Stony Brook University

FiTQun development on Github

Cristóvão Vilela

FiTQun! Workshop
TRIUMF

August 18 2015

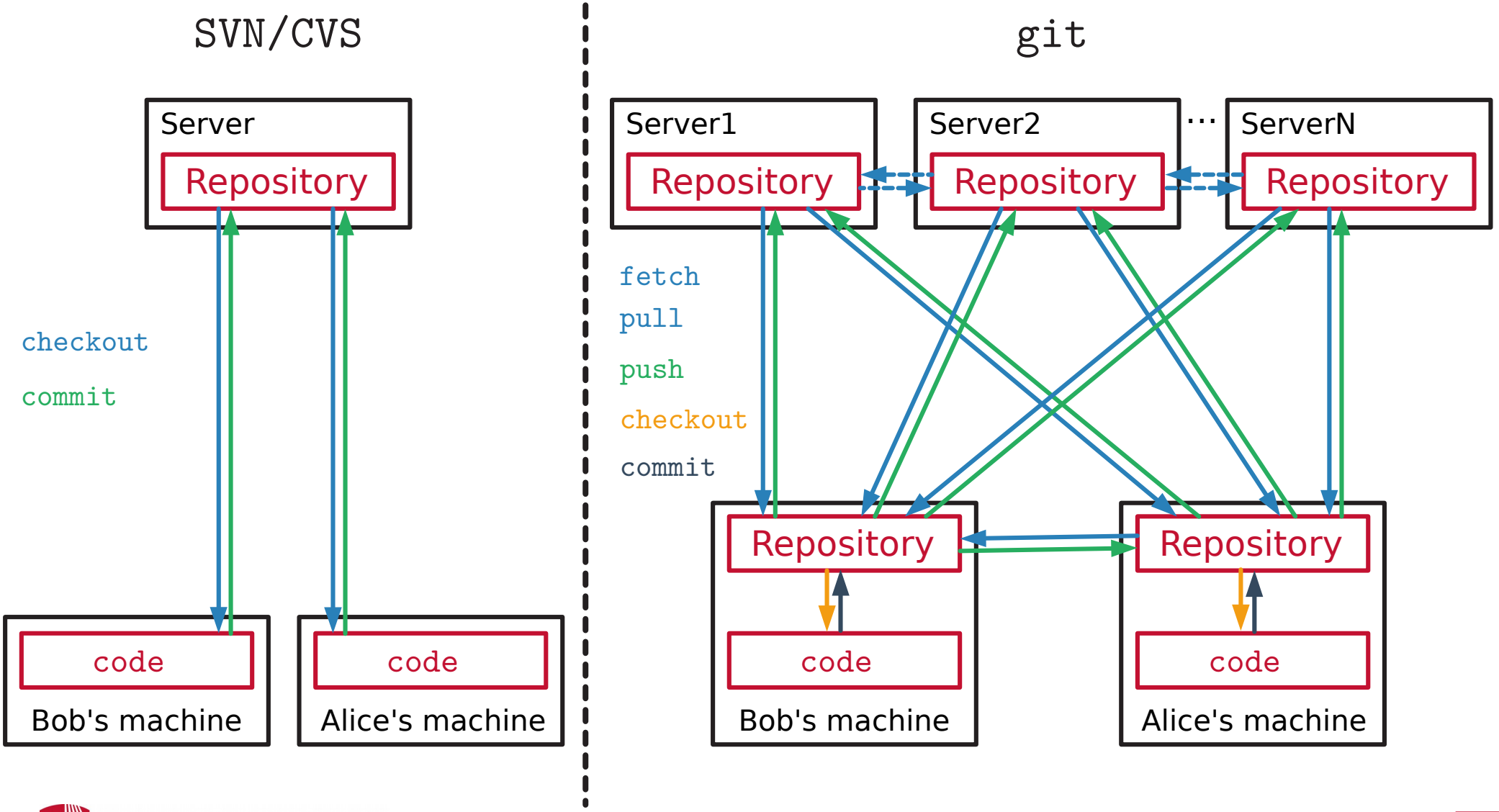
Introduction

- FiTQun is on Github
  - There is a FiTQun private organisation on Github
    - Code is hidden from public, membership by invitation
    - Admins: Mike Wilking, Shimpei Tobayama, CV
  - We have three repositories
    - fiTQun/fiTQun
    - fiTQun/Utilities
    - fiTQun/Macros
  - Code has been updated with SKanada repository up to Aug 4
  - Use `git` for version control
  - Use Github tools for collaborative development
    - Use the same conventions as the WCSim repository

