Distributed version control with Git

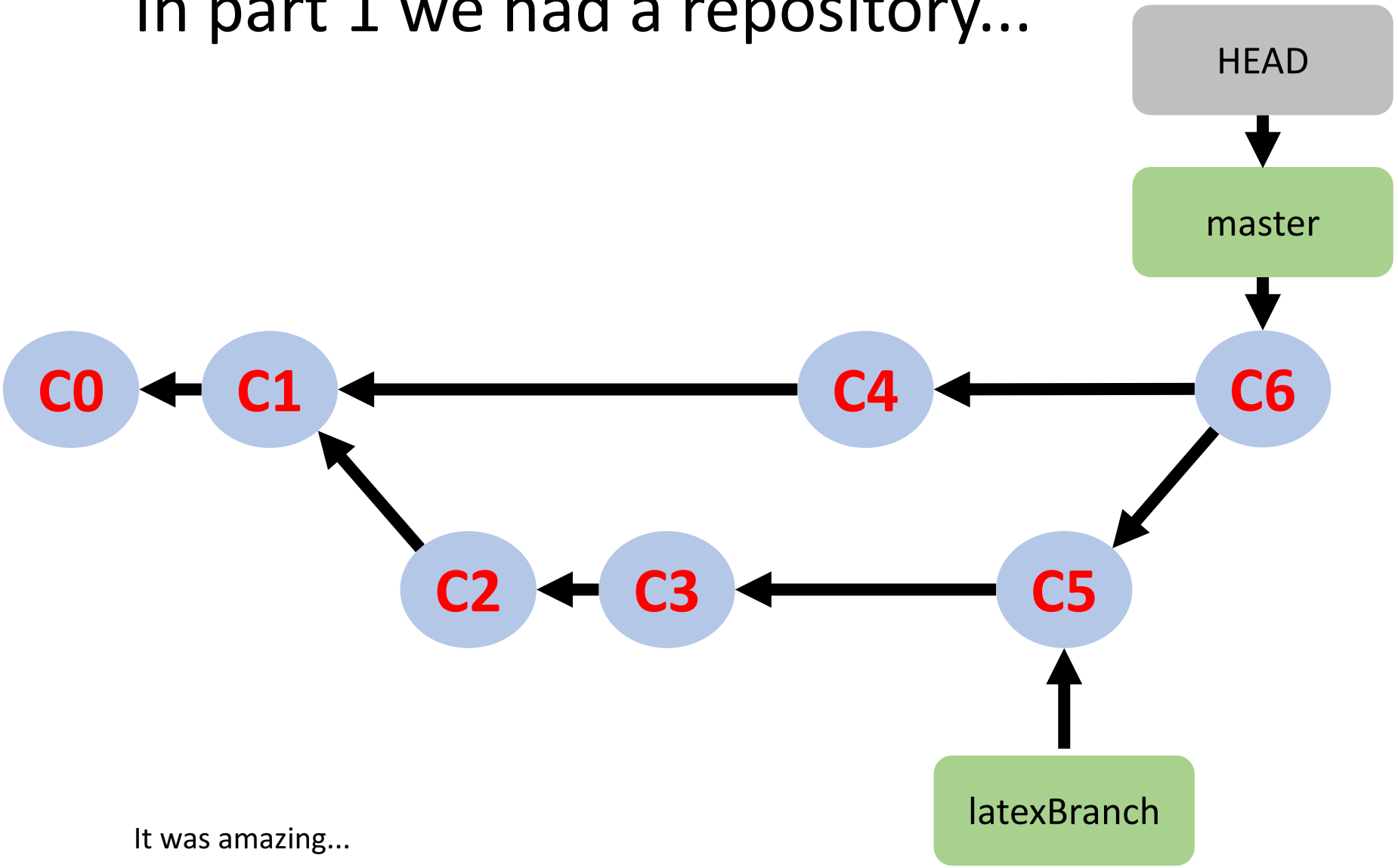
Part 2:
Working with remotes

Daniel Sussman

https://gitlab.com/dmsussman/workingWithGit/

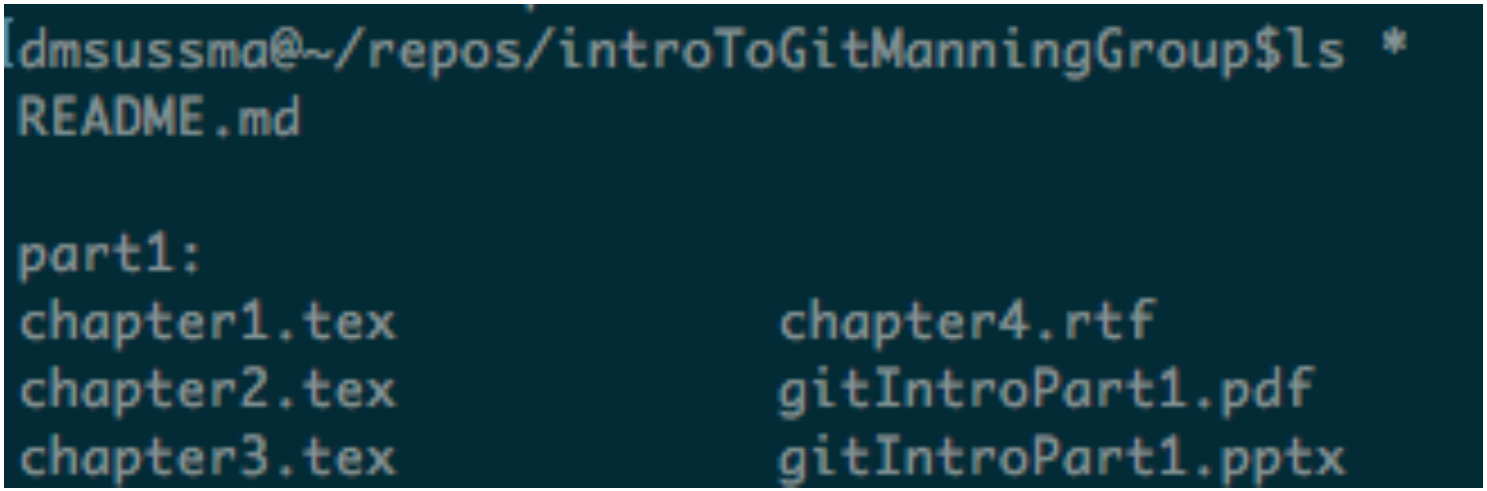
SYRACUSE SOFT
MATTER PROGRAM

In part 1 we had a repository...



It was amazing...

Since then, I've put the files and slides in a subdirectory...



