

GitHub Fundamentals BootCamp

Learn the complete GitHub – from code management to Copilot

Revision 1.2 – 01/03/24

Tech Skills Transformations LLC / Brent Laster

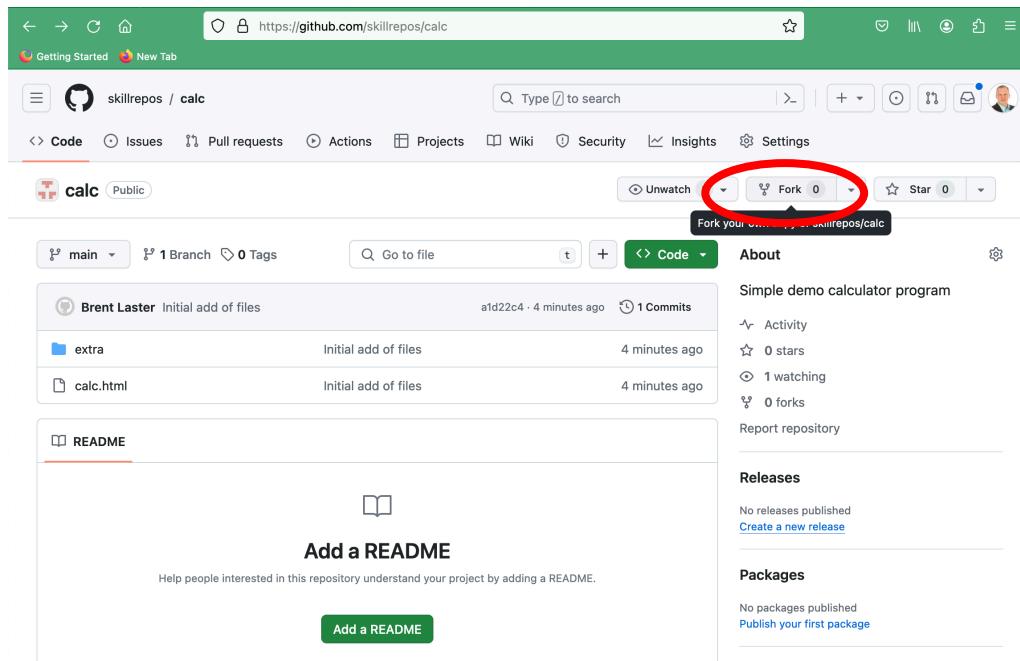
Setup and prerequisites

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.

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.



The screenshot shows the GitHub interface for creating a new fork of a repository named 'calc' from 'skillrepos'. The 'Code' tab is selected. A yellow callout points to the 'Owner' dropdown, which is set to 'brentlaster'. A red circle highlights the checkbox labeled 'Copy the main branch only', which is checked. Another red circle highlights the green 'Create fork' button at the bottom right.

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.

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.

The screenshot shows the GitHub interface for the forked repository 'calc' owned by 'brentlaster'. The 'Code' tab is selected (marked with a red circle 1). The repository details show it was forked from 'skillrepos/calc'. The 'Local' tab is selected (marked with a red circle 3). The 'Clone' section shows options for 'HTTPS', 'SSH', and 'GitHub CLI'. The 'HTTPS' option is highlighted with a red circle 4, and the 'Copy url to clipboard' button is also highlighted with a red circle 5. A red circle 2 highlights the green 'Code' button in the top right corner of the code area.

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 SSH GitHub CLI

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

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 then change to that directory.

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

```
$ cd calc
```

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

```
$ git remote -v
```

6. 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>. The process is described below.

Edit calc.html and change

<title>Calc</title>i

to

<title> *github_user_id*'s Calc</title>

substituting in your GitHub user ID for “github_user_id”.

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

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

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

To create your PAT, follow the instructions for creating a classic token at

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-personal-access-token-classic>

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

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

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 type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input 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

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.

