Version Control with Git (and GitHub)

Who am I?

- Final-year Computer Science
- Tech Officer for TermiSoc
- Also go to Hack Days, Conferences, etc.

An Outline

- Some Basic Concepts and Why?
- Staging Files
- Committing
- History
- Remotes & GitHub
- Branching & Merging
- Some tips.
- And, finally, some resources.

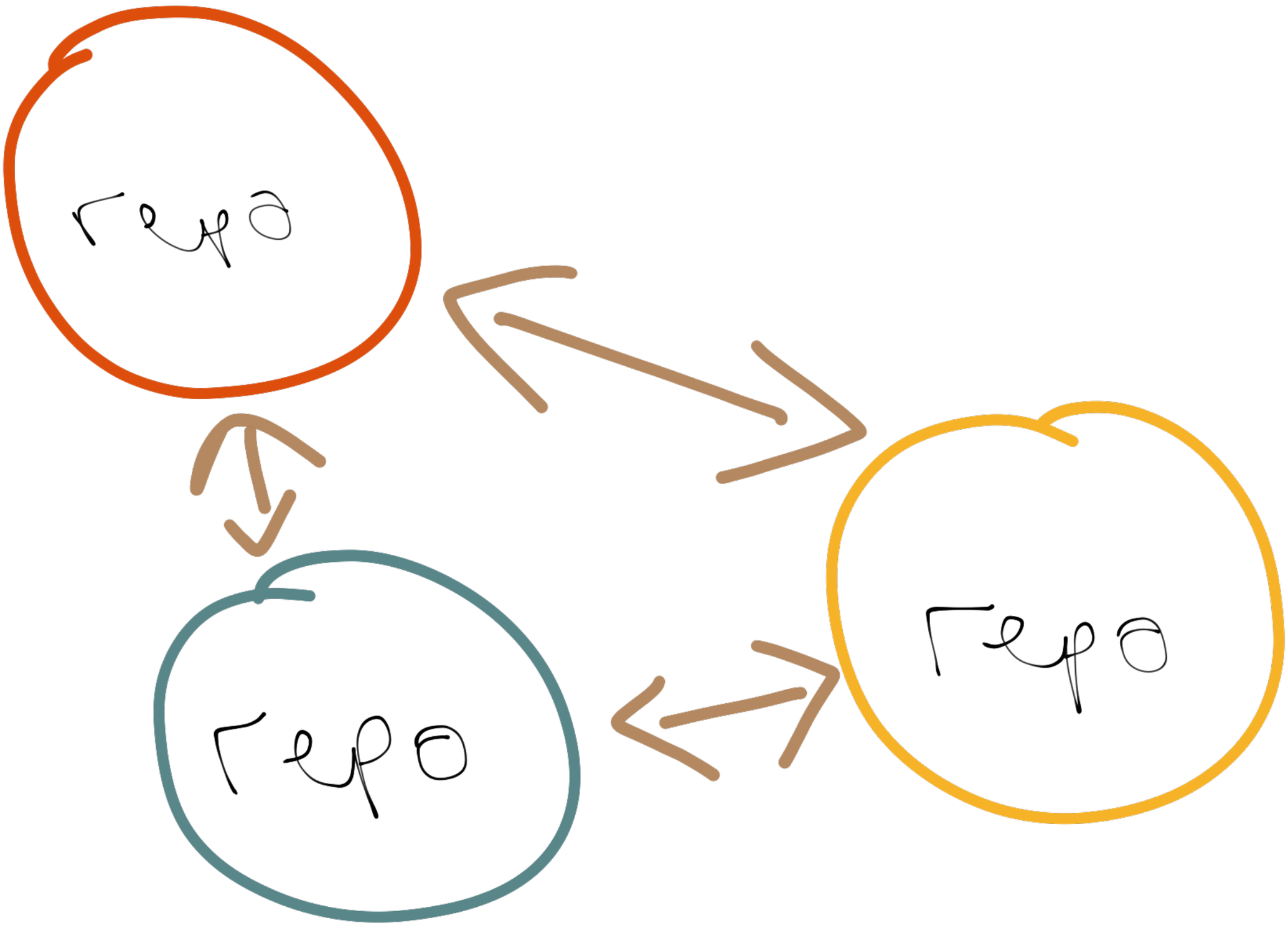
An Overview and Why bother?

