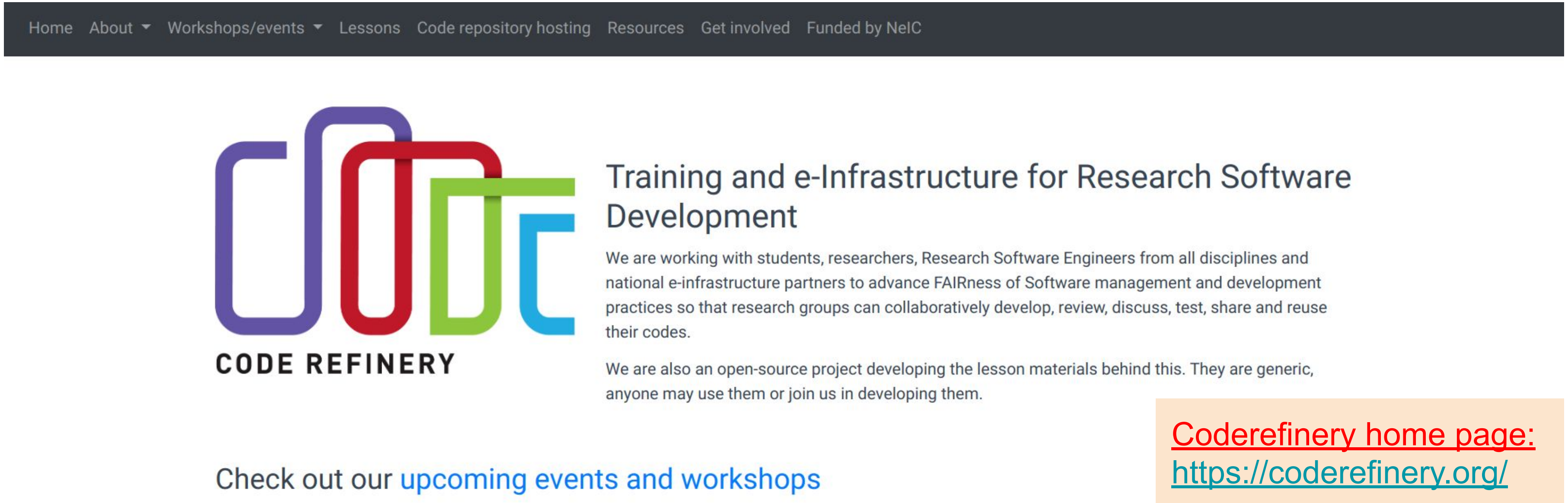
Forking workflow

NorESM User Workshop 2020

Forking workflow

Based on Coderefinery tutorial :
https://coderefinery.github.io/git-collaborative/03-distributed/



Training and e-Infrastructure for Research Software Development

We are working with students, researchers, Research Software Engineers from all disciplines and national e-infrastructure partners to advance FAIRness of Software management and development practices so that research groups can collaboratively develop, review, discuss, test, share and reuse their codes.

We are also an open-source project developing the lesson materials behind this. They are generic, anyone may use them or join us in developing them.

Check out our upcoming events and workshops

Coderefinery home page:
https://coderefinery.org/

Forking layout

A forking workflow is one where developers mainly contribute code by pull requests from personal forks of the main (upstream) repository.

When cloning a personal fork to a computer, the fork becomes the default remote (origin) repository for the clone