Just working locally is good...
but it isn't very distributed...

Why use version control?

Team work:

* Makes working collaboratively much easier

Individual work:

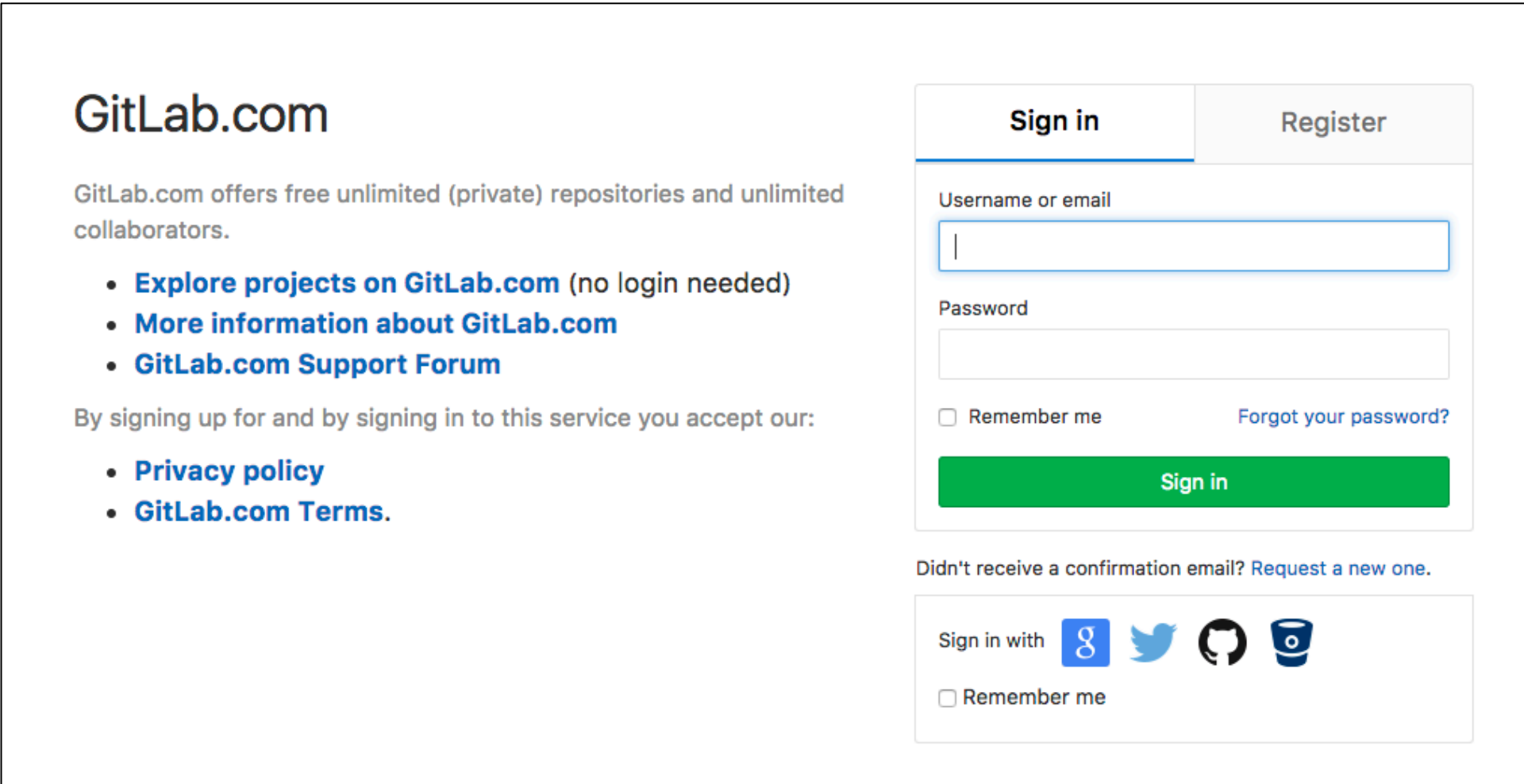
* Scientific reproducibility of code
* Magical time machine for reverting to previous version of code

