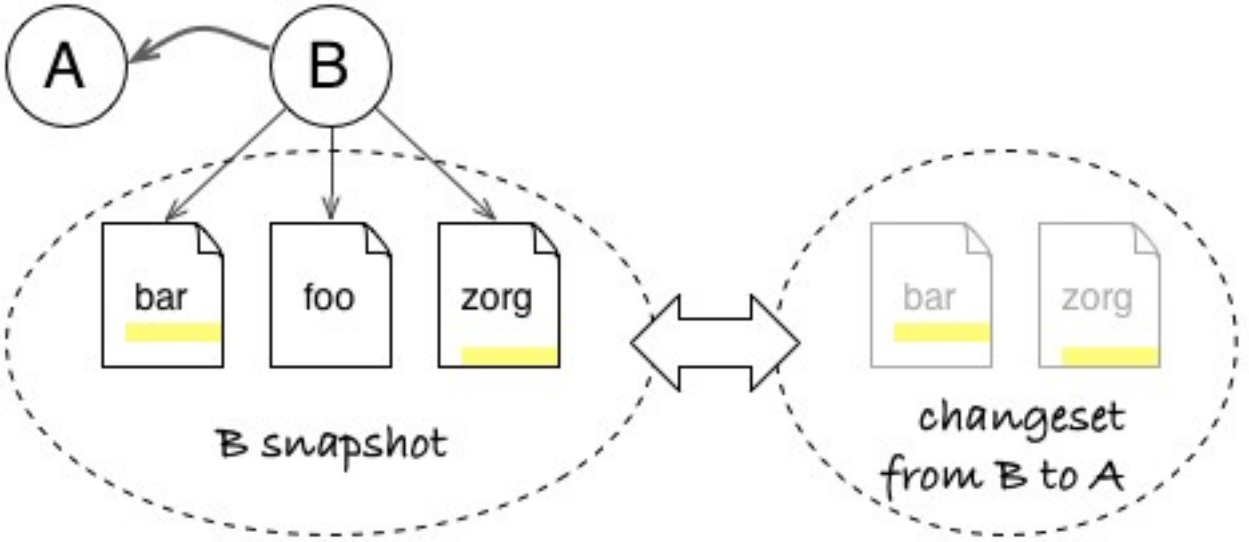
Git represents relationships between commits as a graph



http://sogilis.com/blog/demystifying-git-concepts-to-understand/

Core Concepts

Git can compute changesets between any two commits of your project



http://sogilis.com/blog/demystifying-git-concepts-to-understand/

Core Concepts

Git sees changes at the level of lines in a text file

Fix error in boxplot labelling

master

robschick committed on Jan 20 — 1 parent da9140b

Showing **1 changed file** with **1 addition** and **1 deletion**.

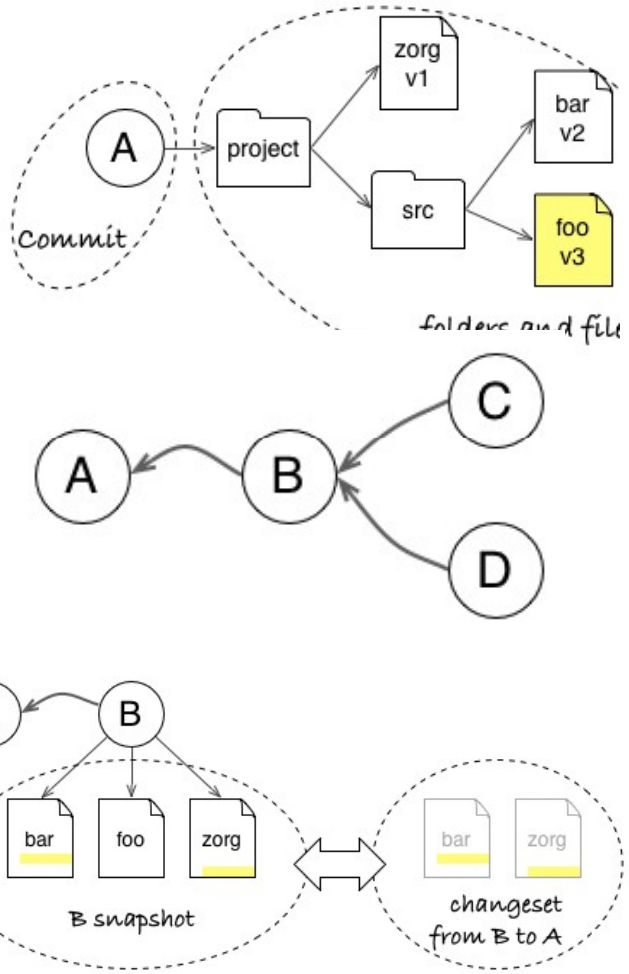
2 R/plotBoxplotHealth.R

```
@@ -35,7 +35,7 @@ plotBoxplotHealth <- function(dfLong, bsize, cval = 4){
35 35            nvals$Freq[nvals$Var1 == 'NonRepFem' & nvals$Var2 == 2],
36 36            nvals$Freq[nvals$Var1 == 'RepFem' & nvals$Var2 == 2],
37 37            nvals$Freq[nvals$Var1 == 'NonRepFem' & nvals$Var2 == 3],
38    -          nvals$Freq[nvals$Var1 == 'RepFem' & nvals$Var2 == 2],
   38 +          nvals$Freq[nvals$Var1 == 'RepFem' & nvals$Var2 == 3],
39 39            nvals$Freq[nvals$Var1 == 'NonRepFem' & nvals$Var2 == 1],
40 40            nvals$Freq[nvals$Var1 == 'RepFem' & nvals$Var2 == 1])
41 41
```

http://sogilis.com/blog/demystifying-git-concepts-to-understand/

3 Core Concepts

- Snapshot
- Graph
- Changeset



States of a git Repository

The Repository

- Collection of files managed by git
- History (all of it)
- Encompassing file on the Operating System is considered the working directory
  - Can include files managed by git
  - Files ignored by git
  - Files not yet managed by git
- Quasi-hidden **.git** folder
- Since the repo contains all the history, keep the repos narrowly focused

Three Local States

Working Directory

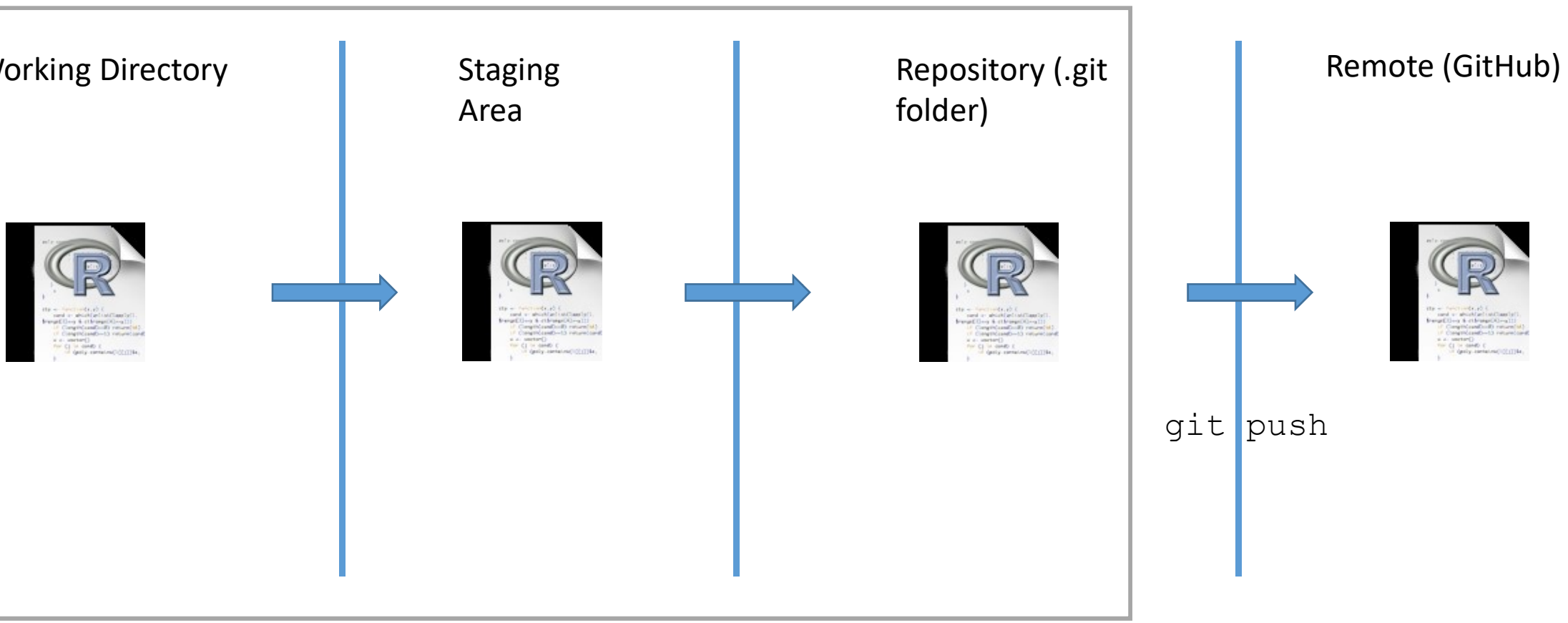
`git add`

Staging Area

`git commit`

Repository (.git folder)

