OS@I: Git Gud!

It's time to **git gud**.

opensourceatillinois.com

Intros!

What's your go-to restaurant?

opensourceatillinois.com/about

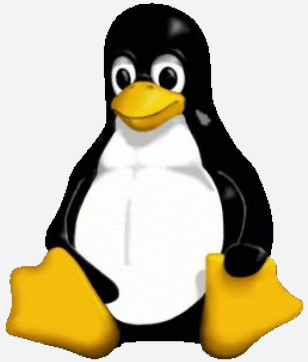
Why do we need version control?

Some ways:

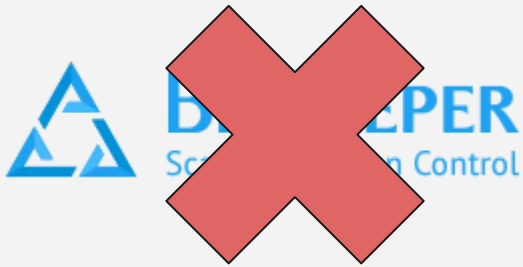
- Send your team updated code files every time you make a change
- Code live on a single file (Like Google Docs or Live Share on VSCode)
- Rage quit

Raise your hand if you know your way around git

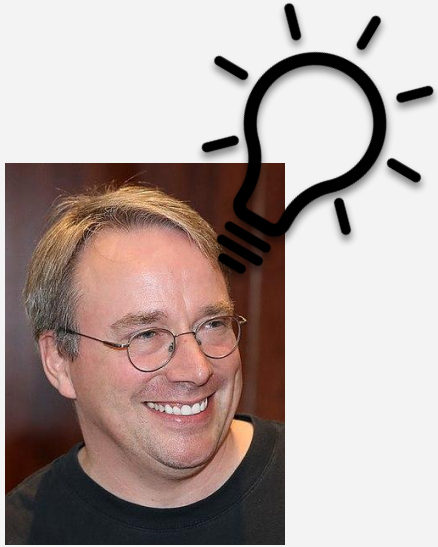
Creative Destruction & Fiery Controversy



Linux



Bitkeeper

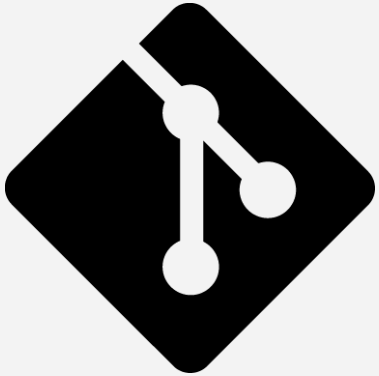


Linus Torvalds

git

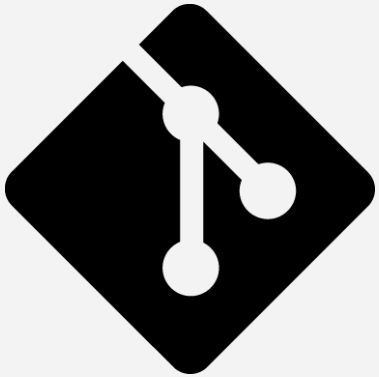
- Version control software
- Initially released in 2005

- Source code at https://github.com/git/git
- Currently at **71,304 commits**!
- **26k forks** - maintainers!



git

Version Control Software



GitHub

Software as a service (SaaS)

How does git know who you are?

git config —global user.name "Your Name"

git config —global user.email "youremail@example.com"

