Introduction to Version Control & Git

./sdc

Haroon Chughtai | 2019-03-01

In this hour, I’ll…

Teach you all the magical things that version control can do for you

In this hour, I’ll…

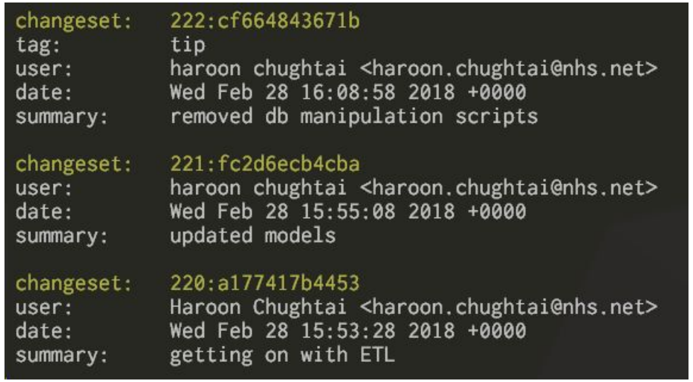
Teac nagical
things th on control
c you

In this hour, I’ll…

Run through the basics of version control using Git*

* I have nothing against, and use Mercurial (Hg) myself on occasion. I’ll explain my choice in a later slide.

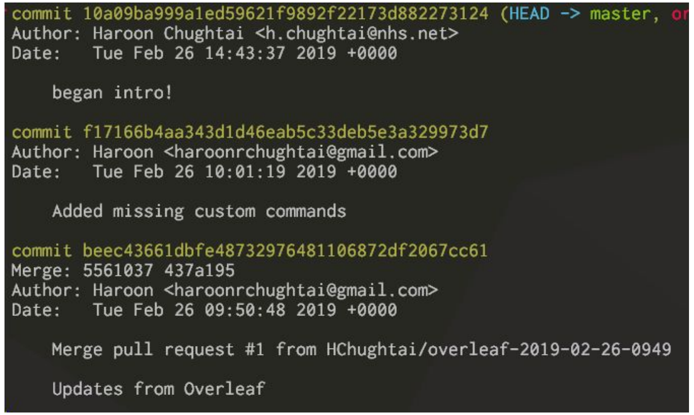
How Do I Use VCS?



```
changeset:   222:cf664843671b
tag:         tip
user:        haroon chughtai <haroon.chughtai@nhs.net>
date:        Wed Feb 28 16:08:58 2018 +0000
summary:     removed db manipulation scripts

changeset:   221:fc2d6ecb4cba
user:        haroon chughtai <haroon.chughtai@nhs.net>
date:        Wed Feb 28 15:55:08 2018 +0000
summary:     updated models

changeset:   220:a177417b4453
user:        Haroon Chughtai <haroon.chughtai@nhs.net>
date:        Wed Feb 28 15:53:28 2018 +0000
summary:     getting on with ETL
```

