Bioinformatics session

A two-day workshop for bioinformaticians and molecular biologists with focus on the TSO500 pipeline in InPreD

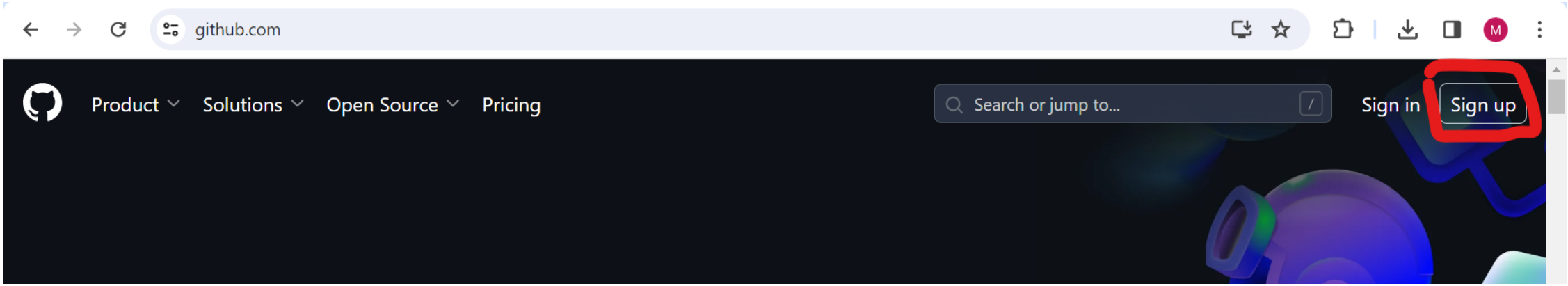
Overview

1. Setup
2. Development & Collaboration
3. Nextflow
