AUTOMATED VERSION CONTROL WITH GIT

Chris Fonnesbeck
VUMC Biostatistics

“WHAT IS VERSION CONTROL AND WHY SHOULD I USE IT?”

OBJECTIVES

1. Understand the benefits of an automated version control system.
2. Understand the basics of how Git works.

Changes are saved sequentially

Different Versions Can be Saved

Multiple Versions Can be Merged

VCS

COMMIT

In case of fire

1. git commit

2. git push

3. leave building

REPOSITORY

```
root@ubuntu:~/gitdemo/.git# tree
.
├── branches
├── config
├── description
├── HEAD
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── prepare-commit-msg.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   └── update.sample
├── info
│   └── exclude
├── objects
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags

9 directories, 13 files
```

THE HISTORY OF VERSION CONTROL SYSTEMS

- RCS
- CVS
- Subversion
- Git
- Mercurial

EXAMPLE PAPER WRITING

» Imagine you drafted an excellent paragraph for a paper you are writing, but later ruin it. How would you retrieve the excellent version of your paragraph? Is it even possible?

» Imagine you have 5 co-authors. How would you manage the changes and comments they make to your paper?

“HOW DO I GET SET UP TO USE GIT?”

When we use Git on a new computer for the first time, we need to configure a few things.

» specify our name and email address,

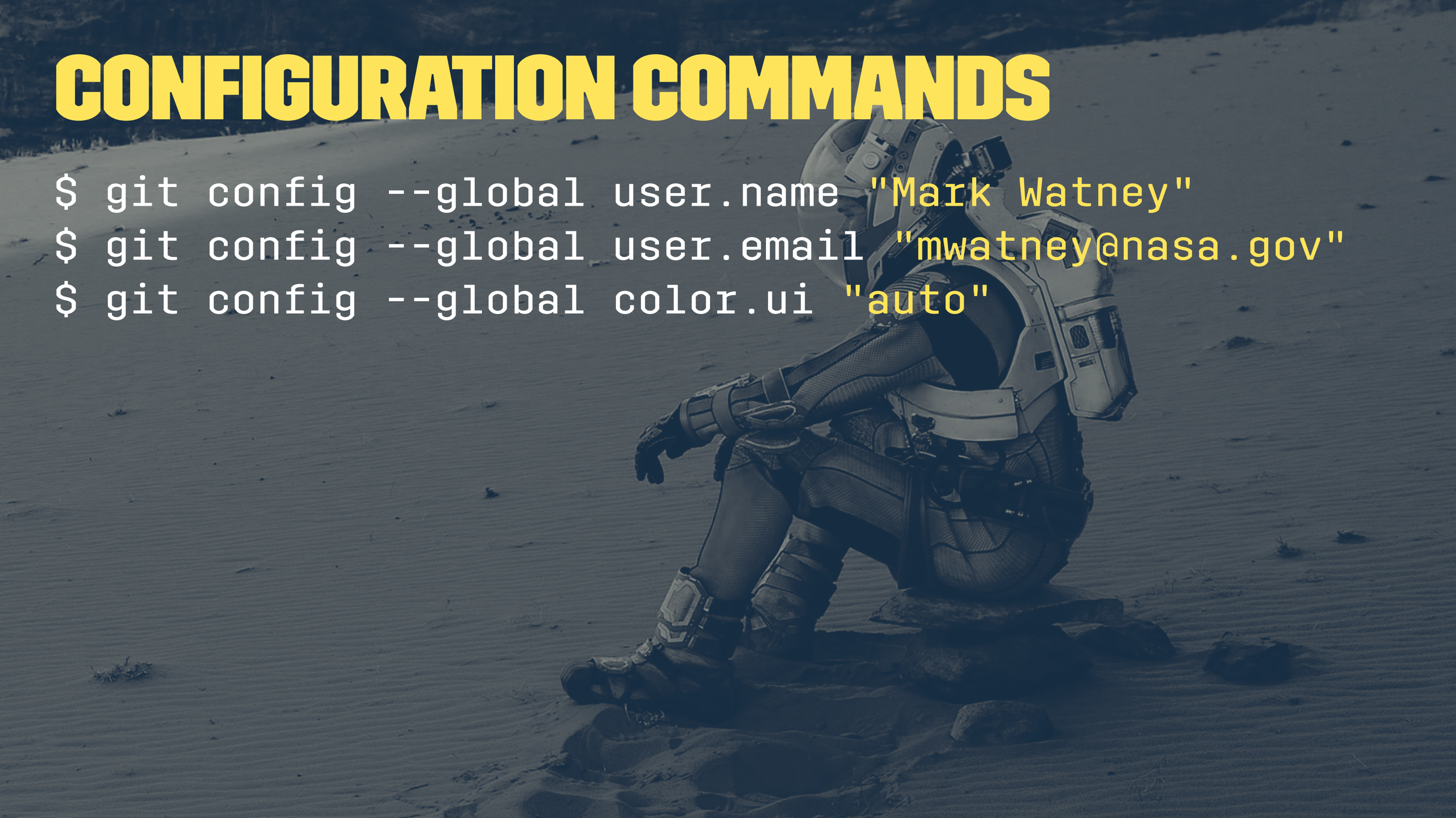
» colorize our output,

» identify our preferred text editor,

» apply settings globally (or not).

CONFIGURATION COMMANDS

```
$ git config --global user.name "Mark Watney"
$ git config --global user.email "mwatney@nasa.gov"
$ git config --global color.ui "auto"
```



Text editor configurations

» Atom:

```
$ git config --global core.editor "atom --wait"
```

» VS Code

```
$ git config --global core.editor "code --wait"
```

» Notepad++ (Windows)

```
$ git config --global core.editor "'c:/program
files/Notepad++/notepad++.exe' -multiInst -
notabbar -nosession -noPlugin"
```

PROXY

```
$ git config --global http.proxy proxy-url
$ git config --global https.proxy proxy-url
```

To disable the proxy, use

```
$ git config --global --unset http.proxy
$ git config --global --unset https.proxy
```

CHECK YOUR SETTINGS

```
$ git config --list

user.name=Christopher Fonnesbeck
user.email=fonnesbeck@gmail.com
color.ui=auto
core.excludesfile=/Users/fonnescj/.gitignore_global
core.editor=nvim
alias.prune=fetch --prune
alias.co=checkout
alias.hist=log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short
alias.switch=!legit switch "$@"
alias.branches=!legit branches
alias.sprout=!legit sprout "$@"
alias.unpublish=!legit unpublish "$@"
```

GIT HELP AND MANUAL

```
$ git config -h
$ git config --help
```

CREATING REPOSITORIES


Create a project directory ...
$ mkdir escape_mars
$ cd escape_mars

Create a project directory ...

```
$ mkdir escape_mars
$ cd escape_mars
```

... then initialize a repository called escape_mars.

```
$ git init
```

WHAT'S INSIDE?

```
$ ls
```

WHAT'S INSIDE?

```
$ ls
```

But if we add the -a flag to show everything ...

```
$ ls -a

.    ..   .git
```

PROJECT STATUS

```
$ git status

# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git add" to track)
```

PLACES TO CREATE GIT REPOSITORIES

Consider the following sequence of commands:

```
cd                      # return to home directory
mkdir escape_mars       # make a new directory escape_mars
cd escape_mars          # go into escape_mars
git init                # make the escape_mars directory a Git repository
mkdir potatoes          # make a sub-directory escape_mars/potatoes
cd potatoes             # go into escape_mars/potatoes
git init                # make the potatoes sub-directory a Git repository
```

» Why is it a bad idea to do this?

» How can Mark Watney undo his last git init?

TRACKING CHANGES

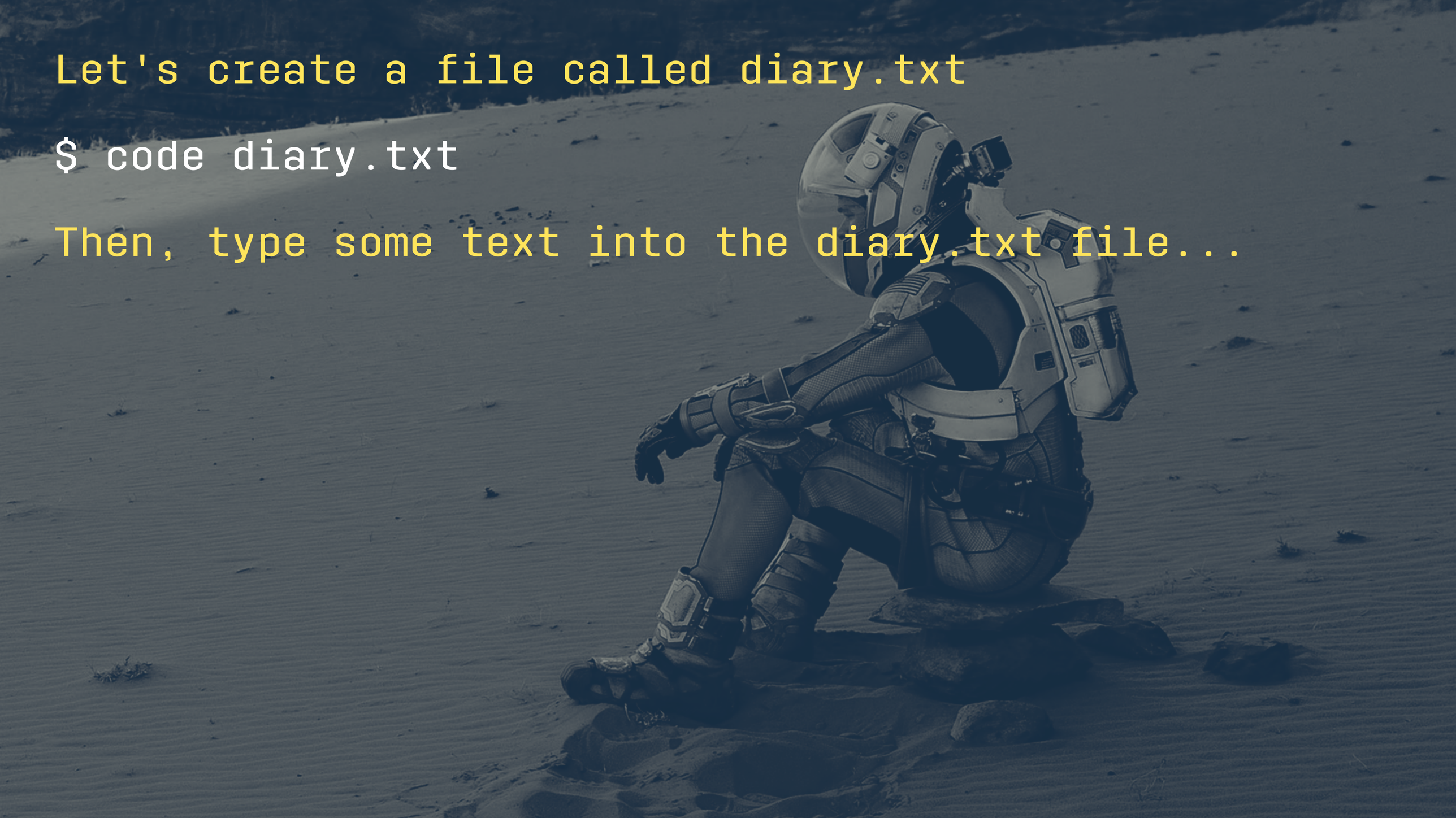
» "How do I record changes in Git?"