De-centralised version control: `git`

- Unlike SVN/CVS there isn't a default "central" repository/server in git – it is de-centralised
- All developers have their own repository in their *local* machine, in addition to any other *remote* repositories
- To get the code, developers "clone" the repository they want to work in, and this builds a local copy of the entire repository
- Developers then edit the code and commit changes to the local repository
- Local commits can be uploaded to the server by "pushing" the local changes

git vs SVN/CVS



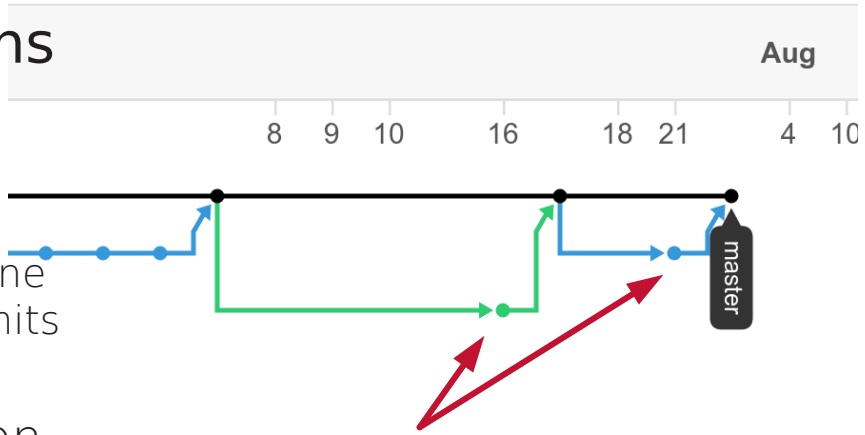
A bit more on `git`: some commands

- Creating a repository
  - `git init [project name]`
    - Creates a new local repository with the specified name
  - `git clone [url]`
    - Downloads a project and its entire version history
- Make Changes
  - `git add [file]`
    - Snapshots the file in preparation for versioning
  - `git commit -m "[descriptive message]"`
    - Records file snapshots permanently in version history
  - `git status`
    - Lists all new or modified files to be commited
- Managing Repositories
  - `git remote -v`
    - Lists all remote repositories available
  - `git remote add [name] [remote address]`
    - Adds a remote repository to the list of remotes under the chosen name
- Group changes
  - `git branch`
    - Lists all local branches in the current repository
  - `git checkout [branch-name]`
    - Switches to the specified branch and updates the working directory
  - `git checkout -b [branch-name]`
    - Switches to a new branch
  - `git branch -d [branch-name]`
    - Deletes the specified branch
- Synchronize changes
  - `git fetch [remote]`
    - Downloads all history from the remote repository
  - `git merge [remote]/[branch]`
    - Combines remote's branch into current local branch
  - `git push [remote] [branch]`
    - Uploads all local branch commits to remote repository
  - `git pull`
    - Downloads remote history and incorporates changes

More in
https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf

A bit more on `git`: branches and remotes

- Git branches are essentially collections of commits
  - Very useful for developing features
    - Use one branch to collect all commits related to one particular feature, and then merge all these commits at once into a main branch
  - The general philosophy is to always work on "feature" branches and then merge these into the "develop" branch
    - more on that later



Example of two "single commit" feature branches being created and merged into the master branch.