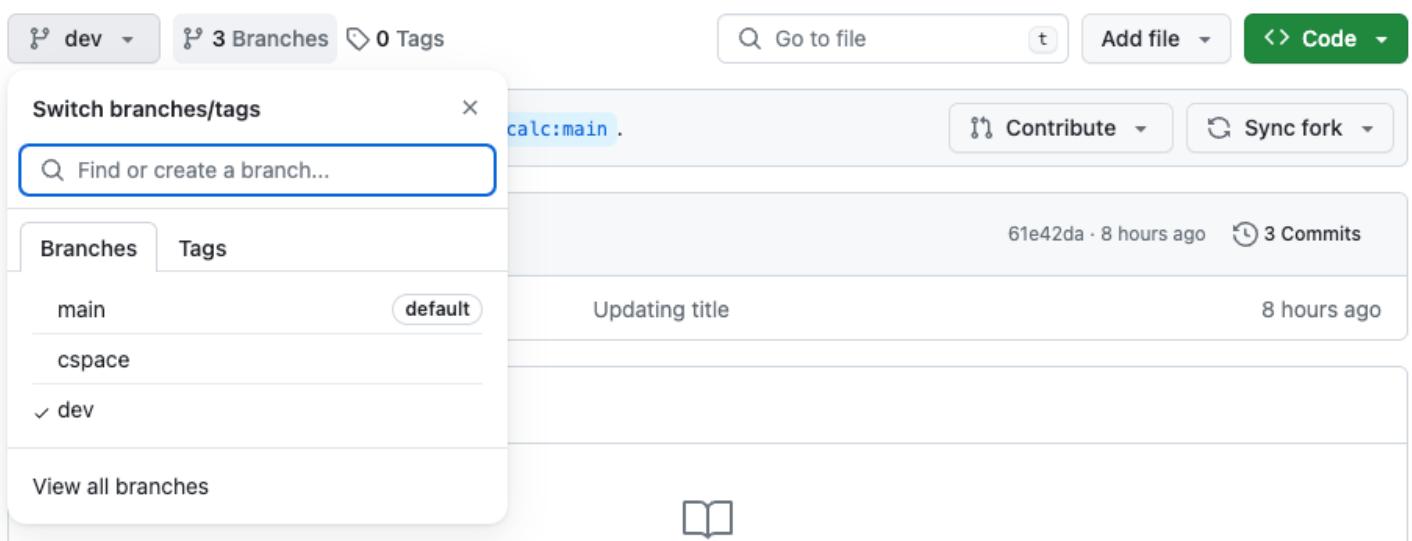
```
developer@Bs-MacBook-Pro calc % vi calc.html
developer@Bs-MacBook-Pro calc % git commit -am "Up
[main d9e79db] Updating title
 1 file changed, 2 insertions(+), 2 deletions(-)
developer@Bs-MacBook-Pro calc % git push -u origin
Username for 'https://github.com': brentlaster
Password for 'https://brentlaster@github.com':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

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 in the browser.)



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](#)

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

Code **Blame** 59 lines (46 loc) · 1.29 KB

Older  Newer

last week	Initial add of files	1	<html>
22 minutes ago	Updating title	2	<head>
last week	Initial add of files	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.

Code **Blame** 59 lines (46 loc) · 1.29 KB

Older  Newer

Brent Laster Updating title · d9e79db · 24 minutes ago · **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.

<> Code Pull requests Actions Projects Wiki Security Insights Settings

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

dev

Brent Laster committed 28 minutes ago 1 parent a1d22c4 commit d9e79db

Browse files

Showing 1 changed file with 2 additions and 2 deletions.

Whitespace Ignore whitespace Split Unified

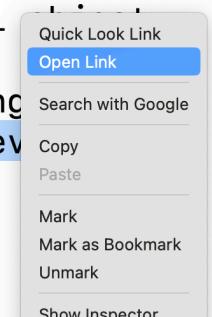
calc.html

...	...	@@ -1,6 +1,6 @@
1	1	<html>
2	2	<head>
3	3	- <title>Calc</title>
	3	+ <title>brentlaster's Calc</title>
4	4	<script language=javascript type="text/javascript">
6	6
56	56	@@ -56,4 +56,4 @@ <h2>Calc</h2>
57	57	...
58	58	</body>
59	59	- </html>
	59	+

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

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

```
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 --

The screenshot shows a GitHub repository page for 'brentlaster / calc'. At the top, there's a yellow banner with the text 'dev had recent pushes 26 minutes ago' and a green 'Compare & pull request' button. Below the banner, there's a gray bar with various repository links like 'Code', 'Pull requests', 'Actions', etc. On the left, there's a profile picture for 'calc' and the text 'forked from skillrepos/calc'. On the right, there are 'Pin' and 'Watch' buttons. The main content area shows a list of recent commits or pushes.