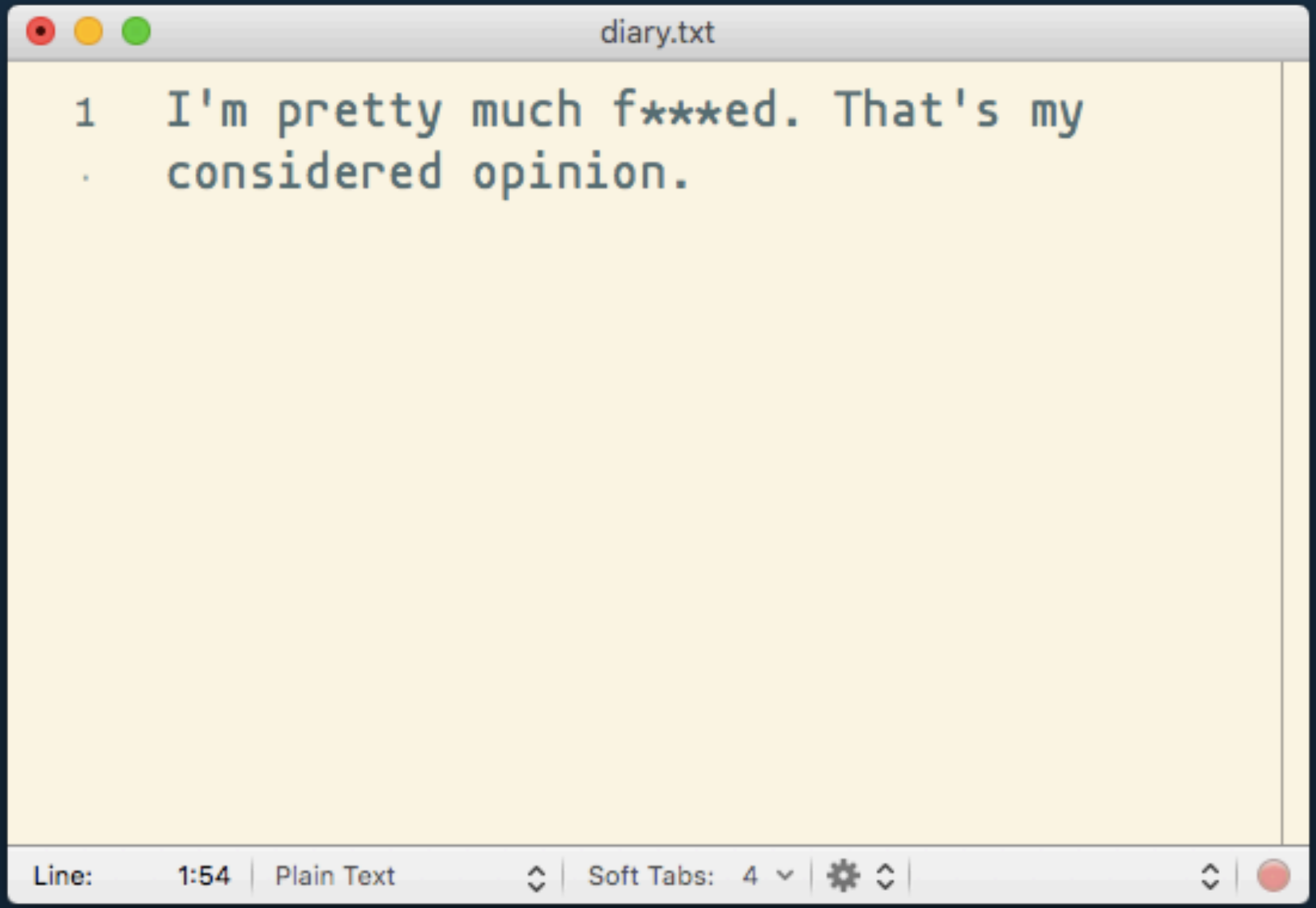
» "How do I record notes about what changes I made and why?"

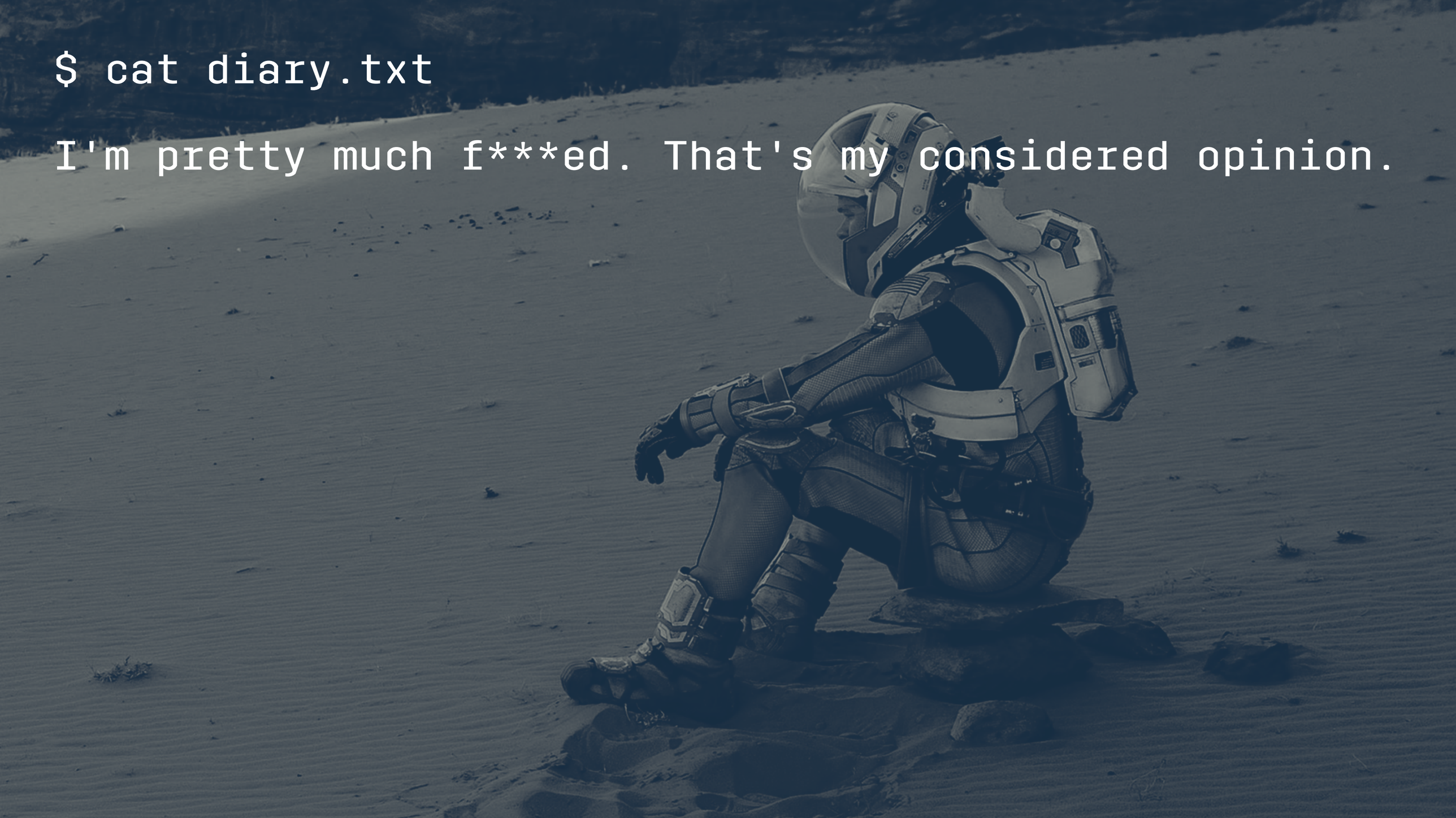
Let's create a file called diary.txt

```
$ code diary.txt
```

Then, type some text into the diary.txt file...




diary.txt
1 I'm pretty much f***ed. That's my
considered opinion.
Line: 1:54
Plain Text
Soft Tabs: 4


$ cat diary.txt
I'm pretty much f***ed. That's my considered opinion.

CHECK STATUS

```
$ git status

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

ADDING FILES

```
$ git add diary.txt
```

and then check that the right thing happened ...