- There is no "central repository" in git
  - We need to tell git ***which*** repository we want to push to or fetch/merge/pull from
  - A **remote** is a "bookmark" for some repository (with a shared history)
    - It is usually in a remote server (accessed through ssh or https) but it can also be local
    - When a repository is cloned, the path to the original will be automatically added to the remote list under the name "origin"

Advantages of de-centralised versioning

- `git` is clearly a bit more complicated to use than `SVN`/`CVS` style repositories, but has several advantages:
  - Developers work on their own clones (or "forks") of the repository, staying out of each other's way when they want to
    - But merging from each other when they want to
  - An official repository is defined, and developers can request that their finished work is merged (or pulled) into the official repository
    - Official repository administrators can moderate contributions
  - Developers can share their work-in-progress between each other without interfering with the official repository
  - Code can be committed offline, and pushed once a connection is available

Organising git projects with Github

- Github is a repository hosting service
  - With a collection of very useful online tools
  - Every user has their own collection of repositories
  - Github's “fork” function allows users to clone repositories in some other user's or organisation's area into their own
    - E.g., fiTQun/fiTQun → 

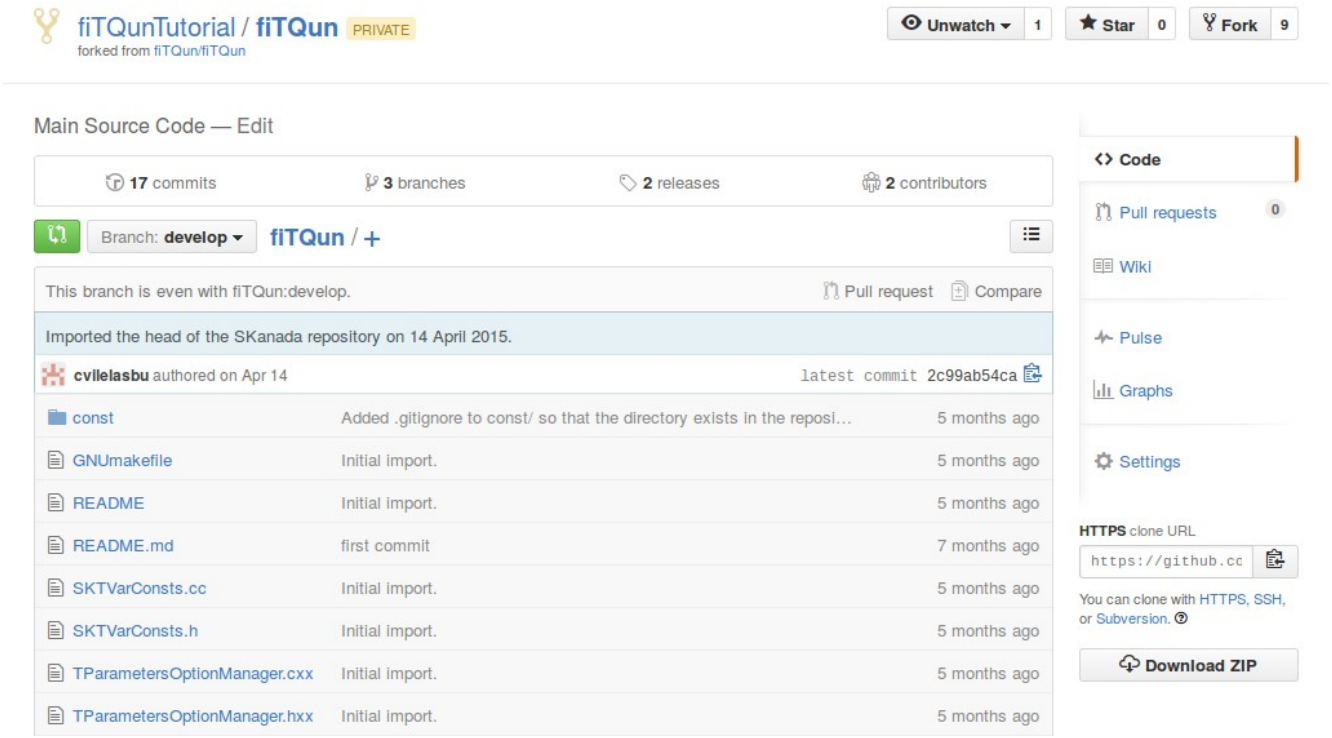
→ cvilelasbu/fiTQun
  - Another very useful tool is the 