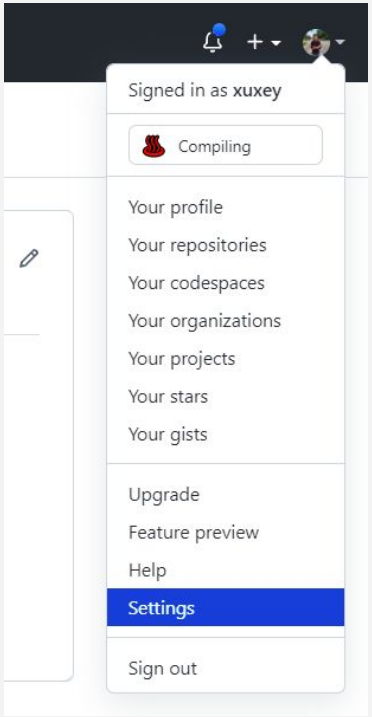
Add your ssh-key to github

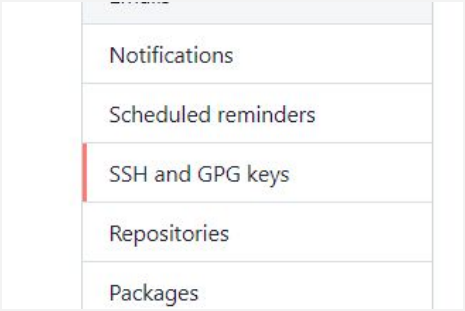
```
ssh-keygen
# Press ENTER for all options
cd ~/.ssh
cat id_rsa.pub
# Copy the key displayed
```

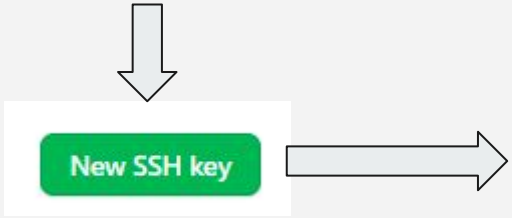
```
ssh-keygen
# Press ENTER for all options
cd %HOMEPATH%/.ssh
type id_rsa.pub
# Copy the key displayed
```

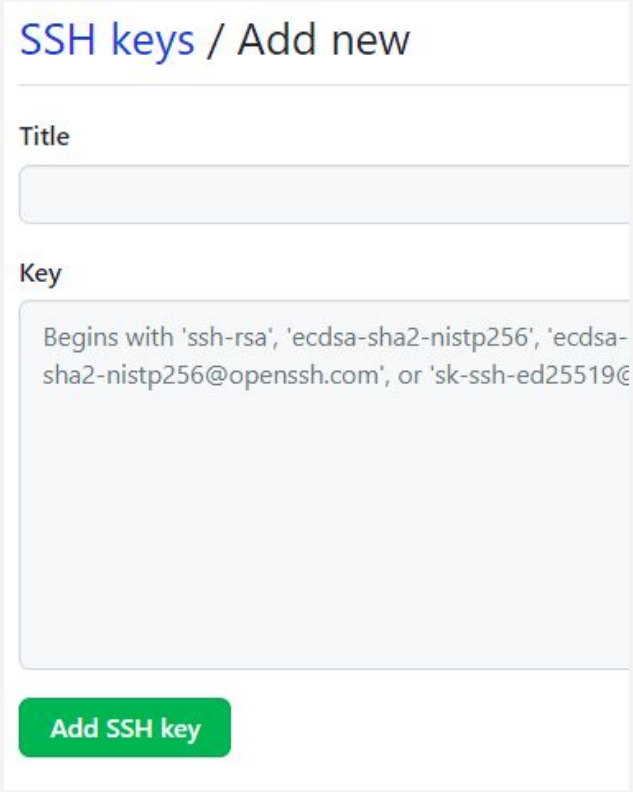
```
C:\Users\soham\.ssh>type id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDUg2OIPxwweBUL6teusCed8hZklSGGrxUgLHBFaG6Vv9vX4pmewwGH4+DfXVH
S2UWbnFd55Vfslp92oFfzP4QgiMK2q6+8Tipkv92lRD1ASeQ1810aSkIRddFiAXLuNfH5ahnwqRgGFe9Be8+WARCFUUwPYRUs4g
fbZTxkprWIBins5Y8+XgWuBEa1e8gL3x1VhDbFCDick+F6axEBIpPOq4JWTQ4LVJpYHeSoKluI8enWcpVxCESXjeqOGfhWZPcb5
DIdmKm+V1jJ3XNIqI4ww+FGtiRdIxdSBjnkVPFFivuGVqXktpePtQKdEVbepahYE+6TFGslHd20jStXsCcEvsfOduusgJH3I3NL
hjjg41q4MNVNwwKSOokPJ1Z1CtM1WIEo4eSF4TVlBJEWjDqiRtl3C2991tawePG7gPvGwpk= soham@LAPTOP-O8KP8F4K
```

