CODE CLUB

UCL INSTITUTE OF HEALTH INFORMATICS

Introduction to Git and GitHub

WORKSHOP S01E01

25 JULY

JUST DO GIT.

git GitHub

"DON'T LET YOUR DREAMS BE DREAMS"

Learn git today!

Content & Aims

- What is Git
- Why use Git
- How to use Git [interactive]
  - create a new repository
  - edit files -> add/commit
  - branches

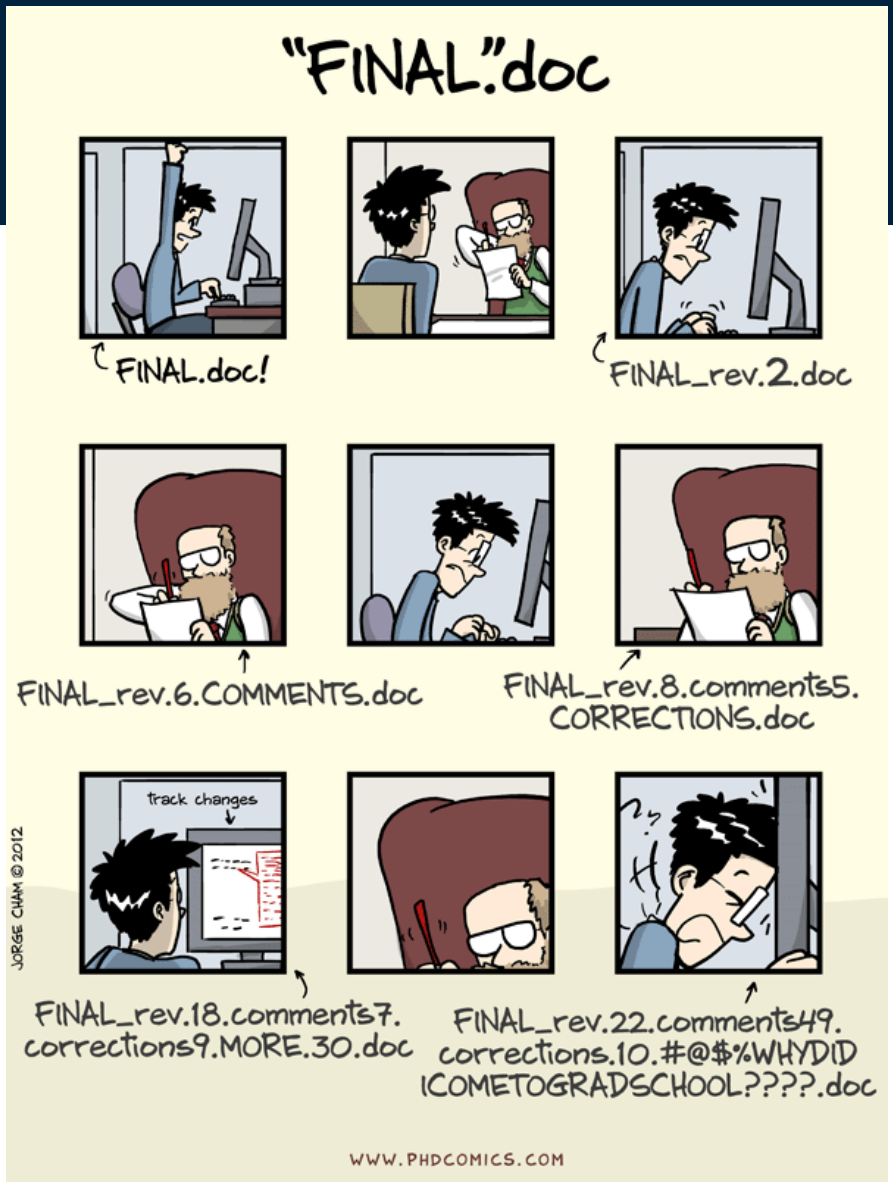
- AIM: be able to create a local Git repo and use basic version control features

- What is GitHub
- Why use GitHub
- How to use GitHub [interactive]
  - fork / clone repo
  - collaboration -> pull requests
  - GitHub Issues

- AIM: be able to push and pull between local and remote repo

git

GitHub


"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

What is Git?

Version Control system – tracks changes to source code over time
means you can undo changes, work collaboratively etc.

Git is distributed version control (started in 2005)
your full code history is kept on your local machine
so you can make changes without internet access

Can store your code in a remote instance of the VCS – a back up
(need internet access to) push and pull between local and remote

There are other version control systems:

Why should we use Git?