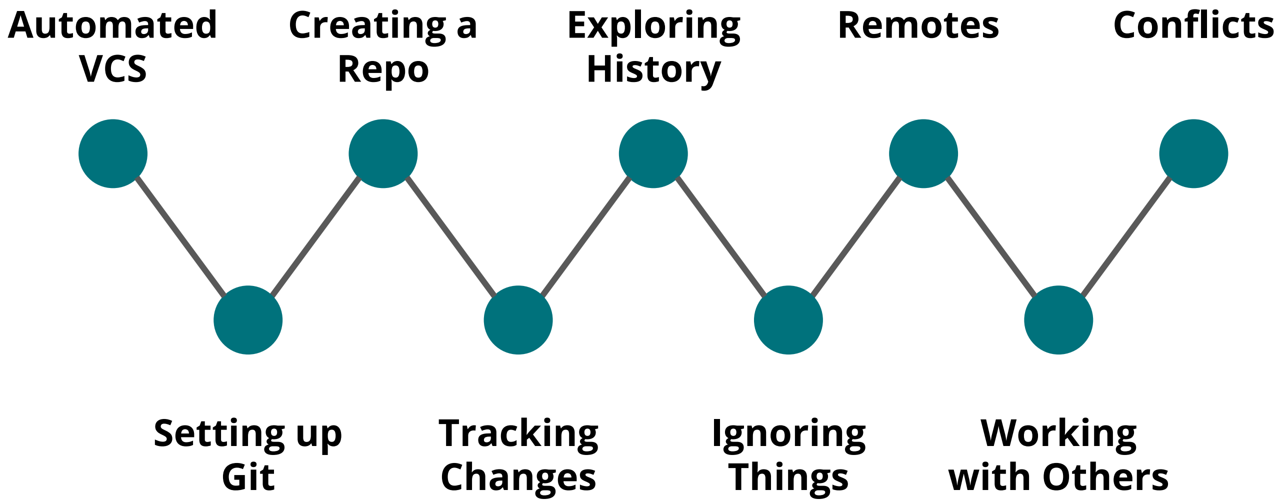
Software Development

Document Control



```
commit 10a09ba999a1ed59621f9892f22173d882273124 (HEAD -> master, or
Author: Haroon Chughtai <h.chughtai@nhs.net>
Date:   Tue Feb 26 14:43:37 2019 +0000

    began intro!

commit f17166b4aa343d1d46eab5c33deb5e3a329973d7
Author: Haroon <haroonrchughtai@gmail.com>
Date:   Tue Feb 26 10:01:19 2019 +0000

    Added missing custom commands

commit beec43661dbfe48732976481106872df2067cc61
Merge: 5561037 437a195
Author: Haroon <haroonrchughtai@gmail.com>
Date:   Tue Feb 26 09:50:48 2019 +0000

    Merge pull request #1 from HChughtai/overleaf-2019-02-26-0949

    Updates from Overleaf
```

