

# GitHub Fundamentals BootCamp Labs

*Learn the complete GitHub – from code management to Copilot*

**Revision 1.6 – 09/22/24**

Tech Skills Transformations LLC / Brent Laster

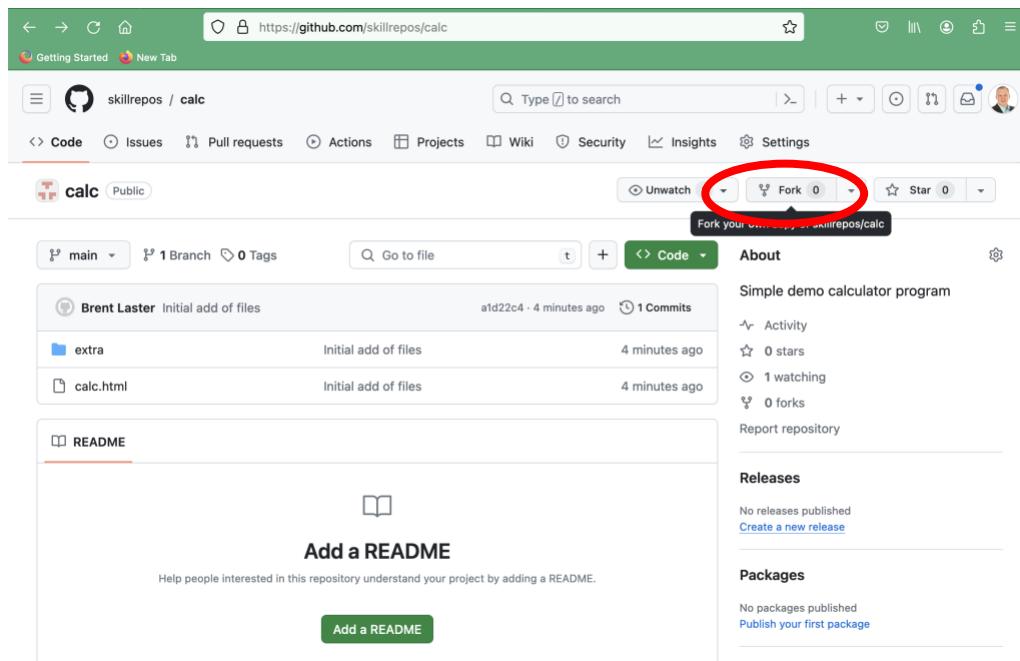
## Setup and prerequisites

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 \* Repository name \*

brentlaster / 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](#)

main 1 Branch 0 Tags

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

Brent Laster Initial add of files

extra

calc.html

README

Local 2

Clone

HTTPS 4 GitHub CLI

<https://github.com/brentlaster/calc.git> 5

Copy url to clipboard

Open with GitHub Desktop

Download ZIP

4. 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
```

5. 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>
```

6. 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>
to
<title> name's Calc</title>
```

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

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>

(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"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> <b>repo:status</b>	Access commit status
<input checked="" type="checkbox"/> <b>repo_deployment</b>	Access deployment status
<input checked="" type="checkbox"/> <b>public_repo</b>	Access public repositories
<input checked="" type="checkbox"/> <b>repo:invite</b>	Access repository invitations
<input checked="" type="checkbox"/> <b>security_events</b>	Read and write security events
<input checked="" type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input checked="" type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> <b>read:packages</b>	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>delete:packages</b>	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 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens' (which is expanded, showing 'Fine-grained tokens' and 'Tokens (classic)'). A 'Beta' button is next to 'Personal access tokens'. The main area is titled 'Personal access tokens (classic)' and contains the text 'Tokens you have generated that can be used to access the [GitHub API](#)'. Below this is a message: 'Make sure to copy your personal access token now. You won't be able to see it again.' A 'Copied!' button with a checkmark is visible above a green box containing the copied token: 'ghp\_Kmdx72enC8FGSUoYQbc4W7gnMJBo3X39avRk'.

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.

A terminal window showing a git session:

```
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 input field, with 'Paste' highlighted.

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.

- 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](https://github.com/<github userid>/calc/tree/dev) in the browser.)

The screenshot shows a GitHub repository interface. At the top, there are navigation buttons: 'dev' (selected), '3 Branches' (which shows '3 Branches'), and '0 Tags'. On the right, there are buttons for 'Go to file', 'Add file', and 'Code'. Below these are two main sections: a 'Switch branches/tags' modal on the left and the repository history on the right. The modal has a search bar 'Find or create a branch...' and tabs for 'Branches' (selected) and 'Tags'. It lists branches: 'main' (default), 'cspace', and '✓ dev'. Below the modal is a link 'View all branches'. The repository history on the right shows a single commit: 'Updating title' by '61e42da · 8 hours ago' with '3 Commits'. There is also a 'Contribute' and 'Sync fork' button.

This screenshot shows the same GitHub repository after switching to the 'dev' branch. The top navigation and buttons are identical. The main repository history section now shows a message: 'This branch is 1 commit ahead of skillrepos/calc:main.' Below this, the commit history for 'Updating title' is shown again. The bottom section shows the repository tree with files: 'calc.html' (selected), 'README', and 'calc.html' (another entry). The blame history for 'calc.html' shows contributions from 'Brent Laster'.

- 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.

Brent Laster Updating title d9e79db · 22 minutes ago History

Code Blame 59 lines (46 loc) · 1.29 KB

```

1 <html>
2 <head>
3 <title>brentlaster's Calc</title>
4
5 <script language=javascript type="text/javascript">
6
7 var plus,minus,divide,multiply
8
9 function initialize(){
10 plus=document.calc.operator.options[0]
11 minus=document.calc.operator.options[1]
12 divide=document.calc.operator.options[2]

```

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

Brent Laster Updating title d9e79db · 24 minutes ago History

Code Blame 59 lines (46 loc) · 1.29 KB

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

The screenshot shows a GitHub commit page for a file named calc.html. The commit was made by Brent Laster 28 minutes ago. It has 1 parent (a1d22c4) and a commit hash d9e79db. The commit message is "Updating title". The diff shows two additions and two deletions. The changes are as follows:

```

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

```

-- 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.

### 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](#).

base repository: skillrepos/calc ▾ base: main ▾ ... head repository: brentlaster/calc ▾ compare: dev ▾

Choose a Base Repository

Filter repos

skillrepos/calc

brentlaster/calc

Automatically merged.

Reviewers: No reviews

Assignees: No one—assign yourself

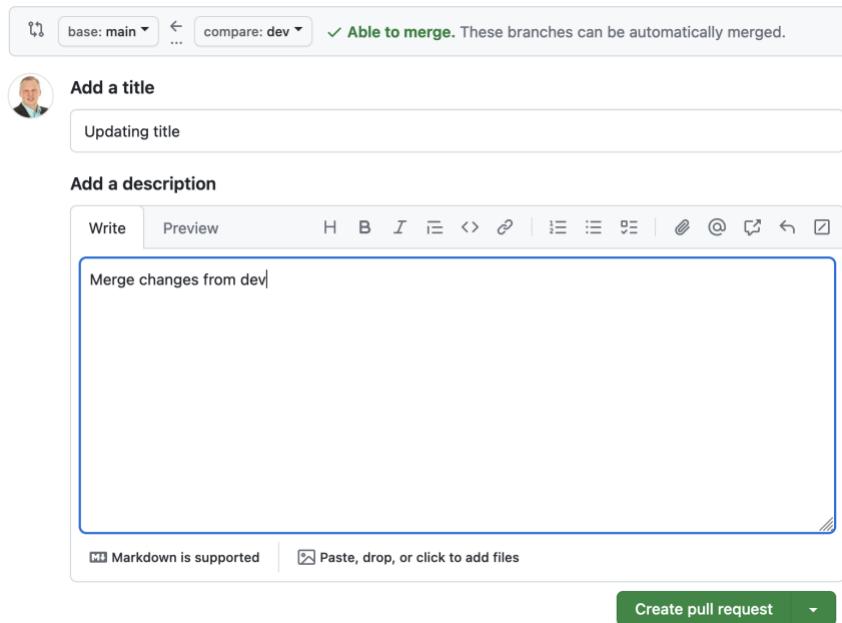
7. After making that change, the gray bar showing the base and compare should look like the screenshot below.

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 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*. You can look at the *Commits* or *File 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 second screenshot). Afterwards, you can click on the button to delete the ***dev*** branch if you want.

The screenshot shows a GitHub pull request interface. At the top, there's a green button labeled "Open" and a title "Updating title #1". Below the title, it says "brentlaster wants to merge 1 commit into `main` from `dev`". A commit history shows one commit titled "Updating title" with hash `d9e79db`. To the right, there are labels (None), projects (None), milestones (None), and notifications (None).

In the center, a modal window is open with the heading "Merge pull request #1 from brentlaster/dev". It contains the commit message "Updating title" and a note "This commit will be authored by bclaster@nclasters.org". At the bottom of the modal are two buttons: "Confirm merge" (green) and "Cancel".

Below the modal, there's a "Add a comment" button and a "Merged" button. The "Merged" button is purple and has a checkmark icon. The main commit history now shows a merged commit titled "Merge changes from dev" with a blue circle icon containing a smiley face. This commit has the same hash `d9e79db` and is associated with the author "brentlaster". A "Revert" button is located to the right of this commit.

At the bottom, a purple box displays the message "Pull request successfully merged and closed" with a "Delete branch" button. There's also a "Delete branch" button in the commit history area.

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**.

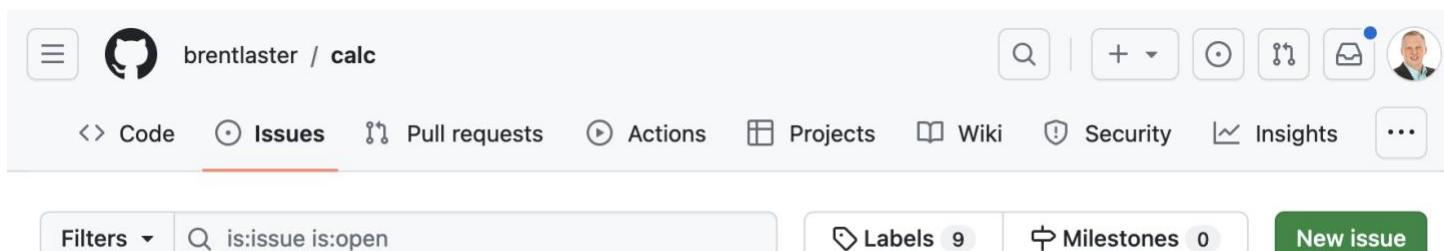


...