```
$ git status

# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#   new file:   diary.txt
#
```

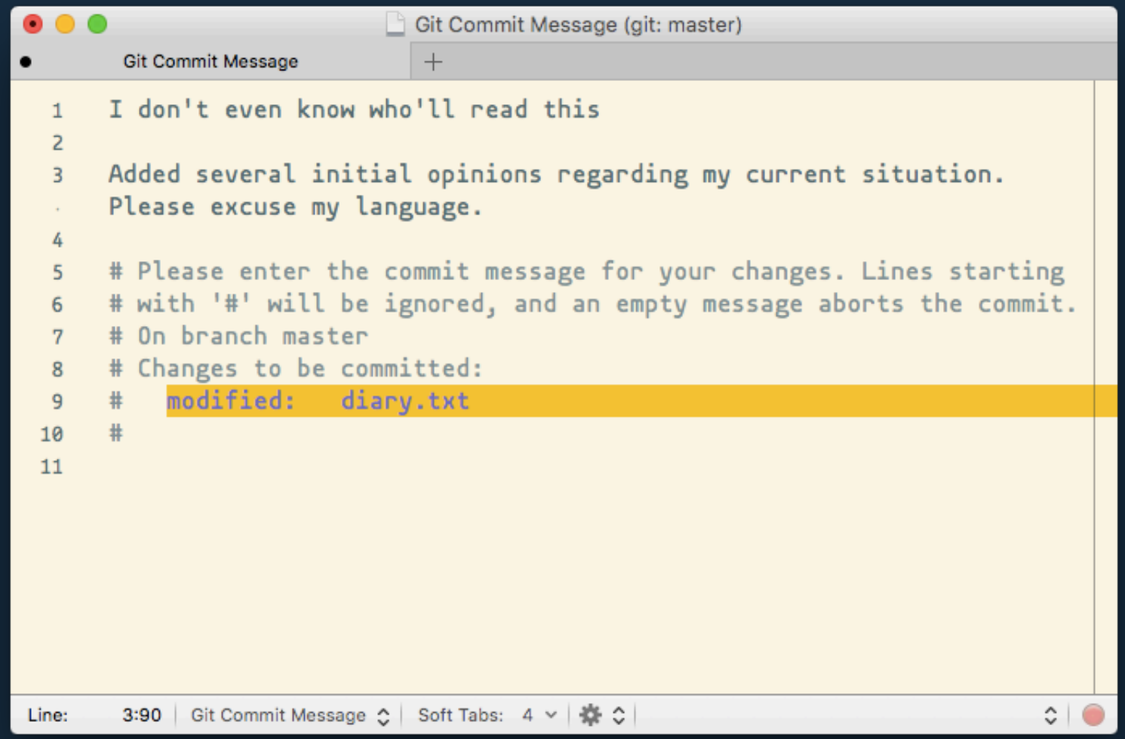
COMMITTING CHANGES

```
$ git commit -m "I don't even know who'll read this"

[master (root-commit) 643a892] I don't even know who'll read this
 1 file changed, 1 insertion(+)
 create mode 100644 diary.txt
```

GOOD COMMIT MESSAGES

» start with a brief (<50 characters) summary of changes made in the commit.

» additional notes may follow a blank line after the summary



If we run git status now:

```
$ git status

# On branch master
nothing to commit, working directory clean
```

PROJECT HISTORY

```
$ git log

commit 643a89258344ec9a02ab8e85e1493945a9b71079
Author: Mark Watney <mwatney@nasa.gov>
Date:   Sun Aug 14 14:53:07 2016 -0700

    I don't even know who'll read this
```

WHERE ARE MY CHANGES?

Have a look at the contents of your project directory now:

```
$ ls
```

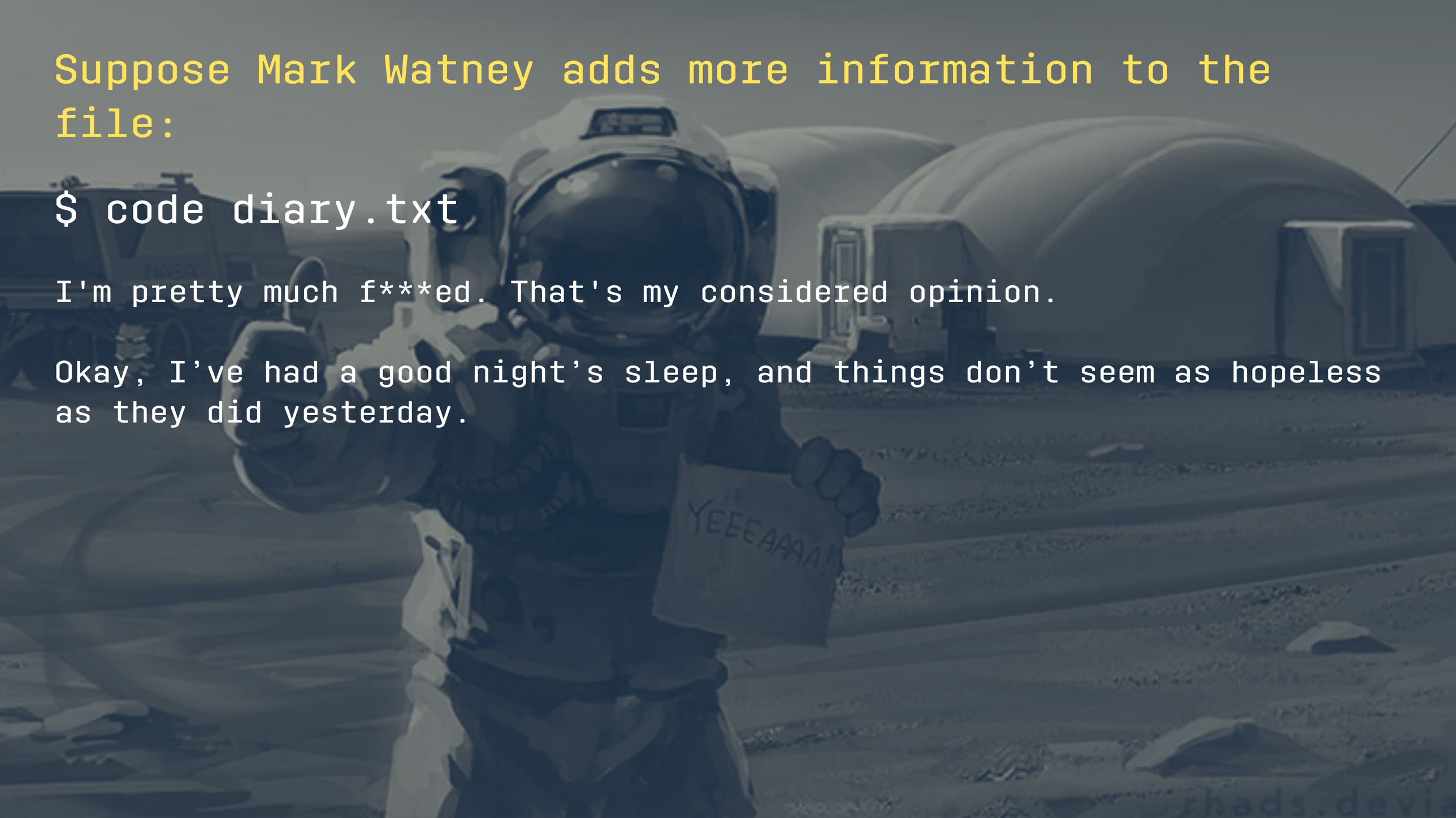
WHERE ARE MY CHANGES?

Have a look at the contents of your project directory now:

```
$ ls

diary.txt
```

```
$ ls .git

COMMIT_EDITMSG config      hooks  info  objects
HEAD           description index  logs  refs
```

Git saves information about files' history in the special .git directory mentioned earlier so that our filesystem doesn't become cluttered.



Suppose Mark Watney adds more information to the file:

```
$ code diary.txt
```

I'm pretty much f***ed. That's my considered opinion.

Okay, I've had a good night's sleep, and things don't seem as hopeless as they did yesterday.

When we run git status now...

```
$ git status
```

When we run git status now...

```
$ git status
```

...it tells us that a file it already knows about has been modified:

```
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:   diary.txt

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

REVIEWING CHANGES

```
$ git diff

diff --git a/diary.txt b/diary.txt
index 218b2e6..1004f7e 100644
--- a/diary.txt
+++ b/diary.txt
@@ -1 +1,4 @@
 I'm pretty much f***ed. That's my considered opinion.
+
+Okay, I've had a good night's sleep, and things don't seem as hopeless
+as they did yesterday.
```

```
diff --git a/diary.txt b/diary.txt
index 218b2e6..1004f7e 100644
--- a/diary.txt
+++ b/diary.txt
@@ -1 +1,4 @@
 I'm pretty much f***ed. That's my considered opinion.
+
+Okay, I've had a good night's sleep, and things don't seem as hopeless
+as they did yesterday.
```

The first line tells us that Git is producing output similar to the Unix diff command, comparing the old and new versions of the file.

```
diff --git a/diary.txt b/diary.txt
index 218b2e6..1004f7e 100644
--- a/diary.txt
+++ b/diary.txt
@@ -1 +1,4 @@
 I'm pretty much f***ed. That's my considered opinion.
+
+Okay, I've had a good night's sleep, and things don't seem as hopeless
+as they did yesterday.
```

The second line tells exactly which versions of the file Git is comparing.

218b2e6 and 1004f7e are unique labels for those versions.

```
diff --git a/diary.txt b/diary.txt
index 218b2e6..1004f7e 100644
--- a/diary.txt
+++ b/diary.txt
@@ -1 +1,4 @@
 I'm pretty much f***ed. That's my considered opinion.
+
+Okay, I've had a good night's sleep, and things don't seem as hopeless
+as they did yesterday.
```

The third and fourth lines are a legend for symbols used to indicate each file.

```
diff --git a/diary.txt b/diary.txt
index 218b2e6..1004f7e 100644
--- a/diary.txt
+++ b/diary.txt
@@ -1 +1,4 @@
 I'm pretty much f***ed. That's my considered opinion.
+
+Okay, I've had a good night's sleep, and things don't seem as hopeless
+as they did yesterday.
```

The remaining lines show us the actual differences and the lines on which they occur.