Remote repositories will help achieve some of these aims

Step 0: Make a GitLab account

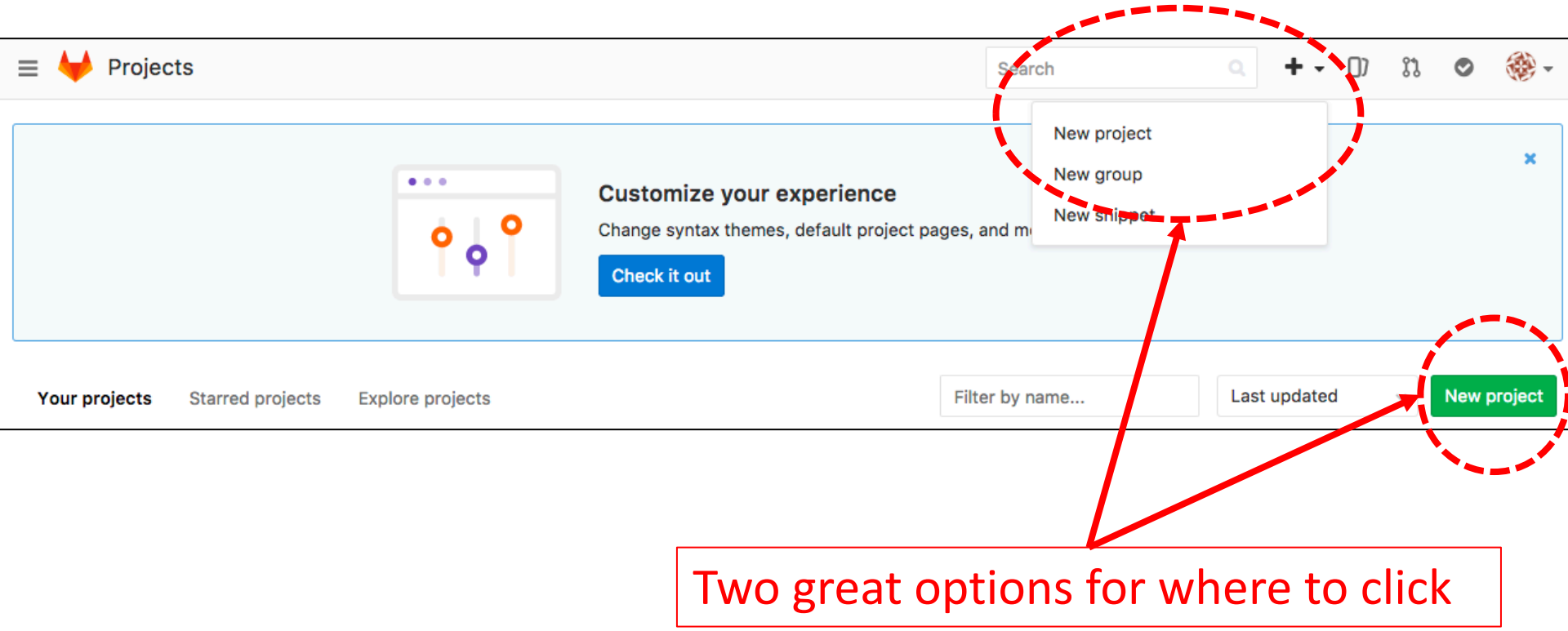
GitLab – an open source, web-based git repository manager

Alternatives: GitHub, BitBucket...