- Store everything
- Go back in time.
- Makes it sensible to collaborate with others.

FORGET ANYTHING ELSE YOU KNOW.

Concepts

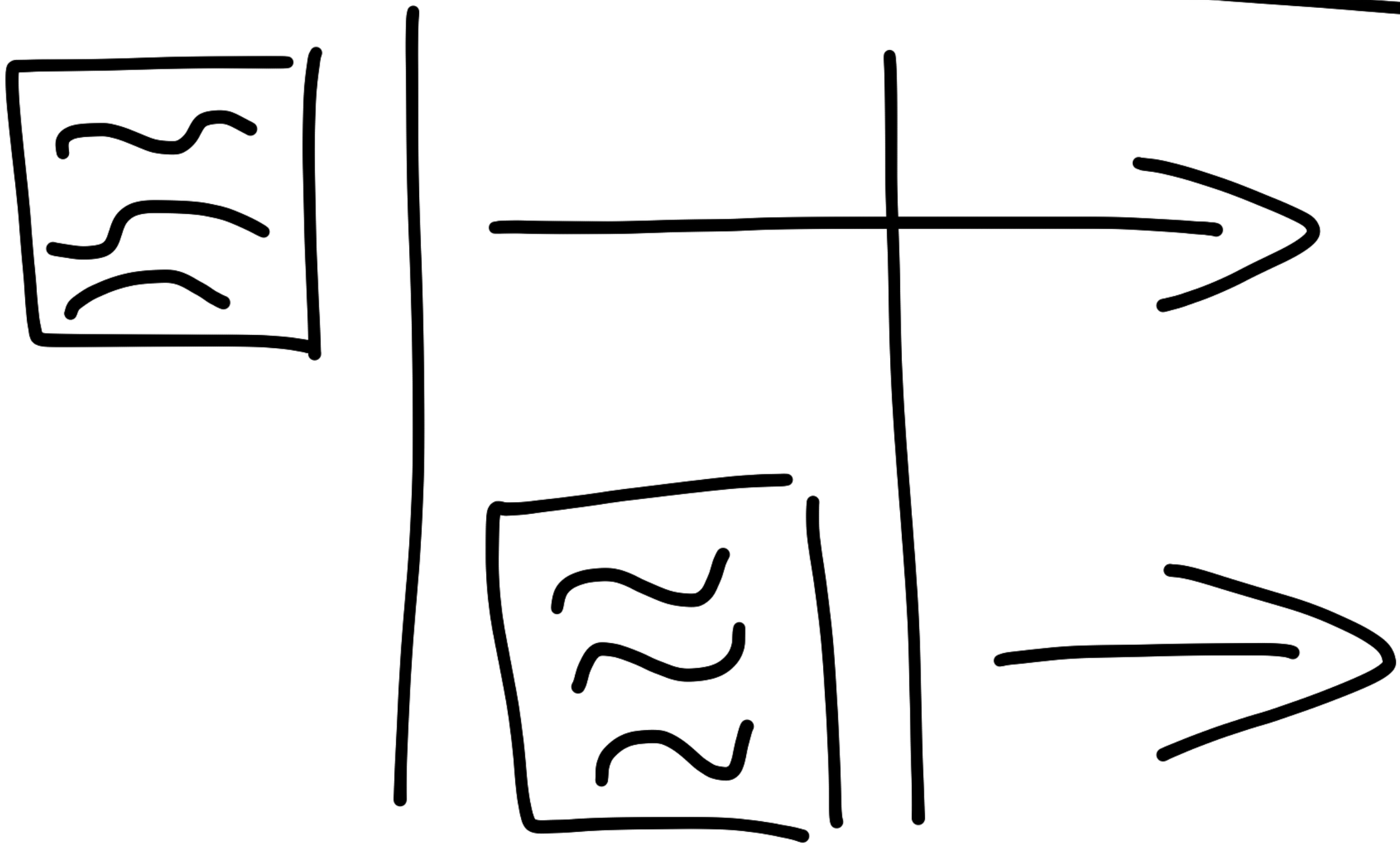
- Git is distributed.
- This keeps everything super fast.
- And means you can work on a train without going nuts.
- Git stores snapshots. Which keeps everything small.
- And it's very hard to lose data once it's in Git. It's there somewhere.


