

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:
<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.
- ▶ Two 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.
- ▶ Two approaches:
 - ▶ Take turns.

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.
- ▶ Two 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.
- ▶ Two 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.
- ▶ Two 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/>

Why version control?²

- ▶ <http://www.phdcomics.com/comics/archive.php?comicid=1531>
- ▶ http://bit.ly/motivate_git

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

Why version control?

[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 Boothe³

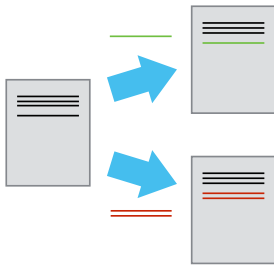
³Quoted at <http://tex.stackexchange.com/a/1135/52>

How it works⁴



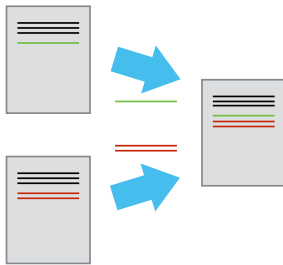
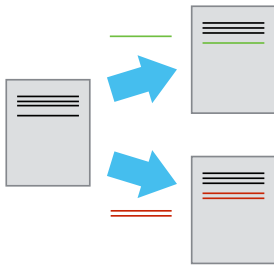
⁴<http://swcarpentry.github.io/git-novice/01-basics/>

How it works⁵



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

How it works⁵



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

Format for git shell commands⁶

`git verb`

E.g.:

- ▶ `git add`
- ▶ `git commit`
- ▶ `git push`
- ▶ `git config`

⁶<http://swcarpentry.github.io/git-novice/02-setup/>

Set-up⁷

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"
```

⁷<http://swcarpentry.github.io/git-novice/02-setup/>

Set-up⁷

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"
```

Check all settings:

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

⁷<http://swcarpentry.github.io/git-novice/02-setup/>

Set-up⁷

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"
```

Check all settings:

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

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 \$s!

Create a directory:

```
$ mkdir planets
```

```
$ cd planets
```

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

Creating a repository⁸

Don't type the \$s!

Create a directory:

```
$ mkdir planets
```

```
$ cd planets
```

Turn it into a git repository:

```
$ git init
```

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

Creating a repository⁸

Don't type the \$s!

Create a directory:

```
$ mkdir planets
```

```
$ cd planets
```

Turn it into a git repository:

```
$ git init
```

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

```
$ ls -a
```

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

Creating a repository⁸

Don't type the \$!

Create a directory:

```
$ mkdir planets
```

```
$ cd planets
```

Turn it into a git repository:

```
$ git init
```

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

```
$ ls -a
```

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 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/>

Exercise: Committing Changes to Git¹²

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"`

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

Exercise: Committing Changes to Git¹²

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"`

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

Exercise: Committing Changes to Git¹²

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"`

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

Exercise: Committing Changes to Git¹²

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"`

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

Exercise: Committing Changes to Git¹²

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"`

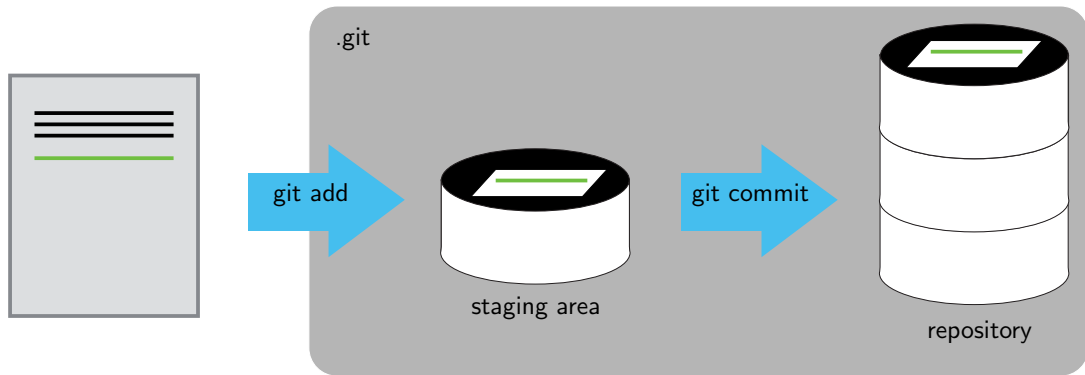
✓ Is correct: first add the file to the staging area, then commit.

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

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

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

Staging area¹³



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

Tracking changes¹⁴

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
```