Add your ssh-key to github









Keywords

So we sound like we know what we're talking about

An analogy

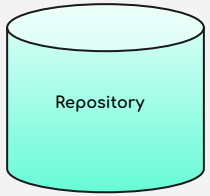
Repositories

- Think of your repository as an address where you can mail code from/to
- git facilitates communication between different addresses

Typically

- Github hosts a remote version of your repository
- Your development machine holds a local version of the repository
- git allows changes to reflect from one unto another

-




```
git
init
```

An analogy

While sending changes, we have a 3 step process

- `git add`: think of this as putting everything you want to send to your address in a box. As long as the box is open, you can add and remove things

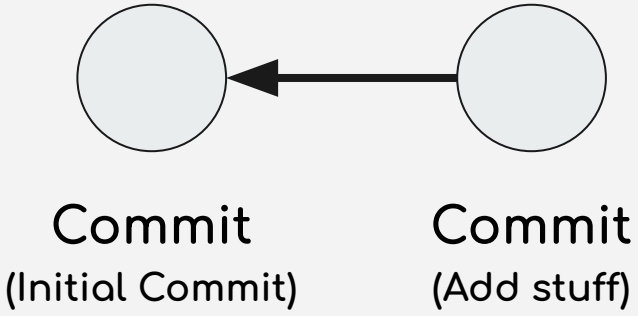
- `git commit`: this is like sealing the box. Once you close it, you can't change the contents. While sealing it, you can put a note in explaining its contents

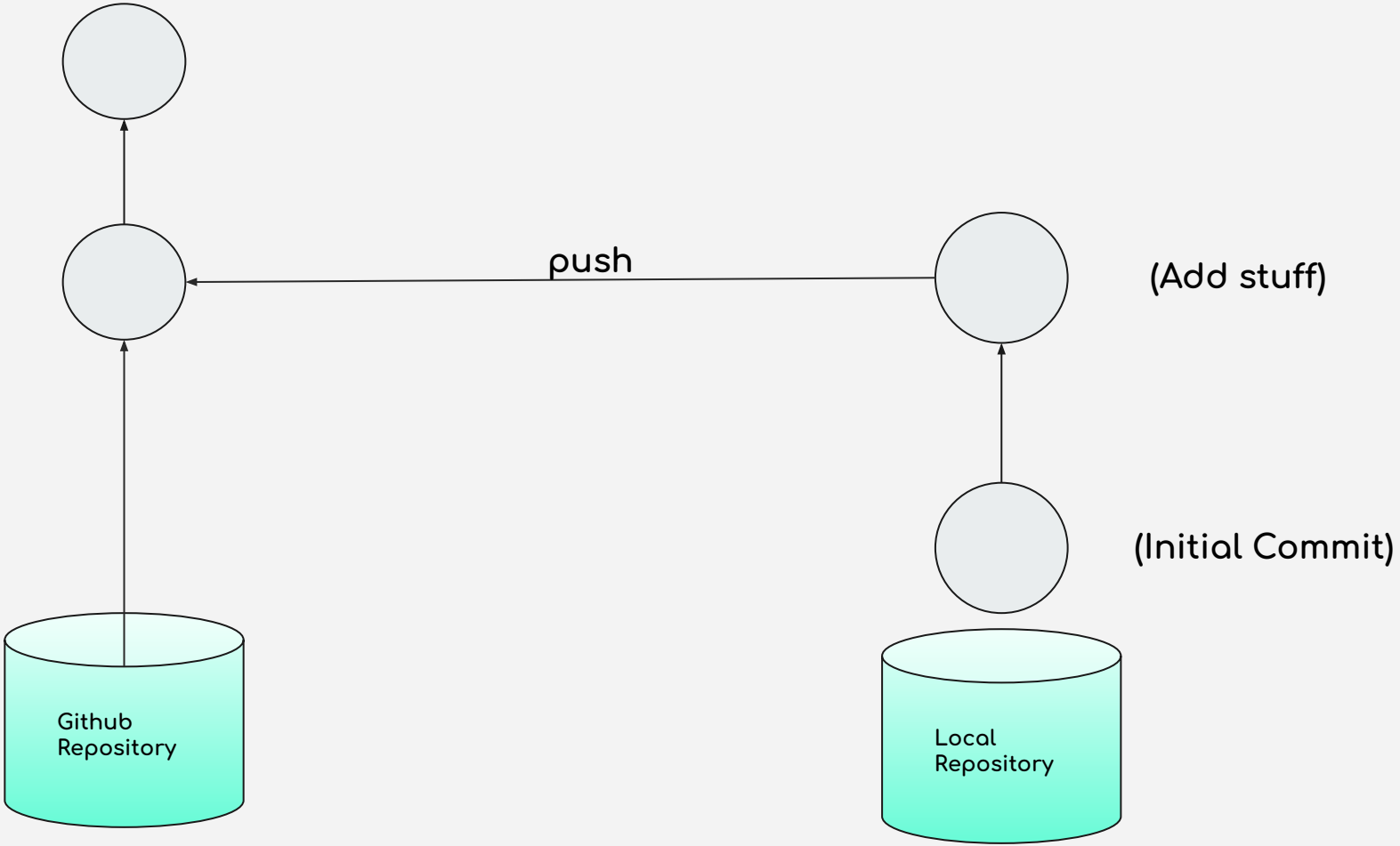
- `git push:` This sends the box to the specified address -


