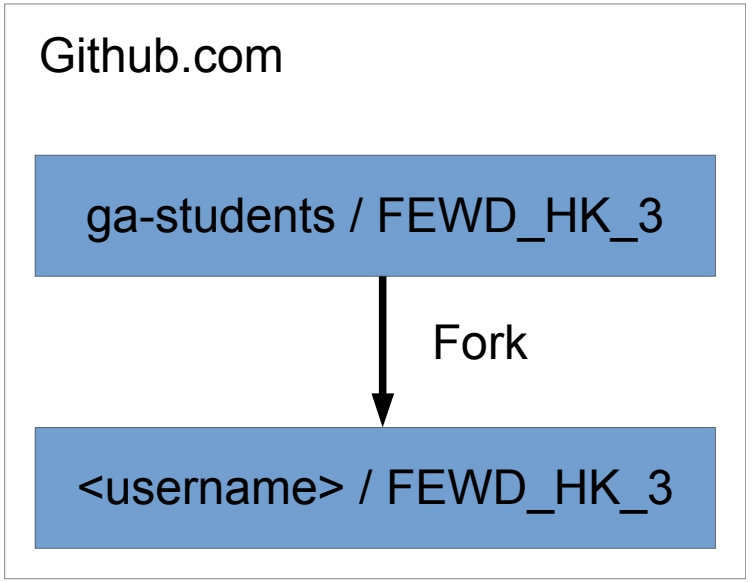
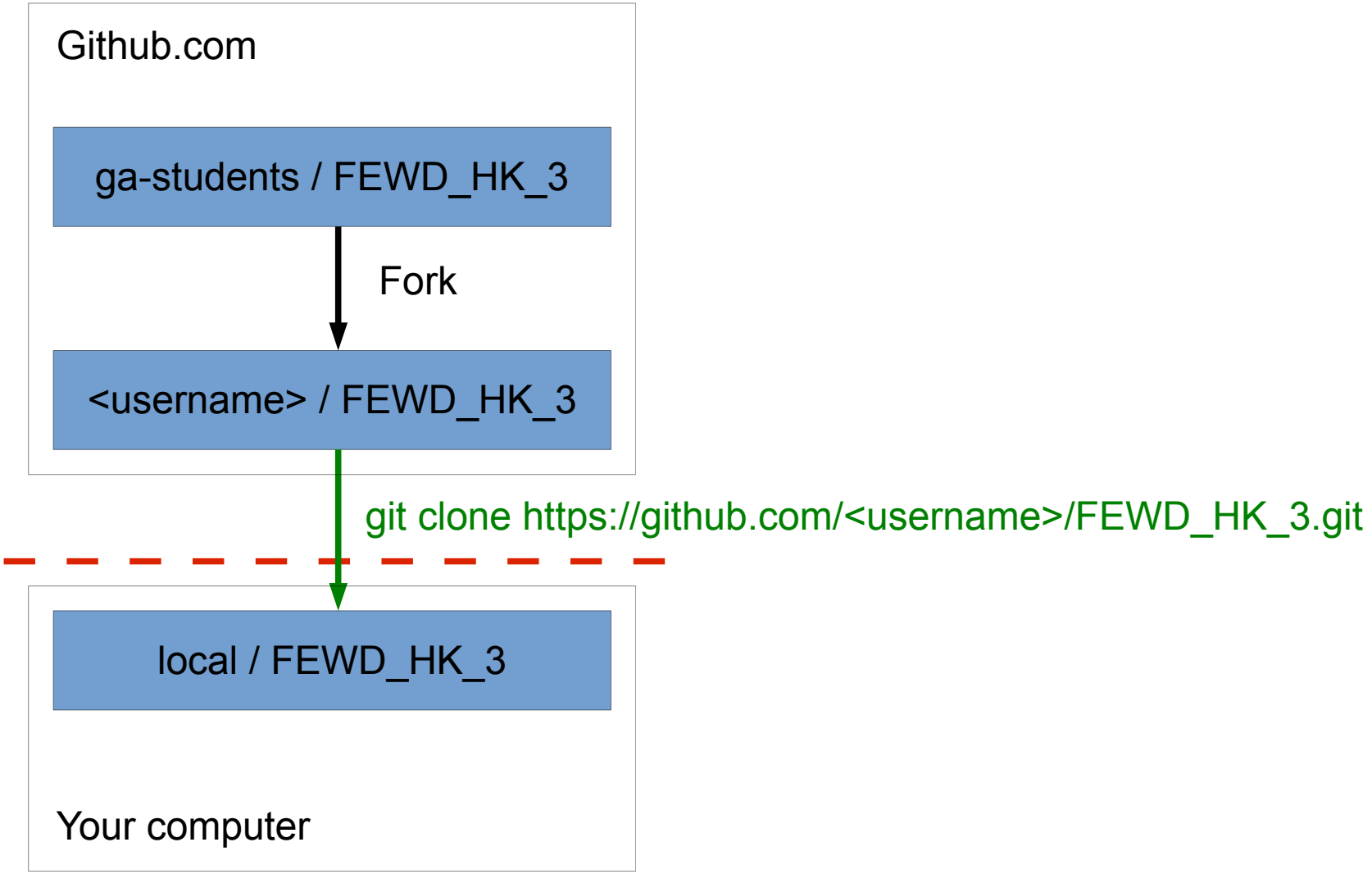
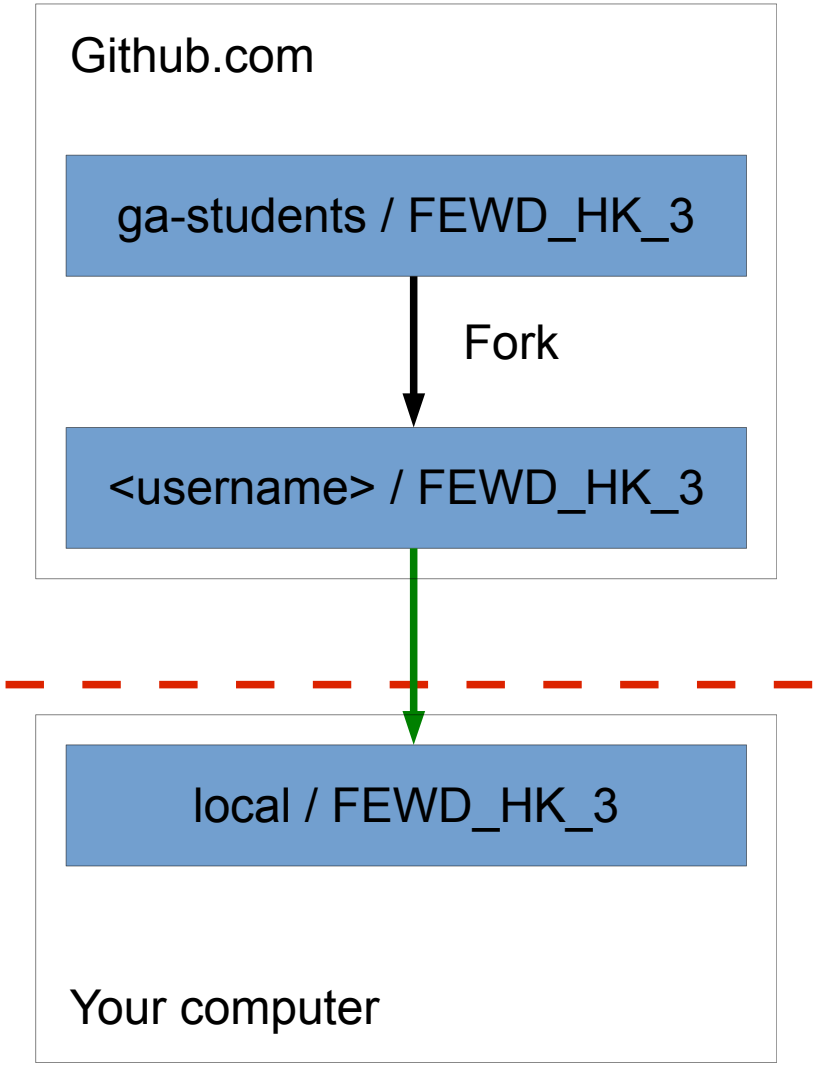
Fork the repository