Individual development

track all changes to your files
undo or go back to previous versions
better organisation
avoids you having multiple-versions of the same file
keeps all your work in one place, e.g. not scattered over emails

Offline usage

Everything is local – all your code and history available offline

Collaborative development

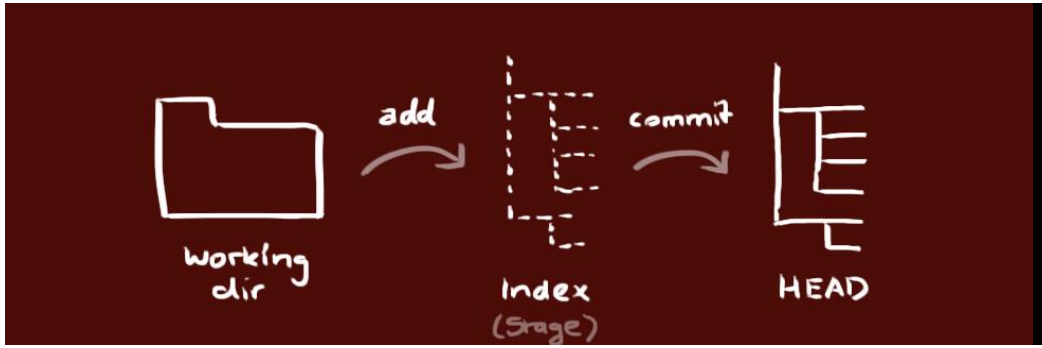
work with others
save your work on the cloud
more on this later…

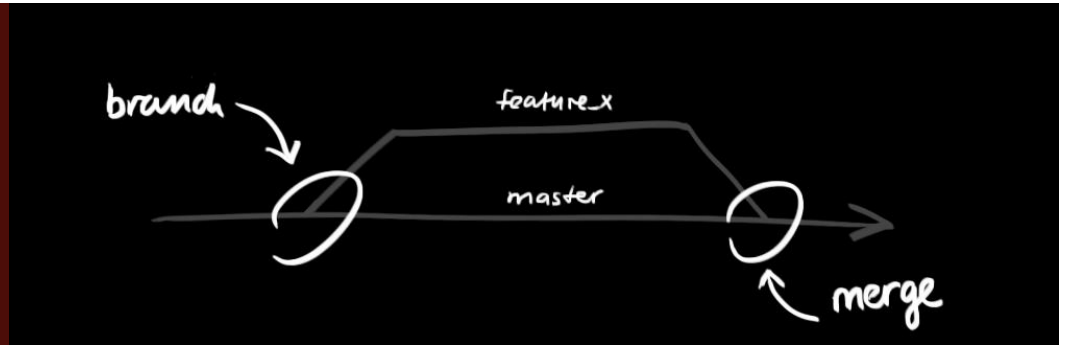
Key Concepts

Repository

Add & Commit changes

Branches





Repository a.k.a Repo

your local repository consists of three "trees" maintained by git. the first one is your `Working Directory` which holds the actual files. the second one is the `Index` which acts as a staging area and finally the `HEAD` which points to the last commit you've made.

A repo is a collection of all your files and their history

Modify files here

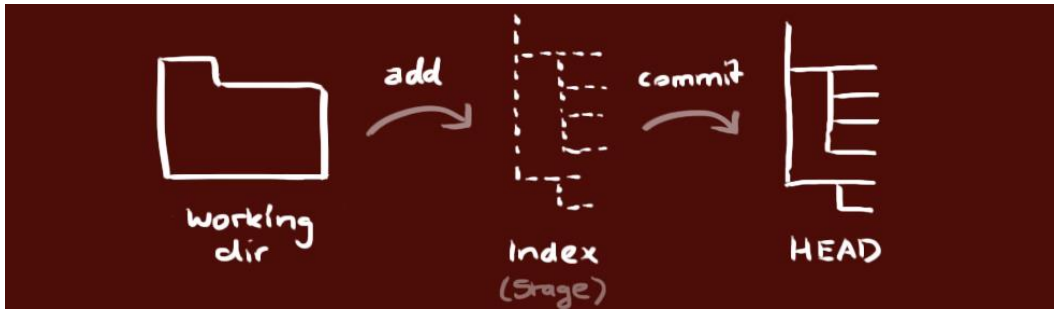
Stage the files
- add snapshots of them to the staging area

Do a commit
to store snapshots permanently to your Git directory / repository

https://rogerdudler.github.io/git-guide/

Add & Commit

A commit contains info on how your files have changed from the previous version



Modify files

Add new files to staging area

```
git add <filename>
git add*
```

Commit changes

```
git commit –m "Commit Message"
```

https://xkcd.com/1296/

NB

'commit' is both a noun and verb

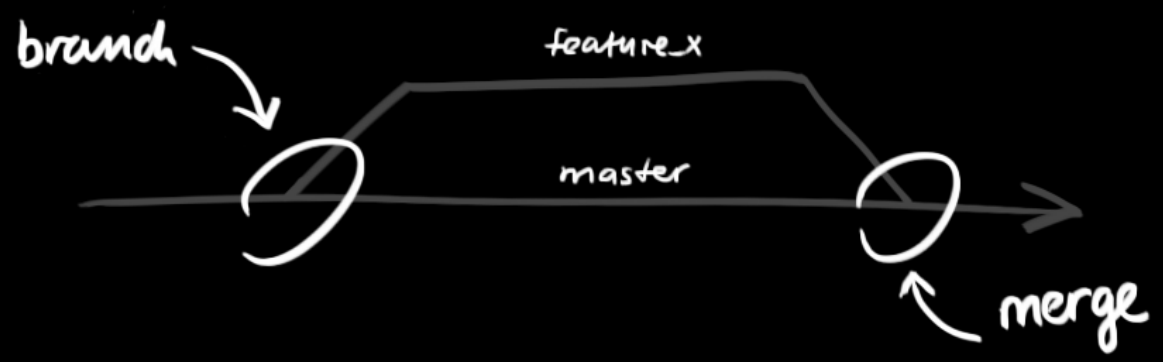
Commit messages should be a short & clear description of changes you've made

