Version Control

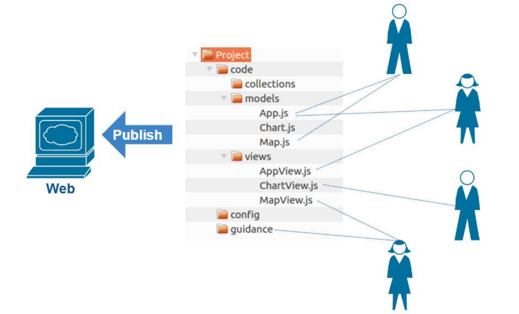
What you will learn today

- What version control is
- Why you need it (or maybe you don't?)
- *Session 1:* basic concepts
- *Session 2:* remote hosting
- *Session 3:* team working
- Up and running with tools and account

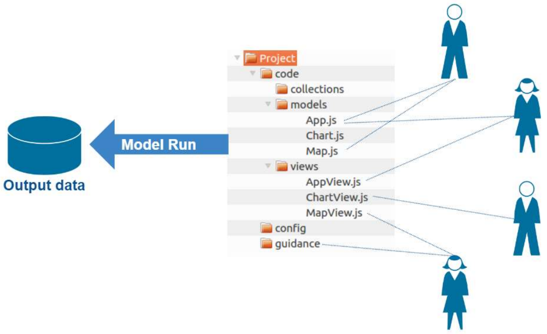
What is version control?

- '*Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later*'
- Great free book: https://git-scm.com/book/en/v2
- Excellent Atlassian tutorial: https://www.atlassian.com/git/tutorials/what-is-version-control

Typical scenario in our group



Typical scenario with a model run?



What do you want from version control?