COMMITTING CHANGES

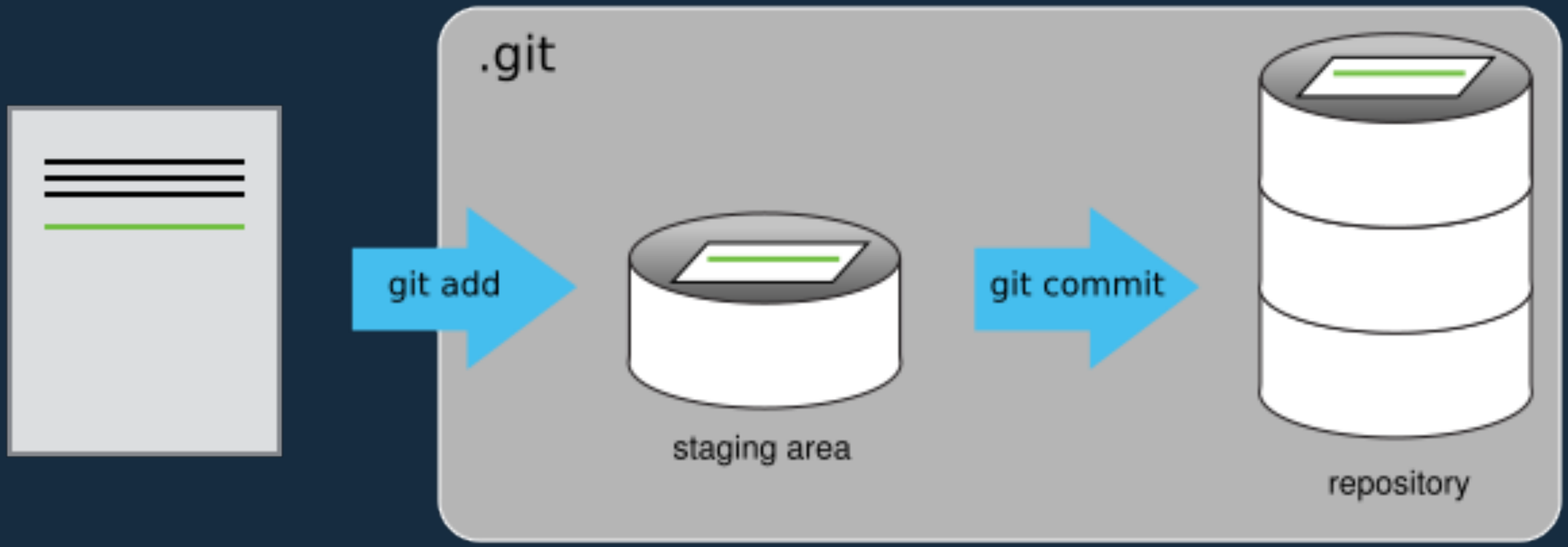
```
$ git commit -m 'Added commentary of second day on Mars'
```

COMMITTING CHANGES

```
$ git commit -m 'Added commentary of second day on Mars'

On branch master
Changes not staged for commit:
    modified:   diary.txt

no changes added to commit
```

```
$ git add diary.txt
$ git commit -m 'Added commentary of second day on Mars'

[master c816814] Added commentary of second day on Mars
 1 file changed, 3 insertions(+)
```

STAGING AREA



ADDING AND COMMITTING

All in one step:

```
$ git commit -a
```

ADD ANOTHER LINE

$ code diary.txt

I'm pretty much f***ed. That's my considered opinion.

Okay, I've had a good night's sleep, and things don't seem as hopeless as they did yesterday.

Of course, I don't have any plan for surviving four years on one year of food.

$ git diff

```
diff --git a/diary.txt b/diary.txt
index 1004f7e..b79ef50 100644
--- a/diary.txt
+++ b/diary.txt
@@ -2,3 +2,6 @@ I'm pretty much f***ed. That's my considered opinion.

 Okay, I've had a good night's sleep, and things don't seem
 as hopeless as they did yesterday.
+
+Of course, I don't have any plan for surviving four years on one
+year of food.
```

```
$ git add diary.txt
```

```
$ git add diary.txt
$ git diff
```

DIFFING WITH STAGED CHANGES

```
$ git diff --staged

index 1004f7e..b79ef50 100644
--- a/diary.txt
+++ b/diary.txt
@@ -2,3 +2,5 @@ I'm pretty much f***ed. That's my considered opinion.

 Okay, I've had a good night's sleep, and things don't seem as hopeless
 as they did yesterday.
+
+Of course, I don't have any plan for surviving four years on one year of food.
```

```
$ git commit -m "Added evaluation of provisions"

[master 4db2cce] Added evaluation of provisions
 1 file changed, 2 insertions(+)
```

```
$ git commit -m "Added evaluation of provisions"

[master 4db2cce] Added evaluation of provisions
 1 file changed, 2 insertions(+)

check our status ...

$ git status

# On branch master
nothing to commit, working directory clean
```

CHECK THE LOG

```
$ git log

commit 4db2cce13e52ee653cdabcc7a52c5a6b3b7cb2b3
Author: Mark Watney <mwatney@nasa.gov>
Date:   Sun Aug 14 15:00:30 2016 -0700

    Added evaluation of provisions

commit c81681408c0b6b4b69ecf5e3ca4413863c8bbb73
Author: Mark Watney <mwatney@nasa.gov>
Date:   Sun Aug 14 14:57:45 2016 -0700

    Added commentary of second day on Mars

commit 643a89258344ec9a02ab8e85e1493945a9b71079
Author: Mark Watney <mwatney@nasa.gov>
Date:   Sun Aug 14 14:53:07 2016 -0700

    I don't even know who'll read this
```

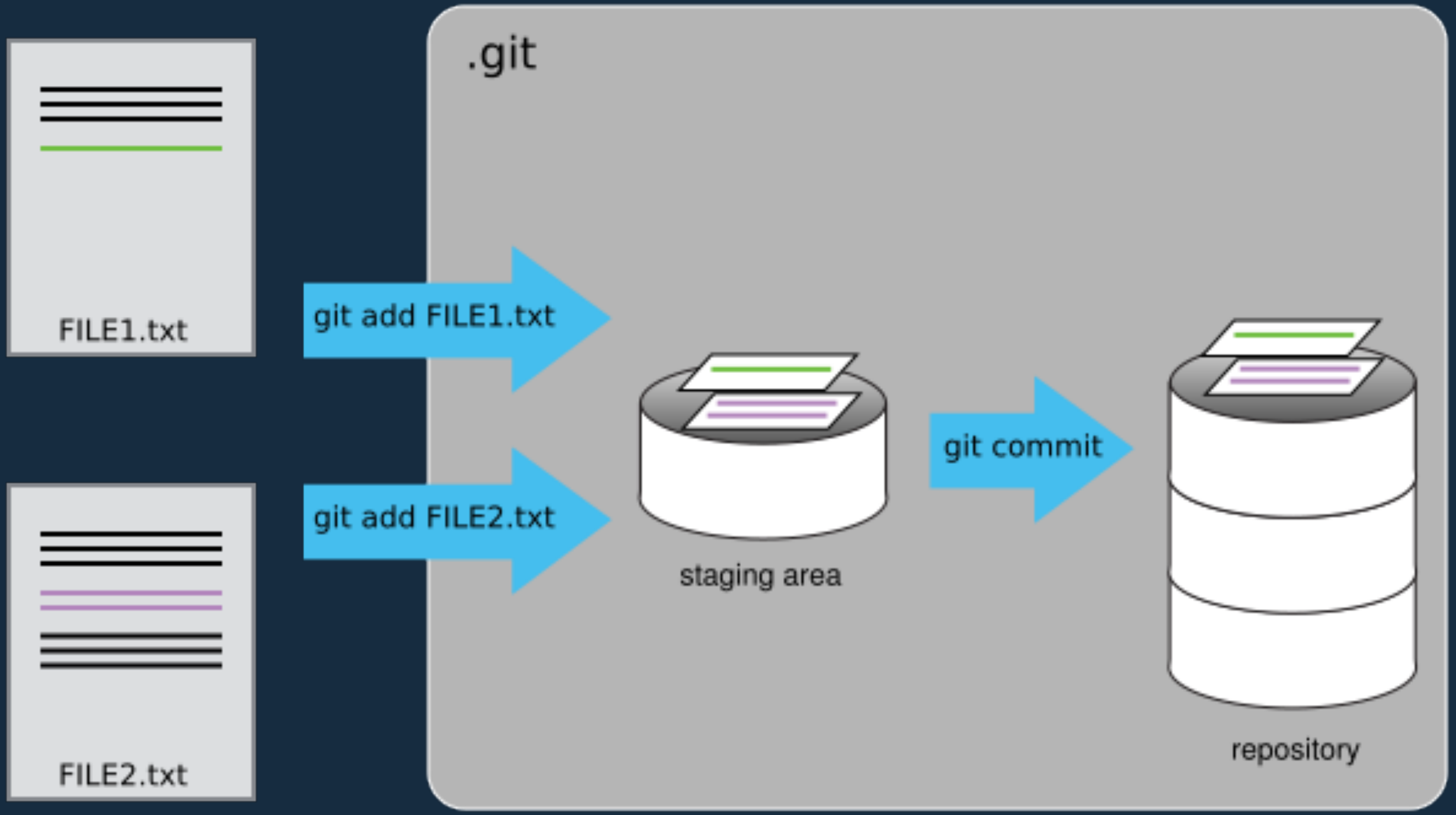
PAGING THE LOG

When the output of git log is too long to fit in your screen, git uses a pager to split it into pages of the size of your screen.

» To get out of the pager, press q.

» To move to the next page, press the space bar.

» To search for some_word in all pages, type
/some_word
and navigate through matches pressing n.


.git
FILE1.txt
FILE2.txt
git add FILE1.txt
git add FILE2.txt
staging area
git commit
repository

CHOOSING A COMMIT MESSAGE

Which of the following commit messages would be most appropriate for the last commit made to diary.txt?

1. "Changes"
2. "Added line 'Of course, I don't have any plan for surviving four years on one year of food.' to diary.txt"
3. "Added evaluation of provisions"

COMMITTING CHANGES TO GIT

Which command(s) below would save the changes of myfile.txt to my local Git repository?

