

GitHub Fundamentals BootCamp Labs

Learn the complete GitHub – from code management to Copilot

Revision 1.1 (custom) – 10/02/24

Tech Skills Transformations LLC / Brent Laster

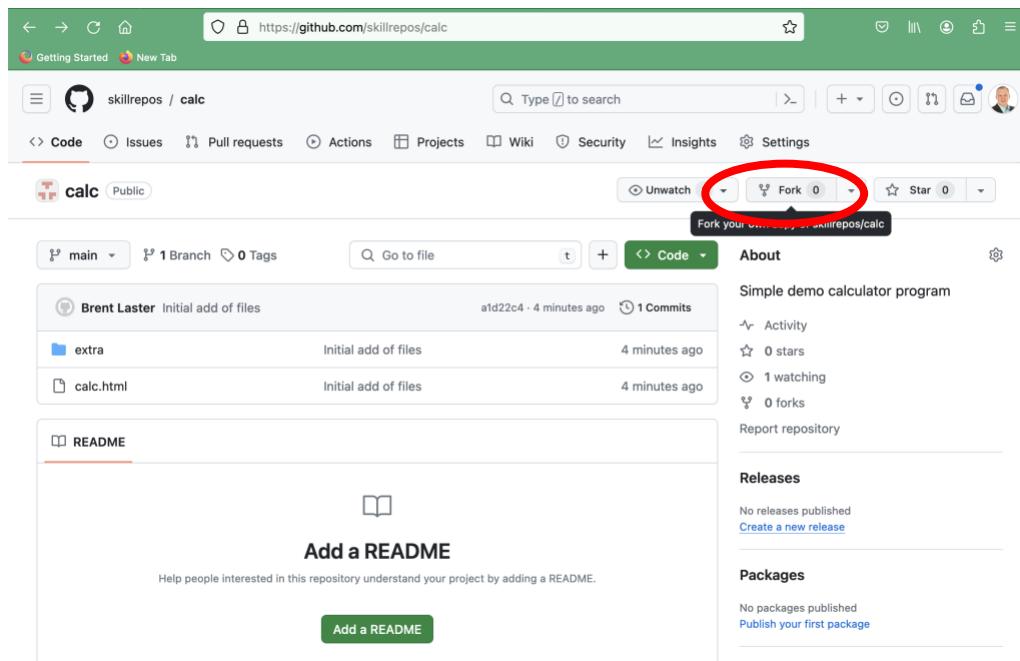
Setup and prerequisites

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](#) 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.



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * brentlaster **Repository name *** calc
 calc is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)
 Simple demo calculator program

Copy the main branch only
 Contribute back to skillrepos/calc by adding your own branch. [Learn more.](#)

ⓘ You are creating a fork in your personal account.

Create fork

- 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.

brentlaster / calc

Code 1 Pull requests Actions Projects Wiki Security Insights Settings

calc Public
forked from [skillrepos/calc](#)

1 Branch 0 Tags

This branch is up to date with [skillrepos/calc:main](#)

Brent Laster Initial add of files

extra

calc.html

README

Go to file +2 <> Code 2 Local Codespaces

Clone

HTTPS 4 SSH GitHub CLI Copy url to clipboard 5

https://github.com/brentlaster/calc.git

Open with GitHub Desktop

Download ZIP

- 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 hit Enter/Return. Then change into the subdirectory that was created from the clone.

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

```
$ cd calc
```

- 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>
```

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

```
$ git remote -v
```

- 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>
      to
<title> name's Calc</title>
```

substituting in your name (or some other text) for "name's".

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

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

- 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>

(Shortcut to token page is <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.**

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

GitHub Fundamentals class

What's this token for?

Expiration *

30 days The token will expire on Mon, Jan 22 2024

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry

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

read:ssh_signing_key Read public user SSH signing keys

Generate token **Cancel**

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

The screenshot shows the GitHub 'Personal access tokens (classic)' page. On the left, there's a sidebar with options like 'GitHub Apps', 'OAuth Apps', 'Personal access tokens' (which is selected and has a 'Beta' badge), 'Fine-grained tokens', and 'Tokens (classic)'. The main area displays a message: 'Tokens you have generated that can be used to access the [GitHub API](#).'. Below this is a light blue box containing the text: 'Make sure to copy your personal access token now. You won't be able to see it again.' A 'Copied!' button is visible above a green box containing a copied token: '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'. There are two checkmarks next to the token.

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.

\$ 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.



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.

The screenshot shows a terminal window with the following command history:

```
developer@Bs-MacBook-Pro calc % vi calc.html
developer@Bs-MacBook-Pro calc % git commit -am "Updating title"
[main d9e79db] Updating title
 1 file changed, 2 insertions(+), 2 deletions(-)
developer@Bs-MacBook-Pro calc % git push -u origin main
Username for 'https://github.com': brentlaster
Password for 'https://brentlaster@github.com':
```

A context menu is open over the password field, with the 'Paste' option highlighted in blue.

NOTE: If you 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.

- 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 only see 2 branches, refresh the page in the browser. (If you still don't see **dev** listed in the branch dropdown, 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.)

Switch branches/tags

Find or create a branch...

Branches Tags

main default

cspace

✓ dev

View all branches

calc:main .

61e42da · 8 hours ago 3 Commits

Updating title 8 hours ago

Code

This branch is 1 commit ahead of skillrepos/calc:main .

Brent Laster Updating title 61e42da · 8 hours ago 3 Commits

calc.html Updating title 8 hours ago

calc.html

README

Code

- 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 when changes were made to the content. Hovering over an entry for the commit message will produce a pop-up that shows who made the change. (Clicking on the entry will take you to a detailed view of the change.)

calc / calc.html

Brent Laster updating title

Code Blame **1** **updating title** +2 -2

Older 9 months ago 11 minutes ago 9 months ago

Brent Laster committed 11 minutes ago

2 **updating title** Initial add of files

```
<title>Brent's Calc</title>
<script language=javascript type="t
var plus,minus,divide,multiply
function initialize(){
plus=document.calc.operator.option
```

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

calc / calc.html

Brent Laster Updating title d9e79db · 24 minutes ago

Code Blame 59 lines (46 loc) · 1.29 KB

Older Newer

History

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.

Commits

History for **calc / calc.html** on **dev**

All users All time

- o Commits on Dec 31, 2023
- Updating title** Brent Laster committed 26 minutes ago Updating title d9e79db
- o Commits on Dec 23, 2023
- Initial add of files** Brent Laster committed last week a1d22c4
- o End of commit history for this file

Updating title

Brent Laster committed 28 minutes ago

1 parent a1d22c4 commit d9e79db

Showing 1 changed file with 2 additions and 2 deletions.

Whitespace Ignore whitespace Split Unified

```

 4 calc.html
@@ -1,6 +1,6 @@
 1 <html>
 2 <head>
 3 - <title>Calc</title>
 3 + <title>brentlaster's Calc</title>
 4 <script language=javascript type="text/javascript">
 5
 6
@@ -56,4 +56,4 @@
 56 <body>
 57 </html>
 58 - </body>
 59 + </html>

```

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*). Highlight/select the link and then right-click and open the 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.)

```

Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local
remote:
remote: Create a pull request for 'dev' on GitHub by visiting
remote:     https://github.com/brentlaster/calc/pull/new/dev
remote:
To https://github.com/brentlaster/calc.git
 * [new branch]      main -> dev
branch 'main' set up to track 'origin/dev'.

```

- Quick Look Link
- Open Link**
- Search with Google
- Copy
- Paste
- Mark
- Mark as Bookmark
- Unmark
- Show Inspector

-- OR --

brentlaster / calc

Type ⌘ to search

Code Pull requests Actions Projects Wiki Security Insights Settings

calc Public

forked from [skillrepos/calc](#)

Pin Watch 0

dev had recent pushes 26 minutes ago

Compare & pull request

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) in the "base repository" box, and select **your repo** from the list. (Hint: You can type your github userid in the "Filter repos" text box to quickly find your repository and then click on it.)

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). Learn more about diff comparison

base repository: skillrepos/calc base: main head repository: brentlaster/calc compare: dev Able to merge. These branches can be automatically merged.

