Version control system (VCS)

- Remember that you are required to **keep** a process-log-book of the whole development
  - solutions with just one commit or with **incomplete** process-log-book (where it is not possible to follow the progress of the project) will not be accepted
  - it does not matter that mistakes are done while developing. What matters is that we can **track** how you solved the problems
- A VCS is ideal to maintain a process-log-book

Version control system (VCS)

- Currently there many VCSs available: svn, git, mercurial, cvs, ...
  - for the class project (for easying the supervision) you are required to use git and the services provided by bitbucket
  - you are also required to use only **private** repositories
- In this guide we will use git as well
  - we will also use bitbucket which offers free (multiuser) private git repositories
  - there are many other free repositories services such as Google code, Sourceforge, etc.

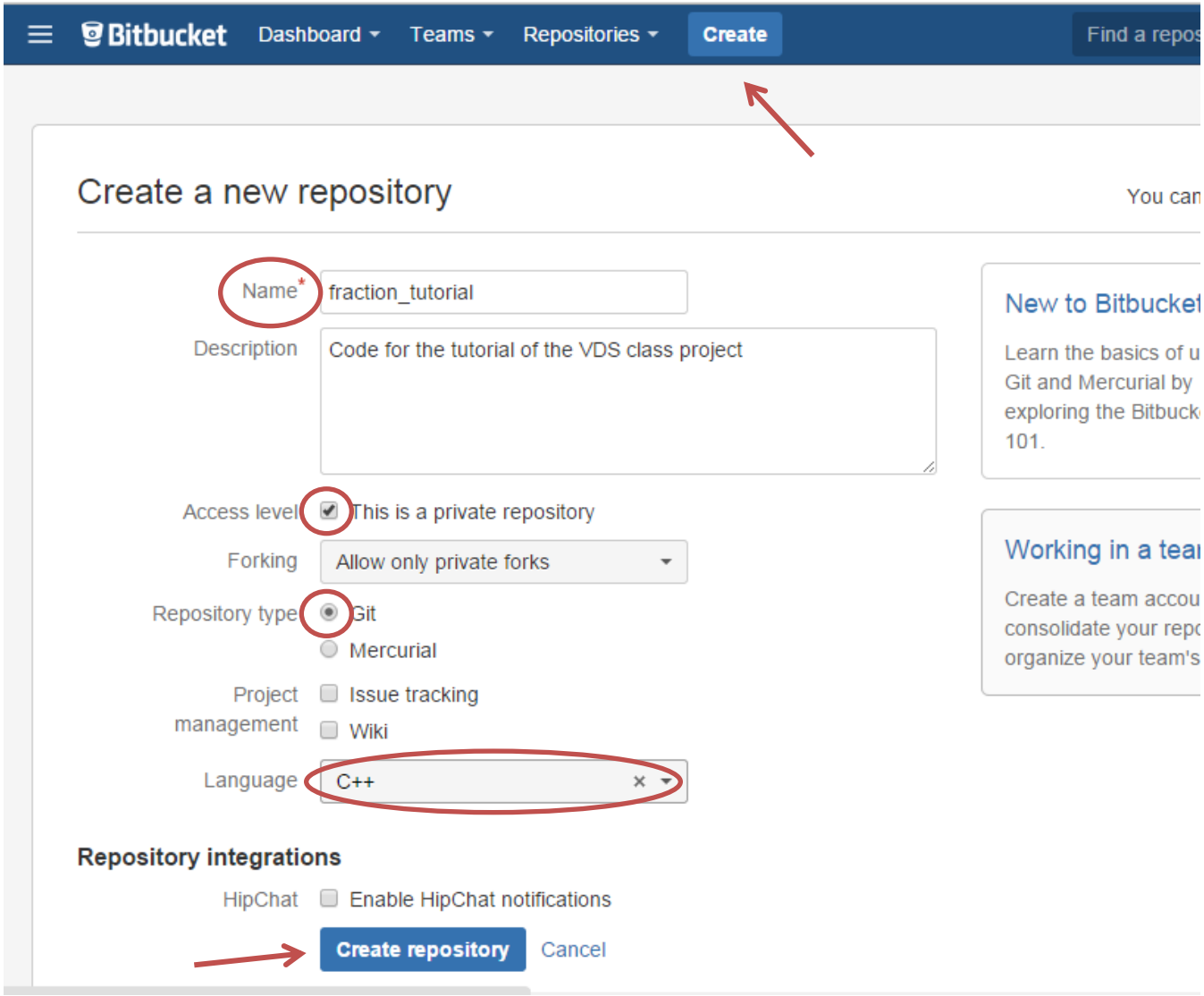
Git: basic documentation

- If you are new to git we recommend you to have a look to the following introductory material:
  - a very basic guide http://rogerdudler.github.io/git-guide/
  - a more elaborated tutorial for beginners: https://www.atlassian.com/git/tutorials
  - a good graphical tutorial: http://marklodato.github.io/visual-git-guide/index-en.html

