

Version Control with Git

Koen Leuveld

June 2, 2023

Program

Time	Item
9:30	Intro & Recap
9:45	6. Ignoring Things
9:50	7. Remotes in GitHub
10:35	Coffee Break
10:50	8. Collaborating
11:15	9. Conflicts
11:30	Lunch Break
12:30	Git in practice
13:00	Reflection
13:30	WEEKEND

"FINAL".doc



FINAL.doc!



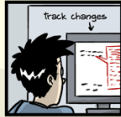
FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc

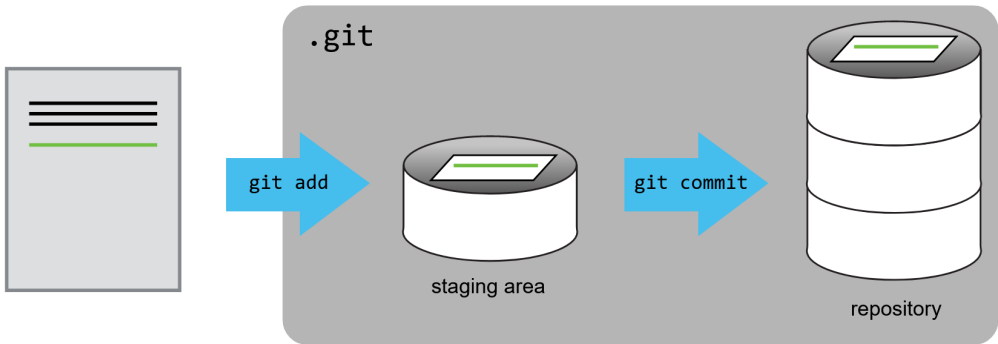


FINAL_rev.22.comments49.
corrections.10.##\$%WHYDID
ICOMETOGRADSCHOOL????.doc

JORGE CHAM © 2012

UNIVERSAL MISSIONS





Basic Git commands

```
$ git <command> <options>
```

E.g.

```
▶ git add myfile.txt
```

Basic Git commands

```
$ git <command> <options>
```

E.g.

- ▶ `git add myfile.txt`
- ▶ `git status`

Basic Git commands

```
$ git <command> <options>
```

E.g.

- ▶ `git add myfile.txt`
- ▶ `git status`
- ▶ `git commit -m "add myfile"`

Basic Git commands

```
$ git <command> <options>
```

E.g.

- ▶ `git add myfile.txt`
- ▶ `git status`
- ▶ `git commit -m "add myfile"`
- ▶ `git log`

More Git commands

► `git commit --help`

More Git commands

- ▶ `git commit --help`
- ▶ `git checkout 5d34bf1 myfile.txt`

More Git commands

- ▶ `git commit --help`
- ▶ `git checkout 5d34bf1 myfile.txt`
- ▶ `git checkout 5d34bf1`

More Git commands

- ▶ `git commit --help`
- ▶ `git checkout 5d34bf1 myfile.txt`
- ▶ `git checkout 5d34bf1`
- ▶ `git checkout HEAD myfile.txt`

More Git commands

- ▶ `git commit --help`
- ▶ `git checkout 5d34bf1 myfile.txt`
- ▶ `git checkout 5d34bf1`
- ▶ `git checkout HEAD myfile.txt`
- ▶ `git revert 5d34bf1`

Episode 5: Recovering Older Versions of a File

Which commands below will let you recover the last committed version of the Python script called `data_cruncher.py`?

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

Episode 5: Reverting a Commit

Below are the right steps and explanations for Jennifer to use `git revert`¹, what is the missing command?

1. _____ # Look at the git history of the project to find the commit ID
2. Copy the ID (the first few characters of the ID, e.g. 0b1d055).
3. `git revert [commit ID]`
4. Type in the new commit message.
5. Save and close

¹The command `git revert` is different from `git checkout [commit ID]` because `git checkout` returns the files not yet committed within the local repository to a previous state, whereas `git revert` reverses changes committed to the local and project repositories.