Choose a Base Repository

brentlaster/ brentlaster/calc Add a description

Reviewers No reviews

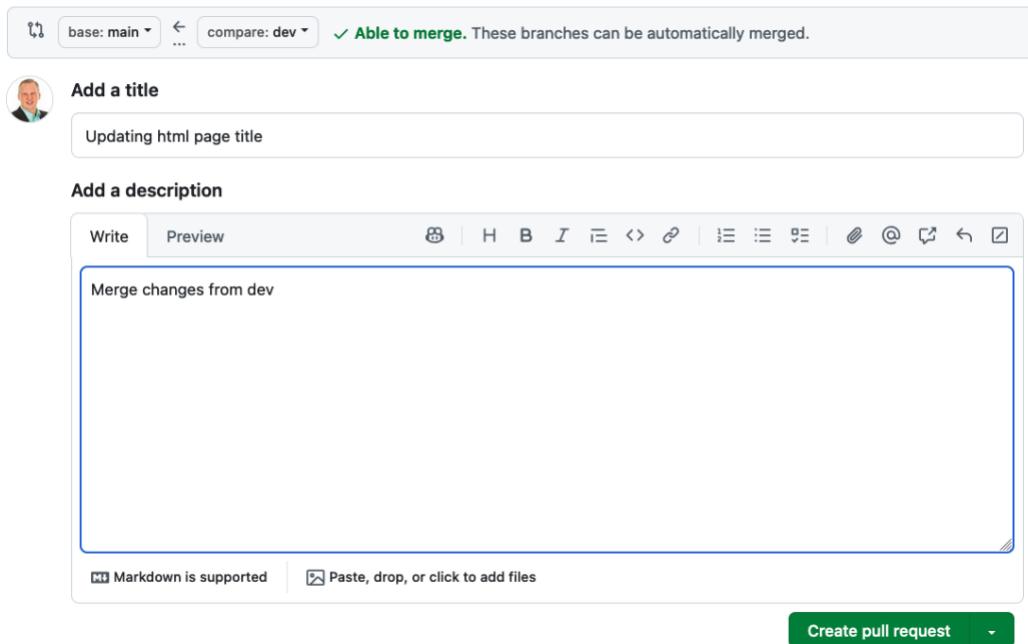
Assignees

7. After making that change, the gray bar showing the base and compare should look like the screenshot below. Notice there are only the branch names showing for the "base" and "compare" items.

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#)

base: main compare: dev Able to merge. These branches can be automatically merged.

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



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*. Look at the *Commits* and *Files Changed* tabs if you want to see more details on the changes.

10. When you're ready, switch back to the ***Conversation*** 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 third screenshot). Afterwards, you can click on the button to delete the ***dev*** branch if you want.

The image consists of three vertically stacked screenshots from a GitHub pull request interface.

Screenshot 1 (Left): Shows the pull request details. The title is "brentlaster wants to merge 1 commit into `main` from `dev`". The conversation tab is selected. A comment from "brentlaster" says "Merge changes from dev". Below it, a note says "Updating title". On the left, there's a sidebar with a green "Merge pull request" button circled in red.

Screenshot 2 (Middle): Shows the "Merge pull request" dialog. It contains the same merge details. The "Merge pull request #1 from brentlaster/dev" section is visible. At the bottom, a "Confirm merge" button is circled in red.

Screenshot 3 (Bottom): Shows the merged commit details. The title is "Merged Update html page title #1". It shows "brentlaster merged 1 commit into `main` from `dev`". The commit message is "Merge changes from dev". The commit hash is "338929b". Below it, a note says "getskillsnow merged commit `7559fe5` into `main` now". A "Revert" button is shown. At the bottom, a purple box says "Pull request successfully merged and closed" and "You're all set—the `dev` branch can be safely deleted.", with a "Delete branch" button.

11. 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.

The screenshot shows the 'Who has access' section of a GitHub repository settings page. On the left, a sidebar lists various access categories: General, Access (with Collaborators highlighted and circled with red number 2), Code and automation, Security, and others. The main area displays that it's a public repository with 0 collaborators. A large button labeled 'Add people' is prominently displayed at the bottom right of this section, also circled with red number 3.

12. 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.

The screenshot shows a modal dialog titled 'Add a collaborator to calc'. It contains a search bar with the text 'gwstudent2' and a list item 'gwstudent2 Invite collaborator'. Below the list is a green button labeled 'Select a collaborator above'.

9. In a different browser or a private tab of the same browser, sign in as your *secondary* GitHub userid.

From here, there are several ways to respond to the top item there. You can click through the link in the email. Or you can go to your notifications and click on the first item there (first screenshot below).

are multiple ways to see the invitation and respond. You can click on the link in the email. Or you can go to <https://github.com/<primary github userid>/calc> and view the invitation via clicking on the button.

Regardless of which method you choose, **make sure to respond and accept the invitation!**

The image contains three screenshots of the GitHub interface:

- Screenshot 1: Notifications**: Shows the GitHub Notifications page with a red circle around the 'Accept' button in the top right corner of the notification card for the 'brentlaster/calc' repository.
- Screenshot 2: Repository Page**: Shows the 'brentlaster / calc' repository page. A red circle highlights the 'View invitation' button in the blue header bar.
- Screenshot 3: Invitation Confirmation**: Shows the repository page again, but now displays a message: 'You now have push access to the brentlaster/calc repository.' A red circle highlights this confirmation message.

END OF LAB

© 2024 Tech Skills Transformations LLC & Brent Laster

Lab 3: Setting up a pull request with reviewers

Purpose: In this lab, you'll use a pull request with a reviewer to make a change.

1. Let's say we want to add a README file to our repository. 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.

The screenshot shows a GitHub repository named 'calc'. The 'Code' tab is selected. At the top, there is a message: 'This branch is 2 commits ahead of skillrepos/calc:main'. Below this, there is a list of recent commits:

- brentlaster Merge pull request #1 from br... dc98b08 · yesterday 3 Commits
- extra Initial add of files last week
- calc.html Updating title yesterday

In the bottom right corner of the main content area, there is a section titled 'Add a README' with the sub-instruction: 'Help people interested in this repository understand your project by adding a README.' A large green button labeled 'Add a README' is present, and it is circled in red.

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>)

Code Issues Pull requests Actions Projects Wiki Security Insights

calc / README.md in main

Cancel changes Commit changes...

Edit Preview Spaces 2 Soft wrap

```
1 This is a simple calculator :abacus: program. :question: can be directed to @brentlaster
2
```

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

Code Issues Pull requests Actions Projects Wiki Security Insights

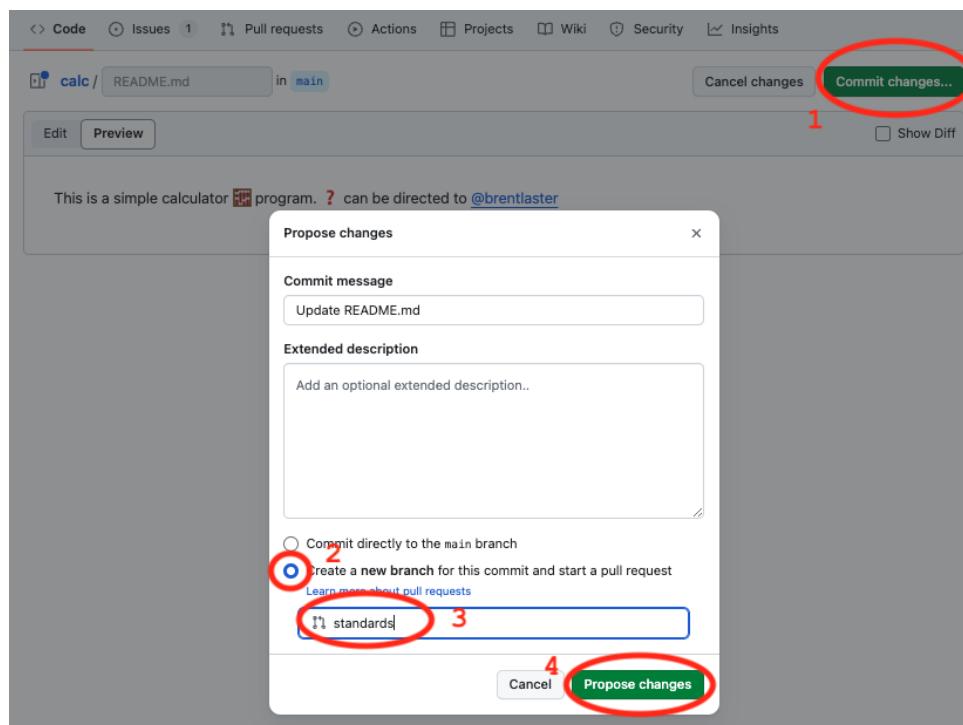
calc / README.md in main

Cancel changes Commit changes...

Edit Preview Show Diff

```
This is a simple calculator :abacus: program. ? can be directed to @brentlaster
```

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 different commit message 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**.