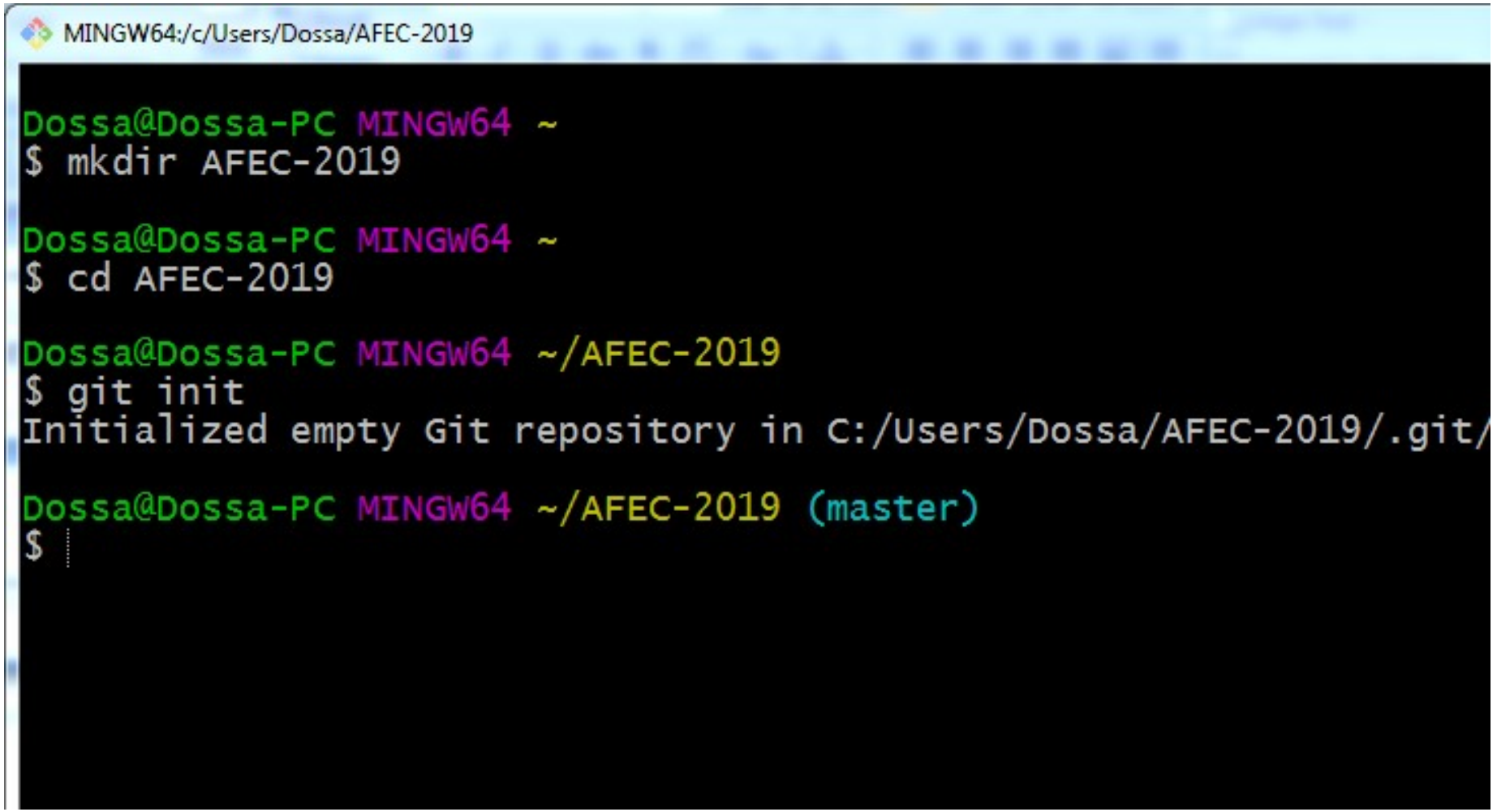
Three Local States with Remote



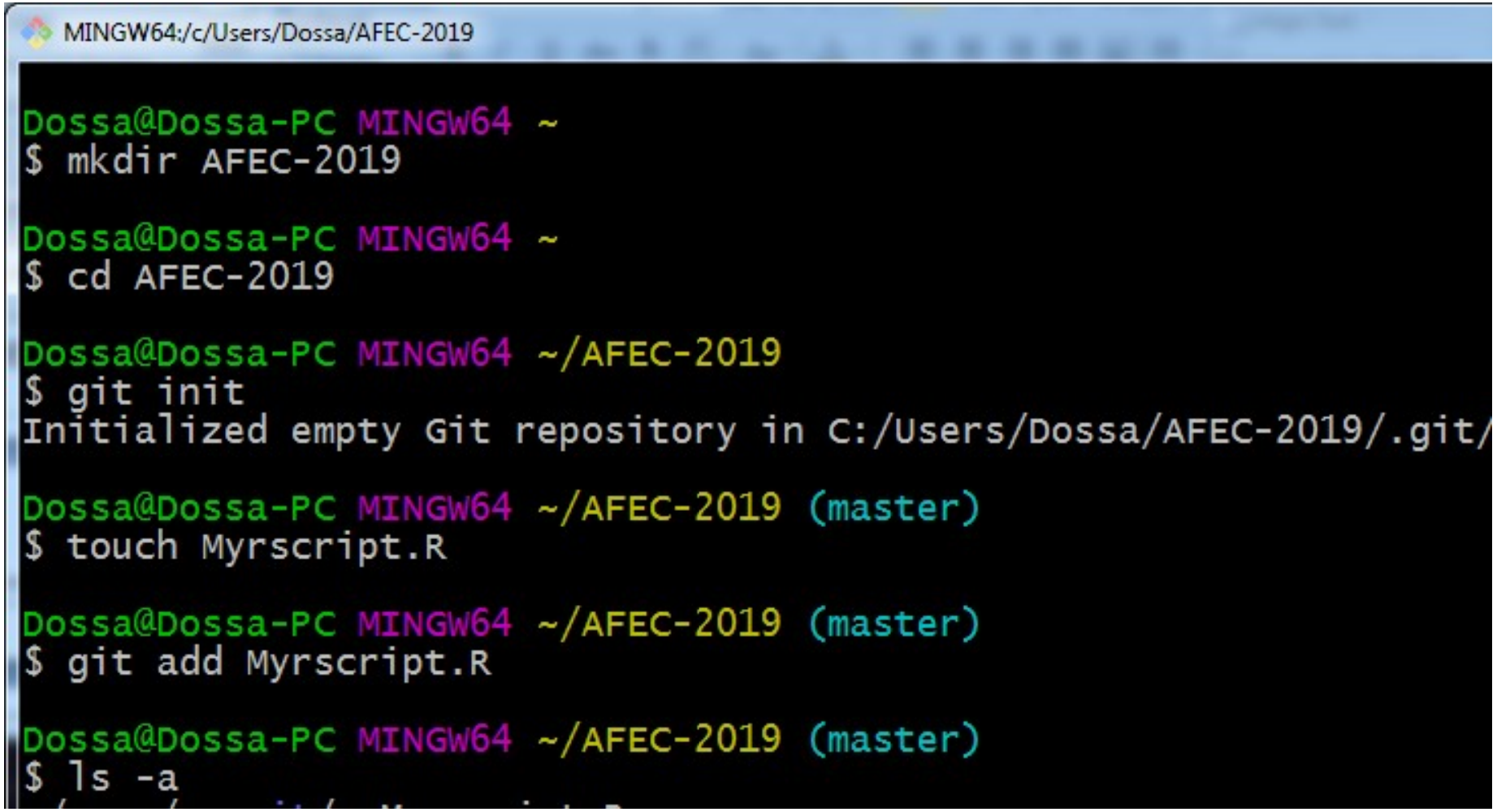
Basic Commands

Mastering a Basic Workflow

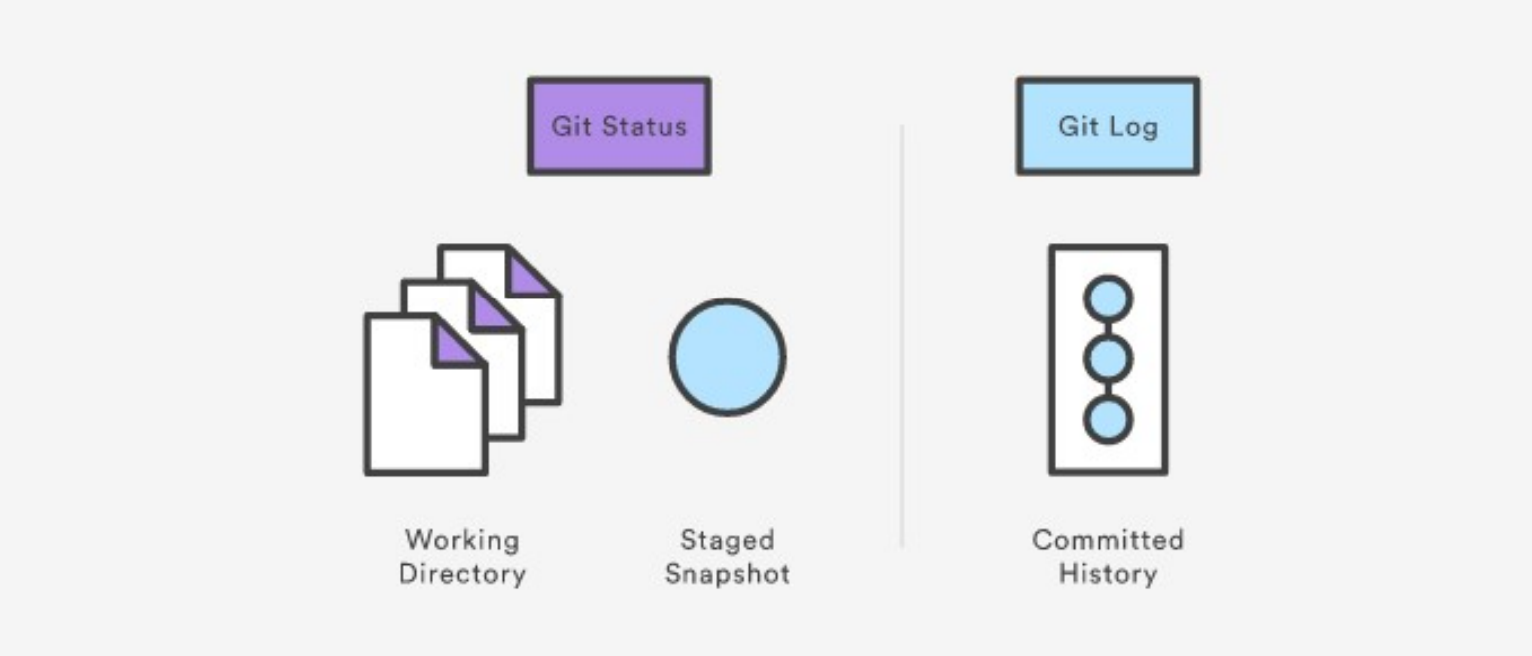
`git init` – Initialize an Empty Repo



`git add` – Add a Document to the Staging Area

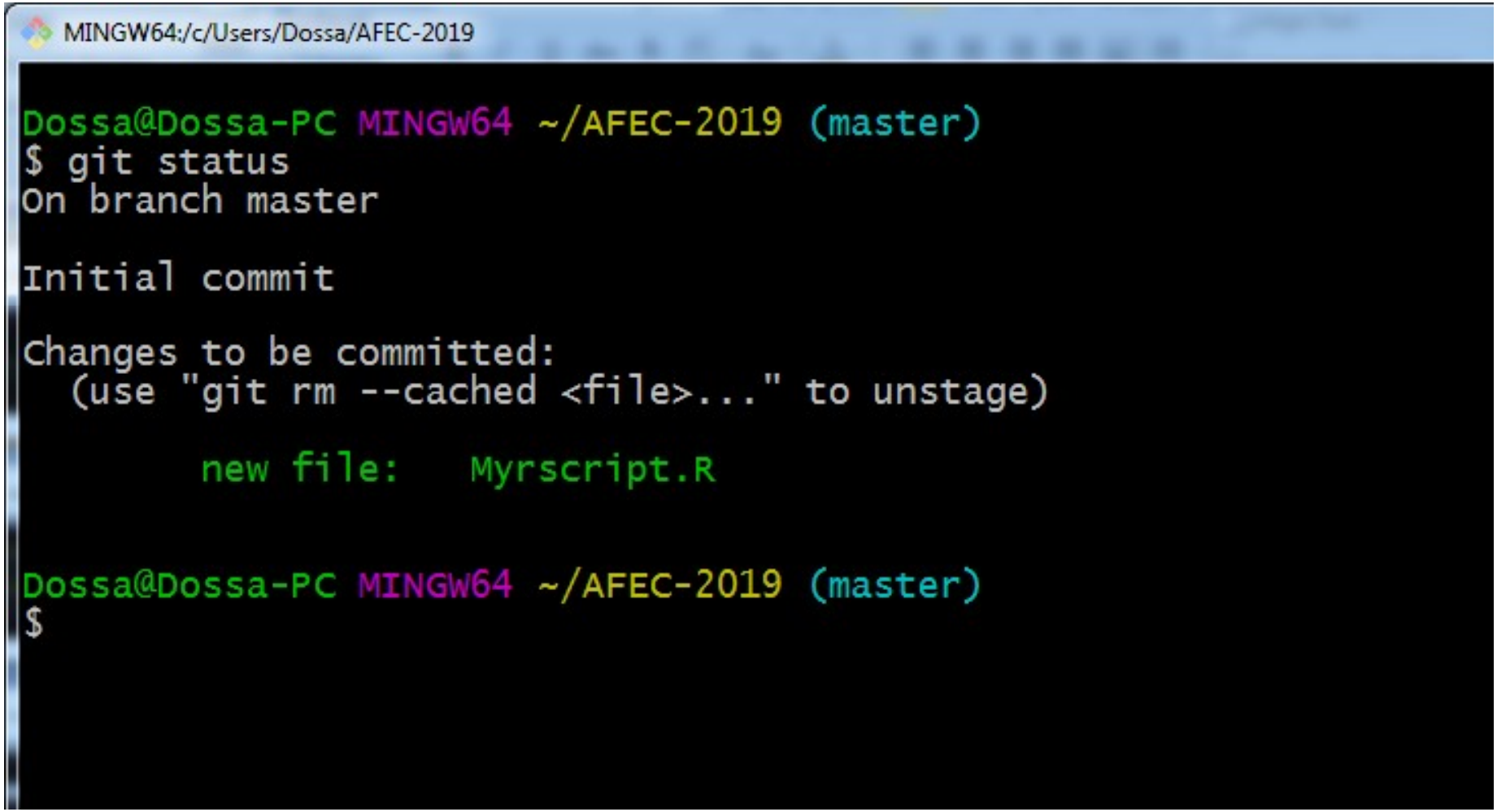


Viewing the Repo

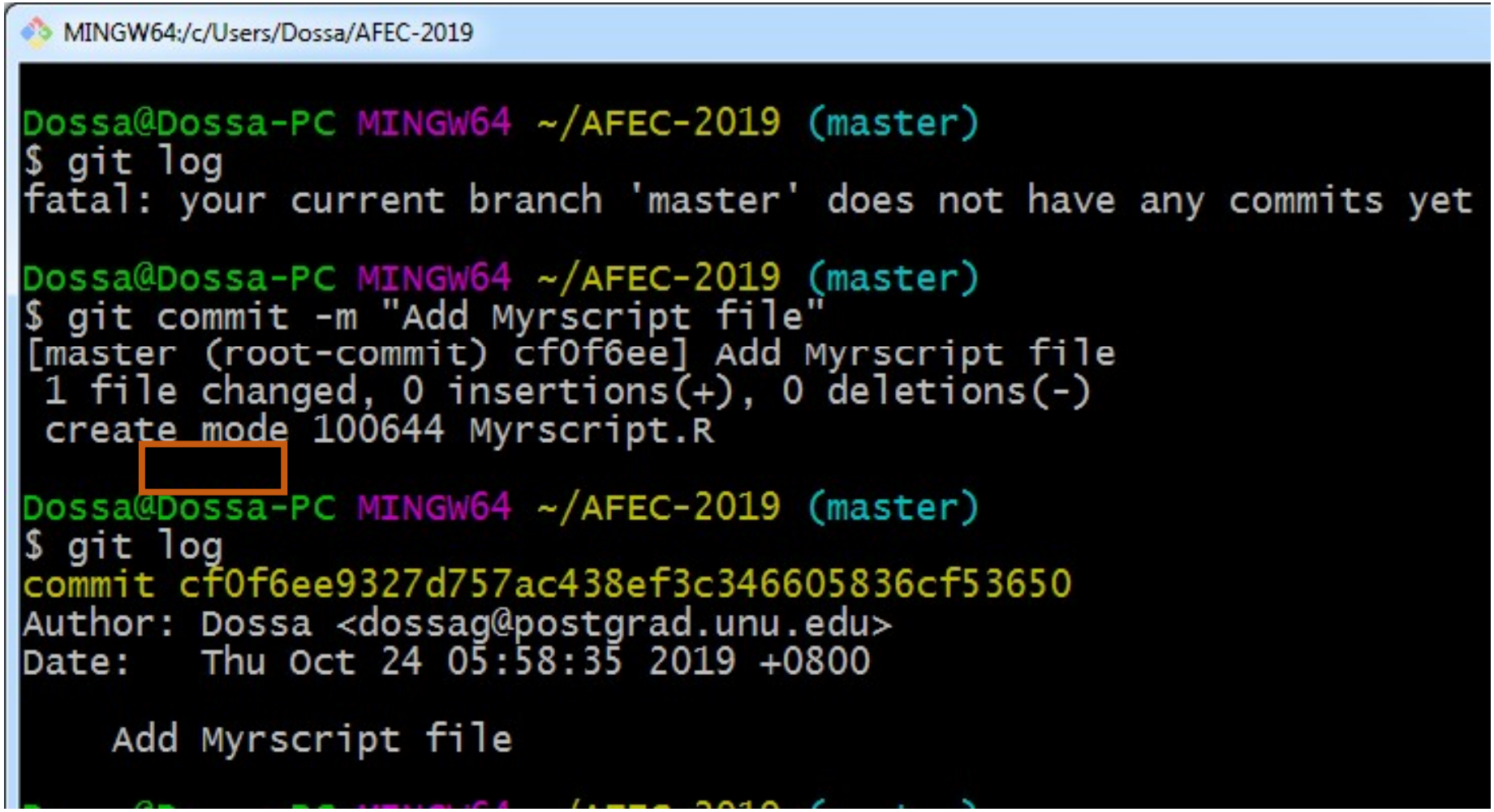


https://www.atlassian.com/git/tutorials/inspecting-a-repository

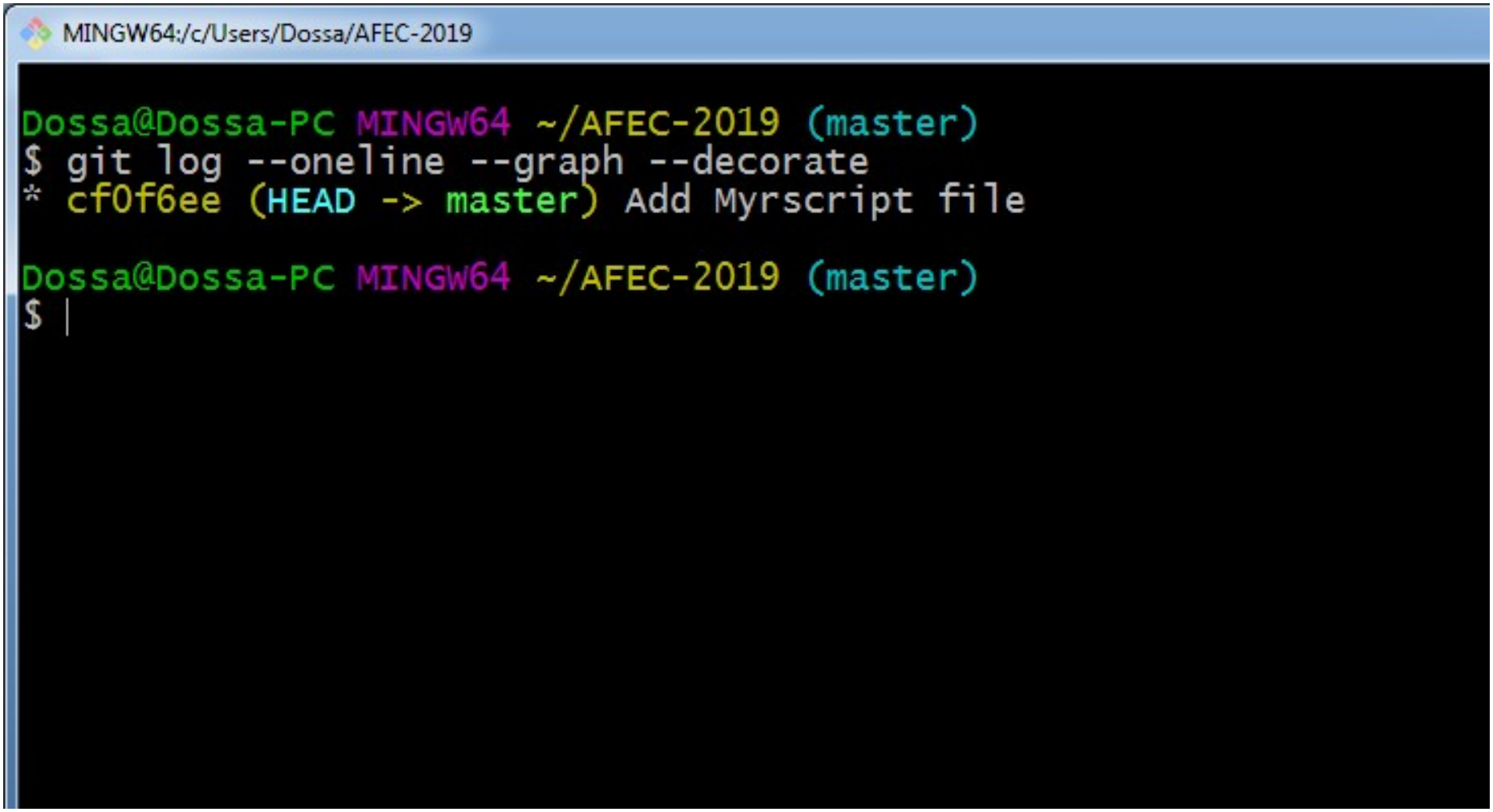
`git status` – What's Happening?



`git log` – To view the history of Repo

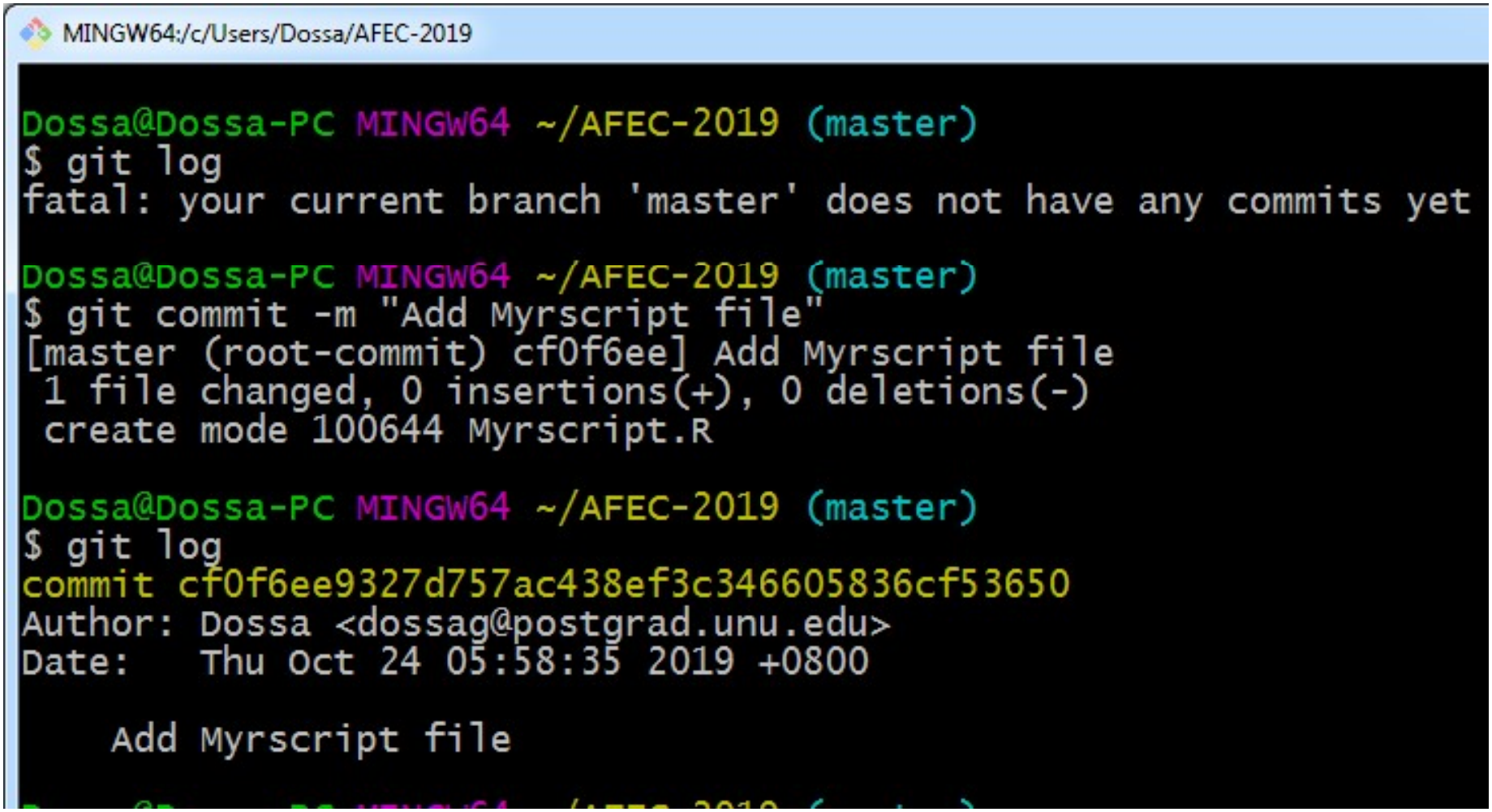


`git log` – With Options

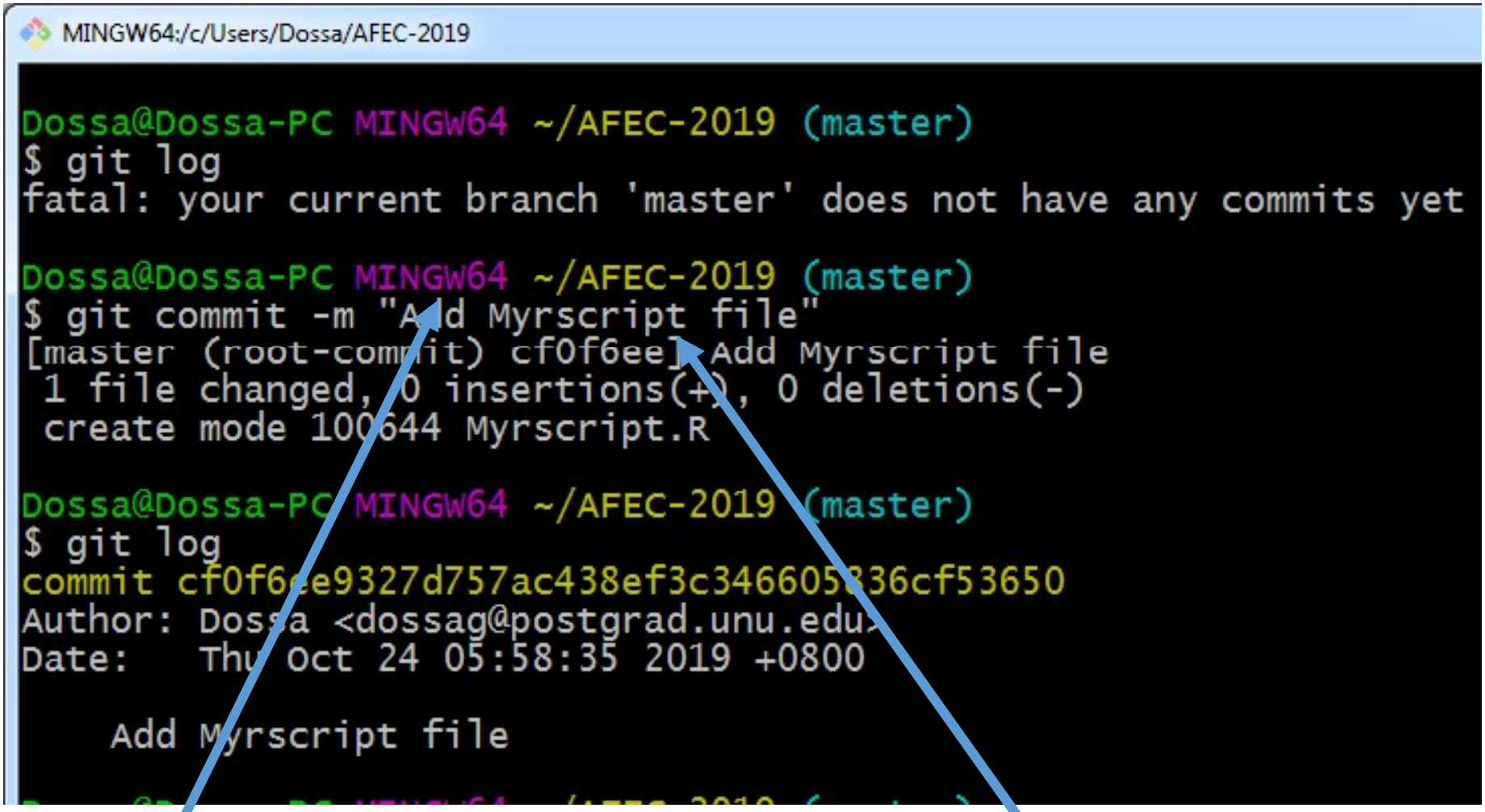


`git commit` – Records changes in the Repo

- `git commit -m "Second check in of my R Script"`



Ok, What Just happened?

