### GitHub Introduction Labs

*Fundamentals of collaboration*

**Revision 1.0 – 01/27/24**

Tech Skills Transformations LLC / Brent Laster

**Setup and prerequisites**

1. In order to do some of the labs in this class, you will need to have a personal access token (PAT) setup and also two separate GitHub userids, as well as a version of Git installed.

2. Git can be installed by going to <https://git-scm.org> and following the instructions there for your OS.

3. To create the second GitHub userid, just select another email address and sign up for the free tier at GitHub.com.

4. You can set up the PAT in advance by following the instructions [here](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-personal-access-token-classic) or do it as part of the first lab.

5. If you are doing the labs on Windows, it is recommended to use the Git Bash shell that can be installed with Git for Windows.

**Lab 1 – Getting Started**

***Purpose:*** *In this lab, we’ll get a quick start learning about GitHub through forking a project, creating a new file and committing it.*

1. Log in to GitHub with your primary GitHub account.
2. Go to <https://github.com/skillrepos/calc> and fork that project into your own GitHub space. Do this by clicking on the ***Fork*** button. On the next screen, **make sure to uncheck** the box next to ***Copy the main branch only*** . Then click the ***Create Fork*** button.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

uncheck

1. Now you’ll be on your fork of the repo. Next, let’s clone your repo down to your local system so we can make changes there. In your project, ensure you are on the ***Code*** tab, then click on the large green ***<> Code*** button. In the ***Local*** tab, select ***HTTPS*** under Clone and then click on the ***copy icon*** to copy your project’s URL.

A screenshot of a computer

Description automatically generated

1. Open a terminal on your system and clone down the repository from GitHub. You can use the following command – just paste (or type) the URL you copied from the step above and then change to that directory. Then change into the local working directory.

**$ git clone** <url from repo>

**$ cd calc**

1. If not already set globally, configure your name and email. Best practice would be for your email to be the same as the one you’re using for your userid on GitHub.

**$ git config user.name** *“your name”*

**$ git config user.email** *<same email as you’re using on GitHub>*

1. After this you can run the command below and see that GitHub is setup as your remote repository.

**$ git remote -v**

7. Let’s make a simple edit to a file so we can have a change to push back to GitHub. Edit the calc.html file and update the line in the file surrounded by <title> and </title> to customize it with your name. The process is described below.

**Edit calc.html and change**

**<title>Calc</title>i**

**to**

**<title> *name’s* Calc</title>**

***substituting in your GitHub user ID for “github\_user\_id”.***

8. Save your changes and commit them back into the repository.

**$ git commit -am "Updating title"**

9. Several aspects of using GitHub rely on options you can set in the user ***Settings*** menu. To demonstrate this and in preparation for the next lab, we’ll go to settings to create your Personal Access Token (PAT) that you’ll need for securely pushing changes over to GitHub in place of a password.

To create your PAT, follow the instructions for creating a classic token at <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-personal-access-token-classic>

# (Alternatively, you can go directly to <https://github.com/settings/tokens/new> )

When setting up your token, ensure that you have the boxes checked for the first four scopes (*repo – delete:packages*) as shown below. **Also make sure to copy and save the token for future use.**  
  


When done, click on the green ***Generate Token*** button.

A screenshot of a computer

Description automatically generated

Make sure to save a copy of the token string from this screen - you won’t be able to see it again.

A screenshot of a chat

Description automatically generated

9. Now, let’s go ahead and push your change back into GitHub. We’ll push to a new branch in preparation for the next lab. Go back to your terminal and enter:

**$ git push -u origin main:dev**

10. After this, you'll be prompted for username (your GitHub username) and then a sign-in/Private Access Token or password. Wherever it asks for a token or a password, you can just copy and paste in the token you generated in GitHub prior to this lab. An example dialog that may come up is shown below.

**![Graphical user interface, application

Description automatically generated]()**

# If instead, you are on the command line and prompted for a password, just paste the token in at the prompt. Note that it will not show up on the line, but you can just hit enter afterwards.

# A screenshot of a computer code Description automatically generated

NOTE: If you hit run into problems trying to push with the token, such as it saying invalid password, you may be getting caught by previously saved credentials. See the very end of this doc for some other options.

