

 **Course:** CS-UH-2012 Software Engineering

 **In-class Activity:** Using git and github PRs

 **Related Lecture(s):** L12 - Construction



# git-practice

This exercise walks through basic usage scenarios of git

# Steps to follow

## Common steps

1. Create a new repository on your github account
2. Clone this new repository by using the git clone <repo address> command. You can find the repo address by clicking on the green Code button on the main page of the repo.

## Exercise 1: basic git usage

1. Add any file in the cloned folder. Let's say and add any code to it.

hello.py

1. Ask git to track this file by adding it to git's index:

git add hello.py

1. Commit your changes

git commit hello.py -m "Add simple python code"

1. Make another change to and commit again

hello.py

1. Push your changes using ; Voila! If you go back to your repo on github,

git push

you should see the commit history of your changes.

## Exercise 2: Branching and Merging

### Exercise 2.1: A simple case merged on the command line

1. git branch my-new-feature
2. git checkout my-new-feature
3. Create a new file main.py and add any code to it
4. Ask git to track this file (remember you are now tracking it on the new branch, not main):

git add main.py

1. Commit your change git commit main.py -m "Add main on branch"
2. Push your change git push . Since this is the first time you are pushing to this

branch, git will ask you to set upstream. Copy and run the command it provides you to set upstream for this branch (it will look something like this

.

git push --set-upstream origin my-new-feature

1. Go to github and change the branch you are viewing from the top left corner and see how you see the commits from main (until the time you did the branch) plus the new commit you added.
2. Now let's merge our feature branch changes in to the main branch. Since we want to merge things into main, we will check out

main first git checkout main

git merge new-feature .

1. We will ask git to merge our new branch
2. Run a and see how it tells you that your local main branch is ahead of

git status

origin/main and that you need to push your changes. This is because the git merge command created the merge commit when it merged the changes (or more precisely since this case is a fast forward, it just added the new commit to the main history).

git push

1. Push to the remote repo
2. Go to github and see how the new change(s) you made are part of the main branch history

### Exercise 2.2: Merge conflict merged on the command line

git status

main by running changes.

to make sure we are on the main branch. If we are not, then change to

. Also make sure you have no uncommitted

git checkout main

1. just to get in the habit of making sure that your repo is always up to date when you start working.

git pull

1. git branch featureX
2. git checkout featureX
3. Edit main.py and make any changes to the code
4. Commit your change git commit main.py -m "Add changes to main" (P.S. this is a pretty bad commit message :-) )
5. Push your change . Since this is the first time you are pushing to this

git push

branch, git will ask you to set upstream. Copy and run the command it provides you to set upstream for this branch (it will look something like this

git push --set-upstream origin featureB

.

1. Just as a sanity check, go to github and change the branch you are viewing from the top left corner and see how you see the commits from main (until the time you did the

branch) plus the new commit you added.

1. Now, let's pretend that someone else made changes on the main branch in the meantime. To make this situation real, we will do the following: a.

b. Make changes to the SAME LINE in but create a slightly different variation of this line (e.g., if in step 2, you changed

git checkout main

main.py

to , here change the same line to c.

x = x + 1

x = y + 1

x = x + 2

Commit and push your changes

1. Now, let's try to merge our changes from the branch into main a.

b. there's a conflict in file main.py

git checkout main

git merge featureX

At this point, git will complain

CONFLICT (content): Merge conflict in main.py

1. So we will open main and find the conflict markers and decide how we want to resolve this conflict.
2. Once I'm done resolving the conflict, I will ask git to add the file back to its staging/index:

. If I do a now, git will tell me that i've resolved

git add main.py

git status

all conflicts but I'm still merging and need to commit to conclude the merge. So let's commit as follows

git commit -am "Merged changes from main"

using git push . I have now integrated the changes I

1. Push our changes

the branch into main and resolved conflicts along the way.

made on

## Exercise 3: Pull Requests

### Exercise 3.1 A simple case merged through a PR

Let's now try the simple case from 2.1, but merge things through a Pull Request rather than through the command line.

git status

main by running changes.

to make sure we are on the main branch. If we are not, then change to

. Also make sure you have no uncommitted

git checkout main

1. just to get in the habit of making sure that your repo is always up to date when you start working.

git pull

1. git branch featureB
2. git checkout featureB
3. Edit main.py and make any changes to the code
4. Commit your change git commit main.py -m "Add changes to main" (P.S. this is a pretty bad commit message :-) )
5. Push your change . Since this is the first time you are pushing to this

git push

branch, git will ask you to set upstream. Copy and run the command it provides you to set upstream for this branch (it will look something like this

git push --set-upstream origin featureB

.

1. Just as a sanity check, go to github and change the branch you are viewing from the top left corner and see how you see the commits from main (until the time you did the branch) plus the new commit you added.
2. Now, let's create a pull request to merge the changes we made on the branch into the main branch. Go to the Pull Requests tab on the github repo web page and select "New

Pull Request". Make sure the base branch shows (since the repo is the same

main

repo, you won't see a separate repo name selection) and the compare branch shows

. You should now see the new commit you made in the branch and you can click on "Create Pull Request"

featureB

1. On the created pull request, you can see that github tells you that this branch has no conflict with main and can be safely merged. Click on "Merge pull request". Once you are done merging, you can delete the branch through the github interface on the PR

main.py

1. Now go to the main branch on github and you will see the that new changes

git checkout main

1. On your command line, switch to main

and you will also see the expected commit history there

and do a

has your

git log

### Exercise 3.2: Ok, let's create a conflict in our PR

Let's now create a similar case to 2.2 but merge through a PR

1. Assuming we are on main and have the latest updates, create a new branch

#note that this is a shorthand for creating and checkout out a new branch in one step

git checkout -b featureC

main.py

1. Change any line in by now)

and commit and push your changes (you know the drill

1. Now, let's pretend that someone else made changes on the main branch in the meantime. To make this situation real, we will do the following: a.

b. Make changes to the SAME LINE in but create a slightly different variation of this line (e.g., if in step 2, you changed

git checkout main

main.py

to , here change the same line to c.

x = x + 1

x = y + 1

x = x + 2

Commit and push your changes

1. Now let's try to merge our changes from featureC into main using the same steps from exercise 2.2: a. We go on github and create a new Pull Request asking to merge featureC into main b. Now notice how github is refusing to show me the merge button and is asking me to resolve conflicts first? Well, let's resolve the conflict. I will resolve it on the command line as follows c. I want to resolve the conflicts on the BRANCH such that github will allow me to merge the pull request into main. Why? I'm the one who's asking to merge my changes on main so I'm responsible for figuring out the conflicts on my new branch and solving them before asking someone to integrate my chages. So:
2. git checkout featureC
3. git merge main This causes git to list the files that have conflicts. iIn this case, we

will see something like

CONFLICT (content): Merge conflict in main.py

1. So we will open main and find the conflict markers and decide how we want to resolve this conflict.
2. Once I'm done resolving the conflict, I will ask git to add the file back to its staging/index:

. If I do a now, git will tell me that i've resolved

git add main.py

git status

all conflicts but I'm still merging and need to commit to conclude the merge. So let's commit as follows

git commit -am "Merged changes from main"

using git push (note this is pushing to our branch)

1. Push our changes
2. Now go back to your PR on github and refresh. You should see that github now allows you to merge your changes into the main branch! Merge your changes into main.

Note: if you have multiple conflicting files, you will have to resolve all conflicts (steps 7-8).