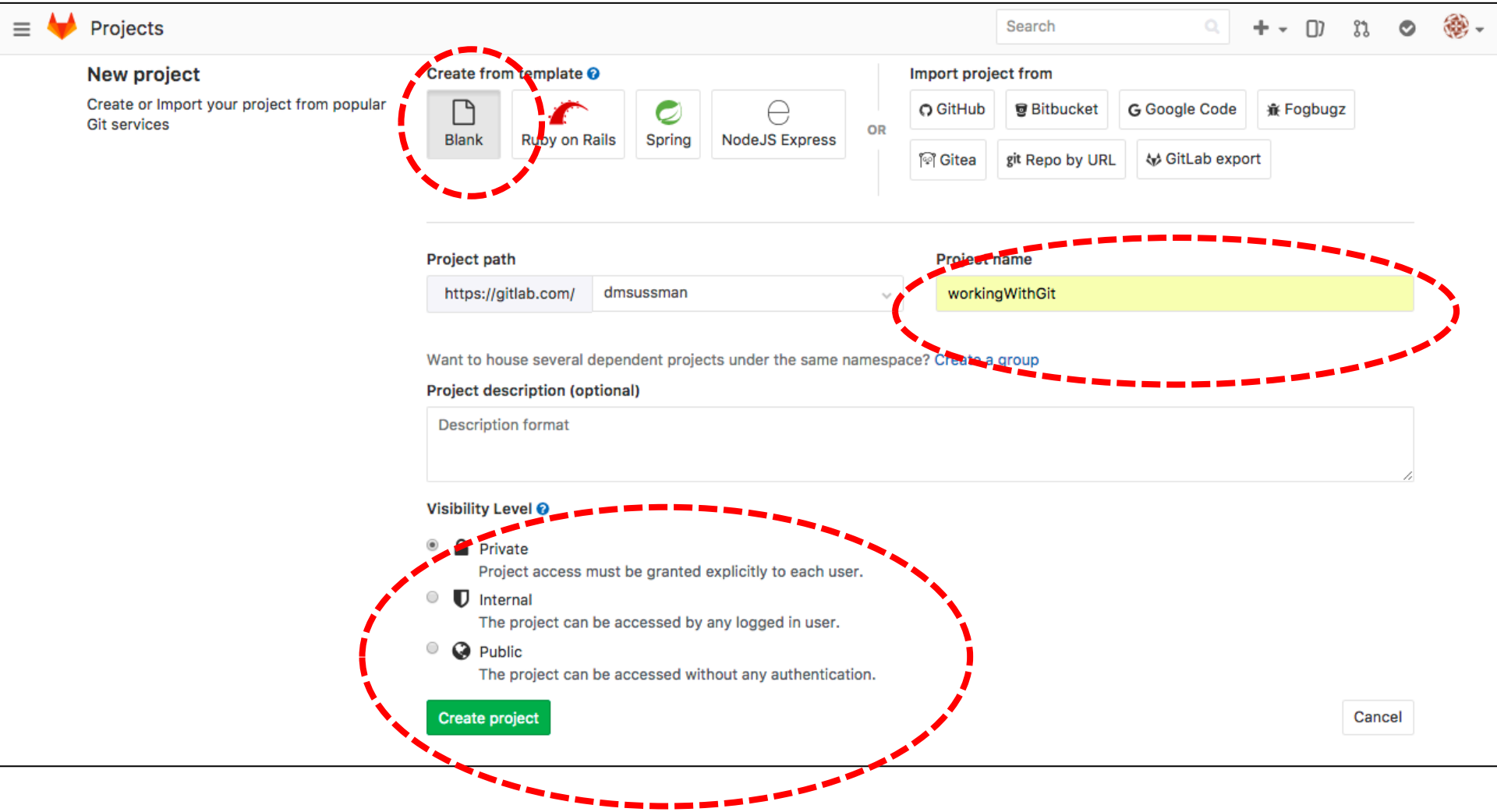


Let’s set up a new project

“Project” being code for “repository”



Let’s set up a new project



And *push* our existing repository to the cloud!

Assuming we're already in the project directory:

```
$ git remote add origin https://gitlab.com/dmsussman/workingWithGit.git
$ git push –u origin --all
```

GitLab provides step by step instructions for exactly this

Comment: google any git-related question, and you will find the answer

There is now an equivalent repository...in the cloud!