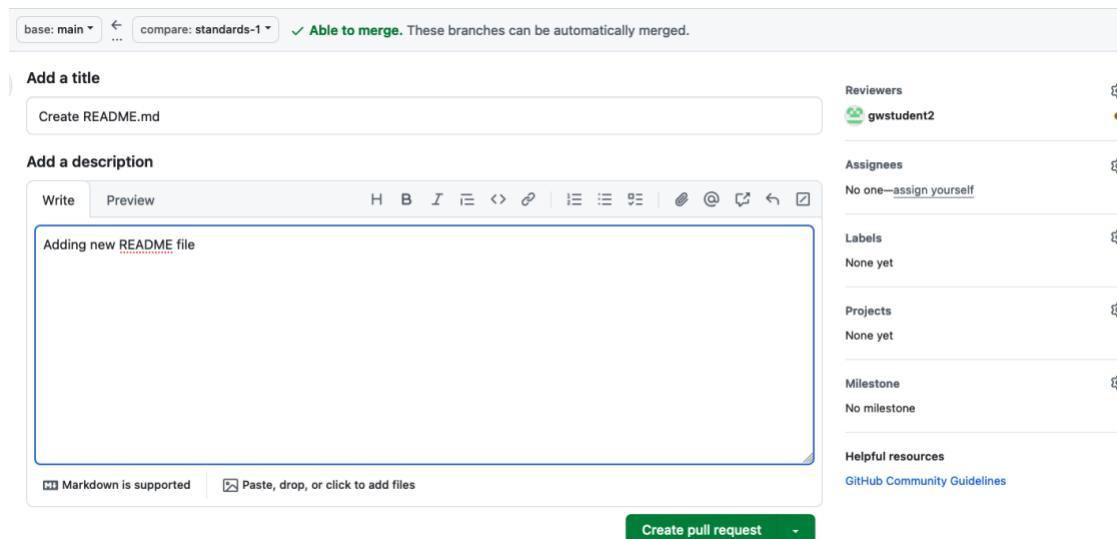
5. At this point, you'll see a screen showing you the changes and what's being compared at the top. This should only be branches in the same repo, not different repos. It should also show a green checkmark with "Able to merge." next to it. We're going to create a pull request to be reviewed. Click on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

The screenshot shows the GitHub interface for comparing changes between two branches. At the top, there are navigation links: Code, Issues (1), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below these, a header says "Comparing changes" and provides instructions to choose two branches or start a new pull request. A green banner at the top indicates that the branches are "Able to merge." and can be automatically merged. The main area displays commit details: 1 commit, 1 file changed, and 1 contributor. A specific commit for "Create README.md" by "brentlaster" is shown, with a "Verified" badge and a commit hash "47b11e1". Below the commit, a diff view shows a single addition to "README.md": "This is a simple calculator :abacus: program. :question: can be directed to [@brentlaster] (<https://github.com/brentlaster>)". A "Create pull request" button is located in the top right corner of the comparison summary, and it is highlighted with a red circle.

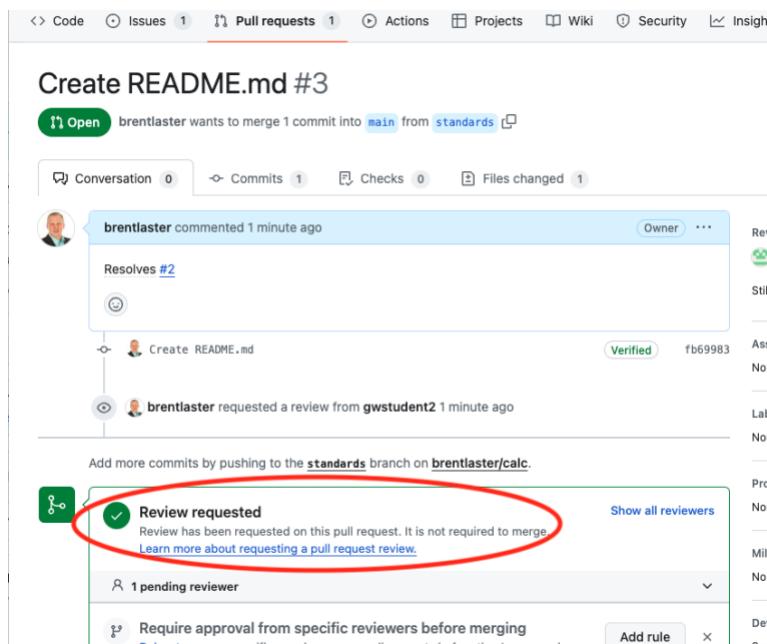
6. You'll now be on the screen to create the pull request. Let's add your secondary GitHub id as a reviewer. In the upper right, click on the **Reviewers** link, then select your other id from the list. (You can just make sure it's checked and hit ESC or type it into the field.) Make sure your other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

The screenshot shows the "Open a pull request" page. At the top, there are fields for "base: main" and "compare: standards", with a green "Able to merge" banner indicating mergeability. Below these, there are sections for "Add a title" (with a placeholder "Create README.md") and "Add a description" (with a rich text editor and a "Markdown is supported" note). To the right, there are configuration options: "Reviewers" (set to 1, with "gwstudent2" selected and highlighted with a red circle), "Request up to 15 reviewers", "Type or choose a user" input field, "Labels" (set to 2, with "gwstudent2" selected and highlighted with a red circle), "None yet" under "Projects", "None yet" under "Milestone", and "Development" with a note about closing keywords. A "Create pull request" button is at the bottom.

7. Add in a description if you and click on the “Create pull request” button.



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.



END OF LAB

Lab 4: Completing a pull request with reviewers

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

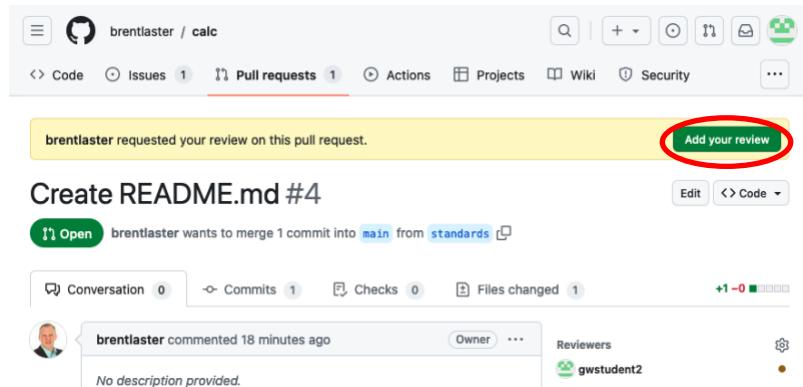
1. In a separate browser or a private tab, log in to your secondary GitHub userid (the one you added as a collaborator and a reviewer). After you log in, you can either go to your notifications to see the item about the requested review or go to <https://github.com/pulls/review-requested>. Then click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

The screenshot shows the GitHub Notifications interface. At the top, there's a header with a search bar, a plus sign button, and a mail icon circled in red with the number '1'. Below the header, a blue bar indicates 'Inbox' with '1' notification. The main area shows a 'Clear out the clutter.' card with a 'Get started' button. On the left, there are filters: 'Saved' (unchecked), 'Done' (checked), 'Assigned' (1 notification), 'Participating' (1 notification), 'Mentioned' (1 notification), 'Team mentioned' (1 notification), and 'Review requested' (1 notification). The 'Review requested' section has a notification for 'brentlaster/calc #4 Create README.md' circled in red with the number '2'. A 'ProTip!' message suggests creating custom filters. At the bottom, there's a 'Repositories' section for 'brentlaster/calc' with '1' repository.

