Version Control with Git

March 2017, Winona State University

- Lesson plan taken from Software Carpentry:
  `http://swcarpentry.github.io/git-novice/`
- This presentation is on GitHub:
  - Direct download of PDF file:
    `https://github.com/christopherphan/Winona-SC-git/raw/master/slides/git-winona-slides.pdf`
  - GitHub repository:
    `https://github.com/christopherphan/Winona-SC-git/`

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.

[1] http://swcarpentry.github.io/git-novice/

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.

[1] http://swcarpentry.github.io/git-novice/

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.
- Possible aproaches:

[1] http://swcarpentry.github.io/git-novice/

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.
- Possible aproaches:
  - Take turns.

[1] http://swcarpentry.github.io/git-novice/

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.
- Possible aproaches:
  - Take turns. ✖ **Problem**: Each one will spend a lot of time waiting for the other to finish.

[1] `http://swcarpentry.github.io/git-novice/`

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.
- Possible aproaches:
  - Take turns. ✖ **Problem**: Each one will spend a lot of time waiting for the other to finish.
  - Work on their own copies and email changes back and forth.

[1] `http://swcarpentry.github.io/git-novice/`

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.
- Possible aproaches:
  - Take turns. ✖ **Problem**: Each one will spend a lot of time waiting for the other to finish.
  - Work on their own copies and email changes back and forth. ✖ **Problem**: Things will be lost, overwritten, or duplicated.

[1] `http://swcarpentry.github.io/git-novice/`

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.
- Possible aproaches:
  - Take turns. ✖ **Problem**: Each one will spend a lot of time waiting for the other to finish.
  - Work on their own copies and email changes back and forth. ✖ **Problem**: Things will be lost, overwritten, or duplicated.
  - Use a version control system, such as Git!

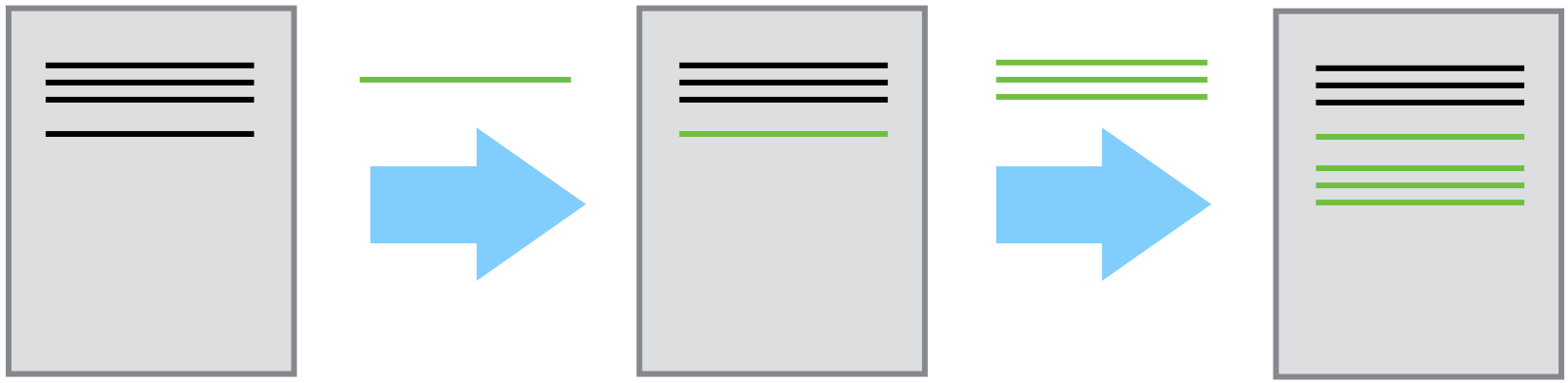
[1] `http://swcarpentry.github.io/git-novice/`

Motivating example[1]

- Wolfman and Dracula have been hired to investigate if it is possible to send a planetary lander to Mars.
- Want to be able to work on the plans at the same time.
- Possible aproaches:
  - Take turns. ✖ **Problem**: Each one will spend a lot of time waiting for the other to finish.
  - Work on their own copies and email changes back and forth. ✖ **Problem**: Things will be lost, overwritten, or duplicated.
  - Use a version control system, such as Git! ✔ Solves these problems!

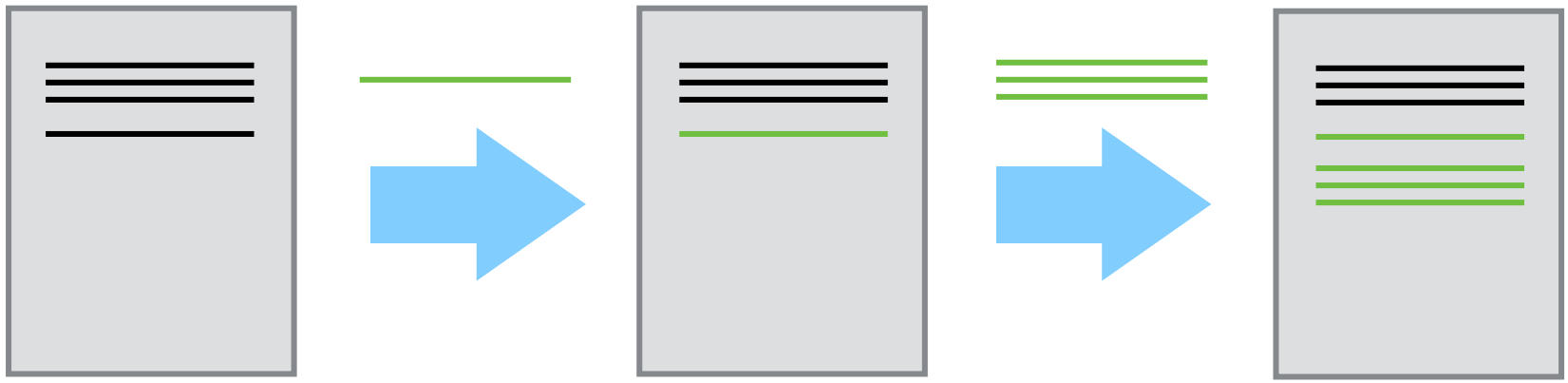
[1] `http://swcarpentry.github.io/git-novice/`

How it works[2]



[2] http://swcarpentry.github.io/git-novice/01-basics/

How it works[2]



"Unlimted undo!"