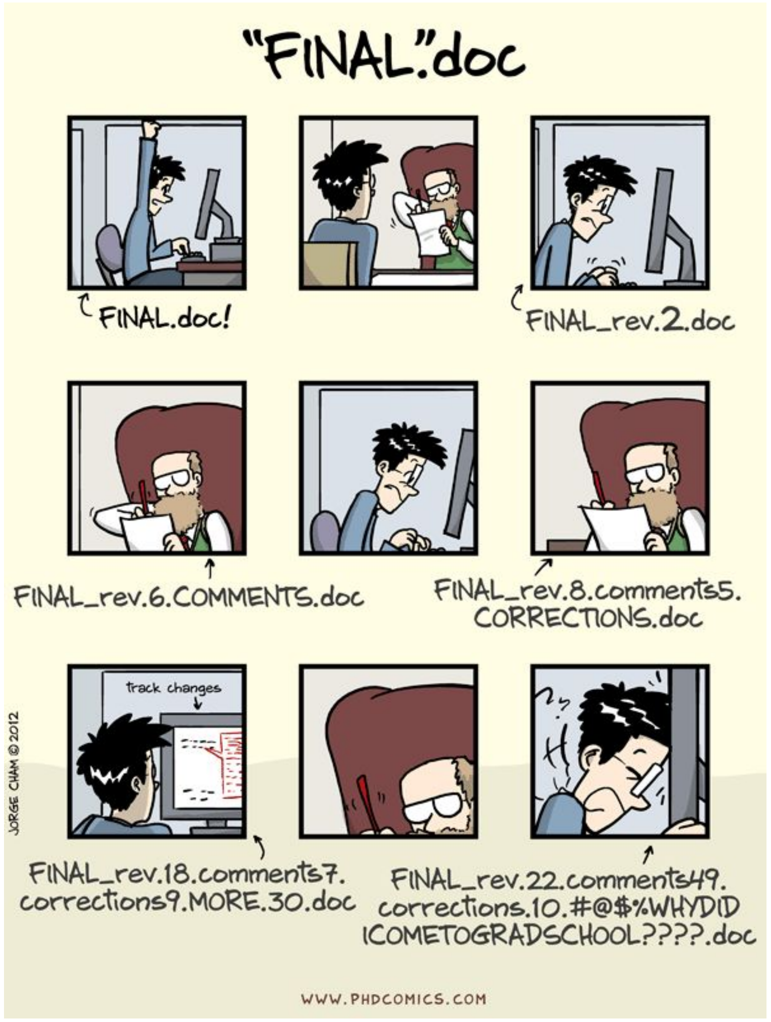
The Plan

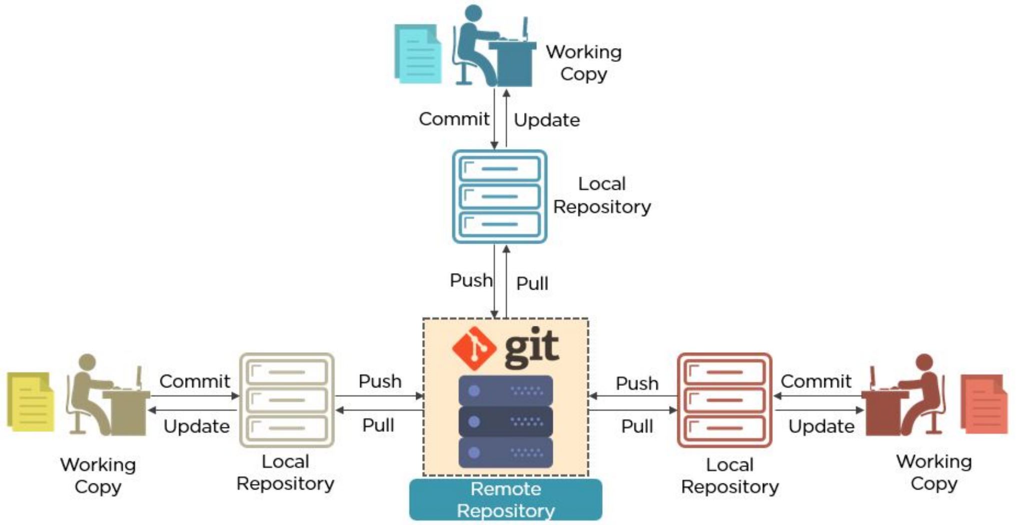


Shamelessly derived from http://swcarpentry.github.io/git-novice

Why Should I Use Version Control?

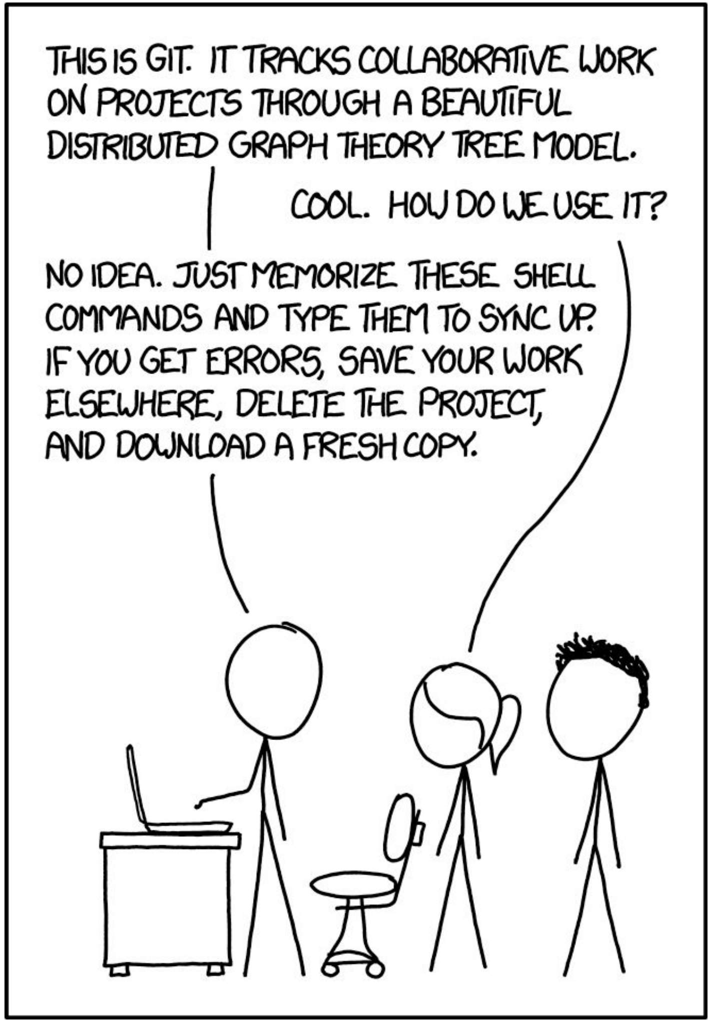
To Avoid This....