Git: creating the central repository

- Create a new repository in bitbucket for that you should:
  - sign up: https://bitbucket.org/
  - log in in your account and create new **private** repository called *fraction_tutorial*
  - see next slide …

Git: creating the central repository



Git tips (1)

- Try to save in the git repositories only text files: .cpp, .hpp, scons scripts, doxygen configuration files, makefiles, …
  - do not save binary files, object files (*.o), executables and libraries
- Before starting configure git in your account (set user name, email, text editor, …). For this follow the steps presented in: https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config

Git: creating working directory

- Now that you have created the central repository it is time to create your <u>working directory</u>.
  - go to the directory *work/fraction_tutorial* which is available on your account. On the linux console type:
    cd work/fraction_tutorial
  - This directory has a template of the code (c++ code, scons scripts, makefiles and doxygen template)
  - Note the directory structure:

    /doc doxygen documentation

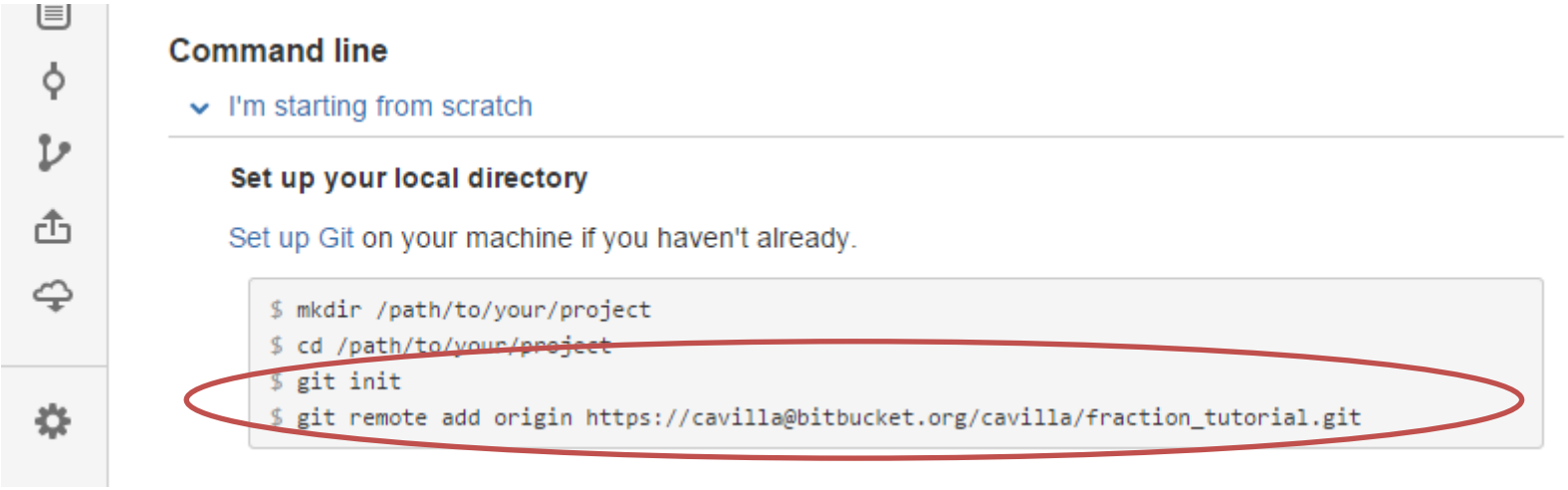
    /src source files

    /src/unittest code with unit tests

    /test code to run the tests

Git: creating the working directory

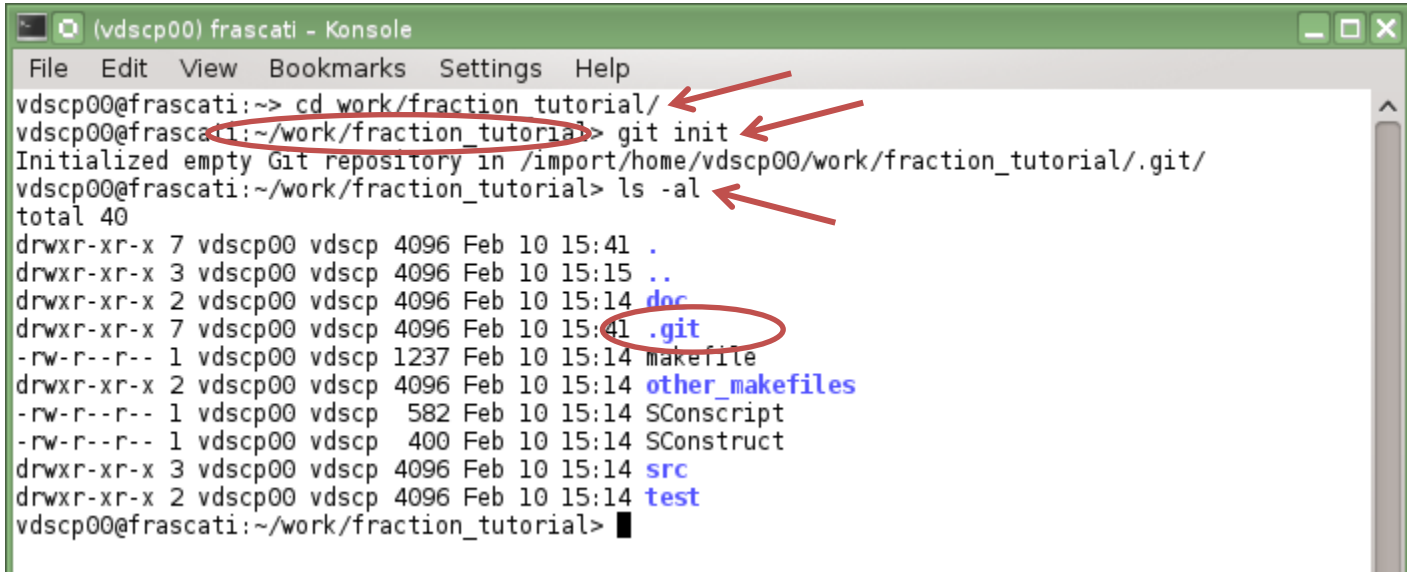
- Use the indications provided by bitbucket in order to connect the local working directory with the central repository



- Next slide gives more details …

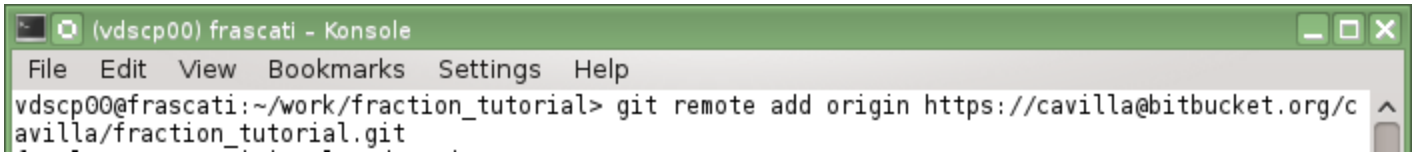
Git: creating working directory

- On the console type (remember you should be in the *fraction_tutorial* directory): *git init*
- This creates a new local repository (so from now we have the central and local repositories). If you type on the console *ls –al* you should be able see the new hidden *.git* folder