Clone your forked repo



Go to your local repo in terminal



Keys: command | terminal response)

1. Assume your local repo is located at /User/kit/Desktop/FEWD_HK_3

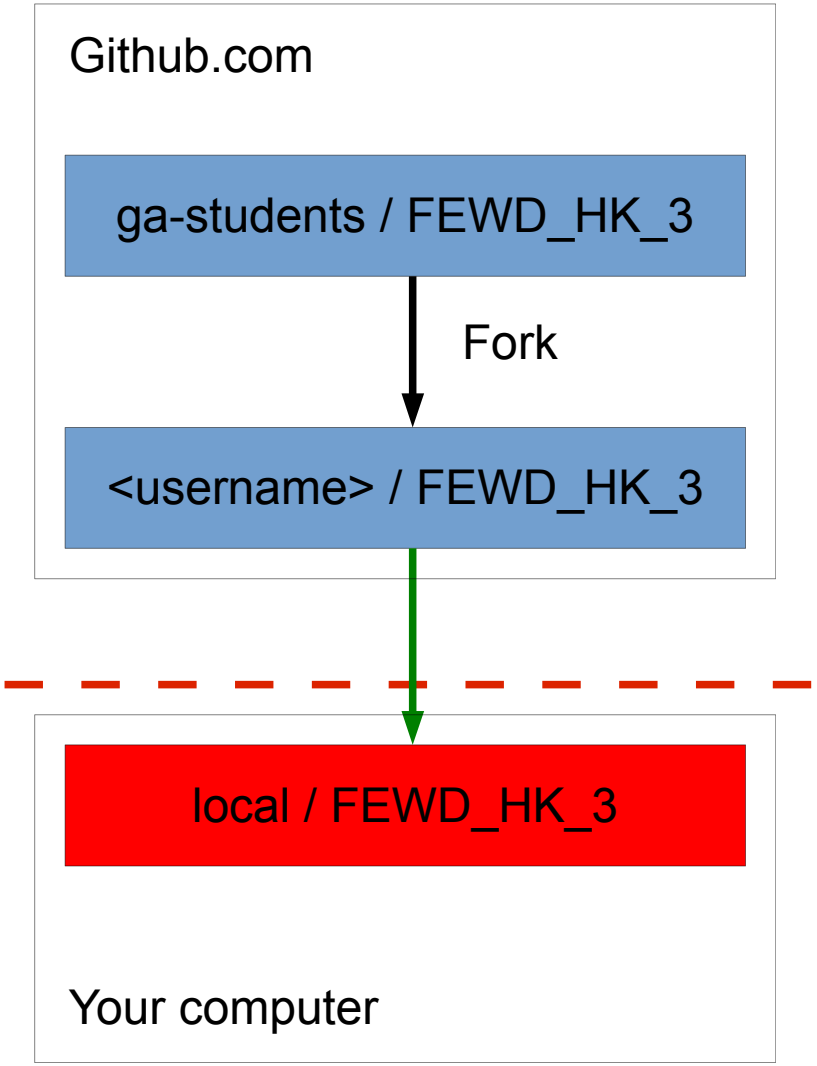
2. Open terminal

3. Input pwd to check the current path
$ pwd
/User/kit/

4. Change directory
$ cd Desktop\FEWD_HK_3

5. Check current repo status
$ git status
# On branch master
nothing to commit, working directory clean

Make some changes to your local

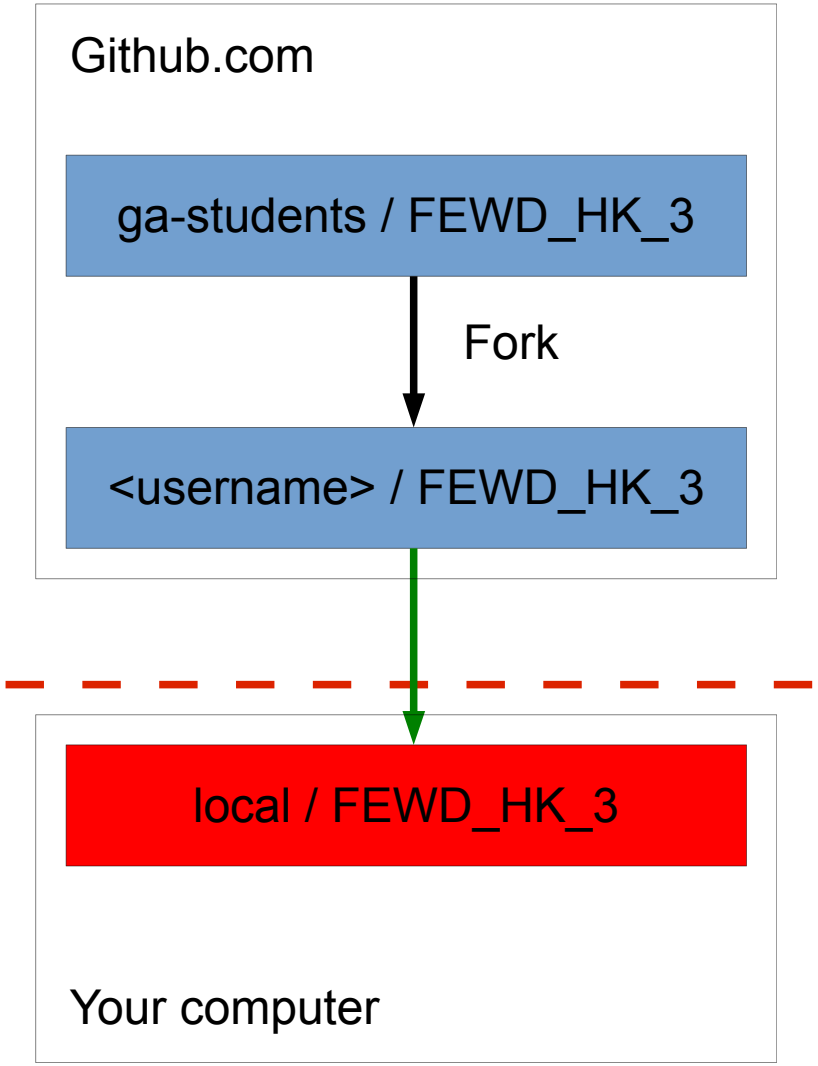


After modified some files, check the status again.

