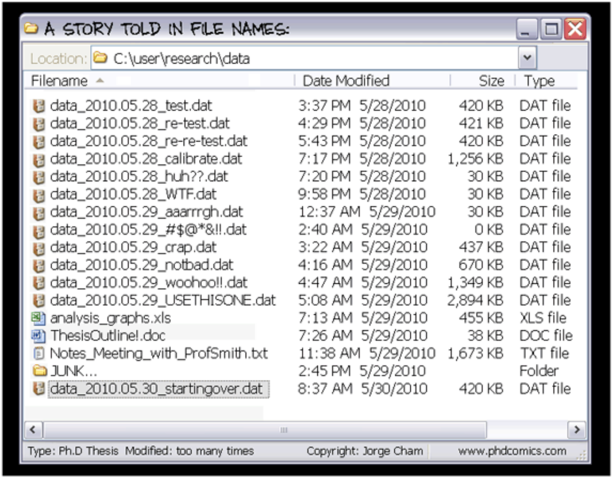
Introduction to Version Control with `Git`

Why do I need version control?

Why do I need version control?



A STORY TOLD IN FILE NAMES:

Location: C:\user\research\data

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com

https://www.phdcomics.com

Why do I need version control?

- Make changes to code with confidence - *can always be reverted if necessary*

Why do I need version control?

- Make changes to code with confidence - *can always be reverted if necessary*
- Reproducibility - *version control can complement your lab notebook*

Why do I need version control?

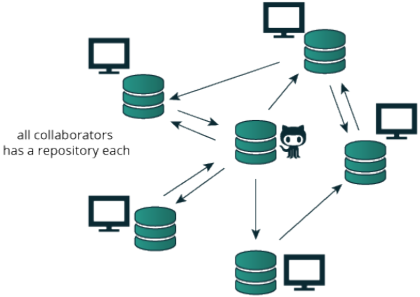
- Make changes to code with confidence - *can always be reverted if necessary*
- Reproducibility - *version control can complement your lab notebook*
- Work as a team - *file names and directory structures are consistent for all team members*

Why do I need version control?

- Make changes to code with confidence - *can always be reverted if necessary*
- Reproducibility - *version control can complement your lab notebook*
- Work as a team - *file names and directory structures are consistent for all team members*
- The list goes on...

Git

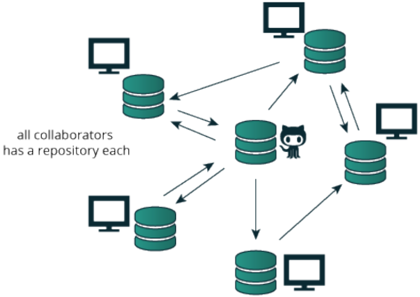
In the scientific world, Git (and Github) is the most widely used version control system.



http://invistruct.com/

Git

In the scientific world, Git (and Github) is the most widely used version control system.



repository: A central storage area where a version control system stores old revisions of files and information about who changed what, when.

http://invistruct.com/

How do you get your own repository?

Let's configure Git first:

```
$ git config --global user.name "Your name goes here"
$ git config --global user.email you@yourdomain.com
$ git config --global core.editor vim
$ git config color.ui auto
```

Then initialize your first repository:

```
$ git init
```

How do you get your own repository?

Let's configure Git first:

```
$ git config --global user.name "Your name goes here"
$ git config --global user.email you@yourdomain.com
$ git config --global core.editor vim
$ git config color.ui auto
```

Then initialize your first repository:

```
$ git init
```

You Try (10 minutes):

Exercises (1) - 2

Git allows you to save snapshots of your directory

commit: snapshots of your directory.

Git allows you to save snapshots of your directory

commit: snapshots of your directory.

- There is metadata associated with each commit (snapshot):
  - the date the snapshot was taken
  - who took it
  - what files were modified
  - the changes made on those files
  - etc.

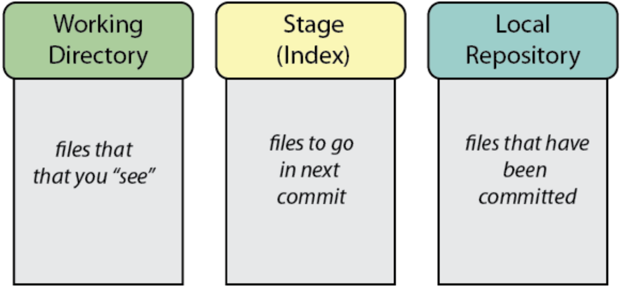
`Git` allows you to save snapshots of your directory

commit: snapshots of your directory.