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

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](#).

The screenshot shows the 'Comparing Changes' screen on GitHub. At the top, there are dropdown menus for 'base repository' (set to 'skillrepos/calc'), 'base' (set to 'main'), 'head repository' (set to 'brentlaster/calc'), and 'compare' (set to 'dev'). Below these, there's a dropdown menu for 'Choose a Base Repository' with 'brentlaster/calc' selected. The main area shows a list of commits with the message 'Automatically merged.' There are sections for 'Reviewers' (no reviews) and 'Assignees' (no one assigned). At the bottom, there are buttons for 'Write' and 'Preview'.

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.

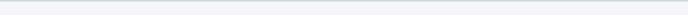
8. Now, with your repo selected for the base, add an optional description if you want and then click on the *Create pull request* button.

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

 Add a title

Updating title

Add a description

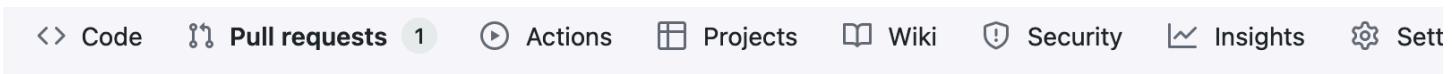
Write Preview 

Merge changes from dev|

 Markdown is supported  Paste, drop, or click to add files

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.



Updating title #1

Open brenlaster wants to merge 1 commit into `main` from `dev`

Conversation 0 **Commits 1** Checks 0 **Files changed 1**

 brenlaster commented now

Merge changes from dev

-o Updating title

d9e79db

Add more commits by pushing to the `dev` branch on [brenlaster/calc](#).

Require approval from specific reviewers before merging
[Branch protection rules](#) ensure specific people approve pull requests before they're merged.

Continuous integration has not been set up
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

- When you're ready, 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.

Open**Updating title #1**brentlaster wants to merge 1 commit into `main` from `dev`

-o-

Updating title

d9e79db

No o

Add more commits by pushing to the `dev` branch on [brentlaster/calc](#).**Merge pull request #1 from brentlaster/dev****Updating title**This commit will be authored by `bclaster@nclasters.org`**Confirm merge****Cancel**

Add a comment

Merged**Updating title #1**brentlaster merged 1 commit into `main` from `dev` now

Merge changes from dev



-o-

Updating title

d9e79db

brentlaster merged commit `dc98b08` into `main` now**Revert****Pull request successfully merged and closed**You're all set—the `dev` branch can be safely deleted.**Delete branch**

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**.

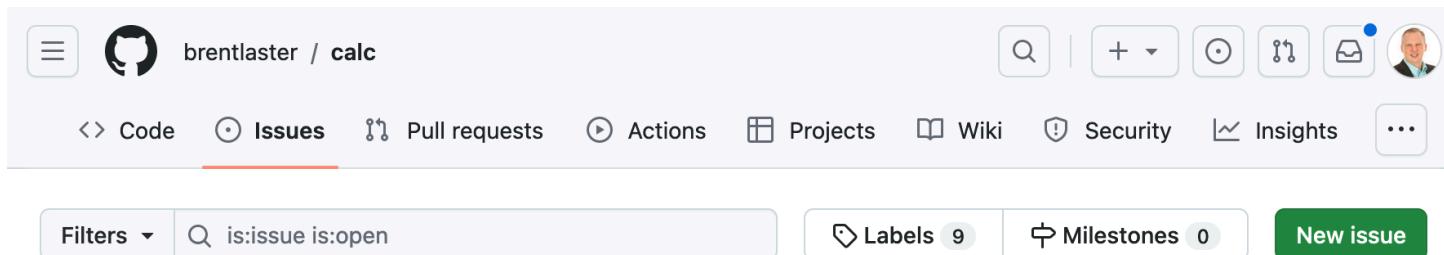


...