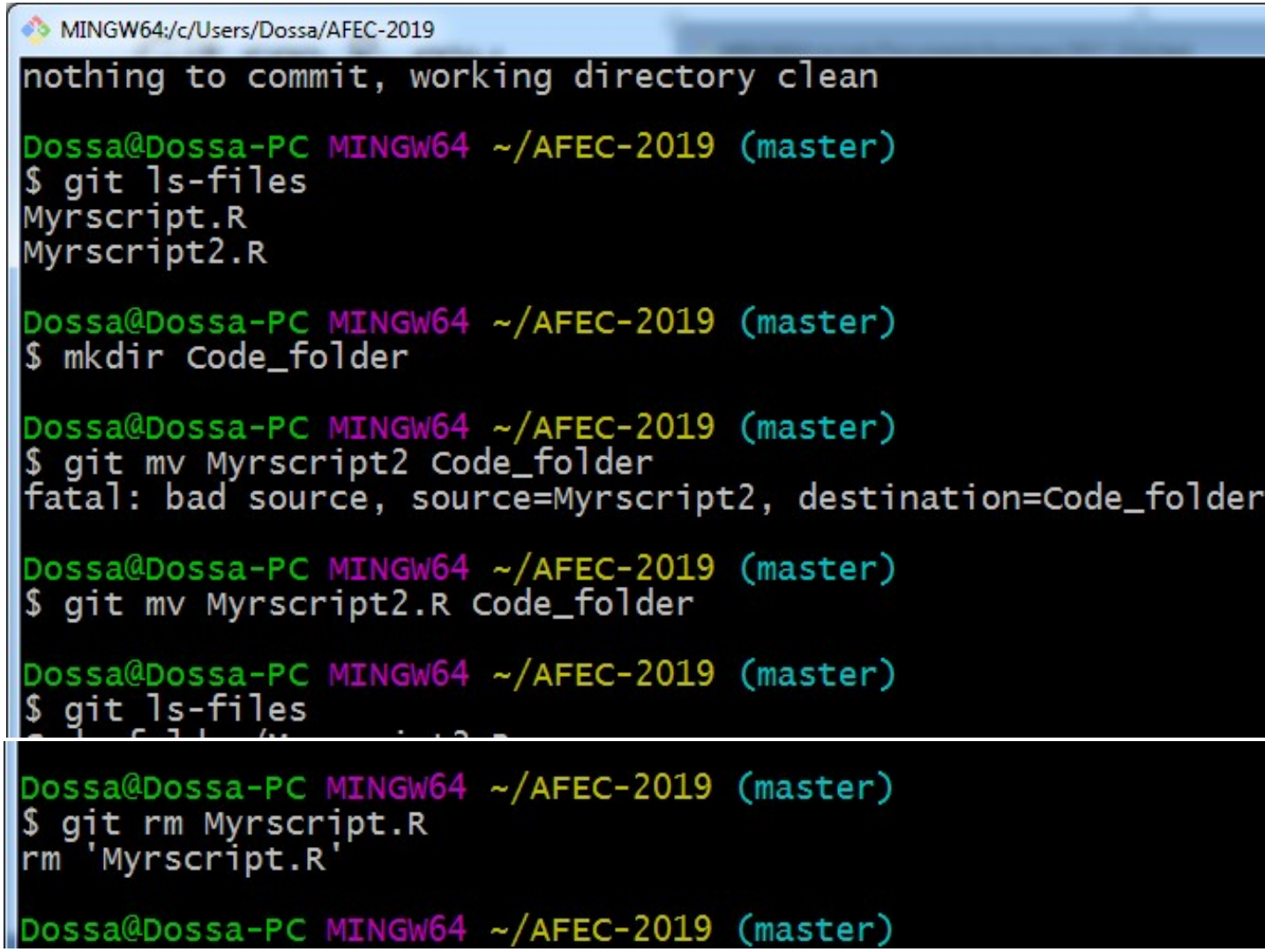


sha-1 commit

What's Changed

Git rm

- Don’t delete or rename tracked files with the OS; use:
- `git rm`
- `git mv`



Git mv

- Don’t delete or rename tracked files with the OS; use:

- `git mv`

Lather, Rinse, Repeat

Good Commit Messages*

- **Be concise, yet evocative.** At a glance, you should be able to see what a commit does. But there should be enough detail so you can remember (and understand) what was done
- **Describe the why, not the what.** Since you can always retrieve the diff associated with the commit, the message doesn't need to say exactly what changed. Instead it should provide a high-level summary that focuses on the reasons for the change

*http://r-pkgs.had.co.nz/git.html#commit-best-practices

Good Commit Messages

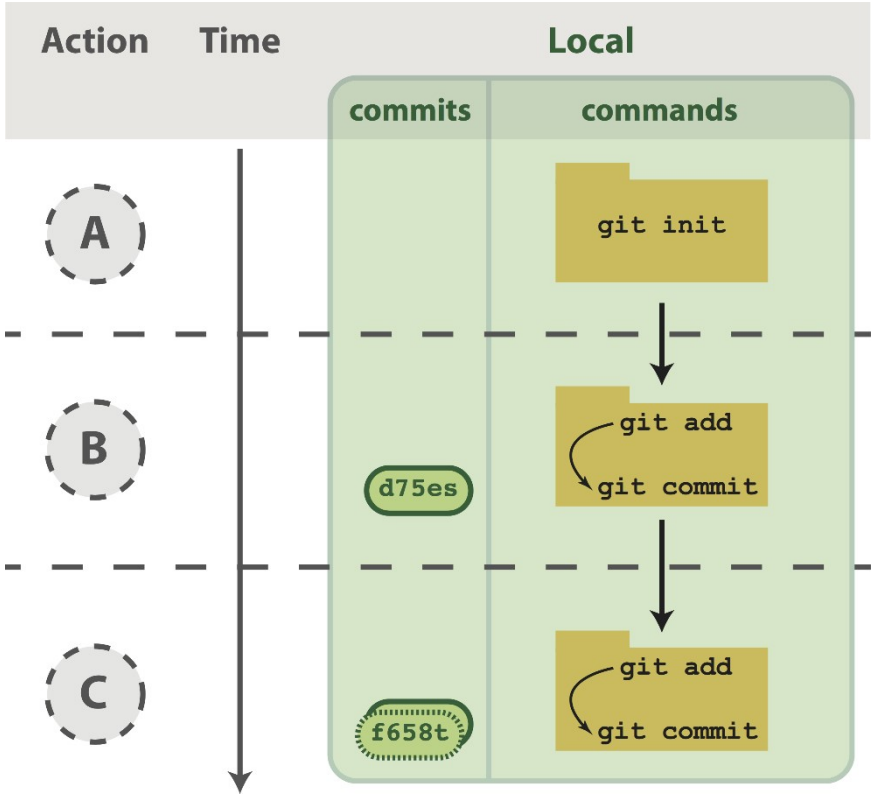
The seven rules of a great Git commit message

> *Keep in mind: This has all been said before.*

1. Separate subject from body with a blank line
2. Limit the subject line to 50 characters
3. Capitalize the subject line
4. Do not end the subject line with a period
5. Use the imperative mood in the subject line
6. Wrap the body at 72 characters
7. Use the body to explain *what* and *why* vs. *how*

https://chris.beams.io/posts/git-commit/

Workflow Visualized



Blischak, John D., Emily R. Davenport, and Greg Wilson. 2016.
"A Quick Introduction to Version Control with Git and GitHub."
PLoS Computational Biology 12 (1): e1004668.

Lifecycle of status

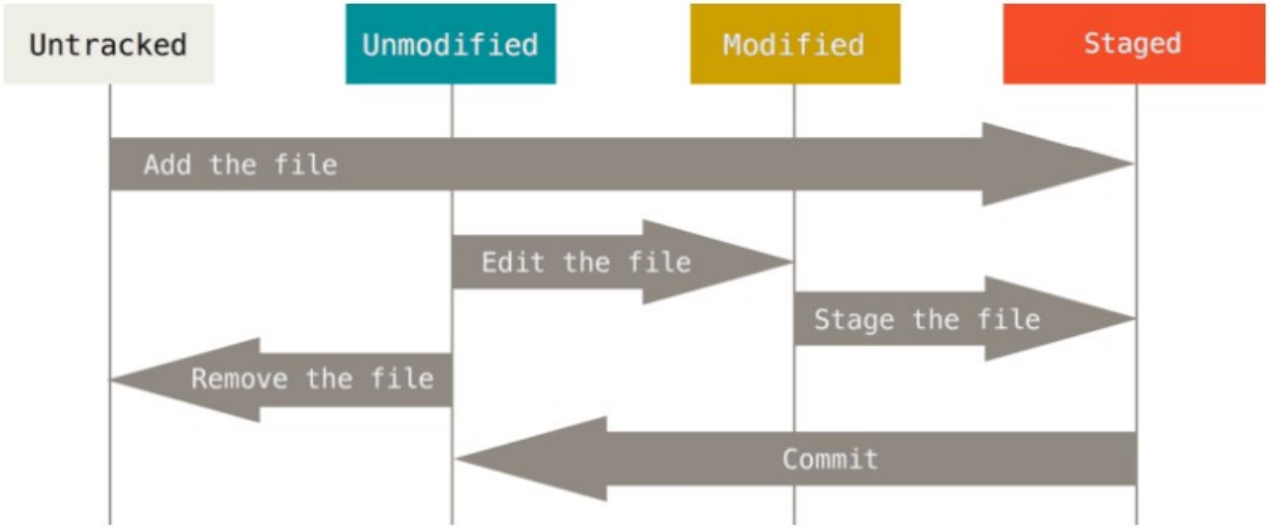
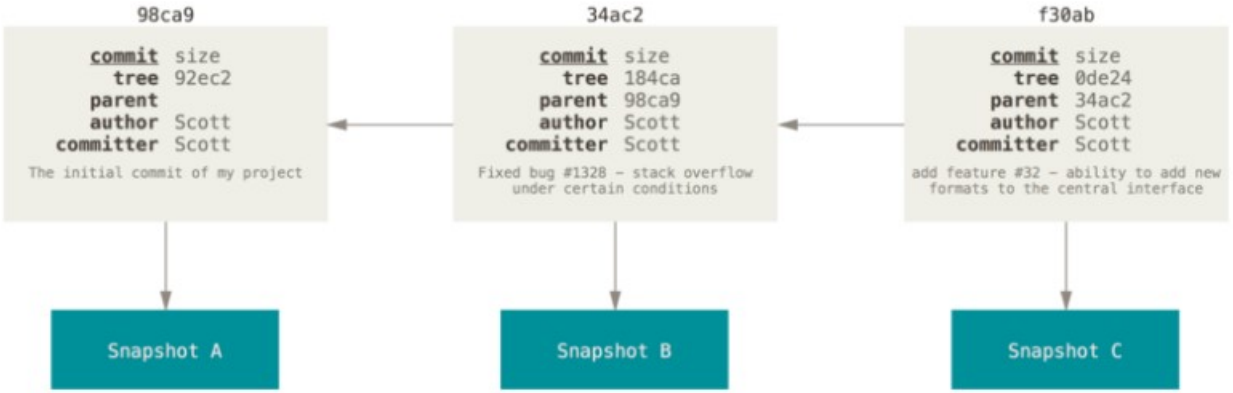


FIGURE 2-1

The lifecycle of the status of your files.

Chacon, S., and B. Straub. 2014. Pro Git. The Expert's Voice. Apress.

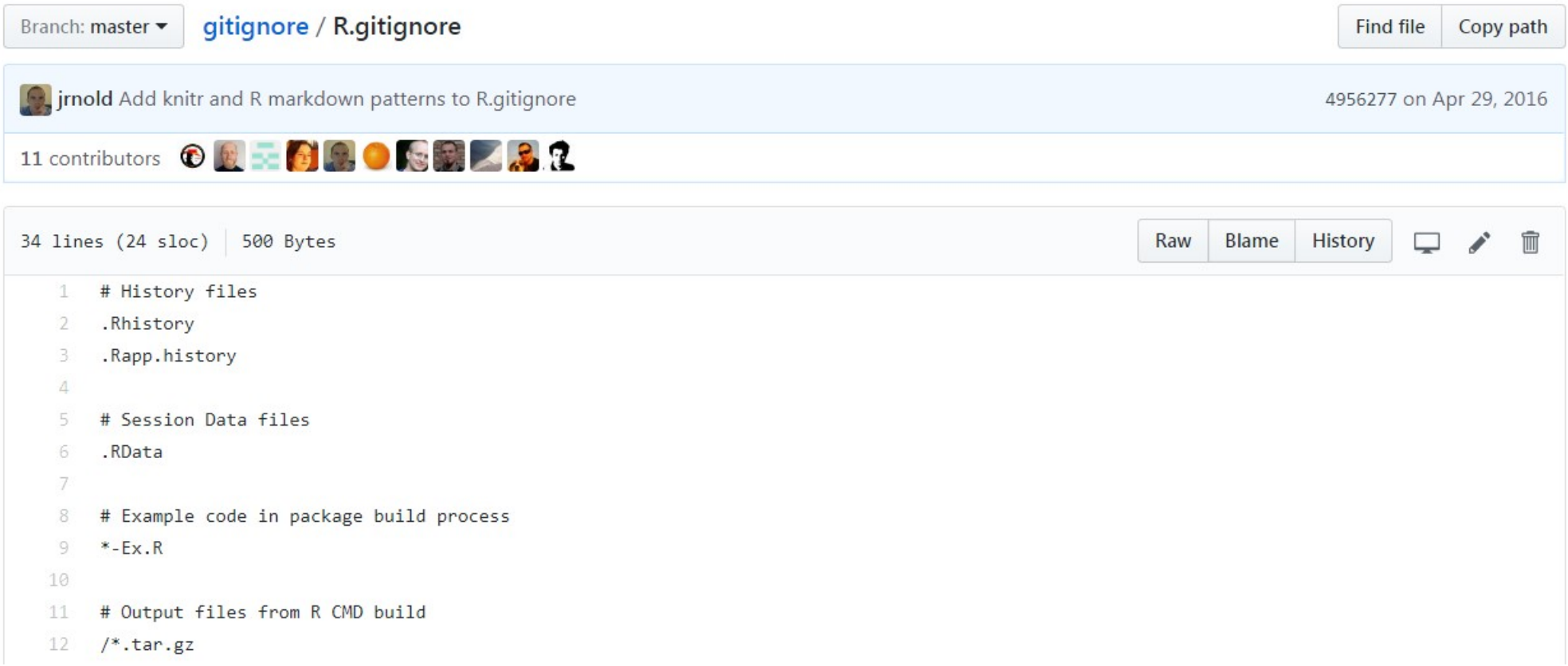
Commit Graph Visualized



A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is `master`. As you start making commits, you're given a `master` branch that points to the last commit you made. Every time you commit, it moves forward automatically.

Chacon, S., and B. Straub. 2014. Pro Git. The Expert's Voice. Apress.

.gitignore

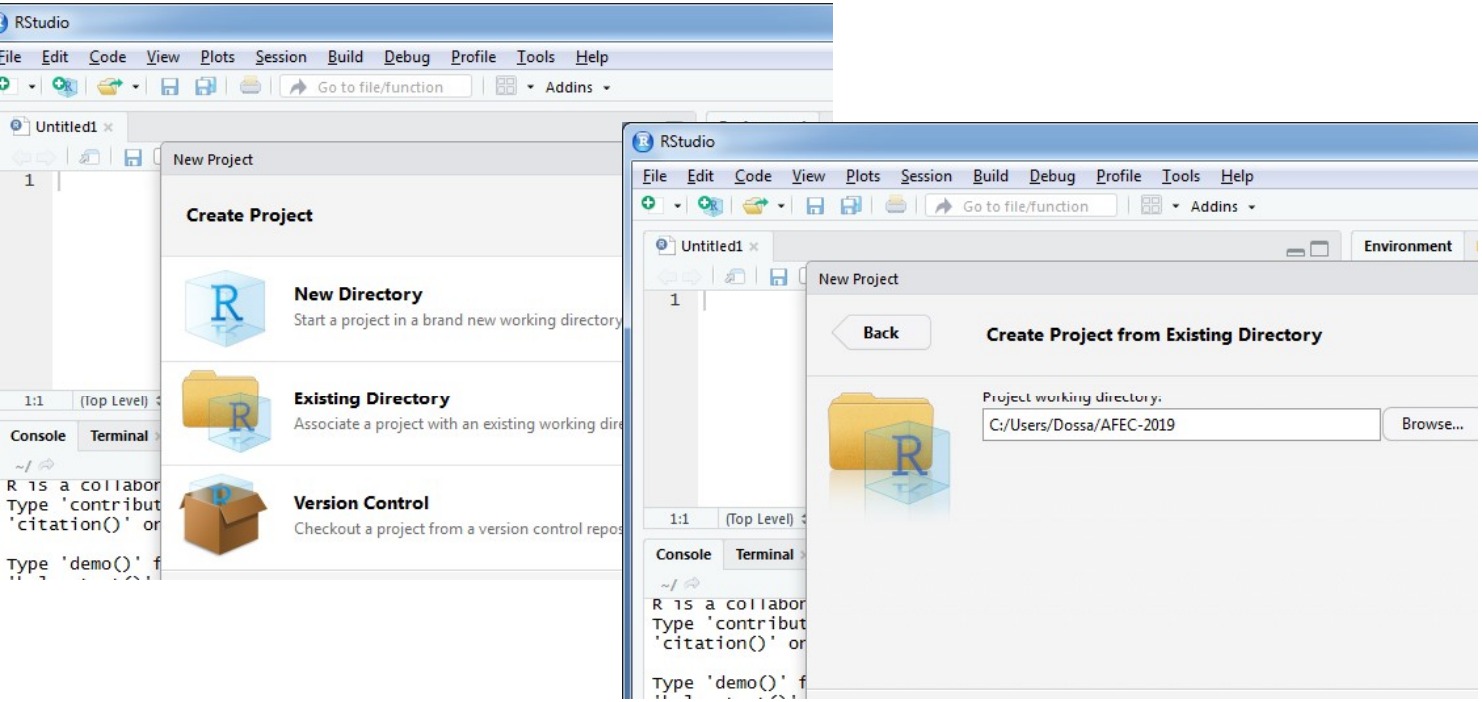

Branch: master
gitignore / R.gitignore
Find file
Copy path
jrnold Add knitr and R markdown patterns to R.gitignore
4956277 on Apr 29, 2016
11 contributors
34 lines (24 sloc)
500 Bytes
Raw
Blame
History
History files
.Rhistory
.Rapp.history
Session Data files
.RData
Example code in package build process
*-Ex.R
Output files from R CMD build
/*.tar.gz

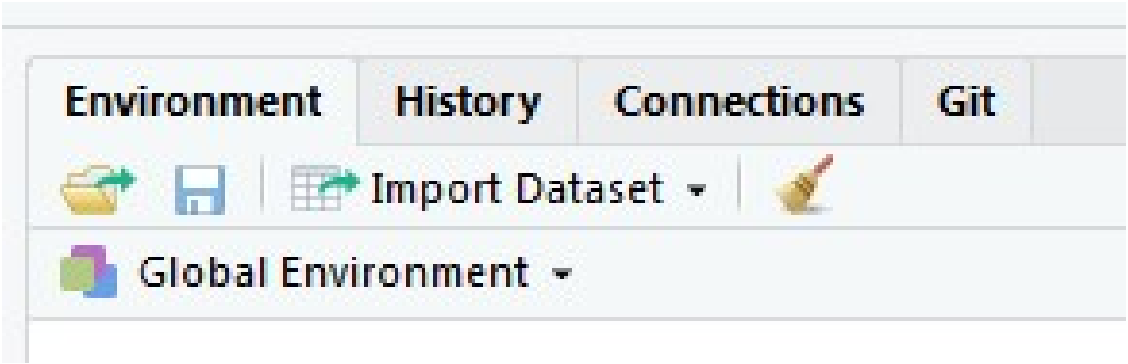
`.gitignore`

- Make the code produce a plot file, say a pdf
- Run the code
- Make a `.gitgnore` file
  - `.pdf`
  - `.Rout`
- Add & commit the `.gitignore`
- Run `ls` (you should see the `pdf` and the `Rout` file)
- Run `git ls-files` (you should *not* see the `pdf` and the `Rout` file)

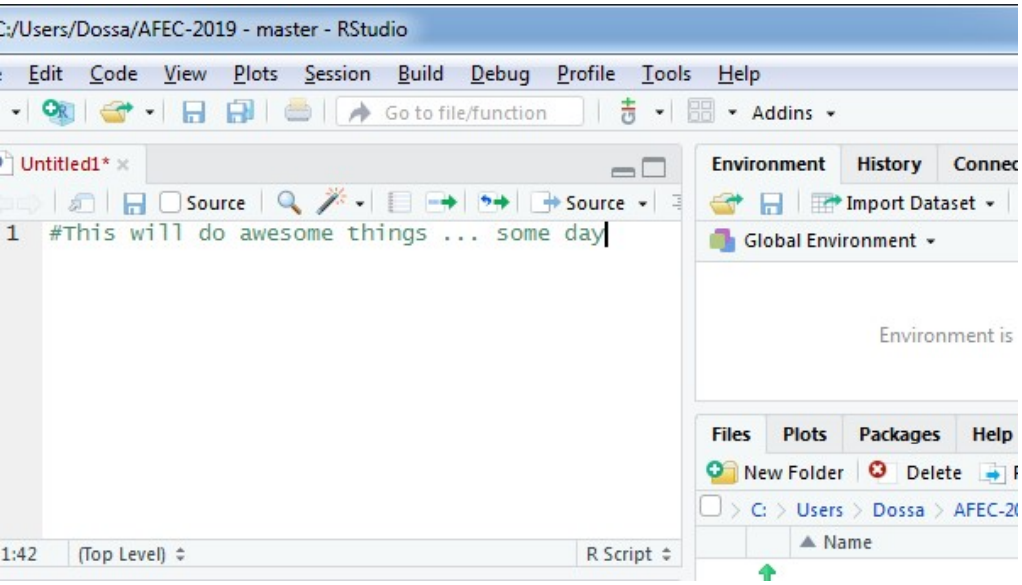
Break Time?