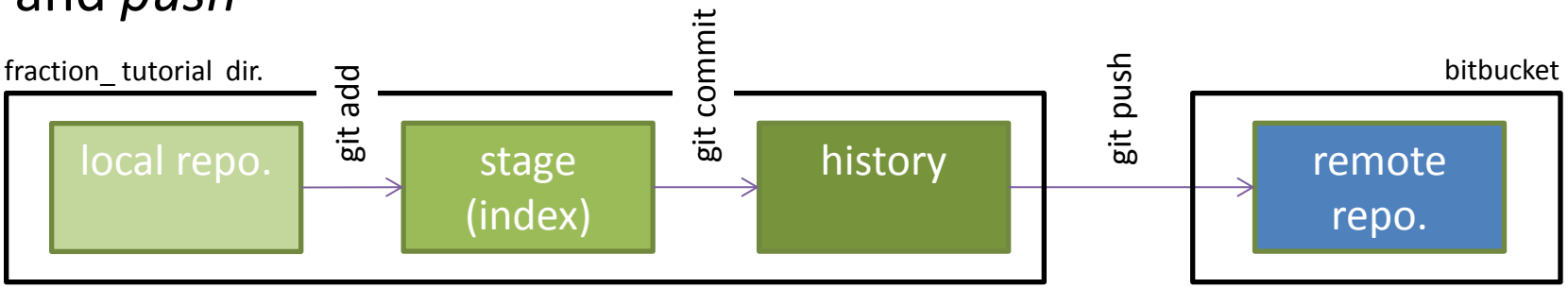


Git: creating working directory

- It is time to connect the local repository with the central one, for that type:
  git remote add origin https://username@bitbucket.org/username/fraction_tutorial.git
  - adjust *username* accordingly



- Since the current code in the local repository does not exist in the central one (type *git status* to check this) we need to save it
- This can be done by following three steps in git*: add, commit* and *push*



Git: saving into the central repo.

- Add files and folders to the stage using *git add*

```
(vdscp00) frascati - Konsole
File  Edit  View  Bookmarks  Settings  Help
vdscp00@frascati:~/work/fraction_tutorial> git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       SConscript
#       SConstruct
#       doc/
#       makefile
#       other_makefiles/
#       src/
#       test/
nothing added to commit but untracked files present (use "git add" to track)
vdscp00@frascati:~/work/fraction_tutorial> git add SConscript SConstruct doc/ makefile other_mak
efiles/ src/ test/
vdscp00@frascati:~/work/fraction_tutorial>
```

- If you type again *git status* you will notice that the files we added are marked as new and that they are ready to be committed
- Note that the new files are in the stage but not in the history

Git: saving into the central repo.

- In order to commit into the history type:
  git commit –m "meaningful message so that we can track and assess what you did"