$ git status

```
# On branch gh-pages
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   index.html
#       modified:   style.css
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Prepare your file for commit

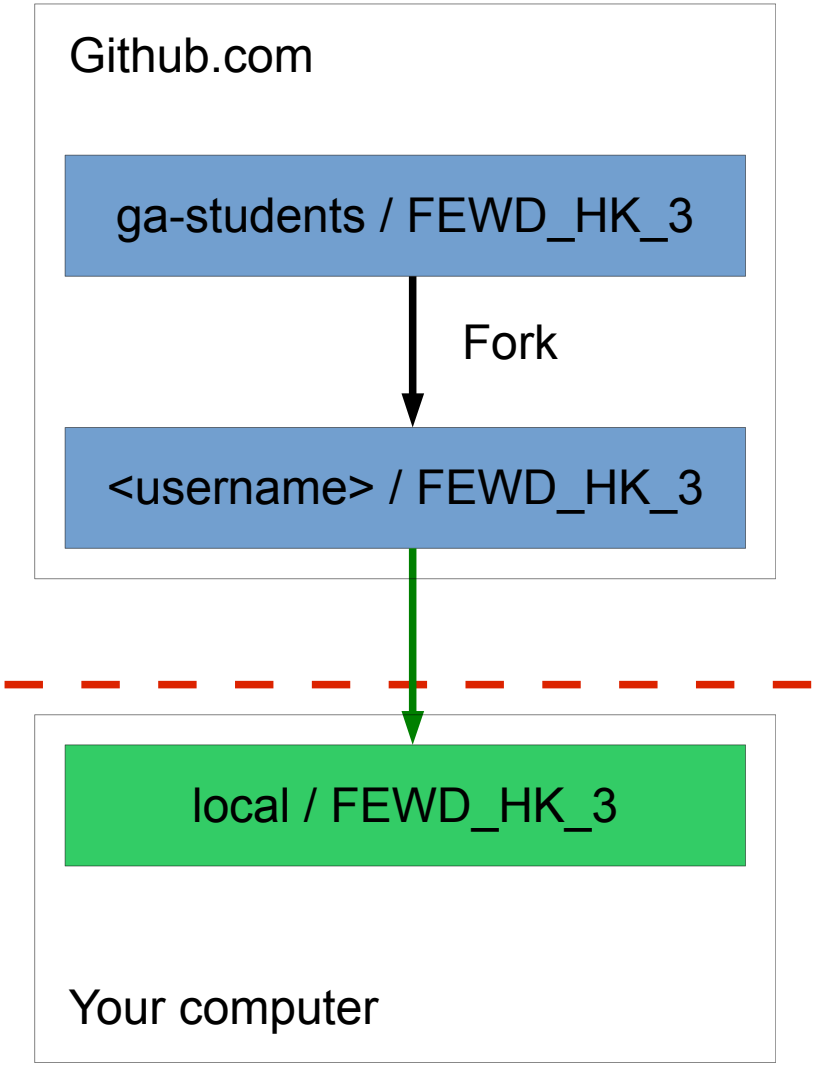


This will add all the modified files to staged for commit.
$ git add .

You can also add the files which you want to commit. (This happens when you do not want to commit all modified files.)
$ git add index.html

On the other hand, if the changes include a deleted file. You need to use git rm to untrack that file from the repository
$ git rm index.html

Commit your changes locally



Commit the changes with a commit message which is wrapped inside the ""

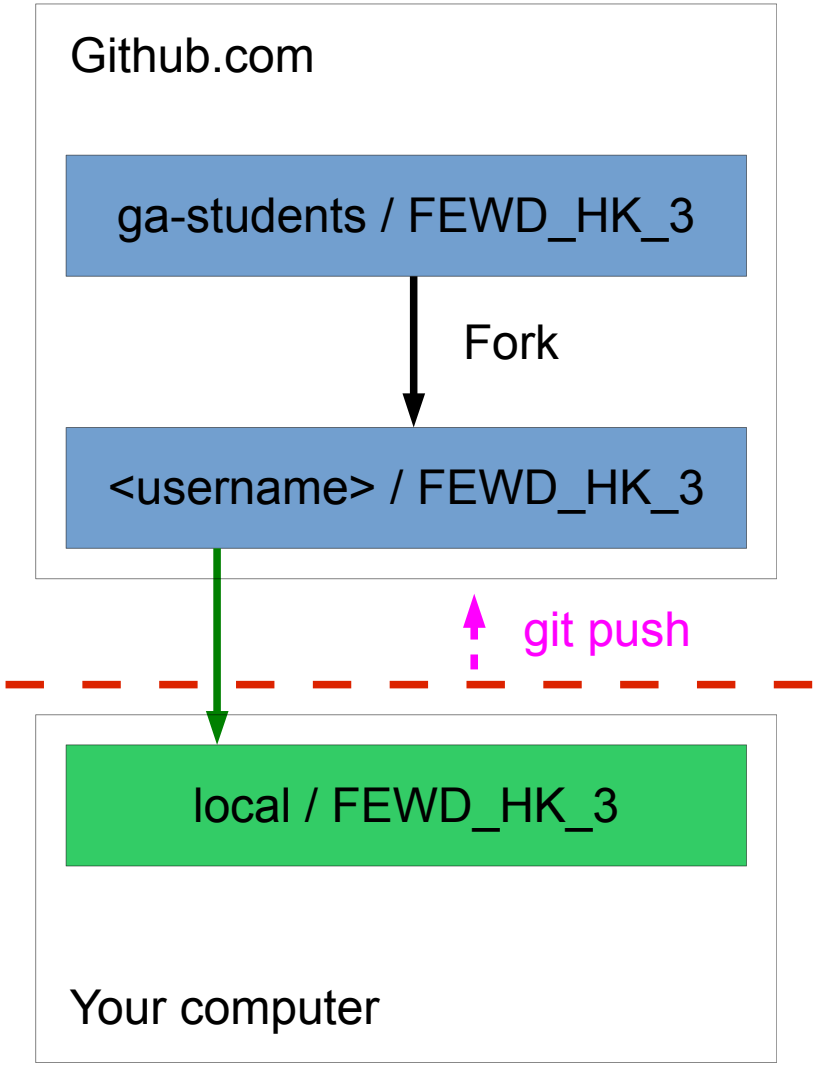
```
$ git commit -m "your message"
[gh-pages 981f857] your message
 1 file changed, 1 insertion(+)
```

After the commit, you can check your local repo status again