New R Project With Your Existing Repo

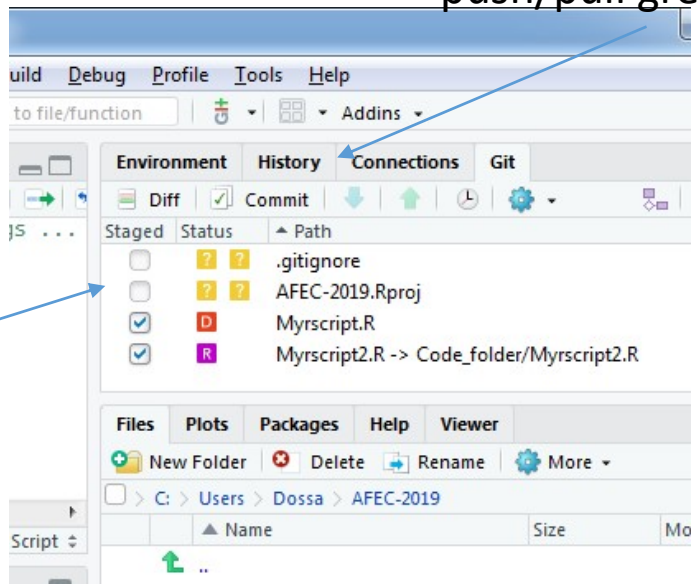




Make the Changes in RStudio

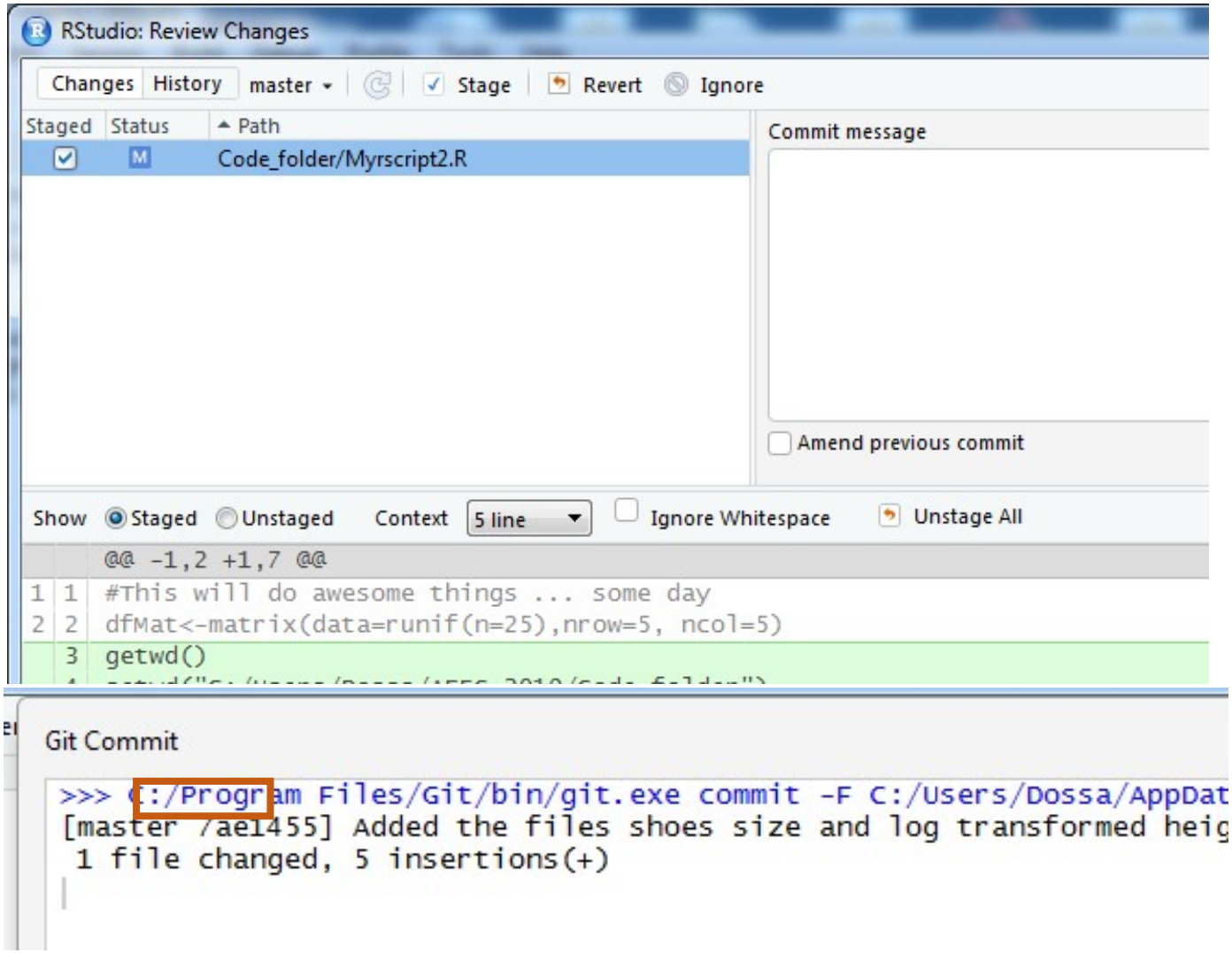


push/pull greyed out

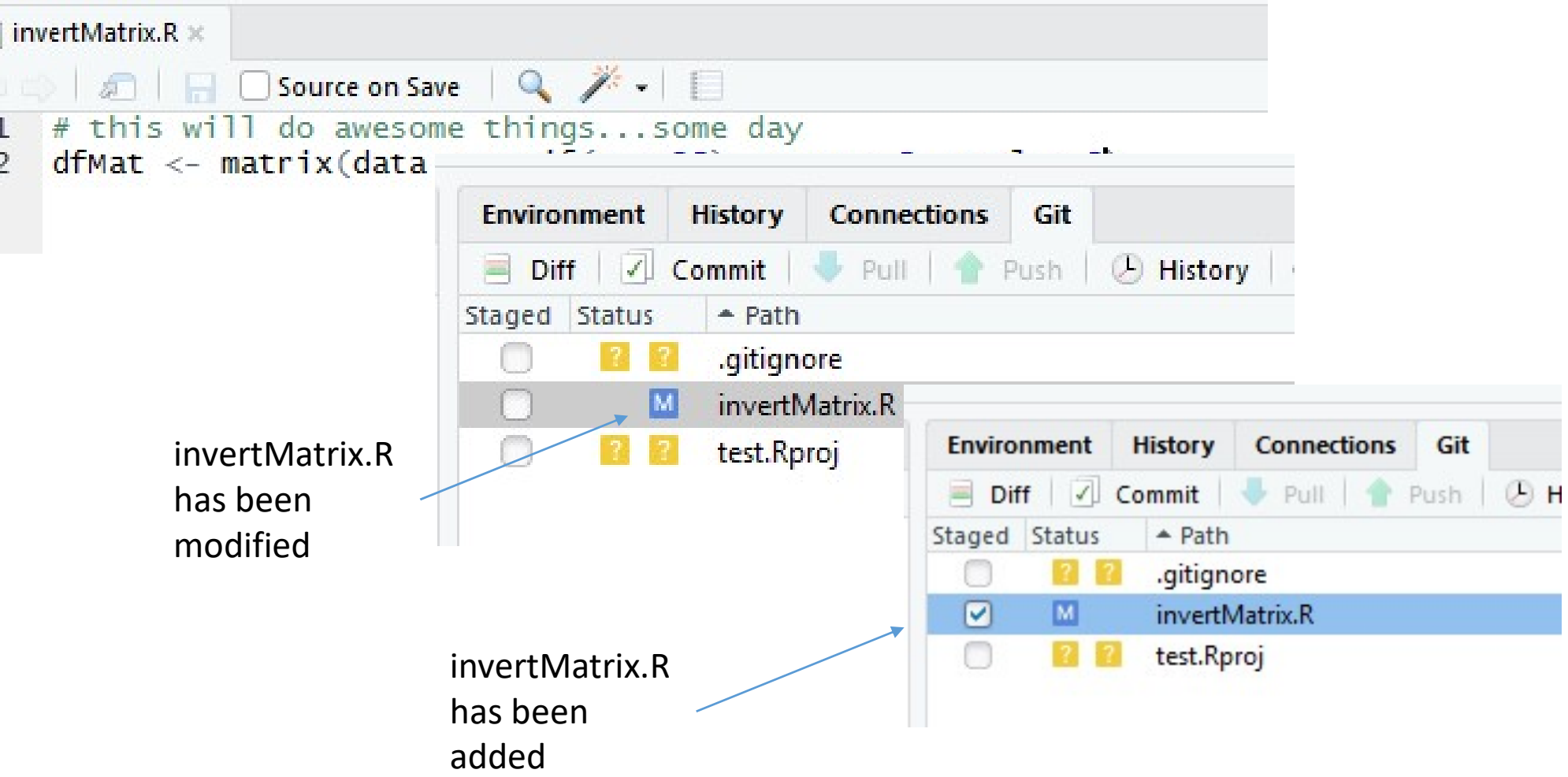


Add / made changes

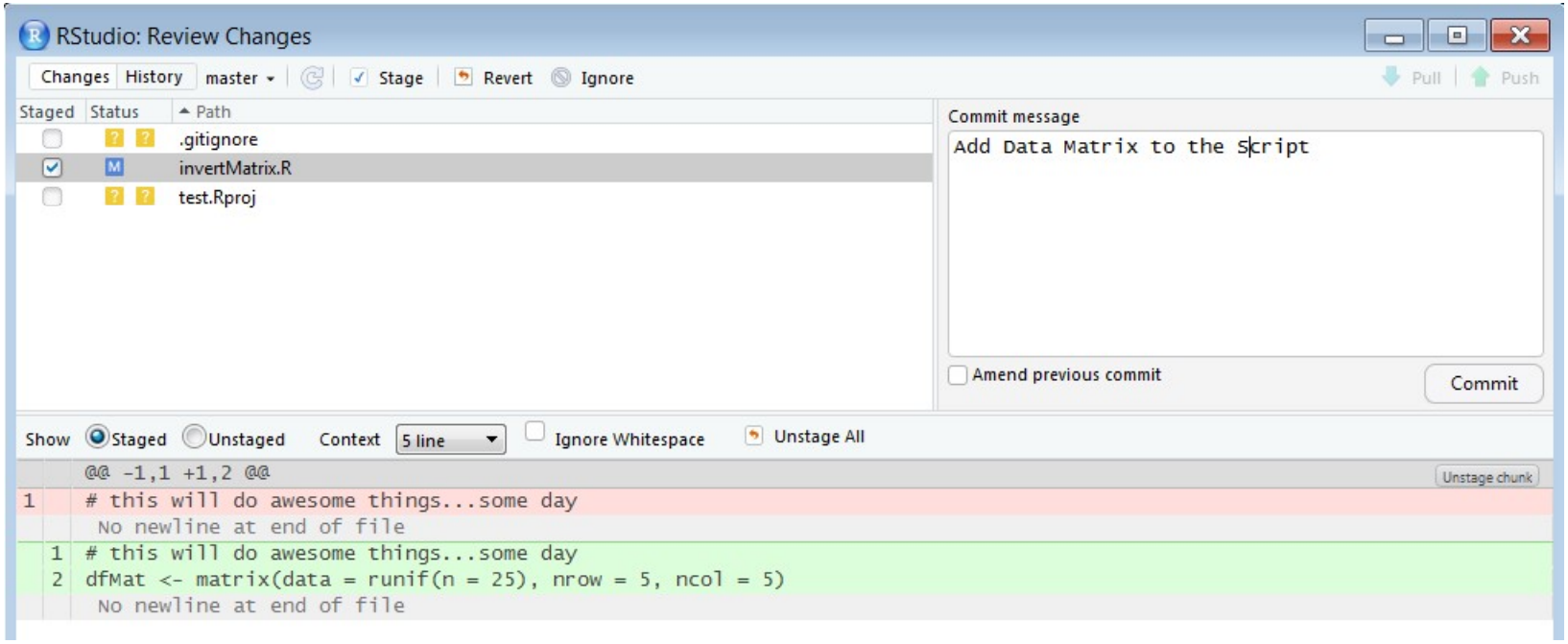
Commit the Changeset



Make Some Modifications



Commit the Modifications



Git in RStudio Covers *Most* of Your Needs

- If RStudio does all of this, why bother with the command line?
- *Most* of the time you won't need to, but when you need it, you need it
  - Merge conflicts, for example

Code Time

Connecting a Local Repo to a Remote Repo

Module 7-3

Git is a *Distributed* Version Control System

- Peer-to-peer as opposed to server-client
- Common operations (commits, viewing history, etc.) are fast since there is no need to communicate with a central server
- Communication is only necessary with sharing changes among peers
- Each working copy effectively functions as a remote backup of a codebase and its change history – protecting against data loss

https://en.wikipedia.org/wiki/Distributed_version_control

How Does My Local Repo Sync with GitHub*

- Several ways to do this:
- Clone an existing repository from GitHub to a local folder
  - Your own, or
  - Another coder's
- Initialize a repository locally and push to GitHub
- Start a new project in RStudio with version control

*or bitbucket, gitlab, etc.

Local First

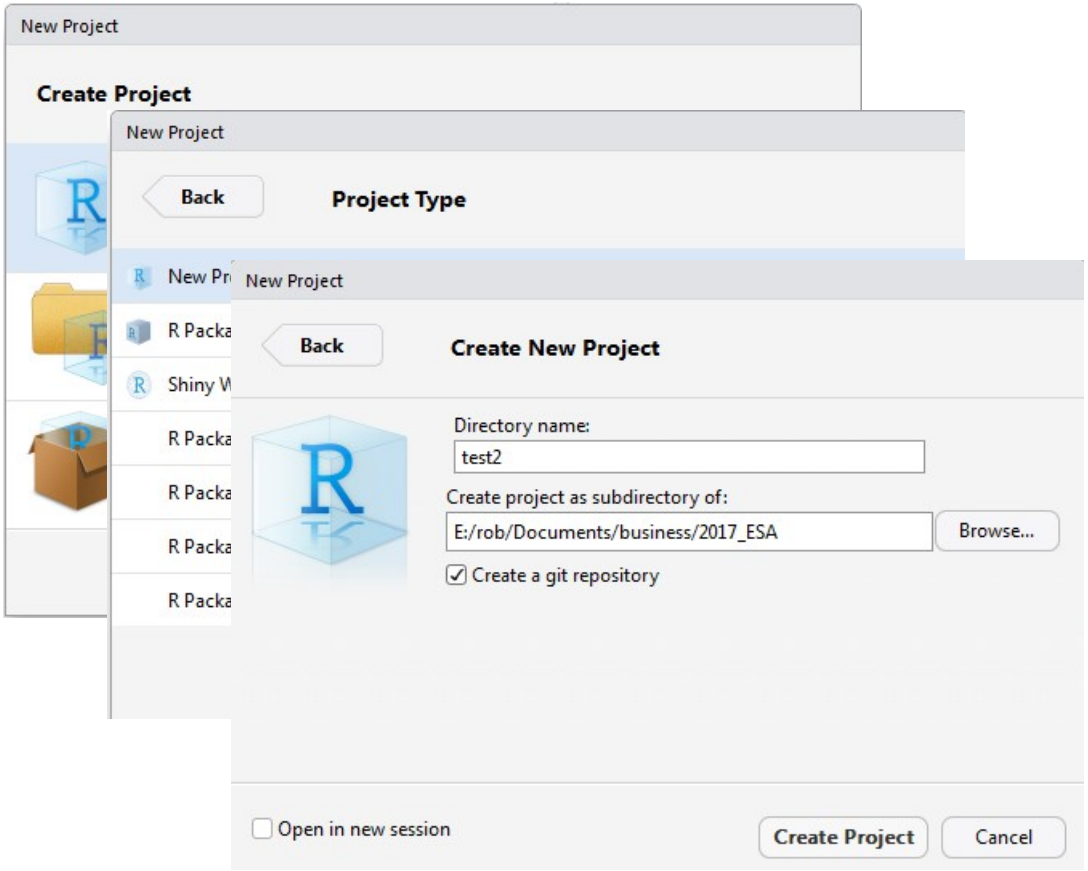
- We’ve started this with git init in a local repo
- How connect to GitHub?
  - Make an empty repo on GitHub, i.e. no README
  - Copy the url of this repo from GitHub
  - Navigate to your current folder
  - Add the remote
  - Push the repo

GitHub First

- You did a version of this when you worked through the Happy Git with R install tutorial
- How connect to GitHub?
  - Make a repo on GitHub (I typically add both a README and a .gitignore file)
  - Copy the url of this repo from GitHub
  - Navigate to your current folder
  - Clone the repo

From RStudio*

- Make a new project
- Project type
- Choose name & git

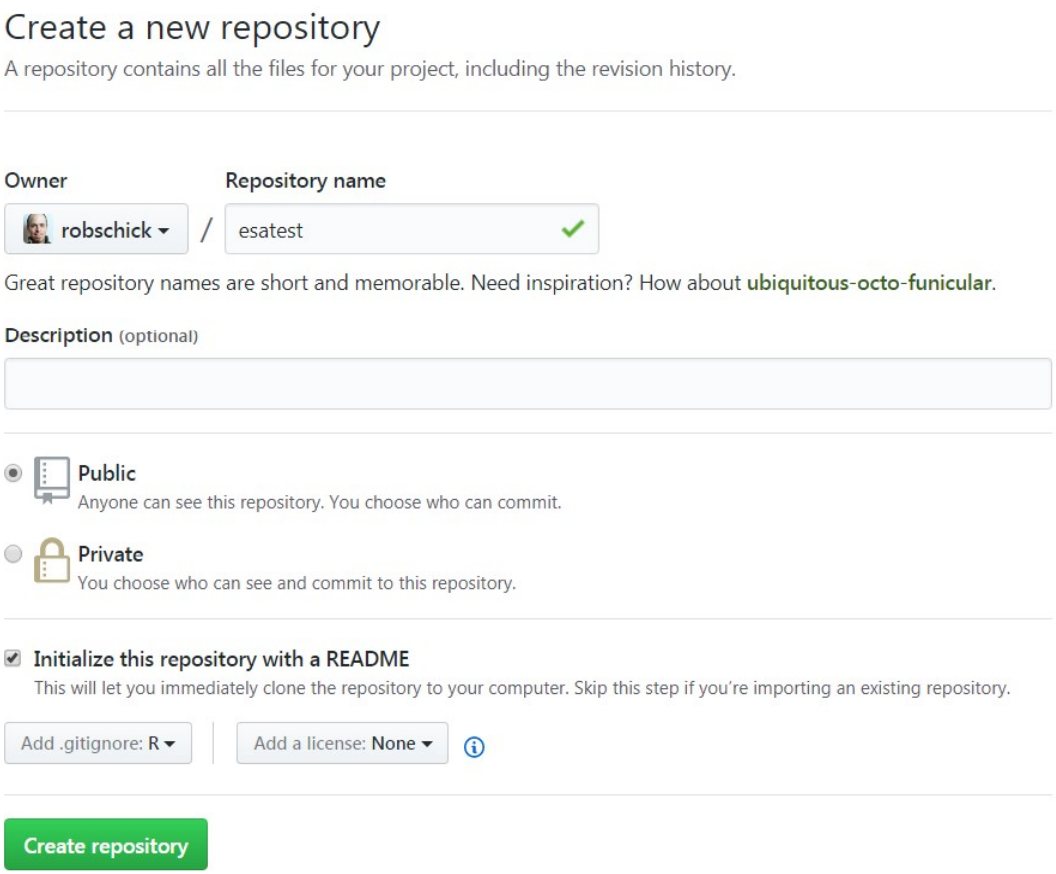


*https://support.rstudio.com/hc/en-us/articles/200532077?version=1.1.322&mode=desktop