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5. CORRECTIONS.doc
track changes
FINAL_rev.18.comments7. corrections9.MORE.30.doc
FINAL_rev.22.comments49. corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM


Working Copy
Commit
Update
Local Repository
Push
Pull
git
Remote Repository
Working Copy
Commit
Update
Local Repository
Push
Pull
Push
Pull
Local Repository
Commit
Update
Working Copy

And do this…

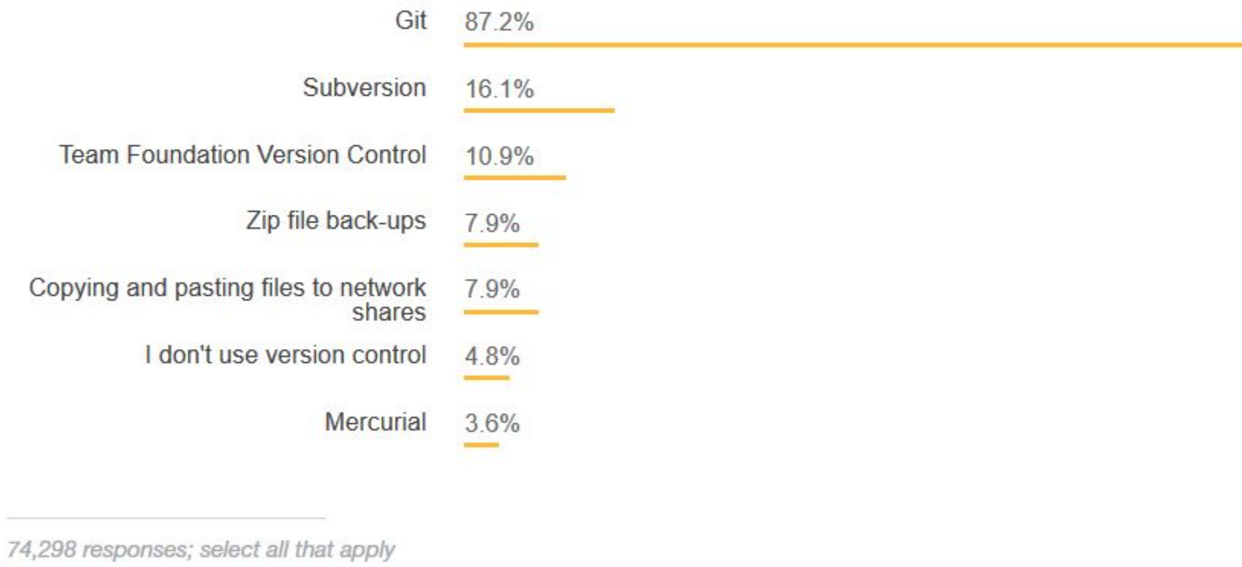
What is Git?


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

git

Yes, there are other VCS systems….

We're using Git as it's widely used in the broader software community, so likely to need it when working with others.



https://insights.stackoverflow.com/survey/2018/#work-version-control

Setting Up Git

```
# Set Up Your Details
$ git config --global user.name "Haroon Chughtai"
$ git config --global user.email "h.chughtai@nhs.net"

# Configure Line Endings
$ git config --global core.autocrlf input # macOS & Linux
$ git config --global core.autocrlf true # Windows

# Choose an Editor
$ git config --global core.editor "nano -w"

# Set Up Proxy
$ git config --global http.proxy proxy-url
$ git config --global https.proxy proxy-url
```

N.B. The UCLH proxy address is `http://<uclh_username>:<uclh_pwd>@www-cache-n:3128`