### Features

- Wikis  
Wikis host documentation for your repository.
- Restrict editing to collaborators only  
Public wikis will still be readable by everyone.
- Issues ✓  
Issues integrate lightweight task tracking into your repository. Keep projects on track with issue labels and milestones, and reference them in commit messages.
- Sponsorships

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.



The screenshot shows the GitHub issue creation interface. At the top, there are navigation links: Code, Issues (which is highlighted), Pull requests, Actions, Projects, Wiki, Security, Insights, and a three-dot menu. Below the title 'Add a title', there is a text input field containing 'Needs README'. To the right, under 'Assignees', it says 'No one—assign yourself'. In the 'Add a description' section, there is a rich text editor with a 'Write' tab selected. The description area contains the text 'Please add README file :book:'. Below the editor, it says 'Markdown is supported' and 'Paste, drop, or click to add files'. On the right side, there are sections for Labels ('None yet'), Projects ('None yet'), Milestone ('No milestone'), Development ('Shows branches and pull requests linked to this issue.'), and Helpful resources ('GitHub Community Guidelines'). At the bottom right is a green 'Submit new issue' button.

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

The screenshot shows the GitHub issue page for issue #2. The title is 'Needs README #2'. It has 1 comment. The first comment is from 'brentlaster' (Owner) and says 'Please add README file :book:'. Below the comments, there is a section for 'Add a comment' with a rich text editor and a placeholder 'Add your comment here...'. At the top, there are navigation links: Code, Issues (with a count of 1), Pull requests, Actions, Projects, Wiki, and a three-dot menu.

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

The screenshot shows a user interface with a top navigation bar containing 'Wiki', 'Security', 'Insights', and a three-dot menu. Below this is a toolbar with 'Edit' and 'New issue' buttons. On the left, there's a sidebar with a three-dot menu icon. The main area has a light blue header labeled 'Assignees' with a gear icon. Below it, the text 'No one—[assign yourself](#)' is displayed with a blue underline.

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

The screenshot shows a user interface with a sidebar on the left containing 'Assignees' and 'Labels' sections. The 'Labels' section is expanded, showing a box titled 'Apply labels to this issue' with a 'Filter labels' input field. Below it is a list of labels: 'documentation' (selected, indicated by a checked radio button), 'duplicate' (unchecked), and 'Something isn't working'. A note below the 'duplicate' label states 'This issue or pull request already exists'.

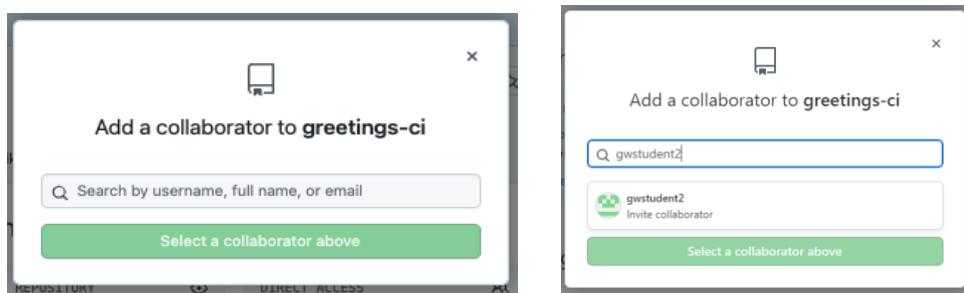
6. After this, if you click on the **Issues** tab at the top, and look at your issue, it should look like the following.

The screenshot shows a browser window for 'github.com/brentlaster/calc/issues'. The top navigation bar includes 'Code', 'Issues 1', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', and 'Insights'. The 'Issues' tab is active. Below the navigation is a search bar and filter controls. The main list shows one open issue: '#2 opened 15 hours ago by brentlaster' with the label 'Needs README documentation' highlighted in blue.

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.

The screenshot shows the GitHub repository settings page. The top navigation bar includes links for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Settings' tab is active. On the left, a sidebar menu under 'General' includes sections for Access (with 'Collaborators' selected), Code and automation (Branches, Tags, Actions, Webhooks, Environments, Pages), and Security (Code security and analysis, Deploy keys, Secrets). The main content area is titled 'Who has access'. It shows that the repository is a 'PUBLIC REPOSITORY' and that there are '0 collaborator have access to this repository. Only you can contribute to this repository.' A 'Manage' link is provided. Below this is the 'Manage access' section, which displays a message 'You haven't invited any collaborator' and a green 'Add people' button.

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.



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 github userid>/calc> you can also view the invitation via clicking on the button.

brentlaster / calc

Type  to search

Code Issues 1 Pull requests Actions Projects Security Insights

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

[View invitation](#)

@brentlaster has invited you to collaborate on this repository

Code Issues 1 Pull requests Actions Projects

[brentlaster invited you to collaborate](#)

[Accept invitation](#) [Decline](#)

 Owners of calc will be able to see:

- Your public profile information
- Certain activity within this repository
- Country of request origin
- Your access level for this repository
- Your IP address

Is this user sending spam or malicious content?  
[Block brentlaster](#)

brentlaster / calc

Code Issues 1 Pull requests Actions Projects ...

You now have push access to the brentlaster/calc repository.

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.

- 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.

The screenshot shows the GitHub interface for the 'calc' repository. The top navigation bar has tabs for Code, Issues (1), Pull requests, Actions, Projects, Wiki, and Settings. The repository card shows it's a public fork from 'skillrepos/calc'. Below the card, there's a message about being 2 commits ahead of 'skillrepos/calc:main'. A list of recent commits by 'brentlaster' is shown, including 'Merge pull request #1 from br...' (dc98b08 · yesterday), 'Initial add of files' (last week), and 'Updating title' (yesterday). A large callout box highlights the 'README' section, which contains a 'Add a README' button. This button is circled in red.

- 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.)

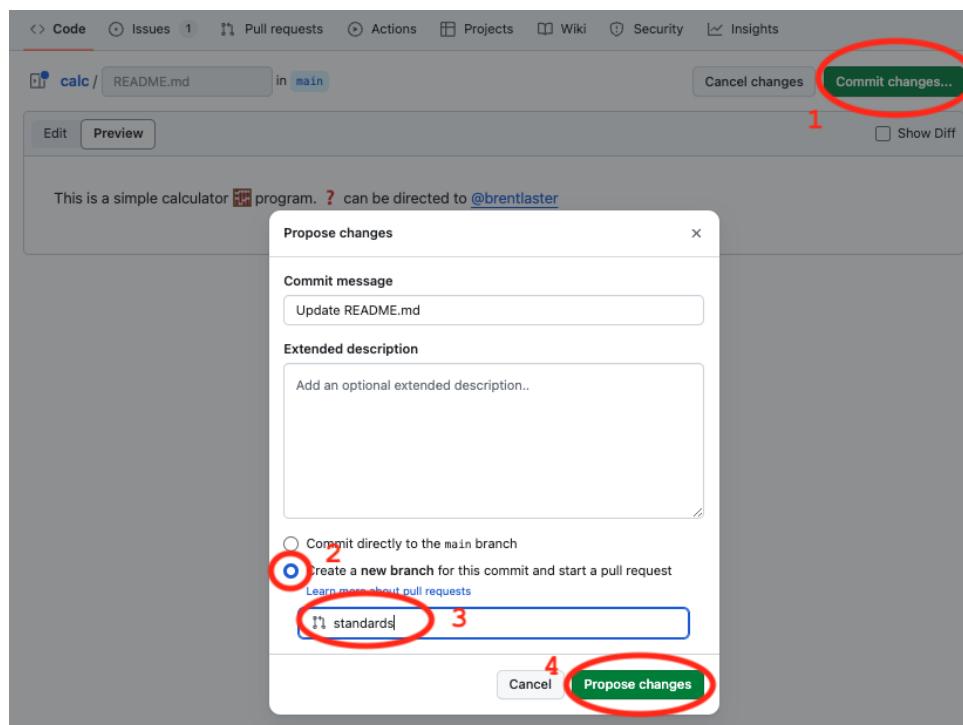
This is a simple calculator :abacus: program. :question: can be directed to [@github-userid](<https://github.com/github-userid>)

A screenshot of a GitHub repository named 'calc'. The file 'README.md' is being edited in the 'main' branch. The commit message contains two lines of text: '1 This is a simple calculator :abacus: program. :question: can be directed to [@brentlaster](https://github.com/brentlaster)(<https://github.com/brentlaster>)' and '2'. At the top right, there are 'Cancel changes' and 'Commit changes...' buttons. Below the buttons are 'Edit' and 'Preview' tabs, and settings for 'Spaces', '2', and 'Soft wrap'.

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

A screenshot of the same GitHub repository and file ('README.md' in 'main'). The 'Preview' tab is selected, showing the rendered content: 'This is a simple calculator :abacus: program. ? can be directed to [@brentlaster](#)'. There is also a 'Show Diff' checkbox at the top right.

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**.



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 shows a commit history with one commit from "brentlaster" on Jan 1, 2024, titled "Create README.md". The diff viewer shows a single addition to README.md: "This is a simple calculator :abacus: program. :question: can be directed to [@brentlaster] (<https://github.com/brentlaster>)". At the bottom right of the comparison view, the "Create pull request" button 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.

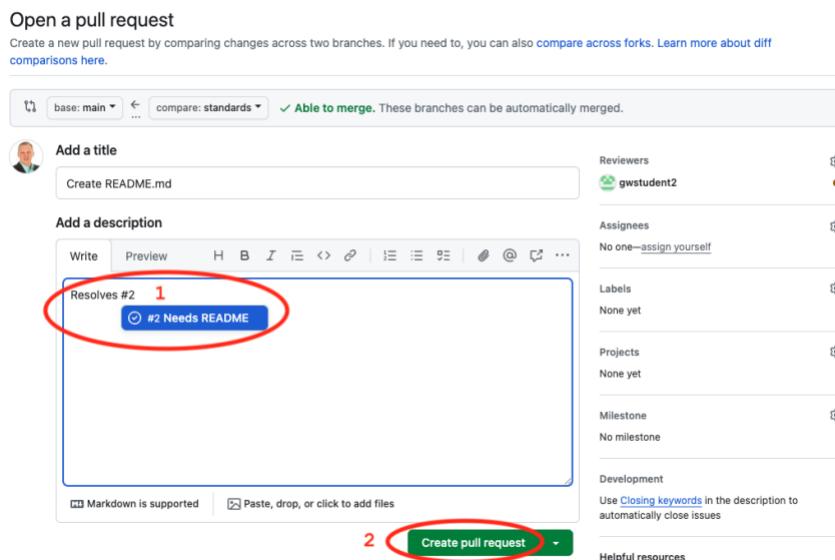
The screenshot shows the GitHub interface for opening a pull request. The top bar includes "base: main" and "compare: standards" dropdowns, and a green "Able to merge" banner. On the left, there are fields for "Add a title" (with "Create README.md") and "Add a description" (with a rich text editor placeholder "Add your description here..."). On the right, there are sections for "Reviewers" (set to 1), "Request up to 15 reviewers", "Type or choose a user" (with "gwstudent2" selected and highlighted with a red circle), "Labels" (2), "Projects" (None yet), "Milestone" (No milestone), and "Development" (with a note about closing keywords). At the bottom, there is a "Create pull request" button and a "Helpful resources" link.

