Women in Computer Science at LSU

Spring 2023

Introduction to Collaborative Projects

A few considerations…

- This is a workshop for all levels…
  - …but we start from the ground up
- Required knowledge: CLI basics, Git basics
- Follow along to the practical demo
- Ask questions
- Have fun! ☺

In the previous chapters…

- Part 1: CLI
  - Navigate the file system
  - Create/delete/edit files
- Part 2: Git
  - Initialize/clone a repository
  - Stage, commit, and push changes
  - Creating, changing, merging branches

```
(base) rmettig@Artemiss-MacBook-Air test % git init
Initialized empty Git repository in /Users/rmettig/Desktop/test/.git/
(base) rmettig@Artemiss-MacBook-Air test % touch file1
(base) rmettig@Artemiss-MacBook-Air test % echo "hello" > file1
(base) rmettig@Artemiss-MacBook-Air test % git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1

nothing added to commit but untracked files present (use "git add" to track)
(base) rmettig@Artemiss-MacBook-Air test % git add *
(base) rmettig@Artemiss-MacBook-Air test % git commit -m "first commit"
[main (root-commit) 18864c0] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1
(base) rmettig@Artemiss-MacBook-Air test %
```

Workshop

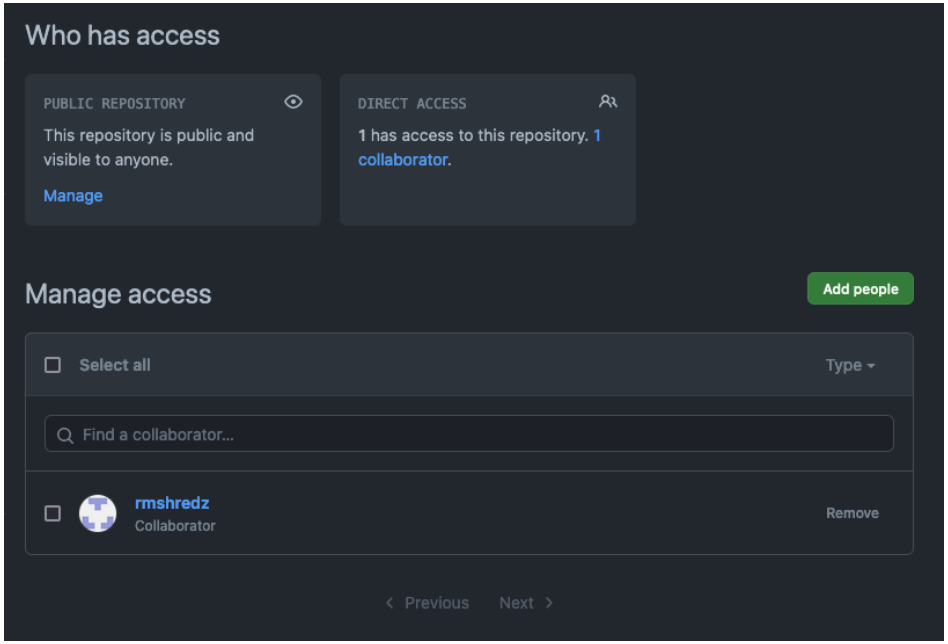
- Linux-based
  - Kali Linux VMs
- Pre-reqs
  - CLI basics (navigation, file mgmt., etc.)
  - Git basics
- Project collaboration
  - Shared repo vs forked repo
  - Pull requests
  - Merge conflicts
  - Issues
- Exercises

Different ways to collaborate

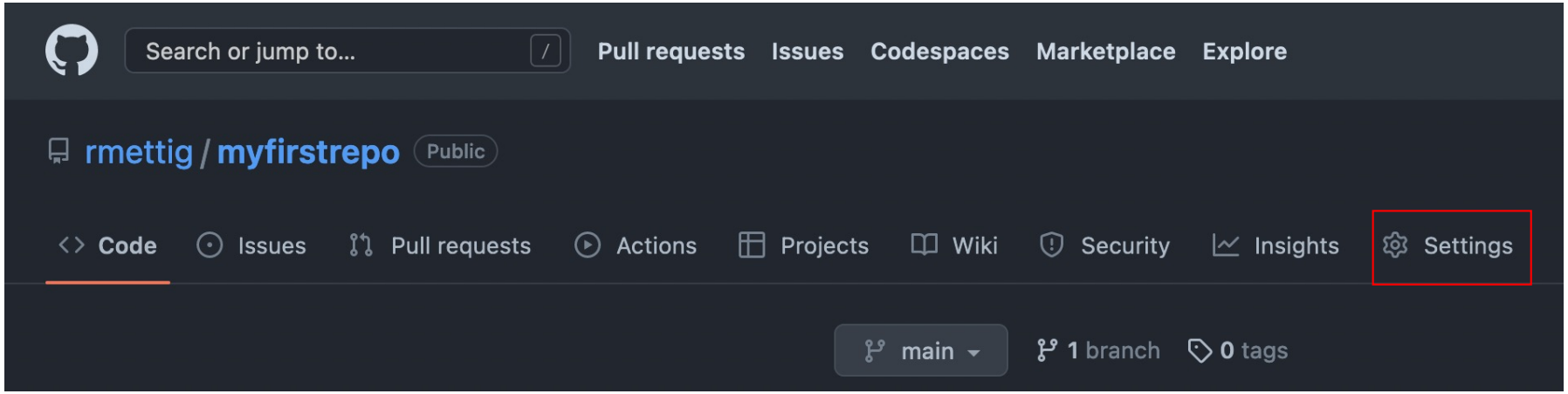
- Shared repository model
- Fork and pull model