[2] http://swcarpentry.github.io/git-novice/01-basics/

Why version control?

> *[M]otivating git: You mostly collaborate with yourself, and me-from-two-months-ago never responds to email.—Karen Cranston*[3]

[3] `http://bit.ly/motivate_git`

[4] Quoted at `http://tex.stackexchange.com/a/1135/52`

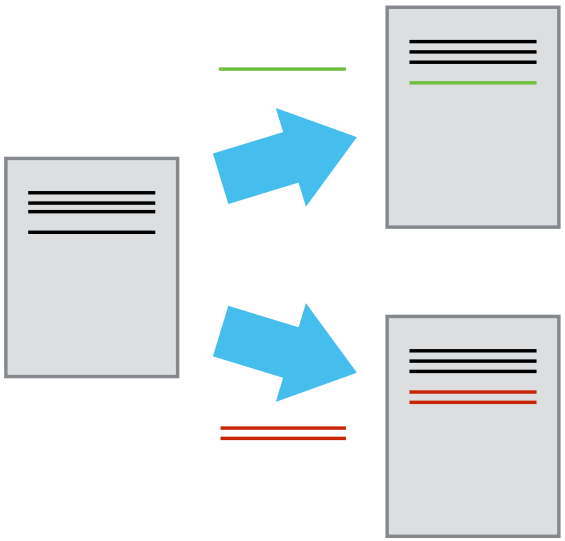
Why version control?

[M]otivating git: You mostly collaborate with yourself, and me-from-two-months-ago never responds to email.—Karen Cranston[3]

[Y]ou can now freely throw away bits and pieces, secure in the knowledge that if you actually want them back, they are there in the revision control system. Interestingly, almost nobody actually uses this feature. Revision control systems are not there to save your old work. They are there to give you permission to throw that old work away.
—Peter Boothe[4]

[3] `http://bit.ly/motivate_git`
[4] Quoted at `http://tex.stackexchange.com/a/1135/52`

How it works[5]



[5] http://swcarpentry.github.io/git-novice/01-basics/

How it works[5]

[5] http://swcarpentry.github.io/git-novice/01-basics/

How it works[5]

Different people can operate on the same file simultaneously.

[5] http://swcarpentry.github.io/git-novice/01-basics/

Set-up[6]

Don't type the $s!

Configure git (replace the name and email address with your own)

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

If you want to use Notepad instead, you can do:

```
$ git config core.editor notepad
```

[6] http://swcarpentry.github.io/git-novice/02-setup/

Set-up[6]

Don't type the $s!

Configure git (replace the name and email address with your own)

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

If you want to use Notepad instead, you can do:

```
$ git config core.editor notepad
```

Check all settings:

```
$ git config --list
```

[6] http://swcarpentry.github.io/git-novice/02-setup/

Set-up[7]

Can always get help, e.g.:

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

Or can consult: *Pro Git*, by Scott Chacon and Ben Straub, available for free at: `https://git-scm.com/book/en/v2`

[7] `http://swcarpentry.github.io/git-novice/02-setup/`

Creating a repository[8]

Don't type the $s!

Create a directory:

```
$ mkdir planets
$ cd planets
$ ls -aF
```

[8] http://swcarpentry.github.io/git-novice/03-create/

Creating a repository[8]

Don't type the $s!

Create a directory:

```
$ mkdir planets
$ cd planets
$ ls -aF
```

Turn it into a git repository:

```
$ git init
```

[8] http://swcarpentry.github.io/git-novice/03-create/

Creating a repository[8]

Don't type the $s!

Create a directory:

```
$ mkdir planets
$ cd planets
$ ls -aF
```

Turn it into a git repository:

```
$ git init
```

See that a new folder called `.git` has been created.

```
$ ls -aF
```

[8] http://swcarpentry.github.io/git-novice/03-create/

Creating a repository[8]

Don't type the $s!

Create a directory:

```
$ mkdir planets
$ cd planets
$ ls -aF
```

Turn it into a git repository:

```
$ git init
```

See that a new folder called `.git` has been created.

```
$ ls -aF
```

Check the status of our repository:

```
$ git status
```

[8] http://swcarpentry.github.io/git-novice/03-create/

Tracking changes[9]

Create a file called `mars.txt` (e.g. using `nano`) with the following content:

```
Cold and dry, but everything is my favorite color
```

[9] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[9]

Create a file called `mars.txt` (e.g. using `nano`) with the following content:

```
Cold and dry, but everything is my favorite color
```

Verify the file exists and its contents:

```
$ ls
$ cat mars.txt
```

[9] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[9]

Create a file called `mars.txt` (e.g. using `nano`) with the following content:

```
Cold and dry, but everything is my favorite color
```

Verify the file exists and its contents:

```
$ ls
$ cat mars.txt
```

Check the status of our repo:

```
$ git status
```

[9] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[9]

Create a file called `mars.txt` (e.g. using `nano`) with the following content:

```
Cold and dry, but everything is my favorite color
```