```
$ git status
# On branch gh-pages
# Your branch is ahead of 'origin/gh-pages' by 1 commit.
#   (use "git push" to publish your local commits)
#
nothing to commit, working directory clean
```

You are now ahead of your forked repo on commit. Next, you are going to push this changes to your forked repo.

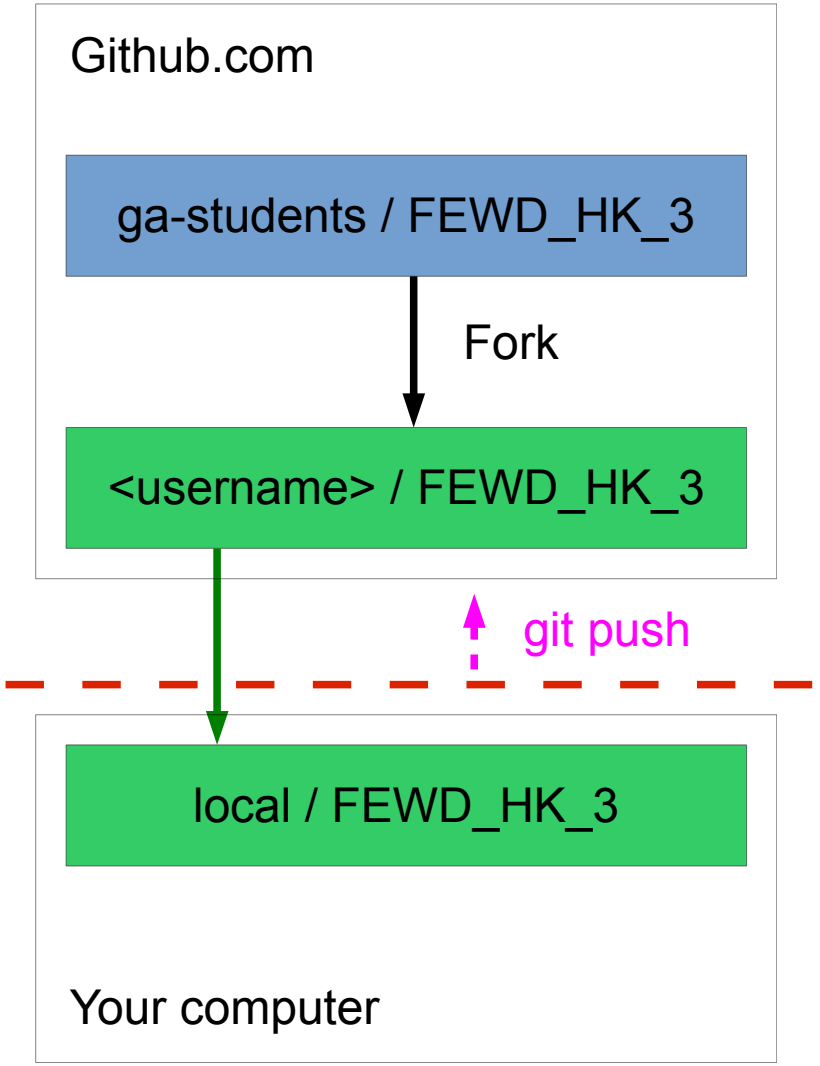
Commit your changes locally



To push you committed changes to your forked repo. Just need to

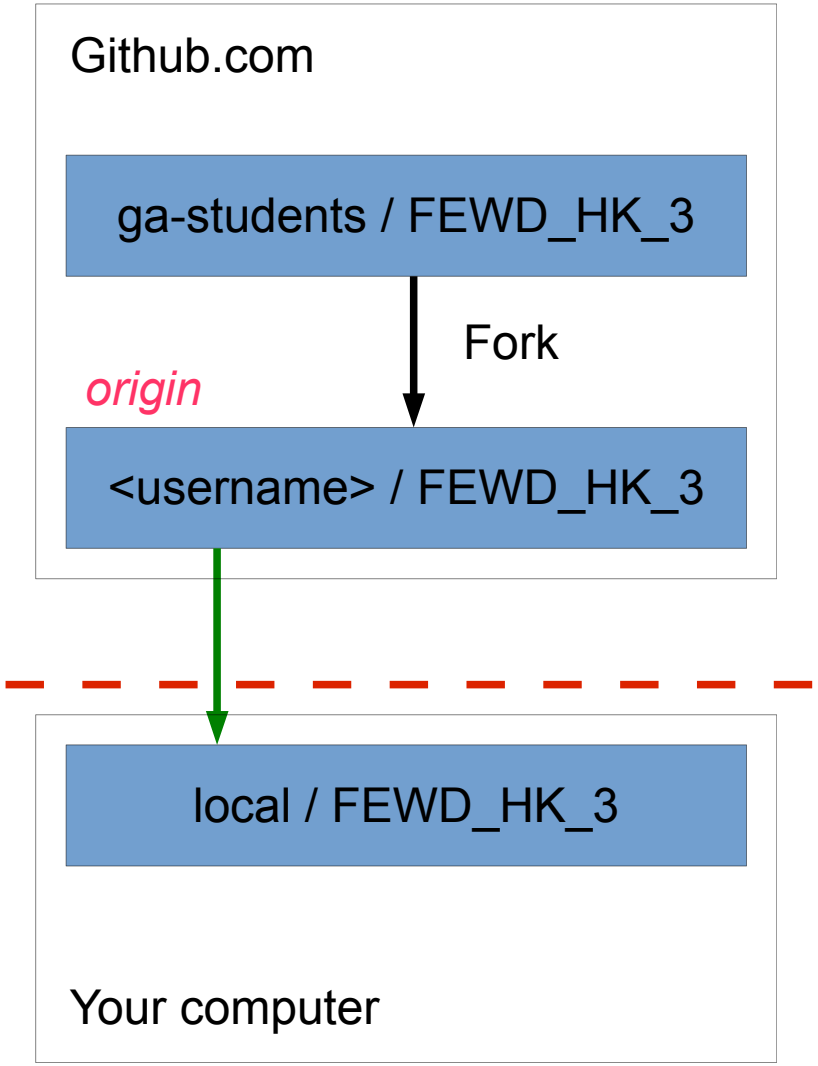
```
$ git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 299 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To https://github.com/<username>/FEWD_HK_3.git