Recovery

Reproducibility

Track changes

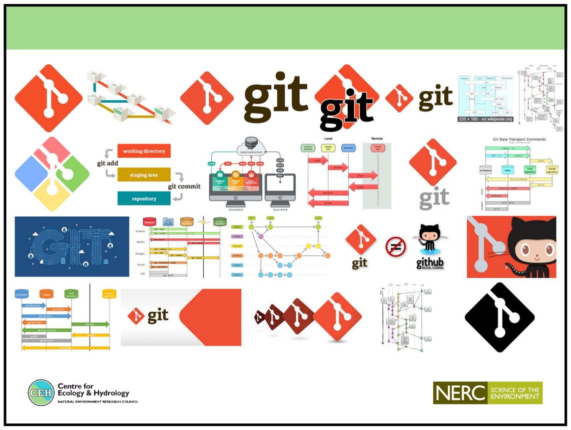
Audit and Traceability

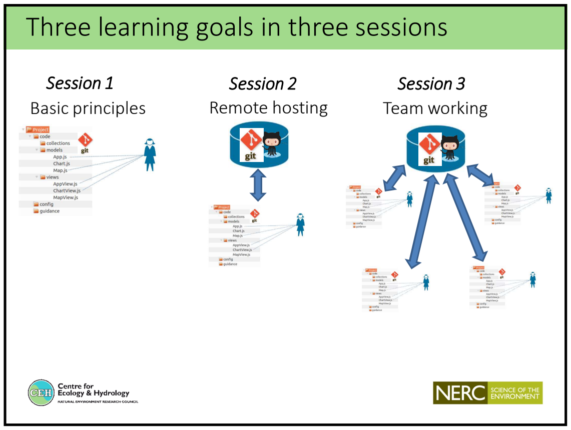
Funding requirement

Safety and resilience

Collaboration

Experiments on code





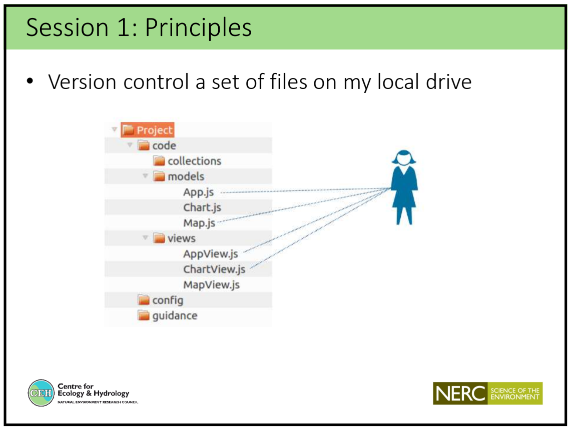
Three learning goals in three sessions

Session 1
Basic principles

Session 2
Remote hosting

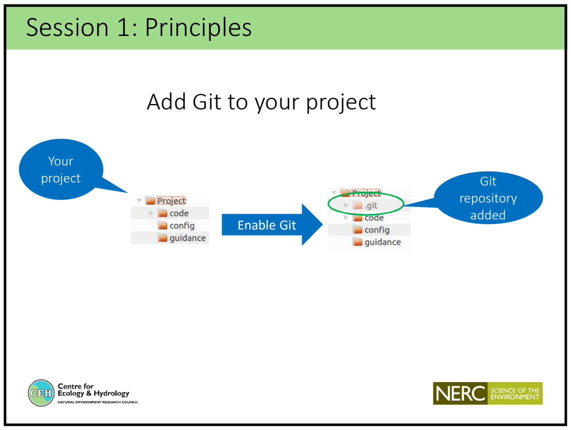
Session 3
Team working

git
git
Centre for Ecology & Hydrology
NERC SCIENCE OF THE ENVIRONMENT



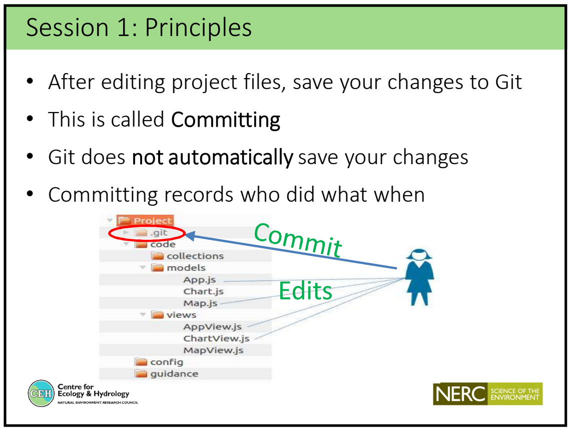
Session 1: Principles

- Version control a set of files on my local drive



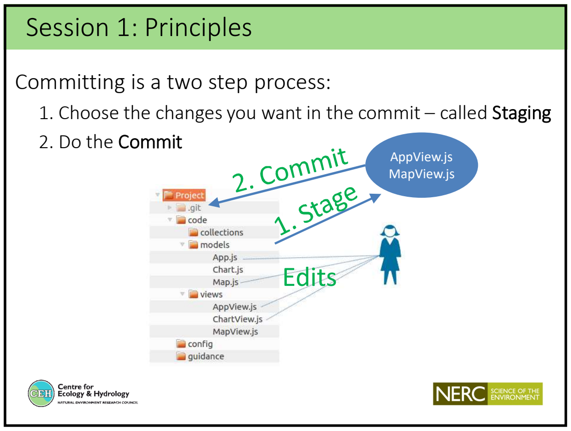
Session 1: Principles

Add Git to your project



Session 1: Principles

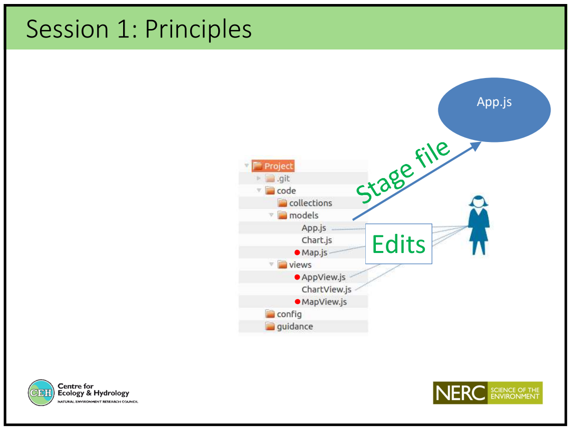
- After editing project files, save your changes to Git
- This is called **Committing**
- Git does **not automatically** save your changes
- Committing records who did what when