repo
repo
repo

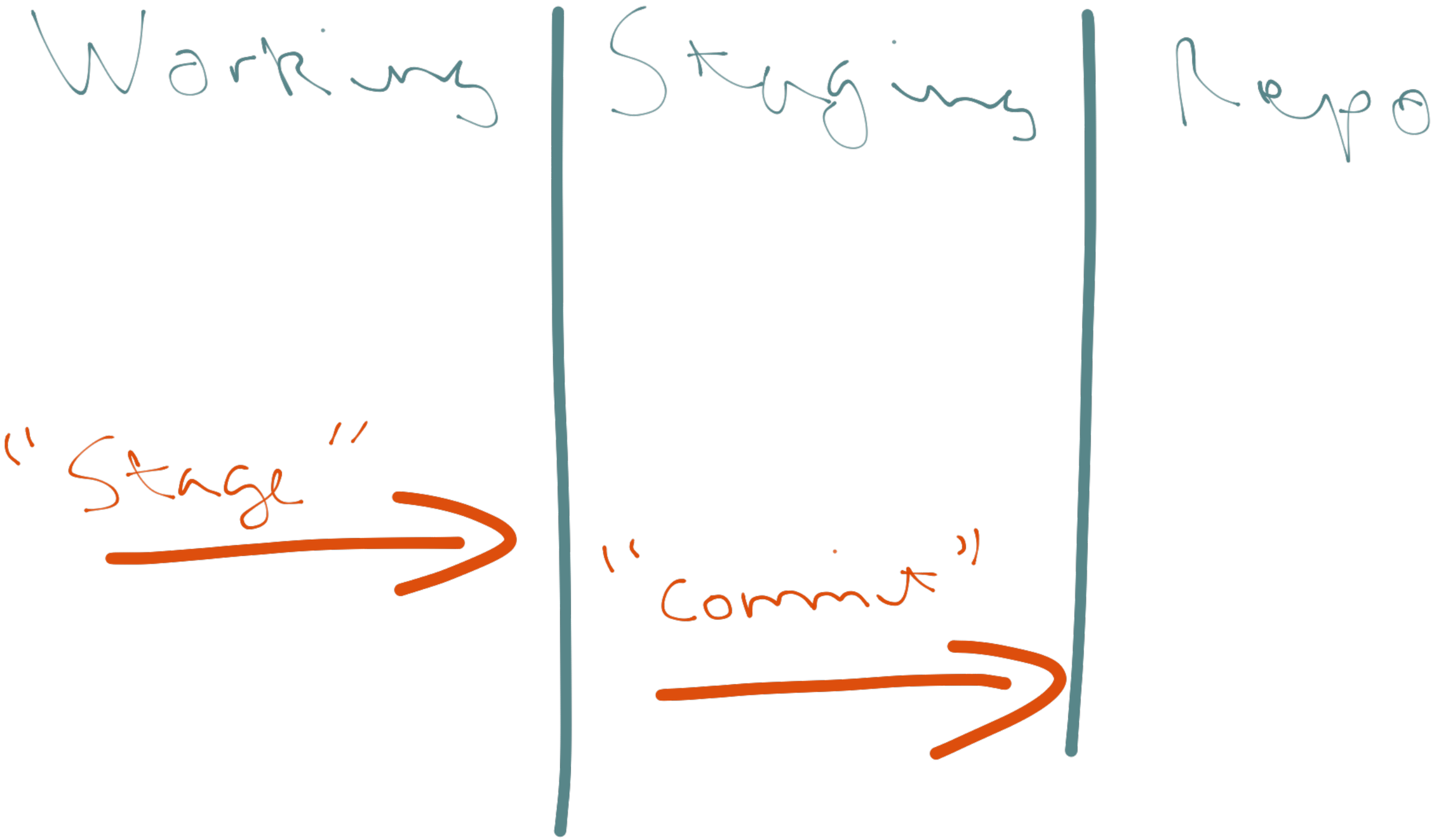
Concepts

- Git is distributed.
- This keeps everything super fast.
- And means you can work on a train without going nuts.
- Git stores snapshots. Which keeps everything small.
- And it's very hard to lose data once it's in Git. It's there somewhere.