¹⁴<http://swcarpentry.github.io/git-novice/04-changes/>

Tracking changes¹⁴

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
```

¹⁴<http://swcarpentry.github.io/git-novice/04-changes/>

Tracking changes¹⁴

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
```

¹⁴<http://swcarpentry.github.io/git-novice/04-changes/>

Tracking changes¹⁴

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
```

¹⁴<http://swcarpentry.github.io/git-novice/04-changes/>

Commit to the repo:

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

```
$ git status
```

```
$ git log
```

Commit to the repo:

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

```
$ git status
```

```
$ git log
```

Directories in git¹⁵

Git does not automatically add directories:

```
$ mkdir directory
```

```
$ git status
```

¹⁵<http://swcarpentry.github.io/git-novice/04-changes/>

Directories in git¹⁵

Git does not automatically add directories:

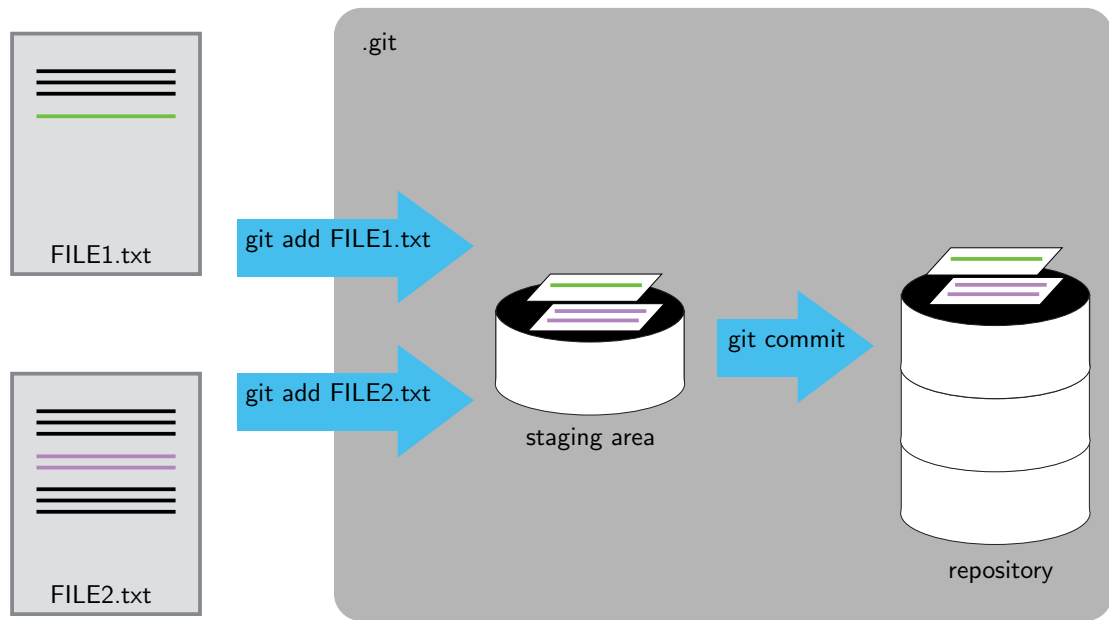
```
$ mkdir directory  
$ git status
```

You have to add it explicitly:

```
$ git add directory  
$ git status
```

¹⁵<http://swcarpentry.github.io/git-novice/04-changes/>

Committing¹⁶



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>

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/>).