Shared repository

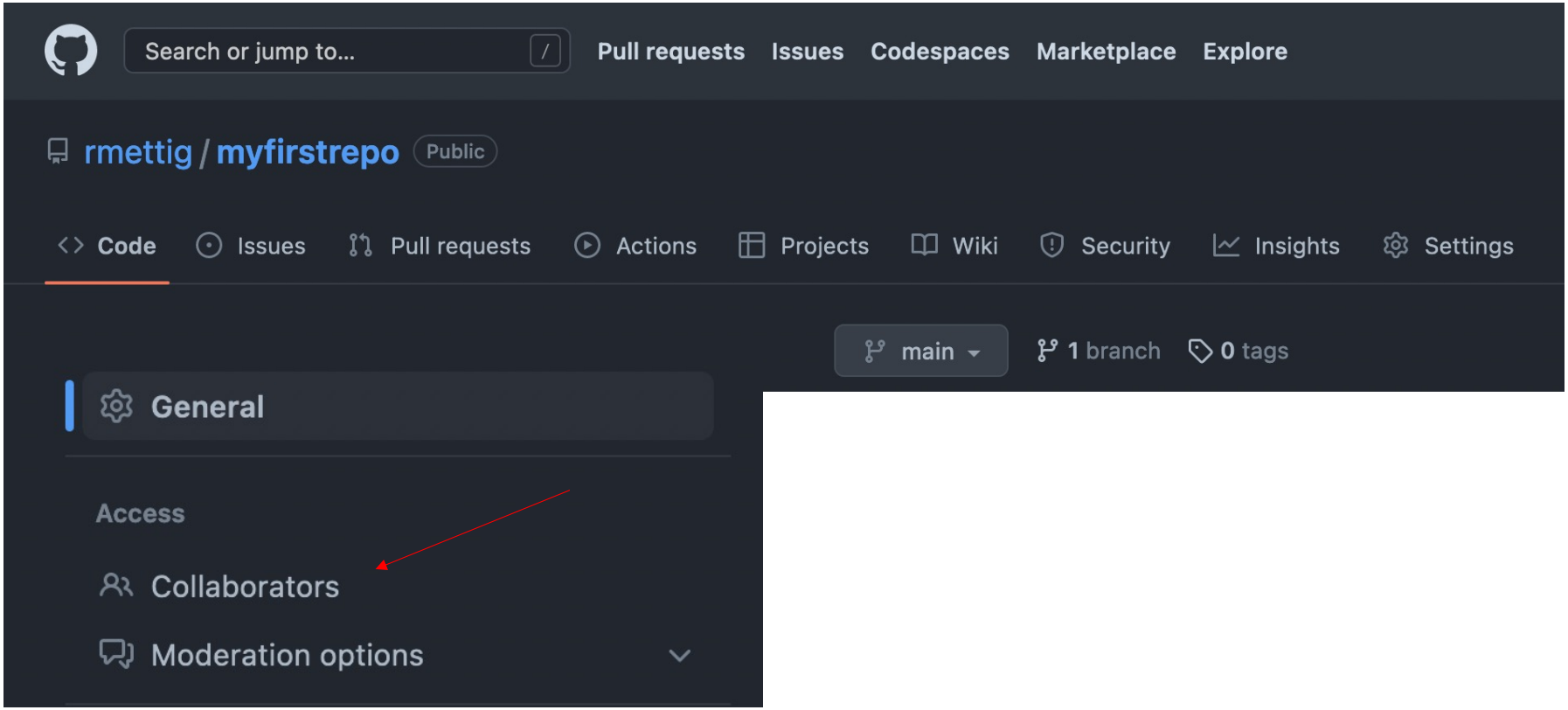
- One repository you <u>own</u>
  - You can control all the settings
- You have the ability to send invites to collaborate
  - The invitee will get an email and has to accept it before accessing
- Add and remove people, transfer ownership