1. $ git commit -m "my recent changes"

2. $ git init myfile.txt
   $ git commit -m "my recent changes"

3. $ git add myfile.txt
   $ git commit -m "my recent changes"

4. $ git commit -m myfile.txt "my recent changes"

COMMITTING MULTIPLE FILES

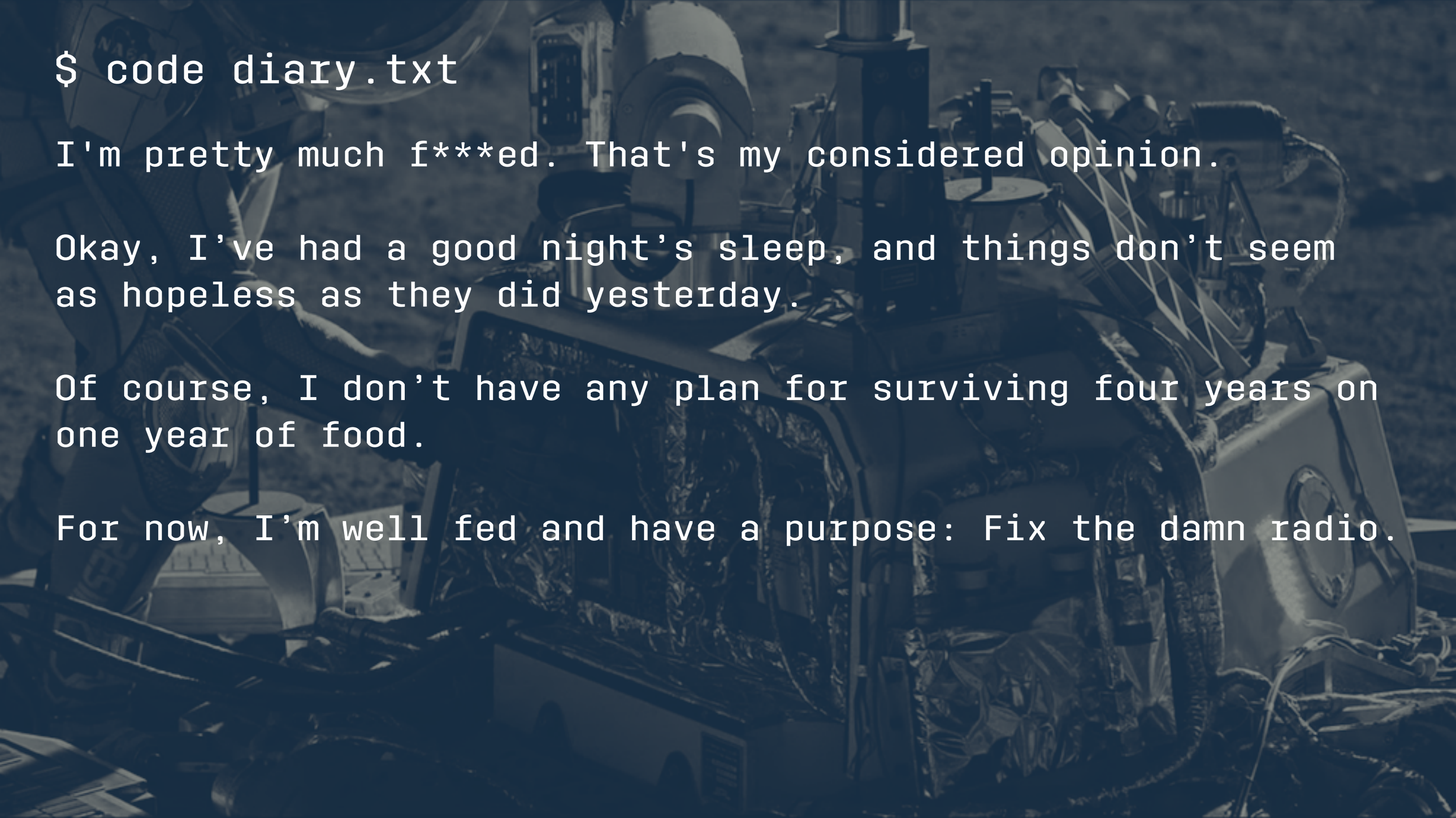
The staging area can hold changes from any number of files that you want to commit as a single snapshot.

1. Add some text to diary.txt noting your decision to reconfigure the mars rover
2. Create a new file rover.txt with your initial thoughts on reconfiguring the rover
3. Add changes from both files to the staging area, and commit those changes.

EXPLORING HISTORY

HEAD

You can refer to the most recent commit of the working directory by using the identifier HEAD.



$ code diary.txt

I'm pretty much f***ed. That's my considered opinion.

Okay, I've had a good night's sleep, and things don't seem as hopeless as they did yesterday.

Of course, I don't have any plan for surviving four years on one year of food.

For now, I'm well fed and have a purpose: Fix the damn radio.

DIFFING WITH HEAD

```
$ git diff HEAD diary.txt

diff --git a/diary.txt b/diary.txt
index b79ef50..e2287f6 100644
--- a/diary.txt
+++ b/diary.txt
@@ -4,3 +4,5 @@ Okay, I've had a good night's sleep, and things don't seem as hopeless
 as they did yesterday.

 Of course, I don't have any plan for surviving four years on one year of food.
+
+For now, I'm well fed and have a purpose: Fix the damn radio.
```

DIFFING WITH HEAD

```
$ git diff HEAD~1 diary.txt

diff --git a/diary.txt b/diary.txt
index 1004f7e..e2287f6 100644
--- a/diary.txt
+++ b/diary.txt
@@ -2,3 +2,7 @@ I'm pretty much f***ed. That's my considered opinion.

 Okay, I've had a good night's sleep, and things don't seem as hopeless
 as they did yesterday.
+
+Of course, I don't have any plan for surviving four years on one year of food.
+
+For now, I'm well fed and have a purpose: Fix the damn radio.
```

COMPARING WITH PREVIOUS COMMITS

```
$ git diff HEAD~2 diary.txt

diff --git a/diary.txt b/diary.txt
index 218b2e6..e2287f6 100644
--- a/diary.txt
+++ b/diary.txt
@@ -1 +1,8 @@
 I'm pretty much f***ed. That's my considered opinion.
+
+Okay, I've had a good night's sleep, and things don't seem as hopeless
+as they did yesterday.
+
+Of course, I don't have any plan for surviving four years on one year of food.
+
+For now, I'm well fed and have a purpose: Fix the damn radio.
```

DIFFING BY HASH

```
$ git diff c81681408c0b6b4b69ecf5e3ca4413863c8bbb73 diary.txt

diff --git a/diary.txt b/diary.txt
index 1004f7e..e2287f6 100644
--- a/diary.txt
+++ b/diary.txt
@@ -2,3 +2,7 @@ I'm pretty much f***ed. That's my considered opinion.

 Okay, I've had a good night's sleep, and things don't seem as hopeless
 as they did yesterday.
+
+Of course, I don't have any plan for surviving four years on one year of food.
+
+For now, I'm well fed and have a purpose: Fix the damn radio.
```

$ git diff c816 diary.txt

```
diff --git a/diary.txt b/diary.txt
index 1004f7e..e2287f6 100644
--- a/diary.txt
+++ b/diary.txt
@@ -2,3 +2,7 @@ I'm pretty much f***ed. That's my considered opinion.

 Okay, I've had a good night's sleep, and things don't seem as hopeless
 as they did yesterday.
+
+Of course, I don't have any plan for surviving four years on one year of food.
+
+For now, I'm well fed and have a purpose: Fix the damn radio.
```

BAD THINGS HAPPEN

Let's suppose we accidentally overwrite our file:

```
$ code diary.txt
```

```
So I ran into a bunch of problems with my water plan.
```

git status now tells us that the file has been changed ...

```
$ git status

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

... but those changes haven't been staged

REVERTING TO HEAD

```
$ git checkout HEAD diary.txt
$ cat diary.txt

I'm pretty much f***ed. That's my considered opinion.

Okay, I've had a good night's sleep, and things don't seem as hopeless
as they did yesterday.

Of course, I don't have any plan for surviving four years on one year of food.
```

If we want to go back even further,
we can use a commit identifier instead:

```
$ git checkout c816814 diary.txt
```

DON'T LOSE YOUR HEAD

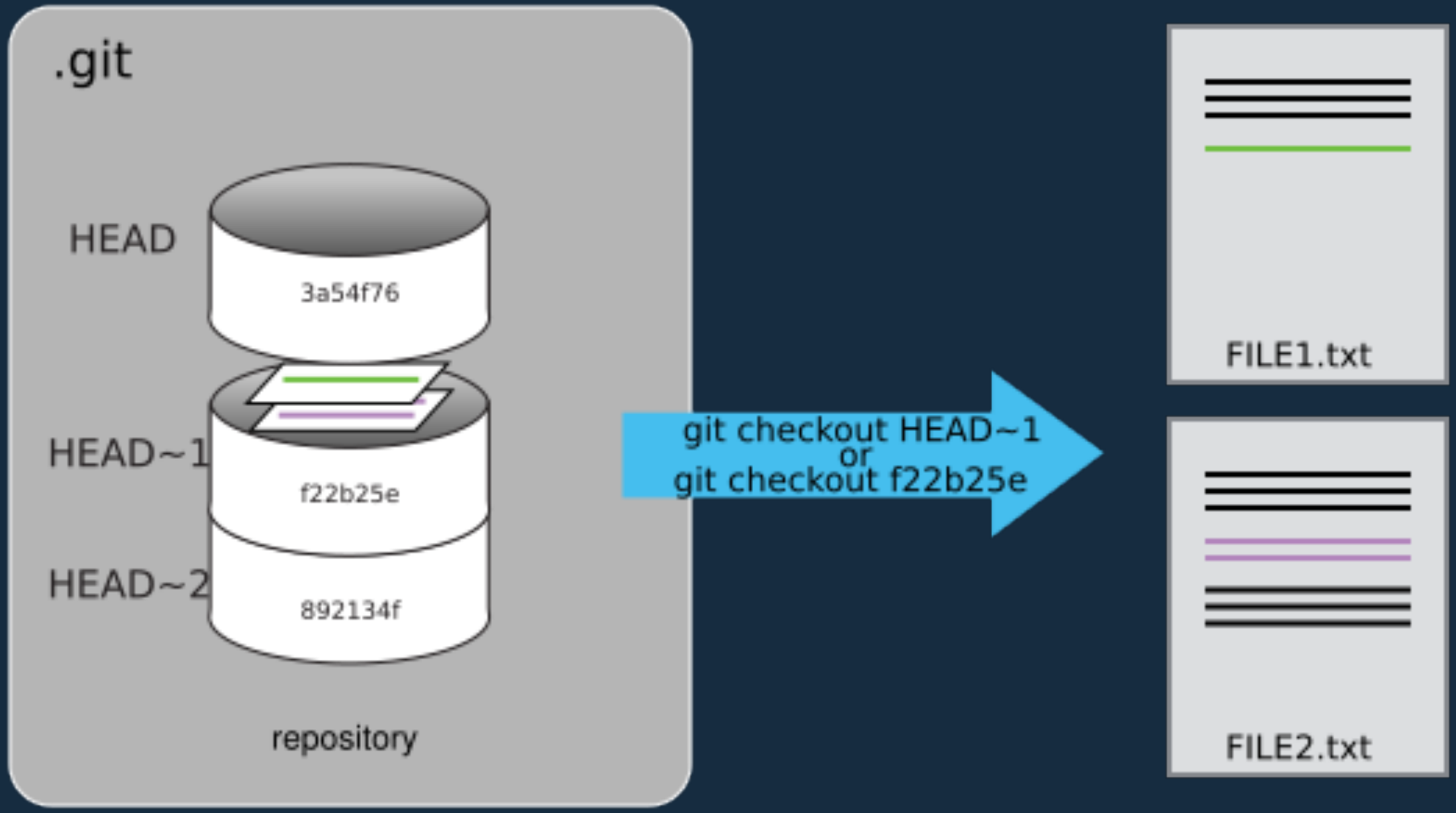
Notice we used