   c746df7..981f857  gh-pages -> gh-pages
```

You have done it!



Go to your forked repository page at GitHub.com and you should be able to find your changes.

The upstream



What we have done so far is between your forked repo and your local repo. There is nothing related to the original repo which is ga-students / FEWD_HK_3.

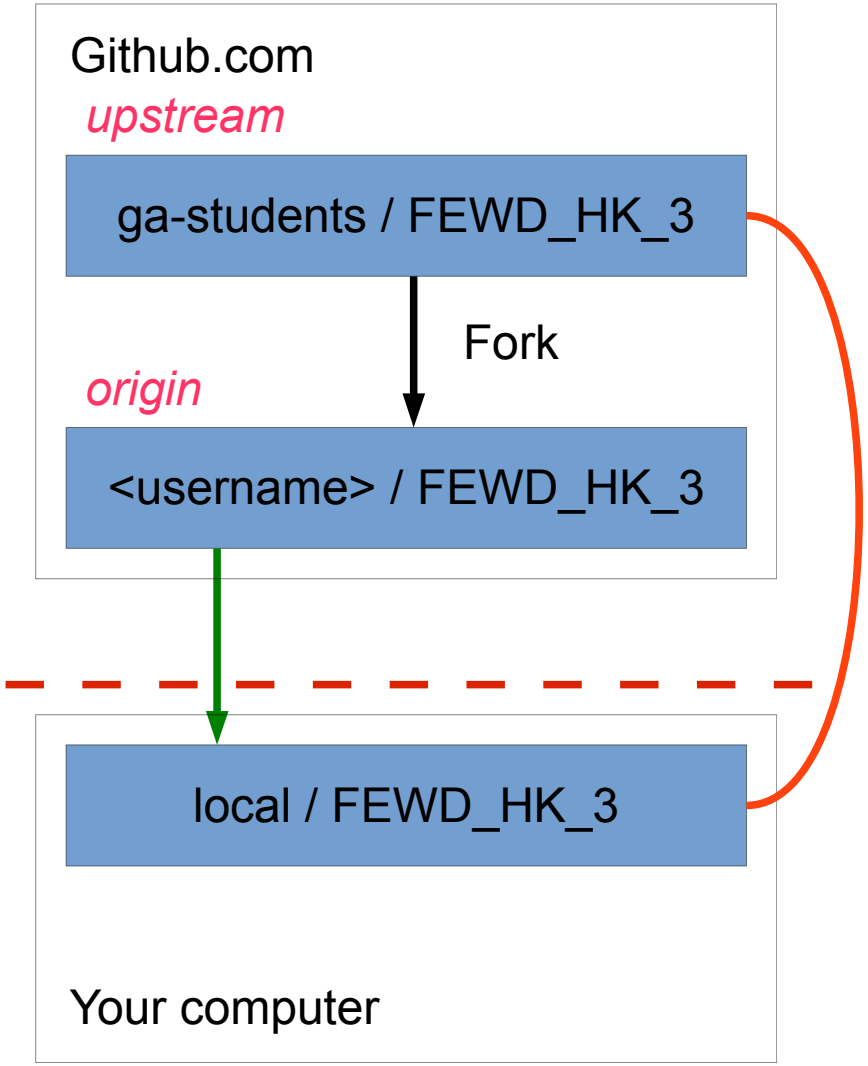
So in order to keep your forked repo in sync with the original repo, we need to link the local repo to the ga-students / FEWD_HK_3.

Before we did anything, input git remote -v to list all the remote repos.