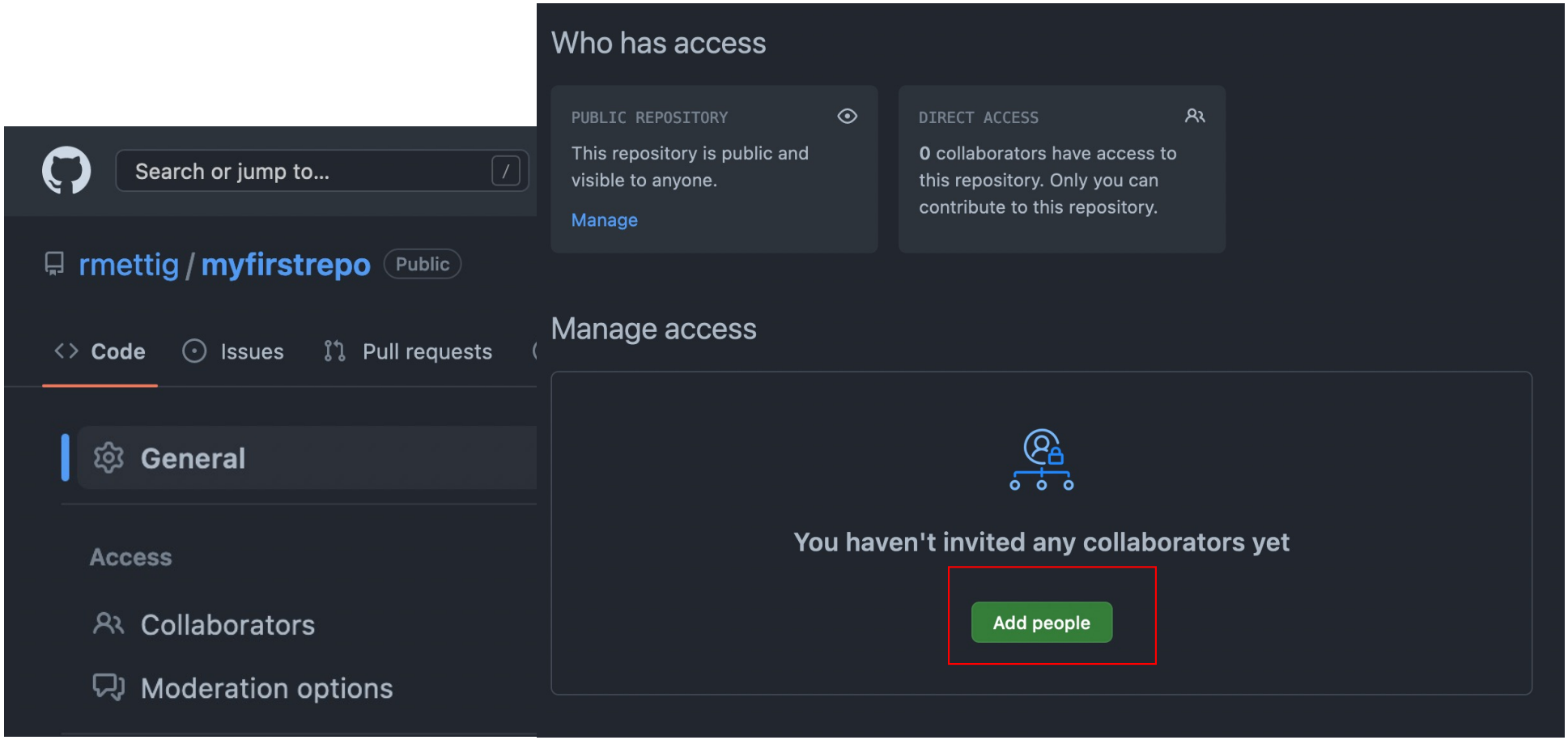
Adding collaborators to a shared repository



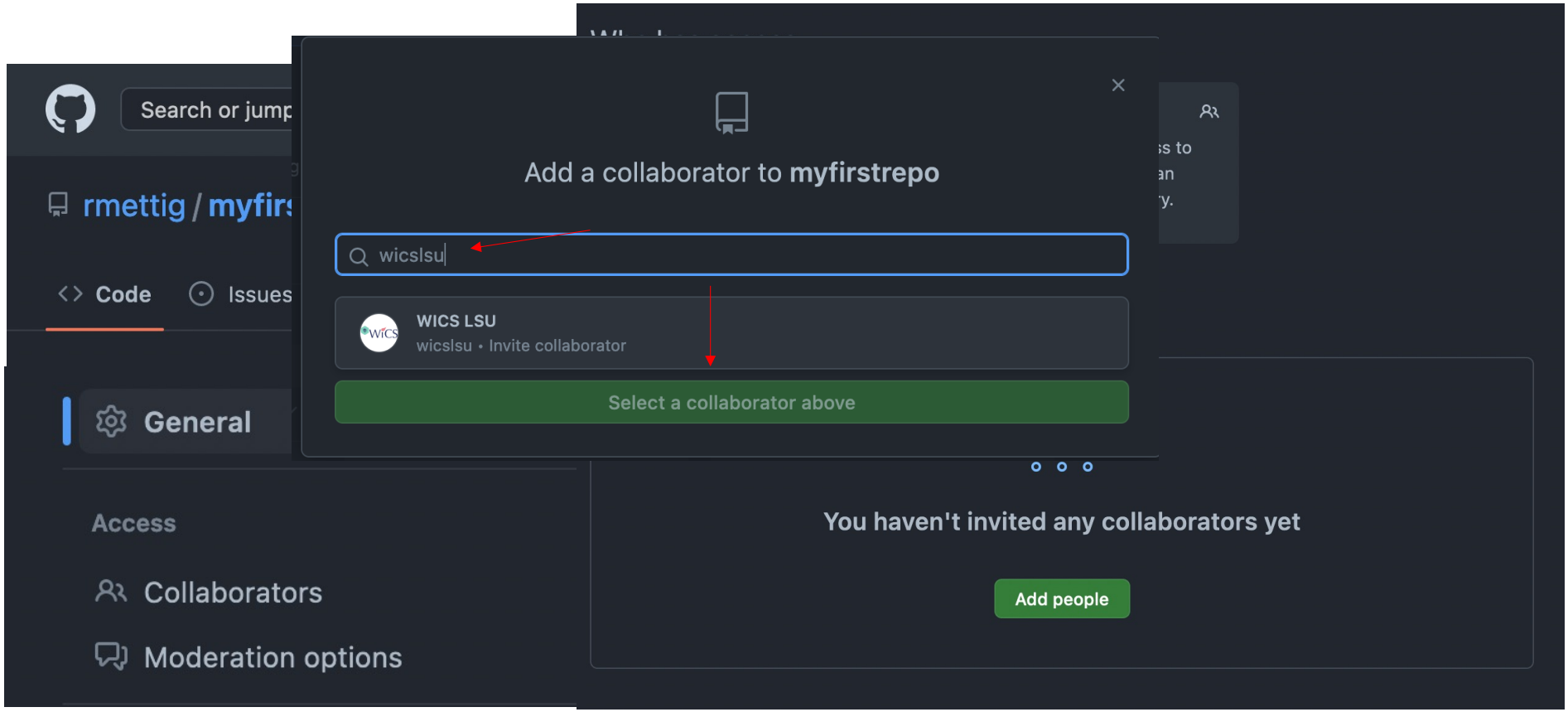
Adding collaborators to a shared repository