git commit
AIRBUS A350-1000
A350
1000
git push
AIRBUS A350-1000
1000
A350
git add .

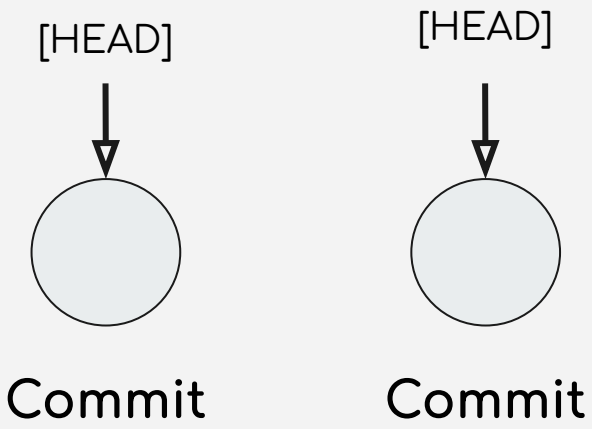
commits

```
git commit -m "Initial Commit"

git commit -m "Add stuff"
```




push
(Add stuff)
(Initial Commit)
Github
Repository
Local
Repository

HEAD



What about secrets?

.gitignore

Staging

Because sometimes you don’t want to commit embarrassing code

```
git add -A
```

Alternatively,

```
git add file1.txt file2
```

Push, Pull, fetch, and merge

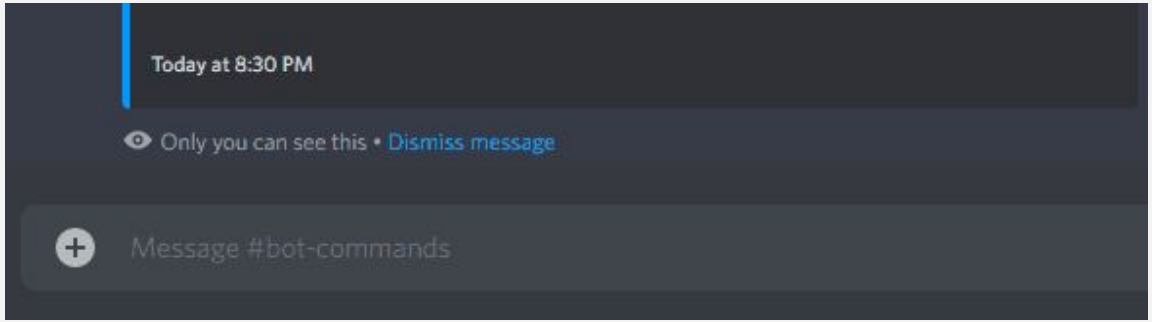
```
git push

git push origin main


git pull

git pull origin main
```

