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 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.
- Two 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.
- Two 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.
- Two 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.
- Two 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[2]

A colleague suggests using version control to manage their work. Version control is better than mailing files back and forth:

- Nothing that is committed to version control is ever lost, unless you work really, really hard at it. Since all old versions of files are saved, its always possible to go back in time to see exactly who wrote what on a particular day, or what version of a program was used to generate a particular set of results.

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

Motivating example[2]

A colleague suggests using version control to manage their work. Version control is better than mailing files back and forth:

- Nothing that is committed to version control is ever lost, unless you work really, really hard at it. Since all old versions of files are saved, its always possible to go back in time to see exactly who wrote what on a particular day, or what version of a program was used to generate a particular set of results.
- As we have this record of who made what changes when, we know who to ask if we have questions later on, and, if needed, revert to a previous version, much like the "undo" feature in an editor.

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

Motivating example[2]

A colleague suggests using version control to manage their work. Version control is better than mailing files back and forth:

- Nothing that is committed to version control is ever lost, unless you work really, really hard at it. Since all old versions of files are saved, its always possible to go back in time to see exactly who wrote what on a particular day, or what version of a program was used to generate a particular set of results.
- As we have this record of who made what changes when, we know who to ask if we have questions later on, and, if needed, revert to a previous version, much like the "undo" feature in an editor.
- When several people collaborate in the same project, it's possible to accidentally overlook or overwrite someone's changes. The version control system automatically notifies users whenever there's a conflict between one person's work and another's.

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

Motivating example[3]

Teams are not the only ones to benefit from version control: lone researchers can benefit immensely. Keeping a record of what was changed, when, and why is extremely useful for all researchers if they ever need to come back to the project later on (e.g., a year later, when memory has faded).

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

Motivating example[3]

Teams are not the only ones to benefit from version control: lone researchers can benefit immensely. Keeping a record of what was changed, when, and why is extremely useful for all researchers if they ever need to come back to the project later on (e.g., a year later, when memory has faded).

Version control is the lab notebook of the digital world: its what professionals use to keep track of what theyve done and to collaborate with other people. Every large software development project relies on it, and most programmers use it for their small jobs as well. And it isnt just for software: books, papers, small data sets, and anything that changes over time or needs to be shared can and should be stored in a version control system.

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

Why version control?[4]

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

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

Why version control?

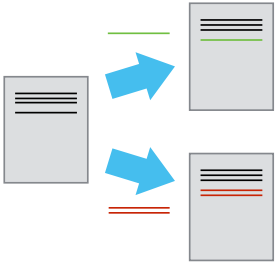
[Y]ou can now freely throw away bits and pieces, secure in the knowledge that if you actually want them back, they are there in the revision control system. Interestingly, almost nobody actually uses this feature. Revision control systems are not there to save your old work. They are there to give you permission to throw that old work away.
—Peter Boothe[5]

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

How it works[6]

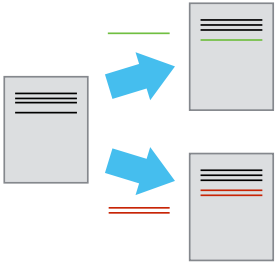
[6] http://swcarpentry.github.io/git-novice/01-basics/

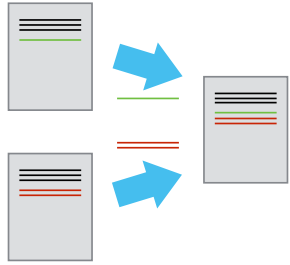
How it works[7]



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

How it works[7]





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

Format for `git` shell commands[8]

`git` *verb*

E.g.:

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

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

Set-up[9]

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"
```

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

Set-up[9]

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
```

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

Set-up[9]

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`

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

Creating a repository[10]

Don't type the $s!

Create a directory:

```
$ mkdir planets
$ cd planets
```

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

Creating a repository[10]

Don't type the $s!