Features

<input checked="" type="checkbox"/> Wikis	Wikis host documentation for your repository.
<input checked="" type="checkbox"/> Restrict editing to collaborators only	Public wikis will still be readable by everyone.
<input checked="" type="checkbox"/> Issues ✓	Issues integrate lightweight task tracking into your repository. Keep projects on track with issue labels and milestones, and reference them in commit messages.
<input type="checkbox"/> 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.



Add a title

Needs README

Add a description

Please add README file :book:

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Shows branches and pull requests linked to this issue.

Helpful resources: GitHub Community Guidelines

Submit new issue

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

Needs README #2

Open brentlaster opened this issue now · 0 comments

brentlaster commented now

Please add README file :book:

Add a comment

Add your comment here...

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

The screenshot shows a GitHub interface for creating a new issue. At the top, there are tabs for Wiki, Security, Insights, and a three-dot menu. Below the tabs are two buttons: 'Edit' and 'New issue'. The main area has a light blue header with three dots. Underneath it, the 'Assignees' section is visible, which currently says 'No one—[assign yourself](#)'. There is also a gear icon for settings.

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

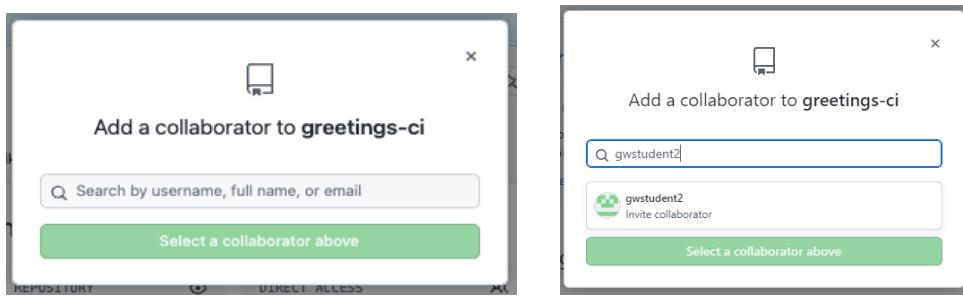
This screenshot shows the 'Labels' section of the GitHub issue editor. It includes a 'Filter labels' input field, a note about 'Something isn't working', and a list of available labels. The 'documentation' label is selected, indicated by a checked radio button and a blue circle. Other labels listed are 'duplicate' (unchecked) and 'Needs README' (unchecked). A note at the bottom 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.

This screenshot shows the GitHub Issues tab for the repository 'brentlaster / calc'. The tab bar includes Code, Issues (with 1 open issue), Pull requests, Actions, Projects, Wiki, Security, and Insights. The Issues tab is active. At the top of the list, there is a search bar with 'Type [] to search' and a 'New issue' button. Below the search bar are filters for 'Filters' (set to 'is:issue is:open'), 'Labels' (9), 'Milestones' (0), and 'Sort'. The first issue in the list is labeled 'Needs README documentation' with a green circle icon. The issue details show '#2 opened 15 hours ago by brentlaster'.

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.

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.

The screenshot shows a GitHub repository page for 'brentlaster / calc'. The top navigation bar includes links for Code, Issues (1), Pull requests, Actions, Projects, Security, and Insights. A search bar at the top right says 'Type ⌘ to search'. Below the navigation, the repository details show a user icon, the name 'calc' (marked as Public), and a link to 'forked from skillrepos/calc'. A 'Watch' button with a count of 0 is also present.

A prominent message box at the top states '@brentlaster has invited you to collaborate on this repository' with a 'View invitation' button on the right, which is circled in red.

The main content area shows two user icons: 'brentlaster' and another user whose profile picture is a green Mario head. The text 'brentlaster invited you to collaborate' is displayed. Below it are two buttons: 'Accept invitation' (circled in red) and 'Decline'.

Information about what the owner can see if accepted:

- 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

Below this, there's a question 'Is this user sending spam or malicious content?' with a 'Block brentlaster' link.