Use imperative/command tense *e.g. Add tests, Fix function_name, Neaten code*

Don't be afraid to commit! Commit often

Branches & Merges

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.



all commits live on a branch

your main branch is called 'master'

make a new branch

work on branch just as before – we can add and commit changes

can switch between branches, and work on the other branch

can merge branches – merges all your commits

https://rogerdudler.github.io/git-guide/

How to use Git

TRY THIS

#0 create GitHub account

#1 download and install Git **or use UCL PCs / Remote Desktop, which has Git installed**

Mac OS Windows Linux > open git bash / command line	
configure git with username, email & preferred text editor	*git config --global user.name "GitHub_username"* *git config --global user.email "GitHub_email_address"* *git config –global core.editor <editor e.g. vim, nano>*

#2 create a new repository

create & open a new directory	*cd ~/<folder e.g. Desktop>* *mkdir <directory>* *cd <directory>*
create & initialise a new git repo	*git init*
check project status	*git status*

#3 add & remove files **you can use the terminal or open a text editor to add e.g. a text file to your repo, whichever you feel comfortable with**

add & edit & save files	*touch <filename>* (creates a new empty file)
check status	*git status*
add changes to INDEX from a **specific** file in your repo	*git add <filename>*
check status	*git status*
remove/delete from INDEX check status again	*git rm --cached <filename>* *git status*
add all changes to INDEX from **all** files in your repo check status	*git add ** *git status*

How to use Git

TRY THIS

#4 commit changes

commit changes (to the HEAD of your local working copy)	*git commit -m "<message>"*
see differences in updated & original state of edited file	*git diff <filename>*
check status	*git status*

#5 create a branch

create and go to new branch	*git checkout –b <branch>*
explore the contents of this branch	*ls*
switch to master branch	*git checkout master*
delete branch	*git branch –d <branch>*

#6 merge branches

create and go to new branch	*git checkout –b <branch>*
make some changes e.g. add & edit a file	*e.g. echo '# README #' > README.md*
check status	*git status*
switch to master branch	*git checkout master*
view changes between the two branches before merging	*git diff master <branch>*
merge new branch into current branch	*git merge <branch>*
check status	*git status*
commit and push changes as in step #4	(repeat #4)

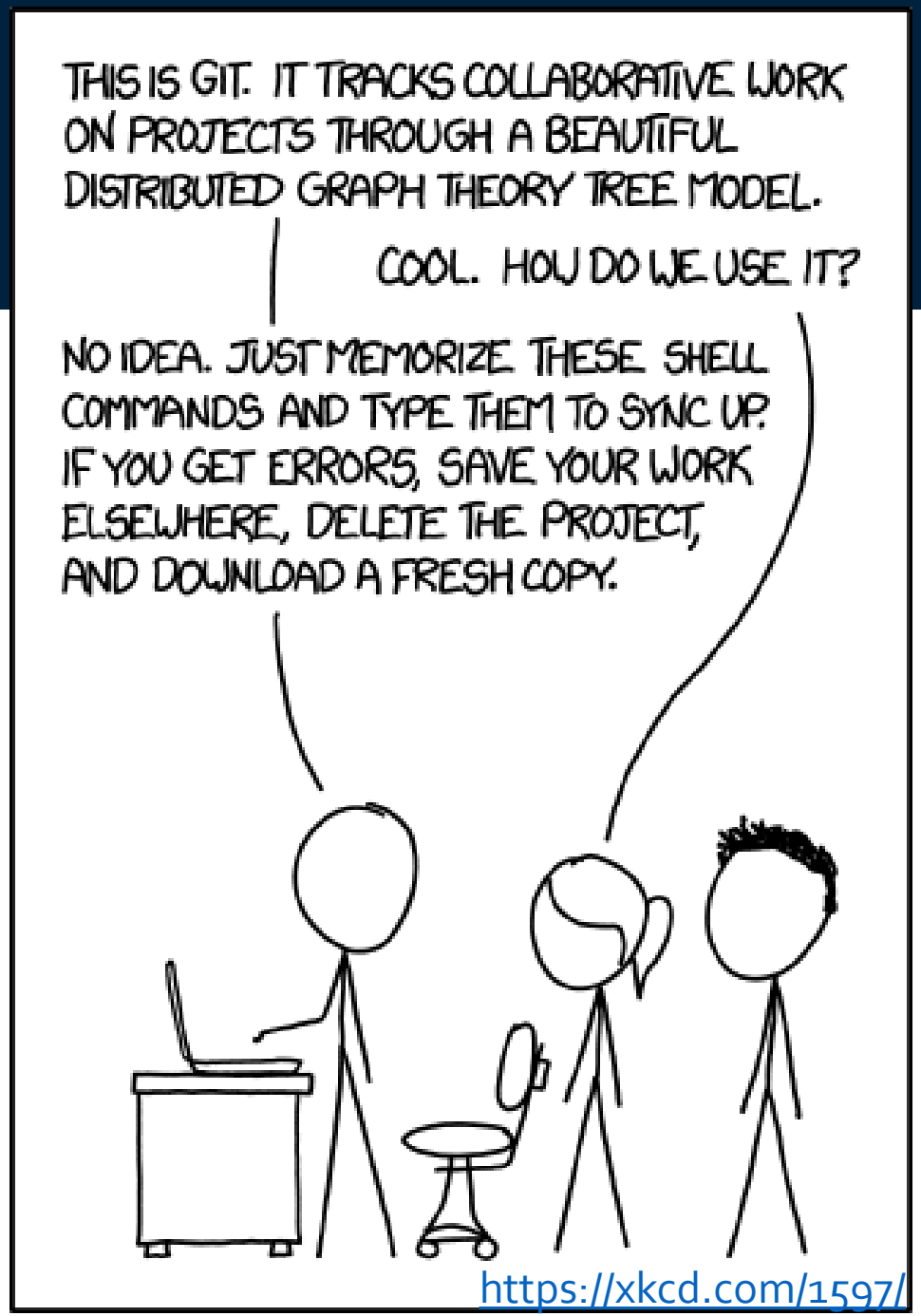
git log shows the history of all your commits