END OF LAB

**Lab 2 – Pull requests**

***Purpose:*** *In this lab, we’ll see how to merge a change using a pull request.*

1. After the push is complete, you can switch back to the GitHub repo in the browser, change the branch to *dev* and click on the calc.html file to see the change. (If you don’t see ***dev*** listed in the branch dropdown list, click on the ***3 Branches*** button next to the dropdown and you should be able to see it there. Alternatively, you can go to ***github.com/<github userid>/calc/tree/dev*** in the browser.)

**A screenshot of a computer

Description automatically generated**

**A screenshot of a chat

Description automatically generated**

2. Click on the file name to open the file in the browser. While you have the file open there, click on the *Blame* button in the gray bar at the top to see additional information about who made changes to the content.

**A screenshot of a computer program

Description automatically generated**

3. Also, click on the *History* button (upper right) to see the change history for the file.

**A screenshot of a computer

Description automatically generated**4. In the history screen, click on the commit message for your change. You’ll then be able to see the differences introduced by your commit.

**A screenshot of a computer

Description automatically generated**

# A screenshot of a computer Description automatically generated

# 5. Let’s now merge our change from the dev branch to main via a pull request. Switch back to the terminal where you did the commit and push.

# In the output from the push, you should see a link (*highlighted in the screenshot below*). Right click and open that link. (Alternatively, you can go back to the main page of your repo and if you see a message there that looks like the second picture below, you can just click on the *Compare & pull request* button.)

# A screenshot of a computer program Description automatically generated

# -- OR –

# A screenshot of a computer Description automatically generated

# 6. Depending on which option you chose in the step above, you may either be on a *Comparing Changes* screen or *Open a pull request* screen. In either case, we need to update the base repository in the gray bar at the top to make the merge go to your repo and NOT to *skillrepos/calc*. Click on the dropdown (small downward pointing arrow) and select your repo from the list.

# A screenshot of a chat Description automatically generated

# 7. After making that change, the gray bar showing the base and compare should look like the screenshot below.

# A screenshot of a phone Description automatically generated

# 8. Now, with your repo selected for the base, add an optional description if you want and then click on the *Create pull request* button.

# A screenshot of a computer Description automatically generated

# 9. At this point, you have created a new pull request. (Note that the *Pull Requests* tab at the top shows 1 pull request in the repo.) It will check for any conflicts for merging.

# We haven’t set up any CI processes or reviewers so there is nothing for those sections. Note the check in the middle section that says *This branch has no conflicts with the base branch*. You can look at the *Commits* or *File Changed* tabs if you want to see more details on the changes.

# A screenshot of a computer Description automatically generated

# 10. When you’re ready, switch back to the *Conversations* tab. Then click on the *Merge pull request* button and then the *Confirm merge button* to complete the pull request. After that, the pull request will be completed and closed (shown in second screenshot). Afterwards, you can click on the button to delete the *dev* branch if you want.

# A screenshot of a computer Description automatically generated

# A screenshot of a computer Description automatically generated

END OF LAB

# Lab 3: Creating GitHub issues

# *Purpose: In this lab, you’ll create an issue, assign it to a user, and add labels for it.*

1.We’d like to have a *README* file in our project to make it more standard. So, let’s create an issue to document that. First, ensure that the repository has the *Issues* feature turned on. On the main repo page, go to the repository’s ***Settings*** tab, and then scroll down until you see the ***Features*** section. Then, check the box for ***Issues***.



…

Graphical user interface, text, application, email

Description automatically generated

2. Now, click on the ***Issues*** tab at the top of the repository page, then the ***New issue*** button on the right. Then fill in the title with something like “*Needs README”.* For the description,you can enter something like “Please add a README file :book:”. (:book: will be changed to an emoji.) Then click the ***Submit new issue*** button.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

3. Take note of what number is assigned to the issue – you will need it later. (It will probably be #2 for you)

A screenshot of a computer

Description automatically generated

# 4. Assign the issue to yourself by clicking on the *Assign yourself* link under the *Assignees* section on the right.

# A screenshot of a chat Description automatically generated

# 5. Add the documentation label to the issue by clicking on *Labels* and selecting the *Documentation* one.

# A screenshot of a computer Description automatically generated

# 6. After this, if you click on the *Issues* tab at the top, and look at your issue, it should look like the following.

# A screenshot of a computer Description automatically generated

7. In preparation for the next lab, we need to add your second GitHub userid as a *collaborator* to this repository. Go to the repository’s ***Settings*** tab and then select ***Collaborators*** on the left under ***Access***. Then click the ***Add people*** button.