*Pull = Fetch + Merge

Event Code: GITGUD23

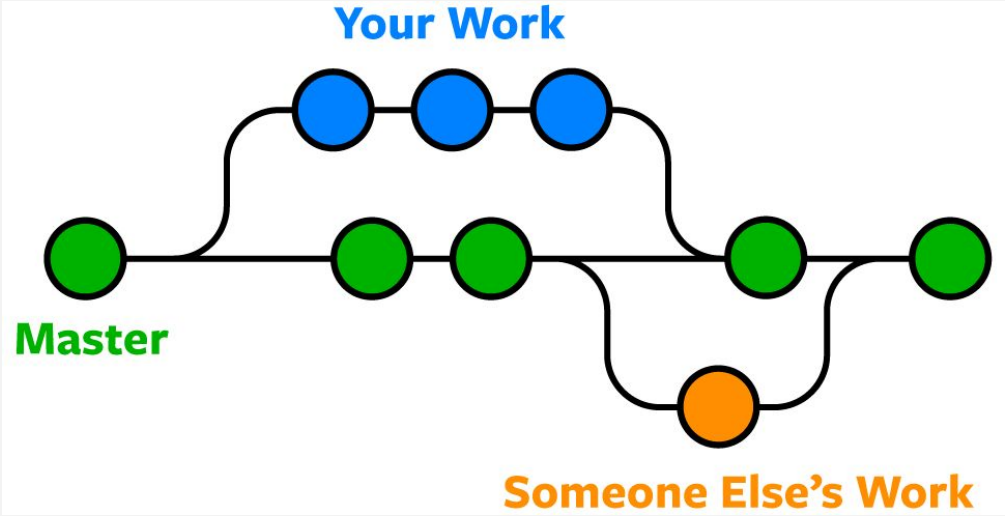
Use /attend GITGUD23 on Discord

[!] On mobile: Make sure to select osai-bot after you type /attend

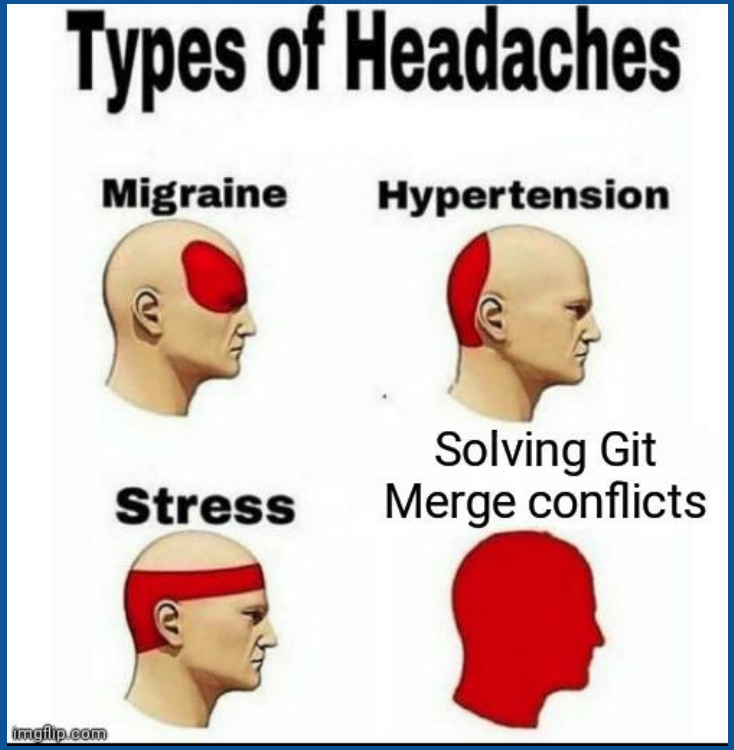


branches?

branches?

