

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 example¹

- ▶ 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 example¹

- ▶ 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 example¹

- ▶ 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 approaches:

¹<http://swcarpentry.github.io/git-novice/> 

Motivating example¹

- ▶ 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 approaches:
 - ▶ 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 example¹

- ▶ 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 approaches:
 - ▶ 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 example¹

- ▶ 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 approaches:
 - ▶ 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 example¹

- ▶ 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 approaches:
 - ▶ 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 example¹

- ▶ 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 approaches:
 - ▶ 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 works²



²<http://swcarpentry.github.io/git-novice/01-basics/>

How it works²



“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 Cranston³

³http://bit.ly/motivate_git

⁴Quoted at <http://tex.stackexchange.com/a/1135/52>

Why version control?

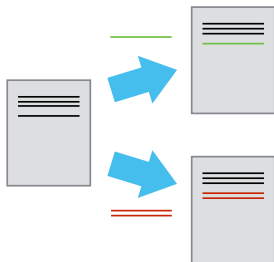
[M]otivating git: You mostly collaborate with yourself, and me-from-two-months-ago never responds to email.—Karen Cranston³

*[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 Boothe⁴*

³http://bit.ly/motivate_git

⁴Quoted at <http://tex.stackexchange.com/a/1135/52>

How it works⁵



⁵<http://swcarpentry.github.io/git-novice/01-basics/>

How it works⁵



⁵<http://swcarpentry.github.io/git-novice/01-basics/>

How it works⁵



Different people can operate on the same file simultaneously.

⁵<http://swcarpentry.github.io/git-novice/01-basics/>

Set-up⁶


Don't type the \$!

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-up⁶

Don't type the \$!

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-up⁷

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 repository⁸


Don't type the \$!

Create a directory:

```
$ mkdir planets
```

```
$ cd planets
```

```
$ ls -aF
```

⁸<http://swcarpentry.github.io/git-novice/03-create/> 

Creating a repository⁸

Don't type the \$!

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 repository⁸

Don't type the \$!

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 repository⁸

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 changes⁹

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 changes⁹


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 changes⁹

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 changes⁹

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 changes¹⁰

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 changes¹⁰


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 changes¹⁰

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 changes¹⁰

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 changes¹¹

See difference between the current folder contents and repo:

```
$ git diff
```

¹¹<http://swcarpentry.github.io/git-novice/04-changes/> 


Tracking changes¹¹

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 changes¹¹

See difference between the current folder contents and repo:


```
$ git diff
```

Now, commit the new changes to your repo:

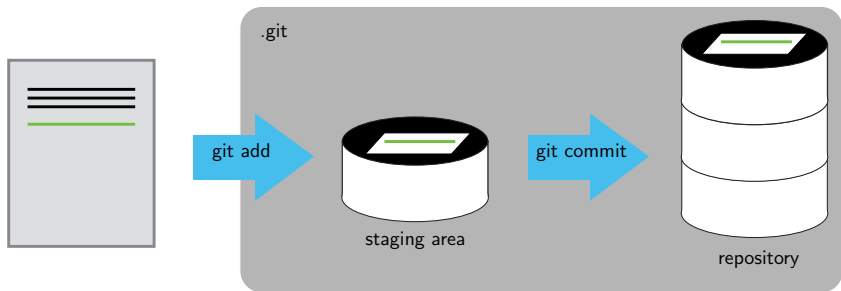
```
$ git commit -m "Add concerns about effects of Mars'  
  ↪ moons on Wolfman"  
$ git status
```

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
```

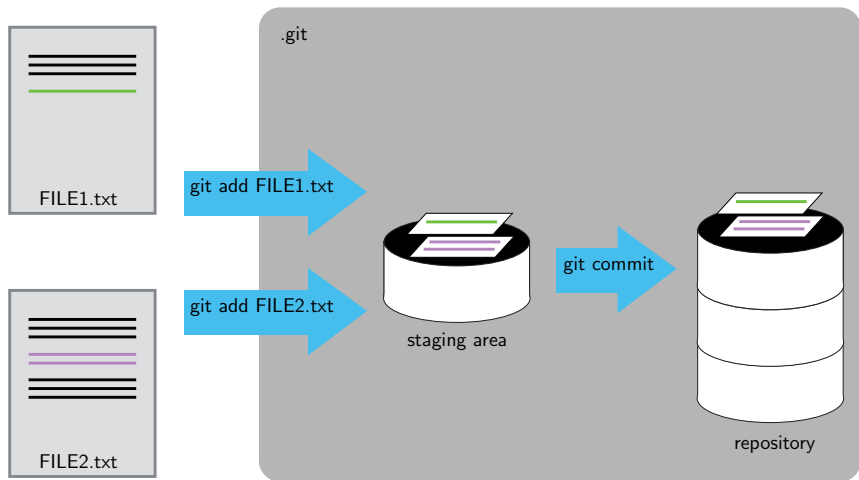
¹¹<http://swcarpentry.github.io/git-novice/04-changes/> 

Staging area¹²



¹²<http://swcarpentry.github.io/git-novice/04-changes/>

Committing¹³



¹³<http://swcarpentry.github.io/git-novice/04-changes/>

Exercise: Committing Changes to Git¹⁴

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/>

Exercise: Committing Changes to Git¹⁴

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/>

Exercise: Committing Changes to Git¹⁴

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/>

Exercise: Committing Changes to Git¹⁴

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/>

Exercise: Committing Changes to Git¹⁴

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/>


Tracking changes¹⁵

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 changes¹⁵

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 changes¹⁵

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 changes¹⁵

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 changes¹⁶

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 changes¹⁶

Commit to the repo:

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

Note: If the log is 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 Message¹⁷

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 Message¹⁷

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 Message¹⁷

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 history¹⁸


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 history¹⁸

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 history¹⁸

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 history¹⁸

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 history¹⁸

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 history¹⁹

Access the log again:

```
$ git log
```

¹⁹<http://swcarpentry.github.io/git-novice/05-history/> 

Exploring history¹⁹

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 history¹⁹

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 history¹⁹

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 history¹⁹

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 history²⁰

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 history²⁰

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 history²⁰

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 history²⁰

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 history²¹

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

```
$ git log
```

²¹<http://swcarpentry.github.io/git-novice/05-history/> 

Exploring history²¹


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 history²¹

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 history²¹

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 file²²

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

Exercise: Recovering older versions of a file²²

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 things²³

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 things²³


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 things²⁴

Check the status:

```
$ git status
```

²⁴<http://swcarpentry.github.io/git-novice/06-ignore/> 

Ignoring things²⁴

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 things²⁴

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 directory²⁵

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 directory²⁵

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>

Copyright notice

The materials here are based on the lesson plans provided by Software Carpentry at <http://swcarpentry.github.io/git-novice/>, and are used under the Creative Commons Attribution 4.0 license (<https://software-carpentry.org/license/>).

These materials are available under the Creative Commons Attribution 4.0 license (<https://software-carpentry.org/license/>).