W

workingWithGit

A repository to house a gentle introduction to working with git

Unstar 2 | Fork 0 | SSH | git@gitlab.com:dmsussman/work: | Global

Files (16.4 MB) | Commits (11) | Branches (2) | Tags (0) | Readme | Add Changelog | Add License | Add Contribution guide | Set up CI

master | workingWithGit / | History | Find file

move additional info up in the presentations
Daniel Sussman committed about 17 hours ago — **b35340fb**

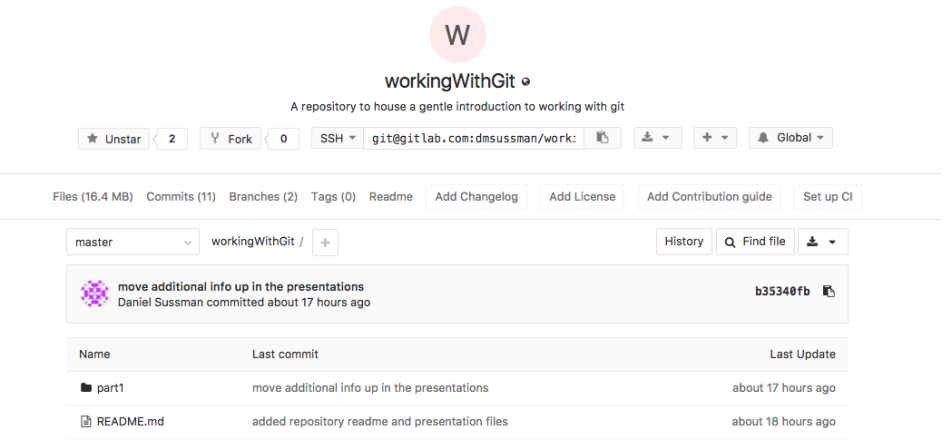
Name	Last commit	Last Update
part1	move additional info up in the presentations	about 17 hours ago
README.md	added repository readme and presentation files	about 18 hours ago

Let's make a change and *git push* it

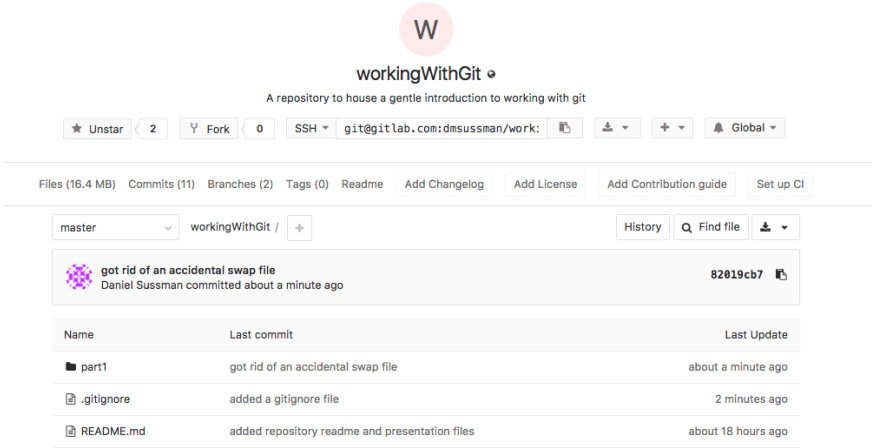
Specifies untracked files or *patterns of files* to ignore

```
$ vim .gitignore
$ git add .
$ git commit –m "added a gitignore file"
$git push
```