SUCCESS!

Next Steps

- Look at old versions
- Compare various versions
- Restore old versions
- Ignore specific files

- Share your changes with others or simply store your files online
  -> remote repositories



https://xkcd.com/1597/

What is GitHub?

Founded in 2008

Largest **web-based git repository hosting service**
-> hosts remote repositories

Collaborate with anyone online

Extra functionality on top of git
e.g. UI, documentation, bug tracking, feature requests, pull requests etc.

There are other popular options:

WHY SHOULD WE USE GITHUB?

..or GitLab or Bitbucket or [your cloud storage of choice]

Benefits of Git, plus:

- **Document**
  e.g. *README.md* files or *GitHub Wikis*
- **Share** your code
- **Collaborate** - work with others
  see who made what changes and when
  work on the same piece of code at the same time
  make Teams – a team repo
- **Access** to other people's code and projects
- **Showcase** your work
  be found on *Google*, build a portfolio, put it on your CV
- **Backup**
  your files are stored in a remote repo
  so you have a backup in case you lose the files on your local machine

GitHub Desktop

Key Concepts

Public & Private Repositories

Local & Remote Repositories

Push & Pull changes

Public & Private Repos

Overview | **Repositories** 14 | Projects 0 | Stars 1 | Followers 1 | Following 1

Test

Updated on 7 Feb

Star

ml-class

Forked from lukas/ml-class

Materials for class on machine learning/deep learning using scikit-learn, tensorflow and keras