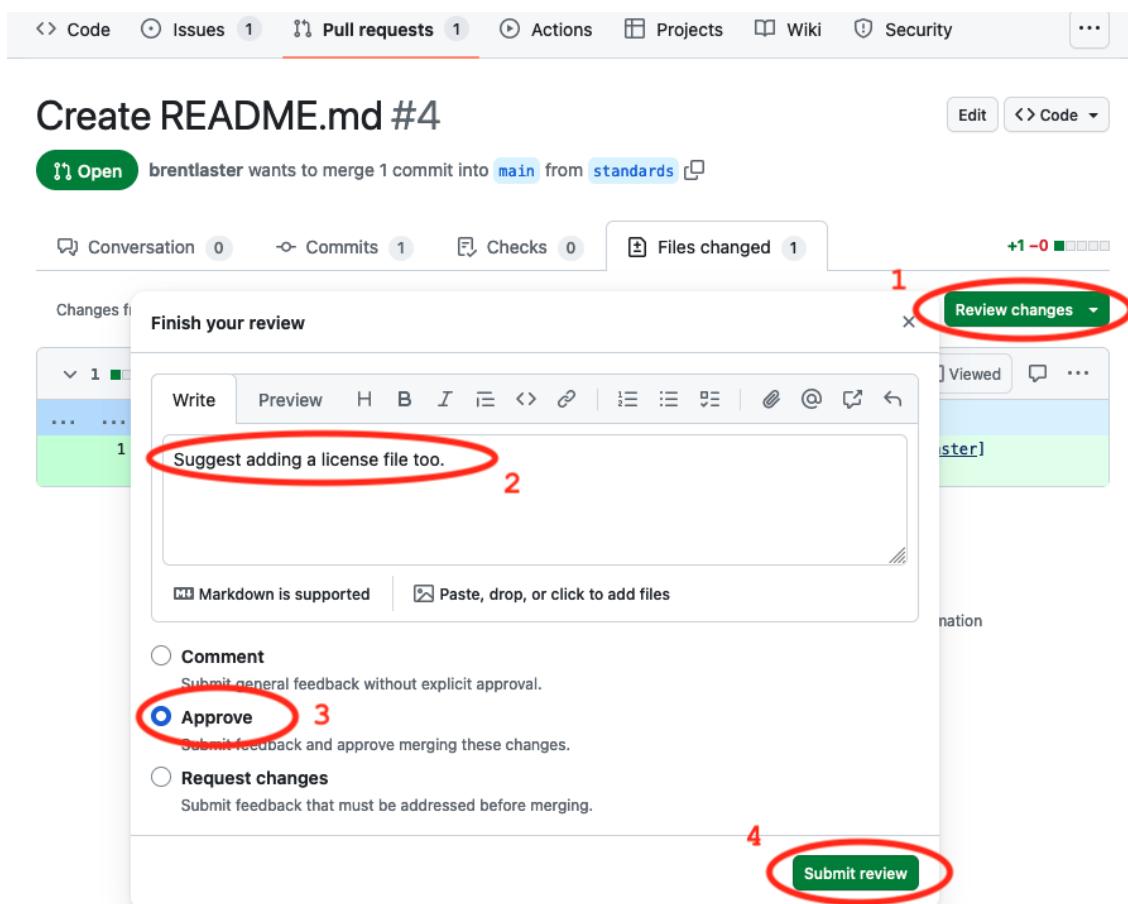
- OR -

The screenshot shows a browser window with the URL 'https://github.com/pulls/review-requested' highlighted with a red circle. The page title is 'Pull Requests'. The top navigation bar includes 'Created', 'Assigned', 'Mentioned', and 'Review requests' (which is selected). A search bar shows the query 'is:open is:pr review-requested:gwstudent2 archive'. Below the navigation, it says '1 Open' and '1 Closed'. A single pull request is listed: '#4 brentlaster/calc Create README.md' (with a circled link), which was opened 14 minutes ago by brentlaster. A 'ProTip!' message at the bottom suggests adding 'no:assignee' to see everything that's not assigned.

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



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”.



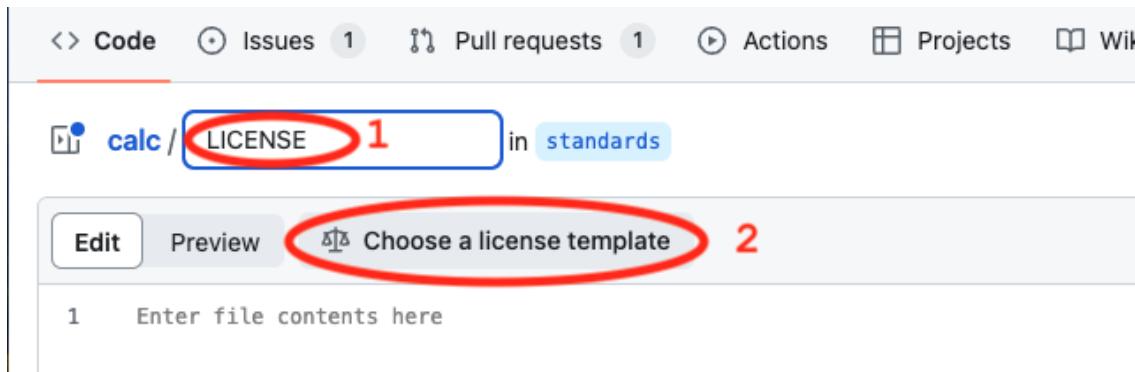
4. Go to the session with your original GitHub userid or log out of the other one and log back in if you need to. Go to the **Pull requests** menu at the top, find the pull request and click on the commit message. Then you should see a screen like below.

The screenshot shows a GitHub pull request page. At the top, the URL is github.com/brentlaster/calc/pull/4. The navigation bar includes 'Code', 'Issues 1', 'Pull requests 1', 'Actions', 'Projects', 'Wiki', and 'Security'. The main title is 'Create README.md #4'. Below it, a green button says 'Open' and indicates 'brentlaster wants to merge 1 commit into main from standards'. The commit message 'Create README.md' is shown with a 'Verified' badge and a commit hash '47b11e1'. A review from 'gwstudent2' is listed, showing they approved the changes 1 minute ago. A comment from 'gwstudent2' suggests adding a license file too.

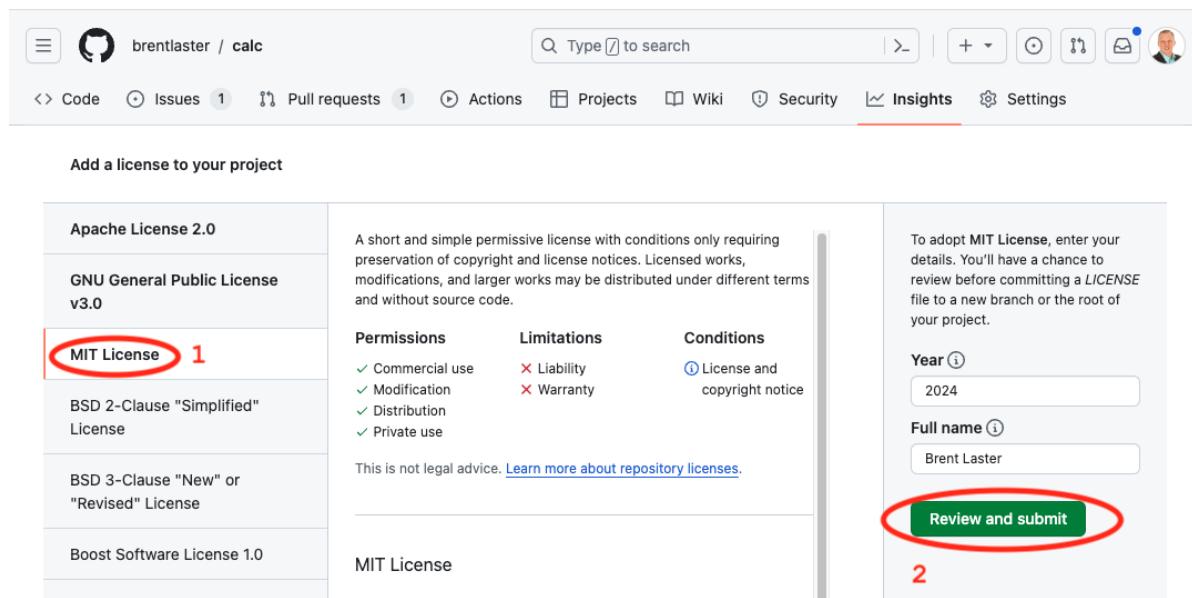
5. Since there was a suggestion to add a license file, that sounds like a good idea, so let's do that. Click on the **Code** tab at the top, then select the **standards** branch (or whatever name you gave the new branch) from the branch dropdown, then select the "+" sign (or "Add file" option) and the option to **+ Create new file**.

The screenshot shows a GitHub repository page for 'calc'. The 'Code' tab is selected (circled in red). The 'standards' branch is selected (circled in red). A context menu is open over the '+' button, with the '+ Create new file' option highlighted (circled in red). The repository details show it is public, forked from 'skillrepos/calc', and has 3 branches and 0 tags. It is 5 commits ahead of 'skillrepos/calc:main'. The commit list shows recent activity by 'brentlaster' including creating a README.md and updating the title. The repository page also includes sections for 'Rel' and 'Pac'.