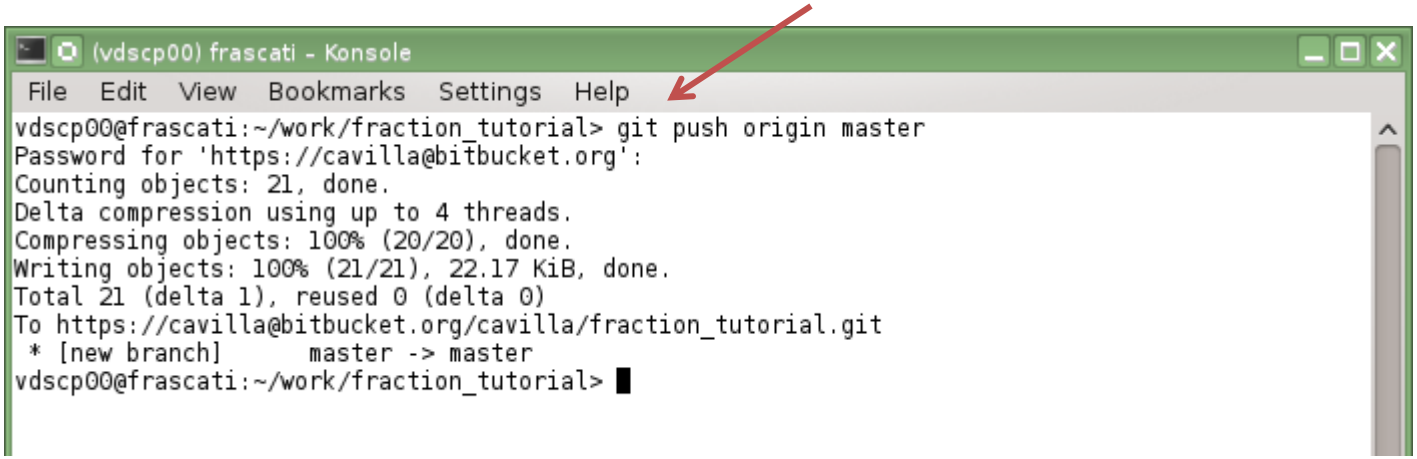
(vdscp00) frascati - Konsole

File Edit View Bookmarks Settings Help

```
vdscp00@frascati:~/work/fraction_tutorial> git commit -m "Initial commit: contains the basic pro
ject structure. - Code for fraction class defines public interface, so far implemented default t
rivial behavior for each public method therefore tests are failing. - Code for unit tests: writt
en according to the expected behavior, - Test runner: written and working"
[master (root-commit) 0ebe050] Initial commit: contains the basic project structure. - Code for
fraction class defines public interface, so far implemented default trivial behavior for each pu
blic method therefore tests are failing. - Code for unit tests: written according to the expecte
d behavior, - Test runner: written and working
 Committer: vdscp-Account (vdscp00) <vdscp00@frascati.eit.uni-kl.de>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 14 files changed, 1883 insertions(+)
 create mode 100644 SConscript
 create mode 100644 SConstruct
 create mode 100755 doc/doxyConfigEx
 create mode 100644 makefile
 create mode 100644 other makefiles/makefile.1
```

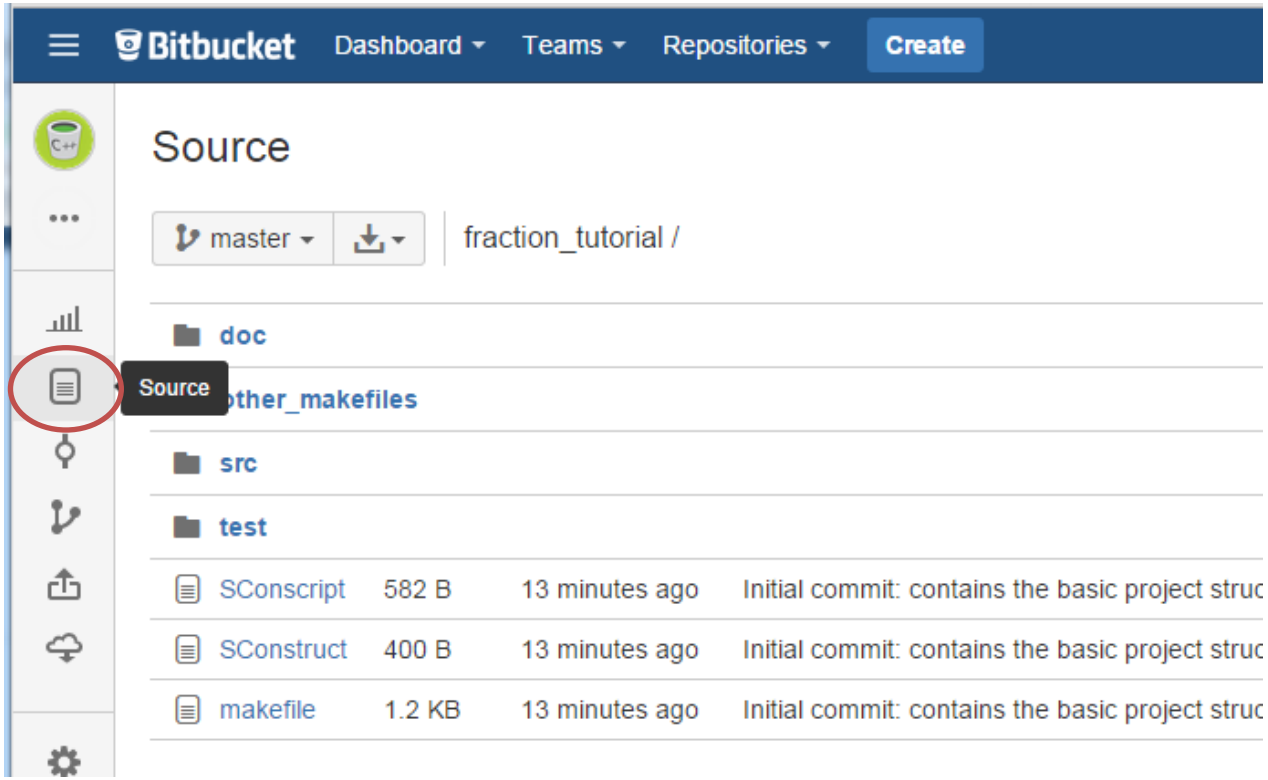
Git: saving into the central repo.