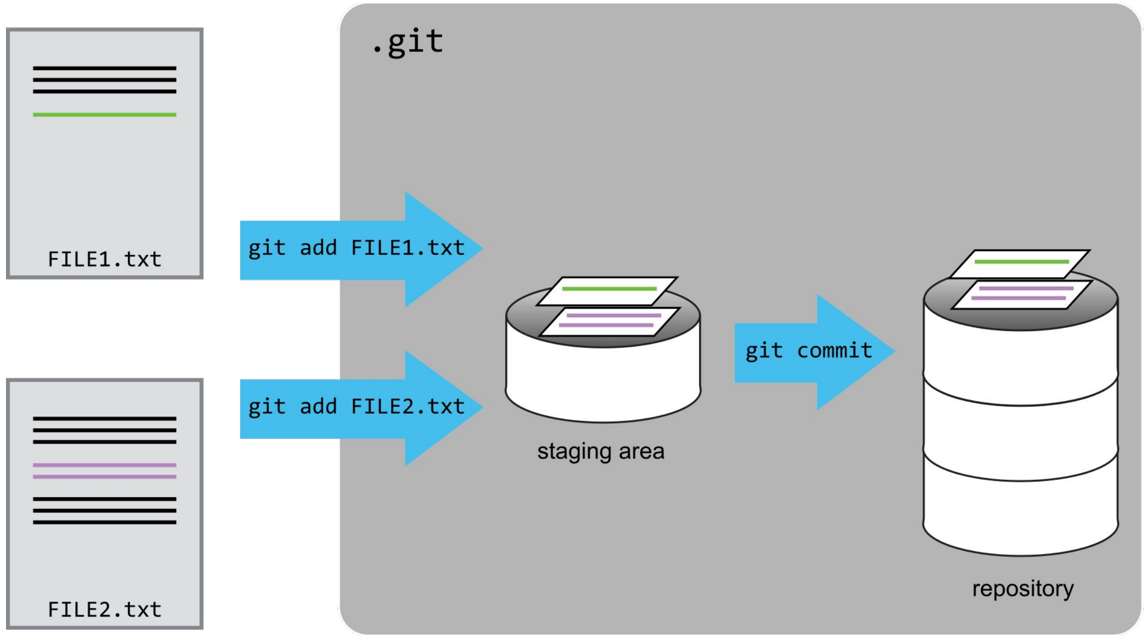
Creating a Repository

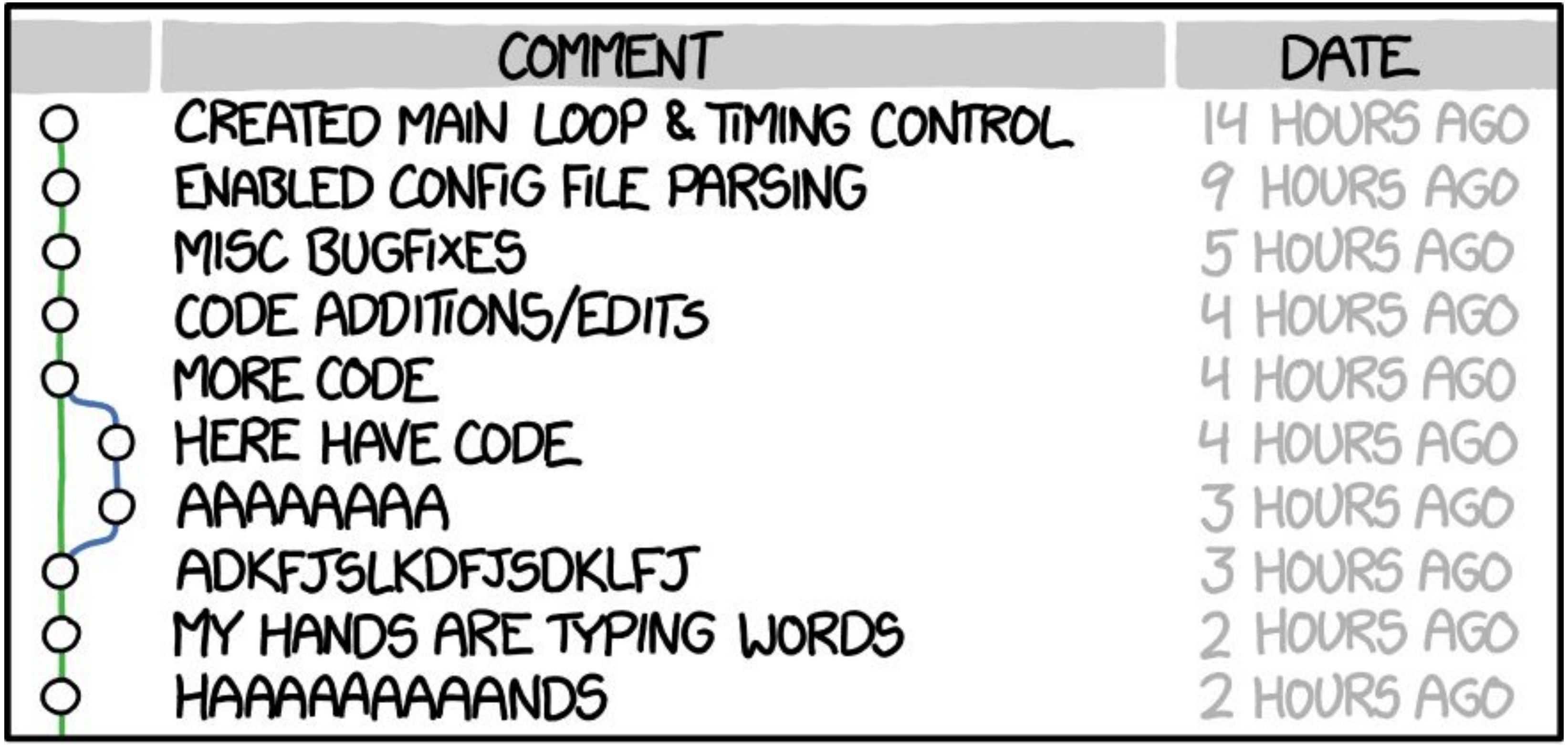
1. Create a project directory
2. Run `git init`