Verify the file exists and its contents:

```
$ ls
$ cat mars.txt
```

Check the status of our repo:

```
$ git status
```

Add the file to our repo:

```
$ git add mars.txt
$ git status
```

[9] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[10]

Commit to our repo:

```
$ git commit -m "Start notes on Mars as a base"
$ git status
```

[10] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[10]

Commit to our repo:

```
$ git commit -m "Start notes on Mars as a base"
$ git status
```

Verify in the log:

```
$ git log
```

[10] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[10]

Commit to our repo:

```
$ git commit -m "Start notes on Mars as a base"
$ git status
```

Verify in the log:

```
$ git log
```

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
```

[10] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[10]

Commit to our repo:

```
$ git commit -m "Start notes on Mars as a base"
$ git status
```

Verify in the log:

```
$ git log
```

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
```

Recheck the status:

```
$ git status
```

[10] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[11]

See difference between the current folder contents and repo:

```
$ git diff
```

[11] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[11]

See difference between the current folder contents and repo:

```
$ git diff
```

Now, commit the new changes to your repo:

```
$ git commit -m "Add concerns about effects of Mars'
↪  moons on Wolfman"
$ git status
```

[11] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[11]

See difference between the current folder contents and repo:

```
$ git diff
```

Now, commit the new changes to your repo:

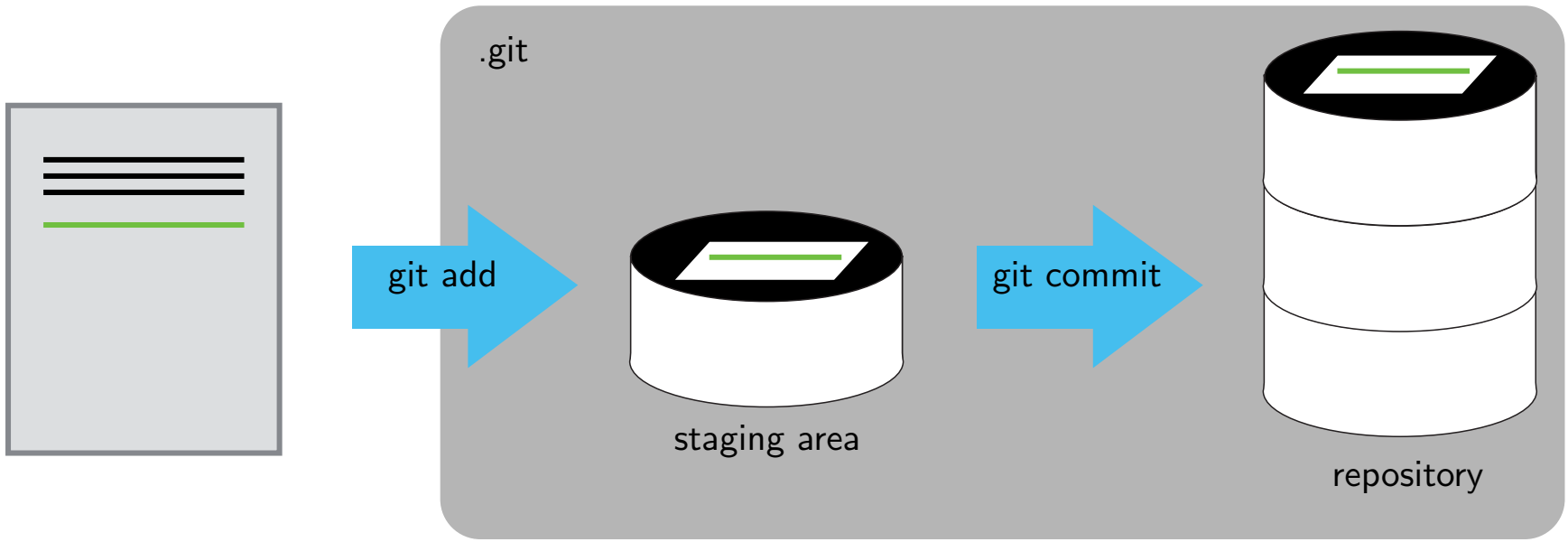
```
$ git commit -m "Add concerns about effects of Mars'
 ↪  moons on Wolfman"
$ git status
```

This didn't work, because we have to add first:

```
$ git add mars.txt
$ git commit -m "Add concerns about effects of Mars'
 ↪  moons on Wolfman"
$ git status
```