4. tso500_nxf_workflow
5. Python

1. Setup

Create a GitHub account

- go to https://github.com/ and click on `Sign up`

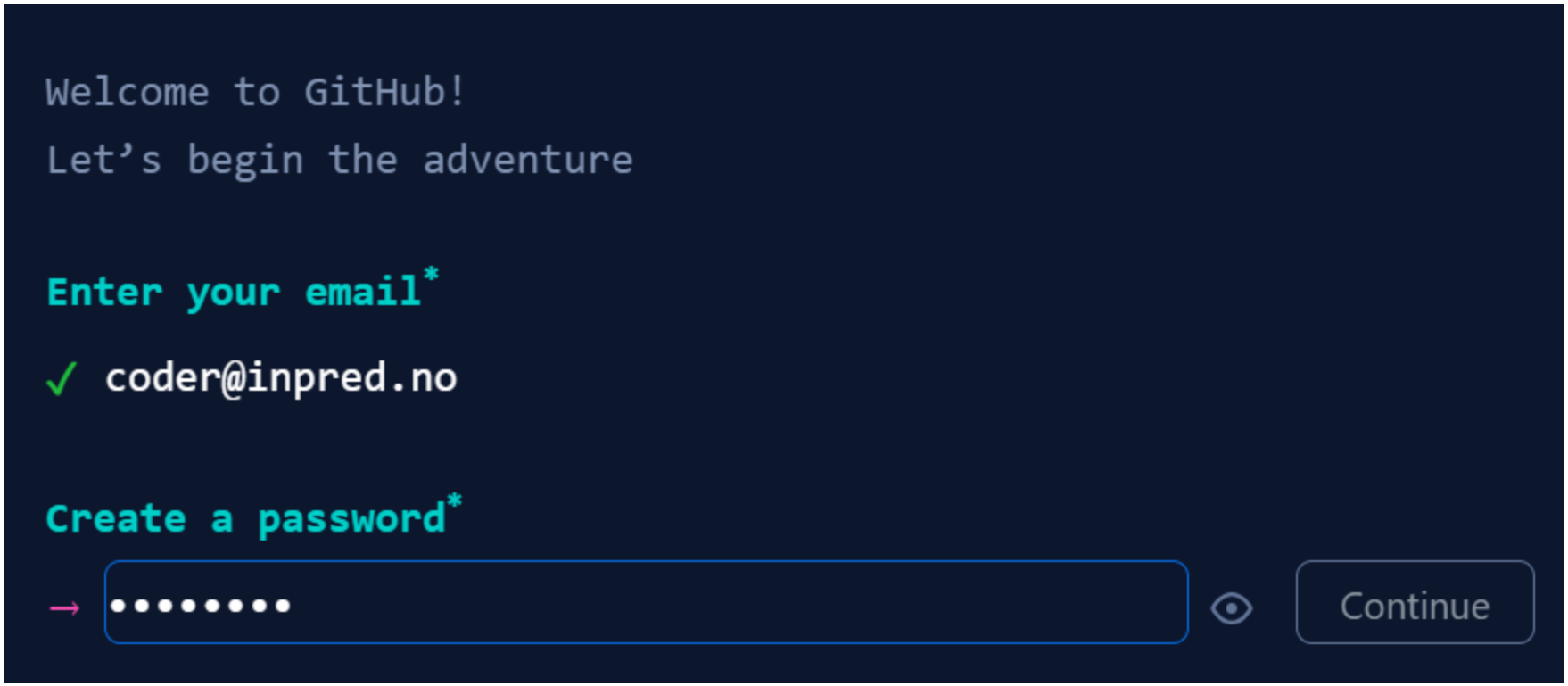


Create a GitHub account

- enter your email

Create a GitHub account

- set a password