```
$ git remote -v
origin  https://github.com/<username>/FEWD_HK_3.git (fetch)
origin  https://github.com/<username>/FEWD_HK_3.git (push)
```

origin is the default repo nickname for any repo which you initially clone from. (fetch) and (push) indicate the direction. It means that both git fetch origin or git push origin points to the same remote repo.

Add a new remote



Let's name the ga-students / FEWD_HK_3 as upstream link it to our local.

$ git remote add upstream https://github.com/ga-students/FEWD_HK_3.git

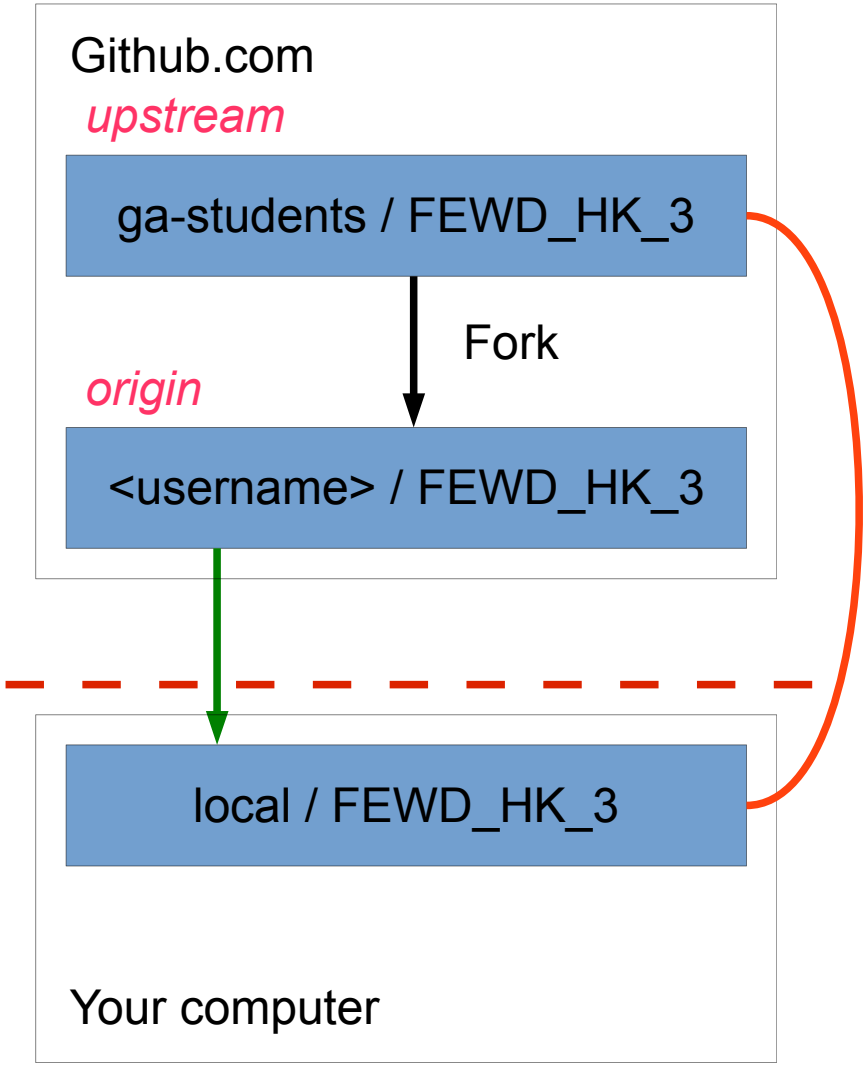
List the remote again
$ git remote -v