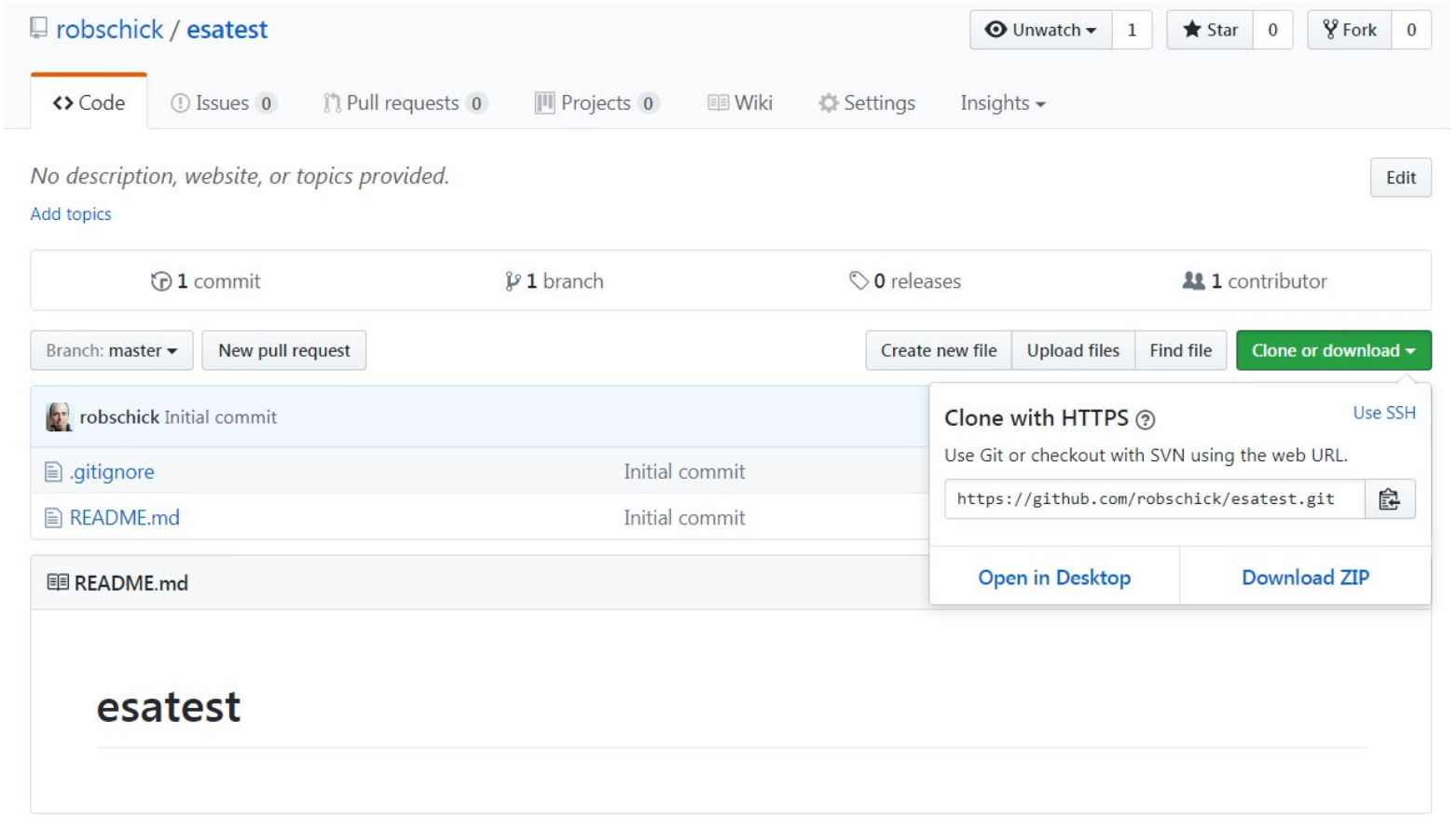
My Preferred Way

- Start on GitHub
- Create an empty repo
  - Include a README file
  - Include a .gitignore file
- Clone it locally on the command line
- Set up a new project in RStudio in the existing directory

Let’s Practice



Let’s Practice

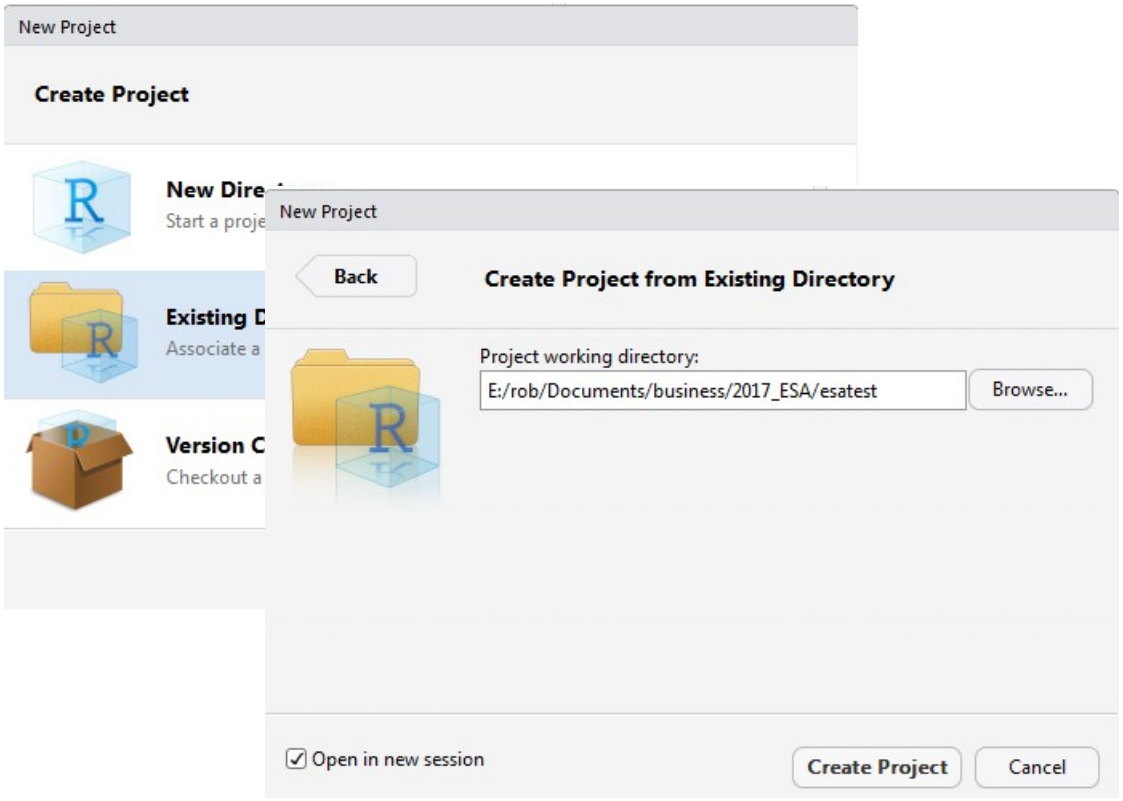


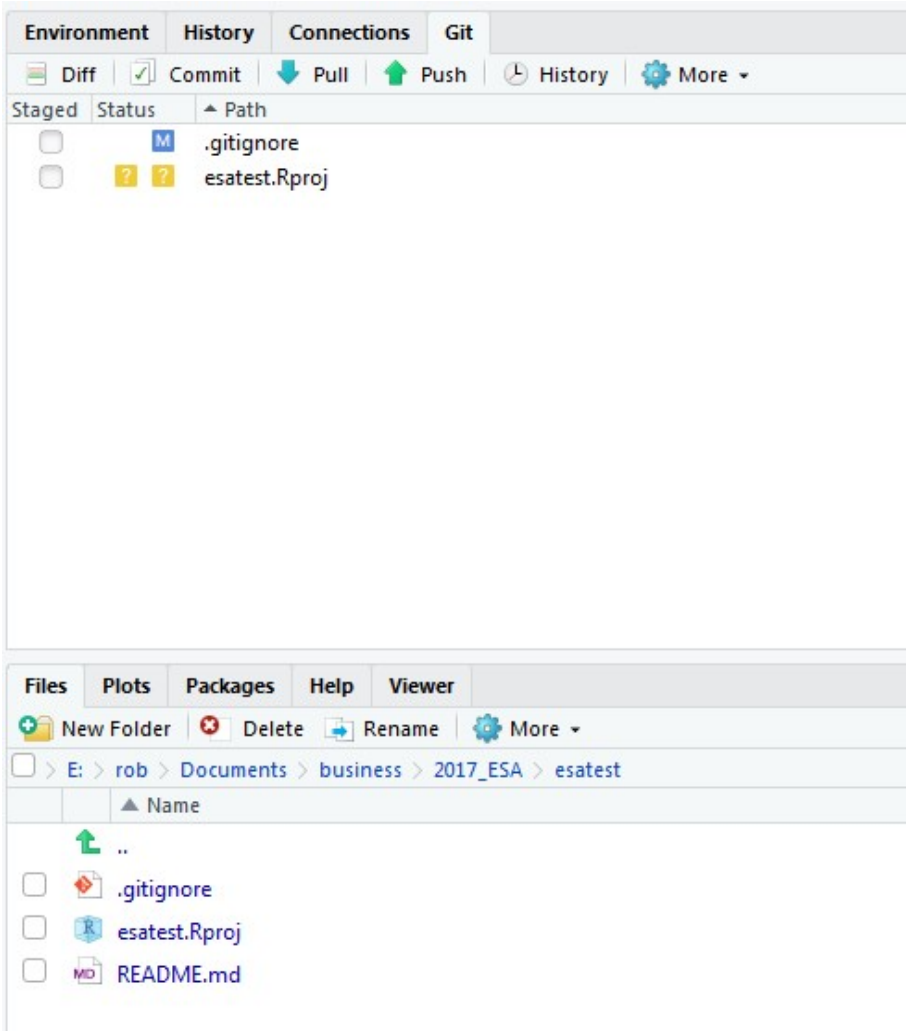
Let’s Practice

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA
$ git clone https://github.com/robschick/esatest.git
```

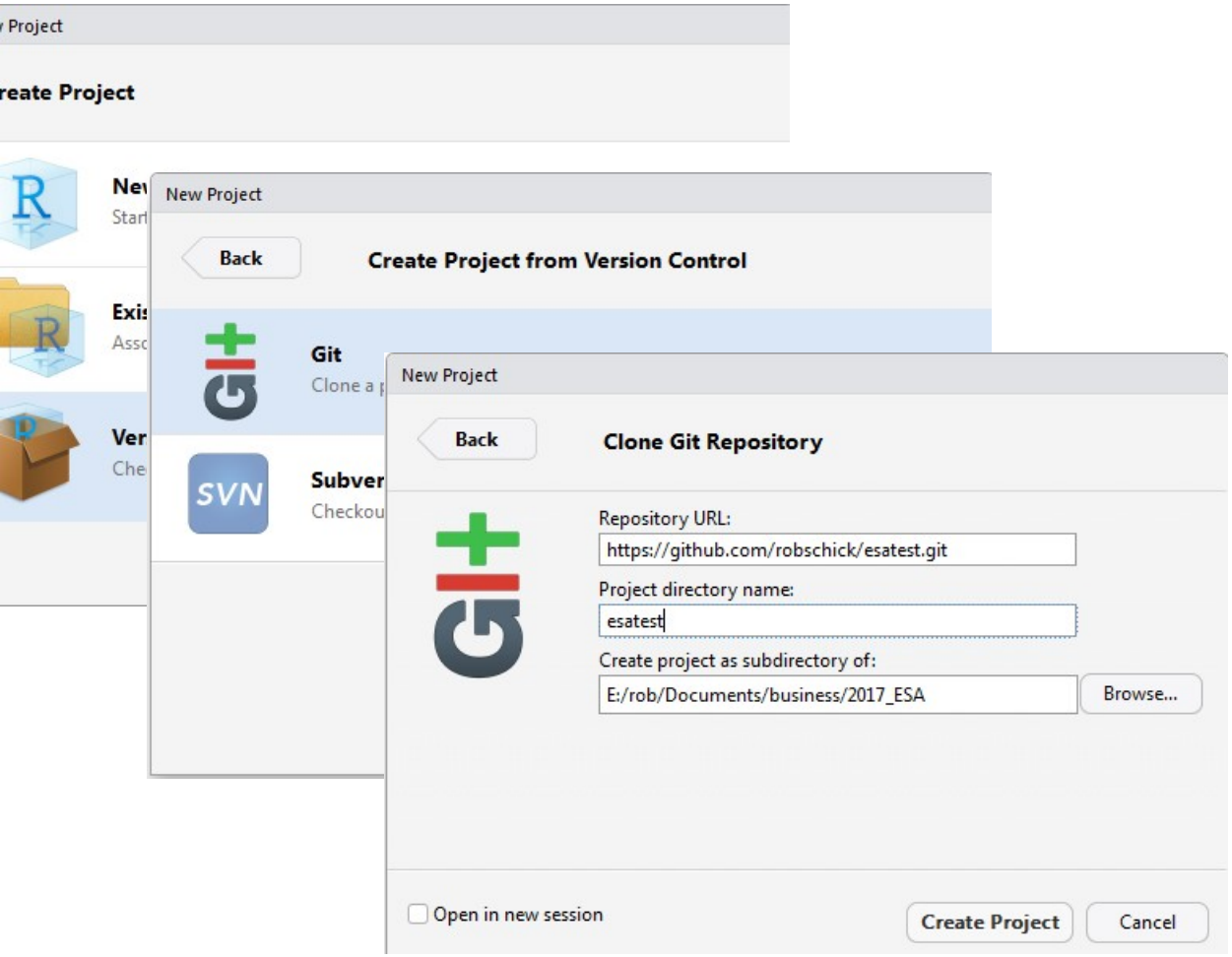
```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA
$ git clone https://github.com/robschick/esatest.git
Cloning into 'esatest'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA
$ cd esatest/

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

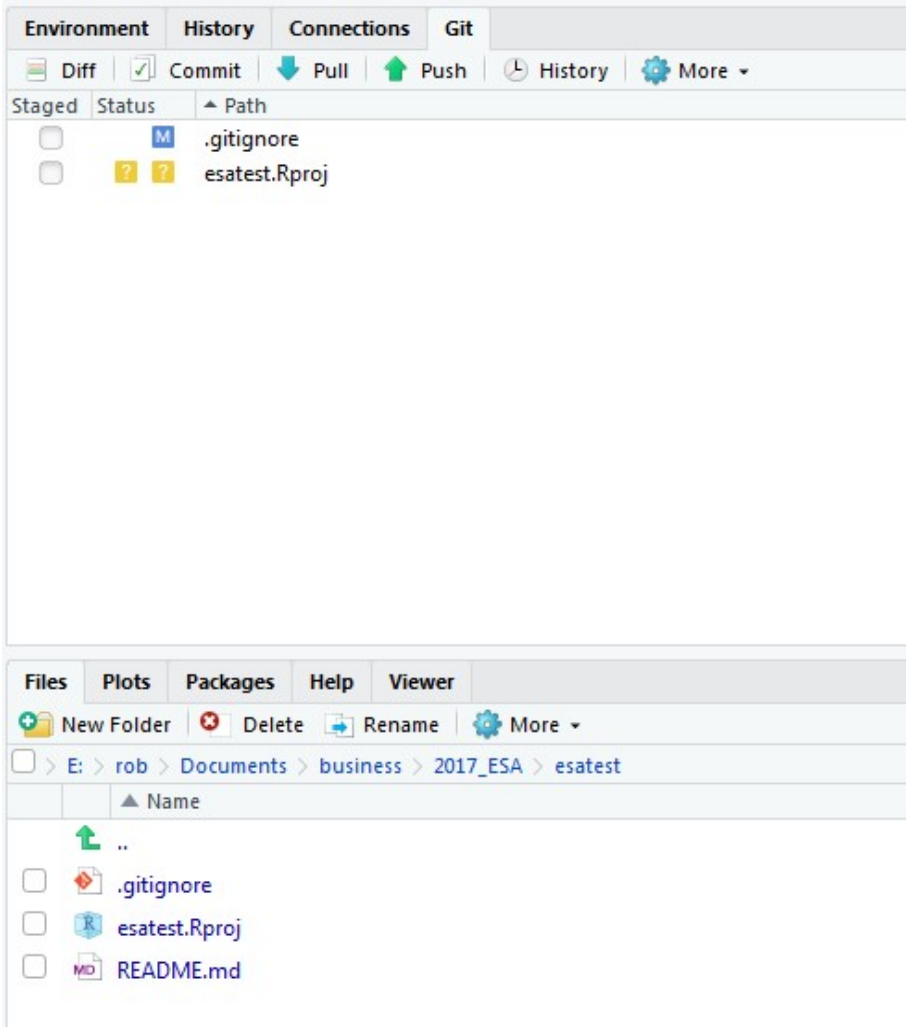
Let’s Practice





Bypass the Command Line





Pushing to GitHub – How Often?

- Several schools of thought on how often to push
- A prominent one (Hadley Wickham) is to push considerably less often than you commit*
  - “Pushing code means publishing code”
  - “Strive to push code that works”
- I tend to push more often than this, because I code on small teams, and I like the back up

*http://r-pkgs.had.co.nz/git.html#commit-best-practices

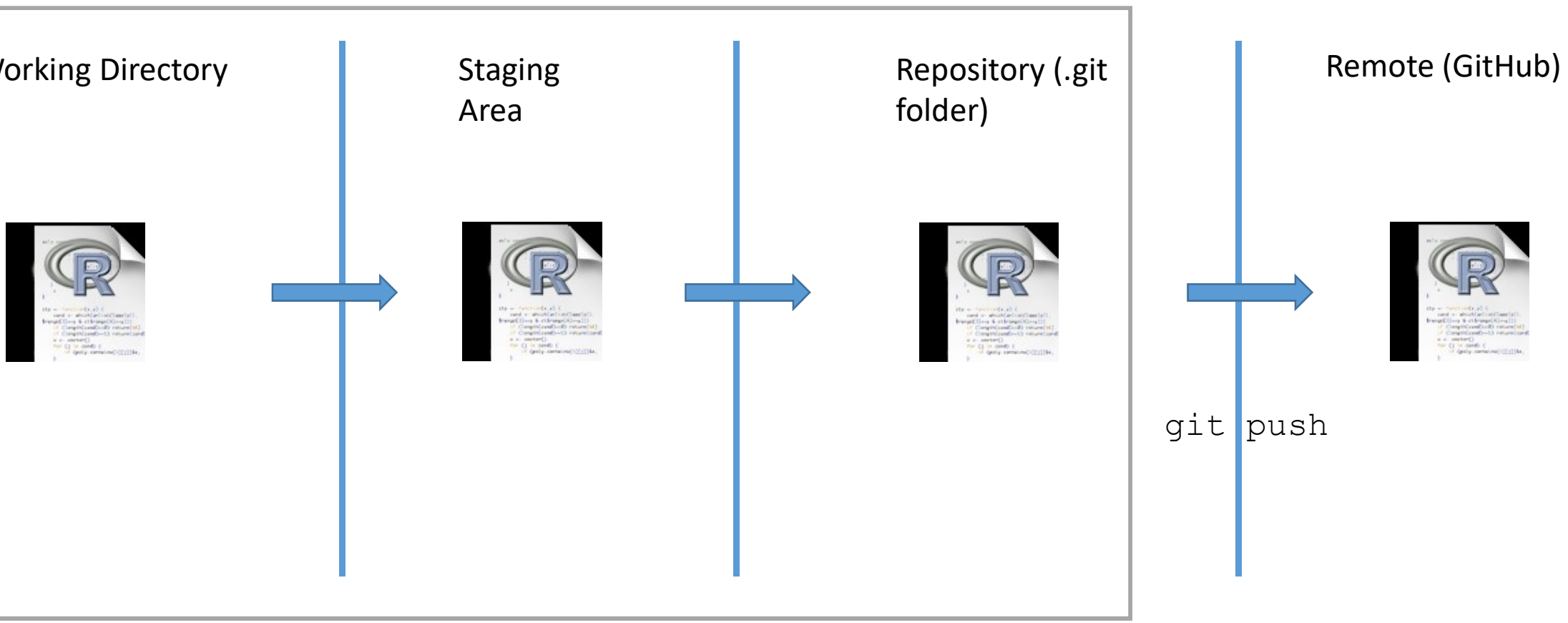
Pushing to GitHub – Good Commit Messages*

- **Be concise, yet evocative.** At a glance, you should be able to see what a commit does. But there should be enough detail so you can remember (and understand) what was done
- **Describe the why, not the what.** Since you can always retrieve the diff associated with the commit, the message doesn't need to say exactly what changed. Instead it should provide a high-level summary that focuses on the reasons for the change