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

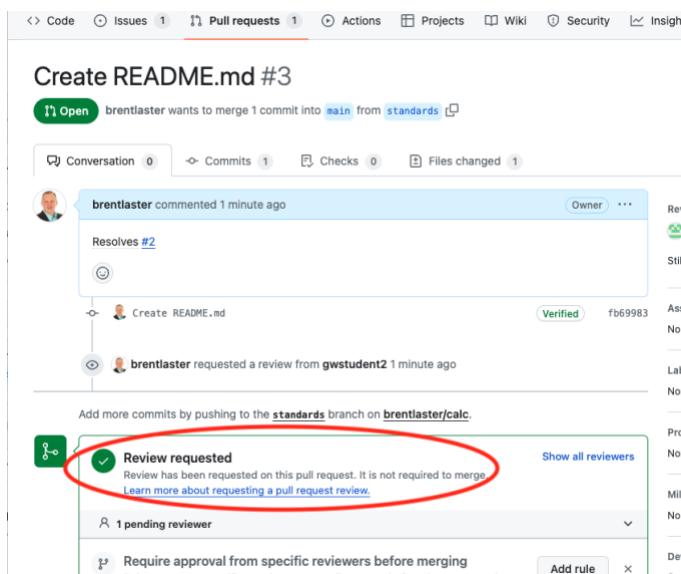
### Resolves #2

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

Then click on the “Create pull request” button.



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 user id.



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.

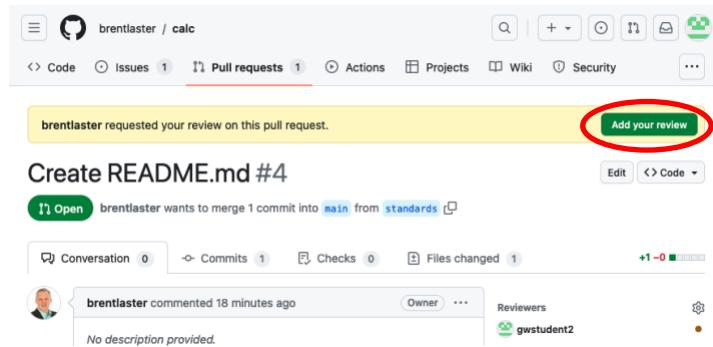
1. In a separate browser or a private tab, log in to your secondary GitHub userid (the one you added as a collaborator and a reviewer). After you log in, you can either go to your notifications to see the item about the requested review or go to <https://github.com/pulls/review-requested>. Then click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

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.

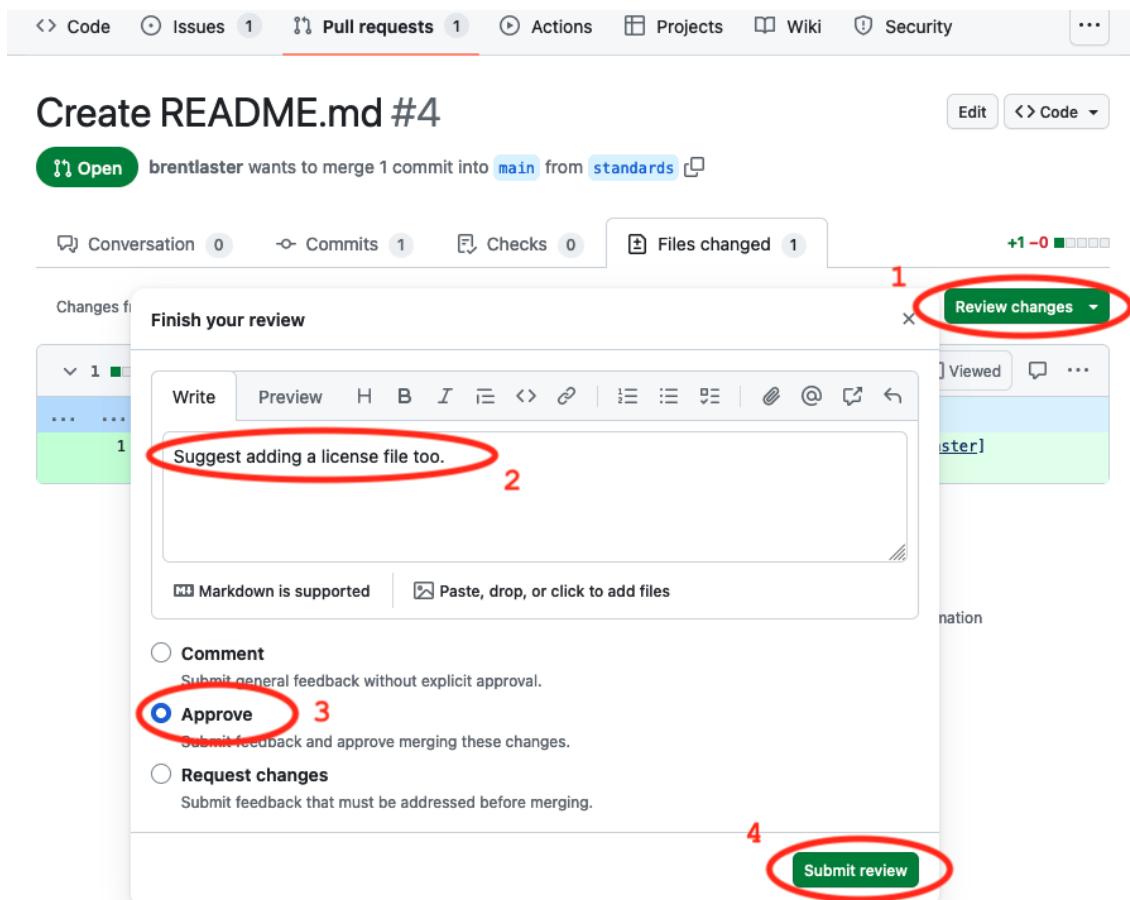
- OR -

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 a search bar with the query 'is:open is:pr review-requested:gwstudent2 archive'. Below the navigation, it says '1 Open' and '1 Closed'. A list shows a single pull request: '#4 brentlaster/calc Create README.md #4 opened 14 minutes ago by brentlaster'. This item is also circled in red. 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”.



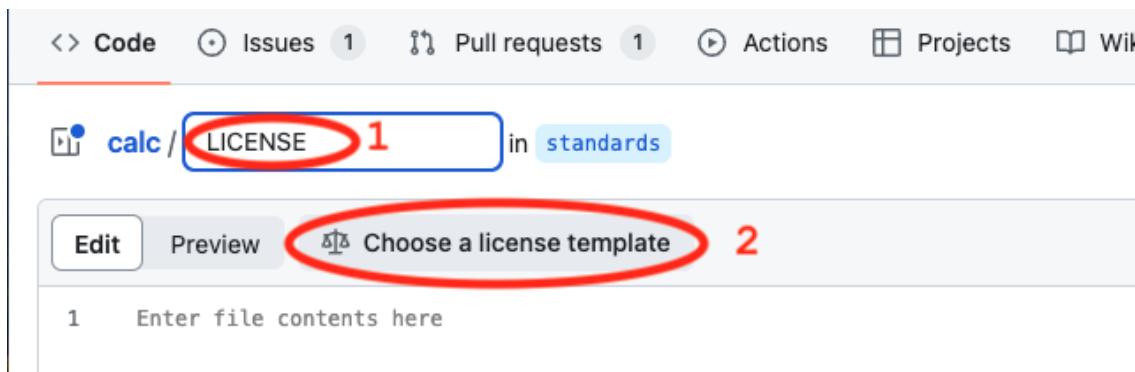
4. Go to the session with your original GitHub userid or log out of the other one and log back in if you need to. Go to the **Pull requests** menu at the top, find the pull request and click on the commit message. Then you should see a screen like below.

The screenshot shows a GitHub pull request interface. At the top, the URL is [github.com/brentlaster/calc/pull/4](https://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 SHA-1 hash '47b11e1'. A review from 'gwstudent2' is listed as approved. A comment from 'gwstudent2' suggests adding a license file.

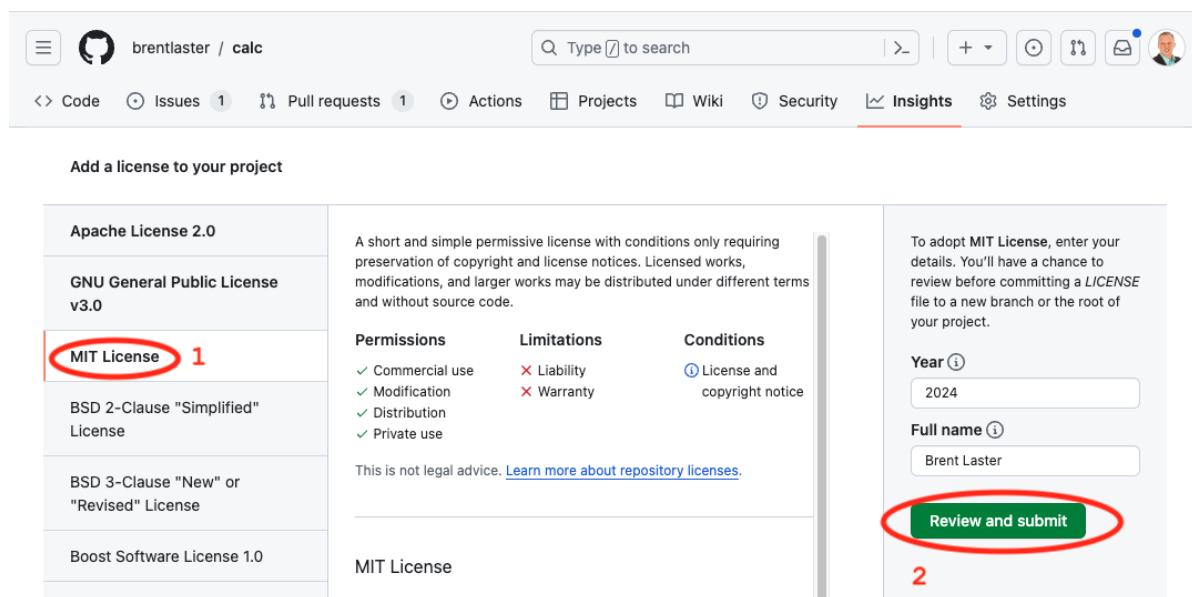
5. Since there was a suggestion to add a license file, that sounds like a good idea, so let's do that. Click on the **Code** tab at the top, then select the **standards** branch (or whatever name you gave the new branch) from the branch dropdown, then select the "+" sign (or "Add file" option) and the 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 history lists several commits by 'brentlaster' and others, including the creation of 'README.md' and 'calc.html'. The repository page also shows a 'README' file with a note about it being a simple calculator program.