```
origin  https://github.com/<username>/FEWD_HK_3.git (fetch)
origin  https://github.com/<username>/FEWD_HK_3.git (push)
upstream https://github.com/ga-students/FEWD_HK_3.git (fetch)
upstream https://github.com/ga-students/FEWD_HK_3.git (push)
```

Now your local has a link to the upstream repo.

Although this link allow you to push to the upstream, it is prohibited because your github account is not assigned with the permission.

Pull from upstream to local



Check if there is update from upstream.

```
$ git fetch upstream
remote: Counting objects: 25766, done.
remote: Compressing objects: 100% (10611/10611), done.
...
```

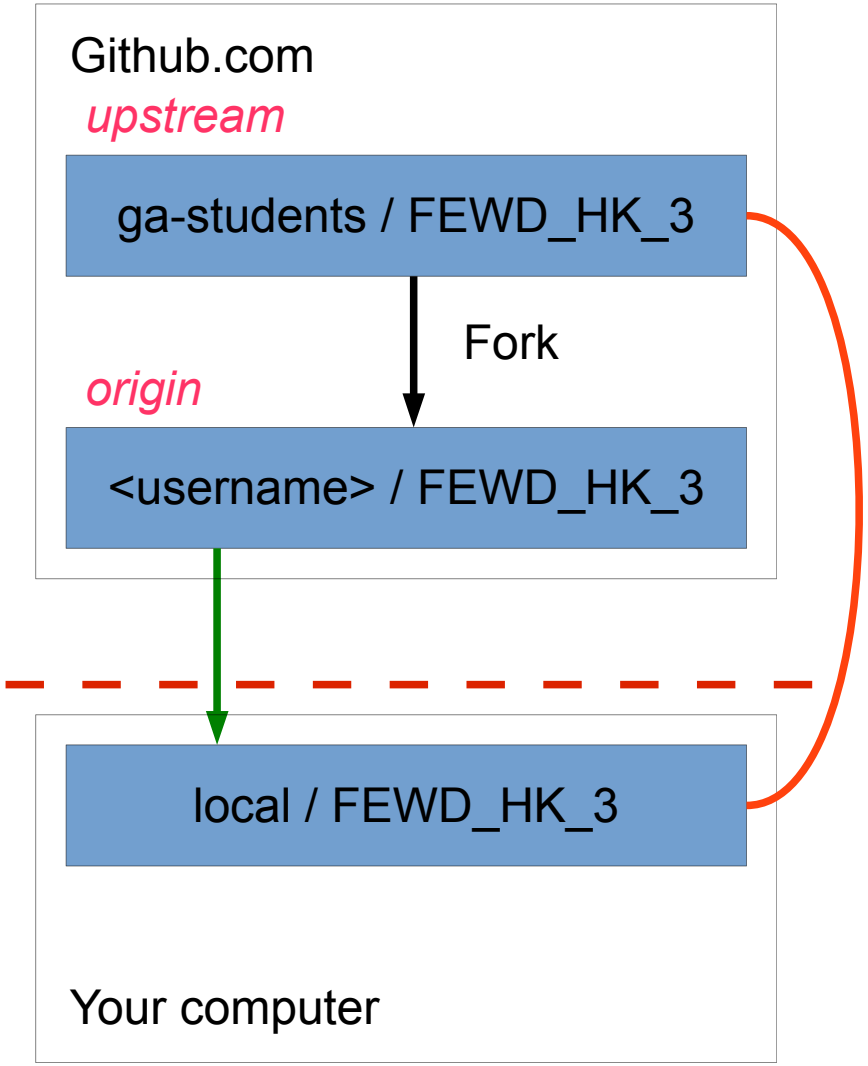
If u see there is response, that indicate there is new update at upstream.

Before you start the merge, follow the steps in the previous slides and commit all your outstanding changes in your local repo to make sure it is clean.

Now you can merge the new update to your local repo.