- There is metadata associated with each commit (snapshot):
  - the date the snapshot was taken
  - who took it
  - what files were modified
  - the changes made on those files
  - etc.
- `Git` will enable you to:
  - track the changes made to files in your directory
  - revert the entire project to a previous snapshot
  - review changes made over time
  - view who modified a file
  - etc.

A little more vocabulary:

There are three main *trees* or *collections of files (and metadata)* in `Git`:



Working Directory

files that that you "see"

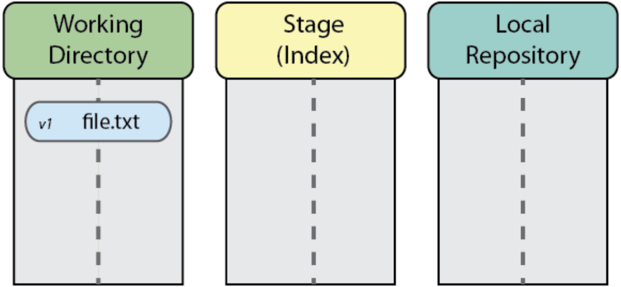
Stage (Index)

files to go in next commit

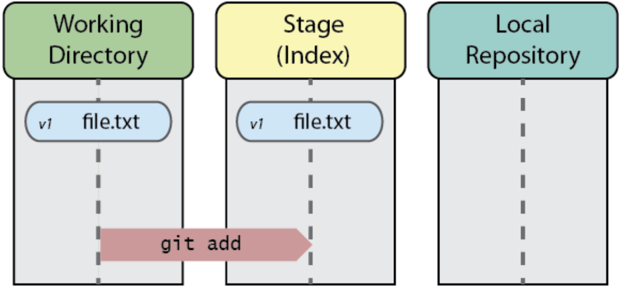
Local Repository

files that have been committed

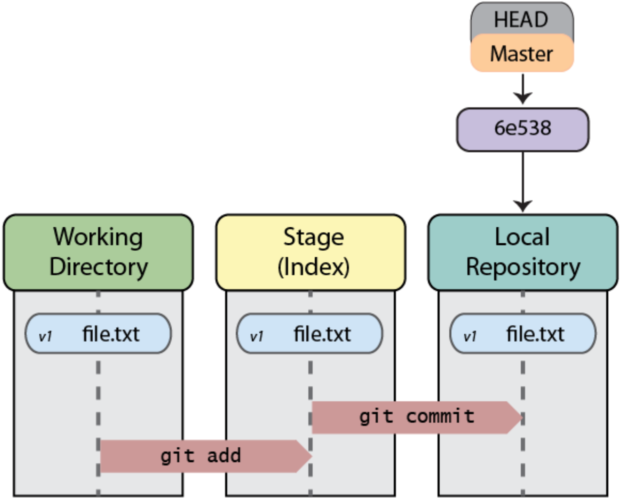
How to save snapshots with Git



How to save snapshots with Git

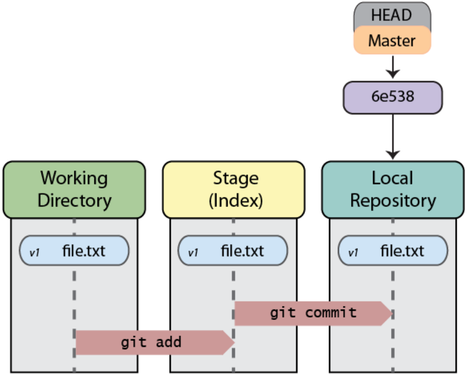


How to save snapshots with Git



How to save snapshots with `Git`

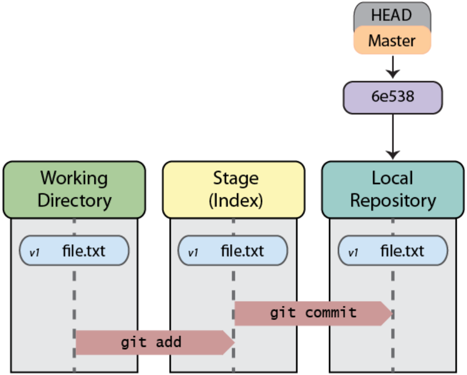
SHA-1 hash: unique 40-digit computer-generated identifier for each revision (or commit)