Graphical user interface, application, email

Description automatically generated

8. In the dialog box that pops up, enter the other GitHub userid you have and then click on the specific id or click on ***Select a collaborator above***. Then, click on ***Add <userid> to this repository***. That userid should then receive an email with the invite which you can accept**.**

Graphical user interface, text, application

Description automatically generated A screenshot of a computer

Description automatically generated

9. **Make sure to respond to the email and accept the invitation!** (You will need to sign in as the invited id in a different browser or a private tab or sign out/sign in, and then view and accept the invitation.). If you sign in as the secondary id and go to [*https://github.com/<primary*](https://github.com/%3cprimary) *github userid>/calc* you can also view the invitation via clicking on the button.

A screenshot of a computer

Description automatically generated

A screenshot of a web page

Description automatically generated

A screenshot of a computer

Description automatically generated

END OF LAB

# Lab 4: Setting up a pull request with reviewers

# *Purpose: In this lab, you’ll use a pull request with a reviewer and an associated issue to make a change.*

# 1. Now, we’ll address adding the README itself per the issue we previously created. If you’re not signed in as your original/primary GitHub userid, sign in as that id now. In the *Code* tab of the *calc* repository, click on the green button to add a README.md file.

# A screenshot of a computer Description automatically generated

# 2. This will bring up the editor in GitHub. Enter the text below in the new file text input area for README.md. Fill in your github userid in both places instead of github-userid. (Notes: Do this on a single line. Also, there is no space between the “]” and “(“. And since we don’t have a calculator emoji, we’re using an abacus emoji. Finally, if you cut and paste from this doc, that may add an image link at the end of the line that has to be removed.)

# This is a simple calculator :abacus: program. :question: can be directed to [@github-userid](<https://github.com/github-userid>)

# A screenshot of a computer Description automatically generated

# 3. Click on the Preview tab (next to Edit) to see how this will render once committed.

# A screenshot of a computer Description automatically generated

4. Now let’s commit these changes to a new branch and open a pull request to merge them. click on the green ***Commit changes…*** button in the upper right corner. In the dialog, enter a comment if you want and select the option to ***Create a new branch…*** You can change the generated branch name if you want. In this case, I’ve changed it to “standards”. Then click ***Propose changes***.

 A screenshot of a computer

Description automatically generated

5. At this point, you’ll see a screen showing you the changes and what’s being compared a t the top. This should only be branches in the same repo, not different repos. It should also show a green checkmark with “*Able to merge*.” next to it. We’re going to create a pull request to be reviewed. Click on the ***Create pull request*** button.

A screenshot of a computer

Description automatically generated

6. You’ll now be on the screen to create the pull request. Let’s add your secondary GitHub id as a reviewer. In the upper right, click on the ***Reviewers*** link, then select your other id from the list. (You can just make sure it’s checked and hit ESC or type it into the field.) Make sure your other userid shows up in the Reviewers section now.

A screenshot of a computer

Description automatically generated

7. Also, we can add in a description that will automatically close the associated issue when we resolve this pull request. Click in the “Add your description here…” field and enter

**Resolves #2**

**If you have a different issue number, change the 2 to your issue number.**

Then click on the “Create pull request” button.

A screenshot of a computer

Description automatically generated

8. Afterwards, you’ll be on the screen for the open pull request. Around the middle of the screen, you can see the conditions that need to be satisfied before the pull request can be merged. This includes the pending review you have from your secondary GitHub userid.

**A screenshot of a computer

Description automatically generated**

END OF LAB

# Lab 5: Completing a pull request with reviewers

# *Purpose: In this lab, we’ll complete the pull request we started in the last lab.*

1. In a separate browser or a private tab, log in to your secondary GitHub userid (the one you added as a collaborator and a reviewer). After you log in, you can either go to your notifications to see the item about the requested review or go to <https://github.com/pulls/review-requested> . Then click on the commit message for the pull request.