*http://r-pkgs.had.co.nz/git.html#commit-best-practices

Practice a Cycle in Local Repo

Three Local States with Remote



Adding GitHub to the Workflow – `git push`

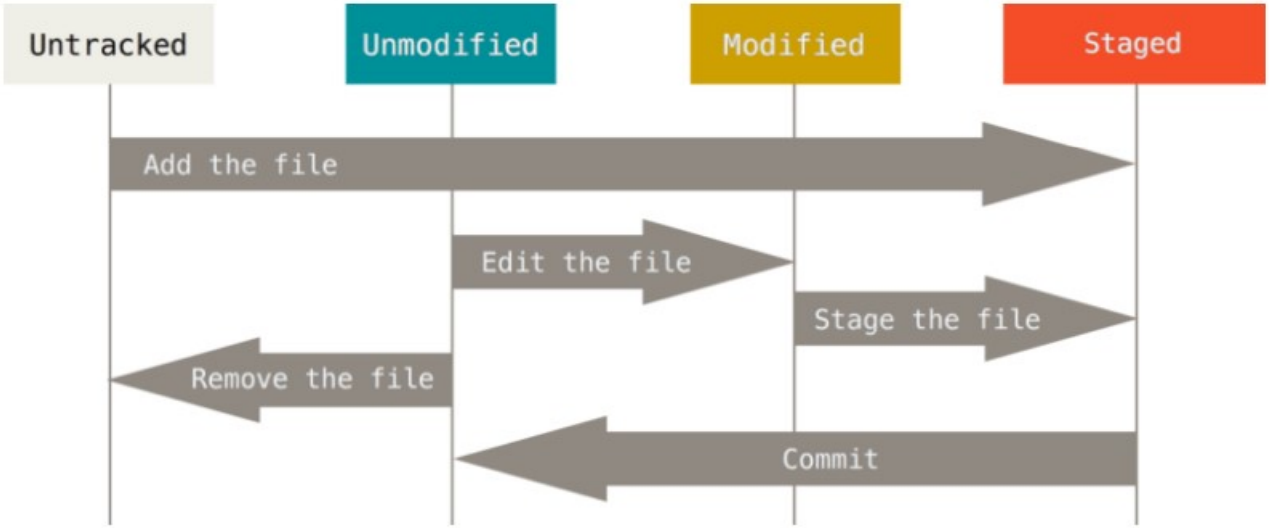
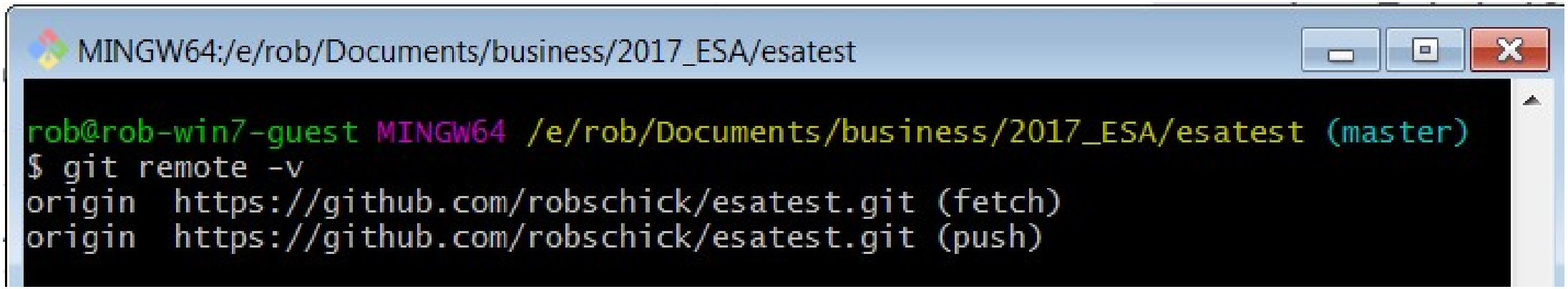


FIGURE 2-1

The lifecycle of the status of your files.

Chacon, S., and B. Straub. 2014. Pro Git. The Expert's Voice. Apress.

Knowing Your Remotes



What is a Merge Conflict?

- A `merge` operation in git is when you try to blend changes made:
  - To the *same* file
  - On two *different* branches
- Wait, but we haven't talked (much) about branches?
- For now, know that we've been working with one branch – `MASTER`
- So we can work with the same branch – `MASTER` – in two (or more repos):
  - Local
  - GitHub
  - Node
- And these can come into conflict – let's try!

Merge Conflicts

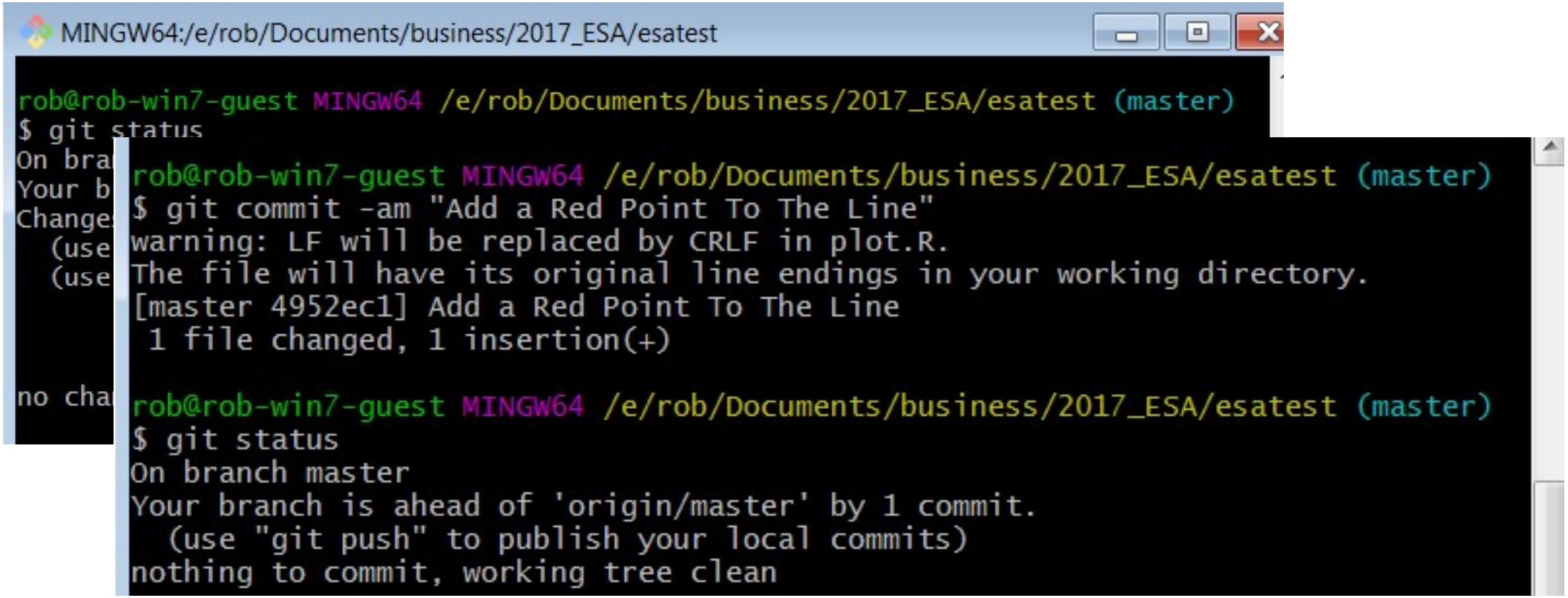
- Make sure your working directory shows a clean status
- Do a git pull
- Do a git push
- Modify the file locally
- Commit it, but don’t push it yet
- Navigate to the repo on GitHub
- Modify the same file on GitHub
- Commit it
- Go back to local repo and attempt a push

Check Status of Your Local Repo

- Best practice to pull before you push (n.b. if it's just you and GitHub, this isn't as big a concern)

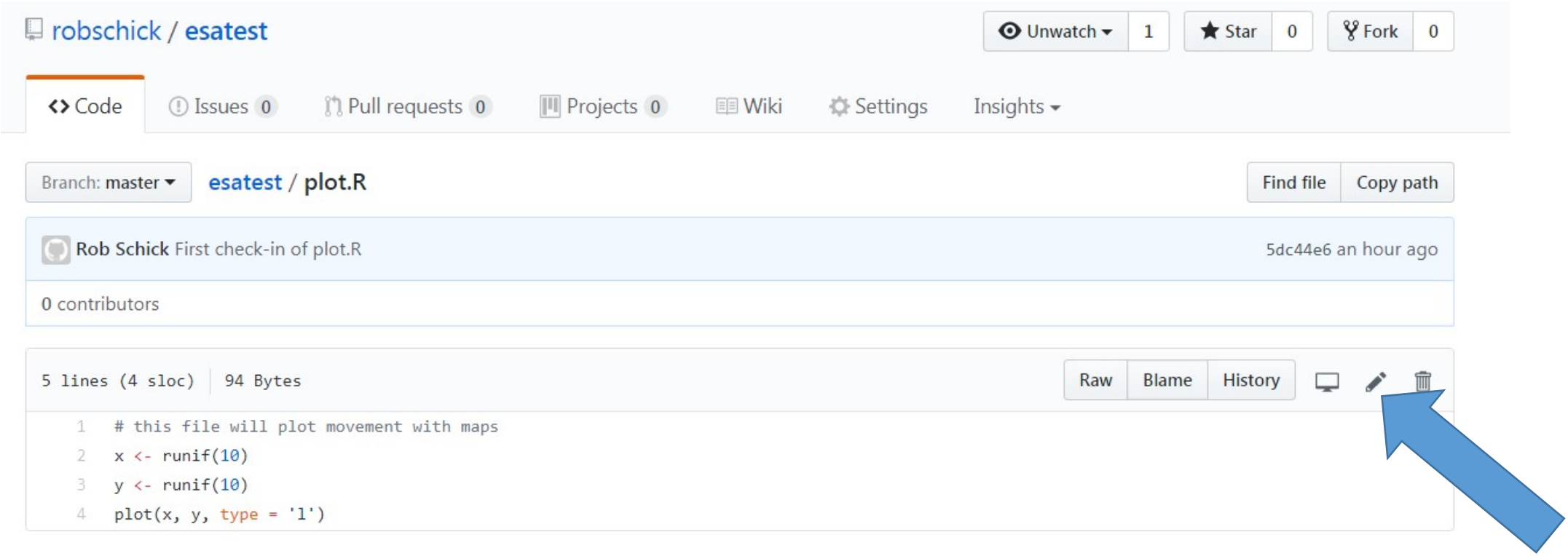
```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git pull
Already up-to-date.
```

Make Local Changes

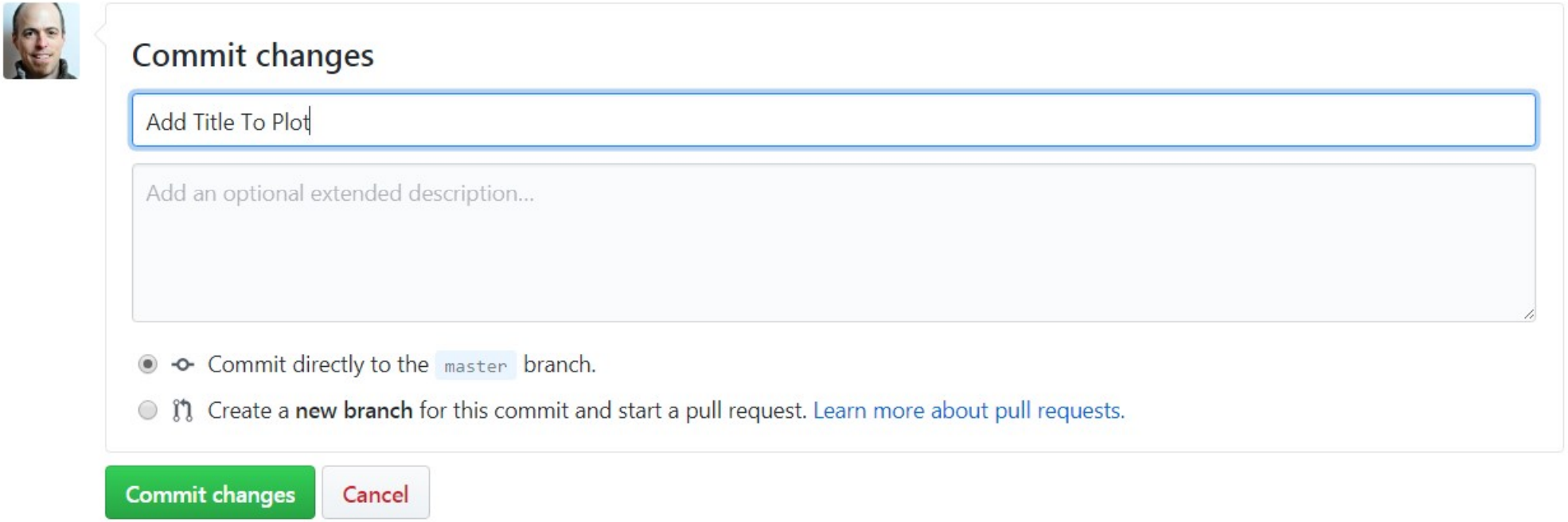


Make Remote Changes

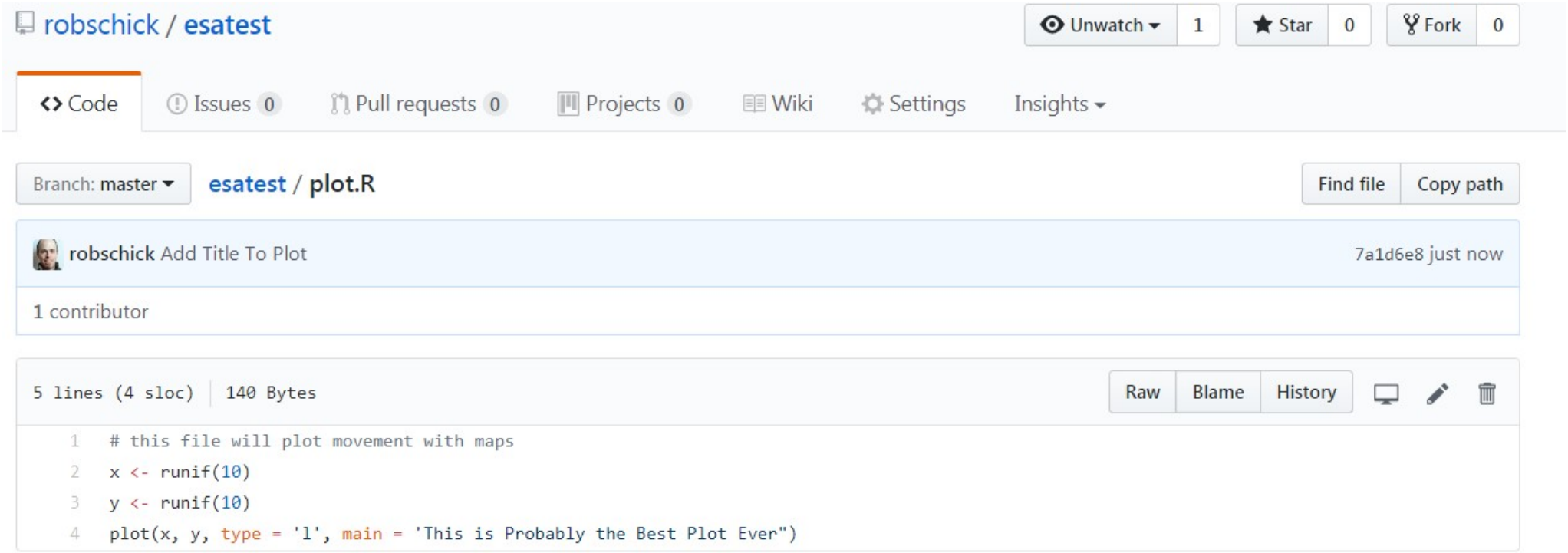
- Edit on GitHub



Commit the Changes

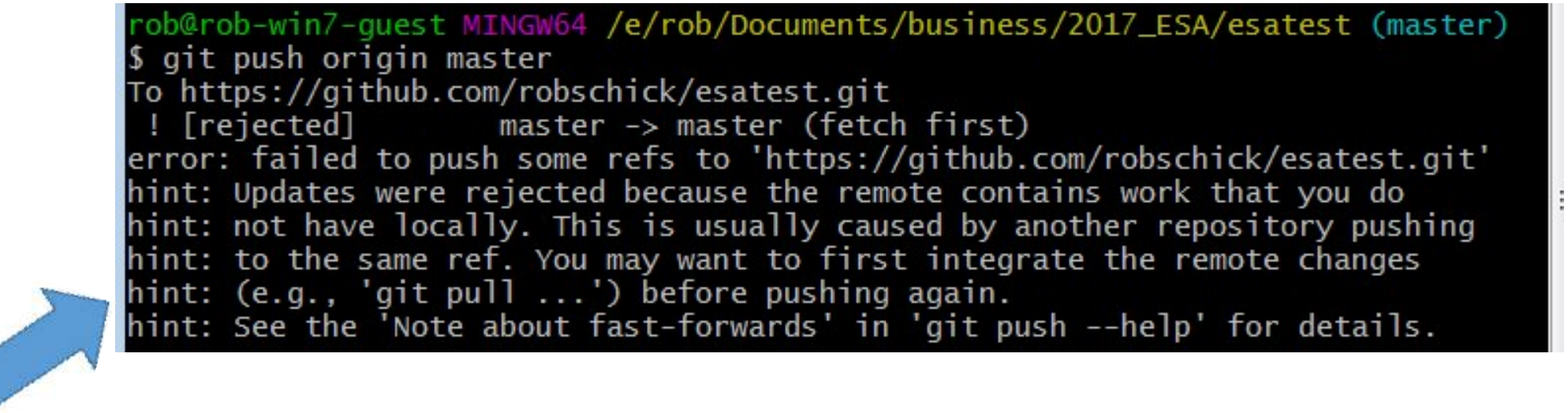


View Status on GitHub

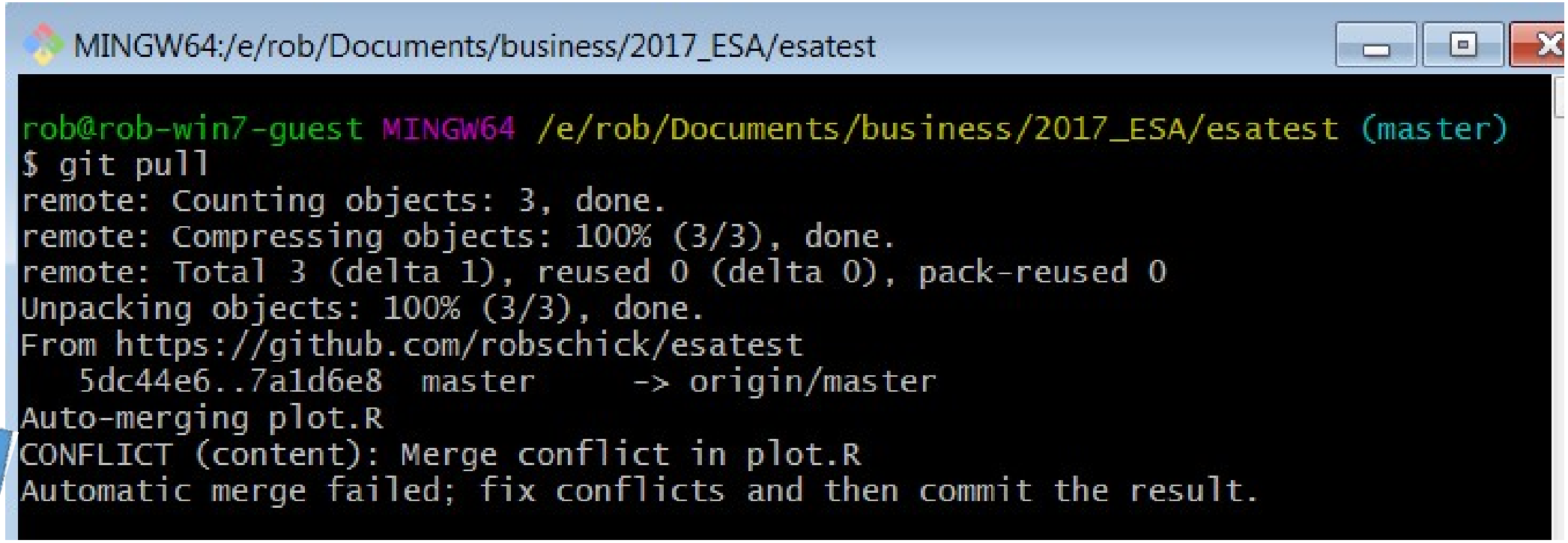


What's Happened

- We have one repo – esatest (or whatever you've named it)
- We've made and committed changes in two different places
- Now let's try to sync the repos by pushing our local to GitHub