6u79B 745JI 9CJ911



Staging

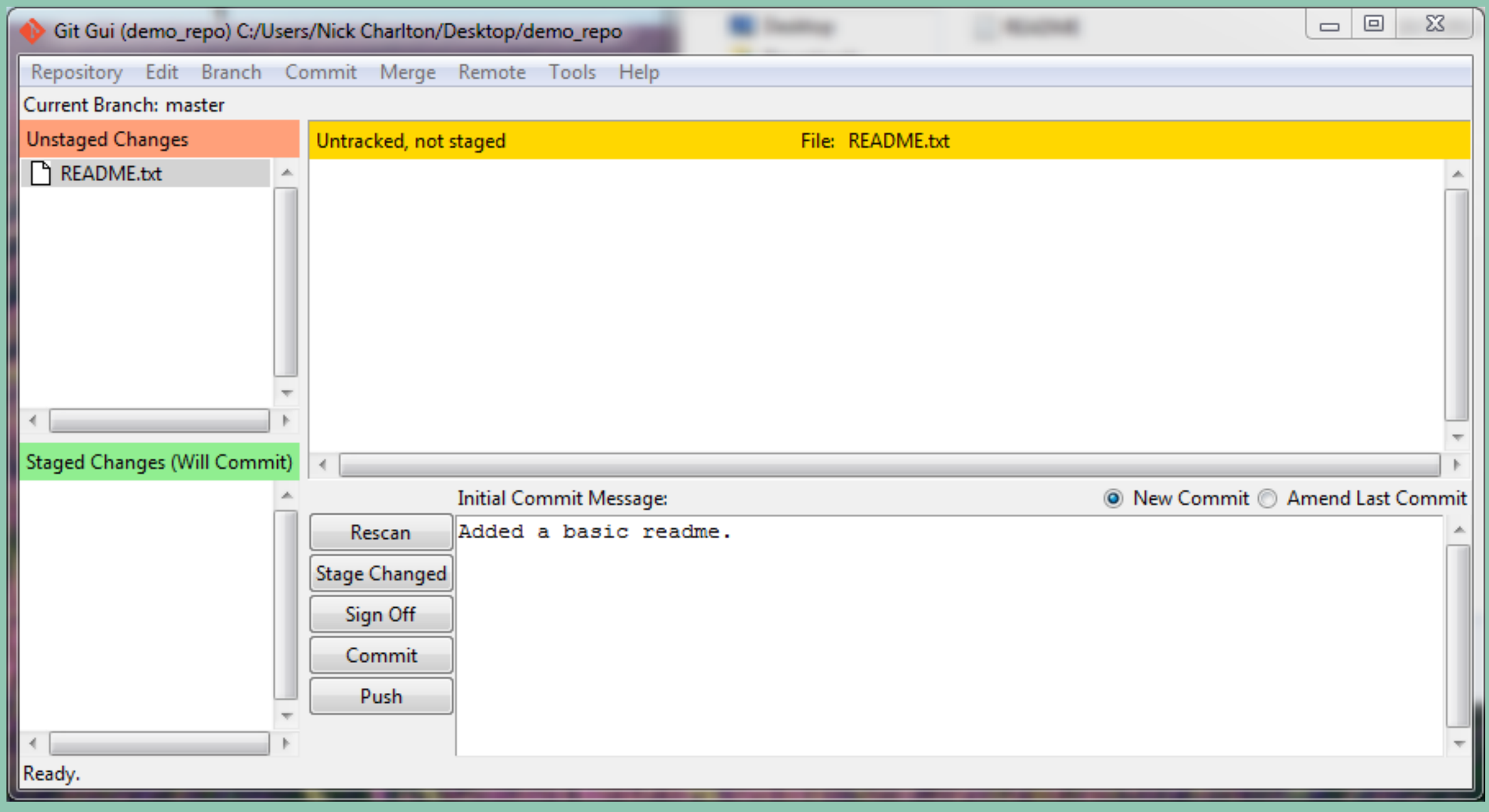


Concepts

- Git is distributed.
- This keeps everything super fast.
- And means you can work on a train without going nuts.
- Git stores snapshots. Which keeps everything small.
- And it's very hard to lose data once it's in Git. It's there somewhere.