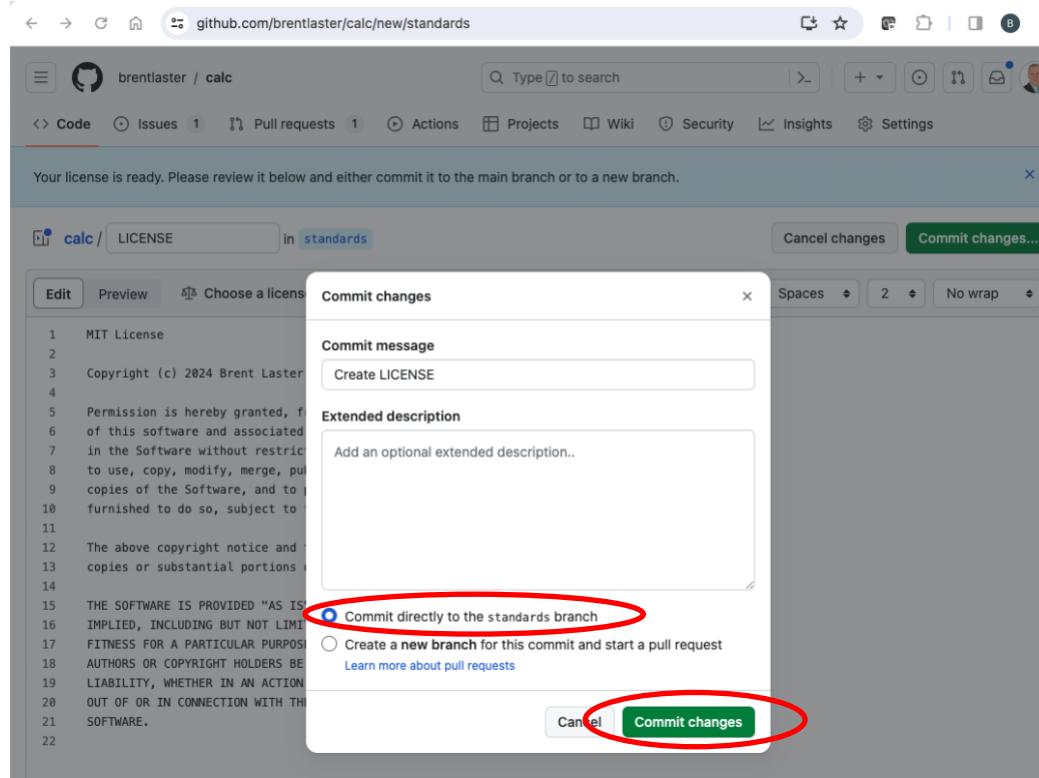
6. In the next screen, there will be a text entry area for the name of the file. Type in “LICENSE” for the name. Then, an option will display that says ***Choose a license template***. Click on that option. You will be asked about discarding changes. It’s ok in this case, so click on “OK”.



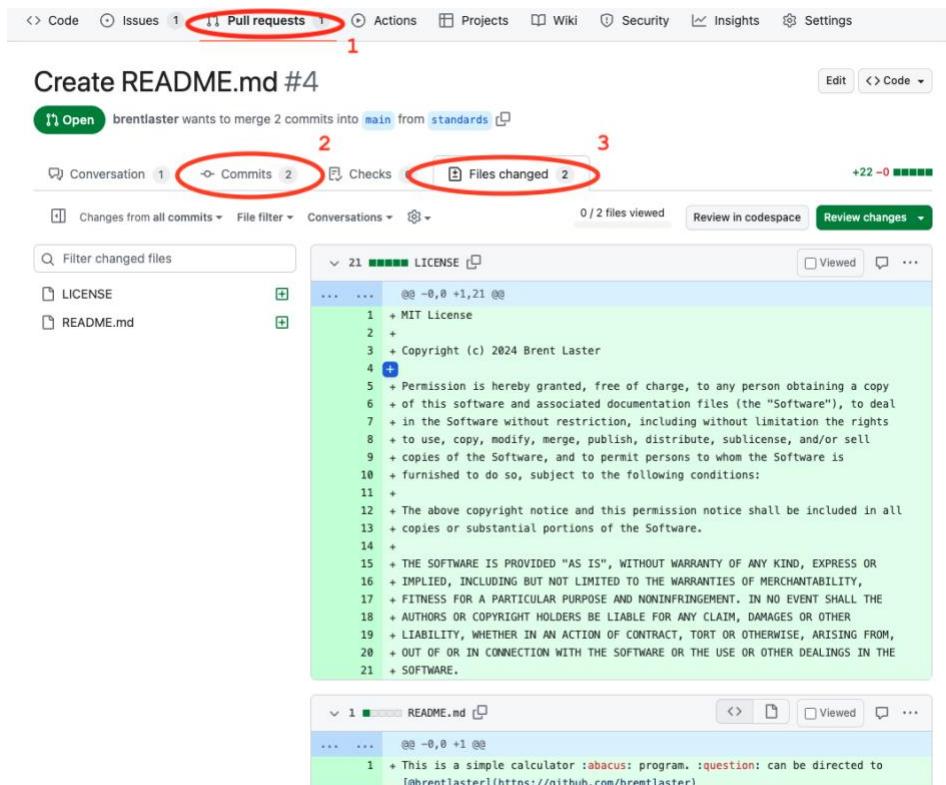
7. On the next screen, you’ll be able to pick the license you want. You can select the “MIT License” or another one if you prefer. Once done, click the “Review and submit” button on the right.



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.



- 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. 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.

The screenshot shows a GitHub issue page for a repository named 'brentlaster / calc'. The issue is titled 'Needs README #2' and is currently closed. The timeline on the left shows the following activity:

- brentlaster commented 19 hours ago: Please add README file
- brentlaster self-assigned this 19 hours ago
- brentlaster added the documentation label 19 hours ago
- brentlaster mentioned this issue 24 minutes ago: Create README.md #4 (status: Merged)
- brentlaster linked a pull request 22 minutes ago that will close this issue: Create README.md #4 (status: Merged)
- brentlaster mentioned this issue 15 minutes ago: Update README.md #5 (status: Merged)
- brentlaster closed this as completed in #5 14 minutes ago

The right side of the page displays issue details such as assignees (brentlaster), labels (documentation), projects (None yet), milestones (No milestone), and development history (Successfully merging a pull request may close this issue). It also shows notification settings and a link to unsubscribe.

END OF LAB

## Lab 6: Adding a GitHub Pages website for your repository

**Purpose:** In this lab, we'll setup a GitHub Pages repo for your repository.

- This lab is done with your primary GitHub id. In order to prepare for publishing a page, let's create a new branch in our repo. In the **Code** tab, if not on the **main** branch, click on the branch dropdown, type in **main** and select it. Click in the branch dropdown again, and, in the text area that says, **Find or create a branch...**, enter the text **pages**. Then click on the “*Create branch: pages from main*” link.

The screenshot shows the GitHub 'Code' tab with two branches visible: 'main' and 'pages'. The 'main' branch is selected. In the dropdown menu for 'main', the text 'Create branch pages from main' is highlighted with a red circle. The 'pages' branch is also visible in the dropdown menu.

2. Now create a new repository to use for the *pages* repo. Go to

<https://github.com/new>

to create a new repository. (Alternatively, you could go to your home page, then to ***Repositories***, then to ***New.***)

Name the new repository precisely <**github-userid**>.**github.io** replacing your actual GitHub userid for the <*github-userid*> item. (This will look like a repeat of your userid since the project has your userid in the name in your github userid space.) You can optionally add a description if you want.

github.com/new

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

**Repository template**

No template ▾

Start your repository with a template repository's contents.

---

**Owner \*** brenlaster  / brenlaster.github.io

 brenlaster.github.io is available.

Great repository names are short and memorable. Need inspiration? How about [redesigned-parakeet](#) ?

**Description (optional)**

Code repo for the web page for the calc code

---

 **Public**  
Anyone on the internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

When done, click on the **Create repository** button at the bottom of the page.

**Create**

3. So that we have content to publish, we'll grab the code from the calc.html file in your local repository from Lab 1 and add it here. On the screen with the ***Quick setup – if you've done this kind of thing before*** instructions, click on the link in the big blue bar for uploading an existing file.

The screenshot shows the GitHub repository setup interface for `brentlaster/brentlaster.github.io`. In the center, there's a section titled "Quick setup – if you've done this kind of thing before". It contains links for "Set up in Desktop" (disabled), "HTTPS" (selected), "SSH", and a URL. Below these links is the text: "Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)". A red oval highlights the "uploading an existing file" link.

4. On the “upload” screen, drag the file `calc.html` from your local directory (where you cloned it in Lab 1) to the indicated area -or- click on the ***choose your files*** link and browse out and select the file (and click *Open*). Then click on the ***Commit changes*** button to add the file to the repo.

The screenshot shows the GitHub commit dialog for the repository `brentlaster/brentlaster.github.io`. At the top, there's a large input field with a red oval around it, labeled "Drag additional files here to add them to your repository" and "Or choose your files". Below this, a file named "calc.html" is listed with a red oval around it and the number "2" next to it. At the bottom, there's a "Commit changes" button with a red oval around it and the number "3" next to it. The "Commit changes" dialog also includes fields for "Add files via upload" and "Add an optional extended description...".

5. You should now be back in the new repo's **Code** tab. Let's change the name and location of this file to make it more consistent for GitHub Pages. Click on the *calc.html* file and then click on the "pencil" icon to edit it.

brentlaster / brentlaster.github.io

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

**calc.html**

brentlaster.github.io / calc.html

brentlaster Add files via upload ✓ f134ea7 · 5 minutes ago History

63 lines (50 loc) · 1.4 KB

Code Blame Edit this file Raw

```

1  <html>
2  <head>
3      <title>Calc</title>

```

5. In the edit dialog, in the text entry box for the name, type over the "calc.html" text with the replacement text of "docs/index.html". Note you are adding the directory *docs* to the path. The screenshots show this in 2 parts for clarity.

brentlaster / brentlaster.github.io

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

**brentlaster.github.io / docs**

Cancel changes Commit changes...

Edit Preview Tabs 8 No wrap

brentlaster / brentlaster.github.io

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

**brentlaster.github.io / docs / index.html**

Cancel changes Commit changes...