```
$ git checkout c816814 diary.txt
```

to revert diary.txt to its state after the commit c816814.

If you forget diary.txt in that command, git will tell you that

“You are in 'detached HEAD' state.”


.git
HEAD
3a54f76
HEAD~1
f22b25e
HEAD~2
892134f
repository
git checkout HEAD~1
or
git checkout f22b25e
FILE1.txt
FILE2.txt

SIMPLIFYING THE COMMON CASE

If you read the output of git status carefully, you'll see that it includes this hint:

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

The double dash -- is needed to separate the names of the files being recovered from the command itself.

RECOVERING OLDER VERSIONS OF A FILE

Jennifer has made changes to the Python script that she has been working on for weeks, and the modifications she made this morning "broke" the script and it no longer runs.

Luckily, she has been keeping track of her project's versions using Git! Which commands below will let her recover the last committed version of her Python script called data_cruncher.py?

1. `$ git checkout HEAD`
2. `$ git checkout HEAD data_cruncher.py`
3. `$ git checkout HEAD~1 data_cruncher.py`
4. `$ git checkout <unique ID of last commit> data_cruncher.py`

EXPLORE AND SUMMARIZE HISTORIES

After several months, the escape_mars project has more than 50 files.

You would like to find a commit with specific text in diary.txt is modified.

When you type git log, a very long list appeared,

How can you narrow down the search?

EXPLORE AND SUMMARIZE HISTORIES

```
$ git log diary.txt
```

Perhaps some of these commit messages are very ambiguous (e.g. "update files").

How can you search through these files?

EXPLORE AND SUMMARIZE HISTORIES

Both git diff and git log are very useful and they summarize different part of the history for you.

Is that possible to combine both? Let's try the following:

```
$ git log --patch diary.txt
```

```
commit 4db2cce13e52ee653cdabcc7a52c5a6b3b7cb2b3
Author: Chris Fonnesbeck <chris.fonnesbeck@vanderbilt.edu>
Date:   Sun Aug 14 15:00:30 2016 -0700

    Added evaluation of provisions

diff --git a/diary.txt b/diary.txt
index 1004f7e..b79ef50 100644
--- a/diary.txt
+++ b/diary.txt
@@ -2,3 +2,5 @@ I'm pretty much f***ed. That's my considered opinion.

 Okay, I've had a good night's sleep, and things don't seem
 as hopeless as they did yesterday.
+
+Of course, I don't have any plan for surviving four years on
+one year of food.
```

```
commit c81681408c0b6b4b69ecf5e3ca4413863c8bbb73
Author: Chris Fonnesbeck <chris.fonnesbeck@vanderbilt.edu>
Date:   Sun Aug 14 14:57:45 2016 -0700

    Added commentary of second day on Mars

diff --git a/diary.txt b/diary.txt
index 218b2e6..1004f7e 100644
--- a/diary.txt
+++ b/diary.txt
@@ -1 +1,4 @@
 I'm pretty much f***ed. That's my considered opinion.
+
+Okay, I've had a good night's sleep, and things don't seem
+as hopeless as they did yesterday.
```

IGNORING THINGS

INTERMEDIATE OUTPUT FILES

```
$ mkdir potato_results
$ touch a.dat b.dat c.dat potato_results/a.out
  potato_results/b.out
$ git status
```

INTERMEDIATE OUTPUT FILES

```
$ mkdir potato_results
$ touch a.dat b.dat c.dat potato_results/a.out
  potato_results/b.out
$ git status

On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    a.dat
    b.dat
    c.dat
    potato_results/

nothing added to commit but untracked files present (use "git add" to track)
```

.GITIGNORE

```
$ code .gitignore

*.dat
potato_results/
```

$ git status

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

Let's add and commit .gitignore:

```
$ git add .gitignore
$ git commit -m "Add the ignore file"
$ git status

# On branch master
nothing to commit, working directory clean
```

Using .gitignore helps us avoid accidentally adding to the repository files that we don't want to track:

```
$ git add a.dat

The following paths are ignored by one of your .gitignore files:
a.dat
Use -f if you really want to add them.
fatal: no files added
```

```
$ git status --ignored

# On branch master
# Ignored files:
#  (use "git add -f <file>..." to include in what will be committed)
#
#       a.dat
#       b.dat
#       c.dat
#       results/

nothing to commit, working directory clean
```

IGNORING NESTED FILES

Given a directory structure that looks like:

```
results/data
results/plots
```

How would you ignore only results/plots and not results/data?

INCLUDING SPECIFIC FILES

How would you ignore all files with a .data extension in your root directory except for final.data?

INCLUDING SPECIFIC FILES

How would you ignore all files with a .data extension in your root directory except for final.data?

You would add the following two lines to your .gitignore:

```
*.data           # ignore all data files
!final.data      # except final.data
```

REMOTES IN GITHUB

GIT HOSTING SERVICES

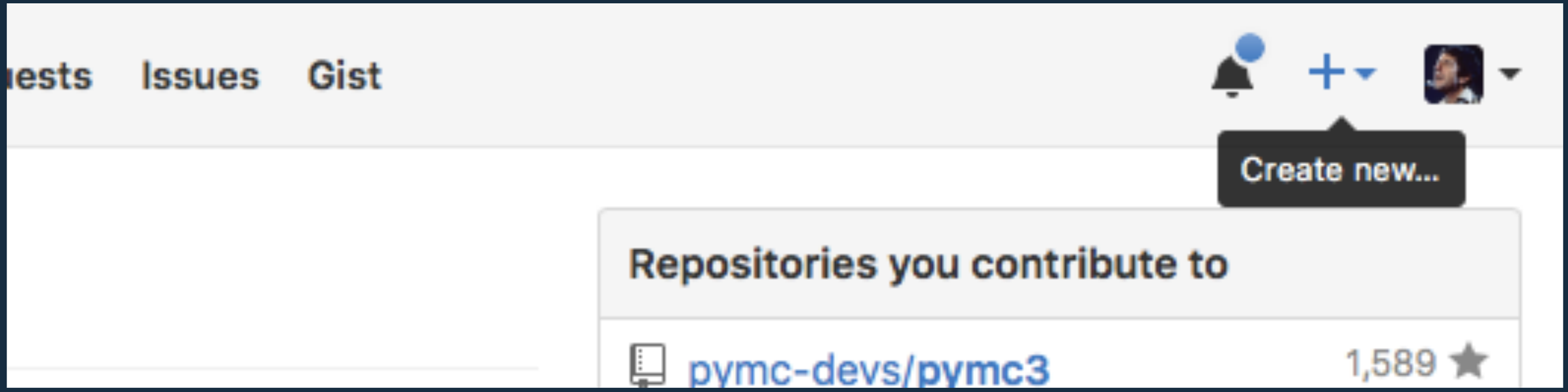
Popular commercial services for hosting Git for collaboration include:

» GitHub

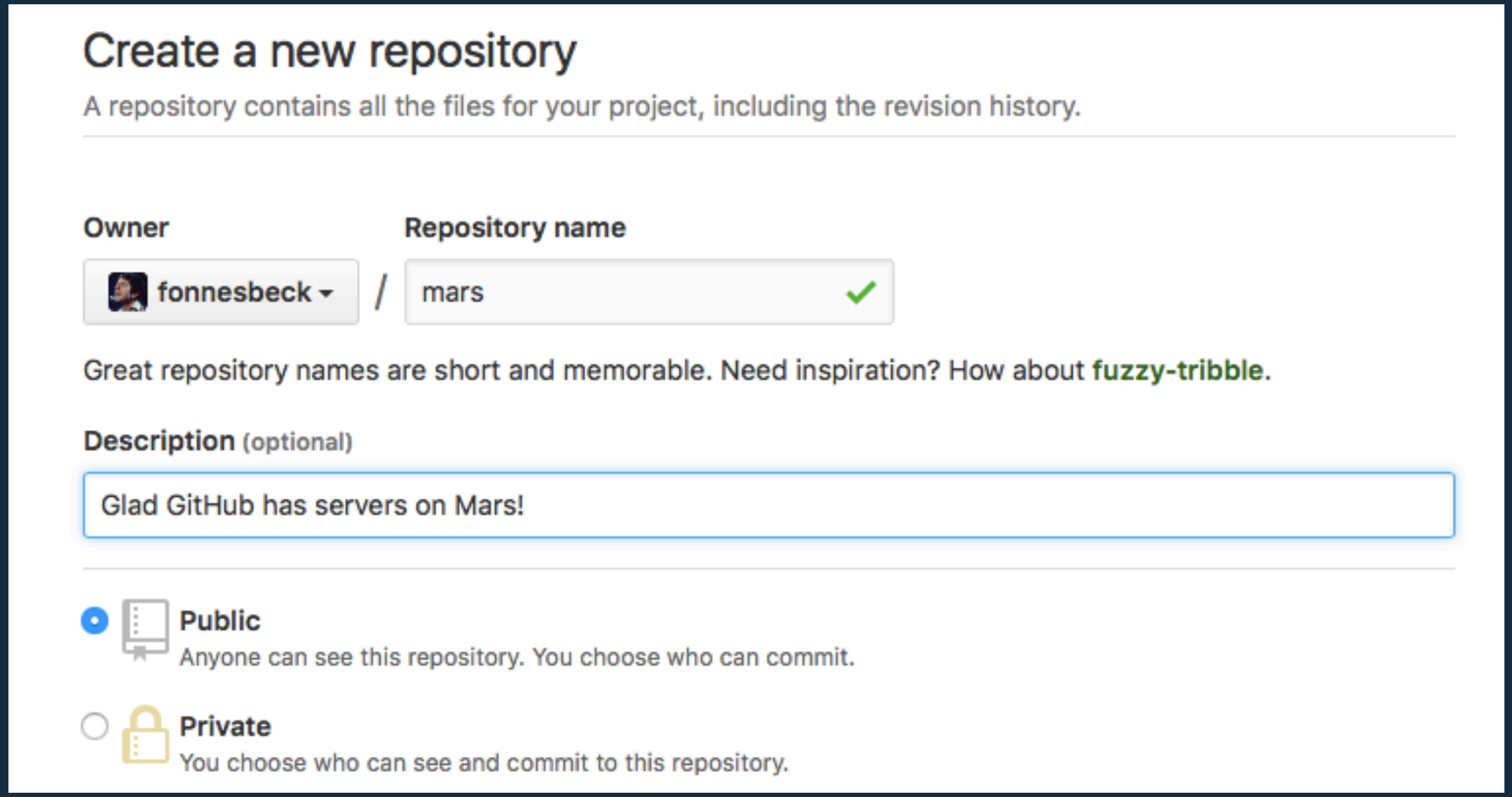
» BitBucket

» GitLab

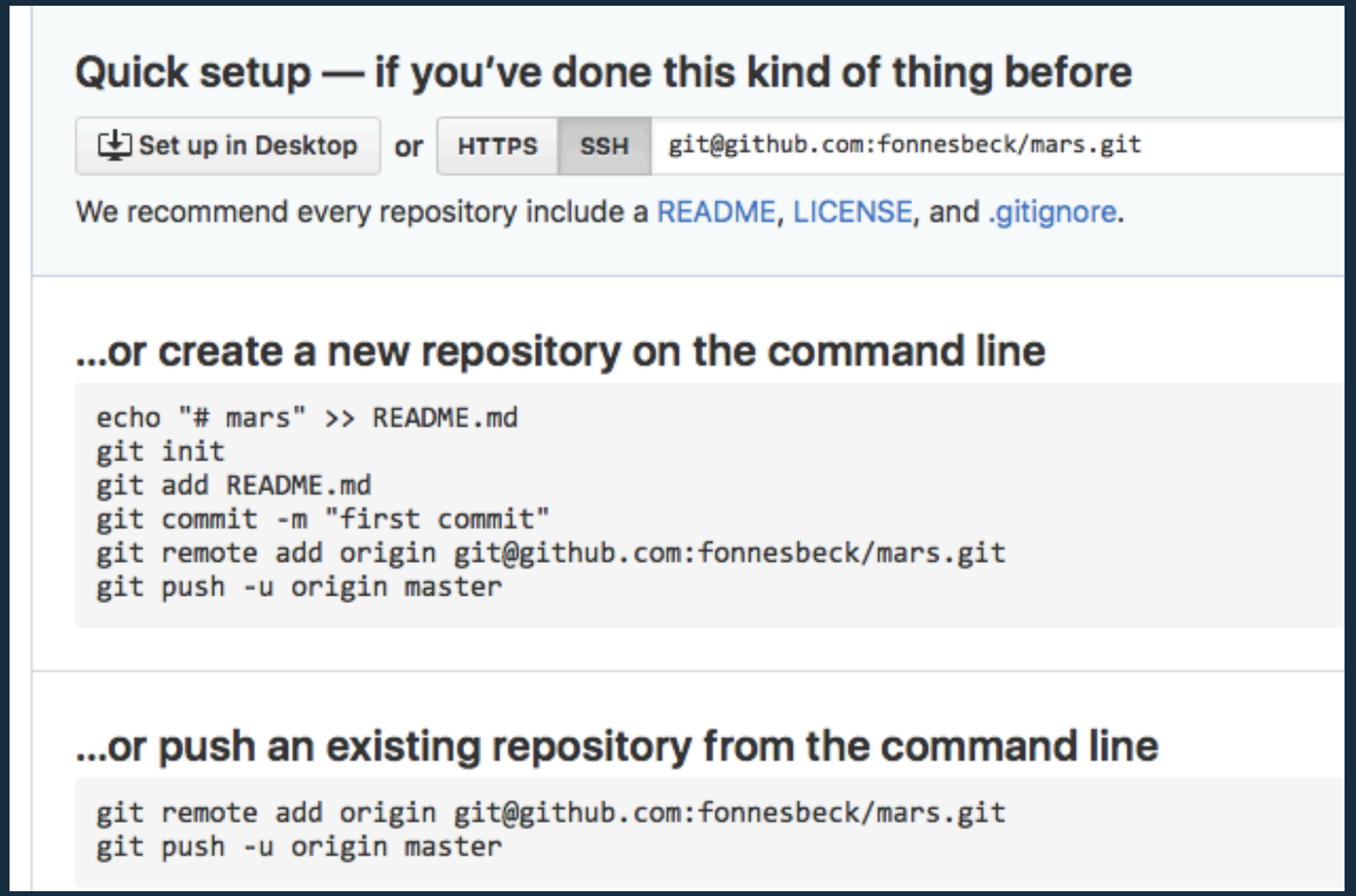
CREATING A REPOSITORY ON GITHUB (STEP 1)

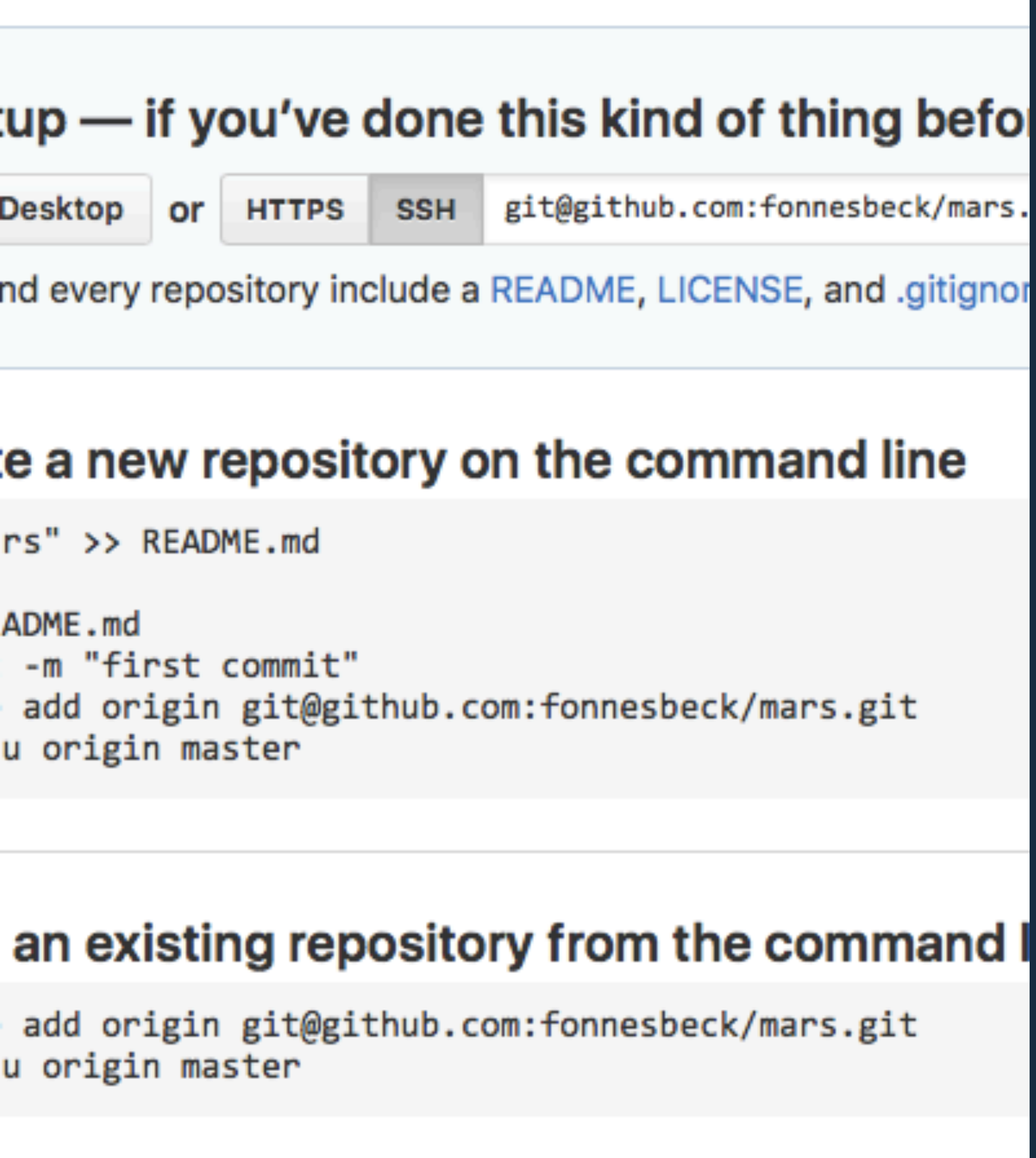


CREATING A REPOSITORY ON GITHUB (STEP 2)



CREATING A REPOSITORY ON GITHUB (STEP 3)

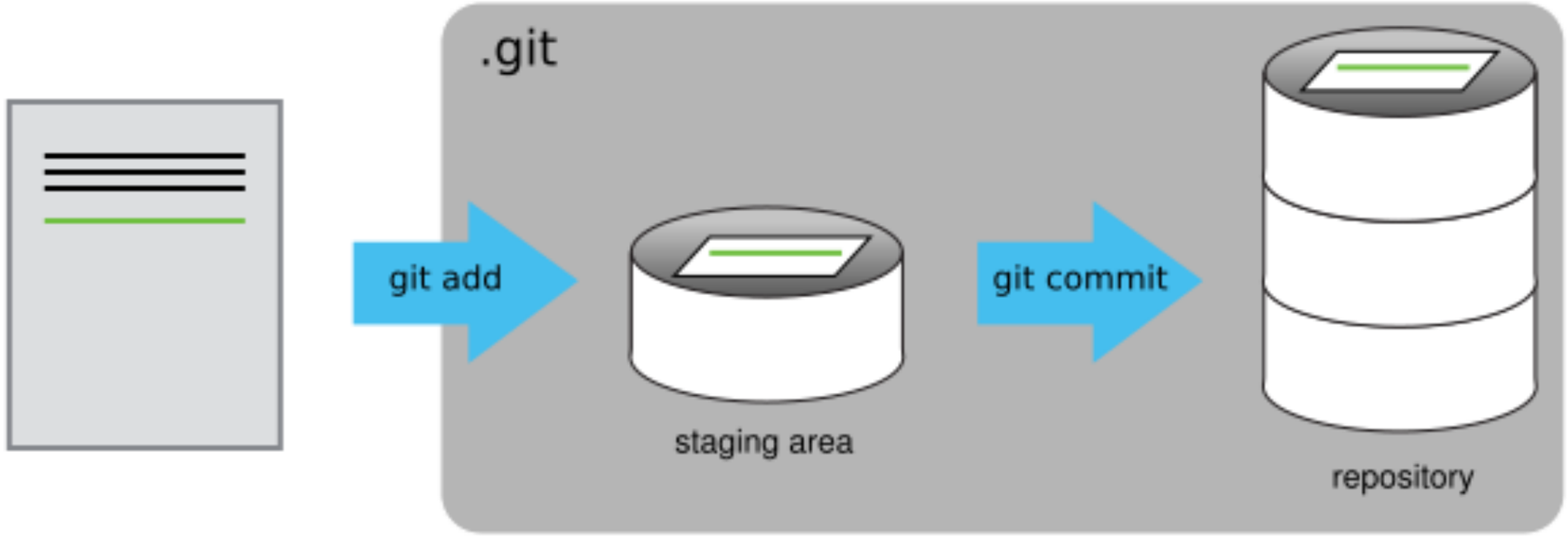




This effectively does the following on GitHub's servers:

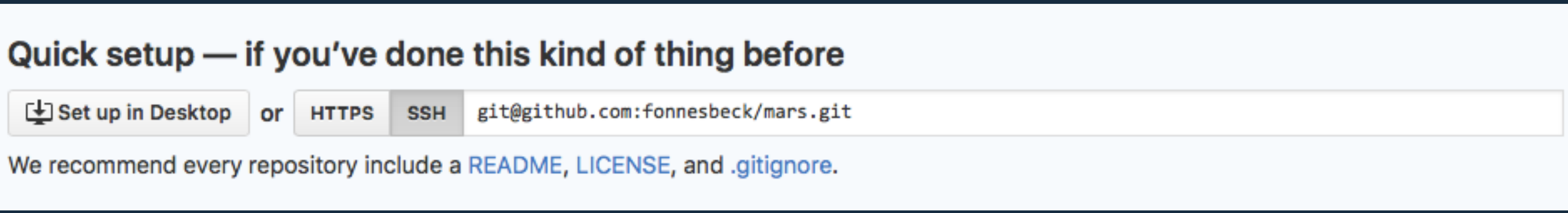
```
$ mkdir mars
$ cd mars
$ git init
```

~/mwatney/mars.git



https://github.com/mwatney/mars.git

The home page of the repository on GitHub includes a copyable URL of the repo...



HTTPS VS. SSH

We use HTTPS here because it does not require additional configuration.

After the workshop you may want to set up SSH access, which is a bit more secure, and does not require repeated entry of passwords.

SPECIFYING A REMOTE REPO

Copy the GitHub repo URL from the browser, and run ...