The screenshot shows the same GitHub repository page after accepting the invitation. The top navigation bar and repository details are identical. A message at the bottom of the page says 'You now have push access to the brentlaster/calc repository.' with a close button (X).

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 'Code' tab is selected. In the main area, the 'README' file is open. At the bottom of the editor, there is a green button labeled 'Add a README'. This button is circled in red to indicate it should be clicked. The rest of the page shows the repository's structure, recent commits, and other file details.

- 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 “j” 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 code editor shows the following content:

```
1 This is a simple calculator :abacus: program. :question: can be directed to @brentlaster(https://github.com/brentlaster)
2
```

The interface includes tabs for 'Edit' and 'Preview', and buttons for 'Cancel changes' and 'Commit changes...'. There are also settings for 'Spaces', '2', and 'Soft wrap'.

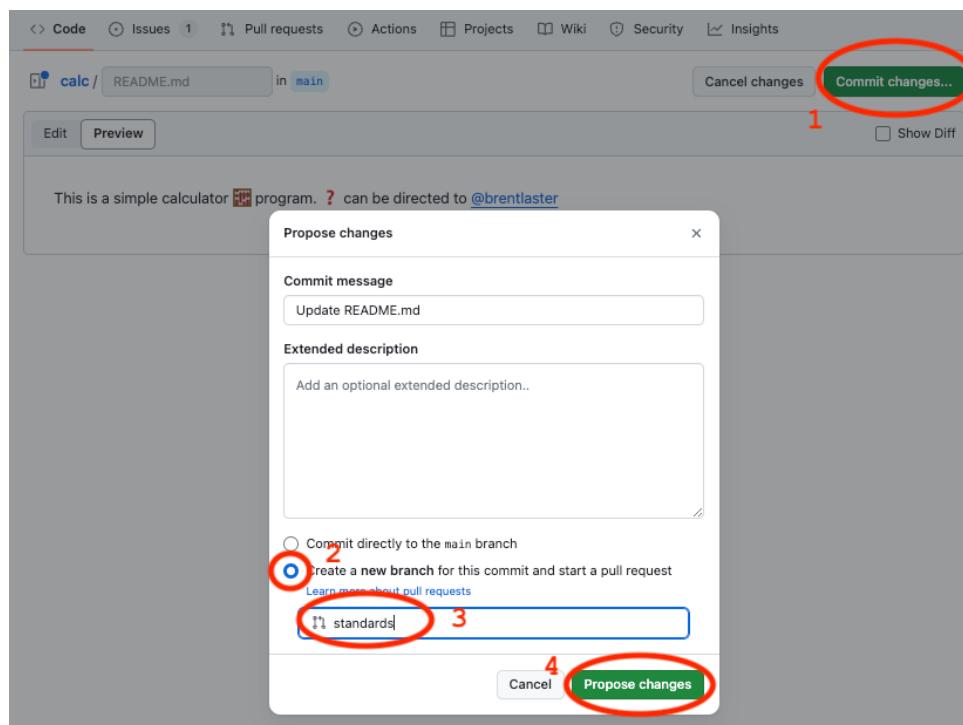
3. Click on the Preview tab (next to Edit) to see how this will render once committed.

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 program. ? can be directed to [@brentlaster](#)

The interface includes tabs for 'Edit' and 'Preview', and a checkbox for 'Show Diff'.

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 'Comparing changes' interface. At the top, there are dropdown menus for 'base: main' and 'compare: standards'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' Below this, a message encourages discussing changes with others and provides a link to 'Learn about pull requests'. A prominent red circle highlights the 'Create pull request' button. The main content area shows a commit from 'brentlaster' on Jan 1, 2024, titled 'Create README.md'. The commit message includes a link to a GitHub profile. A diff view shows one addition and zero deletions in the README.md file. At the bottom, there are 'Split' and 'Unified' options.

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. Then click on **Create pull request**.

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 comparisons here.