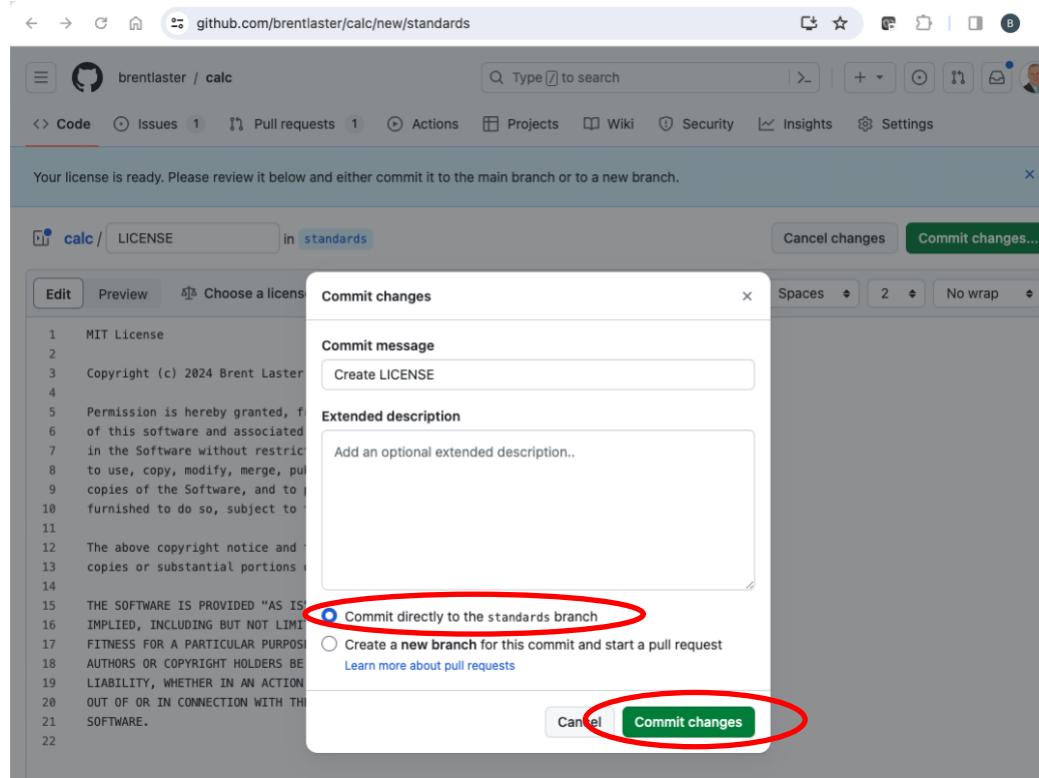
6. In the next screen, there will be a text entry area for the name of the file. Type in “LICENSE” for the name. Then, an option will display that says ***Choose a license template***. Click on that option. You will be asked about discarding changes. It’s ok in this case, so click on “OK”.



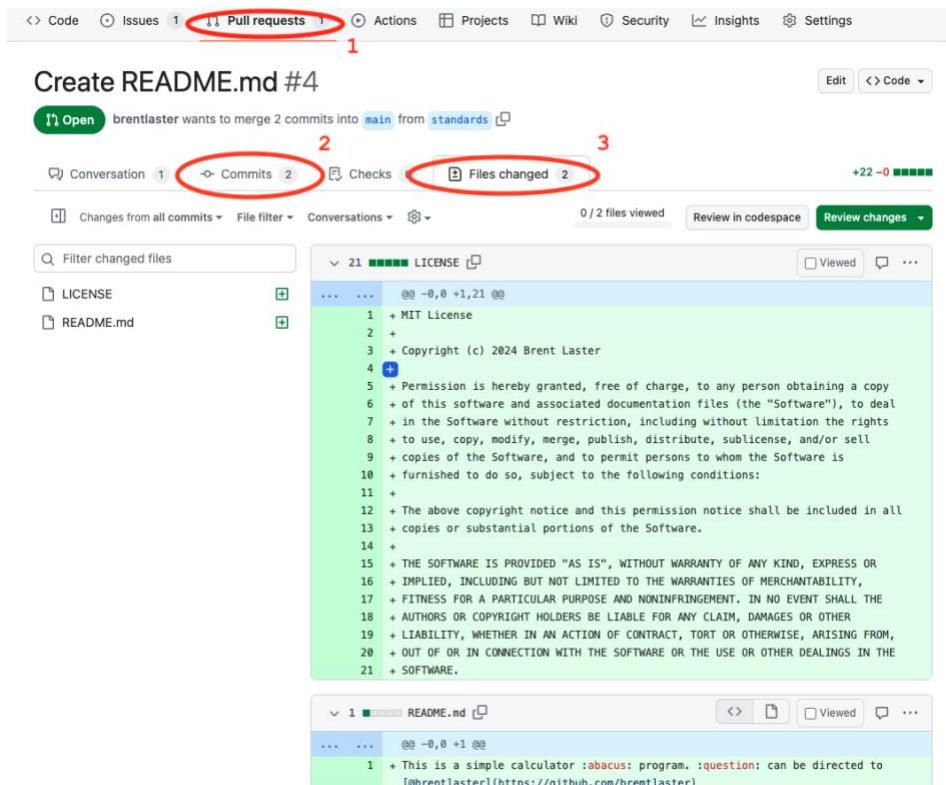
7. On the next screen, you’ll be able to pick the license you want. You can select the “MIT License” or another one if you prefer. Once done, click the “Review and submit” button on the right.



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.



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.



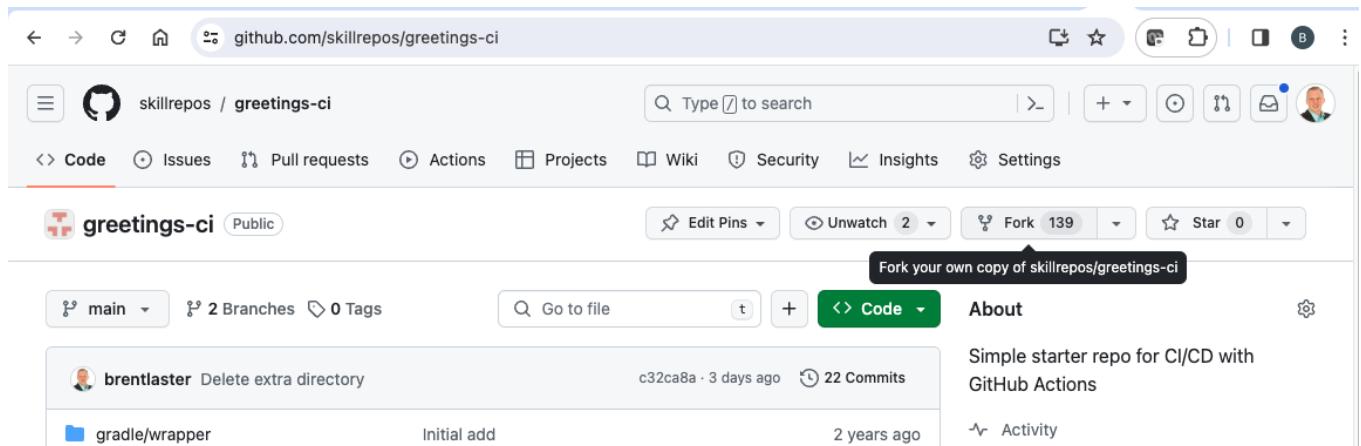
9. Click back on the **Conversation** tab in the pull request and go ahead and merge (*Merge pull request*) and close (*Confirm merge*) the pull request

END OF LAB

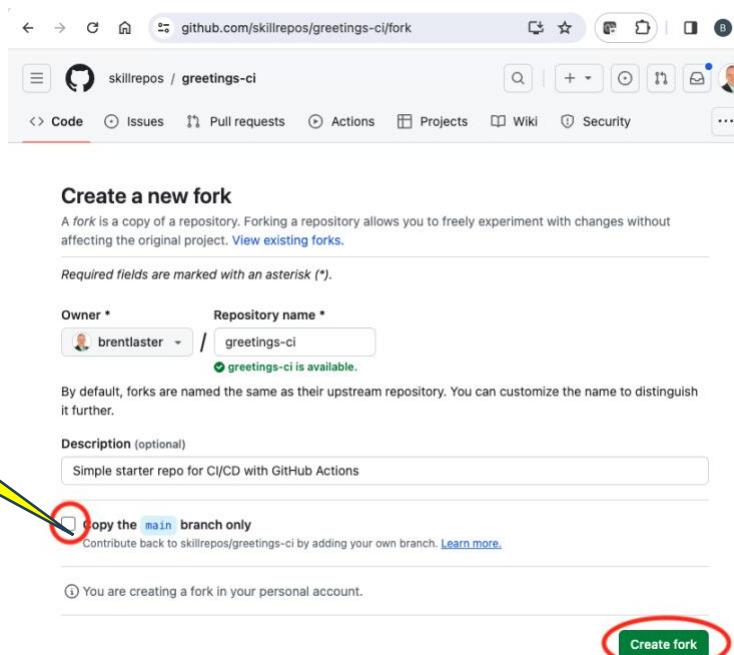
Lab 5: Learning about GitHub Actions

Purpose: In this lab, we'll learn about how GitHub Actions can be used to automate workflows for repositories.

1. Start out in GitHub with your primary GitHub account.
2. Go to <https://github.com/skillrepos/greetings-ci> and fork that project into your own GitHub space. After this, you'll be on the project in your user space. **Make sure again to uncheck the box next to *Copy the main branch only***, so that both branches will be included in the fork.