A screenshot of a computer

Description automatically generated

**- OR -**

A screenshot of a computer

Description automatically generated

2. This will open up the pull request. There is a button at the top to “Add your review”. Click on that.

A screenshot of a computer

Description automatically generated

3. We could click on any of the lines and add a comment if we wanted, but since this is simply adding a README file, it looks ok. However, since this is about standards, let’s make a suggestion to also add a license for the repo. Select the ***Review changes*** button and add a comment to that effect. Then select the ***Approve*** option, and then ***Submit review***.

A screenshot of a computer

Description automatically generated

4. Go to the session with your original GitHub userid or log out of the other one and log back in if you need to. Go to the ***Pull requests*** menu at the top, find the pull request and click on the commit message. Then you should see a screen like below.

A screenshot of a computer

Description automatically generated

5. Since there was a suggestion to add a license file, that sounds like a good idea, so let’s do that. Click on the ***Code*** tab at the top, then select the ***standards*** branch from the branch dropdown, then select the “***+***” sign and the option to ***+ Create new file***.

A screenshot of a computer

Description automatically generated

6. In the next screen, there will be a text entry area for the name of the file. Type in “LICENSE” for the name. Then, an option will display that says ***Choose a license template***. Click on that option. You will be asked about discarding changes. It’s ok in this case, so click on “OK”.

A screenshot of a chat

Description automatically generated

7. On the next screen, you’ll be able to pick the license you want. You can select the “MIT License” or another one if you prefer. Once done, click the ***Review and submit*** button on the right.

A screenshot of a computer

Description automatically generated

You’ll have an opportunity to review the license. When ready, just click on the ***Commit Changes*** buttons to commit the file to the *standards* branch. Be sure to leave it on the *standards* branch so it will be added to the existing pull request.

A screenshot of a computer

Description automatically generated

8. Go back to the pull request by selecting ***Pull requests*** at the top and selecting the one open pull request. You can look at the changes currently in the pull request by clicking on the ***Commits*** tab and also the ***Files changed*** tab.

A screenshot of a computer

Description automatically generated

9. Click back on the ***Conversation*** tab in the pull request and go ahead and **merge and close** (confirm merge) the pull request. After completing the merge, you should be able to click on the ***Issues*** tab and see that your issue has been automatically closed. You can click on the ***Closed*** list and then open the issue to see the automatically generated log of comments and actions if you want.

A screenshot of a computer

Description automatically generated

END OF LAB

That’s all - THANKS!

**Other options for making changes in repo vs https (if the https approach doesn’t work for you) –** choose one of A,B, or C if and only if the https push did not seem to work…

**A. Reseting credential helpers:** Especially on Windows, if you are pasting in your token for the password, but still getting an error message referencing password authentication, you may be running into issues because you have previous credentials stored in the *credential helper*.

One of the things you can try in this case is resetting the stored credentials via:

**$ git config --global credential.helper store**

Then you do your push as per the lab. It will probably pop up a text entry box for you to add your username in and another to paste in your password (PAT) and then will replace your credentials with those and complete the push.

(Note: If you prefer to disable the global credentials helper entirely, you can try

**$ git config --unset --system credentials.helper**

This may or may not work depending on if you have access to do this.)

**B. SSH keys:** If you are familiar with using ssh and have keys, you can add them into GitHub and use those. Ref <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account> for more details.

If you go this route, when you get the remote URL from the browser, select the SSH tab.

A screenshot of a computer

Description automatically generated

**C. Commit directly in GitHub:** Another option is to commit directly to GitHub in the browser. To do this, first create a *dev* branch in the repo. Clic on the branch dropdown under the title of the repo. In the *Find or create a branch* field, type **dev**. Then click on ***Create branch dev from main***.

A screenshot of a computer

Description automatically generated

In the *dev* branch, click on the *calc.html* file and open it up.

A screenshot of a computer

Description automatically generated

Click on the pencil icon to edit the file.

A screenshot of a computer

Description automatically generated

Make the changes noted in Lab 1 in the file.

When done editing, click on the ***Commit changes…*** button in the upper left, then in the dialog that comes up, you can leave all the options as they are, and then click on the ***Commit changes*** button to commit/push the file.

A screenshot of a computer

Description automatically generated