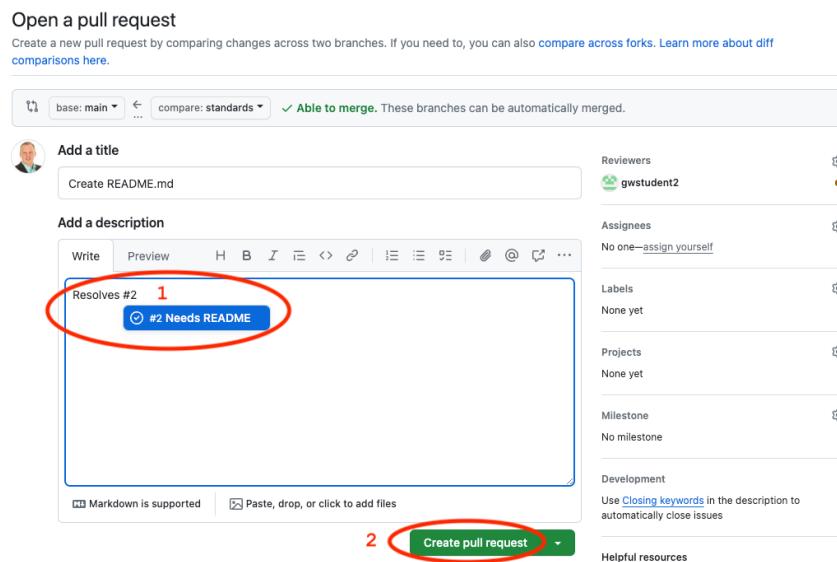
The screenshot shows the 'Open a pull request' page. The 'base: main' and 'compare: standards' dropdowns are visible, along with the 'Able to merge' message. The 'Reviewers' section is highlighted with a red circle, showing a count of '1'. A dropdown menu lists 'Request up to 15 reviewers' and a search bar. A user named 'gwstudent2' is selected, indicated by a blue background and a circled '2'. Other sections include 'Add a title' (with a user icon), 'Add a description' (with a rich text editor and placeholder 'Add your description here...'), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Development' (with a note about closing keywords). A 'Create pull request' button is at the bottom.

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

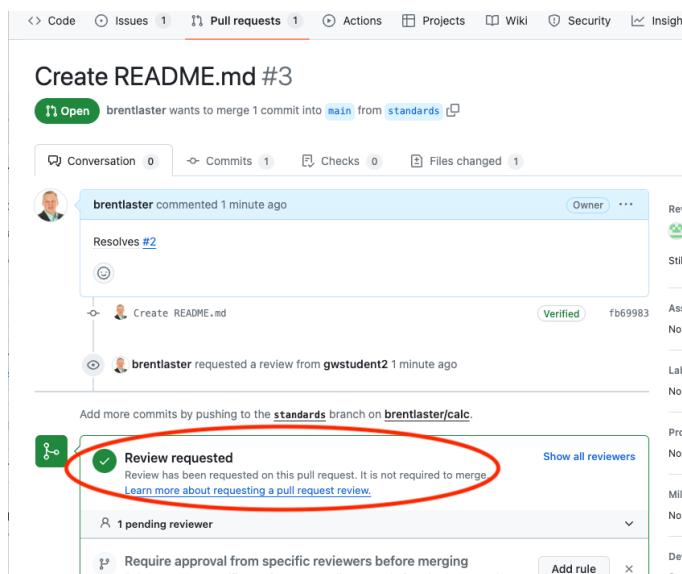
Resolves #2

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, a refresh button, and a mail icon. A red circle with the number '1' is positioned next to the mail icon. Below the header, there are tabs for 'Inbox' (with 1 notification), 'All', 'Unread', and a filter for 'is:unread'. The main area displays notifications categorized by type: 'Saved' (0), 'Done' (0), 'Filters' (0), 'Assigned' (0), 'Participating' (1), 'Mentioned' (0), 'Team mentioned' (0), and 'Review requested' (1). The 'Review requested' section shows a single notification for a pull request from 'brentlaster/calc' with the title 'Create README.md'. This notification is circled with a red '2'. Below the notifications, there's a tip about creating custom filters and a navigation bar with '1-1 of 1', 'Prev', and 'Next' buttons.