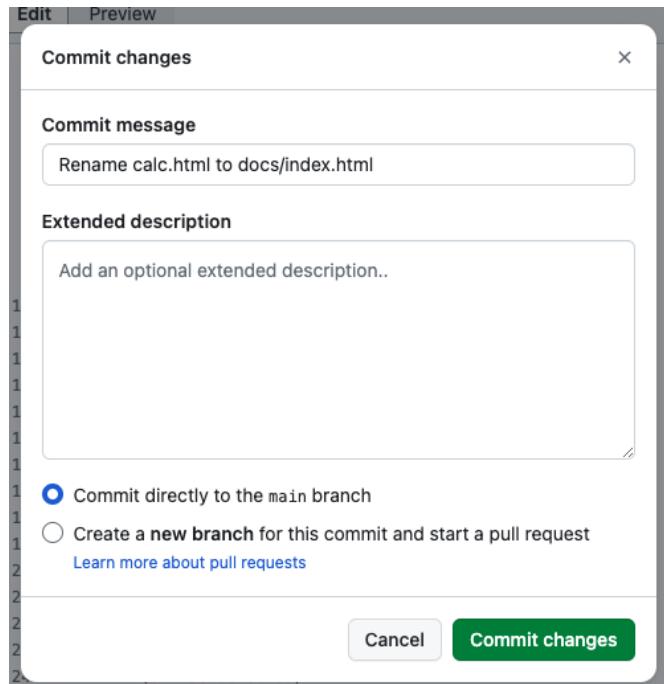
Edit Preview Tabs 8 No wrap

```

1  <html>
2  <head>
3      <title>Calc</title>

```

6. Commit your changes for the rename directly to the *main* branch by clicking on the **Commit changes...** button.



- Now we need to set the source from the repo for the web page. Go to the repo's **Settings** tab. On the left side, select the **Pages** entry. Under the **Build and deployment/Branch** section, under the folder dropdown, select the **/docs** entry and then click the **Save** button.

GitHub Pages

[Your site is live at `https://brentlaster.github.io/`](#)  
Last deployed by  `brentlaster` 15 minutes ago

**Build and deployment**

**Source**  
`Deploy from a branch`

**Branch**  
Your GitHub Pages site is currently being built from the `main` branch. [Learn more about config source for your site.](#)

**Select folder**  
`/docs`

**Custom domain**

- After these changes, you can visit the site at <https://<github-userid>.github.io> and using the button in the page or just go to the URL to see the automatic web page.

## GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.



A screenshot of a GitHub Pages site titled "Calc". The page contains instructions: "Enter a number in the first box and a number in the second box and select the answer button to see the answer. Change the operation via the dropdown selection box if desired." Below the instructions is a form with input fields for numbers, an operator dropdown, and a result field, followed by a "Get answer" button.

### OPTIONAL:

- Change the displayed metadata about the github.io repo to show more details about the project. On the repo's **Code** page, on the right side, click on the gear icon next to **About**.

A screenshot of a GitHub repository page for "Calc". The "About" section is highlighted with a red circle around the gear icon. The text in the About section reads: "Code repo for the web page for the calc code". Other repository details like Unwatch (1), Fork (0), Star (0), and 4 Commits are also visible.

- Add repository details such as the ones below. For the Website, you can just click the checkbox. For Topics, just start typing in the field. Once you are done, click the **Save changes** button and you should see your edits show up on the repo's page.

The image shows two side-by-side screenshots of a GitHub interface. On the left, a modal window titled 'Edit repository details' is open, showing fields for 'Description' (Simple web calculator program), 'Website' (https://brentlaster.github.io/), and 'Topics' (calculator, example-code). Below these are checkboxes for 'Include in the home page' (Releases, Packages, Deployments) and a 'Save changes' button. On the right, the user's GitHub profile page is shown with the name 'brentlaster.github.io/' and activity metrics: 0 stars, 1 watching, and 0 forks.

END OF LAB

## Lab 7: 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 image shows a screenshot of a GitHub repository named 'greetings-ci'. The repository is public and has 2 branches and 0 tags. It was last updated 3 days ago by 'brentlaster' with a commit message 'Delete extra directory'. The repository description is 'Simple starter repo for CI/CD with GitHub Actions'. A 'Fork your own copy of skillrepos/greetings-ci' button is visible in the top right corner of the repository card.

The screenshot shows the GitHub fork creation interface for the repository `skillrepos/greetings-ci`. The URL in the browser bar is `github.com/skillrepos/greetings-ci/fork`. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and a user profile icon.

**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. [View existing forks](#).

Required fields are marked with an asterisk (\*).

**Owner \***: brentlaster / **Repository name \***: greetings-ci  greetings-ci 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 starter repo for CI/CD with GitHub Actions

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

ⓘ You are creating a fork in your personal account.

**Create fork**

A yellow callout box labeled "uncheck" points to the  checkbox for "Copy the main branch only". A red circle highlights the "Create fork" button.

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** tab, click on the **Actions** button in the top menu under the repository name.

The screenshot shows the GitHub repository page for `gwstudent/greetings-ci`. The repository is public and was forked from `skillrepos/greetings-ci`. The top navigation bar includes links for Pull requests, Issues, Marketplace, Explore, Pin, Watch, Fork, Star, and Settings.

The **Actions** tab is highlighted with a red circle.

**Code** dropdown: main, 1 branch, 0 tags

This branch is up to date with `skillrepos/greetings-ci:main`. [Contribute](#) [Fetch upstream](#)

Commit	Author	Date	Commits
extra	Brent Laster	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

**About**: Simple starter repo for CI/CD with GitHub Actions  
 0 stars  
 0 watching  
 1 fork

**Releases**: No releases published [Create a new release](#)

**Packages**: No packages published

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**.

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.

### 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](#) →

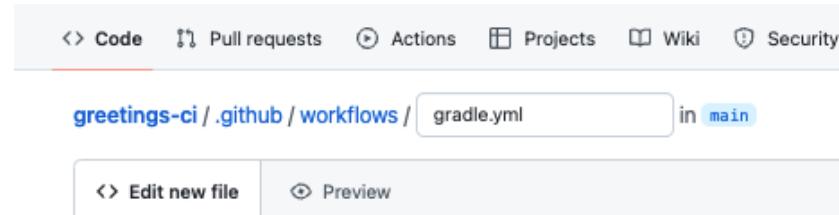
6. From the results, select the **Java with Gradle** one and click the **Configure** button to open a predefined workflow for this.

### Get started with GitHub Actions

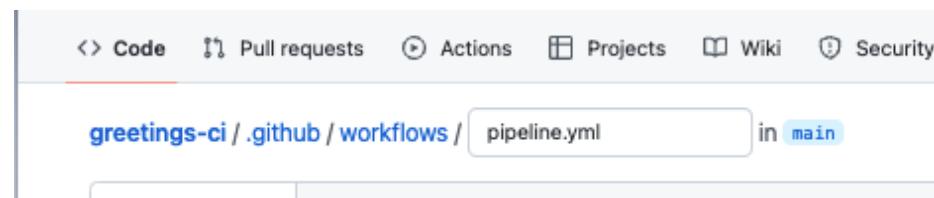
Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

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

7. 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.)



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.

[greetings-ci/.github/workflows/pipeline.yml](#) in main

Edit Preview

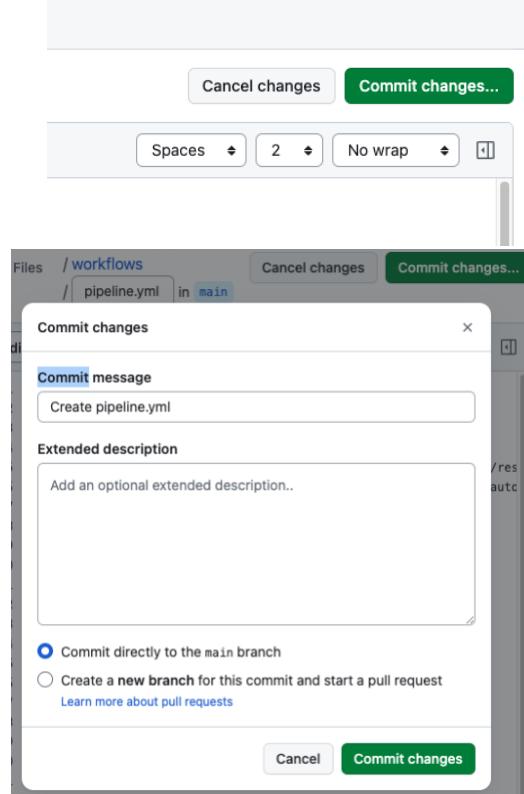
```

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

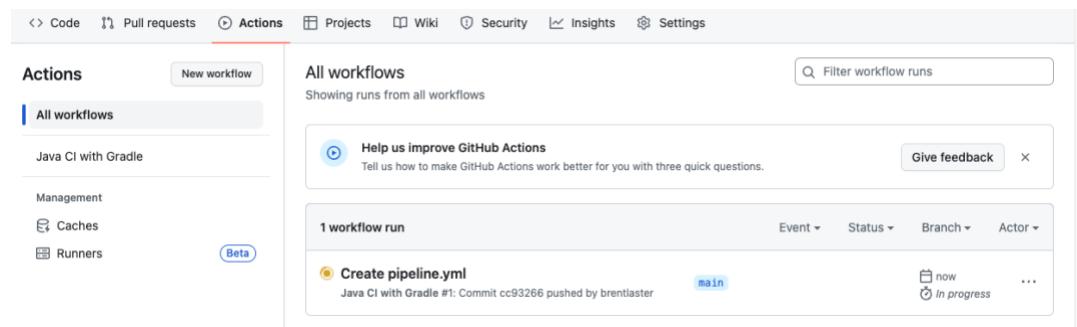
```

A red rectangular box highlights the code starting from line 50, specifically the 'dependency-submission:' job and its steps. The text 'highlight/select, and delete' is written in red at the bottom right of this highlighted area.

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.)



10. 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 interface. On the left, there's a sidebar with 'Actions' selected, followed by 'New workflow', 'All workflows', 'Java CI with Gradle' (which is highlighted), 'Management', 'Caches', 'Runners' (with a 'Beta' badge), and other options like 'Event', 'Status', 'Branch', and 'Actor'. The main area is titled 'Java CI with Gradle' and shows 'pipeline.yml'. It includes a feedback section, a '1 workflow run' summary, and a detailed view of the most recent run. The run is labeled 'Create pipeline.yml' and was triggered by a commit from 'Java CI with Gradle #1: Commit cc93266 pushed by brentlaster'. The run status is 'main' and it succeeded 1 minute ago, taking 36s. A red circle highlights the 'Create pipeline.yml' link in the run list.