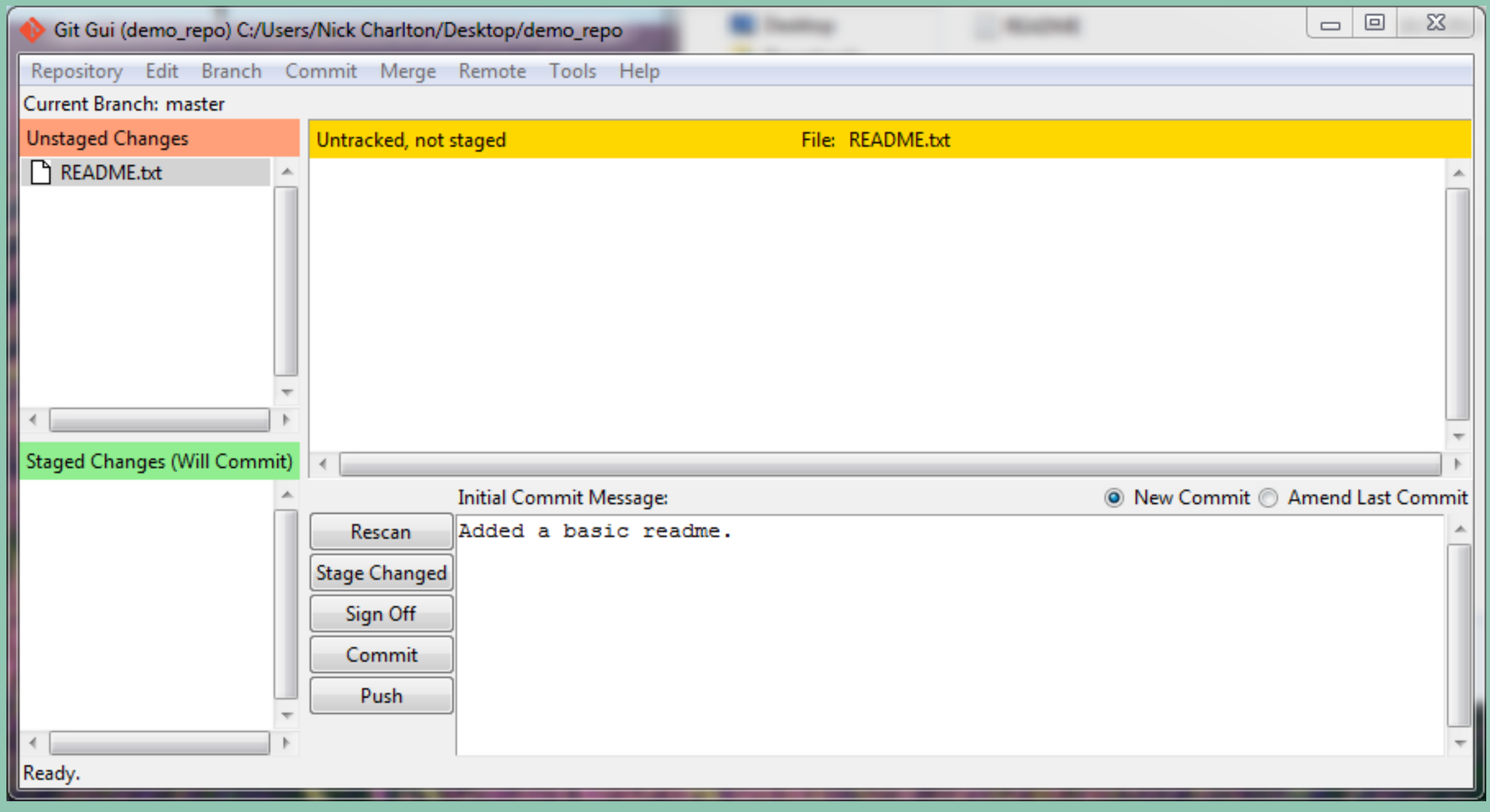
Committing

```
$ git add README.txt
$ git commit –m ‘Message’
```



