Distributed Version Control : `git` and GitHub

A *very* brief intro.

Oscar Branson

Department of Earth Sciences

Previous Experience?

What is Version Control?

- A tool for keeping track of changes to (plain text) files.
- Performed by a piece of software that keeps a record of changes to files.
- Commonly `git`, but also `svn`, `mercurial`, `cvs`, etc.

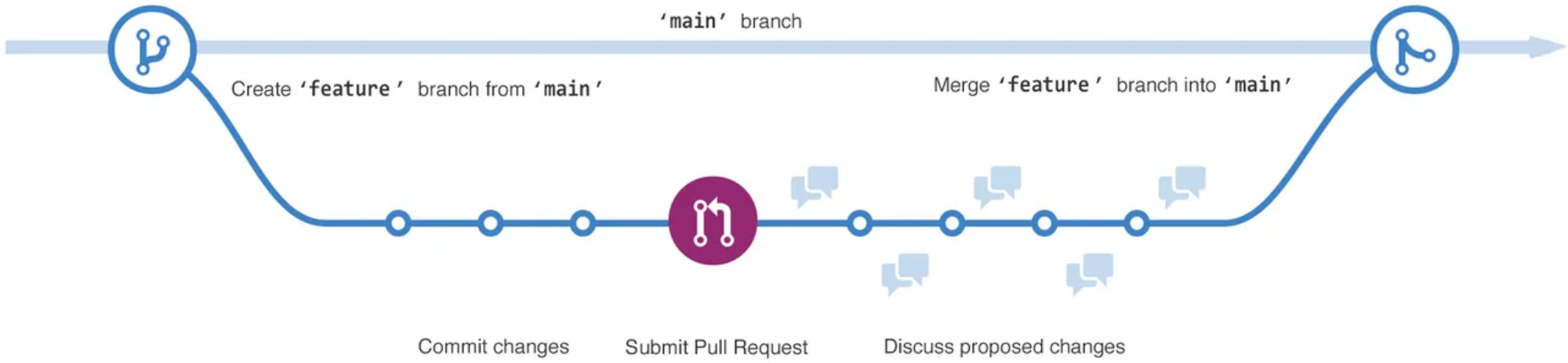
What is Distributed?

- Every machine has a complete history of the project.
- Synchronised with a central server, e.g. GitHub (or other).

Why do we want it?

- A complete, auditable history of all your code
- Who has done what?
- Collaboration on large, complex projects (e.g. Guided Team Challenge).
- 'Rewind' problematic changes
- Industry standard - transferable skill!
- FAIR data / 'open data' / 'open science' best practice

Key Concepts



Key Tools

- Command Line `git` interface
- GUI (e.g. GitHub Desktop)
- Integration into IDEs (e.g. VS Code, Sublime Text, etc.)

The rest of this session

1. **Admin** - get set up on GitHub
2. **Practice** - have a play around with a repository

Lots more excellent resources on the GitHub Skills Pages

1. Admin

- (Create a GitHub Account)
- Get set up on your computer (`git config`)
- Set up ssh access (add public key to your GitHub account)
- Add users `ai4er-cdt` organisation

2. Practice

- Fork the `github-intro` repository
- `git clone` your fork of the repository to your computer
- Make a new branch `git branch my-branch`
- Make some changes!
- Commit changes `git commit -m "a message describing the changes"`
- Push changes `git push`
- Make pull request - online
- Example of merging a pull request
- Issues

Forking

- You will not be able to modify repositories that you do not own.
- Forking creates a copy of a repository on GitHub that *you* own.
- You can make changes to your fork without affecting the original repository.

Cloning

- Cloning creates a copy of a repository on your computer that you can work on.