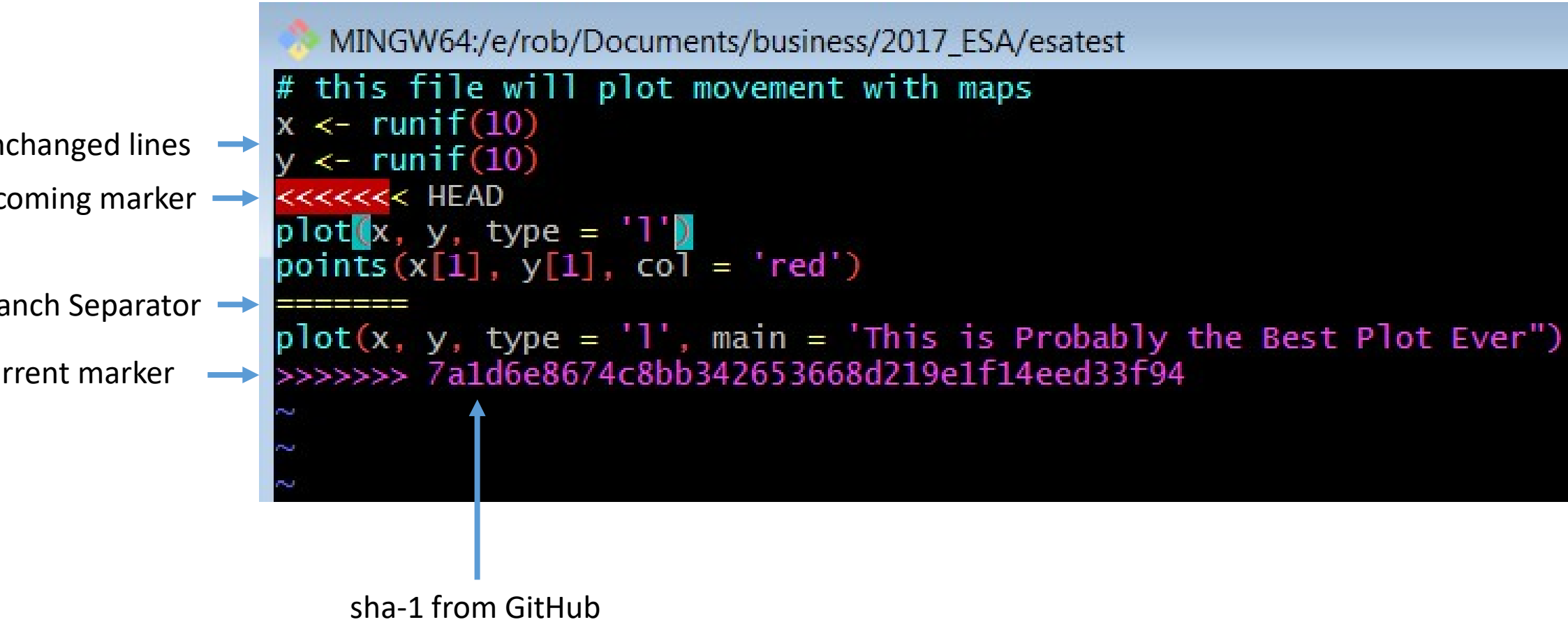
Pull the Changes



Check Status

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master|MERGING)
$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   plot.R

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

View the Conflict



Manually Resolve the Conflict

oose this Block →

Or this Block →

MINGW64:/e/rob/Documents/business/2017_ESA/esatest

```
# this file will plot movement with maps
x <- runif(10)
y <- runif(10)
<<<<<<< HEAD
plot(x, y, type = 'l')
points(x[1], y[1], col = 'red')
=======
plot(x, y, type = 'l', main = 'This is Probably the Best Plot Ever")
>>>>>>> 7a1d6e8674c8bb342653668d219e1f14eed33f94
~
~
~
```

You can configure graphical merge tools like p4merge to make this easier

Add it, Commit it, Push it

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master|MERGING)
$ git add plot.R

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master|MERGING)
$ git commit -m "Resolve Merge Conflict on Plot Line"
[master 713ec2a] Resolve Merge Conflict on Plot Line

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git pull
Already up-to-date.

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git push
Counting objects: 4, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 522 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/robschick/esatest.git
   7a1d6e8..713ec2a  master -> master
```

Note statu

Merge Res

Merge Commits Look a Bit Different

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git show head
commit 713ec2a36e4e4c6d60c2caeb02ac3de7c23c6eac (HEAD -> master, origin/master, origin/HEAD)
Merge: 4952ec1 7a1d6e8
Author: Rob Schick <robschick@gmail.com>
Date:   Wed Aug 2 20:25:24 2017 -0400

    Resolve Merge Conflict on Plot Line


rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git log --oneline --decorate --graph
*   713ec2a (HEAD -> master, origin/master, origin/HEAD) Resolve Merge Conflict on Plot Line
|\
| * 7a1d6e8 Add Title To Plot
* | 4952ec1 Add a Red Point To The Line
|/
* 5dc44e6 First check-in of plot.R
* 771ea28 Initial commit

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
```

Just Another Repo!

- Well, yes, but…
- I'll try a live demo to show what happens
- If it fails, following is an example of the normal bs that you need to put up with
- (But it's worth it!)

SSH to a Linux Box

- Cloned my repo from GitHub
- Created a new file
- Added it
- Pushed it:

```
[rss10@tippy2 esatest]$ git push origin master
error: The requested URL returned error: 403 Forbidden while access
ithub.com/robschick/esatest.git/info/refs

fatal: HTTP request failed
```

- ??? Stackoverflow to the rescue

Need to Use SSH not HTTP

618

I just got the same problem and just figured out what's cause.

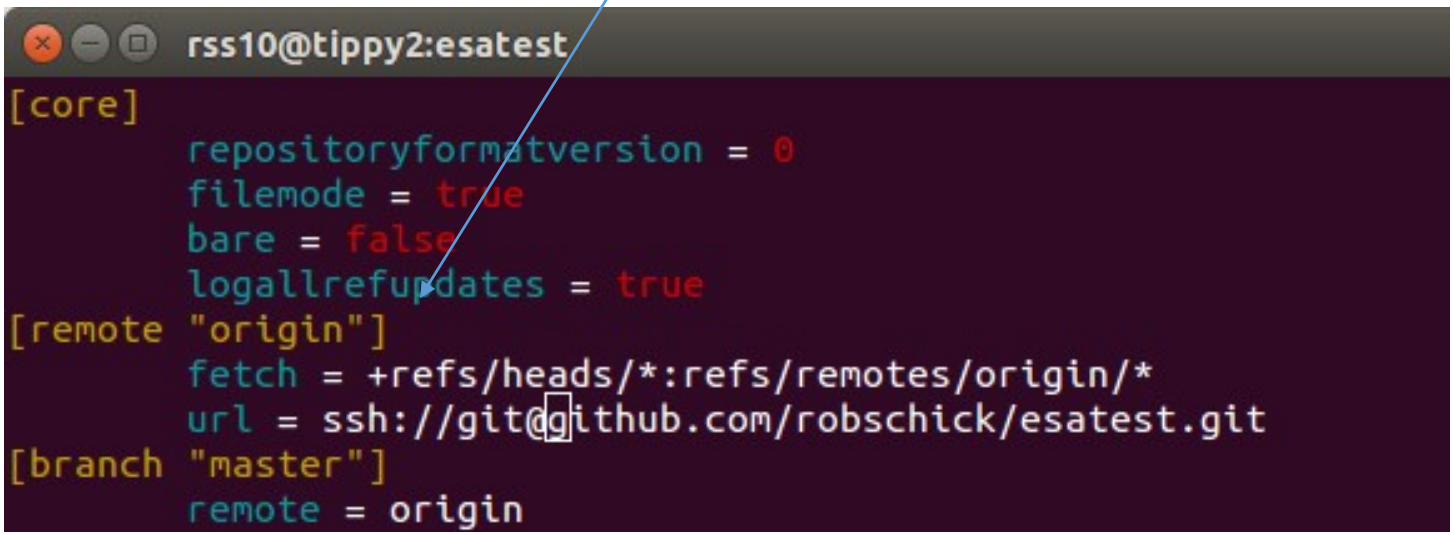
Github seems only supports ssh way to read&write the repo, although https wa
'Read&Write'.

So you need to change your repo config on your PC to ssh way:

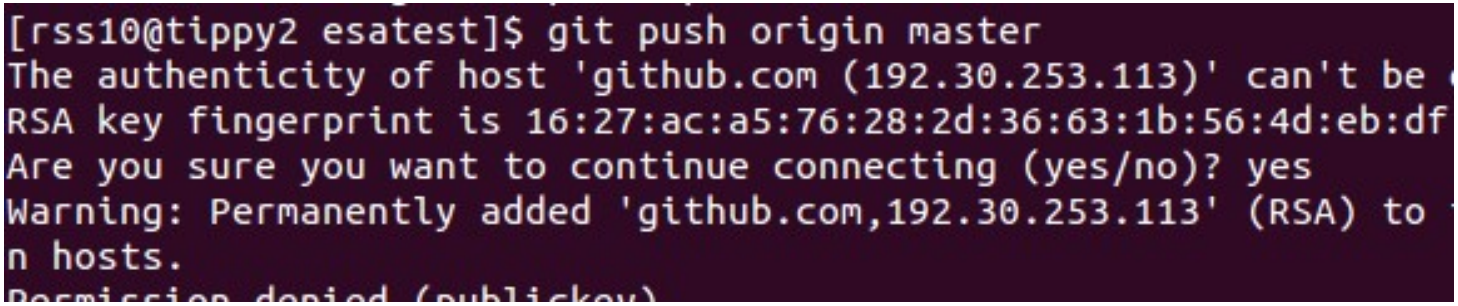
1. edit `.git/config` file under your repo directory
2. find `url=` entry under section `[remote "origin"]`
3. change it from `url=https://MichaelDrogalis@github.com/derekerdman`
   to `url=ssh://git@github.com/derekerdmann/lunch_call.git` . that is, c
   before `@` symbol to `ssh://git`
4. Save `config` file and quit. now you could use `git push origin master`
   on GitHub

```
q: command not found.

shell returned 1

Press ENTER or type command to continue
[rss10@tippy2 esatest]$ more .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
```

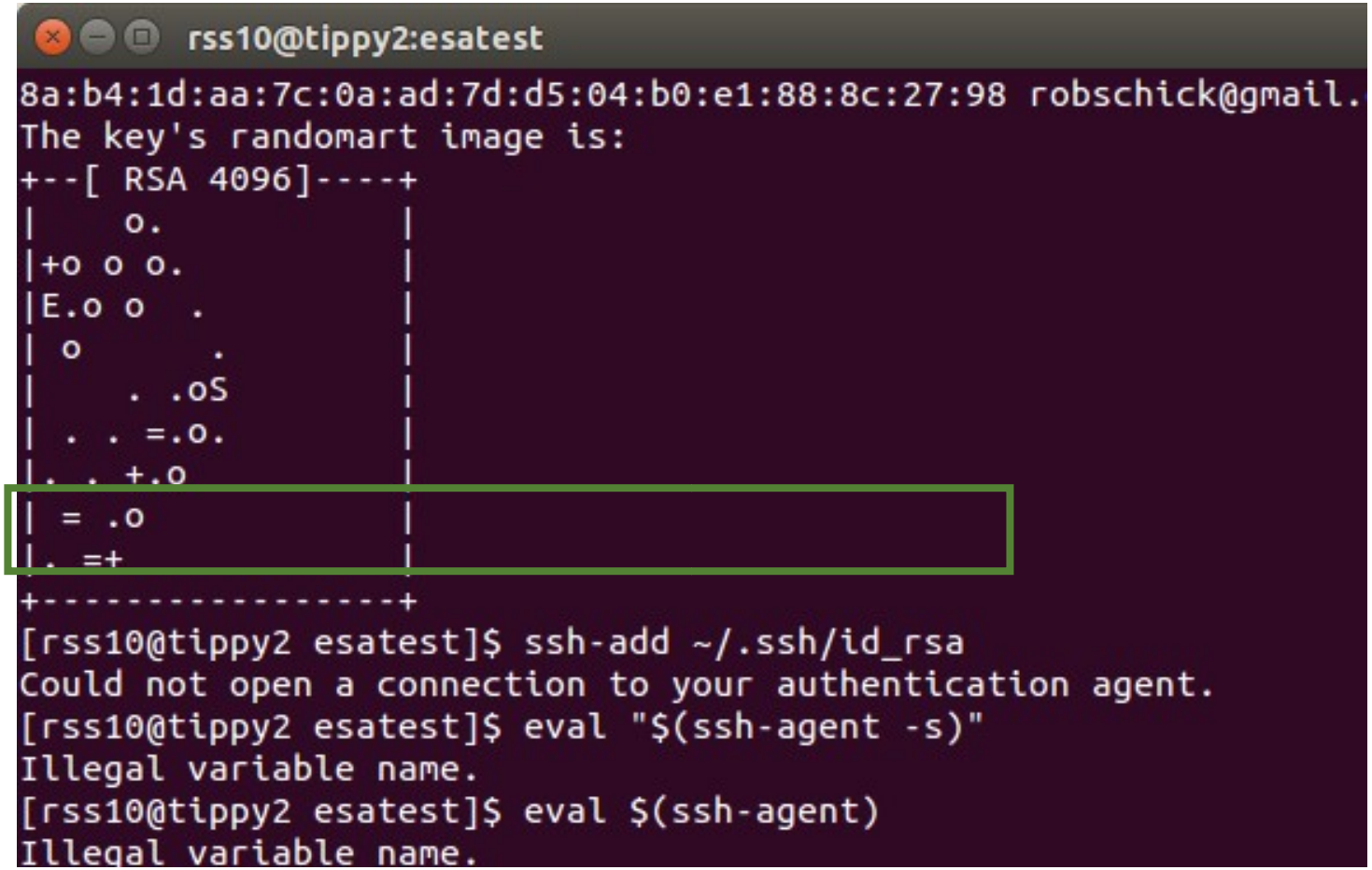
Need to change this

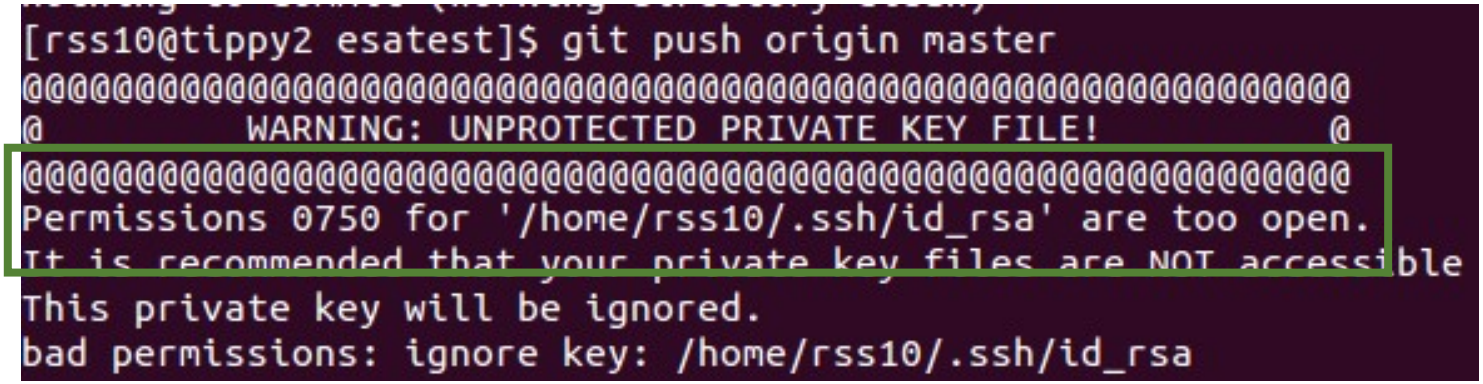


No LUCK!



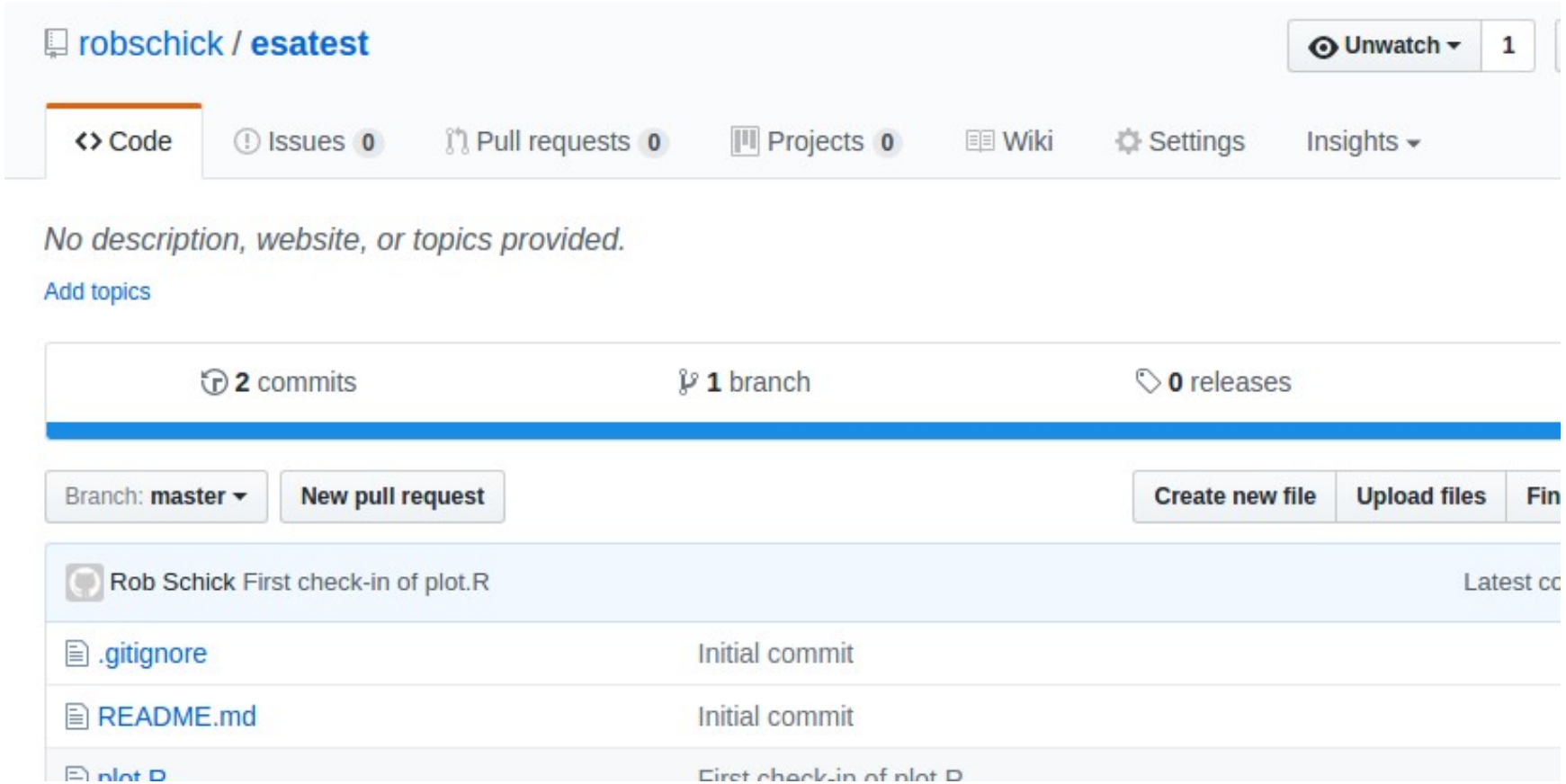
Need to pair the SSH keys….
http://happygitwithr.com/ssh-keys.html

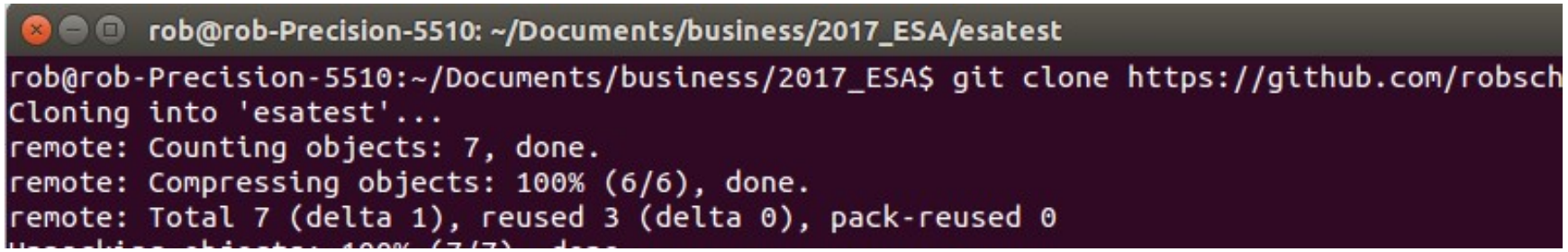

rss10@tippy2:esatest
8a:b4:1d:aa:7c:0a:ad:7d:d5:04:b0:e1:88:8c:27:98 robschick@gmail.
The key's randomart image is:
+--[RSA 4096]----+
| o. |
|+o o o. |
|E.o o . |
| o   . |
| . .oS |
| . . =.o. |
|. . +.o |
| = .o |
|. =+ |
+-----------------+
[rss10@tippy2 esatest]$ ssh-add ~/.ssh/id_rsa
Could not open a connection to your authentication agent.
[rss10@tippy2 esatest]$ eval "$(ssh-agent -s)"
Illegal variable name.
[rss10@tippy2 esatest]$ eval $(ssh-agent)
Illegal variable name.