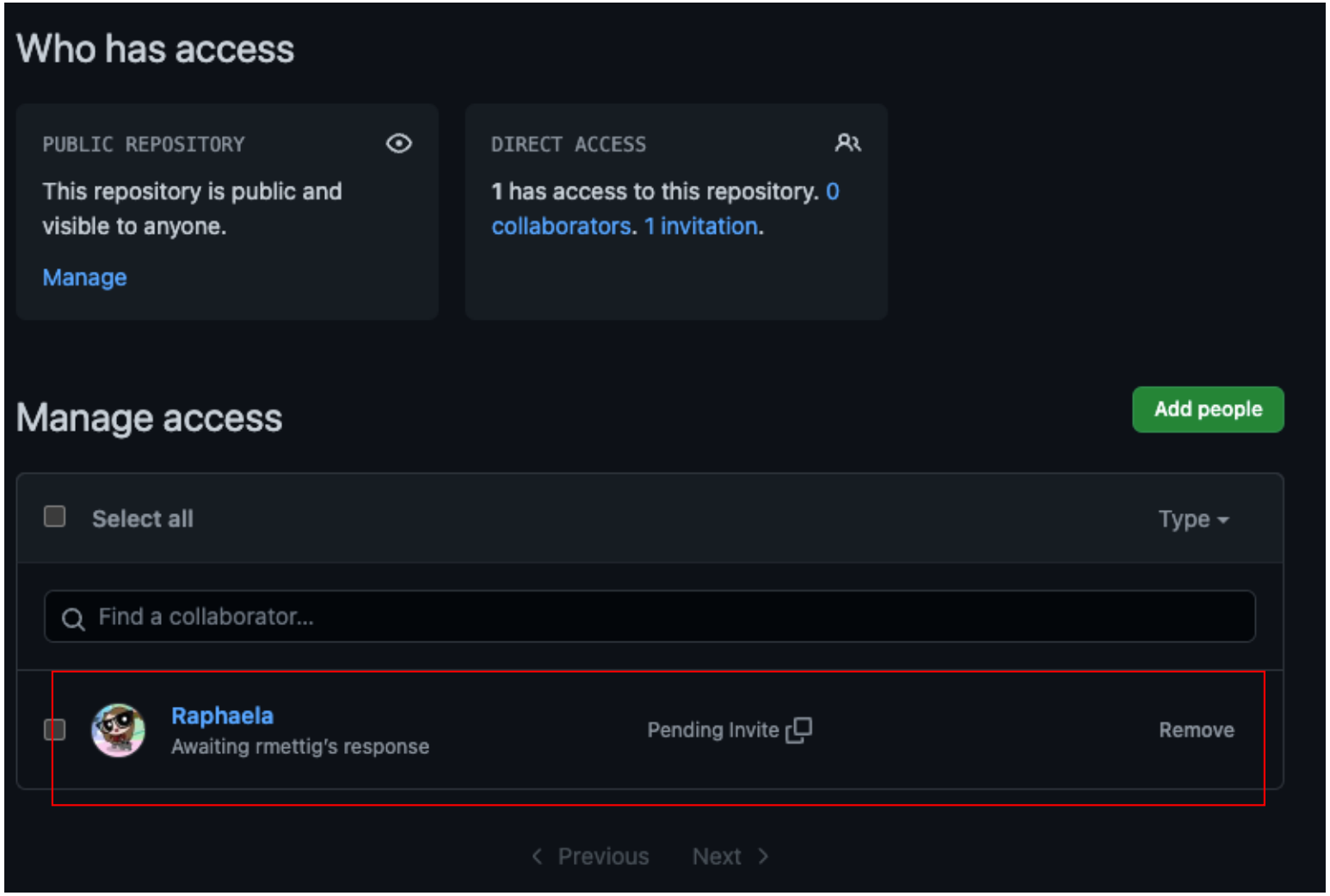


Adding collaborators to a shared repository



Adding collaborators to a shared repository




Who has access
PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage
DIRECT ACCESS
1 has access to this repository. 0 collaborators. 1 invitation.
Manage access
Add people
Select all
Type
Find a collaborator...
Raphaela
Awaiting rmettig's response
Pending Invite
Remove
Previous
Next

Exercise 1 – simple PR (pair up!)