Jupyter Notebook 593 GNU General Public License v2.0 Updated on 15 Nov 2018

Star

Projects

Updated on 13 Jul 2018

Star

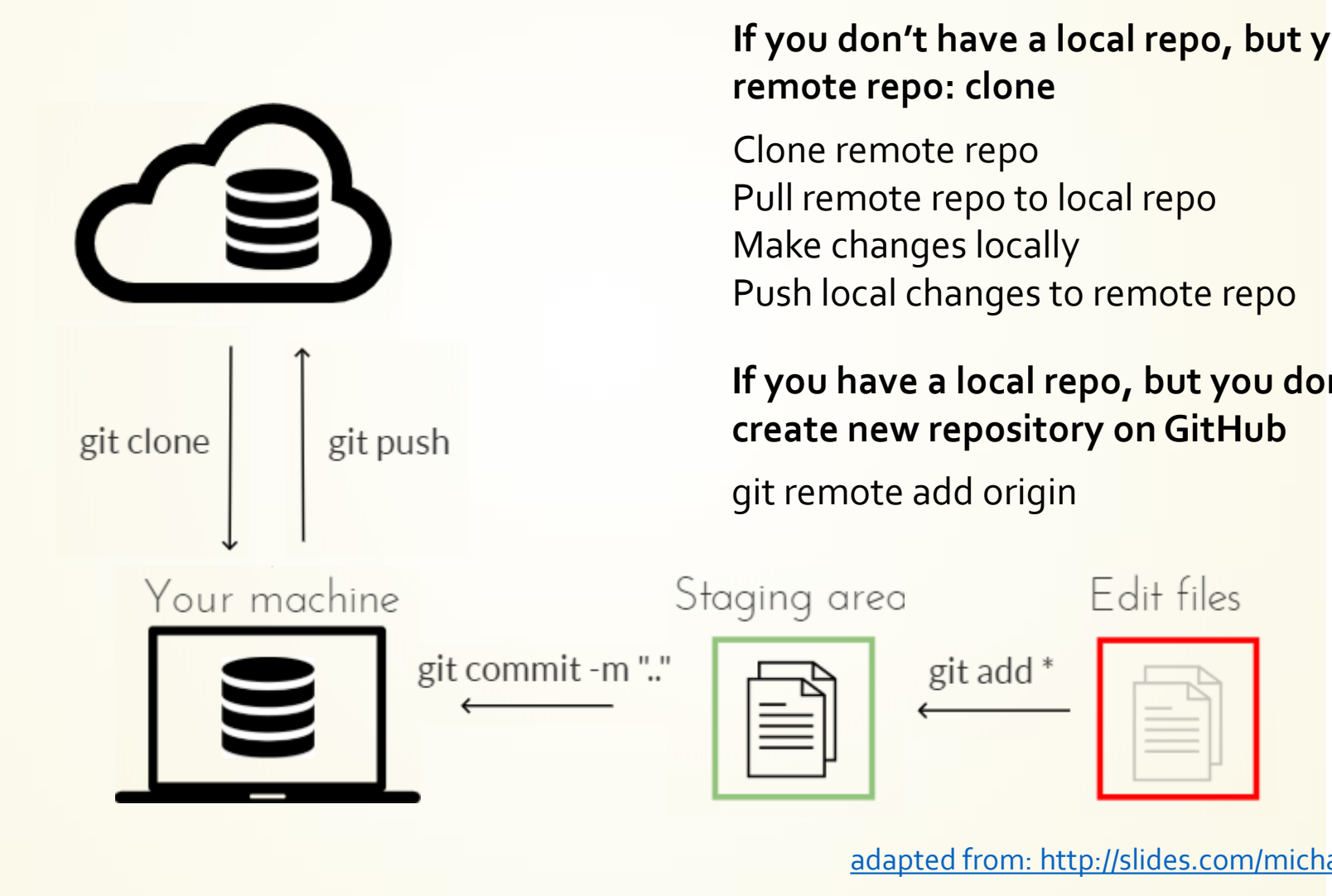
2017-18 Private

Python Updated on 23 Jan 2018

Star

Local & Remote Repos

Your remote repo



Your local repo

If you don't have a local repo, but you have an existing remote repo: clone

Clone remote repo
Pull remote repo to local repo
Make changes locally
Push local changes to remote repo

If you have a local repo, but you don't have a remote repo: create new repository on GitHub

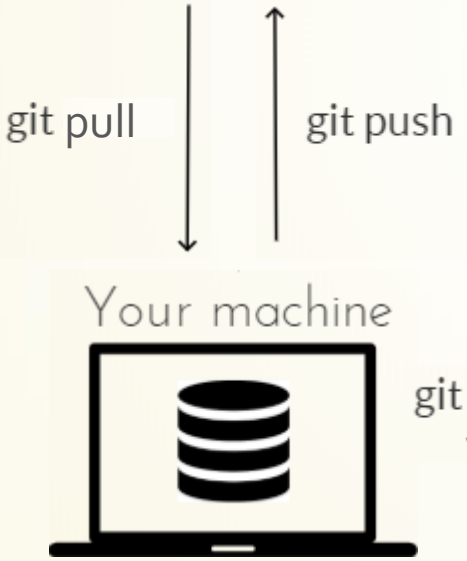
git remote add origin

adapted from: http://slides.com/michaelfreeman/git-collaboration#/

Push & Pull Changes

Your remote repo

changes from:
another collaborator
local repo on your other PC



Your local repo

when local and remote repo are synched
<u>pull before you push</u>
to update your local repo

adapted from: http://slides.com/michaelfreeman/git-collaboration#/

How to use GitHub

#1 create a new remote repository

naming convention: short, clear name, unique to your account.
Spaces will automatically be replaced with hyphens.
Same as making a new directory and initialising it in git
choose *Public* repo
[important] check 'initiate with README.md' if you didn't already create an README file in your local repo

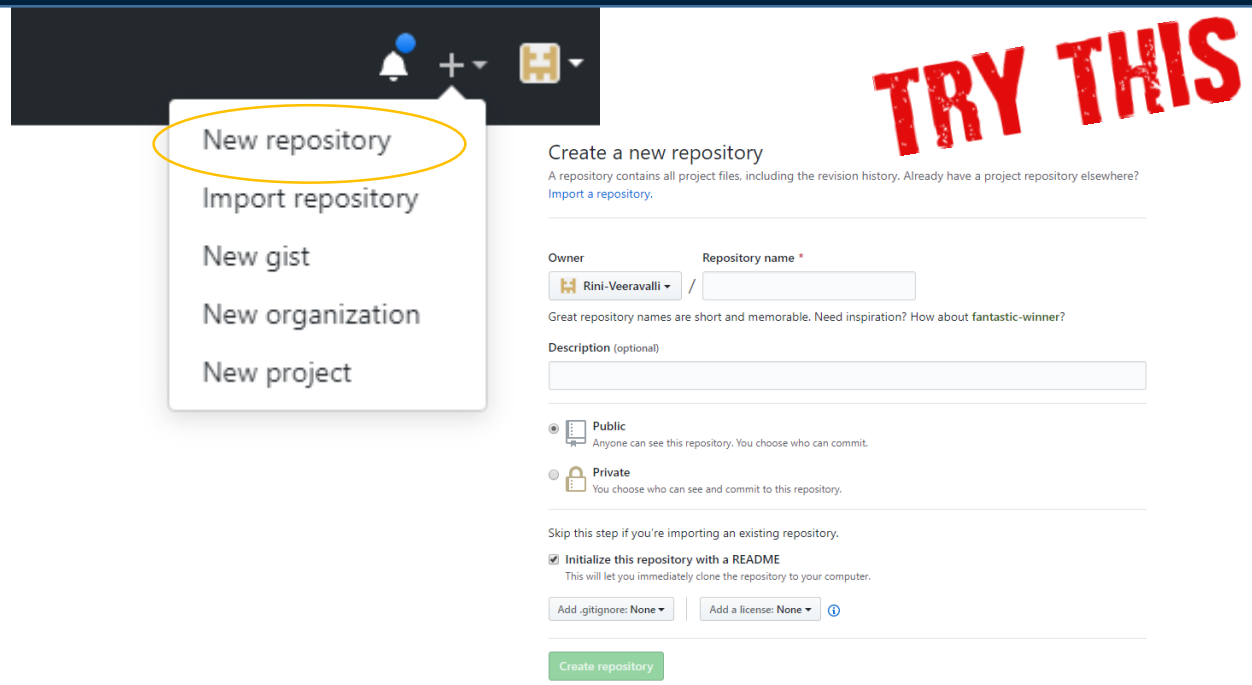
it's good practice to include a README file (text file written in markdown) to let others know what your project/repo does & how it works