- 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.

This screenshot shows the detailed view of a specific GitHub Actions job. At the top, there are navigation links: Code, Pull requests, Actions (which is underlined), Projects, Wiki, Security, Insights, and Settings. Below that, it says '← Java CI with Gradle' and 'Create pipeline.yml #1'. There are buttons for 'Re-run all jobs' and three dots. The main area is titled 'Summary' and shows a 'Jobs' section. Under 'Jobs', there's a list with 'build' highlighted and circled in red. To the right, there's a detailed view of the 'build' job, which succeeded 3 minutes ago in 25s. It has a search bar for logs and settings icons. The job graph shows a single step: 'Set up job' (3s) and 'Run actions/checkout@v3' (1s). The 'Run actions/checkout@v3' step is expanded, showing its sub-steps: 1. Run actions/checkout@v3, 14. Syncing repository: brentlaster/greetings-ci, 15. Getting Git version info, and 19. Temporarily overriding HOME='/home/runner/work/\_temp/77610fae-ac20-4aea-8803530bce6e' before making global git config changes. A red circle highlights the 'Run actions/checkout@v3' step in the job graph.

END OF LAB

## Lab 8: 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.)

```

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).

base repository: skillrepos/greetings-ci | base: main | head repository: gwstudent/greetings-ci | compare: main

Choose a Base Repository

Filter repos

Discussions

skillrepos/greetings-ci

gwstudent/greetings-ci

Create pull request

## 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](#).

base: main | compare: package

Choose a head ref

Find a branch

Branches Tags

main default

package

There isn't anything to compare.

Commits from package. Try [switching the base](#) for your comparison.

Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

1 commit 2 files changed 1 contributor

Commits on Jan 17, 2024

Initial add on package

Brent Laster committed on Jan 17

Showing 2 changed files with 52 additions and 0 deletions.

.github/workflows/publish-package.yml

```
@@ -0,0 +1,25 @@
+ name: Publish package to GitHub Packages
+ on:
+   release:
+     types: [created]
```

Split Unified

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

Code Pull requests 1 Actions Projects Wiki Security Insights Settings

Initial add on package #1

Open gwstudent wants to merge 1 commit into main from package

Conversation 0 Commits 1 Checks 1 Files changed 2

+52 -0

Changes from all commits File filter Conversations

Filter changed files .github/workflows/publish-package.yml

Viewed ...

...

@@ -0,0 +1,25 @@

+ name: Publish package to GitHub Packages

+ on:

+ release:

+ types: [created]

Review in codespace Review changes

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 interface for a repository. The top navigation bar includes links for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main area is titled "Files" and shows a tree view of the repository structure. A file named "greetings-ci/.github/workflows/publish-package.yml" is selected. The code editor displays the following YAML configuration:

```
name: Publish package to GitHub Packages
on:
  release:
    types: [created]
  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.

The screenshot illustrates the GitHub Actions interface for creating and running workflows.

**Top Navigation:** The navigation bar includes links for Code, Pull requests, Actions (circled in red), Projects, Wiki, Security, Insights, and Settings.

**Left Sidebar:** The sidebar lists various actions and management options. The "Publish package to GitHub Packages" action is highlighted with a red circle and circled in red again (labeled 1). Other items include Java CI with Gradle, Management, Caches, and Runners.

**Workflow Details:** The main area shows the "Publish package to GitHub Packages" workflow. It features a "Help us improve GitHub Actions" card, a summary of 0 workflow runs, and a note about the workflow trigger. A "Run workflow" button is present in the top right (circled in red, labeled 2).

**Workflow Execution:** Below the summary, a modal window titled "Use workflow from" shows the "Branch: main" dropdown and a large green "Run workflow" button (circled in red, labeled 3). This button is also highlighted with a red circle (labeled 4) in the bottom right corner of the modal.

**Execution History:** At the bottom, a history section shows a successful run of the workflow, triggered by brentlaster, with a timestamp of 1 minute ago and a duration of 34s.

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 a GitHub repository page for 'greetings-ci'. The 'Code' tab is selected. In the bottom right corner of the main content area, there is a 'Packages' section with a count of 1. A red circle highlights the link for the package 'org.gradle.sample.greetings-ci'.

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 a GitHub profile page for 'brentlaster'. In the top right, there is a user picture with a red circle around it. A dropdown menu appears, with 'Your profile' highlighted by a red circle. On the left, the 'Packages' tab is selected in the navigation bar, indicated by a red circle with the number 2 above it. On the right, the 'Your profile' dropdown menu is open, showing options like 'Set status', 'Your repositories', etc., with 'Your profile' highlighted by a red circle with the number 1 above it. Below the dropdown, a list of packages is shown, with one package circled in red and labeled with the number 3.

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.

**org.gradle.sample.greetings-ci 1.1**

<> Install 1/2: Add this to pom.xml: [Learn more about Maven or Gradle](#)

```
<dependency>
  <groupId>org.gradle.sample</groupId>
  <artifactId>greetings-ci</artifactId>
  <version>1.1</version>
</dependency>
```

Install 2/2: Run via command line

```
$ mvn install
```

Details

brentlaster · January 05, 2024

Assets

- greetings-ci-1.1.pom.md5
- greetings-ci-1.1.pom.sha1
- greetings-ci-1.1.pom
- greetings-ci-1.1.jar.md5
- greetings-ci-1.1.jar.sha1
- greetings-ci-1.1.jar**

END OF LAB

## Lab 9: Creating a release

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

- On the **Code** tab 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>.)

**About**

Simple starter repo for CI/CD with GitHub Actions

Activity

0 stars

0 watching

140 forks

Releases

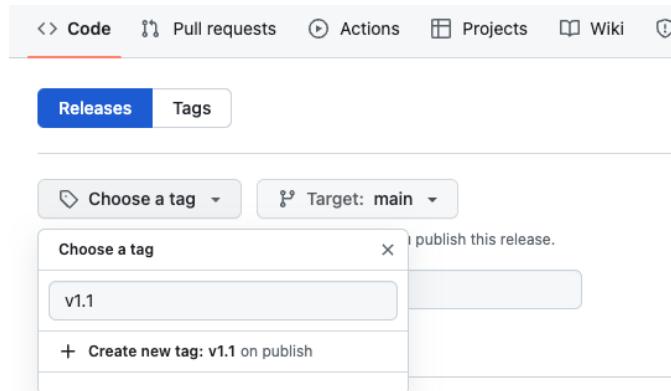
No releases published

[Create a new release](#)

Packages 1

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.



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. The 'Write' tab is active. At the top, there are dropdowns for 'Choose a tag' set to 'v1.1' and 'Target' set to 'main'. A message says 'Excellent! This tag will be created from the target when you publish this release.' Below is a 'Release title' input field. Under the title, there are 'Write' and 'Preview' tabs. A rich text editor toolbar is above a 'Describe this release' text area. At the bottom, there's an input field for attaching files with the placeholder 'Attach files by dragging & dropping, selecting or pasting them.' A red arrow points from this field to a local file named 'greetings-ci-1.1.1.jar' in the 'Downloads' folder. To the right of the attachments section, there's a 'Generate release notes' button.

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

The screenshot shows the bottom of the GitHub Releases page. It includes a 'Set as a pre-release' checkbox and two buttons: 'Publish release' (highlighted with a red arrow) and 'Save draft'.

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

The screenshot shows the GitHub Releases page for a repository named 'Initial release'. At the top, there are navigation links: Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and a three-dot menu. Below these, it says 'Releases / Initial release'. The main content area displays the release titled 'Initial release' (marked as 'Latest'). It shows a profile picture of the user 'brentlaster' and the message 'released this now'. A note below states 'Initial release of content for 1.1.' Under the 'Assets' section, there are three items listed: 'greetings-ci-1.1.jar' (1006 Bytes, 2 minutes ago), 'Source code (zip)' (21 minutes ago), and 'Source code (tar.gz)' (21 minutes ago). There are also 'Compare', 'Edit', and 'Delete' buttons at the top right of the release card.

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 GitHub repository page for 'greetings-ci'. At the top, there are navigation links: Code, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings, and a three-dot menu. Below these, it shows the repository details: 'greetings-ci' (Public), forked from 'skillrepos/greetings-ci'. On the right side, there are social sharing buttons for Pin, Watch (0), Fork (140), and Star (0). The main content area shows the 'main' branch status: '2 commits ahead of, 3 commits behind skillrepos/greetings-ci:main'. Below this, a list of recent commits is shown, starting with a merge pull request from 'brentlaster'. To the right, there is an 'About' section with a description: 'Simple starter repo for CI/CD with GitHub Actions', followed by metrics: Activity (0), Stars (0), Watching (0), and Forks (140). The 'Releases' section is highlighted with a red oval, showing one release: 'Initial release' (Latest, 1 minute ago). The 'Packages' section shows one package: 'org.gradle.sample.greetings-ci'. At the bottom, there is a 'Packages' section with one item: 'org.gradle.sample.greetings-ci'.

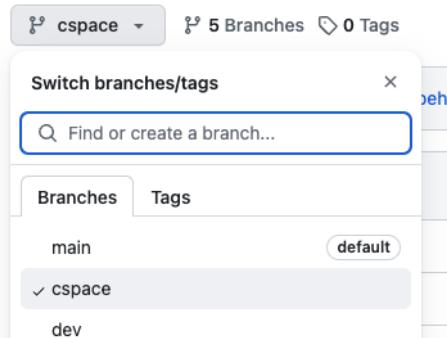
7. You can click on that if you want to see details about the release (same as output in step 5).

END OF LAB

## Lab 10: Working with Codespaces

**Purpose:** In this lab, you'll see how to work with a GitHub Codespace

1. Go back to the `github.com/<github-userid>/calc` project. In that project, select the `cspac` branch.



2. Click on the `<> Code` button, then select the **Codespaces** tab, and then select **Create codespace on cspace**.

This screenshot shows the same GitHub repository page as the previous one, but with additional annotations. A red circle labeled '1' highlights the `<> Code` button at the top right of the main content area. Another red circle labeled '2' highlights the 'Codespaces' tab in the navigation bar. A third red circle labeled '3' highlights the green button labeled 'Create codespace on cspace' in the 'Codespaces' section. The rest of the interface is identical to the first screenshot.

3. Creating the codespace will take a few minutes to complete. When it's done, you'll now have a new codespace with this repo checked out and the calculator webpage open and running. There is also a terminal at the bottom. This is a secondary terminal. To get back to the main terminal, click on the `bash` selection at the far right side. (You can also dismiss any dialogs about linting.)