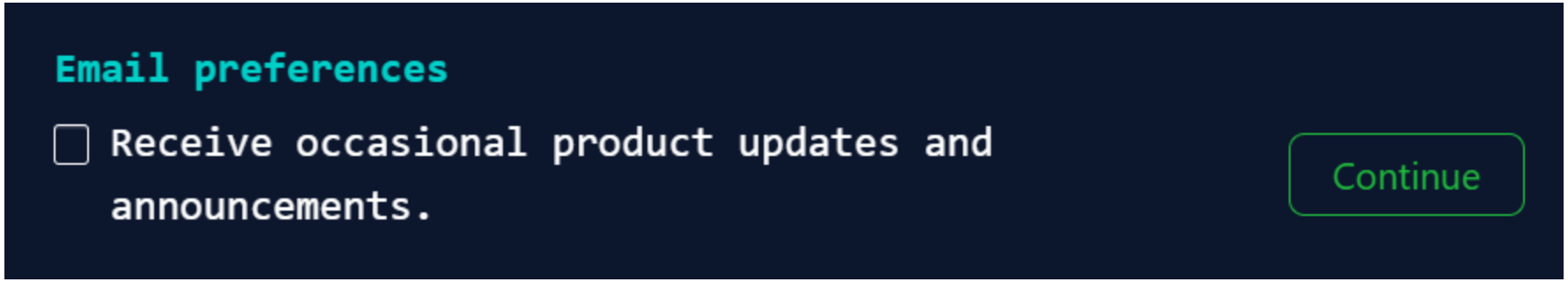


Create a GitHub account

- choose a username

Create a GitHub account

- choose email preferences

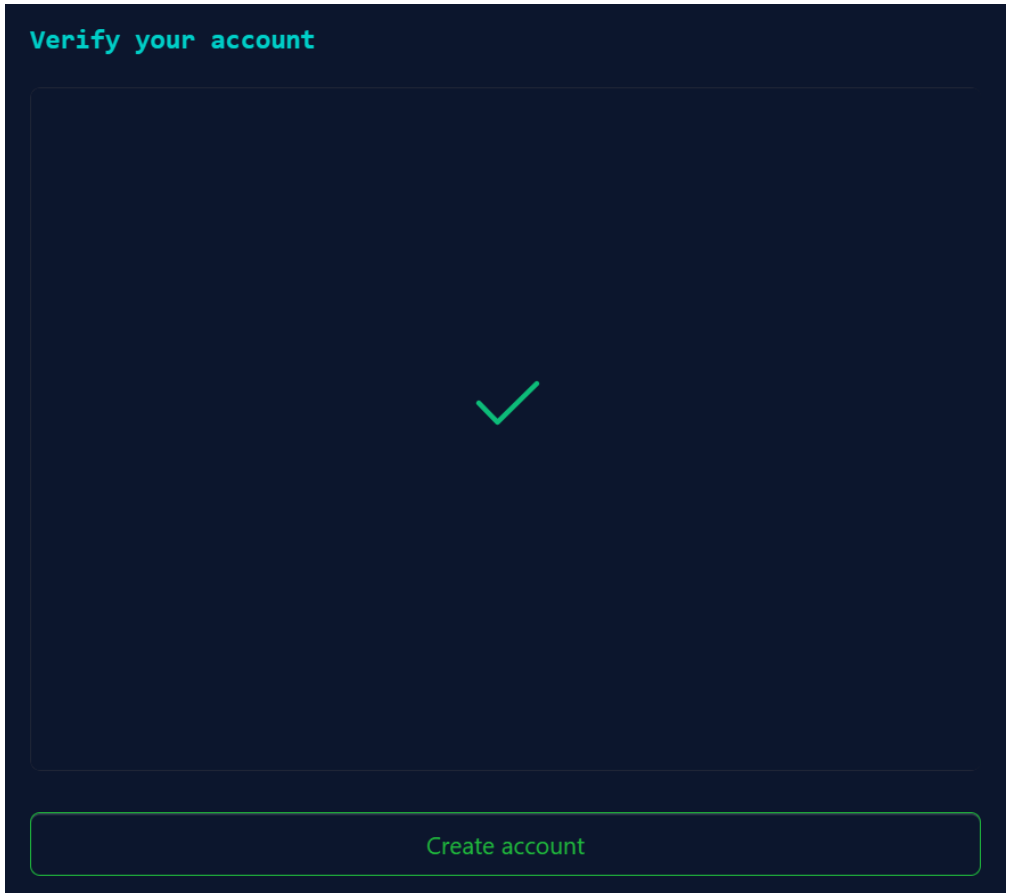


Create a GitHub account

- solve the puzzle