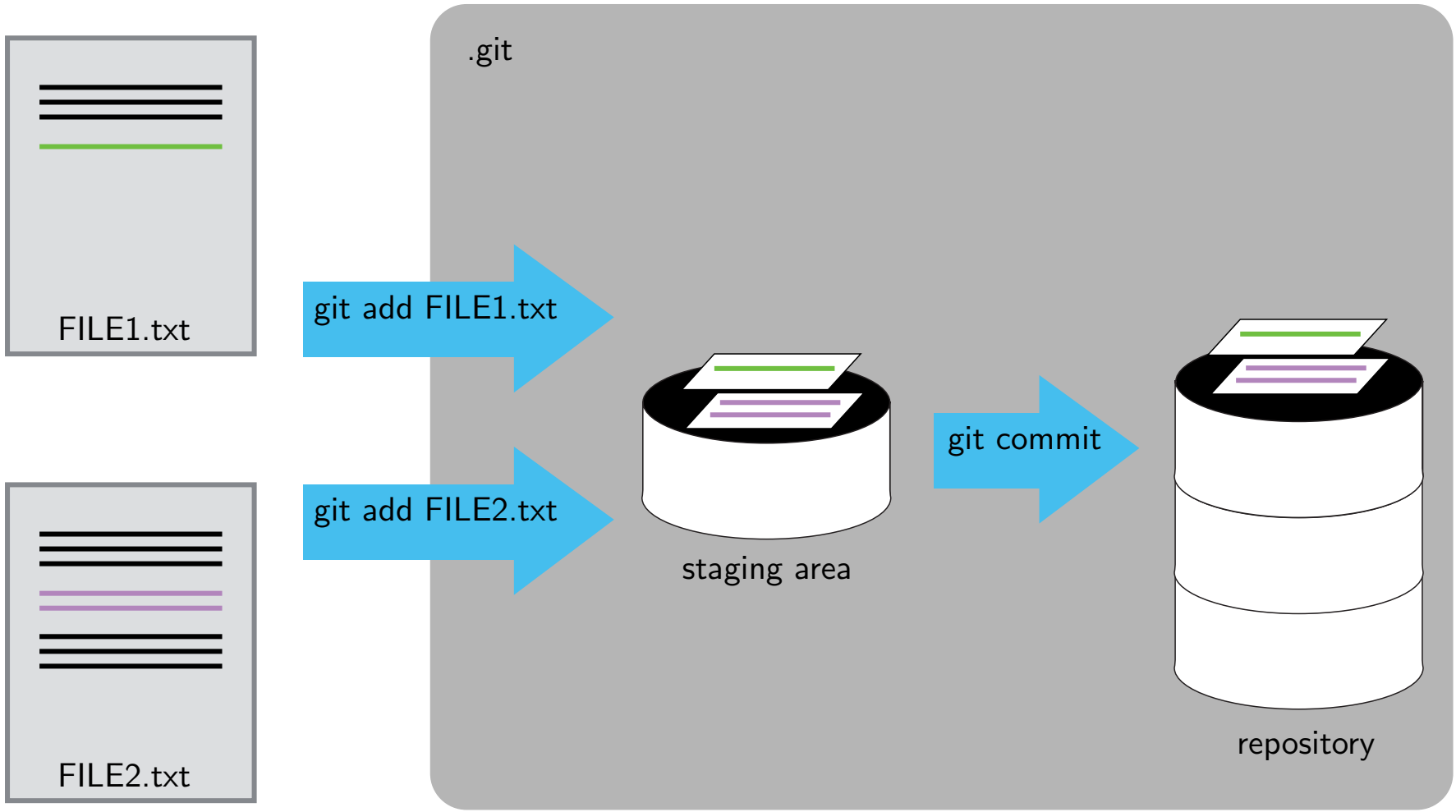
[11] http://swcarpentry.github.io/git-novice/04-changes/

Staging area[12]



[12] http://swcarpentry.github.io/git-novice/04-changes/

Committing[13]



[13] http://swcarpentry.github.io/git-novice/04-changes/

Exercise: Committing Changes to Git[14]

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"
```

[14] http://swcarpentry.github.io/git-novice/04-changes/#committing-changes-to-git

Exercise: Committing Changes to Git[14]

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

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

   ✖ Would only create a commit if files have already been staged.

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"`

[14] http://swcarpentry.github.io/git-novice/04-changes/#committing-changes-to-git

Exercise: Committing Changes to Git[14]

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

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

   ✖ Would only create a commit if files have already been staged.

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

   ✖ Would try to create a new repository.

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

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

[14] http://swcarpentry.github.io/git-novice/04-changes/#committing-changes-to-git

Exercise: Committing Changes to Git[14]

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

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

   ✖ Would only create a commit if files have already been staged.

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

   ✖ Would try to create a new repository.

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

   ✔ Is correct: first add the file to the staging area, then commit.

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

[14] http://swcarpentry.github.io/git-novice/04-changes/#committing-changes-to-git

Exercise: Committing Changes to Git[14]

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

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

   ✖ Would only create a commit if files have already been staged.

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

   ✖ Would try to create a new repository.

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

   ✔ Is correct: first add the file to the staging area, then commit.

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

   ✖ Would try to commit a file "my recent changes" with the message `myfile.txt`.

[14] http://swcarpentry.github.io/git-novice/04-changes/#committing-changes-to-git

Tracking changes[15]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
```

[15] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[15]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
```

See the differences:

```
$ git diff
```

[15] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[15]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
```

See the differences:

```
$ git diff
```

Re-stage the file, then look at differences again:

```
$ git add mars.txt
$ git diff
```

[15] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[15]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
```

See the differences:

```
$ git diff
```

Re-stage the file, then look at differences again:

```
$ git add mars.txt
$ git diff
```

To see the differences, we need the `--staged` flag:

```
$ git diff --staged
```

[15] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[16]

Commit to the repo:

```
$ git commit -m "Discuss concerns about Mars' climate
↪  for Mummy"
$ git status
$ git log
```

[16] http://swcarpentry.github.io/git-novice/04-changes/

Tracking changes[16]

Commit to the repo:

```
$ git commit -m "Discuss concerns about Mars' climate
↪  for Mummy"
$ git status
$ git log
```

Note: If the log it too long, Git will show in a "pager" interface. Press Q to get out.

[16] http://swcarpentry.github.io/git-novice/04-changes/

Exercise: Choosing a Commit Message[17]

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

1. "Changes"
2. "Added line 'But the Mummy will appreciate the lack of humidity' to mars.txt"
3. "Discuss effects of Mars climate on the Mummy"

[17] `http://swcarpentry.github.io/git-novice/04-changes/#choosing-a-commit-message`

Exercise: Choosing a Commit Message[17]

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

1. "Changes" ✖
2. "Added line 'But the Mummy will appreciate the lack of humidity' to mars.txt" ✖
3. "Discuss effects of Mars climate on the Mummy" ✔

[17] `http://swcarpentry.github.io/git-novice/04-changes/#choosing-a-commit-message`

Exercise: Choosing a Commit Message[17]

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

1. "Changes" ✖
2. "Added line 'But the Mummy will appreciate the lack of humidity' to mars.txt" ✖
3. "Discuss effects of Mars climate on the Mummy" ✔

Answer 1 is not descriptive enough, and answer 2 is too descriptive and redundant, but answer 3 is good: short but descriptive.

[17] `http://swcarpentry.github.io/git-novice/04-changes/#choosing-a-commit-message`

Exploring history[18]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
An ill-considered change
```