That’s It!

Tracking Changes

```
$ git add <filename>
$ git commit -m "Reason for change"
```





	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Exploring History

```
# Compare with previous versions of a file
$ git diff HEAD <filename> # the last commit
$ git diff HEAD~1 <filename> # the commit 1 behind the last commit
$ git diff <commit_id> <filename> # a specific commit

# Restore a previous version of a file to the working directory
$ git checkout HEAD <filename>

# Detaches head and should be used as a read-only view
$ git checkout <commit_id>
# Reattaches head and puts repo back into a safe state
$ git checkout master
```

Ignoring Things

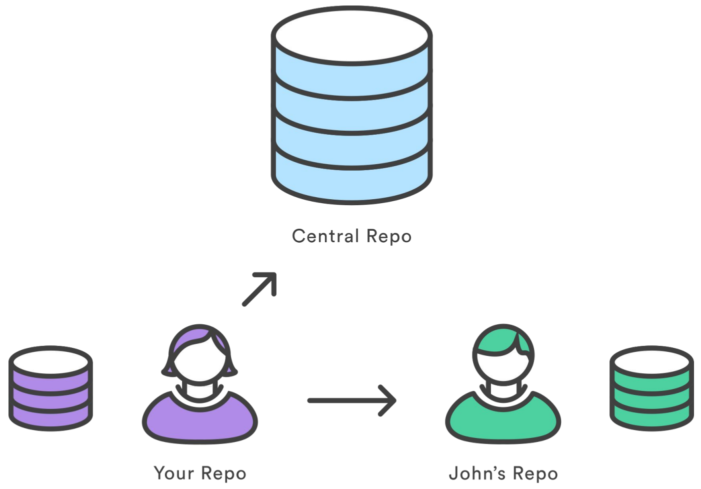
1. Create and commit a `.gitignore` file in the project's root directory

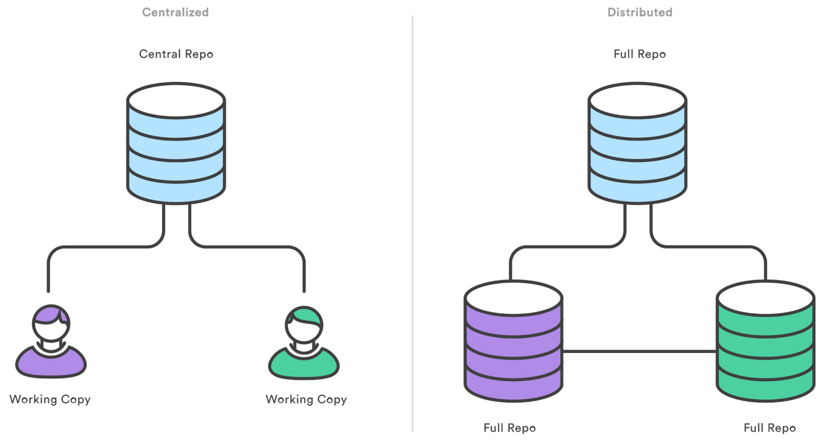
2. Fill it with files and folders you want to ignore

```
### Python ###
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
pip-wheel-metadata/
share/python-wheels/
```