Episode 5: Understanding Workflow and History

What is the output of the last command in:

```
$ cd planets
$ echo "Venus is beautiful and full of love" > venus.txt
$ git add venus.txt
$ echo "Venus is too hot to be suitable as a base" >> venus.txt
$ git commit -m "Comment on Venus as an unsuitable base"
$ git checkout HEAD venus.txt
$ cat venus.txt
```

Episode 5: Understanding Workflow and History

What is the output of the last command in:

```
$ cd planets
$ echo "Venus is beautiful and full of love" > venus.txt
$ git add venus.txt
$ echo "Venus is too hot to be suitable as a base" >> venus.txt
$ git commit -m "Comment on Venus as an unsuitable base"
$ git checkout HEAD venus.txt
$ cat venus.txt
```

Answer: Venus is beautiful and full of love

Episode 6: Ignoring Nested Files

Given a directory structure that looks like:

```
results/data
```

```
results/plots
```

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

Episode 6: Ignoring Nested Files

Given a directory structure that looks like:

```
results/data
```

```
results/plots
```

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

```
results/plots/
```

Episode 6: Including specific files

How would you ignore all `.dat` files in your root directory except for `final.dat`?

Hint: Find out what `!` (the exclamation point operator) does

Episode 6: Including specific files

How would you ignore all .dat files in your root directory except for `final.dat`?

Hint: Find out what `!` (the exclamation point operator) does

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

Episode 6: Ignoring Nested Files: Variation

Given a directory structure that looks similar to the earlier Nested Files exercise, but with a slightly different directory structure:

```
results/data  
results/images  
results/plots  
results/analysis
```

How would you ignore all of the contents in the results folder, but not results/data?

Hint: think a bit about how you created an exception with the ! operator before.

Episode 6: Ignoring Nested Files: Variation

Given a directory structure that looks similar to the earlier Nested Files exercise, but with a slightly different directory structure:

```
results/data  
results/images  
results/plots  
results/analysis
```

How would you ignore all of the contents in the results folder, but not results/data?

Hint: think a bit about how you created an exception with the ! operator before.

```
results/*           # ignore everything in results folder  
!results/data/      # do not ignore results/data/ contents
```


Episode 9: A Typical Worksession

Put the following actions in order, and give the commands needed to achieve the action:

- ▶ Make changes by appending the number 100 to a text file numbers.txt
- ▶ Update remote repository to match the local repository
- ▶ Celebrate your success with some fancy beverage(s)
- ▶ Update local repository to match the remote repository
- ▶ Stage changes to be committed
- ▶ Commit changes to the local repository

Episode 9: A Typical Worksession

order	action	command
1		
2		
3		
4		
5		
6	Celebrate!	AFK

Episode 9: A Typical Worksession

order	action	command
1	Update local	git pull origin main
2		
3		
4		
5		
6	Celebrate!	AFK

Episode 9: A Typical Worksession

order	action	command
1	Update local	git pull origin main
2	Make changes	echo 100 >>numbers.txt
3		
4		
5		
6	Celebrate!	AFK

Episode 9: A Typical Worksession

order	action	command
1	Update local	<code>git pull origin main</code>
2	Make changes	<code>echo 100 >>numbers.txt</code>
3	Stage changes	<code>git add numbers.txt</code>
4		
5		
6	Celebrate!	AFK

Episode 9: A Typical Worksession

order	action	command
1	Update local	<code>git pull origin main</code>
2	Make changes	<code>echo 100 >>numbers.txt</code>
3	Stage changes	<code>git add numbers.txt</code>
4	Commit changes	<code>git commit -m "Add 100 to numbers.txt"</code>
5		
6	Celebrate!	AFK

Episode 9: A Typical Worksession

order	action	command
1	Update local	<code>git pull origin main</code>
2	Make changes	<code>echo 100 >>numbers.txt</code>
3	Stage changes	<code>git add numbers.txt</code>
4	Commit changes	<code>git commit -m "Add 100 to numbers.txt"</code>
5	Update remote	<code>git push origin main</code>
6	Celebrate!	AFK

Git in Practice

- ▶ I am a big fan!
- ▶ Great way to be transparent
- ▶ GitHub can be linked to OSF and Zenodo
- ▶ No need to use command line

Reflection

- ▶ What would you like to do with git?
- ▶ What do like about git?
- ▶ What don't you like about git?