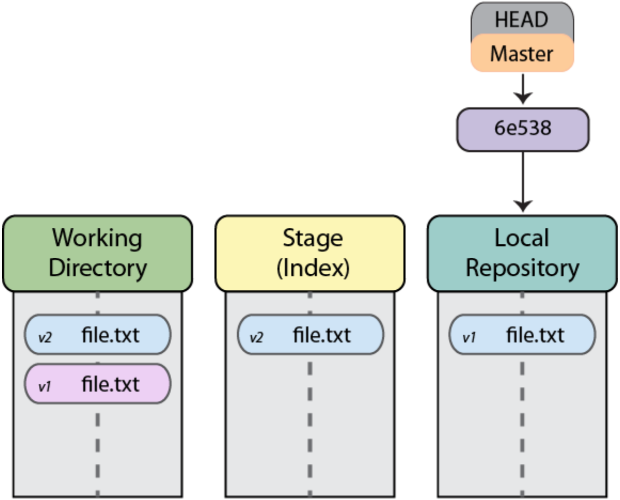
How to save snapshots with `Git`

SHA-1 hash: *unique* 40-digit computer-generated identifier for each revision (or commit)

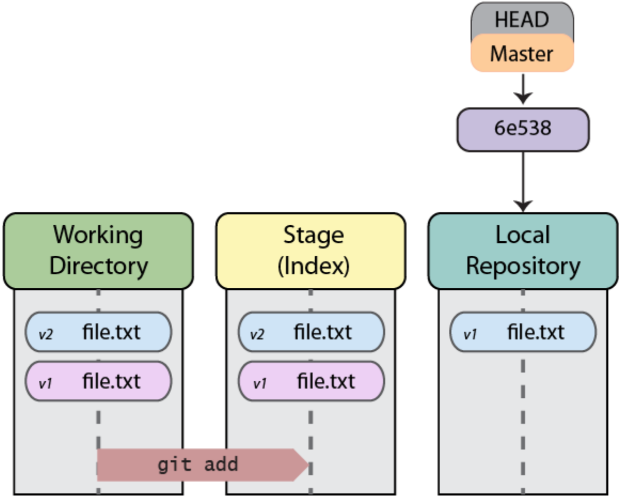
HEAD: reference to the current branch or commit



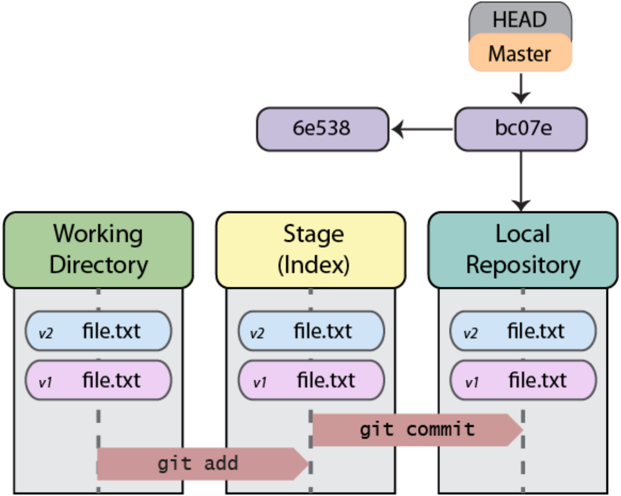
How to save snapshots with `Git`: Keep working!



How to save snapshots with Git: Keep working!



How to save snapshots with Git: Keep working!

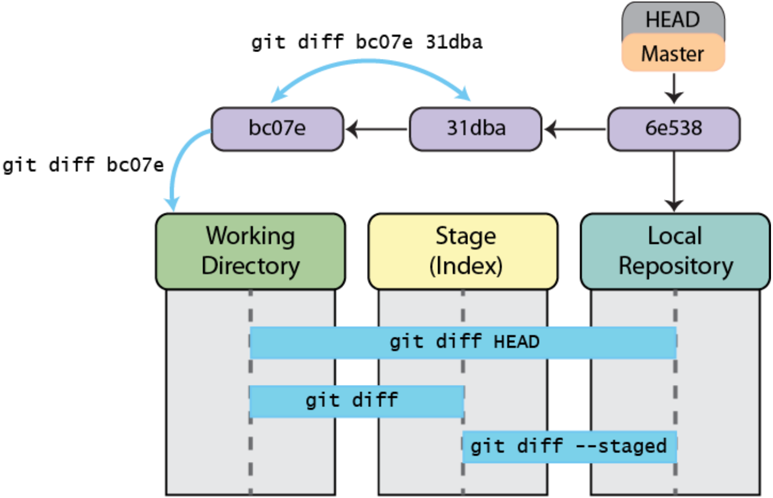


A simple story so far: what else can we do?!

`git log`: view the change history (commits) of the current repository.

A simple story so far: what else can we do?!

`git diff`: view changes between files and commits



How do we do this for real?

An Example

Now it's your turn.

Questions?

Now it's your turn.

Questions?

You Try (15 minutes):
Exercises 3