- Create a new repository with your partner, and invite them as a collaborator
- One of you will create a README with contents and commit it to main
- Each of you will create a personal working branch and individually create new files and commit to your personal branches (play around with it!)
  - You will see that it will prompt you to create a pull request
- Look over the pull requests with your partner, merge them through
  - Delete branches after merge

Merge conflicts

- Starting from the same commit
- Two people making changes that conflict with each other when merging
  - Edit the same line in the same file
  - Delete a file in a commit where the other person still has it

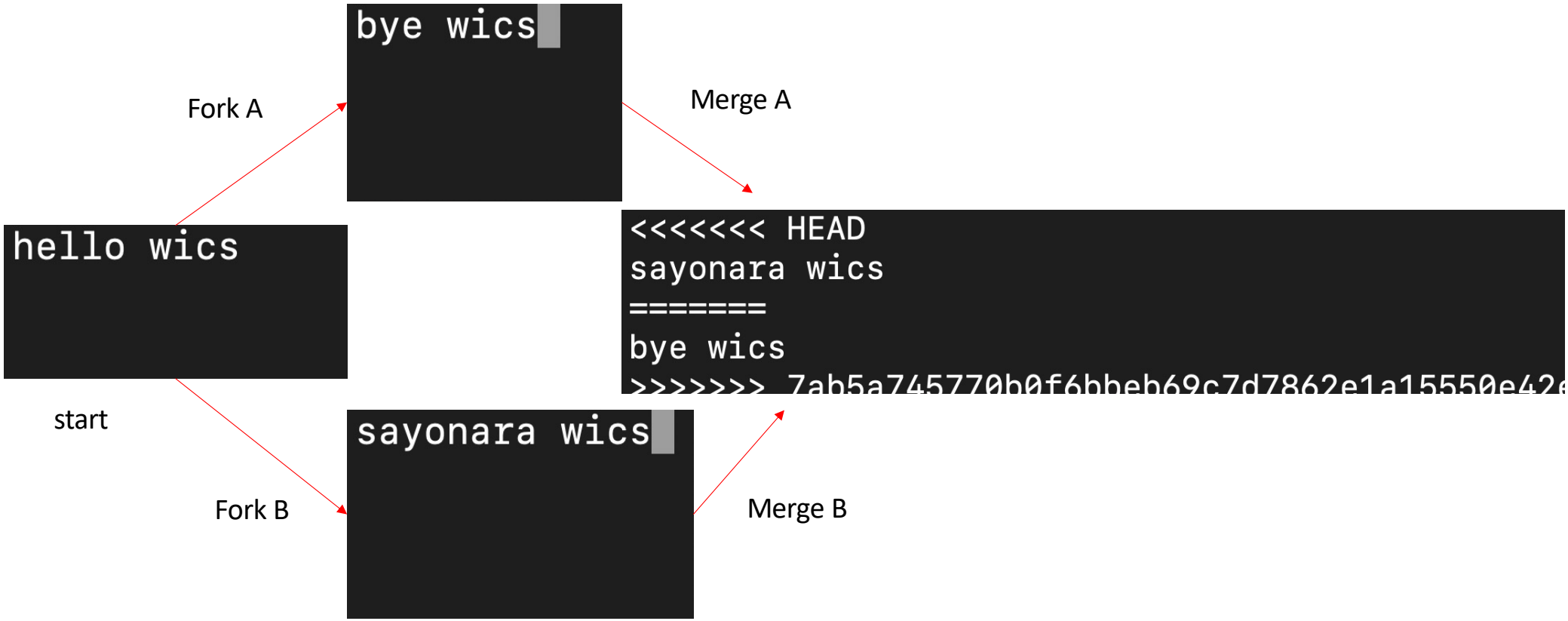
```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
rmettig@Raphaelas-MBP myfirstrepo % git status
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
rmettig@Raphaelas-MBP myfirstrepo % cat README.md
<<<<<<< HEAD
sayonara wics
=======
bye wics
>>>>>>> 7ab5a745770b0f6bbeb69c7d7862e1a15550e42e
```

Resolving merge conflicts



Resolving merge conflicts

We need to figure out what text between the arrows we want to keep!

```
<<<<<<< HEAD
sayonara wics
=======
bye wics
>>>>>>> 7ab5a745770b0f6bbeb69c7d7862e1a15550e42e
```

```
sayonara means "bye", wics
```

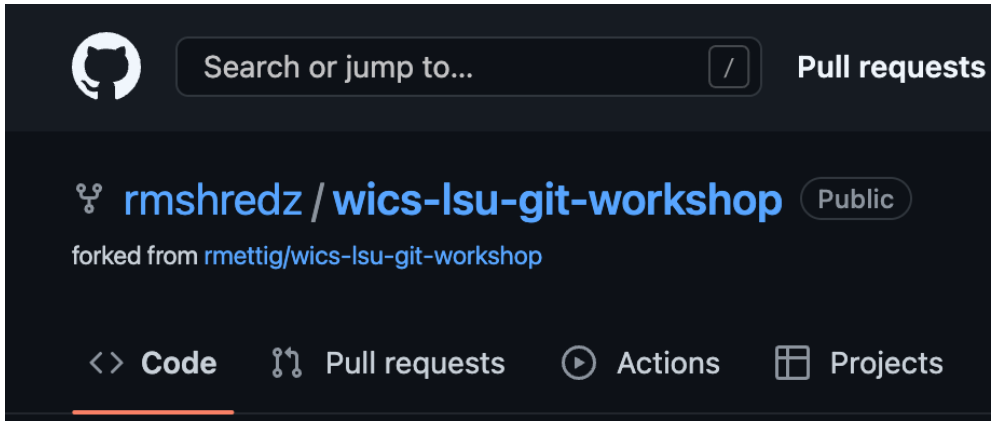
Stage, commit, push

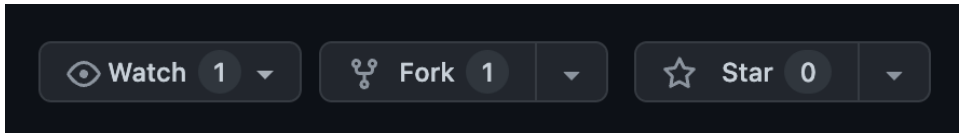
Exercise 2 – merge conflict (pair up!)