- Now the files are in the history and need to be pushed into the central repository
- Type: *git push origin master*
  - *origin* refers to the central repository and *master* to the current branch
  - when asked type the password of your bitbucket account



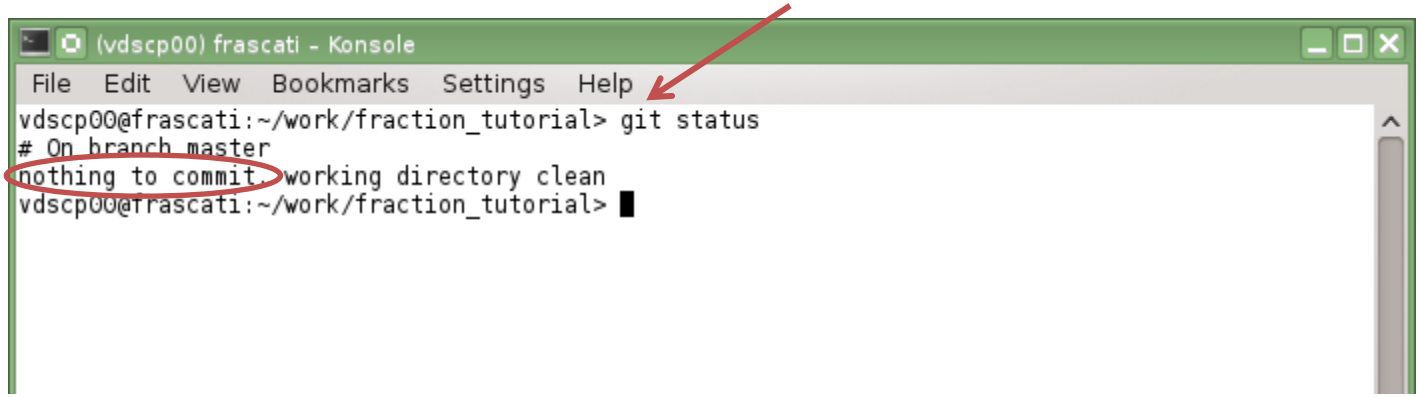
Git: saving into the central repo.

- You may check that the changes done are now saved in the central repository:



Git: saving into the central repo.

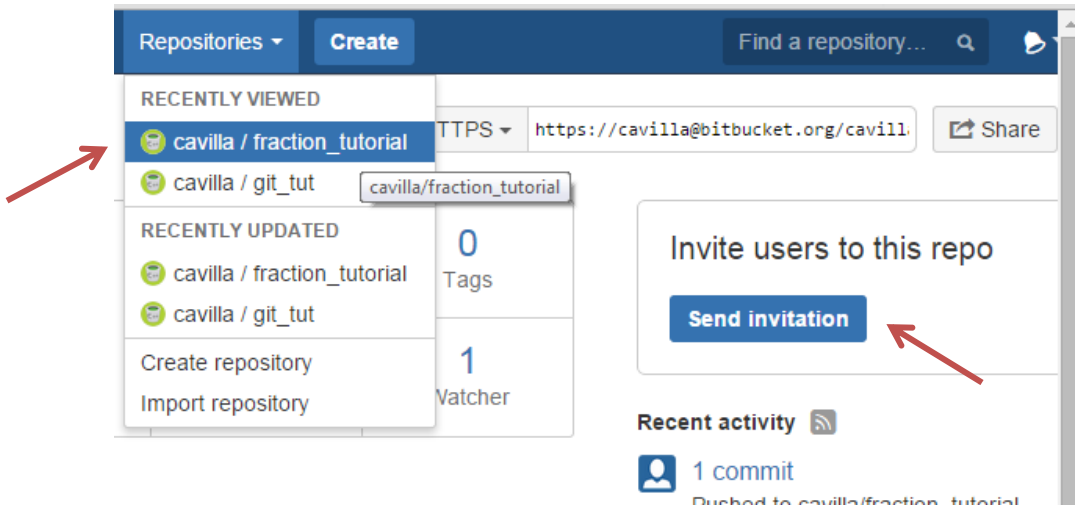
- Type again *git status* to check that now everything is up-to-date



- So far you are done with git stuff!
- Now lets have a look to what you just pushed …

Git tips (2)

- Here some extra tips:
- If you want to give access to a team member you can send an invitation in bitbucket



- Your team colleague can then use *git clone* (see tutorials) in order to work on the files
- To learn about git flows read this documentation: https://www.atlassian.com/git/tutorials/comparing-workflows