The screenshot shows the VS Code interface within a browser window. On the left is the Explorer sidebar with a tree view of files: CALC [CODESPACES: FRIENDLY SPACE ROBOT] containing \_\_pycache\_\_, .devcontainer, templates (with calc.html selected), .gitignore, app.py, LICENSE, README.md, and requirements.txt. The center-left pane displays the code for calc.html:

```

1 <html>
2 <head>
3   <title>brentlaster's Calc</title>
4
5 <script language=javascript type="text/javascript">
6
7 var plus,minus,divide,multiply
8
9 function initialize(){
10   plus=document.calc.operator.options[0]
11   minus=document.calc.operator.options[1]
12   divide=document.calc.operator.options[2]
13   multiply=document.calc.operator.options[3]
14 }
15

```

The center-right pane shows a browser window titled "Simple Browser" displaying the web application "Calc". The page contains instructions: "Enter a number in the first box and a number in the second box and select the answer button to see the answer. Change the operation via the dropdown selection box if desired." Below this are input fields for numbers and an "answer" button. The bottom right of the browser window has a red circle around the "bash" link in the dropdown menu.

The bottom right of the main interface shows the terminal output:

```

flask --debug run
@brentlaster ~ /workspaces/calc (cspac) $ flask --debug run
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 275-577-525
127.0.0.1 - - [02/Jan/2024 11:54:00] "GET /?vscodeBrowserReqId=1704196436028 HTTP/1.1" 200 -

```

4. To see how to edit in a codespace, let's change the title displayed in the webapp. The file *calc.html* was already opened automatically for you. Click in the *calc.html* pane and scroll down to line 34 where the title is. Just type into that line and add your name in front of "Calc". The change is automatically saved.

The screenshot shows the VS Code interface within a browser window. On the left is the Explorer sidebar with a tree view of files: PACE ROBOT] containing M. The center-left pane displays the code for calc.html:

```

26   document.calc.answer.value = a * b
27 }
28
29 </script>
30
31 </head>
32
33 <body onLoad="initialize()">
34 <h2>Brent's Calc</h2>
35
36 Enter a number in the first box and a number
37 <br>
38 Change the operation via the dropdown selecti
39 <form name="calc" action="post">
40 <input type="text" name="val1" size=10>

```

The center-right pane shows a browser window titled "Simple Br" displaying the web application "Calc". The page contains instructions: "Enter a number in the first box and a number in the second box and select the answer button to see the answer. Change the operation via the dropdown selection box if desired." Below this are input fields for numbers and an "answer" button.

5. Now, in the Simple Browser pane, click the circular arrow icon to reload the webapp. You should see your change being displayed.

The screenshot shows a DevTools window with two main panes. On the left is a code editor for a file named 'calc.html' with the following content:

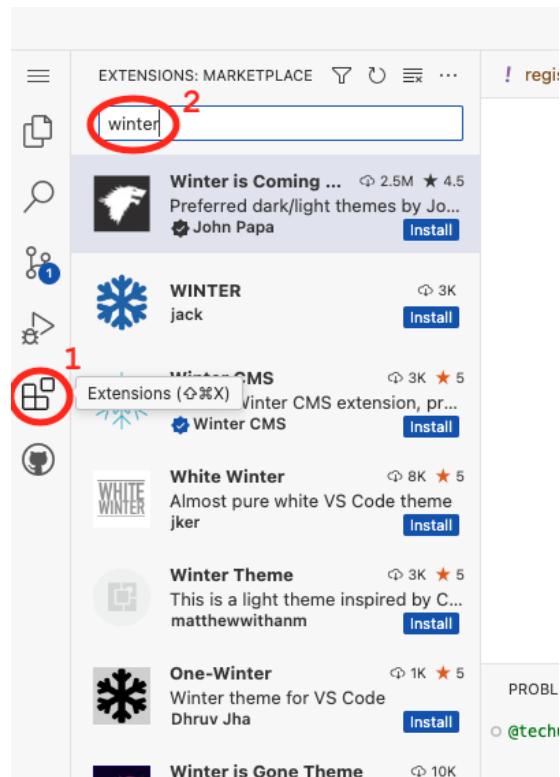
```

26     document.calc.answer.value = a * b
27 }
28
29 </script>
30
31 </head>
32
33 <body onLoad="initialize()">
34   <h2>Brent's Calc</h2>

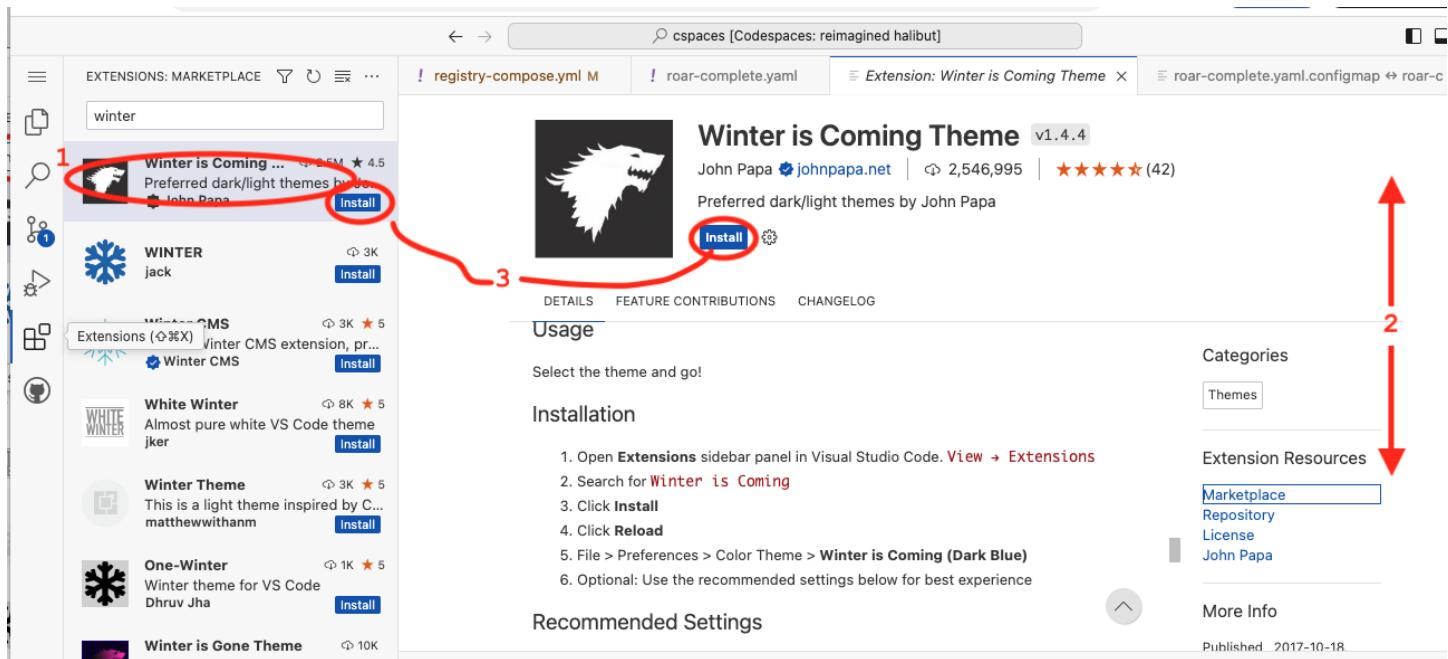
```

On the right is a 'Simple Browser' pane displaying the web application. The title bar says 'Simple Browser X'. The address bar shows the URL 'https://friendly-space-rob'. The page content includes the heading 'Brent's Calc' and instructions: 'Enter a number in the first box and a second box and select the answer button. Change the operation via the dropdown'.

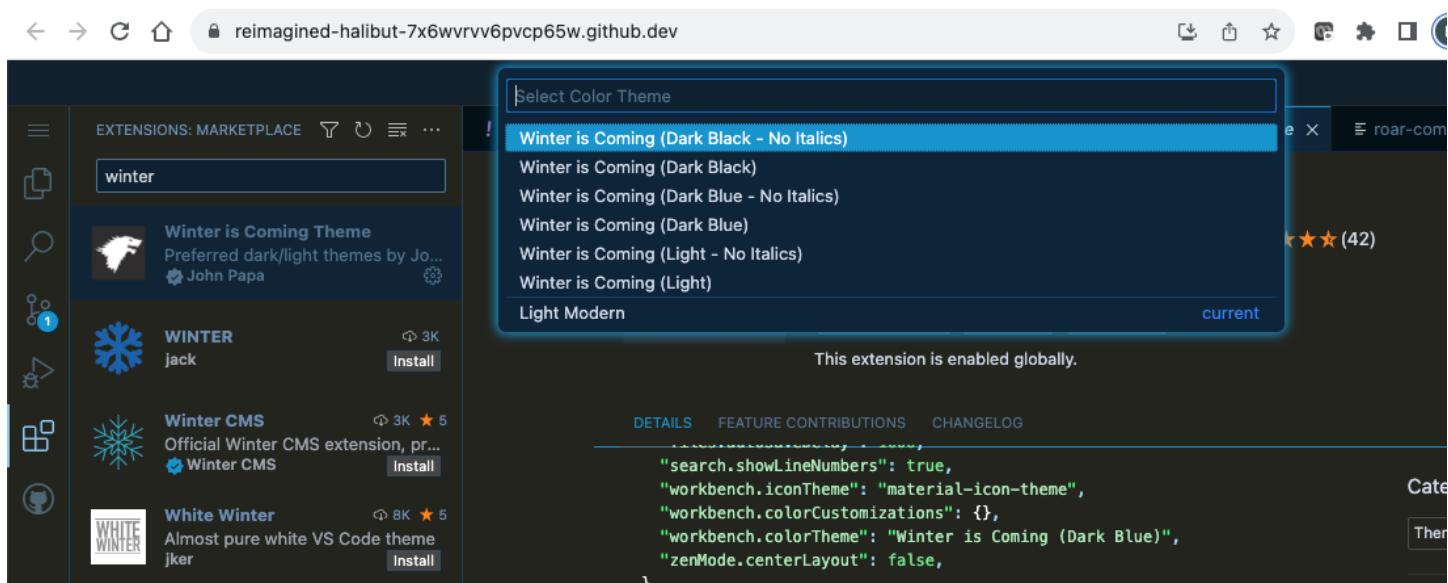
6. Next, let's see how to modify our codespace environment. We'll install an extension for the color theme. For practice, we'll use the "Winter is Coming" theme. Click on the *Extensions icon* (#1 in figure below), then in the *search bar* type in "winter" to quickly find the extension (#2).



7. Once found, you can directly install the extension (#3 in figure below) or click on it (#1) and bring up the info in an editor page and scroll around it (#2) to get more details. Go ahead and install it (via the *Install* button) when ready.



8. After installing, you'll see a list where you can select one of the new color themes. You can choose another one from the list if desired. (If you happen to get an error, try clicking on *Set Color Theme* and pick again.)