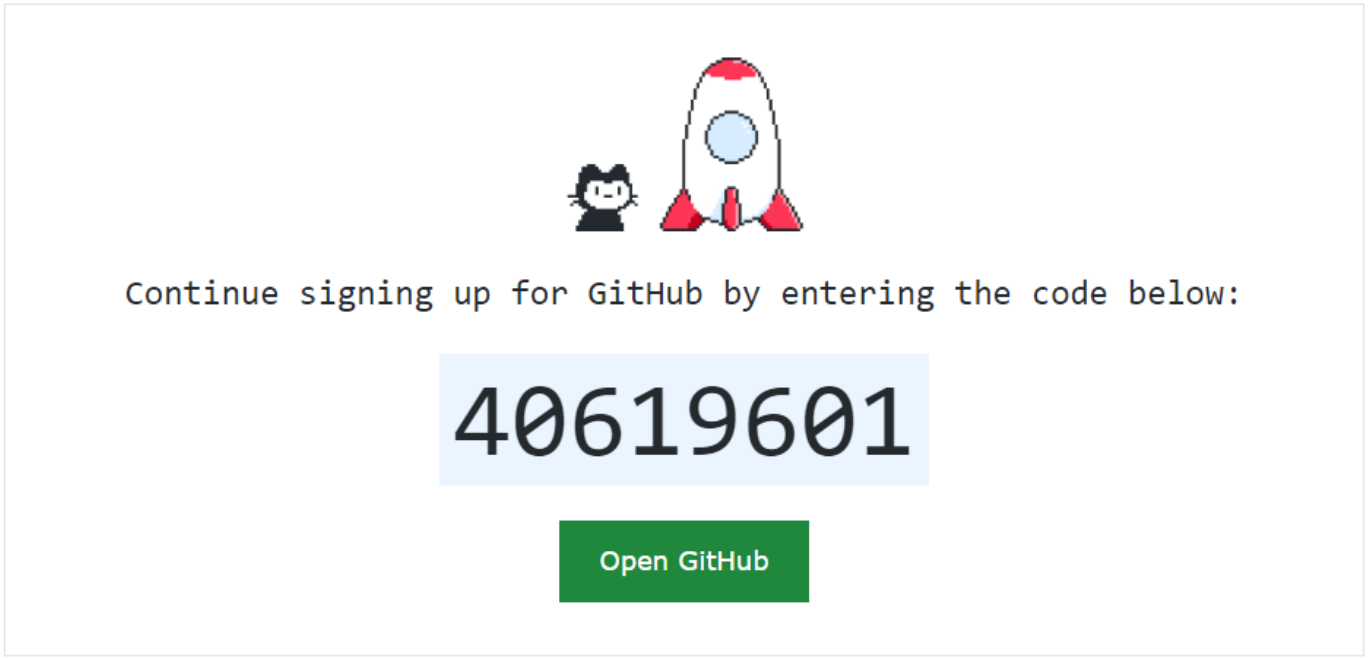
Create a GitHub account

- create your account



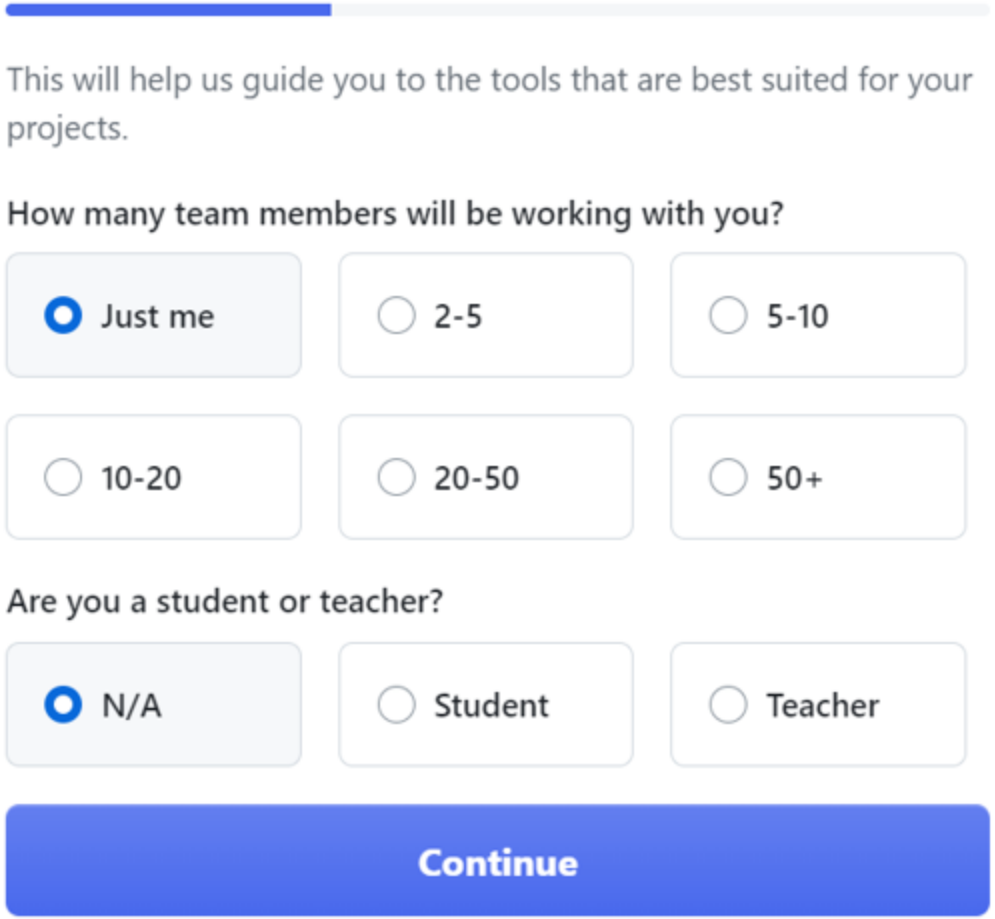
Create a GitHub account

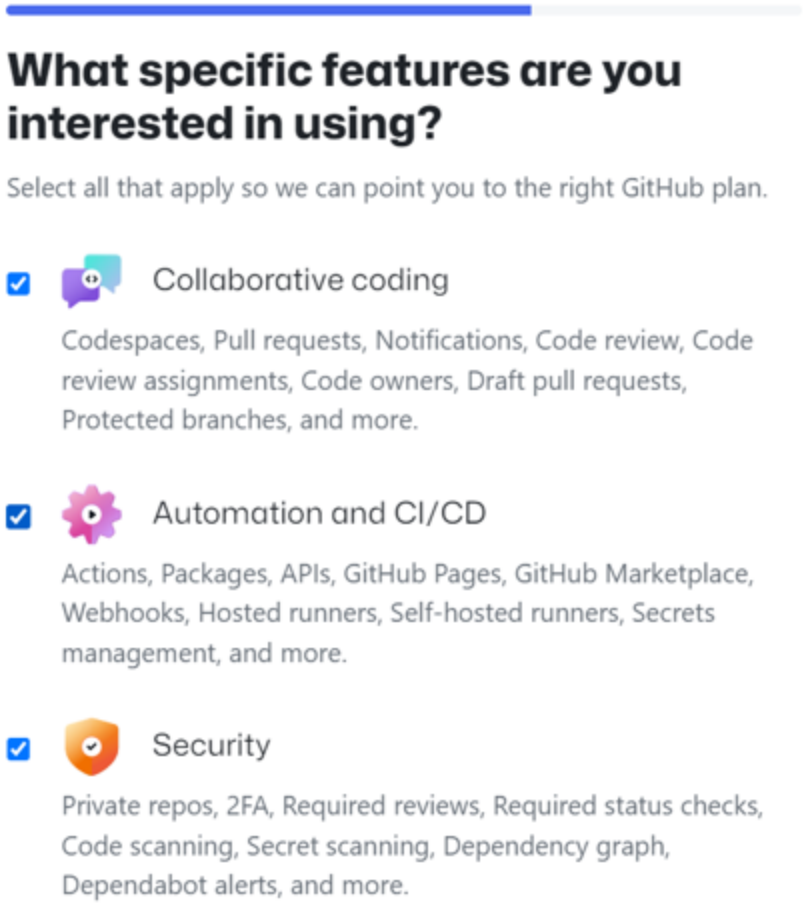
- find the activation code in the email you received