Remotes

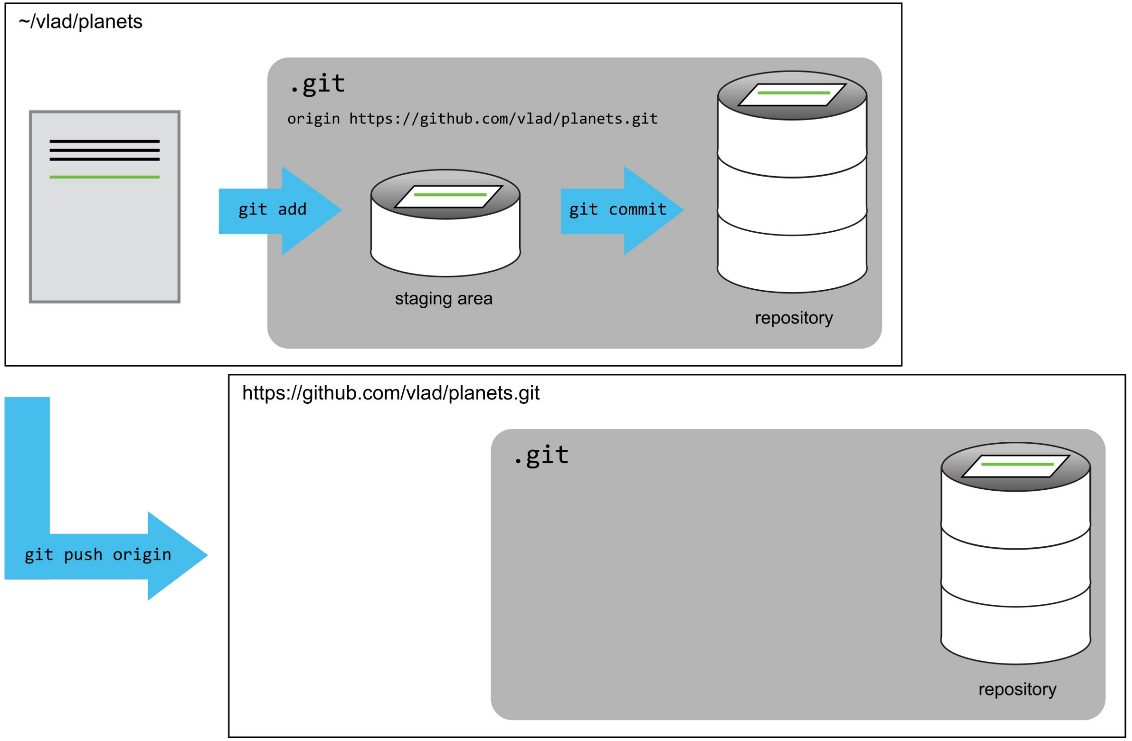
In order to collaborate we need to be able to copy changes from one repository to another





We can move between any two repositories, but normally use one hosted copy as a central hub

```
$ git remote add origin <remote repository address>
$ git push origin master
```