[18] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[18]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
An ill-considered change
```

Compare with previous versions:

```
$ git diff HEAD mars.txt
```

[18] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[18]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
An ill-considered change
```

Compare with previous versions:

```
$ git diff HEAD mars.txt
```

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

[18] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[18]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
An ill-considered change
```

Compare with previous versions:

```
$ git diff HEAD mars.txt
```

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

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

[18] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[18]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
An ill-considered change
```

Compare with previous versions:

```
$ git diff HEAD mars.txt
```

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

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

Show commit message as well:

```
$ git show HEAD~2 mars.txt
```

[18] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[19]

Access the log again:

```
$ git log
```

[19] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[19]

Access the log again:

```
$ git log
```

In the middle will be a message such as:

```
commit 4eece5059b74acd7223fad89d192b262ec41651f
Author: Christopher Phan <cphan@chrisphan.com>
Date:   Mon Mar 6 19:46:20 2017 -0600

    Add concerns about effects of Mars' moons on Wolfman
```

The string `4eece5059b74acd7223fad89d192b262ec41651f` is a unique identifier for the commit (will be different for you).

[19] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[19]

Access the log again:

```
$ git log
```

In the middle will be a message such as:

```
commit 4eece5059b74acd7223fad89d192b262ec41651f
Author: Christopher Phan <cphan@chrisphan.com>
Date:   Mon Mar 6 19:46:20 2017 -0600

    Add concerns about effects of Mars' moons on Wolfman
```

The string `4eece5059b74acd7223fad89d192b262ec41651f` is a unique identifier for the commit (will be different for you).

Can get a diff in relation to this commit:

```
$ git diff 4eece5059b74acd7223fad89d192b262ec41651f
↪  mars.txt
```

[19] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[19]

Access the log again:

```
$ git log
```

In the middle will be a message such as:

```
commit 4eece5059b74acd7223fad89d192b262ec41651f
Author: Christopher Phan <cphan@chrisphan.com>
Date:   Mon Mar 6 19:46:20 2017 -0600

    Add concerns about effects of Mars' moons on Wolfman
```

The string `4eece5059b74acd7223fad89d192b262ec41651f` is a unique identifier for the commit (will be different for you).

Can get a diff in relation to this commit:

```
$ git diff 4eece5059b74acd7223fad89d192b262ec41651f
↪  mars.txt
```

This is suboptimal!

[19] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[19]

Access the log again:

```
$ git log
```

In the middle will be a message such as:

```
commit 4eece5059b74acd7223fad89d192b262ec41651f
Author: Christopher Phan <cphan@chrisphan.com>
Date:   Mon Mar 6 19:46:20 2017 -0600

    Add concerns about effects of Mars' moons on Wolfman
```

The string `4eece5059b74acd7223fad89d192b262ec41651f` is a unique identifier for the commit (will be different for you).

Can get a diff in relation to this commit:

```
$ git diff 4eece5059b74acd7223fad89d192b262ec41651f
↪  mars.txt
```

This is suboptimal! Can abbreviate:

```
$ git diff 4eece mars.txt
```

[19] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[20]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
We will need to manufacture our own oxygen
```

Delete the other lines.

[20] `http://swcarpentry.github.io/git-novice/05-history/`

Exploring history[20]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
We will need to manufacture our own oxygen
```

Delete the other lines.

```
$ git status
```

[20] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[20]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
We will need to manufacture our own oxygen
```

Delete the other lines.

```
$ git status
```

Suppose we didn't like the change. Can undo it:

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

[20] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[20]

Edit the file `mars.txt` so that it reads (e.g. using `nano`):

```
We will need to manufacture our own oxygen
```

Delete the other lines.

```
$ git status
```

Suppose we didn't like the change. Can undo it:

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

Back to the last commit!

[20] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[21]

Suppose we want to go back to the way things were:

```
$ git log
```

[21] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[21]

Suppose we want to go back to the way things were:

```
$ git log
```

Take note of the first commit:

```
commit 4223fa9e5a953eb98381d9f724a3d715bff0e88f
Author: Christopher Phan <cphan@chrisphan.com>
Date:   Mon Mar 6 19:36:24 2017 -0600

    Start notes on Mars as a base
```

[21] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[21]

Suppose we want to go back to the way things were:

```
$ git log
```

Take note of the first commit:

```
commit 4223fa9e5a953eb98381d9f724a3d715bff0e88f
Author: Christopher Phan <cphan@chrisphan.com>
Date:   Mon Mar 6 19:36:24 2017 -0600

    Start notes on Mars as a base
```

We can go back there. (Replace `4223fa` with the first few characters shown on your system.)

```
$ git checkout 4223fa mars.txt
$ cat mars.txt
$ git status
```

[21] http://swcarpentry.github.io/git-novice/05-history/

Exploring history[21]

Suppose we want to go back to the way things were:

```
$ git log
```

Take note of the first commit:

```
commit 4223fa9e5a953eb98381d9f724a3d715bff0e88f
Author: Christopher Phan <cphan@chrisphan.com>
Date:   Mon Mar 6 19:36:24 2017 -0600

    Start notes on Mars as a base
```

We can go back there. (Replace `4223fa` with the first few characters shown on your system.)