END OF LAB

## Lab 11: GitHub Command Line

**Purpose:** In this lab, you'll get to work with the GitHub CLI.

1. In your codespace, the GitHub command line interface (CLI) is already installed. You can try it out from the terminal. Click in the terminal and run the command **gh** by itself to see available options. You can page through the output.

**\$ gh | more**

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2 COMMENTS

○ @brentlaster + /workspaces/calc (cspace) $ gh | more
Work seamlessly with GitHub from the command line.

USAGE
  gh <command> <subcommand> [flags]

CORE COMMANDS
  auth:      Authenticate gh and git with GitHub
  browse:    Open the repository in the browser
  codespace: Connect to and manage codespaces
  gist:      Manage gists
  issue:    Manage issues
  org:      Manage organizations
  pr:       Manage pull requests
  project:  Work with GitHub Projects.
  release:  Manage releases
  repo:     Manage repositories

GITHUB ACTIONS COMMANDS
  
```

2. Take a look at the codespaces you have with the following command:

**\$ gh codespace list**

3. For some commands, you need to set the default remote. Set it now to your current repo.

**\$ gh repo set-default <github-userid>/calc**

4. Let's look at the issue you created in this repo for the earlier lab. (If your issue number was not 1, then use the appropriate issue number.) First, we'll look at it in the terminal and then in the browser.

**\$ gh issue view 1**

**\$ gh issue view 1 --web**

5. You can also do the same for one of your pull requests – just pick a number of one of them.

```
$ gh pr view 1
```

```
$ gh pr view 1 --web
```

6. You can also clone repos easily with the command line. Change up one directory to not clone within the current directory. Then run the command to clone down your other repo.

```
$ cd ..
```

```
$ gh repo clone <github-userid>/greetings-ci
```

OPTIONAL:

7. Your codespace will time out eventually. But if you want stop it now or delete it, etc., you can go to <https://github.com/codespaces> and click on the ... in its row to manage it.

The screenshot shows the GitHub Codespaces interface. At the top, there's a search bar and various navigation icons. Below that, a sidebar on the left lists repositories: 'All' (with 1 item) and 'Templates'. Under 'By repository', there's a card for 'gwstudent/calc' with a '1' next to it. The main area is titled 'Your codespaces' and contains a section for 'Explore quick start templates' with cards for 'Blank', 'React', and 'Jupyter Notebook'. Below that is a section for 'Owned by gwstudent' which shows the 'friendly goldfish' codespace. To the right of the codespace card is a detailed view with a 'Use' button and a context menu. The context menu includes options like 'Rename', 'Export changes to a branch', 'Change machine type', 'Stop codespace', 'Auto-delete codespace' (which is checked with a checkmark), 'Open in Browser', 'Open in Visual Studio Code', 'Open in JetBrains Gateway', 'Open in JupyterLab', and 'Delete'. A red circle highlights the three-dot menu icon at the bottom right of the codespace card.

END OF LAB

**Demo: Copilot**

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.

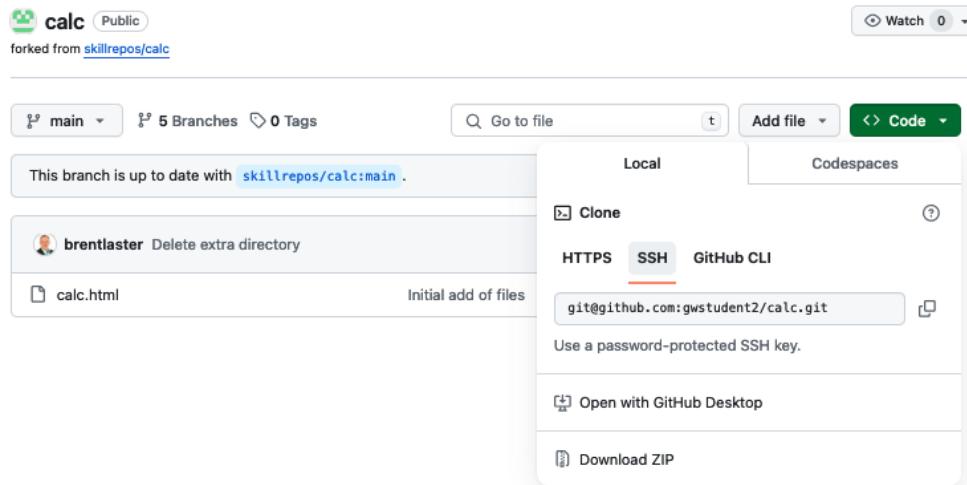
(Note: If you prefer to disable the global credentials helper entirely, you can try

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

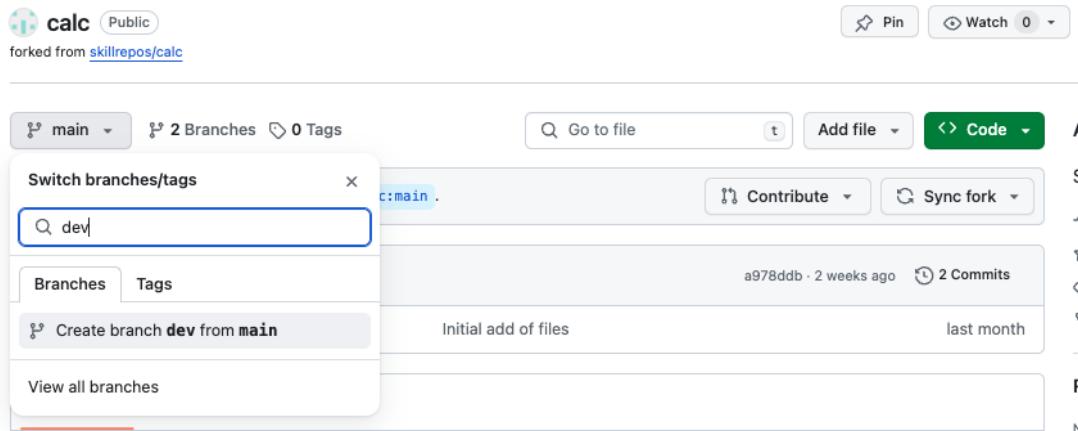
This may or may not work depending on if you have access to do this.)

**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.

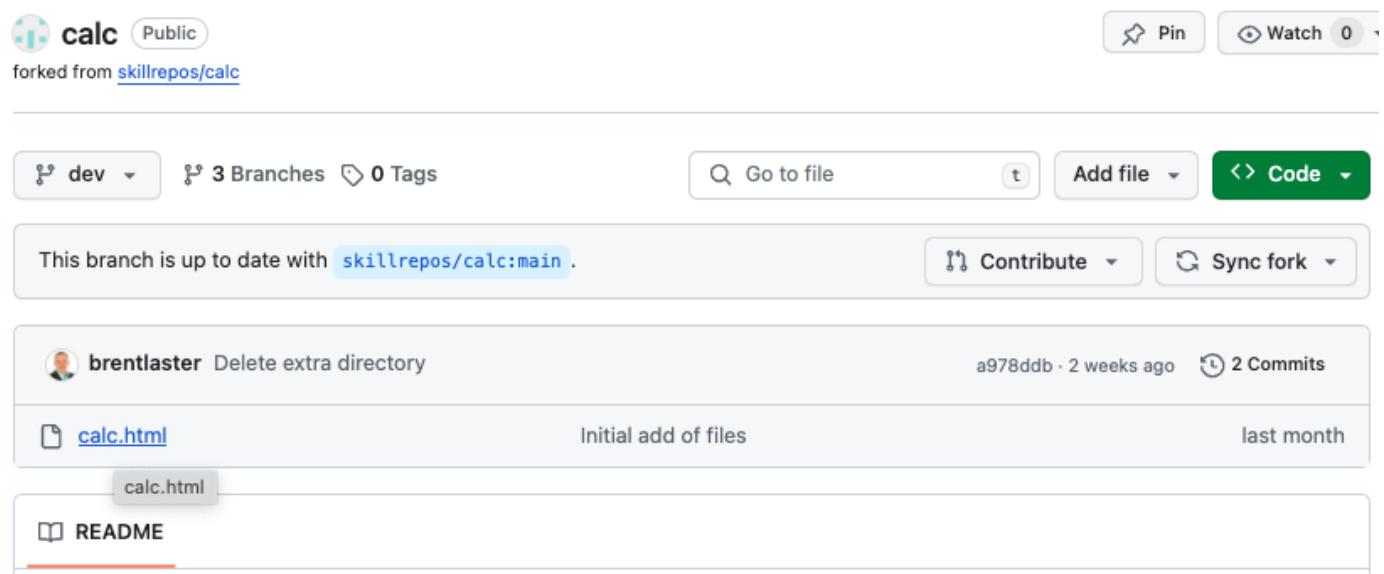
If you go this route, when you get the remote URL from the browser, select the SSH tab.



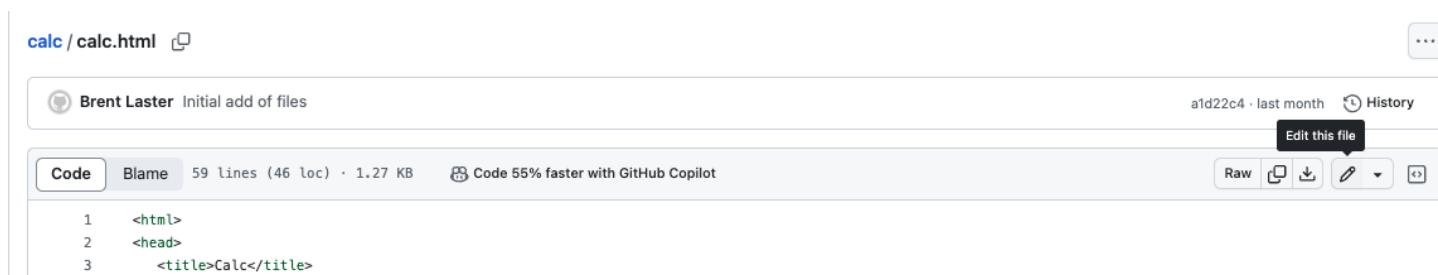
**C. Commit directly in GitHub:** Another option is to commit directly to GitHub in the browser. To do this, first create a *dev* branch in the repo. Click on the branch dropdown under the title of the repo. In the *Find or create a branch* field, type **dev**. Then click on **Create branch dev from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



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



Click on the pencil icon to edit the file.



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 named 'gwstudent / calc'. The user is editing the file 'calc.html' in the 'dev' branch. A modal dialog titled 'Commit changes' is open, prompting for a commit message. The message 'Update calc.html' is entered in the text 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 buttons are present: '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, and code editor settings like 'Spaces', '2', and 'No wrap' on the right.