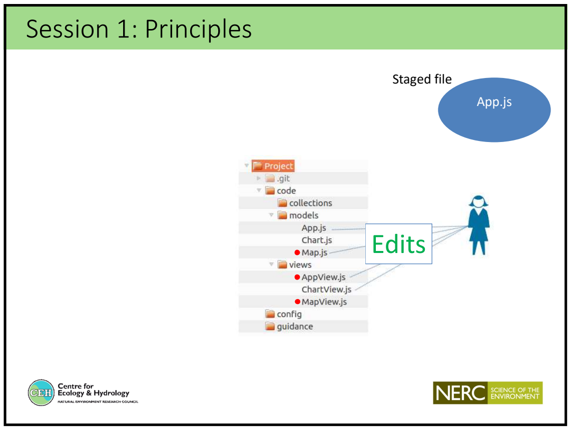
Session 1: Principles

Committing is a two step process:

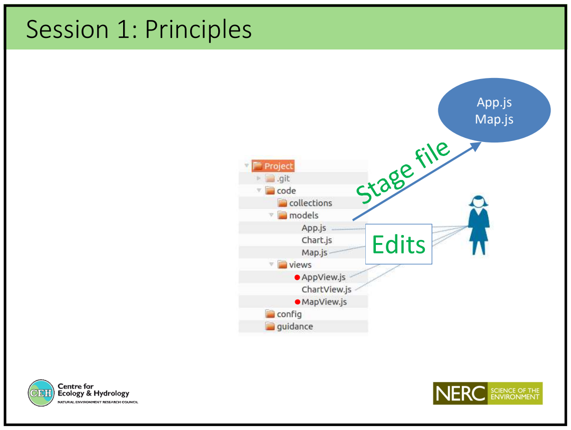
1. Choose the changes you want in the commit – called **Staging**
2. Do the **Commit**