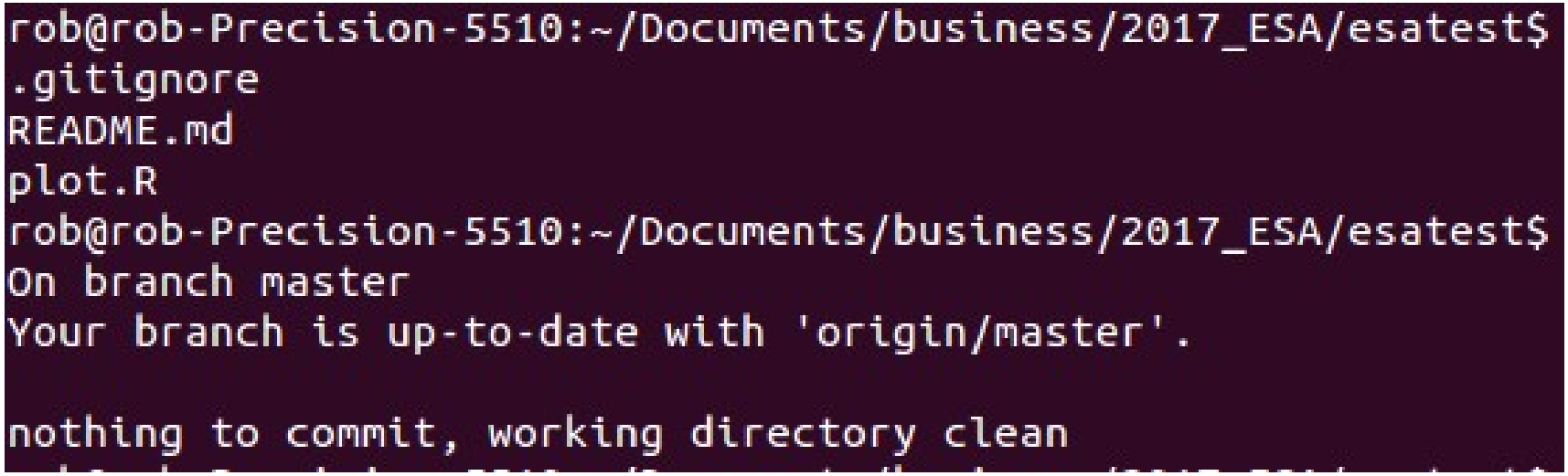


https://stackoverflow.com/questions/9270734/ssh-permissions-are-too-open-error


[rss10@tippy2 esatest]$ chmod 600 ~/.ssh/id_rsa
[rss10@tippy2 esatest]$ git push origin master
Warning: Permanently added the RSA host key for IP address '192.30.253.112' to the l
Enter passphrase for key '/home/rss10/.ssh/id_rsa':
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 404 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)


robschick / esatest
Unwatch 1
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Settings
Insights
No description, website, or topics provided.
Add topics
2 commits
1 branch
0 releases
Branch: master
New pull request
Create new file
Upload files
Fin
Rob Schick First check-in of plot.R
Latest cc
.gitignore
Initial commit
README.md
Initial commit
plot.R
First check-in of plot.R


rob@rob-Precision-5510: ~/Documents/business/2017_ESA/esatest
rob@rob-Precision-5510:~/Documents/business/2017_ESA$ git clone https://github.com/robsch
Cloning into 'esatest'...
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 3 (delta 0), pack-reused 0

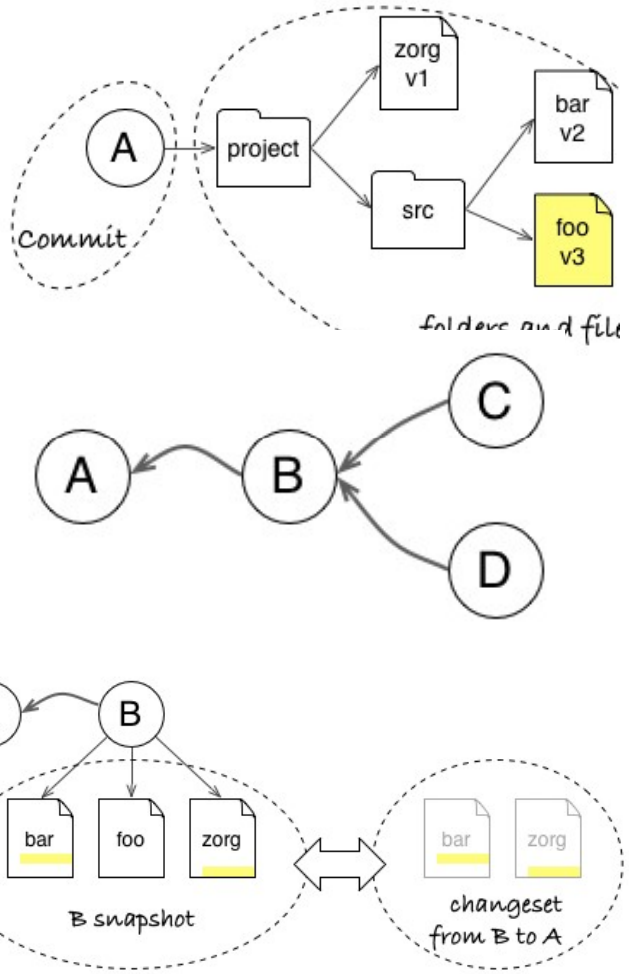

rob@rob-Precision-5510:~/Documents/business/2017_ESA/esatest$
.gitignore
README.md
plot.R
rob@rob-Precision-5510:~/Documents/business/2017_ESA/esatest$
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean

How Does the Repo Change?

Lesson 7-4

3 Core Concepts

- Snapshot
- Graph
- Changeset

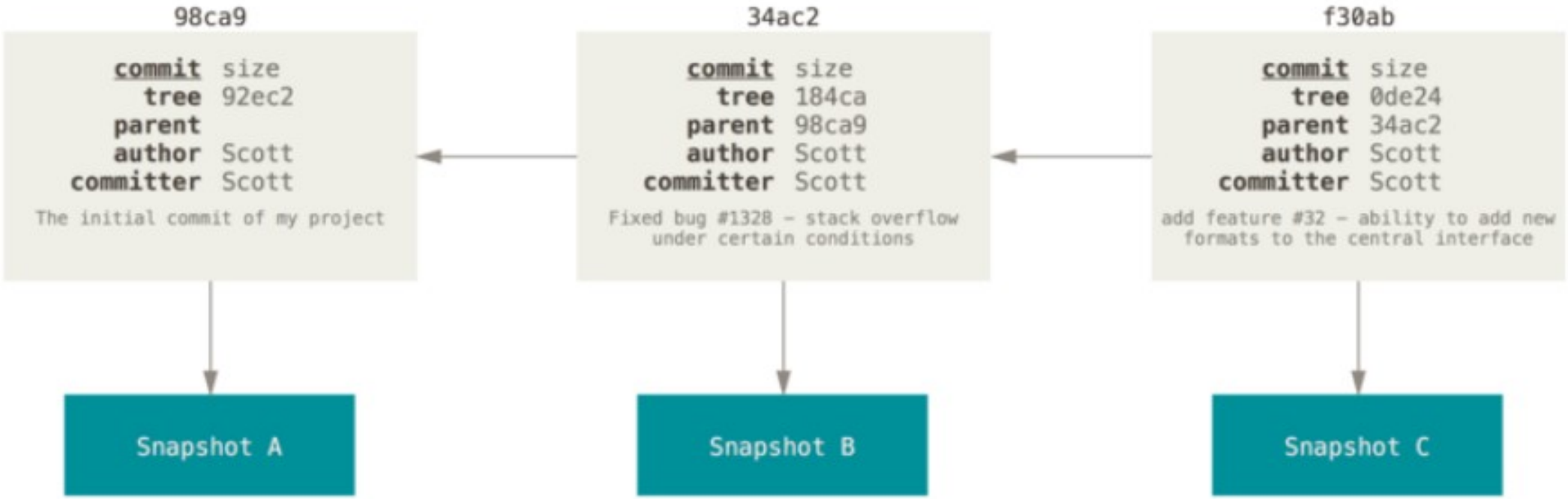


With These Concepts, We Can

- Use the sha-1 unique identifier to view a snapshot
- Compare the differences from one snapshot to a next
- Compare two different snapshots/commits
- Revert to the project at specific points (commits) in time

sha-1 Identifier – These Are Crucial

- These are the nodes along the graph
- We use them to view and/or navigate



Seeing Changes Before a Commit

alize Changes
re a commit

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git ls-files
invertMatrix.R

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ vi invertMatrix.R

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff
diff --git a/invertMatrix.R b/invertMatrix.R
index d84a7d4..736e263 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,3 +1,4 @@
 # this will do awesome things...some day
 dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
 head(dfMat)
+dfMat[1, ]

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$
```

Seeing Changes Between Two Diffs

1 commit IDs

alize differences
ween 2 commits

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git log --oneline
7e23a0b (HEAD -> master) See First Row of Matrix
c8e3e5e Look at first 6 lines
c572b41 Add Data Matrix to the Script
354062b First Commit of invertMatrix.R

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff 7e23a0b c572b41
diff --git a/invertMatrix.R b/invertMatrix.R
index 736e263..fd757e0 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,4 +1,2 @@
 # this will do awesome things...some day
-dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
-head(dfMat)
-dfMat[1, ]
+dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
\ No newline at end of file

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$
```

Order Matters Between Two Diffs

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff 7e23a0b c572b41
diff --git a/invertMatrix.R b/invertMatrix.R
index 736e263..fd757e0 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,4 +1,2 @@
 # this will do awesome things...some day
-dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
-head(dfMat)
-dfMat[1, ]
+dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
\ No newline at end of file
```

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff c572b41 7e23a0b
diff --git a/invertMatrix.R b/invertMatrix.R
index fd757e0..736e263 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,2 +1,4 @@
 # this will do awesome things...some day
-dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
\ No newline at end of file
+dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
+head(dfMat)
+dfMat[1, ]
```

Seeing Changes Between Two Diffs

- `git diff master~1 master`
- `git diff <sha-1> <sha-1>`
- `git diff` (this just shows changes made but not added)

- Why all the hassle?
  - Diffs tell you what changed
  - Commit messages tell you why
  - With both, you can easily navigate through the history of a repo

Going to a Particular Commit

- `git checkout <sha-1>`

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git checkout c572b41
Note: checking out 'c572b41'.
```

Going to a Particular Commit

- `git checkout c572b41`

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git checkout c572b41
Note: checking out 'c572b41'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at c572b41... Add Data Matrix to the Script
```

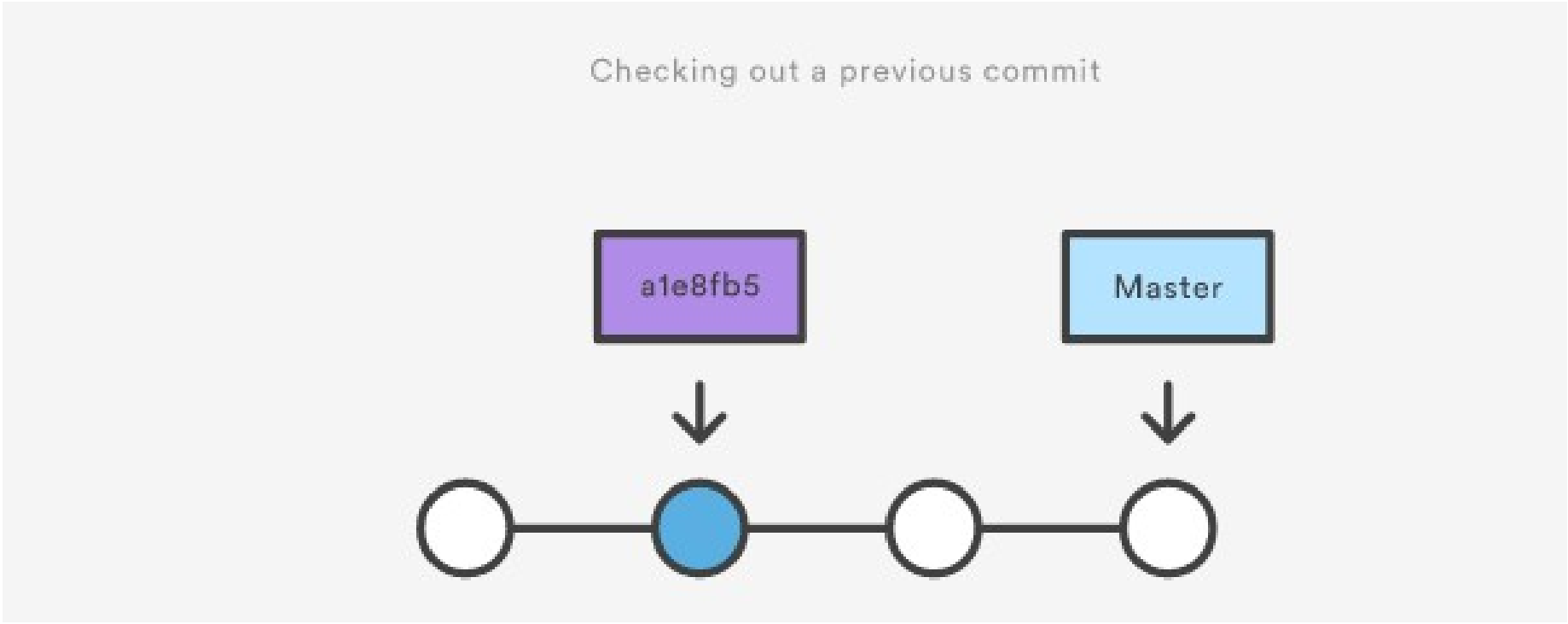
Detached Head???

- `git log --oneline --decorate`

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...)
)
$ git log --oneline --decorate
c572b41 (HEAD) Add Data Matrix to the Script
354062b First Commit of invertMatrix.R
```

- Has to do with where the branch is pointing – now it's to a previous commit
- Explore your script and you'll see it from that state:

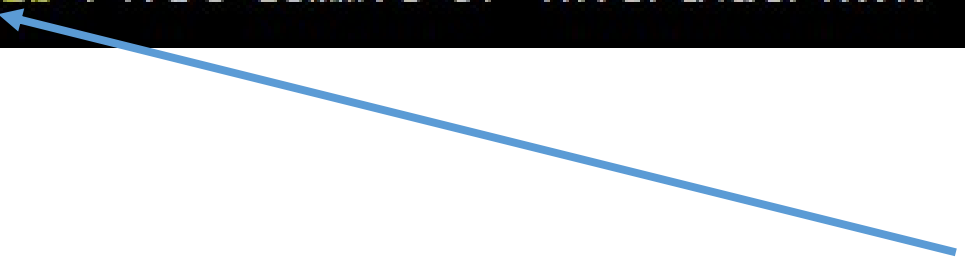
Where are we?



Detached Head???

- `git log --oneline --decorate`

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...)
)
$ git log --oneline --decorate
c572b41 (HEAD) Add Data Matrix to the Script
354062b First Commit of invertMatrix.R
```



Where did the other commits go?

The Commits Are Still There - Phew!

- `git checkout master`

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...)
)
$ git checkout master
Previous HEAD position was c572b41... Add Data Matrix to the Script
Switched to branch 'master'

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git log --oneline --decorate
7e23a0b (HEAD -> master) See First Row of Matrix
c8e3e5e Look at first 6 lines
c572b41 Add Data Matrix to the Script
354062b First Commit of invertMatrix.R
```

Why Would You Want to Do This?

- Allows you to work with the file as it was at that time:

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...)
)
$ cat invertMatrix.R
# this will do awesome things...some day
dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
```

- And then bring those changes into the current commit
- I'll argue for using a branch to do this in the last section

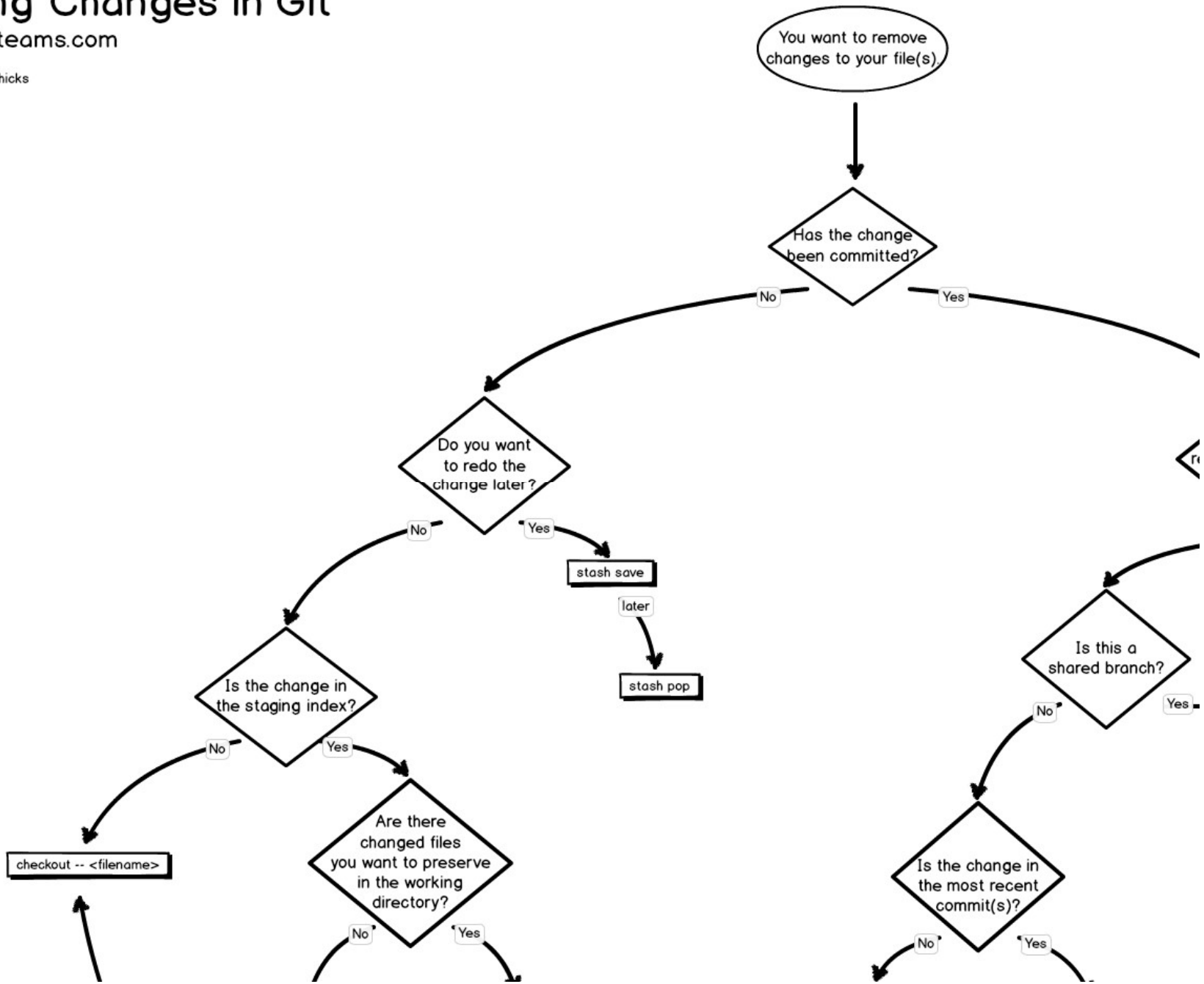
Backing Out Changes

- This can get complicated!
- Three main commands:
  - `git reset (hard, soft, mixed)`
  - `git rebase`
  - `git revert`
  - `git cherry-pick`
- This diagram helps:

Undoing Changes in Git

www.gitforteams.com

Author: emmajane
Contributors: jameyhicks



More Resources

- https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things
- https://www.atlassian.com/git/tutorials/undoing-changes
- https://www.atlassian.com/git/tutorials/rewriting-history