Before



After



Getting new local copies: *git clone*

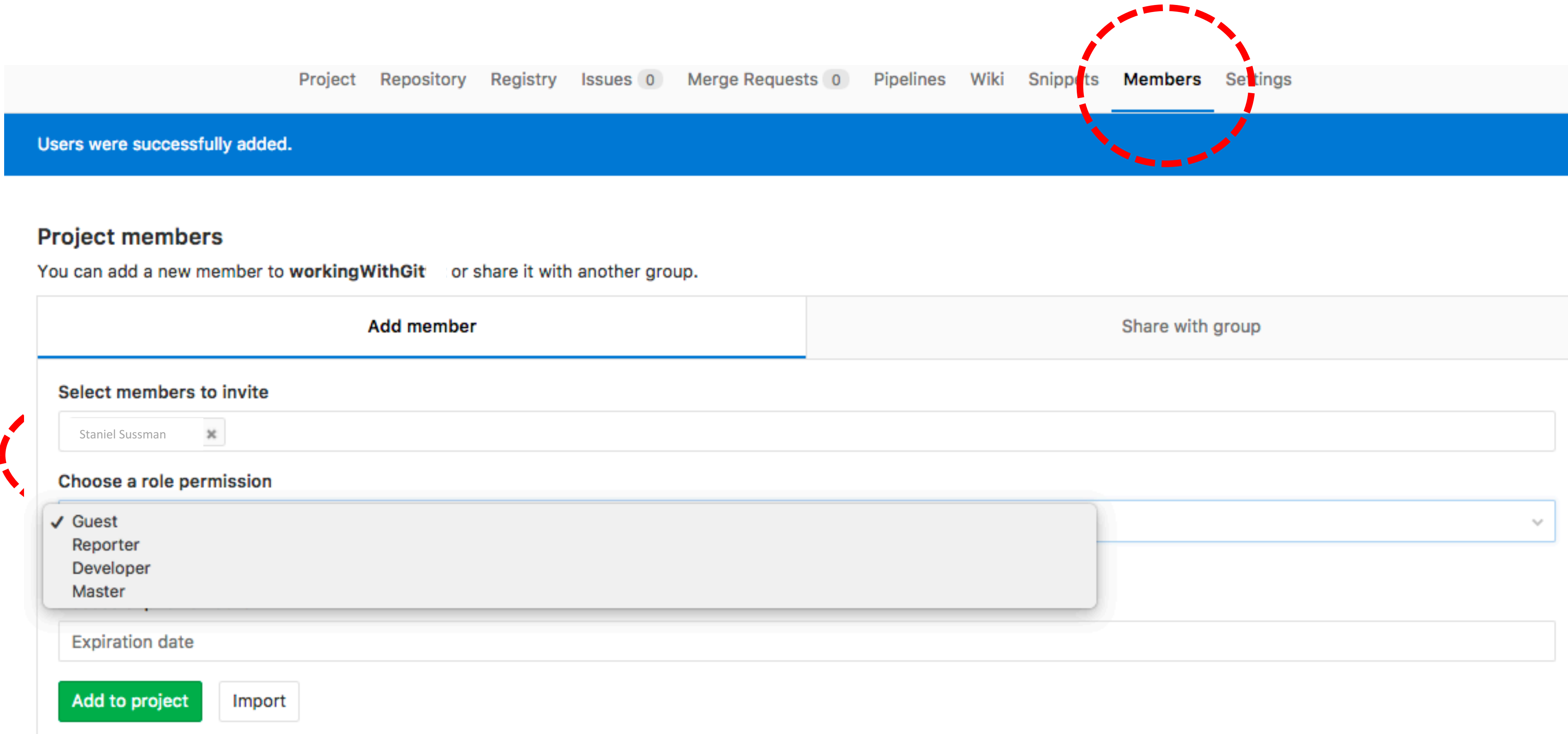
It is easy to clone new copies:

```
$ git clone https://gitlab.com/dmsussman/workingWithGit.git
                                                    newDirectory
```

This copies the repository at the specified location and puts it in a new directory (called, in this case “newDirectory”)

Work on multiple machines, use as a backup, ...

Let’s get collaborative!



Just searched for my friend, “Staniel Sussman” and added him to the project

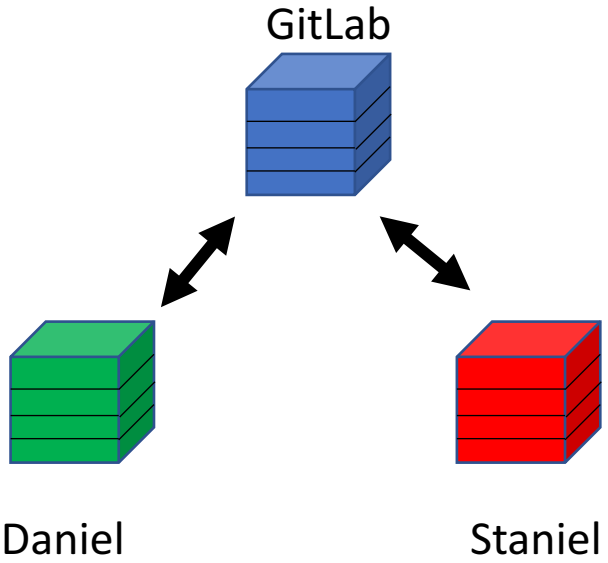
I’ll make him a “Master”, which gives him (almost) complete access to the repository

Staniel *git clones* the repository

Plot twist: “Staniel Sussman” is also me, using the Manning Group GitLab account.

