

How To Use PLaF With Github

Sarang Hadagali
Version 1

March 2, 2023

1 Why Github

Using Github to access and modify the PLaF folder is great because it allows you to save your work on an online repository that won't be deleted if something goes wrong on your computer. You are also very easily able to update your folder if Professor ever decides to make changes to it.

Note:

Create and setup a Github account before completing the rest of this. Also you have to delete any local PLaF copies you have before starting this, so if you have any work on the PLaF directory make sure to save it and then copy it in later!

1.1 Cloning vs. Forking

There are two options when using Github, you could either clone the repository or fork it. Cloning the repository tends to be easier as there are less steps, however, there are limitations that you will experience as we do not have push access to the origin repo. For example, when cloning if you make edits on your local copy and then want to pull from Professor's origin branch then you may have to stash your changes which can cause headaches to get back. A better solution is to fork the repository, which is the same thing as cloning but with an extra step. Forking the repo, allows you to make a new duplicate repo that is still connected to the Professor's original branch but also allows you to push and save your changes at the same time, avoiding the issue of stashing your changes.

2 Forking Setup

2.1 Forking PLaF

1. Open up the [PLaF](#) repo and in the top right corner of the screen select Fork
2. Title your repo whatever you'd like, you can just leave it PLaF
3. Leave the default setting selected and click "create fork"
4. You should now be in a screen that has "you github username"/PLaF as opposed to ebonelli/PLaF
5. Select the green "Code" button



6. Select "HTTPS" and copy the link "https://github.com/"YOUR-USERNAME"/PLaF.git
7. Navigate to a terminal and cd into the folder you want to create your PLaF folder in
8. Type "git clone" and paste the link you just copied

```
CS496 % git clone https://github.com/"gitUsername"/PLaF.git
```

9. Enter and then cd into the PLaF folder

- Enter "ls" and you should be able to see the PLaF folder within your directory

2.2 Setting Up Local Git Repository

Now that we have the PLaF folder we can move onto setting up the local git repository that will allow us to push and pull the new Github repo without effecting the Professor's repo.

In your PLaF folder type "git remote -v"

```
terminal@your-terminal PLaF % git remote -v
origin https://github.com/"githubUsername"/PLaF.git (fetch)
origin https://github.com/"githubUsername"/PLaF.git (push)
```

2.3 Setting Upstream

The next part linking this to the upstream repo, AKA, the Professors repo. Follow the code snippet below in your PLaF directory

```
$ git remote add upstream https://github.com/ebonelli/PLaF.git
$ git pull upstream main
```

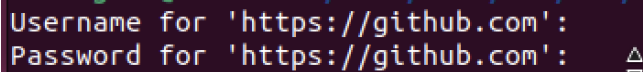
Once again run "git remote -v" your output should now be

```
terminal@your-terminal PLaF % git remote -v
origin https://github.com/"githubUsername"/PLaF.git (fetch)
origin https://github.com/"githubUsername"/PLaF.git (push)
upstream      https://github.com/ebonelli/PLaF.git (fetch)
upstream      https://github.com/ebonelli/PLaF.git (push)
```

Congratulations You have now set up a forked repository!

2.4 Login to Github

If you are using a virtual machine or have never used github before locally then you may encounter an issue when trying to push something. If you are prompted with "git config —global user.email" and "git config —global user.name" just input the email you signed up with github and your github user name. If your terminal output asks for username and password follow the steps below. If not you can skip this step.

A terminal window with a dark background. It shows two lines of text: "Username for 'https://github.com':" and "Password for 'https://github.com':" followed by a cursor icon. The text is in a light color, likely white or light blue.

```
Username for 'https://github.com':
Password for 'https://github.com':
```

1. Enter your normal github username and press enter

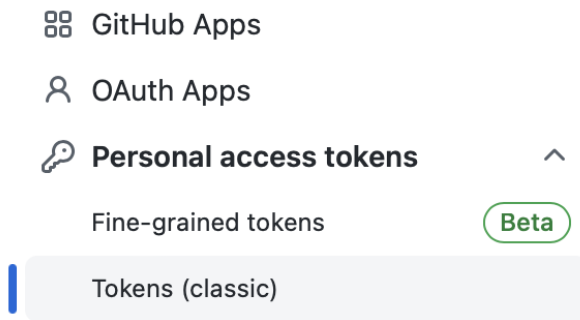
- Do not type in your password this will not work as Github has changed its security policy and now requires a special token

2. Go to github and navigate to settings

3. Scroll to the bottom of the page and select developer settings on the left sidebar

4. Select the Personal access tokens drop down

5. Select the second option Tokens(classic)



6. Click generate new token at the top
7. Select the second option "Generate new token(classic)"
8. You can type in some note that explains what you are using it for such as "PLaF Access"
9. You can set the expiration date to however long you want, but if you set it up too short then you may have to redo these steps in the future
10. Since this is a repo for yourself you can select all the permissions that are given
11. You should now have a long generated token, copy this and paste it into the password field
12. You should have access to your github repo locally now, and should be able to push and pull.

With this you should be able to push to your repo. Make sure to keep this token safe as github will not display it once you close the tab so you will have to redo this if you don't copy it somewhere.

3 How to Use

Now that you have created your own forked repo that is connected to the Professors repo here are a couple of pointers on how to effectively use git.

- To push your changes to your remote repository first make sure you are on your main branch by typing "git branch" and there should be *main outputted then

```
$ git add .  
$ git commit -m "new commit"  
$ git push origin main
```

- If Professor ever asks you to pull from the repo, first follow the step mentioned above to push your changes then do
 - git pull upstream main
 - This should automatically merge your changes with the new changes that Professor has added. However if you have merge conflicts you are going to have to manually resolve that.

If you are having any issues with forking follow this [Stack Overflow post](#) as this is what I used to figure it out.

Everything should be working now and you can continue with the normal process of dune build and dune install. Hopefully the process wasn't too bad and you learned a thing or two about working with Git/Github. There are a lot more features to git so check them out it is a useful skill to have as a CS major! Also feel free to add onto this! The link to this repo is [here](#). If you find any mistakes or any ways to improve it create your own branch and make a pull request and I can take a look at it. Good luck!