- 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 main content shows a table with one open pull request. The first row of the table is circled with a red '2'. The table has columns for 'Created', 'Assigned', 'Mentioned', and 'Review requests'. The 'Review requests' column is currently selected. A search bar at the top right contains the query 'is:open is:pr review-requested:gwstudent2 archive'. The table shows 1 open pull request from 'brentlaster/calc' with the title 'Create README.md'. A note at the bottom says 'ProTip! Add [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”.

Finish your review

Suggest adding a license file too.

Comment
Submit general feedback without explicit approval.

Approve
3
Submit feedback and approve merging these changes.

Request changes
Submit feedback that must be addressed before merging.

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.

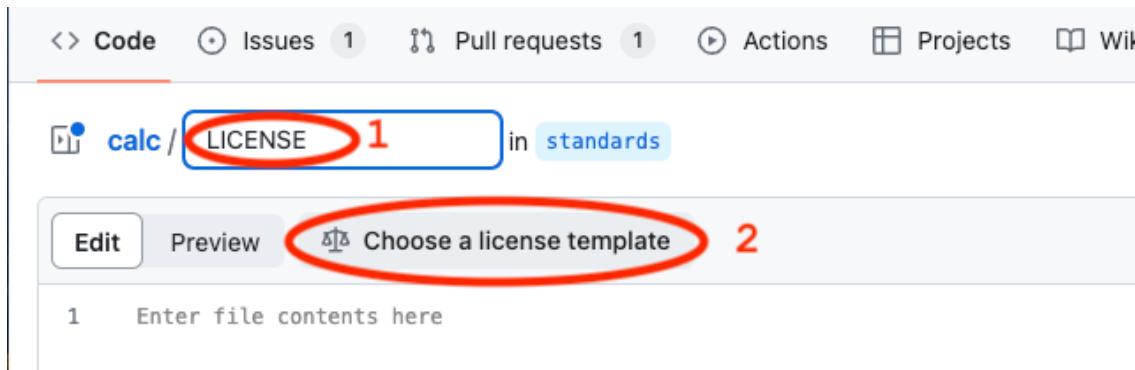
The screenshot shows a GitHub pull request interface. At the top, it says "Create README.md #4". Below that, it indicates "brentlaster wants to merge 1 commit into main from standards". The conversation shows the following messages:

- brentlaster commented 41 minutes ago: "No description provided."
- brentlaster (Verified) 47b11e1: "Create README.md"
- brentlaster requested a review from gwstudent2 41 minutes ago
- gwstudent2 approved these changes 1 minute ago: "View reviewed changes"
- gwstudent2 left a comment: "Suggest adding a license file too."

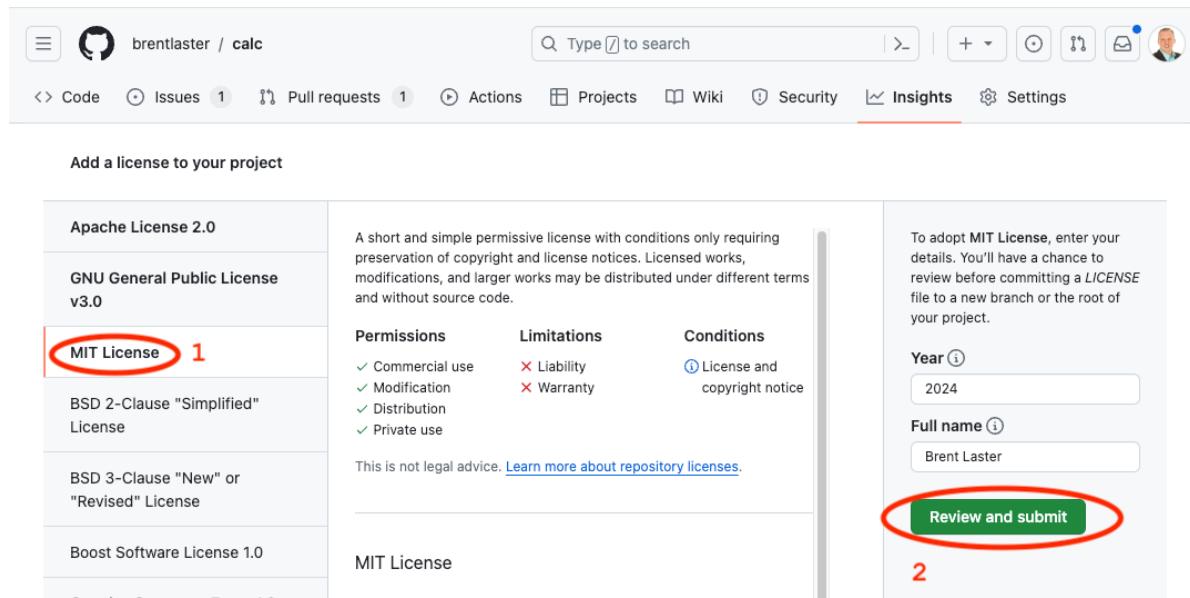
5. Since there was a suggestion to add a license file, that sounds like a good idea, so let's do that. Click on the “Code” tab at the top, then select the “standards” branch from the branch dropdown, then select the “+” sign and the option to “+ Create new file”.