The screenshot shows the GitHub repository page for 'greetings-ci'. At the top, there's a 'Fork' button with the text 'Fork your own copy of skillrepos/greetings-ci'. Below the fork button, the repository details are shown: 'main' branch, 2 branches, 0 tags, and 22 commits from brentlaster. The 'About' section describes it as a 'Simple starter repo for CI/CD with GitHub Actions'. At the bottom of the page, there's a summary of recent activity.



The screenshot shows the 'Create a new fork' form. It includes fields for 'Owner' (set to 'brentlaster'), 'Repository name' (set to 'greetings-ci'), and a note that 'greetings-ci is available'. There's a description field containing 'Simple starter repo for CI/CD with GitHub Actions'. A yellow callout box with the word 'uncheck' points to the 'Copy the main branch only' checkbox, which is circled in red. Below the form, a note says 'You are creating a fork in your personal account.' and a green 'Create fork' button is highlighted with a red circle.

3. We have a simple java source file named `echoMsg.java` in the subdirectory `src/main/java`, a Gradle build file in the root directory named `build.gradle`, and some other supporting files. We could clone this repository and build it manually via running Gradle locally. But let's set this to build with an automatic CI process specified via a text file. On the **Code** page, click on the **Actions** tab in the top menu under the repository name.

The screenshot shows the GitHub repository page for 'gwstudent/greetings-ci'. The 'Actions' tab is highlighted with a red circle. The main content area displays a list of recent commits:

- Brent Laster add extra dir (97df205, 7 minutes ago)
- extra (add extra dir, 7 minutes ago)
- gradle/wrapper (Initial add, 14 minutes ago)
- src/main/java (Initial add, 14 minutes ago)
- build.gradle (Initial add, 14 minutes ago)
- gradlew (Initial add, 14 minutes ago)
- gradlew.bat (Initial add, 14 minutes ago)

On the right side, there are sections for 'About', 'Releases', and 'Packages'.

4. This will bring up a page with categories of starter actions that GitHub thinks might work based on the contents of the repository. We'll select a specific CI one. Scroll down to near the bottom of the page under **Browse all categories** and select **Continuous integration**.

The screenshot shows the 'Browse all categories' section. A red circle highlights the 'Continuous integration' link under the 'Automation' heading.

Automation			
Greetings By GitHub Actions Greets users who are first time contributors to the repo Configure	Stale By GitHub Actions Checks for stale issues and pull requests Configure	Manual workflow By GitHub Actions Simple workflow that is manually triggered. Configure	Labeler By GitHub Actions Labels pull requests based on the files changed Configure

Browse all categories

- Automation
- Continuous integration** (circled in red)
- Deployment
- Security

5. In the CI category page, let's search for one that will work with Gradle. Type `Gradle` in the search box and press Enter.

The screenshot shows the GitHub Actions search interface. A red circle highlights the search bar containing the text 'Gradle'. Below the search bar, a dropdown menu shows 'Continuous integration' selected. The search results section displays five workflow cards: 'Android CI', 'Java with Ant', 'Clojure', 'Publish Java Package', and 'Java with Gradle'. Each card includes a 'Configure' button.

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

[Skip this and set up a workflow yourself →](#)

This screenshot is similar to the previous one, but the 'Java with Gradle' workflow card is circled in red. The card includes a 'Configure' button.

- From the results, select the **Java with Gradle** one and click the **Configure** button to open a predefined workflow for this.

The screenshot shows the GitHub Actions configuration page for the 'Java with Gradle' workflow. The top navigation bar and search bar are identical to the previous screenshots. The main content area shows the workflow details, including steps like 'Run Java build script' and 'Publish Java package'.

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

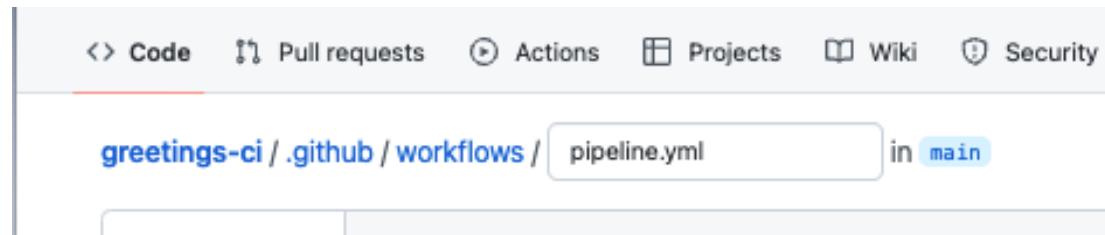
[Skip this and set up a workflow yourself →](#)

This screenshot shows the GitHub Actions search results again, but the 'Java with Gradle' workflow card is circled in red. The 'Configure' button is also circled in red.

- This will bring up a page with a starter workflow for CI that we can edit as needed. We need to make two edits here. The first edit is to change the name. In the top section where the path is, notice that there is a text entry box around `gradle.yml`. This is the current name of the workflow. Click in that box and edit the name to be `pipeline.yml`. (You can just backspace over or delete the name and type the new name.)

The screenshot shows the GitHub workflow editor for the 'gradle.yml' file. The URL in the address bar is 'greetings-ci/.github/workflows/gradle.yml'. A red box highlights the 'gradle.yml' part of the URL. Below the address bar, there are 'Edit new file' and 'Preview' buttons.

TO



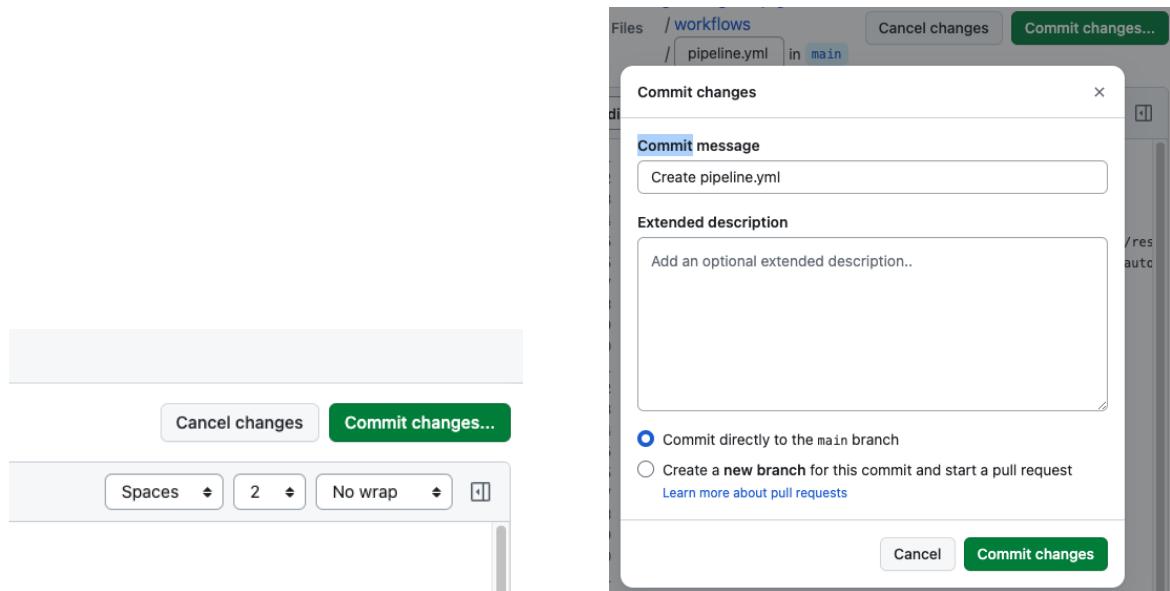
8. The second edit is to remove the second job in this workflow since it would require some additional setup. To do this we will just highlight/select the code from line 50 on and hit delete. (*If you have trouble just selecting that code, try starting at the bottom and selecting/highlighting from the bottom up.*) The code to be deleted is highlighted in the next screenshot.

```

40      # If your project does not have the Gradle Wrapper configured, you can use the following configuration to run
41      #
42      # - name: Setup Gradle
43      #   uses: gradle/actions/setup-gradle@417ae3ccd767c252f5661f1ace9f835f9654f2b5 # v3.1.0
44      #   with:
45      #     gradle-version: '8.5'
46      #
47      # - name: Build with Gradle 8.5
48      #   run: gradle build
49
50      dependency-submission:
51        runs-on: ubuntu-latest
52        permissions:
53          contents: write
54
55        steps:
56          - uses: actions/checkout@v4
57          - name: Set up JDK 17
58            uses: actions/setup-java@v4
59            with:
60              java-version: '17'
61              distribution: 'temurin'
62
63          # Generates and submits a dependency graph, enabling Dependabot Alerts for all project dependencies.
64          # See: https://github.com/gradle/actions/blob/main/dependency-submission/README.md
65          - name: Generate and submit dependency graph
66            uses: gradle/actions/dependency-submission@417ae3ccd767c252f5661f1ace9f835f9654f2b5 # v3.1.0
67
68

```

9. Now, we can go ahead and commit the new workflow via the **Commit changes...** button in the upper right. In the dialog that comes up, you can enter an optional comment if you want. Leave the **Commit directly...** selection checked and then click on the **Commit changes** button.