#2 connect local repo to remote repo (to be able to push changes to a remote server)

from your local repo (made in git practical)
quick setup, click *HTTPS*, copy *<server>* & paste into bash
Origin = remote repo

#3 push changes from local repo to remote repo

When pushing – you'll be prompted to login to GitHub

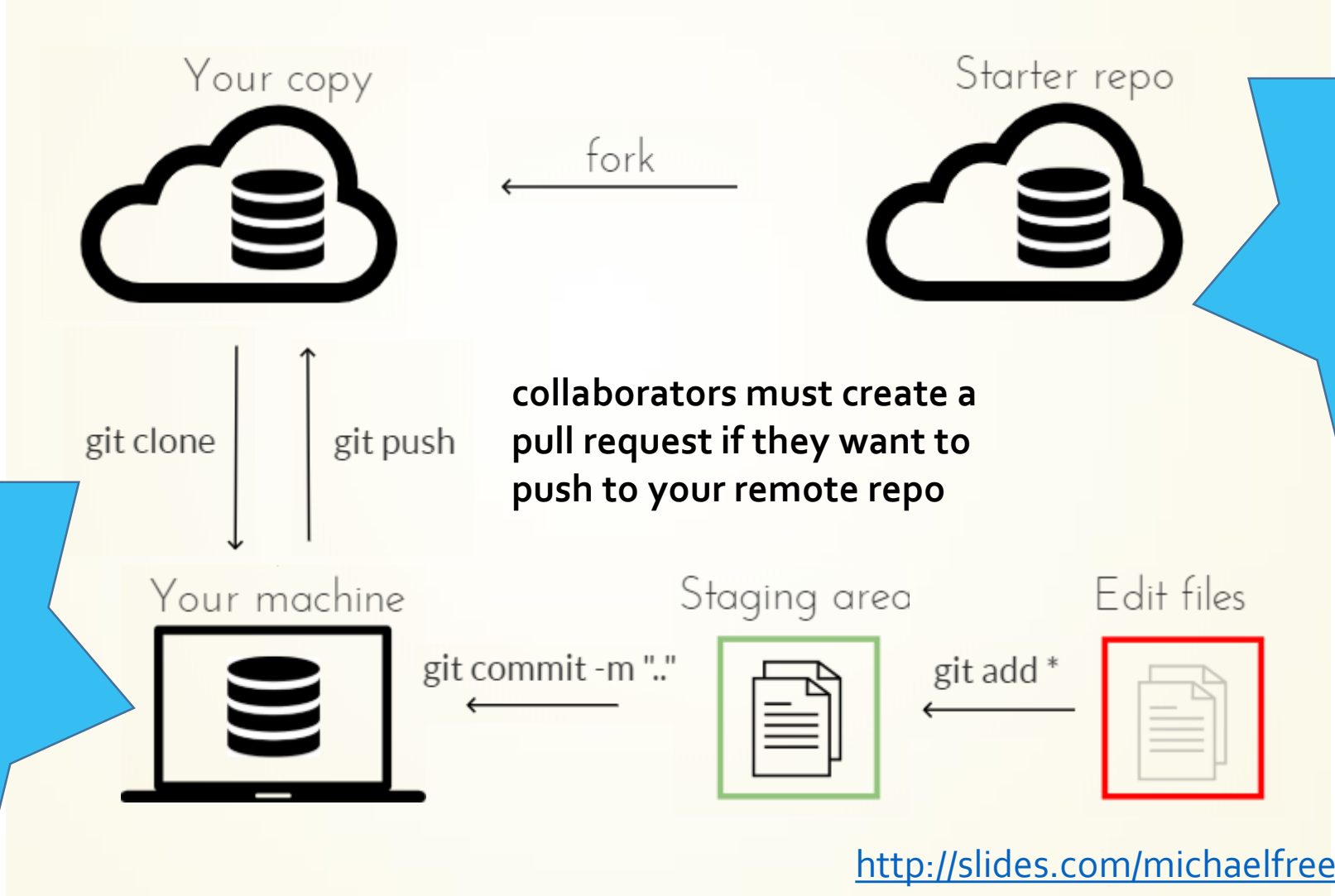


git remote add origin <server>

git push –u origin master
<GitHub username>
<GitHub password>

https://launchschool.com/books/git/read/github

Collaboration Workflow



Add collaborators

collaborators must create a pull request if they want to push to your remote repo