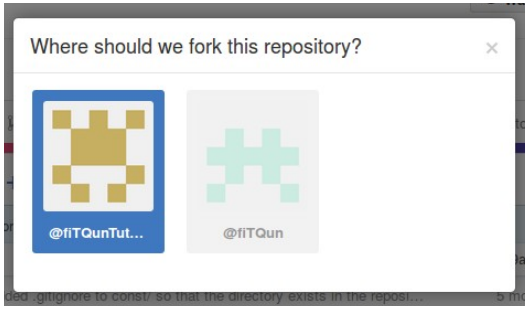
    - This allows developers to request the repository administrators to pull branches they have developed in their own repositories into the main/official repository
      - This creates a discussion page where an article can be written and comments can be posted
        - E.g. validation plots, justification for the changes, discussion, …
  - In addition Github has Wiki and Issue tracking functionalities

Tutorial: Forking the repository

1) The first step is to fork the fiTQun/fiTQun repository into your own Github area:

- Press the Fork 8 button and select your username:

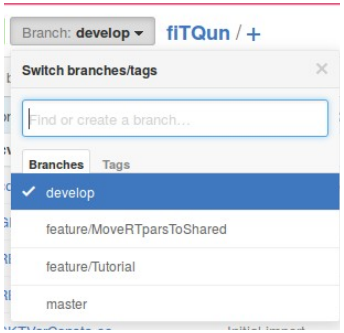




- Now you have your own fiTQun repository
- Use the clone URL to clone the repository in your local machine

Tutorial: Creating a feature branch

- First, clone the repository in your machine
  - `git clone https://github.com/fiTQunTutorial/fiTQun.git`
- You now have a copy of the fiTQun repository
- To check what branch we are in, run **`git branch`**:
  - `* develop`
    - This is the default branch
- Create a new branch and move into it:
  - `git checkout -b feature/Tutorial`
    - `Switched to a new branch 'feature/Tutorial'`
- Now make it count, push the branch up to github.com
  - `git push origin feature/Tutorial`