Create a GitHub account

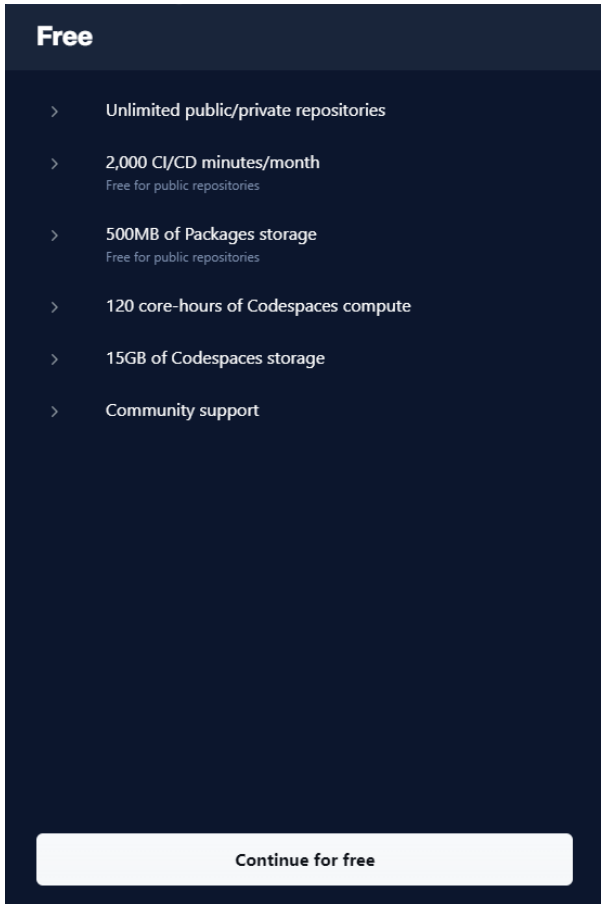
- select the desired options





Create a GitHub account

- choose the free plan



1. Setup

Be added to InPreD organisation at GitHub

1. Resources

- Getting started with your GitHub account

2. Development & Collaboration

Short `git` introduction

- distributed version control system
- tracks history of changes commited by different contributors
- every developer has full copy of project and its history

Short `git` introduction

Basic `git` commands

`git init` : initialises new git repository

`git clone <repository url>` : creates local copy of remote repository

`git add <file/s>` : stage new or changed files (anything that should be committed to the repository)

`git commit -m "feat: my new feature"` : commit changes to the repository

Basic `git` commands

commit message conventions

`<type>[optional scope]: <description>`

- `feat` : new feature
- `fix` : patching bug
- `refactor` : code change that neither is neither feat nor fix
- `build` : build system related changes
- `perf` : improving performance

commit message conventions

`<type>[optional scope]: <description>`

- `chore`: code unrelated changes, e.g. dependencies
- `style`: code change that does not change meaning
- `test`: changes to tests
- `docs`: adding/updating documentation
- `ci`: continuous integration, e.g. github actions