- Still in the same repository as ex 1, each person will create another working branch each
- In each branch, each of you will make a change to <u>the same line in the README both of you should have from the initial branch</u>
- Try to merge – it should result in a conflict
- Fix the conflict and push the change

Fork and pull

- Find a project you want to work on
- The repository is owned by someone else
- You make a personal copy of it to do your work
- Once done, you ask the original project maintainer to review your changes



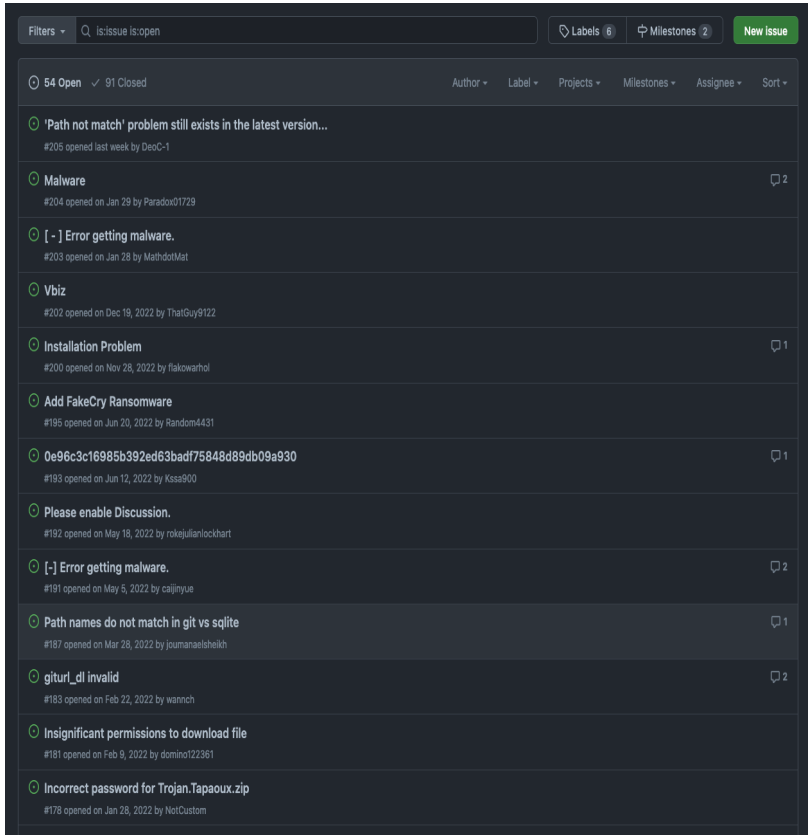


Exercise 3 – individual group work ☺

- Create a personal fork of: https://github.com/rmettig/wics-lsu-git-workshop
- Create 3 files with lyrics to 3 songs that you like (one per commit)
- Add your name to the readme and the songs you included
  - We will review merge conflicts together
- One you're done, create the pull request to the original project
- We will review them together and I will approve them!

Github Issues

- Discussion forum within each project
- Always good to scan it before starting to work on an existing project
- Some etiquette expected
  - Follow StackOverflow question guidelines when in doubt (link in references slide)



what is the best song out at the moment #4

Edit | New issue

Closed | rmettig opened this issue 1 minute ago · 2 comments

rmettig commented 1 minute ago — Owner

all opinions welcome!

rmshredz commented now — Collaborator

country roads, take me home

rmettig commented now — Owner | Author

you're done

rmettig closed this as completed now

Write | Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Create a branch for this issue or link a pull request.

Notifications Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

2 participants

Lock conversation

What's next?

- We're done!
- Use it as much as you can
  - Practice, practice, practice!
- Find cool projects to work on
  - Part 2 slides have some tips on how to do that
- Explore what else Github has to offer!
  - Student discouts, automation, configuration, test/build pipelines... so much more!

Further reading

- “Linux Basics for Hackers”
  - NoStarchPress link: https://nostarch.com/linuxbasicsforhackers
- Github documentation (refs in next slide)