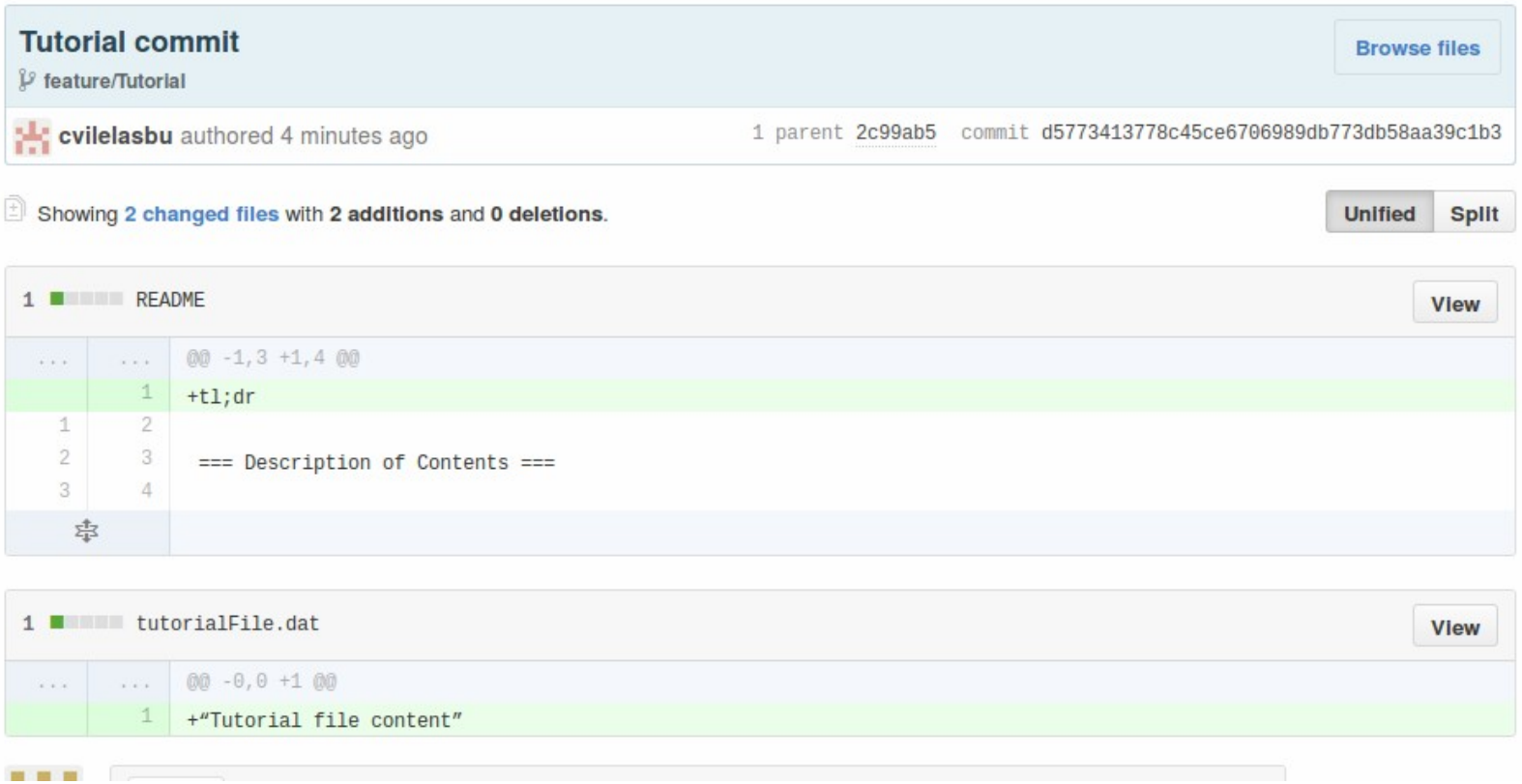
Tutorial: Committing changes

- Let's create a new file:
  - `echo "Tutorial file content" > tutorialFile.dat`
- And add it to the list of tracked files
  - `git add tutorialFile.dat`
- Let's also modify an existing file:
  - `sed -i '1itl;dr' README`
  - `git add README`
- Now running `git status` will show two files ready for commit:

```
modified:   README
new file:   tutorialFile.dat
```

  - `git commit tutorialFile.dat README -m 'Tutorial commit'`
    - Can also do '-a' for all files, but not recommended
- And again push the new commits up:
  - `git push origin feature/Tutorial`

Tutorial: Committing changes



Tutorial: Aside

- Let's say Mike has been working on a feature that would be useful for me to have now
  - Add Mike's repository to my remotes:
    - `git remote add mike https://github.com/wilking/fiTQun.git`
  - You can list the repositories you have already added with
    - `git remote -v`

```
mike    https://github.com/wilking/fiTQun.git (fetch)
mike    https://github.com/wilking/fiTQun.git (push)
origin  https://github.com/fiTQunTutorial/fiTQun.git (fetch)
origin  https://github.com/fiTQunTutorial/fiTQun.git (push)
```