Create a directory:

```
$ mkdir planets
$ cd planets
```

Turn it into a git repository:

```
$ git init
```

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

Creating a repository[10]

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
```

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

Creating a repository[10]

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
```

Check the status of our repository:

```
$ git status
```

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

Tracking changes[11]

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

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

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

Tracking changes[11]

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
```

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

Tracking changes[11]

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
```

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

Tracking changes[11]

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
```

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

Tracking changes[11]

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
```

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

Tracking changes[12]

Commit to our repo:

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

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

Tracking changes[12]

Commit to our repo:

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

Verify in the log:

```
$ git log
```

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

Tracking changes[12]

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
```

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

Tracking changes[12]

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
```

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

Tracking changes[13]

See difference between the current folder contents and repo:

```
$ git diff
```

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

Tracking changes[13]

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
```

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

Tracking changes[13]

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
```

[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"`

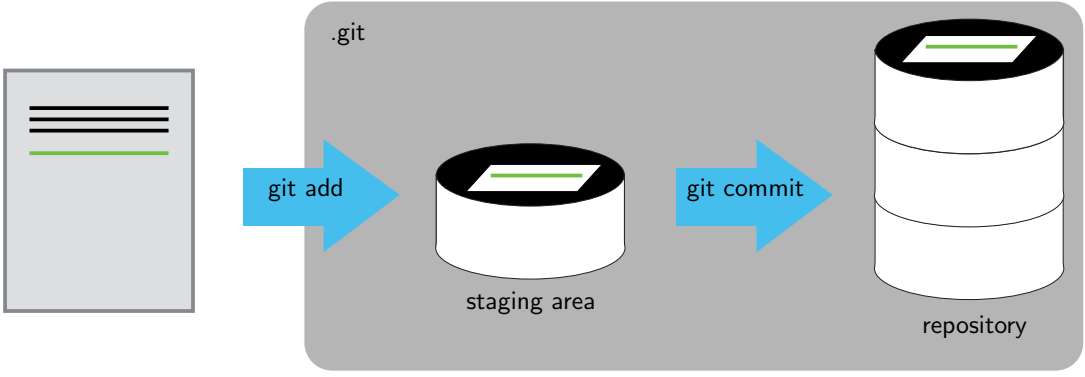
   ✔ Is correct: first add the file to the staging area, then commit.

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`

Staging area[15]



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

Tracking changes[16]

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
```

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

Tracking changes[16]

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
```

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

Tracking changes[16]

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
```

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

Tracking changes[16]

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
```

[16] 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 git[17]

Git does not automatically add directories:

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

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

Directories in git[17]

Git does not automatically add directories:

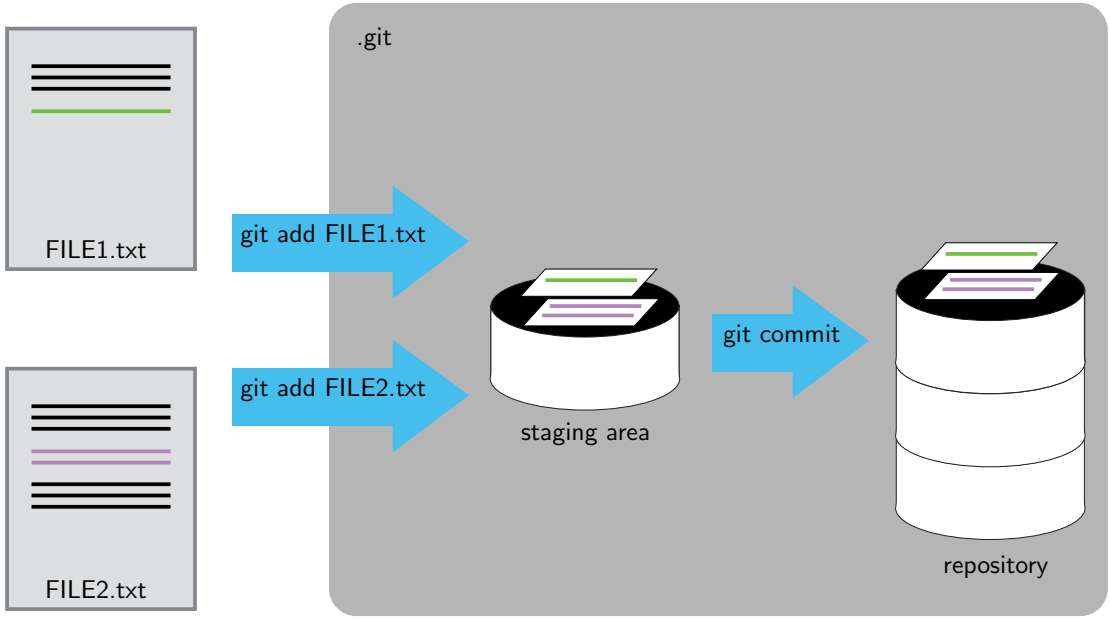
```
$ mkdir directory
$ git status
```

You have to add it explicitly:

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

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

Committing[18]



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

Exercise: Choosing a Commit Message[19]

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"

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

Exercise: Choosing a Commit Message[19]

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" ✔

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

Exercise: Choosing a Commit Message[19]

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.

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

Copyright notice