https://nostarch.com/linuxbasicsforhackers

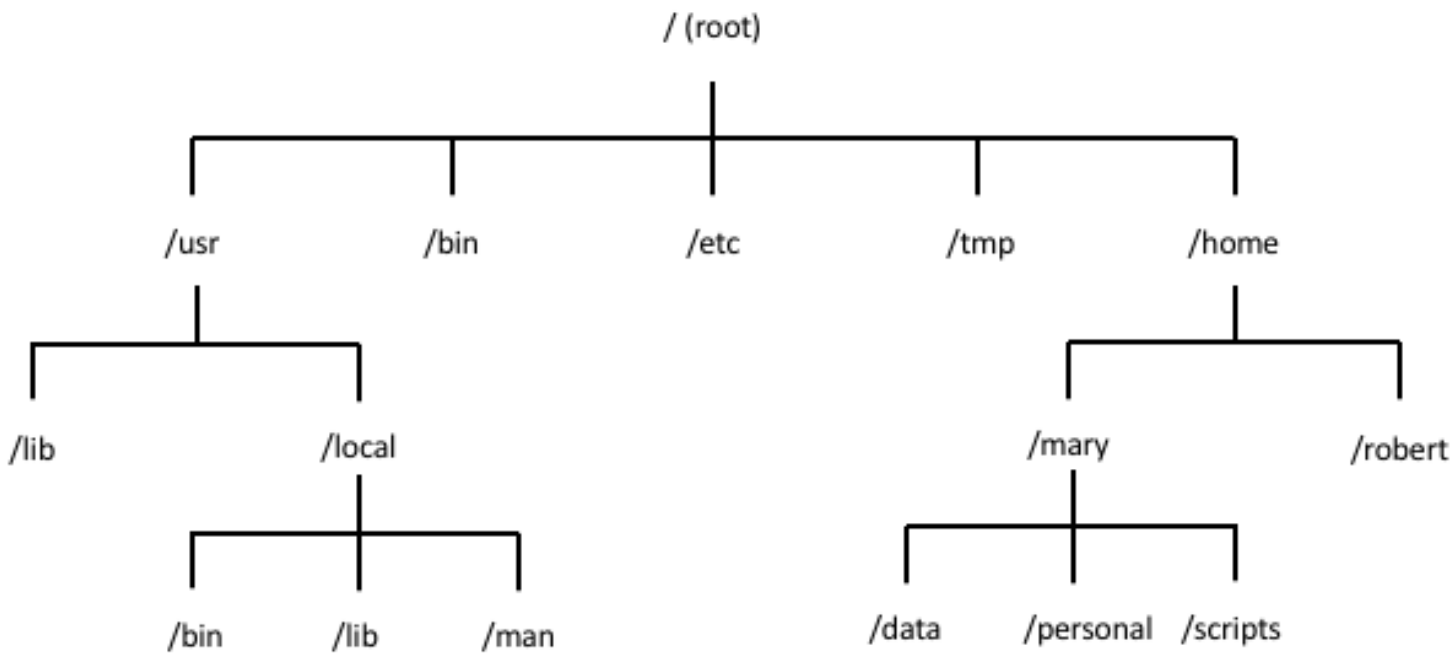
References

- Git documentation - https://git-scm.com/
- Github documentation - https://docs.github.com/en
- Github access tokens - https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token
- Github PR doc - https://docs.github.com/en/pull-requests
- Github forks - https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/about-forks
- StackOverflow question guidelines - https://stackoverflow.com/help/how-to-ask

Appendix A

CLI basics reference

Linux File System Overview



Navigation Commands

Command	Purpose	Command	Purpose
pwd	path of working dir	cd ..	change back one level
ls	list contents of dir	cd ../..	change back two levels
ls –l	list contents long format	cd ../../..	change back three levels
ls –a	list all contents in dir	…	so on and so forth
ls –t	list and sort by time	cd /	change to root dir
ls <target dir>	list contents of target dir	cd	change to home dir
cd <target dir>	change directory to target dir	cd ~	change to home dir

File/dir creation and editing

Command	Purpose	Command	Purpose
man <command>	manual!! Check this when in doubt!!	mv <old name> <new name>	renames a new file or dir
touch <file name>	creates an empty file	rm <file name>	deletes a file or empty dir
mkdir <name>	creates empty dir	rm –r <dir name>	recursively deletes all files in non-empty dir
mv <file> <target>	moves a file to the specified target dir	rm –rf <dir name>	same as above, but forces and overrides any warnings
cp <file> <target file>	copies a file into the specified target file	head -10 <file name>	shows first 10 lines in file
cat	see file contents	tail -10 <file name>	shows last 10 lines in file
mkdir	creates new dir	nl <file name>	shows the contents with numbered lines

Adding content to file

- Append or overwrite with echo

```
rmettig@ubuntu: ~
rmettig@ubuntu:~$ echo "hello " > hello.txt
rmettig@ubuntu:~$ cat hello.txt
hello
rmettig@ubuntu:~$ echo "world" >> hello.txt
rmettig@ubuntu:~$ cat hello.txt
hello
world
rmettig@ubuntu:~$ echo "goodbye" > hello.txt
rmettig@ubuntu:~$ cat hello.txt
goodbye
rmettig@ubuntu:~$
```

- Text editor

Vim basics

Command	Purpose	Command	Purpose
vim <file>	open file in Vim	:q	quit
[I]	insert mode (edit file)	:w	write to file (save)
[ESC]	return to normal mode	:wq	write then quit
[UP, DOWN, RIGHT, LEFT]	navigate the editor	:x	write changes and close file
:q!	discard changes and quit	:w <new name>	save current file as new file

Vim cheat sheet:
https://www.cs.cmu.edu/~15131/f17/topics/vim/vim-cheatsheet.pdf

Package management

Command	Purpose	Command	Purpose
sudo <command>	run command with admin privileges	sudo apt-get upgrade	updates the installed packages in your system
apt-get install <package>	install the package		
apt-get remove <package>	uninstall the package		
apt-get purge <software>	uninstall and remove configuration files		
apt-get update	updates list of packages available for download		