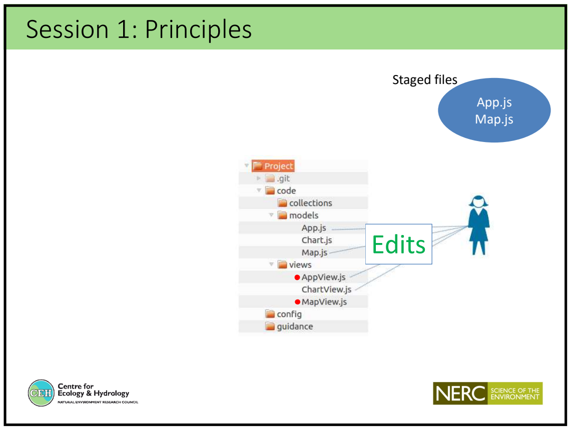
Session 1: Principles



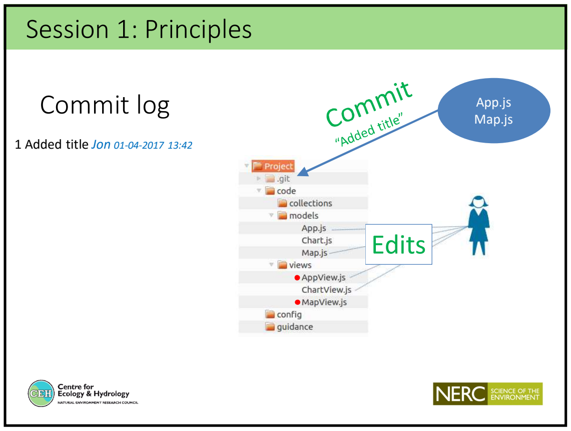
Session 1: Principles



Session 1: Principles



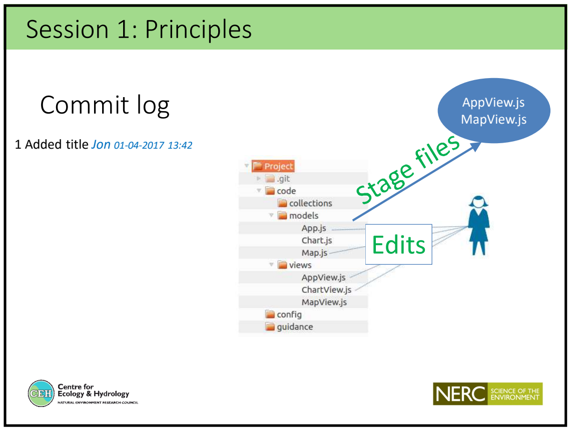
Session 1: Principles



Session 1: Principles

Commit log

1 Added title *Jon 01-04-2017 13:42*



Session 1: Principles

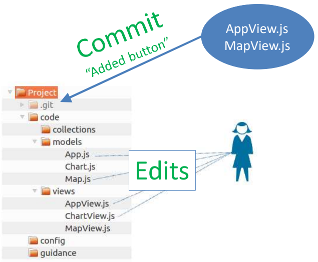
Commit log

1 Added title *Jon 01-04-2017 13:42*

Session 1: Principles

Commit log

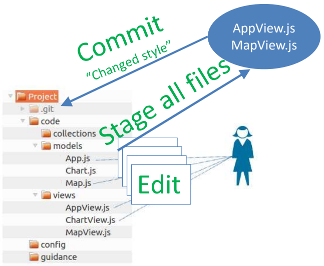
1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*



Session 1: Principles

Commit log

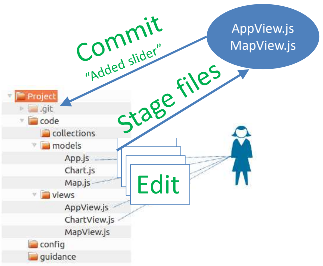
1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*
3 Changed style *Jon 01-04-2017 15:23*



Session 1: Principles

Commit log

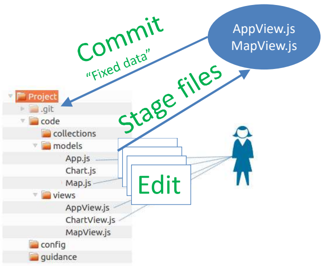
1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*
3 Changed style *Jon 01-04-2017 15:23*
4 Added slider *Jon 03-04-2017 08:32*



Session 1: Principles

Commit log

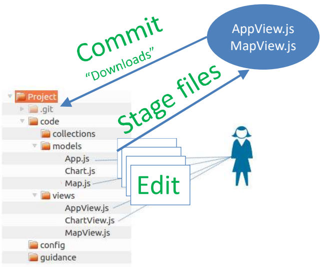
1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*
3 Changed style *Jon 01-04-2017 15:23*
4 Added slider *Jon 03-04-2017 08:32*
5 Fixed data *Jon 03-04-2017 10:15*



Session 1: Principles

Commit log

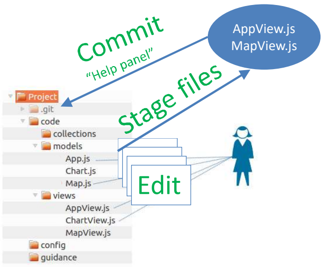
1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*
3 Changed style *Jon 01-04-2017 15:23*
4 Added slider *Jon 03-04-2017 08:32*
5 Fixed data *Jon 03-04-2017 10:15*
6 Downloads *Jon 03-04-2017 10:15*



Session 1: Principles

Commit log

1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*
3 Changed style *Jon 01-04-2017 15:23*
4 Added slider *Jon 03-04-2017 08:32*
5 Fixed data *Jon 03-04-2017 10:15*
6 Downloads *Jon 03-04-2017 10:15*
7 Help panel *Jon 03-04-2017 13:27*



Session 1: Principles

Commit log

1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*
3 Changed style *Jon 01-04-2017 15:23*
4 Added slider *Jon 03-04-2017 08:32*
5 Fixed data *Jon 03-04-2017 10:15*
6 Downloads *Jon 03-04-2017 10:15*
7 Help panel *Jon 03-04-2017 13:27*