Fork a repo = create a remote copy of an existing remote repo

Clone a repo = create a local copy of an existing remote repo

http://slides.com/michaelfreeman/git-collaboration#/

How to use GitHub - Collaboration

In teams of 2

#1 member 1 fork IHI repository

#2 member 1 will add member 2 as collaborator to their forked repository

#3 both members clone this repository to create a local repo and edit files on their own laptops
git clone <https://github.com/....>

#4 both members edit the file - open '..html' and add new lines

#5 member 1 commit and push their changes

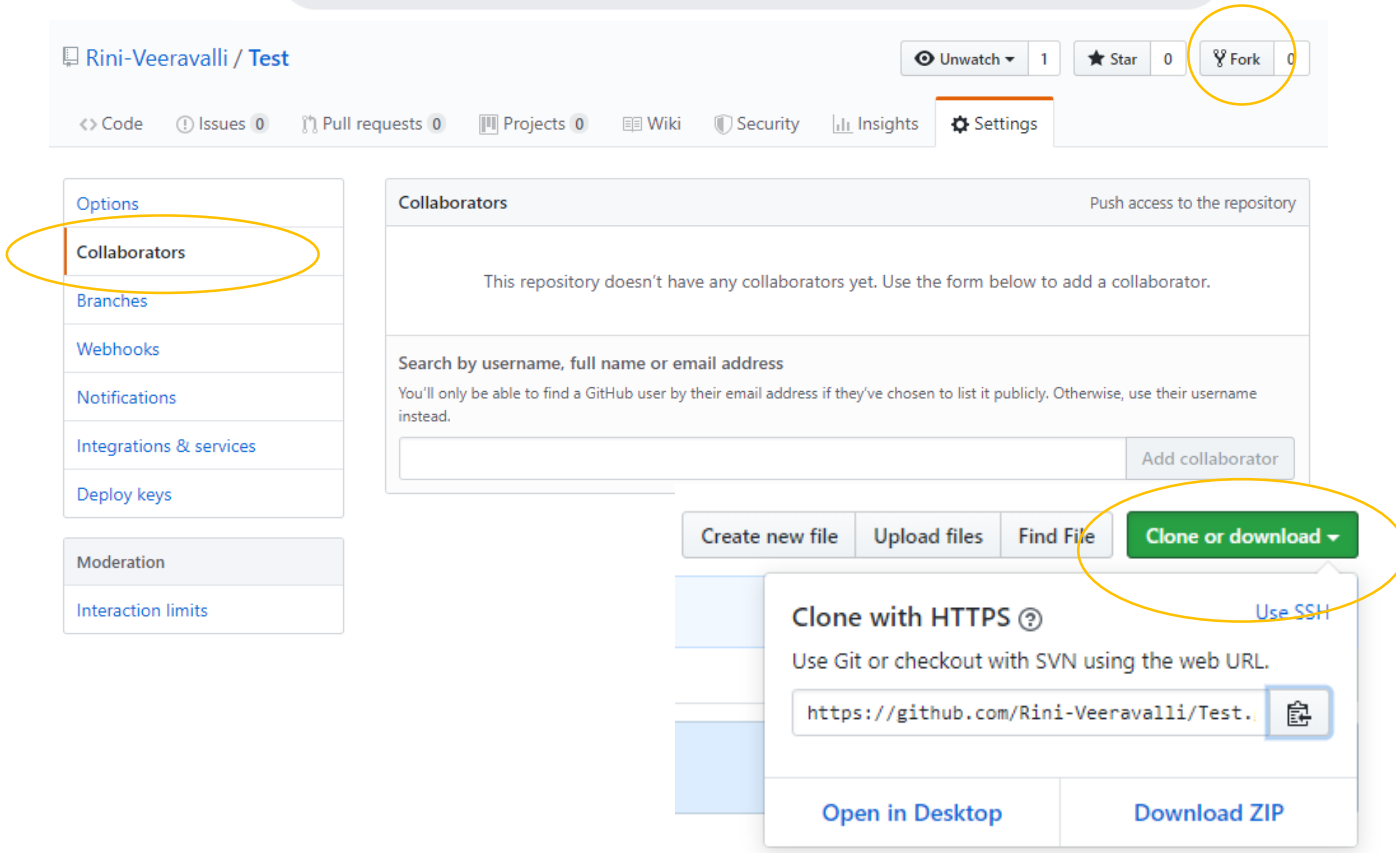
#6 member 2 commit and push their changes

#7 member 2 create pull request

#8 member 1 merge pull request

TRY THIS

https://github.com/ucl-ihi



SUCCESS!