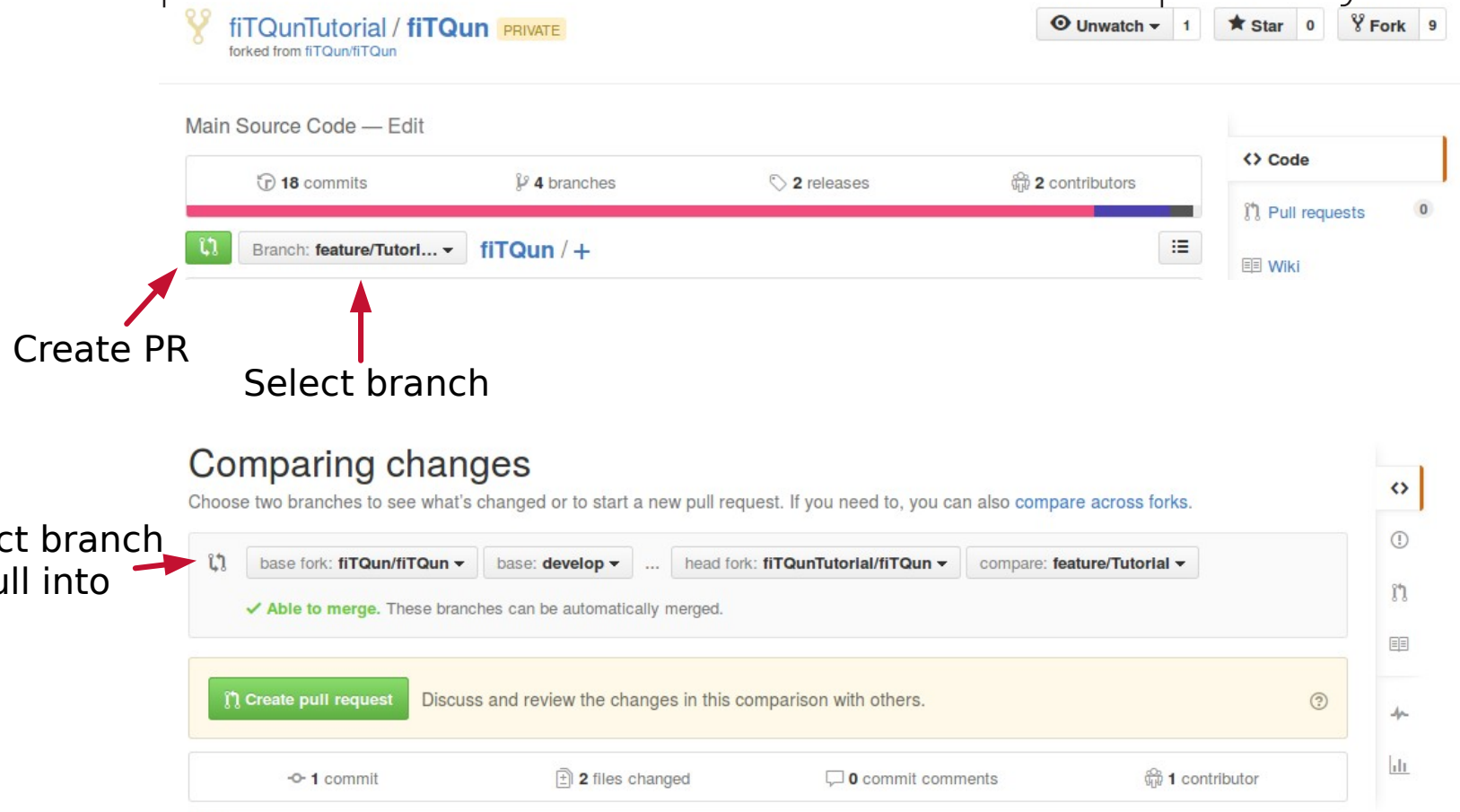
  - And to download an image of Mike's repository
    - `git fetch mike`
  - And now if you list the branches you'll see Mike's branches too
    - `git branch -a`

```
  develop
* feature/Tutorial
  remotes/mike/feature/AfewMoreParsToShared
  remotes/mike/feature/PDK_muGamma
  remotes/mike/feature/anotherQuickFix
  remotes/mike/feature/fixFilename
  remotes/mike/master
  remotes/origin/HEAD -> origin/develop
  remotes/origin/develop
  remotes/origin/feature/MoveRTparsToShared
  remotes/origin/feature/Tutorial
  remotes/origin/master
```

  - So if you wanted to merge in quick fix:
    - `git merge mike/feature/anotherQuickFix`