History

```
$ git log
```

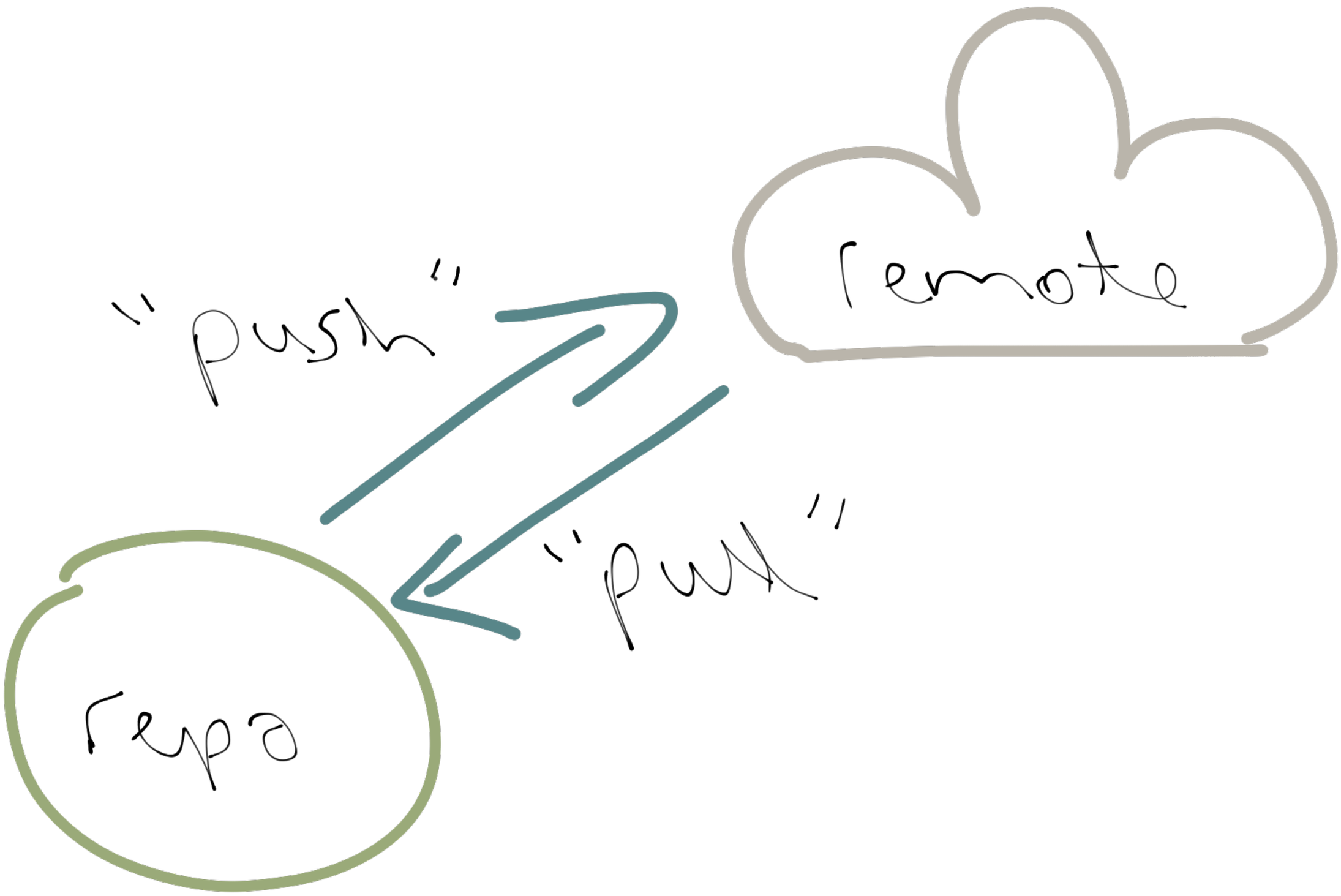


Give it a go.

1. Create a repository
2. Add a file.
3. Commit it.

Remotes and GitHub

- A URL pointing a repository somewhere else.
- Often, this is GitHub, or BitBucket
- But, it can be just another machine somewhere.
- You “pull” and “push” your changes.
- And, is the key bit in collaborating with others.