```
$ git remote add origin https://github.com/mwatney/mars.git
```

Check that the command has worked ...

```
$ git remote -v
```

```
origin   https://github.com/mwatney/mars.git (push)
origin   https://github.com/mwatney/mars.git (fetch)
```

PUSHING CHANGES

```
$ git push origin master

Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 821 bytes, done.
Total 9 (delta 2), reused 0 (delta 0)
To https://github.com/mwatney/mars
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

PULLING CHANGES

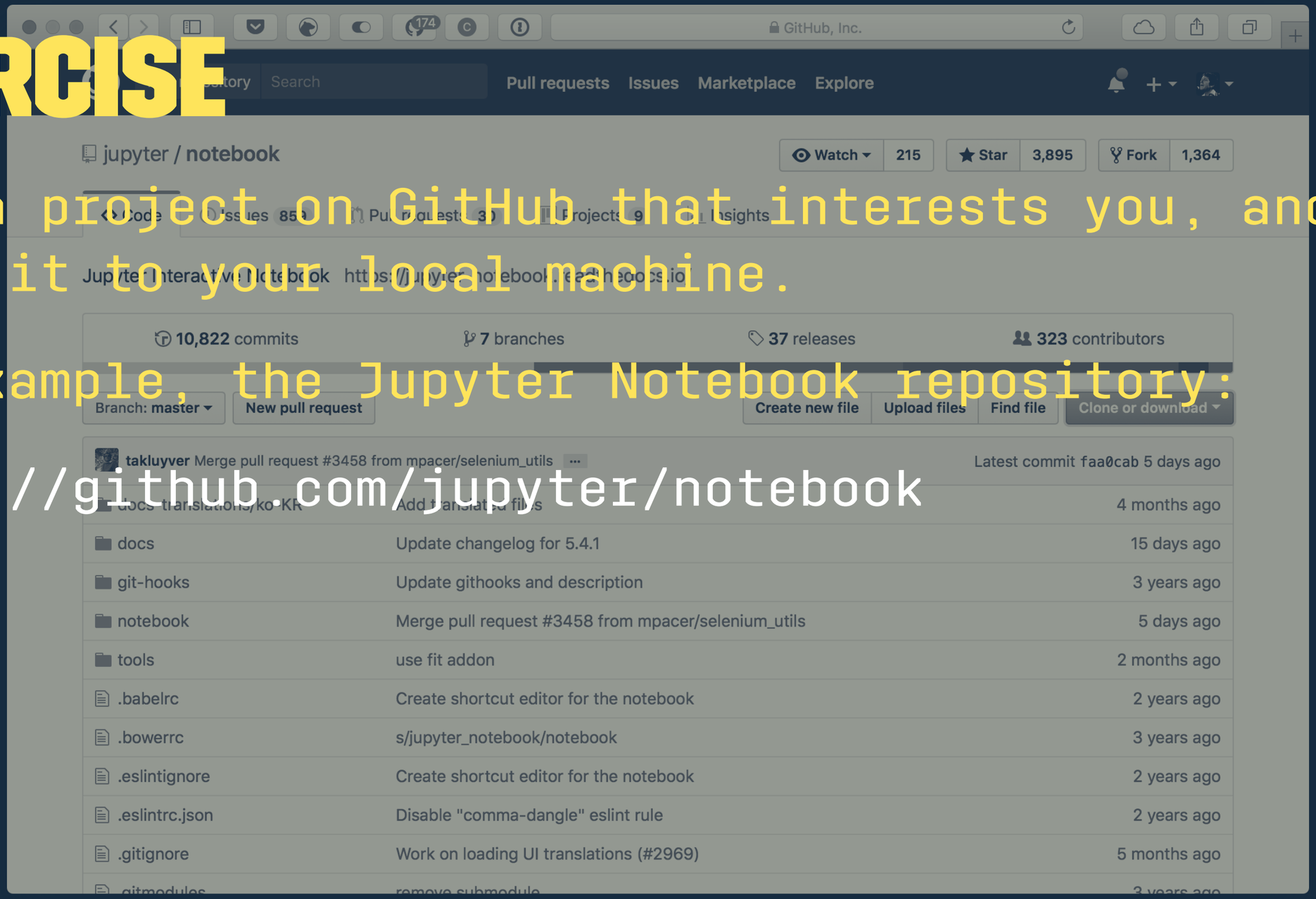
```
$ git pull origin master

From https://github.com/mwatney/mars
 * branch            master     -> FETCH_HEAD
Already up-to-date.
```

EXERCISE

Find a project on GitHub that interests you, and clone it to your local machine.

For example, the Jupyter Notebook repository:

https://github.com/jupyter/notebook



CONFLICTS

```
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1)
Unpacking objects: 100% (3/3), done.
From https://github.com/mwatney/mars
 * branch            master     -> FETCH_HEAD
Auto-merging diary.txt
CONFLICT (content): Merge conflict in diary.txt
Automatic merge failed; fix conflicts and then commit the result.
```

COLLABORATION

4db2cce 2 days ago

Edit this file

Raw | Blame | History

m as hopeless

on one year of food.

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs

mars / diary.txt or cancel

Edit file | Preview changes

```
1  I'm pretty much f***ed. That's my considered opinion.
2
3  Okay, I've had a good night's sleep, and things don't seem as hopeless
4  as they did yesterday.
5
6  Of course, I don't have any plan for surviving four years on one year of food.
7
8  Mark, we are coming to get you!
```

Commit changes

Added message to stranded astronaut

Add an optional extended description...

- Commit directly to the `master` branch.
- Create a **new branch** for this commit and start a pull request. Learn more ab

Commit changes | Cancel

LOCAL CHANGE

```
$ echo "Im still cowering in the rover, but Ive had time to
        think." >> diary.txt
$ git add diary.txt
$ git commit -m "Progress report on rover reconfiguration"

[master 2104602] Progress report on rover reconfiguration
 1 file changed, 1 insertion(+)
```

PUSH LOCAL CHANGES

```
$ git push

To git@github.com:mwatney/mars.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:mwatney/mars.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

RESOLVING THE CONFLICT

Let's start by pulling:

```
$ git pull

remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:mwatney/mars
   6326f7d..9fca57b  master     -> origin/master
Auto-merging diary.txt
CONFLICT (content): Merge conflict in diary.txt
Automatic merge failed; fix conflicts and then commit the result.
```

RESOLVING THE CONFLICT

```
$ cat diary.txt

I'm pretty much f***ed. That's my considered opinion.

Okay, I've had a good night's sleep, and things don't seem as hopeless
as they did yesterday.

Of course, I don't have any plan for surviving four years on one year of food.
<<<<<<< HEAD
Im still cowering in the rover, but Ive had time to think.
=======

Mark, we are coming to get you!
>>>>>>> 9fca57bfa92c4e4aa23566695f5bd21535addb87
```

MERGING CHANGES

$ code diary.txt

I'm pretty much f***ed. That's my considered opinion.

Okay, I've had a good night's sleep, and things don't seem as hopeless as they did yesterday.

Of course, I don't have any plan for surviving four years on one year of food.

Now that NASA can talk to me, they won't shut the hell up.

COMPLETE MERGE

```
$ git add diary.txt
$ git status

On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
  (use "git pull" to merge the remote branch into yours)
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

    modified:   diary.txt
```

COMPLETE MERGE

```
$ git commit -m "Merging changes from GitHub"

[master e536f96] Merging changes from GitHub
```

PUSH MERGED BRANCH

```
$ git push

Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 776 bytes | 0 bytes/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local objects.
To git@github.com:mwatney/mars.git
   9fca57b..e536f96  master -> master
```


NASA
YEEEAAAA!