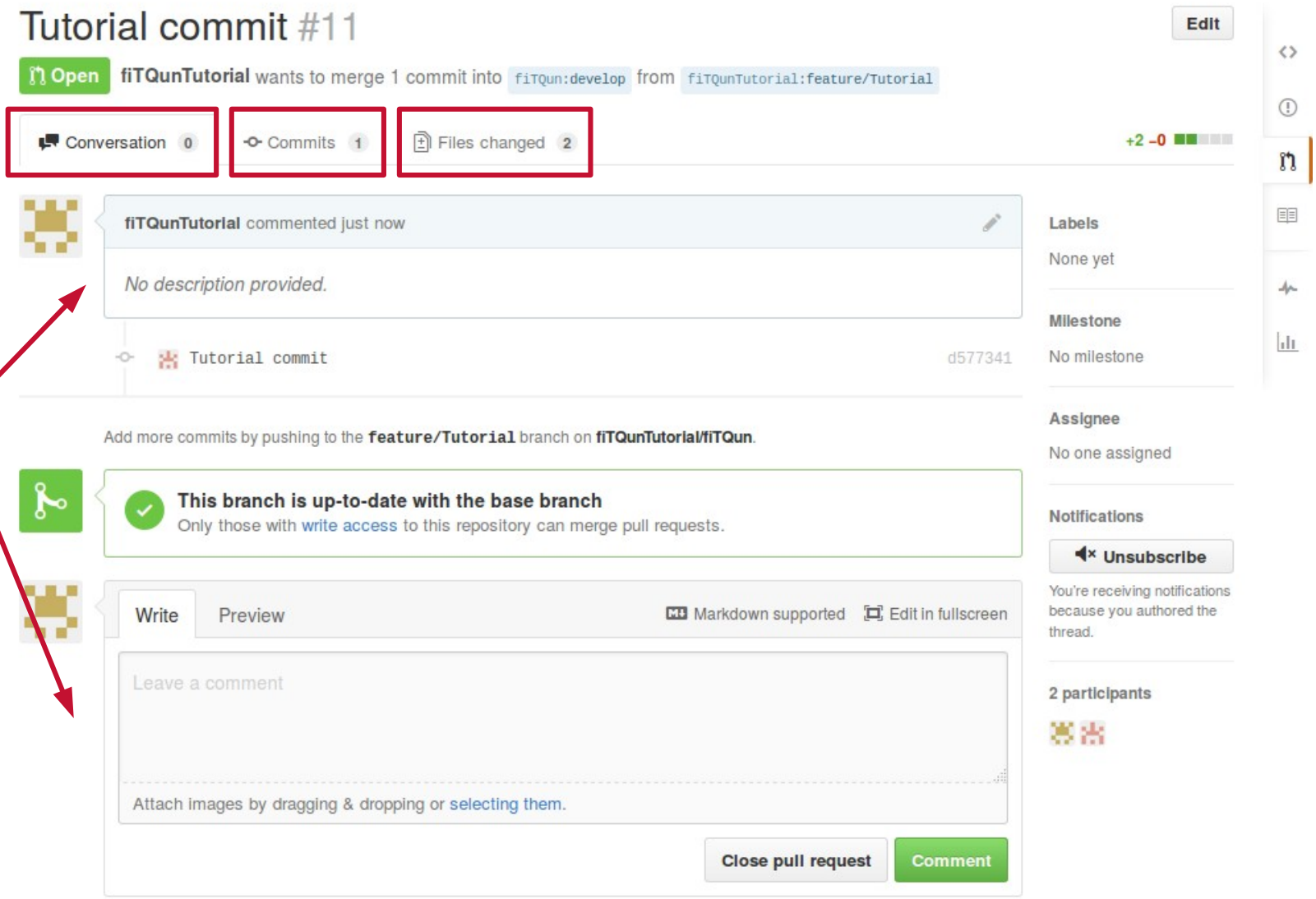
Tutorial: Creating a pull request

- Now that we have pushed our completed "feature" up to github, we can generate a pull request, for the feature to be incorporated into a branch in the official repository