```
$ git merge upstream/gh-pages -m "your message"
[gh-pages 6e7f843] your message
 1 file changed, 1 insertion(+)
```

Finally!

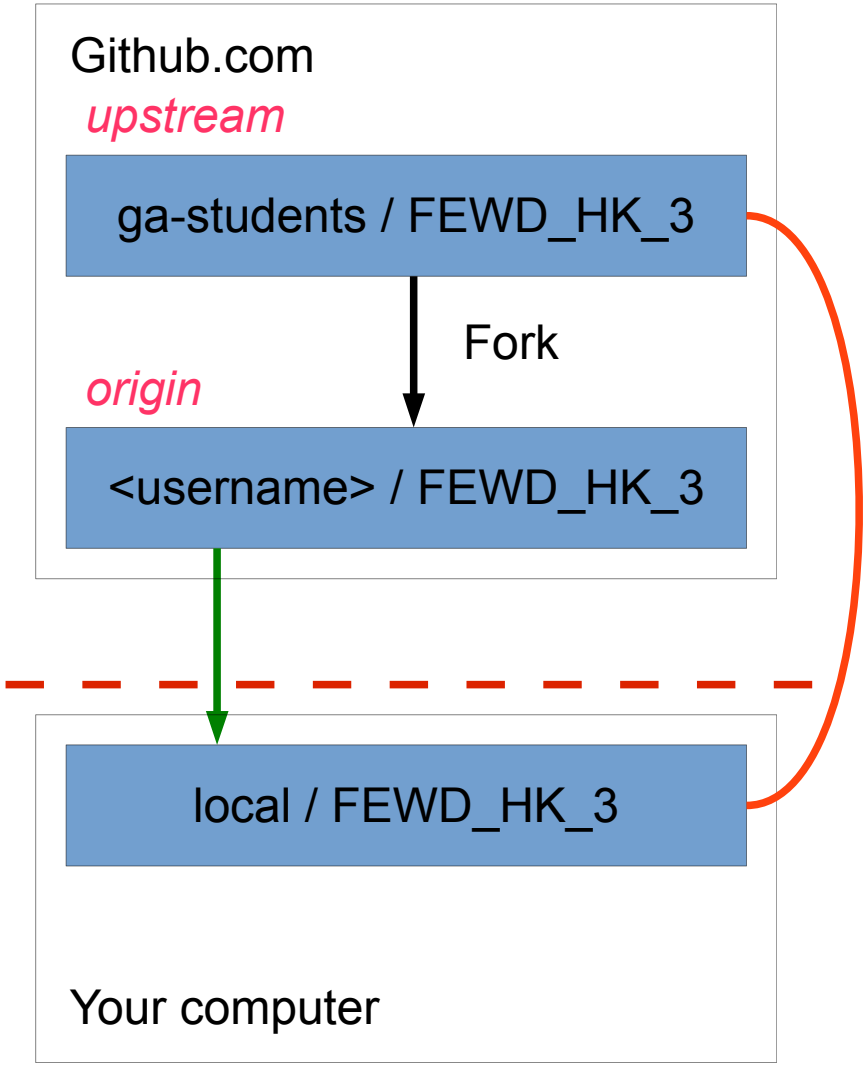


After the merge, you can push the latest update to your forked repo. Just input

```
$ git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 299 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To https://github.com/<username>/FEWD_HK_3.git
   c746df7..981f857  gh-pages -> gh-pages
```

That's how to maintain your repository.

gh-pages?



You may wonder what's the meaning of gh-pages. Actually that is just a remote branch name but it has special meaning.

GitHub.com will host your source files under gh-pages branch at the webserver called github.io.

Assume you have finish your homework and the index.html is at
/FEWD_HK_3/homework/week1/xxx/resume/index.html

To access it on browser, simply go to
http://xxx.github.io/FEWD_HK_3/homework/week1/xxx/resume/index.html

So whenever you push to your forked repo, it will trigger a build on github.io such that your website there will be updated. Normally it will take a few mins for that.