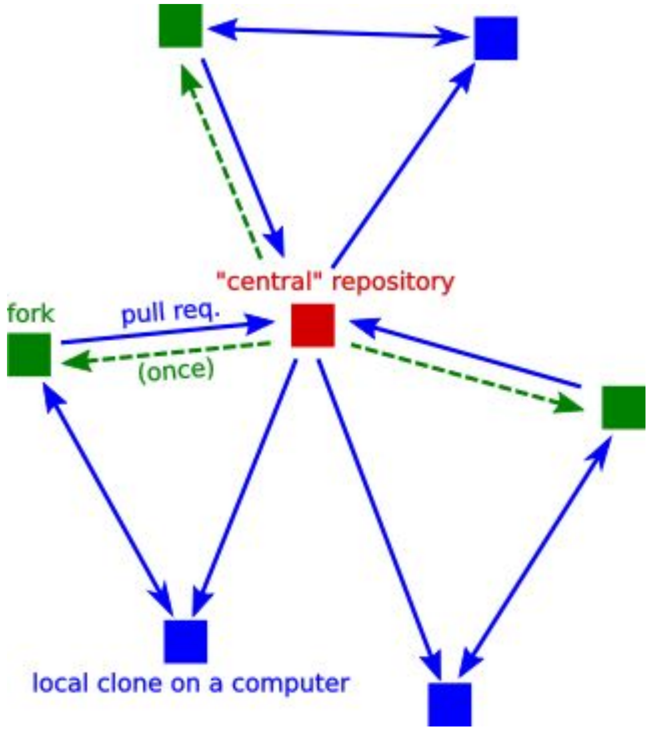
The terms “origin” and “upstream” are conventions, they have no special meaning for the git system itself.

In local clone, find the “origin” repository

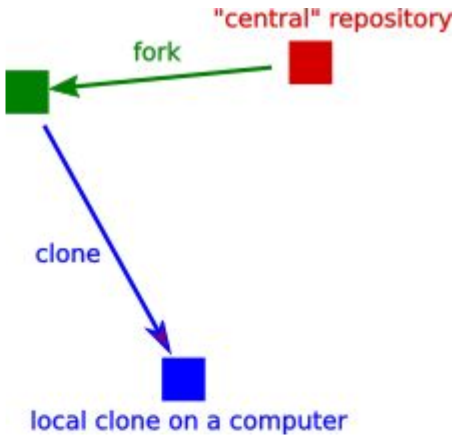
```
$ git remote -v
```

The central repository can be added as an additional remote

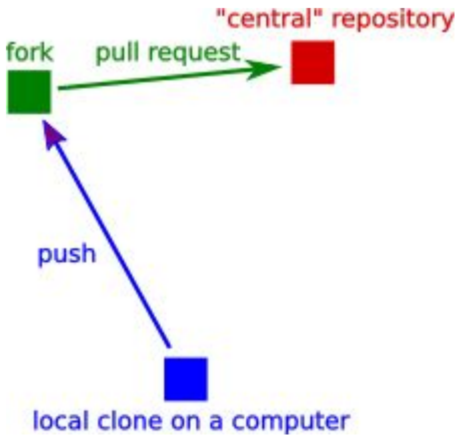
```
$ git remote add upstream https://github.com/NorESMhub/NorESM.git
```



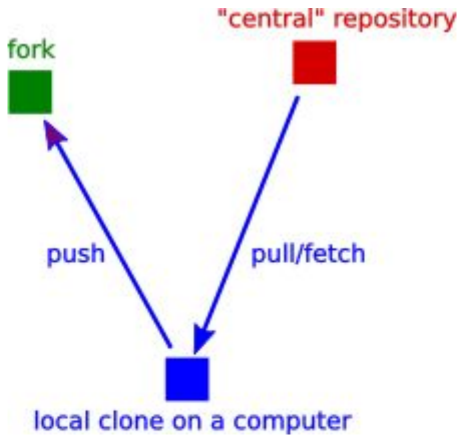
Forking layout



Fork layout setup; initial clone



Contribute changes to main repository



Update your clone and fork with latest changes from main repository

Forking workflow - good practices

- Keep branches inherited from “upstream” clean, develop your changes in feature branches
  - Try to avoid introducing extra “merge commits” in the repo history
- Commands that change commit history (e.g. “git reset”, “git rebase”) are fine in personal forks and clones. Do not change commit history in repos that are shared with others.
- Create pull requests early and often, rather than one big pull request when you are all done.
  - Avoids having your feature branch lagging far behind the main repository
  - If you are unsure about something, create a “draft” pull request and invite others to review and comment

Links to more information

Coderefinery lessons:

- Git intro : https://coderefinery.github.io/git-intro/
- git for collaborative work : https://coderefinery.github.io/git-collaborative/
- gitHub help pages : https://docs.github.com/en/github

DEMO :

Spacemacs distribution of Emacs, using the magit interface