Science & Technology
Facilities Council

An introduction to using Git: the stupid content tracker

"*I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git.*"

Linus Torvalds

Config settings

- Tell git your name and email address, these are added to commit messages:

  git config --global user.name “Your Name”

  git config --global user.email you@somewhere.ac.uk

- Make the CLI a bit prettier

  git config --global color.ui auto

Heirarchy

- Working copy
  - The copy of the code that you're working on
- Index (staging area)
  - List of all changes to go into the next commit
- Local repository
- Remote repository (CCPForge)

Local and remote repositories

- You need these commands to sync your local repository with CCPForge. They assume you are using ‘remote-tracking’ branches (I’ll come onto that later), if not you have to specify a source and destination branch.

- To download updates from CCPForge

  git pull

- To upload changes to CCPForge

  git push

Useful commands

- View current status of working copy and index

  git status

- View changes

  git diff

- View commit history

  git log

Adding or editing a file

- Add the change to the index (staging area)

  git add <*file*>

- Commit the changes in the index to the local repository

  git commit

- Upload the changes in the local repository to CCPForge

  git push

Deleting a file

- Delete the file and remove it from the index (staging area)

  git rm <*file*>

- Commit the changes in the index to the local repository

  git commit

- Upload the changes in the local repository to CCPForge

  git push

Renaming a file

- Move the file, add the new file to the index and remove the old file from the index

  git mv <*src*> <*dest*>

- Commit the changes in the index to the local repository

  git commit

- Upload the changes in the local repository to CCPForge

  git push

That looks like too much work

- Commit all changes to the local repository, bypassing the index (you still have to ‘git add’ new files first)

  ```
  git commit -a
  ```

- Upload the changes in the local repository to CCPForge

  ```
  git push
  ```

Oops, I made a mistake!

- Undo staged changes
  - git reset <*file*>
- Undo committed changes
  - git revert <*commit*>

Branches

- Branches are where the development of the code diverges, they are used to develop different features on different branches.
- When a feature is stable, its branch is ‘merged’ back into the main branch.
- In git, you have ‘local branches’ that exist in your local repository, and ‘remote branches’ that exist in a remote repository (CCPForge).

Listing and changing branches

- To list local branches:

  git branch

- To list remote branches (those on CCPForge):

  git branch -r

- To change branch:

  git checkout <*branch-name*>

Remote-tracking branches

- A remote-tracking branch is a local branch that is automatically updated when you do a ‘git pull’ or a ‘git push’ without any arguments.

- When creating a local branch by checking out a remote branch, it is a remote-tracking branch by default.

Branching

- Checkout the source branch

  git checkout <*src-branch*>

- Create the new branch locally

  git branch <*branch*> OR git checkout -b <*branch*>

- Then upload it to CCPForge, the ‘-u’ makes the local branch a remote-tracking branch

  git push -u origin <*branch*>

Merging

- Checkout the branch you want to merge into

  git checkout *<branch1>*

- Merge the change from the other branch into it

  git merge *<branch2>*

- You may have to resolve conflicts

- Then upload that change to CCPForge

  git push

Deleting a branch

- If you're finished with a branch, you may want to delete the local copy
  - git branch -d <*branch*>

- If it's on CCPForge, you can always check it out again.

Further help

- There is a git-testing project on CCPForge. You can join this project and use it as a sandbox.
- Documentation on the git website
- The git man pages are well-detailed, and have examples