Basic `git` commands

`git status` : overview over untracked, modified and staged changes

`git branch` : show local branches

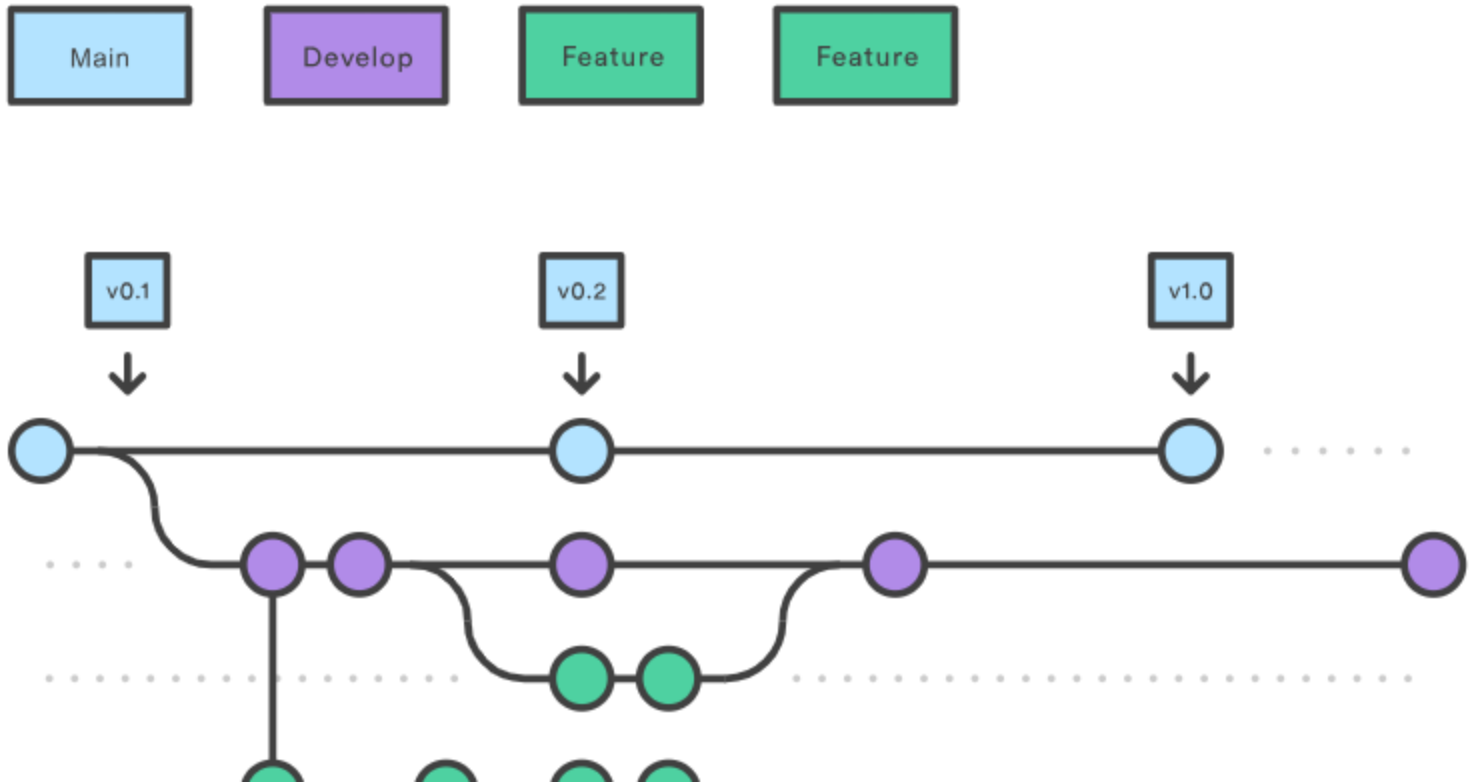
`git merge` : merge branches

`git pull` : load changes from remote counterpart

`git push` : upload changes to remote counterpart

Branching model: Gitflow

- start with two branches to record project history: `main` and `develop`
- each new feature resides in its own branch (feature branch)
- feature branch is generally created off latest `develop` commit
- upon feature completion, feature branch is merged into `develop`



- github actions (linting, testing, building)
- pull requests (best practice)
- release and semantic versioning
- licensing

2. Resources

- About Git
- Gitflow workflow

3. Nextflow

- general (install, best practice)
- nf-core template
- stubbing

4. `tso500_nxf_workflow`

- status
- demonstration

5. Python

- general (best practice, cli)
- unit testing (pytest)

Resources

- github actions
- nf-core
- pytest unittesting
- semantic versioning