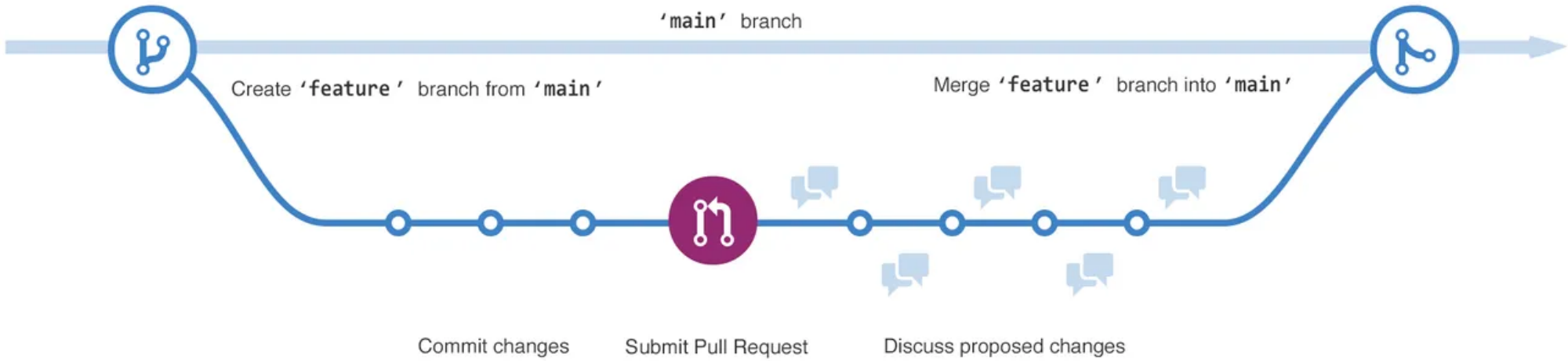
```
git clone
```

Branching

- Create a new `branch` of the repository that diverges from the main code, where you can develop and test your new idea/feature

```
git branch branch-name
```

```
git checkout branch-name
```



Making changes

- Go ahead and make some changes! Edit files, create files, delete files, etc.
- Run `git status` (or use your GUI/IDE) to see what has changed

Commit changes

- commits are the record of the changes you have made
- commit early and often
- write useful descriptions of changes

```
git add
```

```
git commit -m "description of changes"
```

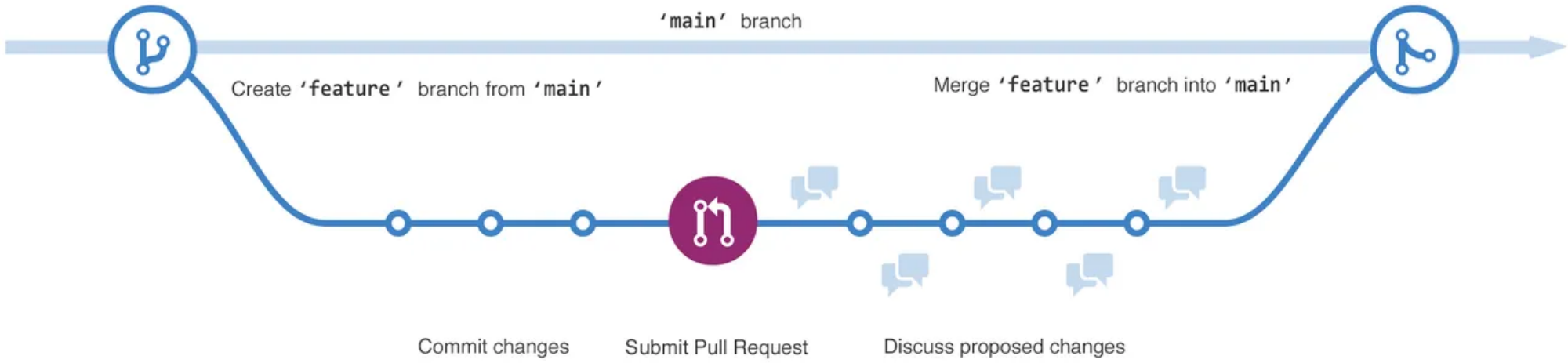
Push changes

- Pushing moves changes on your local computer to the remote repository on GitHub.
- Once pushed, your changes are 'live' in the repository; anyone can see them and download them.

```
git push
```

Pull request

- Merge the changes in your local fork back into the main codebase.
- You can also use pull requests within a single repository - you don't need to fork to use them.



Issues

- A place to discuss and track problems, ideas, and tasks for a project.

Next Steps

- GitHub Actions
  - Automated deployment
  - Testing
  - Code style/documentation checks
- GitHub Pages for project websites
- Make your own Repository for a new project
- GitHub CoPilot - AI code suggestions

Other Resources:

GitHub Docs, GitHub Skills