Next Steps

- Resolve conflicts [important!]
  - can occur when multiple people are working on the same file e.g. from their local repos or on different branches and you try to merge

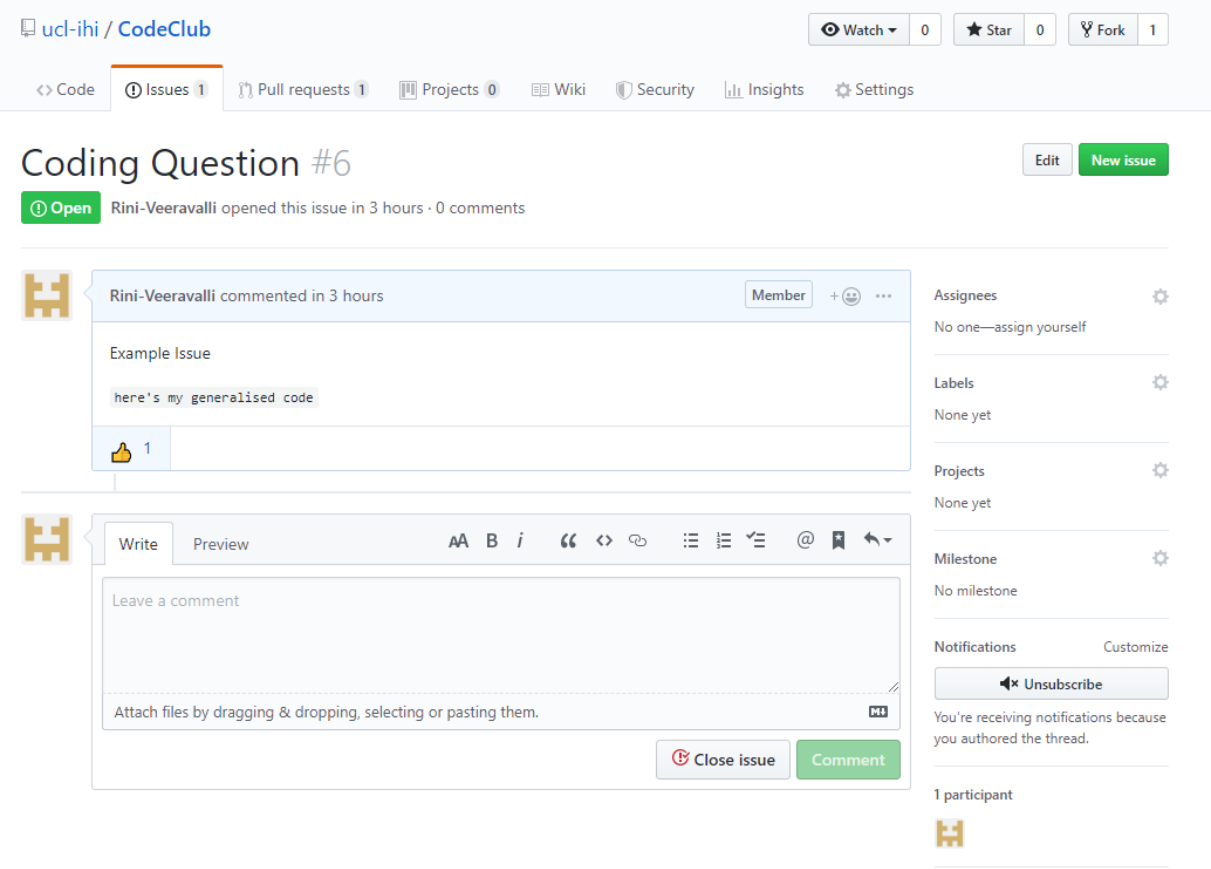
- Communicate with others online

  -> GitHub Issues

GitHub Issues

TRY THIS

Post your problems on the *Code Club repository issue page* !



- Assign and track tasks
- Great for planning projects
- Communicate with others

You can assign members to issues and close issues

Have a go: post, answer, upvote

Resources

- Git cheat sheet from git - the simple guide
- official git pages
- Software carpentry lesson – Version Control with Git
- A Visual Git Reference
- Git 101: Git and GitHub for Beginners
- Interactive git tutorial
- GitHub guides : Issues
- Git collaboration slides
- Git and GitHub for R users
- An intro to Git and GitHub tutorial for beginners – blog post

NEXT ON CODE CLUB

Code Club
Workshop Series

Season 1 Ep. 02

> Command line
> Bash
> SSH

Want to join Code Club and receive calendar invites to all our sessions?
Want to join the IHI GitHub organisation?
Want to collaborate on the Code Club repository?
Membership form

ucl-ihi.github.io/CodeClub

rini.veeravalli.18@ucl.ac.uk

**Time and location TBC*

How to – Clean Up

#1 delete remote repo

- **Go to your repo on GitHub > *Settings* > *Danger Zone* > *delete this repository***

- ***Or on command line / BASH***

check what remote repos you have	*git remote –v*
delete remote repo by name	*git remote rm <repo name>*
check repo is deleted	*git remote -v*

#2 delete local repo

within your repo, view hidden git files with *ls –a* (may need different command for Macs)

- to delete git repo, but keep your files: *rm –rf .git*
  check that hidden .git file is gone

- You can then delete the whole directory (including the files) if you want

Be cautious when deleting

TROUBLESHOOTING

DISCUSSION

QUESTIONS?

PROBLEMS?

COMMENTS