Other useful commands

Command	Purpose	Command	Purpose
history	shows CLI history	ssh	connect to remote server over ssh
!<history number>	runs the command in history	chmod	change file permissions
find <start> -type <f|d> -name <name>	scans the FS for a target file or directory	cowsay <text>	prints cow made of ASCII art with input text
grep <string>	will only show results containing string	clear	clear the contents in the terminal window
less <file>	display file contents in a fixed amount of lines	[UP, DOWN]	scroll through recent command history
sed	text replacement	which	checks whether a given command is installed in $PATH
wc	counts output lines		

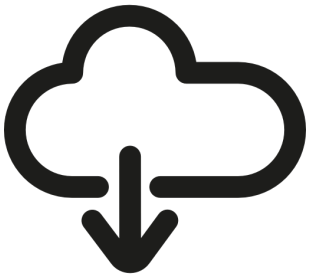
Appendix B

Git basics reference

Useful Terminology

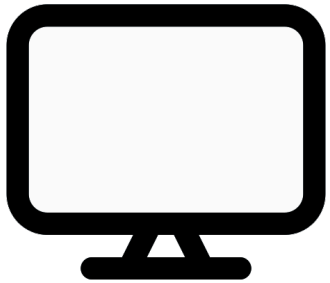
Remote

- Needs network connection
- Only keeps backup of what you push



Local

- Works offline
- On your machine only



Git Terminology

- Repository
  - Contains project files and stores file version history
- Branch
  - A version timeline within a repository
- Commit
  - “Save” the changes done to the project to the working version
- Clone
  - A local copy of a remote repository
- Fork
  - Personal copy of another user’s repository
  - Changes made to your fork don’t affect the other user’s directly

https://docs.github.com/en/get-started/quickstart/github-glossary

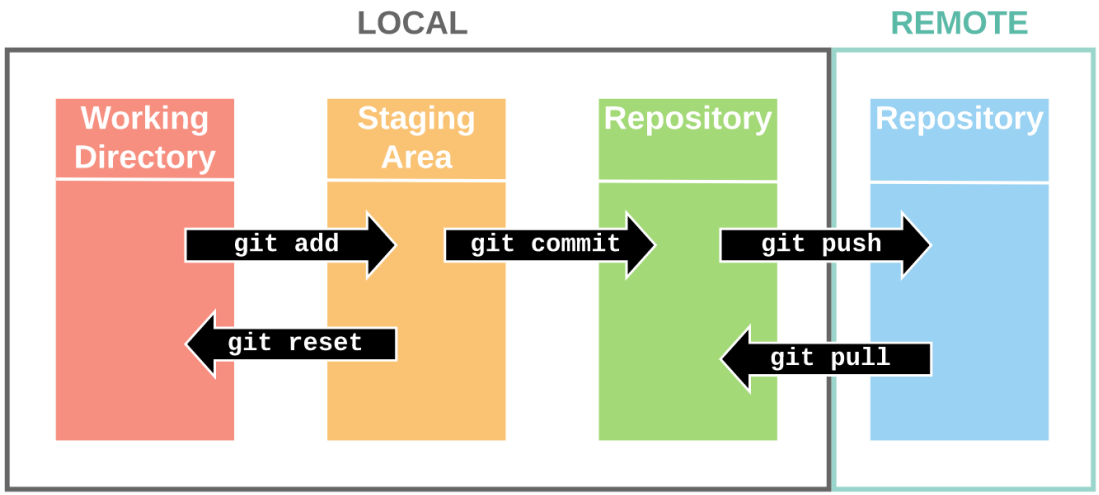
Git Terminology (cont.)

- Fetch
  - Get changes from a remote repository without merging them
- Merge
  - Get changes from one branch and applies it into another
  - Either in the same repository or on another fork (aka Pull Request)
- Pull
  - Fetch and merge changes in one command
- Push
  - Send committed changes to a remote repository
- Pull Request (PR)
  - proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators

https://docs.github.com/en/get-started/quickstart/github-glossary

General Workflow

- Clone/Fork/Create repository
- Create working branch
- Work and make changes
- Commit changes (frequently)
- Push local changes
- Merge changes into main branch



Img src: https://support.nesi.org.nz/hc/en-gb/articles/360001508515-Git-Reference-Sheet

Git Commands

Command	Purpose	Command	Purpose
git config	check local repo config	git branch	
git config user.name “username”	set local repo username (--global flag can set for all)	git config user.email “email”	set local repo email (--global flag can set for all)
git init	initialize git repo	git push –u origin <dest-branch>	push local changes to destination branch
git clone <url>	download remote repo	git pull	fetch and merge remote changes to working repo
git add <file>	add files to staging	git merge <branch>	merge changes into your current branch
git status	check staging status	git diff <branch-1> <branch-2>	show changes
git commit –m “<message>”	commit current changes to local working dir	git reset --hard origin/main	rollback working dir to that of last commit erasing changes
git remote	show remote version of repository	git reset --soft origin/main	rollback working dir to that of last commit keeping changes
git checkout <branch>	change branch	git log	list of commits on a branch

More commands

Command	Purpose	Command	Purpose
rm –rf .git	undo $git init	git checkout –b <new-branch>	create a new branch and switch to it
git branch –a	list all branches including remote	git help	git CLI manual!
git --version	check current version (or if it's installed!)	git branch -M main	rename branch as main
git config --list --show-origin	see current config settings	git remote add origin <url>	link working repo to remote repo!!
git rm –cached <file>	remove file from staging	git restore <file>	reverts changes in staged file