10. Since we've committed a new file and this workflow is now in place, the `on: push:` event is triggered and the CI automation kicks in. Click on the **Actions** menu again to see the automated processing happening. (You may have to wait a moment for it to start.)

The screenshot shows the GitHub Actions page. The left sidebar has 'Actions' selected, with 'All workflows' chosen. Under 'All workflows', 'Java CI with Gradle' is listed. The main area shows 'All workflows' with a 'Help us improve GitHub Actions' card. Below it is a '1 workflow run' card for 'Create pipeline.yml', which was triggered by a 'Java CI with Gradle #1: Commit cc93266 pushed by brentlaster' on the 'main' branch. The status is 'In progress'.

11. After a few moments, the workflow should succeed. (You may need to refresh your browser.) After it is done, you can click on the commit message for the run to get to the details for that particular run.

The screenshot shows the GitHub Actions page for 'Java CI with Gradle'. The 'Actions' sidebar has 'Java CI with Gradle' selected. The main area shows a '1 workflow run' card for 'Create pipeline.yml', which was triggered by a 'Java CI with Gradle #1: Commit cc93266 pushed by brentlaster' on the 'main' branch. The status is 'Succeeded'. A red circle highlights the commit message 'Create pipeline.yml' in the workflow run card.

12. From here, you can click on the build job in the graph or the *build* item in the list of jobs to get more details on what occurred on the runner system. You can expand any of the steps in the list to see more details.

The screenshot shows a GitHub Actions pipeline named "Create pipeline.yml #1". The "Summary" tab is selected. Under the "Jobs" section, the "build" job is highlighted with a red circle. Clicking on it expands the job details. The expanded view shows the "Set up job" step and the "Run actions/checkout@v3" step, both of which are also circled in red. The log output for the "Run actions/checkout@v3" step includes steps 1 through 19, detailing the repository sync, Git version info, and temporary HOME variable overriding.

END OF LAB

Lab 6: Creating packages

Purpose: In this lab, we'll see how to create GitHub packages.

1. We'll continue working in your fork of the **greetings-ci** repo under your primary userid. In a separate branch named **package**, we have an updated **build.gradle** file and a new Actions workflow file - **.github/workflows/publish-package.yml**. You can switch to the **package** branch and look at those if you want. (You don't need to make any changes.)

The screenshot shows the GitHub repository "greetings-ci". The "package" branch is selected. The repository has 2 branches and 0 tags. The "Switch branches/tags" dropdown shows "Find or create a branch..." and lists "main" and "package". The "package" branch is checked. The repository has 0 tags. The "Files" sidebar shows the directory structure: .github/workflows/publish-package.yml, .gradle, gradle, build.gradle, gradlew, and gradlew.bat. The right panel displays the content of the "publish-package.yml" workflow file:

```

name: Publish package to GitHub Packages
on:
  release:
    types: [created]
  workflow_dispatch:
jobs:
  publish:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
  
```

2. Let's create a pull request to merge those into *main*. Go the Pull Requests menu and open a new pull request (via the button) to merge the *package* branch into the *main* branch - **on your fork NOT skillrepos/greetings-ci**. Make sure to set the **base repository = <your repo> main** and **compare = package** in the gray bar so you are merging in the same repo and **NOT** into skillrepos/greeting-ci. After you make those changes, go ahead and create the pull request (by clicking through the *Create pull request* buttons on the screens).

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

3. Look at the pull request and review the changes we've made to publish the package via the *Commits* and *Files changed* tabs.

The screenshot shows a GitHub pull request titled "Initial add on package #1". The "Files changed" tab is selected, showing a single file: ".github/workflows/publish-package.yml". The code in the file is:

```

1  name: Publish package to GitHub Packages
2  on:
3    release:
4      types: [created]
5

```

4. Back in the ***Conversation*** tab, merge the pull request. You can choose to delete the ***package*** branch or not.

5. Open the new ***.github/workflows/publish-package.yml*** file on the ***main*** branch. Notice that it has a ***workflow-dispatch*** trigger. This allows the workflow to be invoked manually. (No changes need to be made.)

The screenshot shows the ".github/workflows/publish-package.yml" file on the "main" branch. The code in the file is:

```

1  name: Publish package to GitHub Packages
2  on:
3    release:
4      types: [created]
5    workflow_dispatch:

```

6. Switch to the ***Actions*** menu, then select the ***Publish package to GitHub Packages*** workflow on the left and select the ***Run workflow*** button that shows up in the blue bar. Leave the branch in the dialog as ***main***.

The screenshot shows the GitHub Actions interface. A red circle labeled '1' highlights the 'Actions' tab in the top navigation bar. A red circle labeled '2' highlights the 'Publish package to GitHub Packages' workflow in the left sidebar. A red circle labeled '3' highlights the 'Run workflow' button in the top right corner of the workflow run card. A red circle labeled '4' highlights the 'Run workflow' button again, this time inside the workflow run card itself.

6. After this, you can switch to the **Code** tab and you should be able to see the new package listed in the **Packages** area in the lower right of the screen. Click on the link to find out more details about it.

The screenshot shows the GitHub repository page for 'greetings-ci'. A red circle highlights the 'Code' tab in the top navigation bar. In the bottom right corner of the repository summary, there is a 'Packages' section with a red circle around the link 'org.gradle.sample.greetings-ci'. The rest of the page displays repository details like branches, commits, and releases.

7. You can also see the new package in your profile area. Click on your picture in the upper right, then select **Your profile** and then the **Packages** tab.

The screenshot shows the GitHub Packages interface for a user named brentlaster. The top navigation bar includes links for Overview, Repositories, Projects, Packages (which is highlighted with a red circle), and Stars. A large circular profile picture of Brent Laster is on the left. The main content area shows a list of packages under the heading "1 package". The first item in the list is "org.gradle.sample.greetings-ci", which is also circled in red. The right side of the screen displays a sidebar with a user profile for brentlaster, a "Your profile" section (also circled in red), and a list of links including "Your repositories", "Your projects", "Your organizations", "Your enterprises", "Your stars", "Your sponsors", and "Your dists".

8. You can also download the individual artifacts by clicking on the link to the package and then in the list of assets, clicking on individual items. Do this for the ***greetings-ci-1.1 jar*** file.

This screenshot shows the details page for the "greetings-ci-1.1" package. The top navigation bar has "Code" selected. The main content area shows the package name "org.gradle.sample.greetings-ci 1.1". Below it, there are two sections: "Install 1/2: Add this to pom.xml" containing Maven/Gradle dependency code, and "Install 2/2: Run via command line" with the command "\$ mvn install". To the right, a sidebar titled "Details" shows the author "brentlaster" and the date "January 05, 2024". The "Assets" section lists several files, with "greetings-ci-1.1.jar" circled in red at the bottom.

END OF LAB

Lab 7: Creating a release

Purpose: In this lab, we'll create a new release of our project's code.

1. On the **Code** page of the *greetings-ci* repo, on the right-hand side, find the **Releases** section and click on the **Create a new release** link. (You can also go directly to the page by going to <https://github.com/<github-userid>/greetings-ci/releases/new>.)

The screenshot shows the GitHub interface for the 'greetings-ci' repository. At the top, there's a green 'Code' button with a dropdown arrow. Below it is an 'About' section with a brief description: 'Simple starter repo for CI/CD with GitHub Actions'. To the right is a gear icon. On the left, there's a sidebar with various time-based filters like 'fork', 'commits', 'ours ago', 'nths ago', 'years ago', 'earlier', 'utes ago', 'earlier', and 'earlier'. The main content area has sections for 'Activity', '0 stars', '0 watching', '140 forks', and 'Releases'. The 'Releases' section contains the text 'No releases published' and a blue 'Create a new release' button, which is circled in red. Below this is a 'Packages' section with one entry: 'org.gradle.sample.greetings-ci'.

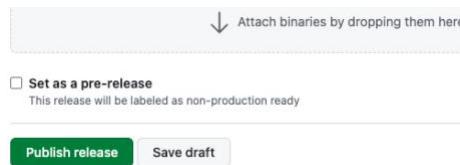
2. We need to create a tag on the repo before we create a release. Click on the **Choose a tag** dropdown and enter *v1.1* (or some other name if you prefer) for the tag name. Then click on the **+ Create new tag: v1.1 on publish** line.