- Snapshot of files at each commit
- Who did what when
- Checkout previous version

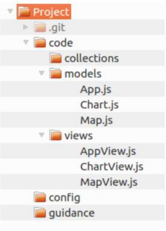
Session 1: Principles

Commit log

All files as they were at commit 3

1 Added title *Jon 01-04-2017 13:42*
2 Added button *Jon 01-04-2017 14:47*
3 Changed style *Jon 01-04-2017 15:23*
4 Added slider *Jon 03-04-2017 08:32*
5 Fixed data *Jon 03-04-2017 10:15*
6 Downloads *Jon 03-04-2017 10:15*
7 Help panel *Jon 03-04-2017 13:27*

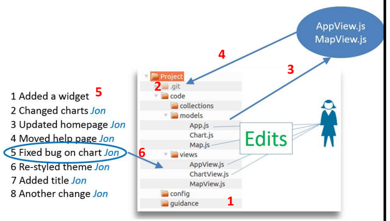
Checkout



Working copy

Session 1: Principles

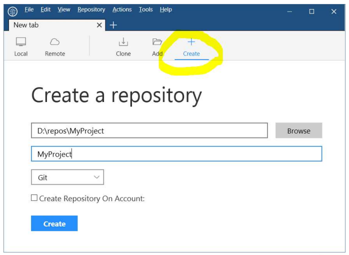
1 Working copy
2 Git repository
3 Stage files
4 Commit files
5 Commit log
6 Checkout


1 Added a widget
2 Changed charts *Jon*
3 Updated homepage *Jon*
4 Moved help page *Jon*
5 Fixed bug on chart *Jon*
6 Re-styled theme *Jon*
7 Added title *Jon*
8 Another change *Jon*

Session 1: Principles

All exercises use SourceTree

https://nerc-ceh.github.io/version_control/install_sourcetree_windows



Create a repository

D:\repos\MyProject

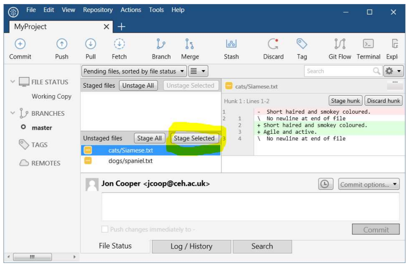
MyProject

Git

□ Create Repository On Account:

Session 1: Principles

Example of selecting files to stage in SourceTree



cats/Siamese.txt
dogs/spaniel.txt

cats/Siamese.txt

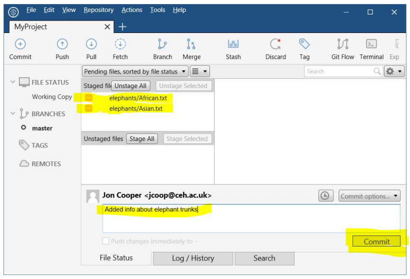
Hunk 1 : Lines 1-2

```
1      -  Short haired and smokey coloured.
2   1  \  No newline at end of file
    2  + Short haired and smokey coloured.
    3  + Agile and active.
    4  \  No newline at end of file
```

Jon Cooper <jcoop@ceh.ac.uk>

Session 1: Principles

Example of selecting committing staged files



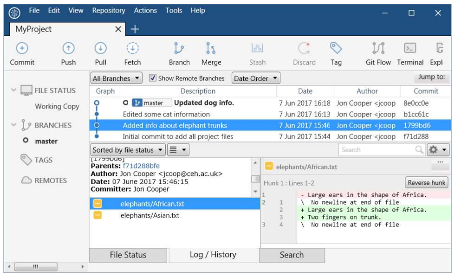
elephants/African.txt
elephants/Asian.txt