Collaborating

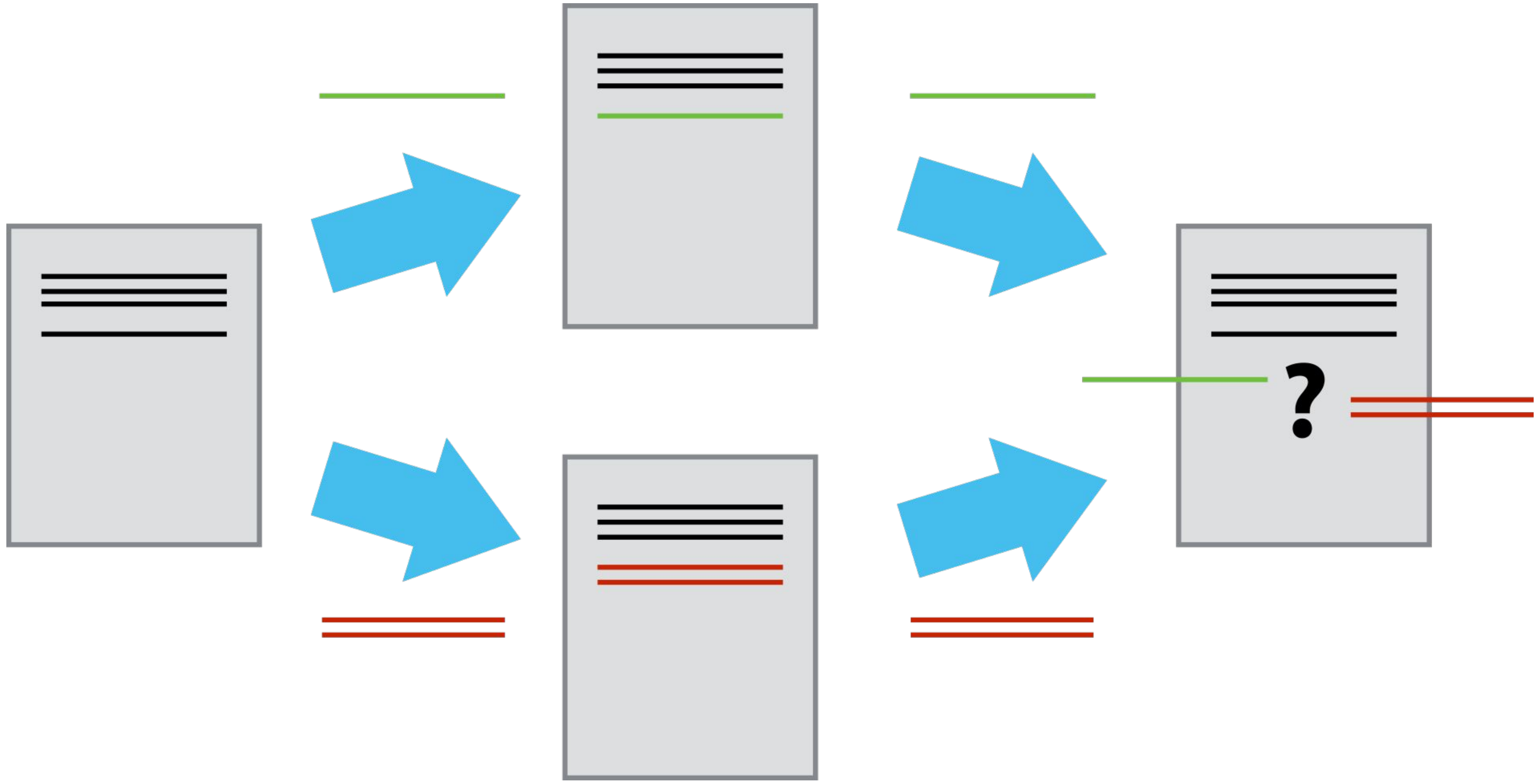
```
# get someone’s repo
$ git clone <someone's repo> <your local file path>

# to get changes
$ git pull origin master

# to make changes
$ git add <file>
$ git commit -m "meaningful comment"
$ git push origin master
```

```
git pull = git fetch + git merge
```

Conflicts

If two people are working in parallel, they will step on each other’s toes.

Version control helps us manage these *conflicts* by giving us tools to *resolve* overlapping changes.



```
To https://github.com/...
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/...'
hint: Updates were rejected because the tip of your current branch is
behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
Some common text here
<<<<<<< HEAD
A change existing in one version
=======
The change that only exists in the other version
>>>>>>> dabb4c8c450e8475aee9b14b4383acc99f42af1d
```

1. Resolve the conflicted file

2. Add and commit

3. Push the changes to the central repo

Technical approaches to reducing conflicts:

- Pull from upstream more frequently
- Use topic branches to segregate work, merging to master when complete
- Make smaller more atomic commits
- Break large files into smaller ones so that it is less likely that two authors will alter the same file simultaneously

Project management strategies to reduce conflicts:

- Clarify who is responsible for what areas with your collaborators
- Discuss what order tasks should be carried out in with your collaborators so that tasks expected to change the same lines won't be worked on simultaneously
- If the conflicts are stylistic, establish a project convention that is governing and use code style tools to enforce

What We Didn’t Cover

Branching

Collaborative Workflows

Pull Requests & Review

Hosting Choices

Advanced Git Commands

Resource List

# What this talk was based on
http://swcarpentry.github.io/git-novice/

# Some notes that I like
https://www.atlassian.com/git/tutorials

# An amazing reference
https://github.com/k88hudson/git-flight-rules

# An interactive git tutorial app
https://github.com/jlord/git-it-electron

# An interactive git branching tutorial
https://learngitbranching.js.org/

# Lots of posts by devs at different career stages
https://dev.to/search?q=Git