```
$ git checkout 4223fa mars.txt
$ cat mars.txt
$ git status
```

I could commit this, if I wanted. But let's revert again:

```
$ git checkout -f master mars.txt
$ cat mars.txt
```

[21] http://swcarpentry.github.io/git-novice/05-history/

Exercise: Recovering older versions of a file[22]

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. She has spent about one hour trying to fix it, with no luck ...

[22] http://swcarpentry.github.io/git-novice/05-history/#recovering-older-versions-of-a-file

Exercise: Recovering older versions of a file[22]

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. She has spent about one hour trying to fix it, with no luck ...
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 <ID of last commit> data_cruncher.py`

[22] http://swcarpentry.github.io/git-novice/05-history/#recovering-older-versions-of-a-file

Exercise: Recovering older versions of a file[22]

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. She has spent about one hour trying to fix it, with no luck ...
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 <ID of last commit> data_cruncher.py`
   ✔

[22] http://swcarpentry.github.io/git-novice/05-history/#recovering-older-versions-of-a-file

Ignoring things[23]

Create some dummy files:

```
$ mkdir results
$ touch a.dat b.dat c.dat results/a.out results/b.out
$ ls -lF
$ ls -lF results
$ git status
```

The Unix command `touch` creates empty files, but let's pretend they have stuff in them *that we want Git to ignore*.

[23] `http://swcarpentry.github.io/git-novice/06-ignore/`

Ignoring things[23]

Create some dummy files:

```
$ mkdir results
$ touch a.dat b.dat c.dat results/a.out results/b.out
$ ls -lF
$ ls -lF results
$ git status
```

The Unix command `touch` creates empty files, but let's pretend they have stuff in them *that we want Git to ignore*.

Create a file called `.gitignore` with the following contents (e.g. using `nano`):

```
*.dat
results/
```

[23] `http://swcarpentry.github.io/git-novice/06-ignore/`

Ignoring things[24]

Check the status:

```
$ git status
```

[24] http://swcarpentry.github.io/git-novice/06-ignore/

Ignoring things[24]

Check the status:

```
$ git status
```

We want to track `.gitignore` (so if someone else clones our repo and generates these files, they will be ignored).

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

[24] http://swcarpentry.github.io/git-novice/06-ignore/

Ignoring things[24]

Check the status:

```
$ git status
```

We want to track `.gitignore` (so if someone else clones our repo and generates these files, they will be ignored).

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

Git will even prevent us from adding things it's been told to ignore:

```
$ git add a.dat
$ git status --ignored
```

[24] http://swcarpentry.github.io/git-novice/06-ignore/

Exercise: Ignoring all data files in a directory[25]

Given a directory structure that looks like:

```
results/data/position/gps/a.data
results/data/position/gps/b.data
results/data/position/gps/c.data
results/data/position/gps/info.txt
results/plots
```

What's the shortest `.gitignore` rule you could write to ignore all `.data` files in `result/data/position/gps`? Do not ignore the `info.txt`.

[25] `http://swcarpentry.github.io/git-novice/06-ignore/#ignoring-all-data-files-in-a-directory`

Exercise: Ignoring all data files in a directory[25]

Given a directory structure that looks like:

```
results/data/position/gps/a.data
results/data/position/gps/b.data
results/data/position/gps/c.data
results/data/position/gps/info.txt
results/plots
```

What's the shortest `.gitignore` rule you could write to ignore all `.data` files in `result/data/position/gps`? Do not ignore the `info.txt`.

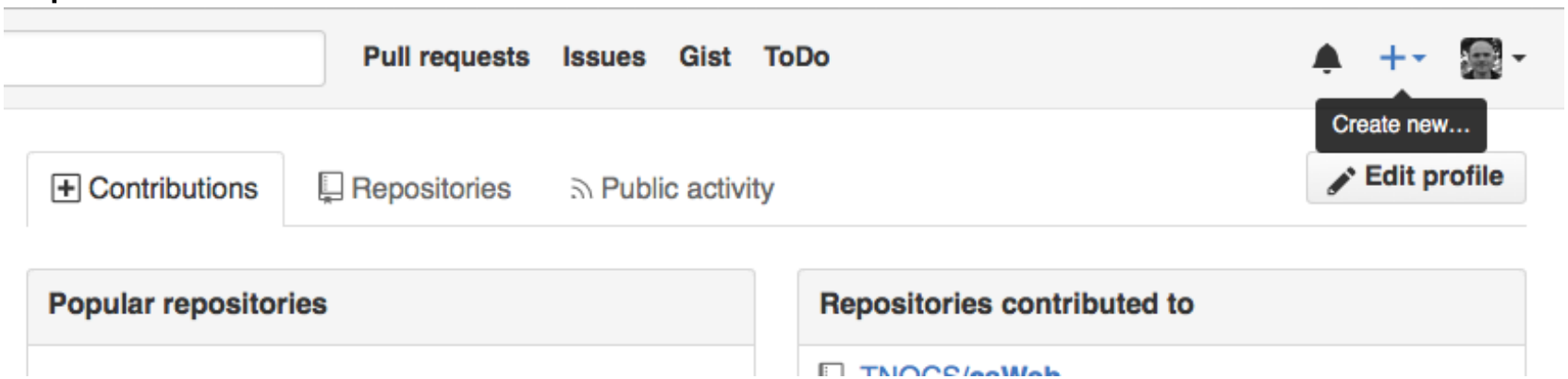
Appending `results/data/position/gps/*.data` will match every file in `results/data/position/gps` that ends with `.data`. The file `results/data/position/gps/info.txt` will not be ignored.

---

[25] `http://swcarpentry.github.io/git-novice/06-ignore/#ignoring-all-data-files-in-a-directory`

Remotes in GitHub[26]

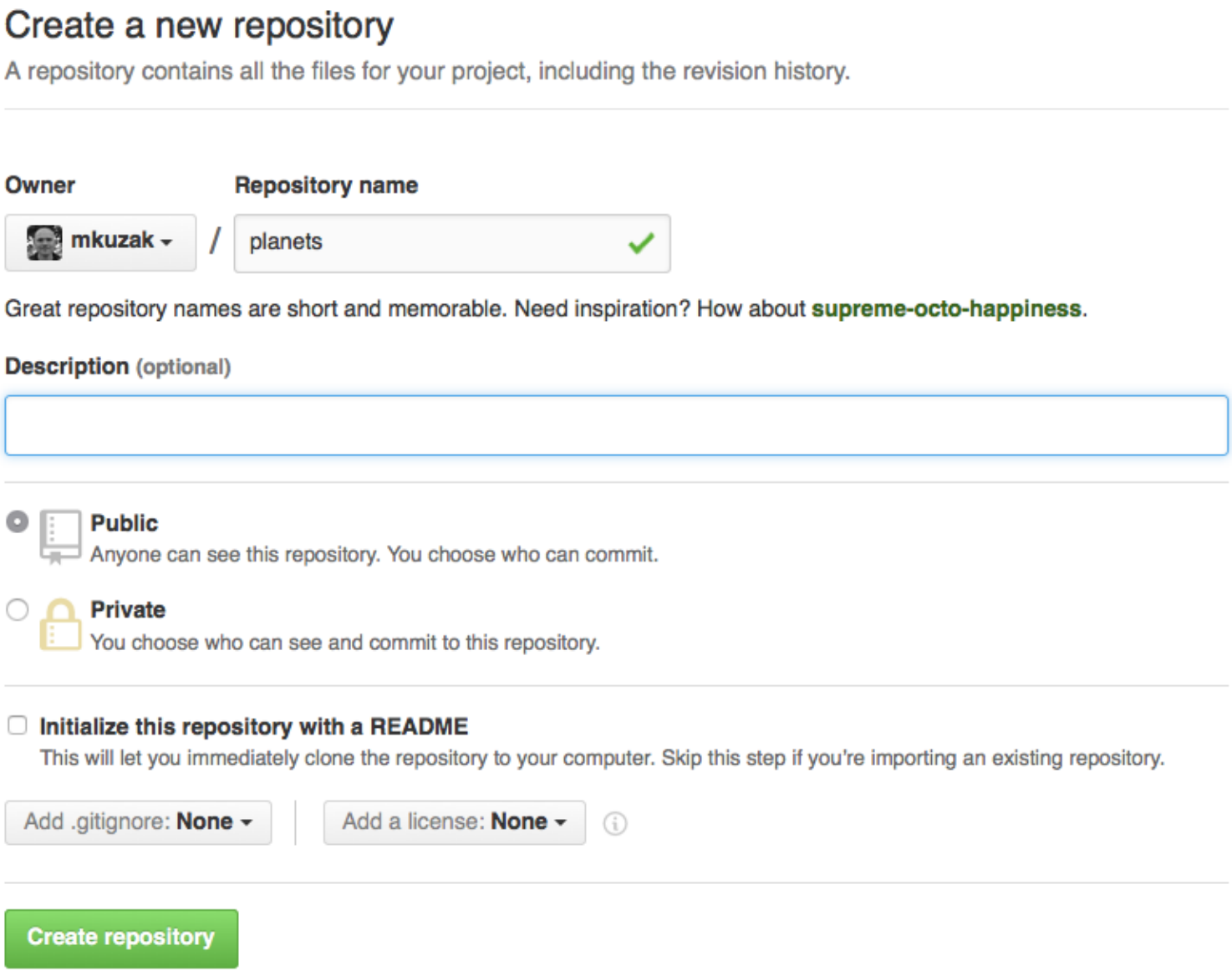
Log into GitHub, and click the plus in the corner to create a new repo:



[26] http://swcarpentry.github.io/git-novice/07-github/

Remotes in GitHub[27]

Give it the name `planets`:



[27] http://swcarpentry.github.io/git-novice/07-github/

Remotes in GitHub[28]

Copy the URL to the repo by clicking the clipboard icon:

[28] http://swcarpentry.github.io/git-novice/07-github/

Remotes in GitHub[28]

Copy the URL to the repo by clicking the clipboard icon:

Then use the URL to connect our local repo to the GitHub repo:

```
$ git remote add origin <URL>
$ git remote -v
$ git push origin master
```

[28] http://swcarpentry.github.io/git-novice/07-github/

Remotes in GitHub[28]

Copy the URL to the repo by clicking the clipboard icon:

Then use the URL to connect our local repo to the GitHub repo:

```
$ git remote add origin <URL>
$ git remote -v
$ git push origin master
```

To pull changes off GitHub:

```
$ git pull origin master
```

[28] http://swcarpentry.github.io/git-novice/07-github/

Remotes in GitHub[28]

Copy the URL to the repo by clicking the clipboard icon:

Then use the URL to connect our local repo to the GitHub repo:

```
$ git remote add origin <URL>
$ git remote -v
$ git push origin master
```

To pull changes off GitHub:

```
$ git pull origin master
```

Think-pair-share Q: In this lesson, we introduced the `git push` command. How is `git push` different from `git commit`?

[28] http://swcarpentry.github.io/git-novice/07-github/

Collaborating[29]

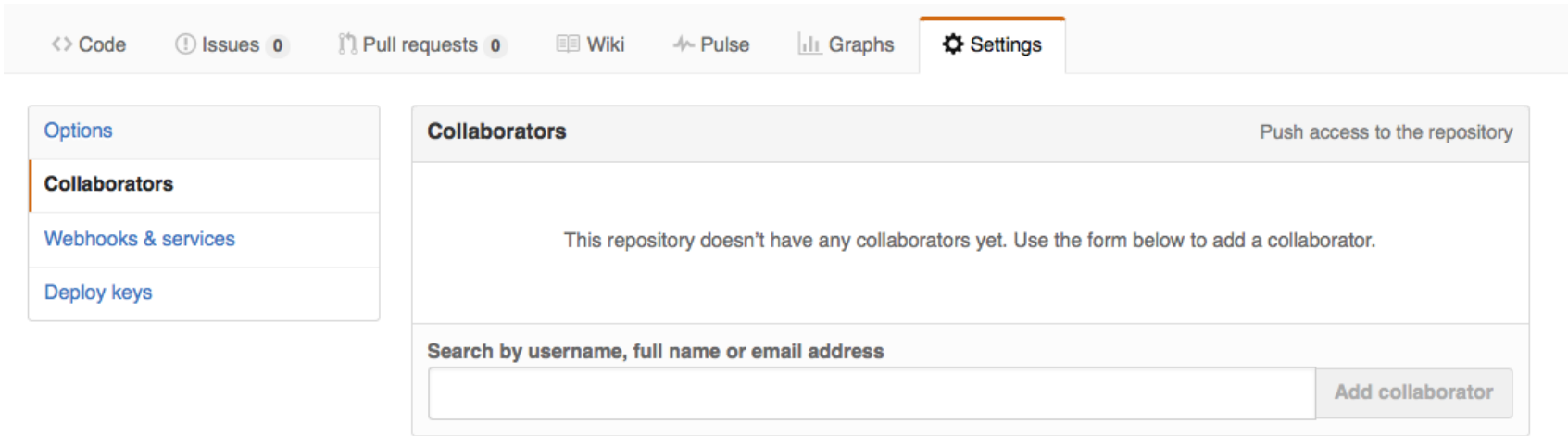
Get into pairs!

[29] http://swcarpentry.github.io/git-novice/08-collab/

Collaborating[29]

Get into pairs!

Person 1: Go into the settings tab on GitHub for your `planets` repo, select "Collaborators" from the side, and then add your partner as a collaborator:



Person 2: Go to `https://github.com/notifications` and accept your partner's request.

[29] `http://swcarpentry.github.io/git-novice/08-collab/`

Collaborating[30]

Person 2: Clone your partner's repo (replace `vlad` with your partner's username):

```
$ git clone https://github.com/vlad/planets.git ~/Desktop/vlad-planets
```

Navigate into the repo:

```
$ cd ~/Desktop/vlad-planets
```

Create a file called `pluto.txt` (e.g. using `nano`) with the following contents:

```
It is so a planet!
```

Then add, commit, and push your changes:

```
$ git add pluto.txt
$ git commit -m "Some notes about Pluto"
$ git push origin master
```

[30] http://swcarpentry.github.io/git-novice/08-collab/

Collaborating[31]

Person 1: Pull the changes from the repo:

```
$ git pull origin master
$ cat pluto.txt
```

[31] http://swcarpentry.github.io/git-novice/08-collab/

Collaborating[31]

Person 1: Pull the changes from the repo:

```
$ git pull origin master
$ cat pluto.txt
```

Now, switch roles and repeat!

[31] http://swcarpentry.github.io/git-novice/08-collab/

Conflicts[32]

Person 1: Change the file `mars.txt` so that it reads:

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
This line added to Wolfman's copy
```

And then commit and push to GitHub:

```
$ git add mars.txt
$ git commit -m "Adding a line in our home copy"
$ git push origin master
```

Person 2: DO NOT pull this change. Leave your copy of the repo alone.

[32] `http://swcarpentry.github.io/git-novice/09-conflict/`

Conflicts[33]

Person 2: Change the file `mars.txt` so that it reads:

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
We added a different line in the other copy
```

And then commit and push to GitHub:

```
$ git add mars.txt
$ git commit -m "Adding a line in my copy"
$ git push origin master
```

You will get an error.

[33] http://swcarpentry.github.io/git-novice/09-conflict/

Conflicts[33]

Person 2: Change the file `mars.txt` so that it reads:

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
We added a different line in the other copy
```

And then commit and push to GitHub:

```
$ git add mars.txt
$ git commit -m "Adding a line in my copy"
$ git push origin master
```

You will get an error. Try:

```
$ git pull origin master
$ cat mars.txt
```

[33] http://swcarpentry.github.io/git-novice/09-conflict/

Conflicts[34]

Person 2: Change the file `mars.txt` so that it reads:

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
We removed the conflict on this line
```

And then commit and push to GitHub:

```
$ git add mars.txt
$ git status
$ git commit -m "Merging changes from GitHub"
$ git push origin master
```

[34] http://swcarpentry.github.io/git-novice/09-conflict/

Conflicts[34]

Person 2: Change the file `mars.txt` so that it reads:

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
We removed the conflict on this line
```

And then commit and push to GitHub:

```
$ git add mars.txt
$ git status
$ git commit -m "Merging changes from GitHub"
$ git push origin master
```

Person 1: Try:

```
$ git pull origin master
$ cat mars.txt
```

[34] http://swcarpentry.github.io/git-novice/09-conflict/

Recap

- `git config` — access configuration
- `git init` — create a (local) repo
- `git add` — add a file to repo or stage changes
- `git status` — check the status of the repo
- `git commit` — commit (staged) changes to repo
- `git push` — push local commits to the remote copy of the repo
- `git pull` — pull commits from remote copy of the repo to the local copy of repo

Copyright notice