The screenshot shows the GitHub 'Releases' page for the 'greetings-ci' repository. At the top, there are tabs for 'Releases' (which is active and highlighted in blue) and 'Tags'. Below the tabs are two dropdown menus: 'Choose a tag' and 'Target: main'. A modal window is open over the page, titled 'Choose a tag', with a text input field containing 'v1.1'. At the bottom of the modal, there's a button labeled '+ Create new tag: v1.1 on publish'.

3. Now let's update a file for the release. Near the bottom of the screen, drag and drop or select a file. For simplicity, just drag and drop the file you downloaded locally at the end of the last lab. This will add the *greetings-ci.jar* file to the release.

The screenshot shows the GitHub Releases interface. At the top, there are tabs for 'Releases' (selected) and 'Tags'. Below that, a dropdown shows 'v1.1' and 'Target: main'. A message says 'Excellent! This tag will be created from the target when you publish this release.' There is a 'Release title' input field, a 'Write' button, a 'Preview' button, and a rich text editor toolbar. Below the toolbar is a 'Describe this release' text area and a file attachment section with a placeholder 'Attach files by dragging & dropping, selecting or pasting them.' A red arrow points from this field to a 'Downloads' window on the right. The 'Downloads' window shows a file named 'greetings-ci-1.1.jar' with a size of '0.00 MB'. The file path is 'Dropbox/greetings-ci-1.1.jar'. The 'Favorites' section includes 'Dropbox' and 'My Drive'.

4. Click the button at the bottom of the page to publish the release.



5. After this, you'll see the published release page.

The screenshot shows a GitHub release page for a repository. At the top, there's a navigation bar with links for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and more. Below the navigation bar, it says 'Releases / Initial release'. The main title of the release is 'Initial release' with a 'Latest' button next to it. A note indicates that 'brentlaster released this now'. The description of the release is 'Initial release of content for 1.1.'. Under the 'Assets' section, there is one item listed: 'greetings-ci-1.1.jar' (1006 Bytes, 2 minutes ago), followed by two source code files ('Source code (zip)' and 'Source code (tar.gz)', both 21 minutes ago). There are also edit and delete icons for the release.

6. If you switch back to the main page of the repo, you'll see the new release under the *Releases* section on the right side of the page.

The screenshot shows the main page of a GitHub repository named 'greetings-ci'. It has a public status, was forked from 'skillrepos/greetings-ci', and has 140 forks. The repository has a 'main' branch with 2 commits ahead of 3 commits behind 'skillrepos/greetings-ci:main'. The commit list includes a merge pull request from 'brentlaster' and several initial adds for files like '.github/workflows', 'gradle/wrapper', 'src/main/java', 'build.gradle', 'gradlew', and 'gradlew.bat'. On the right side, there's an 'About' section with details like 'Simple starter repo for CI/CD with GitHub Actions', activity metrics (0 stars, 0 watching, 140 forks), and a 'Releases' section. The 'Initial release' from step 5 is circled in red, showing it was made 1 minute ago. Below it, there's a 'Packages' section listing 'org.gradle.sample.greetings-ci'.

7. You can click on that if you want to see details about the release (same as output in step 5).

END OF LAB

Demo: GitHub Command Line

That's all - THANKS!

APPENDIX

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. Resetting 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.

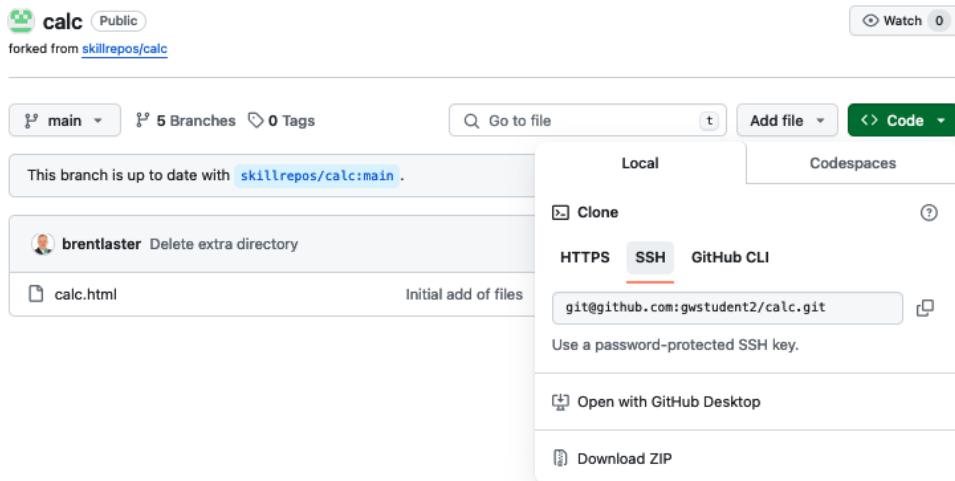
(Note: If you prefer to disable the global credentials helper entirely, you can try

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

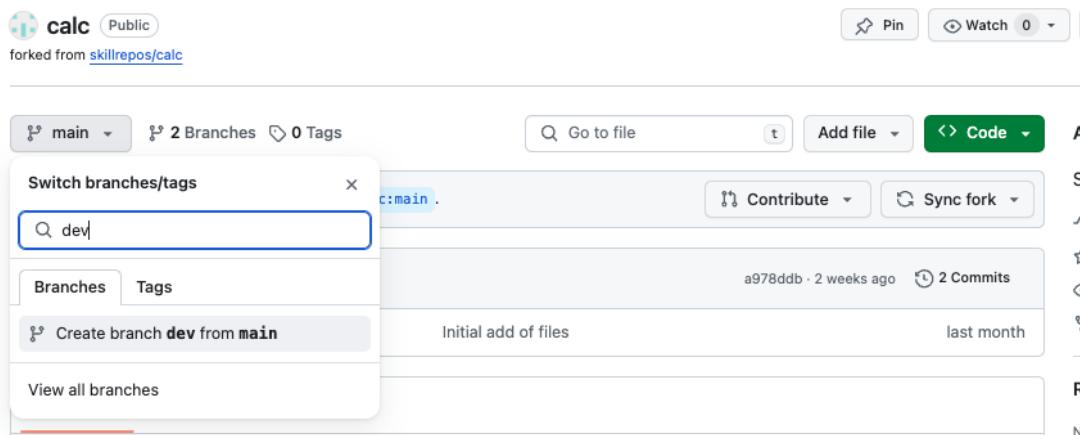
This may or may not work depending on if you have access to do this.)

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.



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. Click 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**.



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

The screenshot shows a GitHub repository page for a project named 'calc'. The repository is public and has been forked from 'skillrepos/calc'. At the top right, there are buttons for 'Pin' and 'Watch' with a count of 0. Below the repository name, it says 'forked from skillrepos/calc'. In the top navigation bar, there are dropdowns for 'dev' (selected), '3 Branches', '0 Tags', and search fields for 'Go to file' and 'Add file'. A green 'Code' button is also present. A message at the top states 'This branch is up to date with skillrepos/calc:main'. Below this, a commit by 'brentlaster' is shown: 'Delete extra directory' (commit a978ddb, 2 weeks ago, 2 commits). The commit details show 'calc.html' was added initially (last month) and 'README' was also added. The 'calc.html' file is currently selected.

Click on the pencil icon to edit the file.

The screenshot shows the 'calc.html' file editor on GitHub. The file content is as follows:

```

<html>
<head>
<title>Calc</title>

```

At the top, there are buttons for 'Code' (selected), 'Blame', and 'Raw'. To the right, there are download and copy buttons, and a prominent 'Edit this file' button. The commit history for this file shows a single commit by 'Brent Laster' titled 'Initial add of files' (commit a1d22c4, last month). The commit message includes a link to 'History'.

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.

The screenshot shows a GitHub repository page for 'gwstudent / calc'. The user is editing the file 'calc.html' in the 'dev' branch. A modal dialog titled 'Commit changes' is open, prompting the user to enter a commit message. The message 'Update calc.html' is typed into the input field. Below the message, there is an optional 'Extended description' field with placeholder text 'Add an optional extended description..'. At the bottom of the dialog, two radio button options are visible: 'Commit directly to the dev branch' (selected) and 'Create a new branch for this commit and start a pull request'. A link 'Learn more about pull requests' is provided for the second option. The GitHub interface includes standard navigation and search tools at the top.

```
1 <html>
2 <head>
3     <title>Brent's Calc</title>
4
5     <script language="javascr
6
7         var plus,minus,divide,multi
8
9         function initialize(){
10             plus=document.calc.operat
11             minus=document.calc.operat
12             divide=document.calc.operat
13             multiply=document.calc.operat
14         }
15
16         function calculate(){
17             a = parseInt(document.calc.an
18             b = parseInt(document.calc.an
19             if (plus.selected)
20                 document.calc.an
21             if (minus.selected)
22                 document.calc.an
23             if (divide.selected)
24                 document.calc.an
25             if (multiply.selected)
26                 document.calc.an
27         }

```