Jon Cooper <jcoop@ceh.ac.uk>

Added info about elephant trunks

Session 1: Principles

Example of viewing the commit log



Graph	Description	Date	Author	Commit
	master **Updated dog info.**	7 Jun 2017 16:18	Jon Cooper <jcoop	8e0cc0e
	Edited some cat information	7 Jun 2017 16:13	Jon Cooper <jcoop	b1cc61c
	Added info about elephant trunks	7 Jun 2017 15:46	Jon Cooper <jcoop	1799bd6
	Initial commit to add all project files	7 Jun 2017 15:44	Jon Cooper <jcoop	f71d288

[1799bd6]
Parents: f71d288bfe
Author: Jon Cooper <jcoop@ceh.ac.uk>
Date: 07 June 2017 15:46:15
Committer: Jon Cooper

elephants/African.txt
elephants/Asian.txt

elephants/African.txt

Hunk 1 : Lines 1-2

```
1      - Large ears in the shape of Africa.
2   1  \  No newline at end of file
    2  + Large ears in the shape of Africa.
    3  + Two fingers on trunk.
3   4  \  No newline at end of file
```

Session 1: Principles

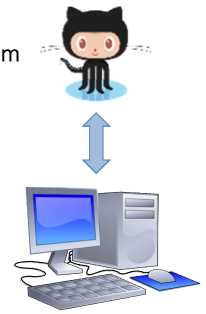
Exercise 1

https://nerc-ceh.github.io/version_control/exercise1

- Enable Git on a folder
- Review the status of files in the folder
- Repeatedly edit files and add your changes to Git
- Review changes
- Checkout a previous version of your changes
- Apply a tag to a specific version for future reference
- Exclude files from version control

Session 2: Remote repository

Github.com



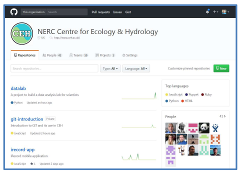
Others are available

- Gitlab
- Stash
- Bitbucket
- etc…

Session 2: Remote repository

- Register with Github.com
- Free for public and open source projects
- Join the NERC-CEH organisation on Github for free CEH ***private*** projects

https://github.com/NERC-CEH



NERC Centre for Ecology & Hydrology
UK http://www.ceh.ac.uk/

datalab
A project to build a data analysis lab for scientists
Python Updated an hour ago

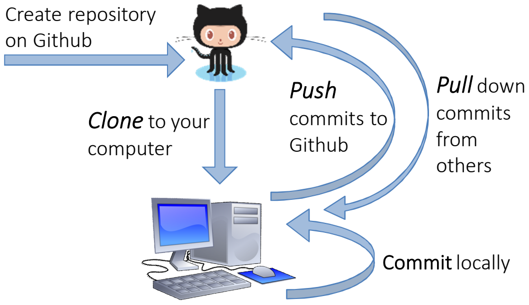
git-introduction Private
Introduction to GIT and its use in CEH
JavaScript Updated 2 hours ago

irecord-app
iRecord mobile application
JavaScript ★ 1 Updated 2 days ago

Session 2: Remote repository

- Create empty repository on Github
- Download a local copy – called *Cloning*
- Work locally just as you did in session 1, committing your changes
- Periodically synchronise with Github using *Push* and *Pull*