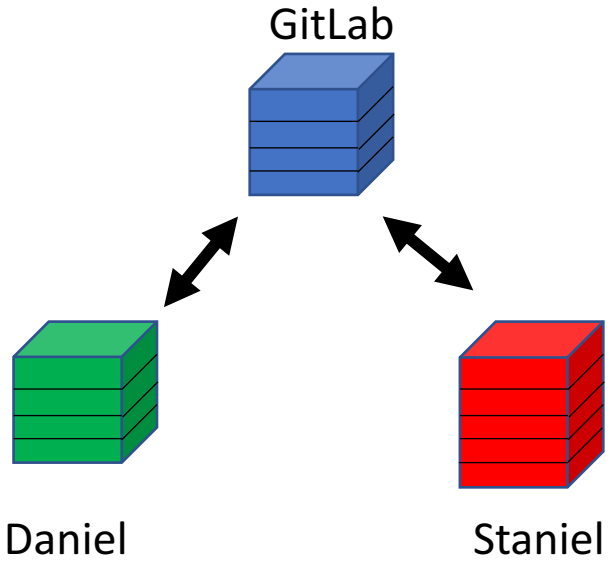
```
[dmsussma@~/repos/garbage$git clone https://ManningGroup@gitlab.com/dmsussman/workingWithGit.git
 Cloning into 'workingWithGit'...
[Password for 'https://ManningGroup@gitlab.com':
 remote: Counting objects: 51, done.
 remote: Compressing objects: 100% (48/48), done.
 remote: Total 51 (delta 19), reused 0 (delta 0)
 Unpacking objects: 100% (51/51), done.
```

Three independent but linked copies



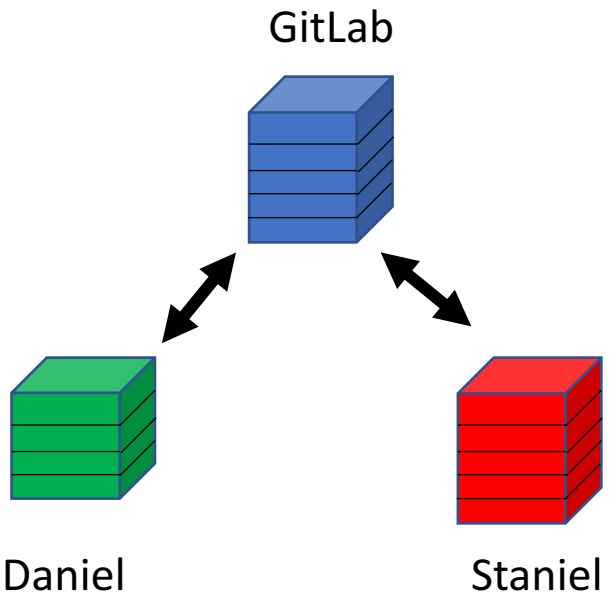
Staniel *edits* the repository

```
$ mkdir part2
$ vim part2/file1.txt
$ git add .
$ git commit -m "starting part2"
```



Staniel *pushes* the changes

```
$ git push
```

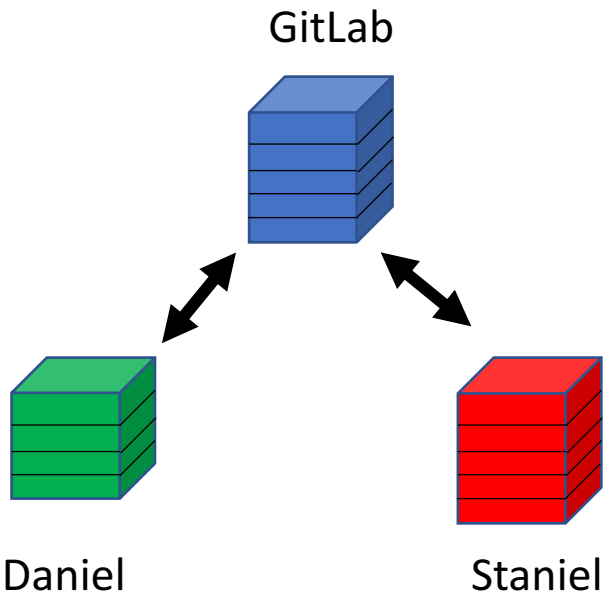


Daniel's repository *does not* auto-update

```
[dmsussma@~/repos/introToGitManningGroup$ls *
README.md

part1:
chapter1.tex            chapter3.tex            gitIntroPart1.pdf
chapter2.tex            chapter4.rtf            gitIntroPart1.pptx
```

```
[dmsussma@~/repos/introToGitManningGroup$git status
 On branch master
 Your branch is up-to-date with 'origin/master'.
 nothing to commit, working tree clean
```



Daniel *git pulls* changes from repository

```
dmsussma@~/repos/introToGitManningGroup$git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From https://gitlab.com/dmsussman/workingWithGit
   82019cb..a5ce1ff  master     -> origin/master
Updating 82019cb..a5ce1ff
Fast-forward
 part2/file1.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 part2/file1.txt
dmsussma@~/repos/introToGitManningGroup$ls *
README.md

part1:
chapter1.tex            chapter3.tex            gitIntroPart1.pdf
chapter2.tex            chapter4.rtf            gitIntroPart1.pptx

part2:
file1.txt
```

Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes
(2) Daniel edits *the same file* without pulling
(3) Daniel tries to push his changes?

Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel tries to push his changes?

```
$ vim part2/file1.txt
$ git add .
$ git commit -m “file1 in part2 edited by Stan”
$ git push
```

Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel tries to push his changes?

```
$ vim part2/file1.txt
$ git add .
$ git commit -m “file1 in part2 edited by Dan”
```

Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel tries to push his changes?

```
$ git push
```

```
dmsussma@~/repos/introToGitManningGroup$git push
To https://gitlab.com/dmsussman/workingWithGit.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://gitlab.com/dmsussman/workingWithGit.
git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel tries to pull?

```
$ git pull
```

```
[dmsussma@~/repos/introToGitManningGroup$git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From https://gitlab.com/dmsussman/workingWithGit
   7889268..478d09e  master     -> origin/master
Auto-merging part2/file1.txt
CONFLICT (content): Merge conflict in part2/file1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel checks the status?

$ git status

```
[dmsussma@~/repos/introToGitManningGroup$git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   part2/file1.txt
```

Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel looks at that file?

```
$ vim part2/file1.txt
```

introToGitManningGroup — vim part2/file1.txt

```
1 <<<<<<< HEAD
2 Staniel has struck first. Even so, Daniel is better.
3 =======
4 Staniel has struck first.
5
6 Also, Staniel is the best.
7 >>>>>>> 478d09e2fe29c22aed756a78f5e6e647eb48b31c
```

Managing conflicts

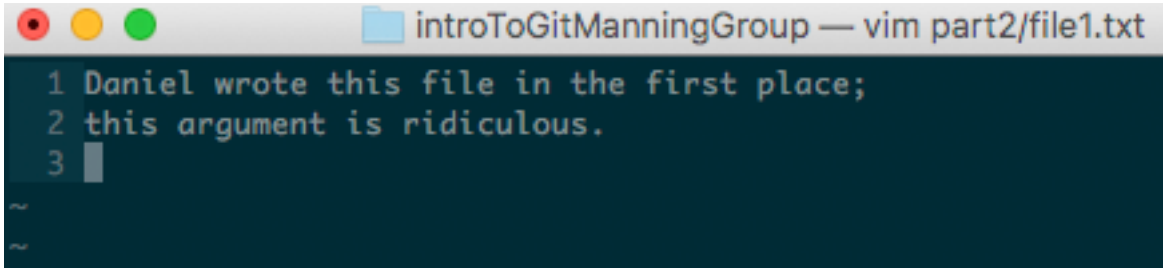
Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel *resolves the conflict however he wants*

```
$ vim part2/file1.txt
```



Managing conflicts

Suppose:

(1) Staniel edits a file and pushes the changes

(2) Daniel edits *the same file* without pulling

(3) Daniel pushes the changes to the repository

```
$ git add .
$ git commit -m "conflict managed"
$ git push
```

```
[dmsussma@~/repos/introToGitManningGroup$git push
Counting objects: 8, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 731 bytes | 0 bytes/s, done.
Total 8 (delta 2), reused 0 (delta 0)
To https://gitlab.com/dmsussman/workingWithGit.git
   478d09e..bcc741f  master -> master
```

Managing conflicts

All of the attempted commits, even the one Daniel couldn't push, become part of the repository's history...

```
[dmsussma@~/repos/introToGitManningGroup$git lg2
*   bcc741f - Wed, 13 Sep 2017 12:21:41 -0400 (5 minutes ago) (HEAD -> master, origin/master)
|\            conflict managed - Daniel Sussman
| * 478d09e - Wed, 13 Sep 2017 12:09:56 -0400 (17 minutes ago)
| |           file1 in part2 edited by Stan - Daniel Sussman
* | 5730c52 - Wed, 13 Sep 2017 12:11:15 -0400 (16 minutes ago)
|/            file1 in part2 edited by Dan - Daniel Sussman
* 7889268 - Wed, 13 Sep 2017 12:02:29 -0400 (24 minutes ago)
|           file1 in part 2 edited - Daniel Sussman
* a5ce1ff - Wed, 13 Sep 2017 11:31:52 -0400 (55 minutes ago)
|           starting part2 - Daniel Sussman
* 82019cb - Wed, 13 Sep 2017 09:29:27 -0400 (3 hours ago)
|           got rid of an accidental swap file - Daniel Sussman
```

...so collaborate nicely