Tutorial: Pull requests

Discuss code changes here. You can add validation plots, etc



Tutorial: Cleaning up

- Once the pull request has been accepted and merge into the main branch, the feature branch is not needed anymore
  - Delete the branch in your local repository
    - `git branch -d feature/fiTQunTutorial`
  - Delete the branch in the remote repository (weird syntax)
    - `git push origin :feature/fiTQunTutorial`
  - Update your local repository

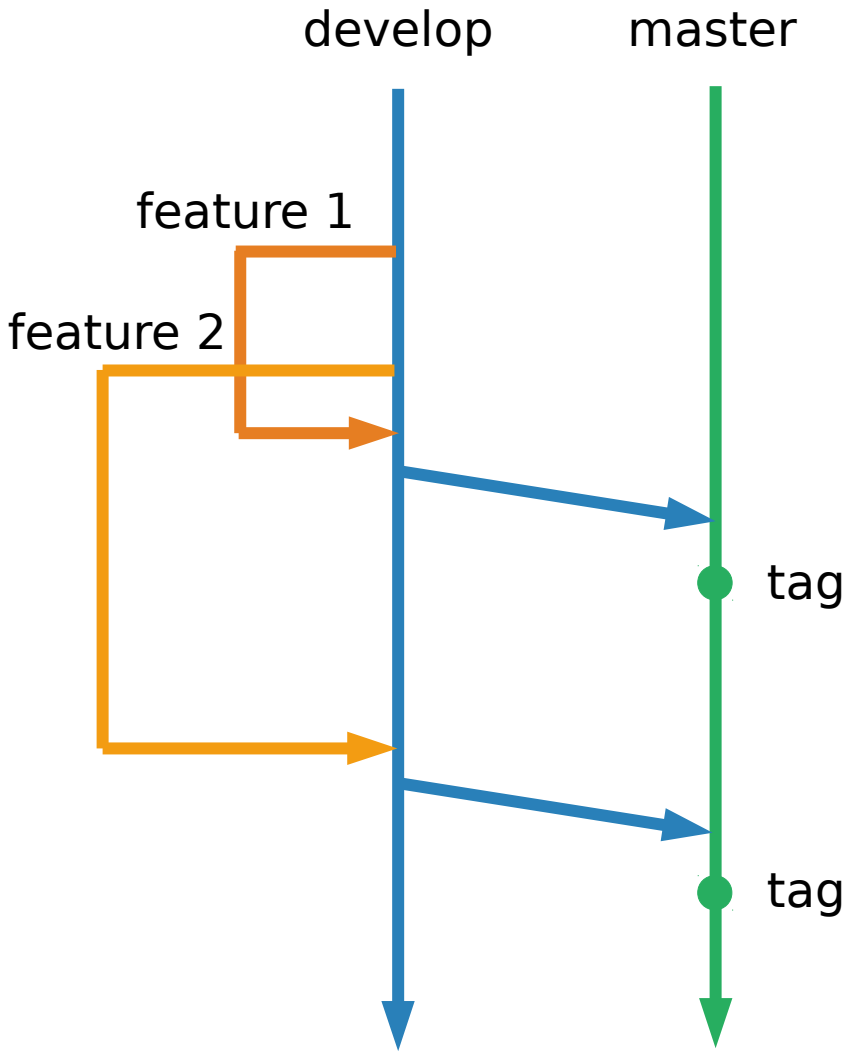
    ```
    git remote add upstream https://github.com/fiTQun/fiTQun.git
    ```

    - `git fetch upstream`
    - `git merge upstream/develop`
    - `git push origin develop`

- All done!

Organization of fiTQun repository

- There are two permanent branches, with the usual names
  - master: code in this branch should always be stable/validated
    - official tags are made in this branch
  - develop: this is where developers pull their completed features into
    - code in develop might be buggy
- Features are developed in their own branches, which are deleted once the respective pull request is merged



Summary

- FiTQun is on github
- Git + github development feels weird and cumbersome at first
- But it's great and you'll get used to it quickly!
- Please email one of us your github username so we can add you to the fiTQun organisation
  - if you haven't already
- Start developing in your own feature branch!