Session 2: Remote repository



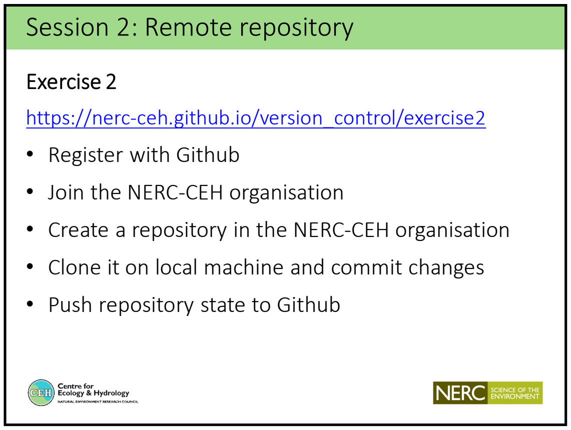
Create repository on Github

Clone to your computer

Push commits to Github

Pull down commits from others

Commit locally

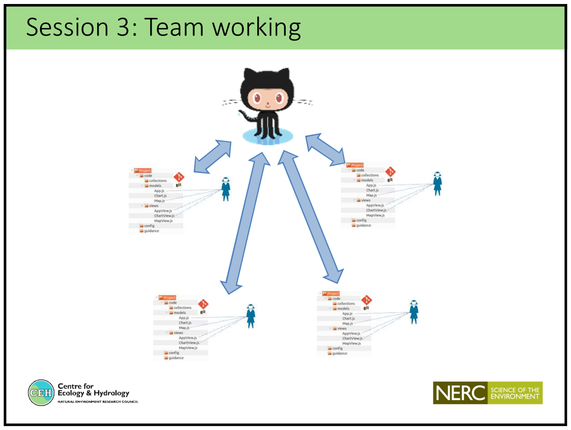


Session 2: Remote repository

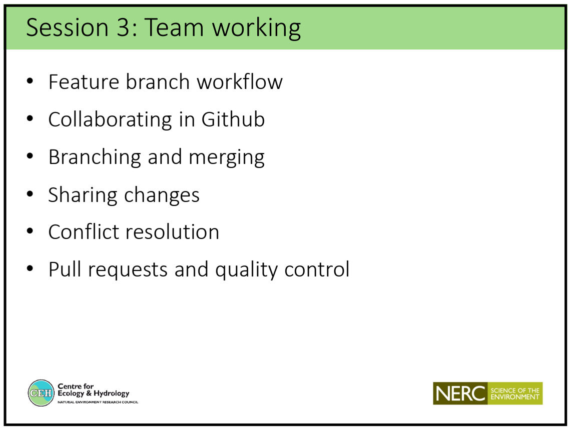
Exercise 2

https://nerc-ceh.github.io/version_control/exercise2

- Register with Github
- Join the NERC-CEH organisation
- Create a repository in the NERC-CEH organisation
- Clone it on local machine and commit changes
- Push repository state to Github

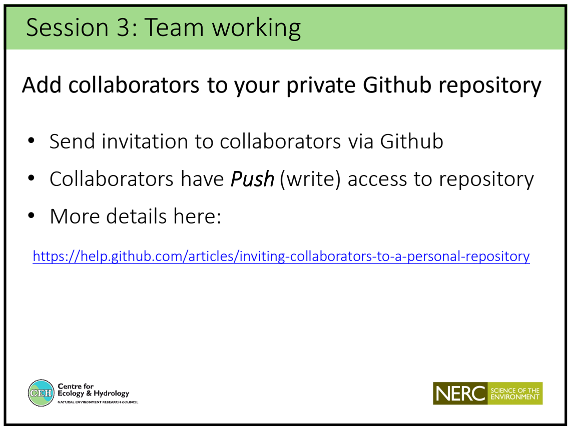


Session 3: Team working



Session 3: Team working

- Feature branch workflow
- Collaborating in Github
- Branching and merging
- Sharing changes
- Conflict resolution
- Pull requests and quality control

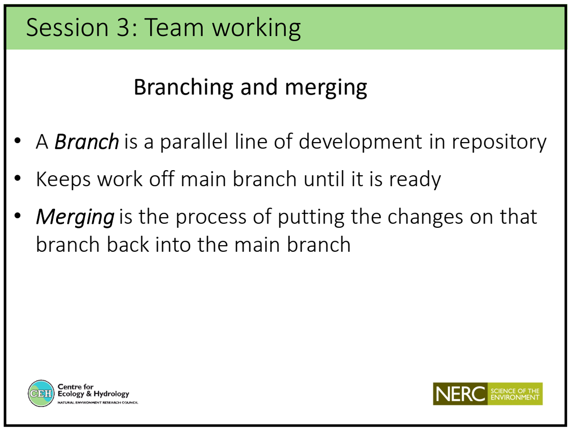


Session 3: Team working

Add collaborators to your private Github repository

- Send invitation to collaborators via Github
- Collaborators have *Push* (write) access to repository
- More details here:

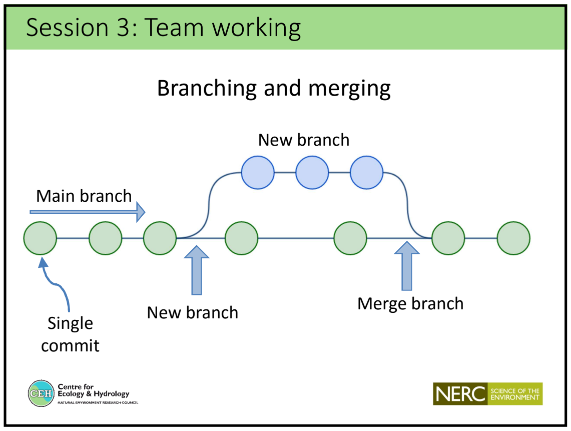
https://help.github.com/articles/inviting-collaborators-to-a-personal-repository



Session 3: Team working

Branching and merging

- A *Branch* is a parallel line of development in repository
- Keeps work off main branch until it is ready
- *Merging* is the process of putting the changes on that branch back into the main branch

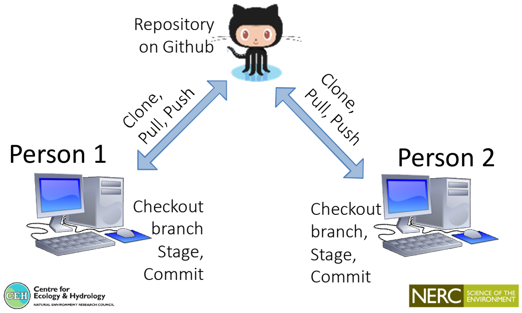


Session 3: Team working

Branching and merging

Session 3: Team working

Multiple people accessing one Github repository



Repository on Github

Clone, Pull, Push

Clone, Pull, Push

Person 1

Person 2

Checkout branch Stage, Commit

Checkout branch, Stage, Commit

Session 3: Team working

Conflict resolution

- Two collaborators edit the same text in the same file
- Person 1 Pushes their change to Github
- Person 2 Pulls down changes and gets a *Conflict*
- Person 2 cannot Push to Github until it is resolved
- Conflict must be edited and marked as *Resolved*
- Demonstrated in exercise
- Merge tools available to help

Session 3: Team working

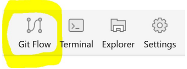
Feature branch workflow recap

- One repository on Github
- Team of collaborators
- Collaborators clone repository
- Feature branch created for all to work on
- Local work versioned with Stage and Commit
- Changes shared with Push, Pull and Conflict resolution
- Pull request completes work and merges branch

Session 3: Team working

Git Flow

- For your interest
- Another slightly different workflow is Git Flow
- It is available in SourceTree:



Session 3: Team working

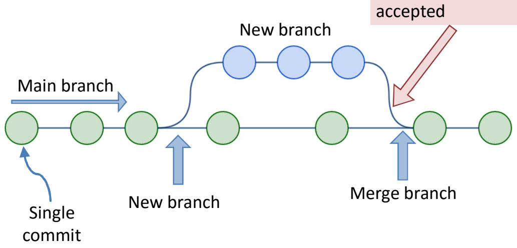
Pull Request

- Pull Request is a Quality Assurance step around a Merge
- Lets collaborators know a branch is ready to be merged into the main branch
- Changes can be reviewed, discussed, altered, etc
- Branch merged only when Pull Request is accepted

Session 3: Team working

Pull Request

Merge prevented until Pull Request accepted



New branch

Main branch

Single commit

New branch

Merge branch

Session 2: Team working

Exercise 3

https://nerc-ceh.github.io/version_control/exercise3

- Inviting collaborators to your repository
- Working as a team on a new branch
- Creating and resolving conflicts
- Issuing and accepting a pull request

More than just version control

- Integrates with issue management
  - Branches matched to issues
- Workflows triggered by commits, pull requests, etc
  - Automates unit and integration tests
  - Automates deployment to servers
- Open sourcing
  - Promotes collaboration and quality
  - Increases reputation

Version Control

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.