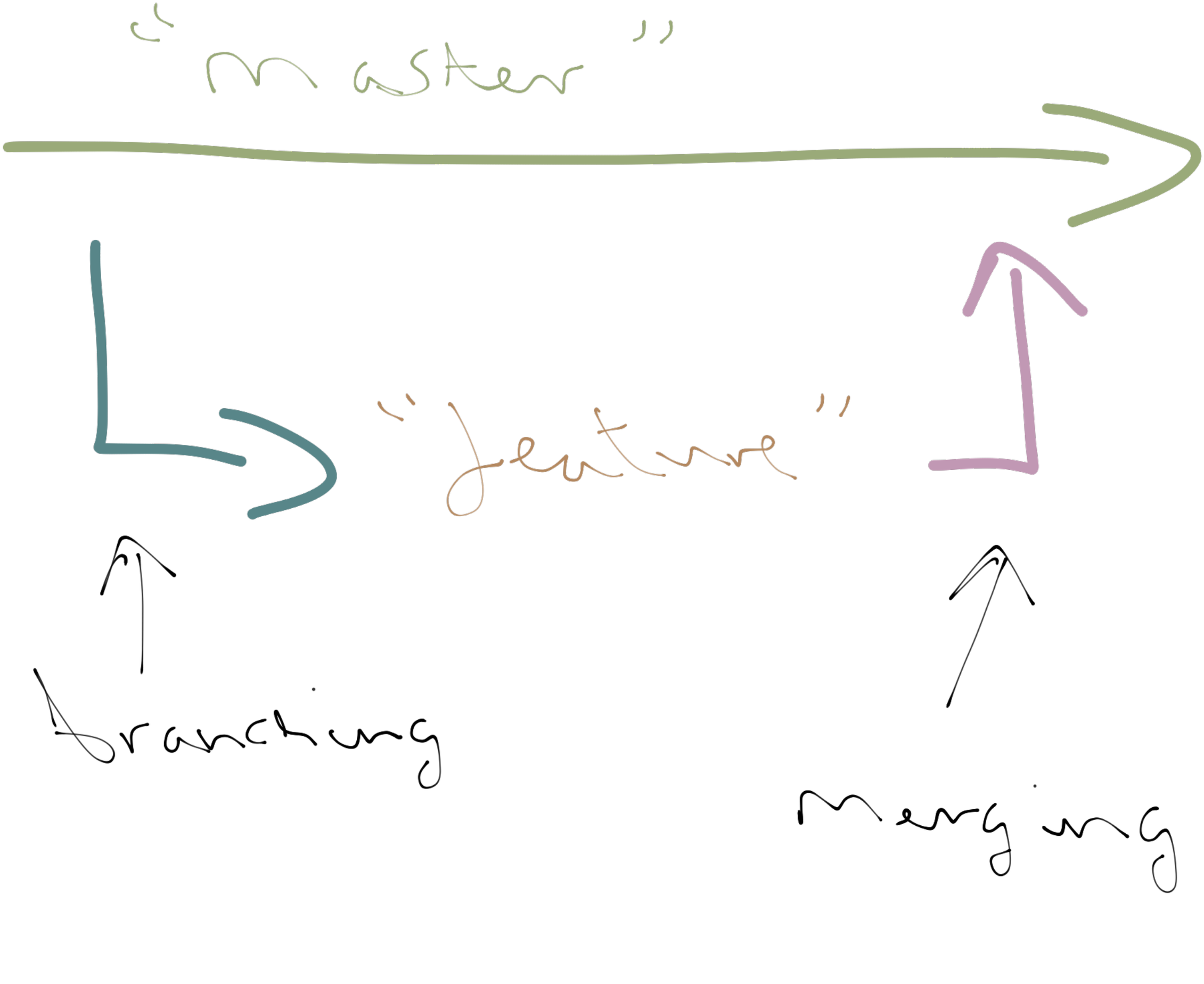

"push"
remote
"pull"
repo

Give it a go.

1. Create a repository on GitHub.
2. Add the remote to your local repository.
3. Push it up.

And so...

Branching and Merging


"master"
"feature"
branching
merging

Nick's Tips

- Use feature branches.
- Commit often.
- Stick with one commit message style.
- Use Git for everything.
- Push to a remote often.

Resources

- All of this: http://github.com/nickcharlton/git-workshop
- Pro Git Book (the best, if you want to know it all): http://progit.org/
- Git Reference: http://gitref.org/
- Free GitHub: https://github.com/edu

Thanks.

Questions?

Email me: nick.charlton@students.plymouth.ac.uk

(but make sure COMP206 is in the subject.)

- I'm @nickcharlton on Twitter.
- Also: http://github.com/nickcharlton
- And I'll link to stuff from: http://nickcharlton.net/

Thanks.