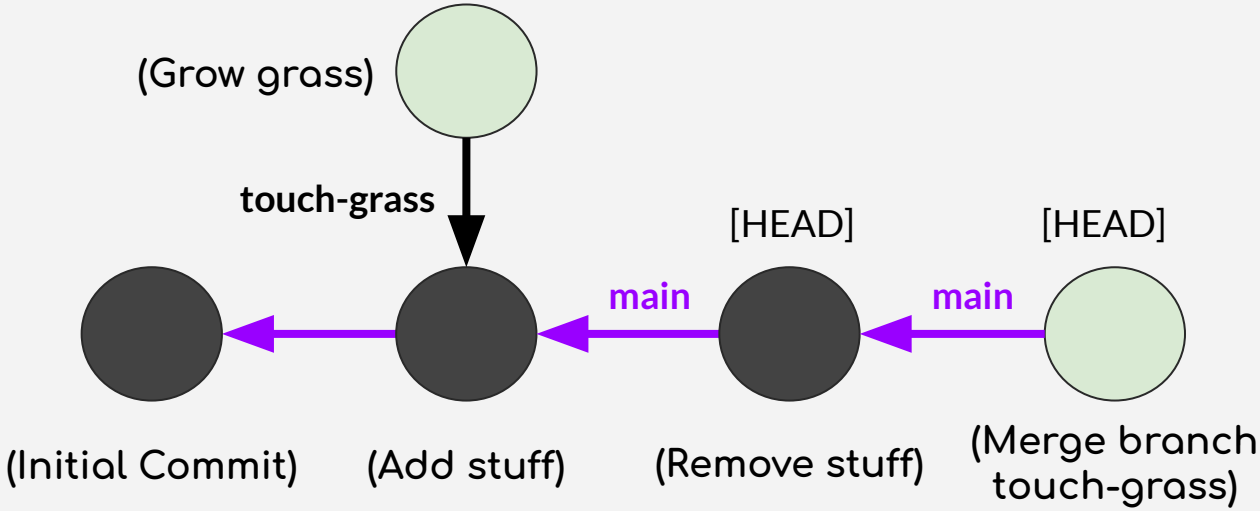


Merge Conflicts



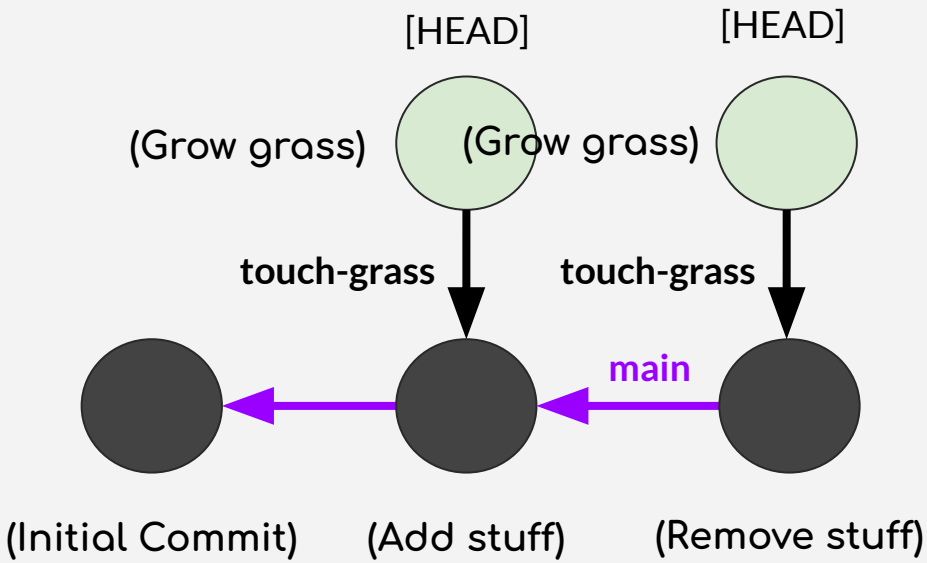
Merging

```
git checkout main

git merge touch-grass
```



Rebasing

```
git checkout touch-grass

git rebase main
```



When should you rebase over merging?

Rebase is preferred over merge in two scenarios:

- When you know **exactly** what you are doing
- See above

Visualizing this mess

- https://learngitbranching.js.org/?locale=en_US

Conflicts

- Happen when two branches change the same lines
- Contrary to popular belief, this is a good thing

Conflict markers (Everything in red)

```
<<<<<<< HEAD
Content that exists in the current
branch, where [HEAD] is pointing
=======
Content that is present in the merging
branch
>>>>>>> touch_grass
```

How do I fix this?

1. Run `git status` to see all conflicting files.
2. Open each file in your favorite editor (vim, nano, VSCode, whatever)
3. Remove conflict markers, and keep code that you want (and remove code you don't want)
4. Save, stage & commit!

Resources

- https://github.com/open-source-at-illinois/git-gud/
- https://learngitbranching.js.org/
-

Noughts and Crosses

Tables pair off to 1v1 in Noughts and Crosses, but with a twist - you can revert twice!

Conditions:

- Reverts can only be used if no one has one
- You can revert only to **the other team's previous commit**
- Turns alternate, essentially, your turn to play, if the head was committed by the other team

Thank you!

We hope to see you soon!