The screenshot shows a GitHub repository page for the 'calc' repository. The 'Code' tab is circled in red with the number 1. The 'standards' branch is selected in the dropdown menu, circled in red with the number 2. A context menu is open over the '+' button, with the 'Create new file' option circled in red with the number 4. The menu also includes 'Upload files'.

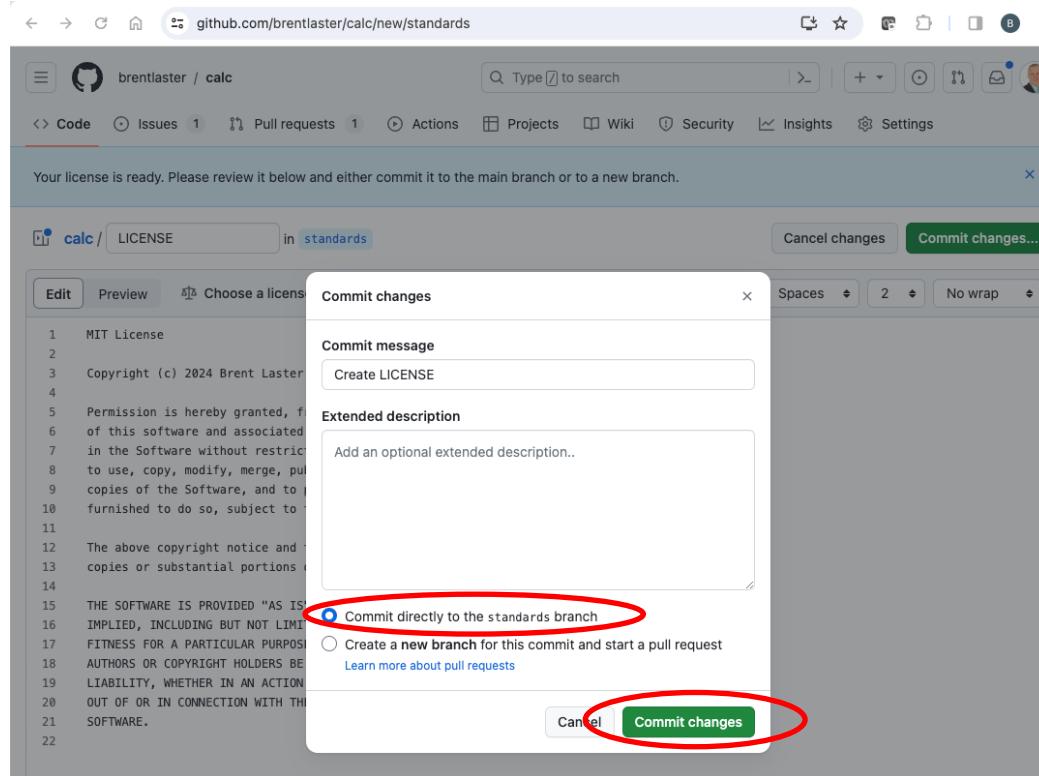
6. In the next screen, there will be a text entry area for the name of the file. Type in “LICENSE” for the name. Then, an option will display that says “Choose a license template”. Click on that option. You will be asked about discarding changes. It’s ok in this case, so click on “OK”.



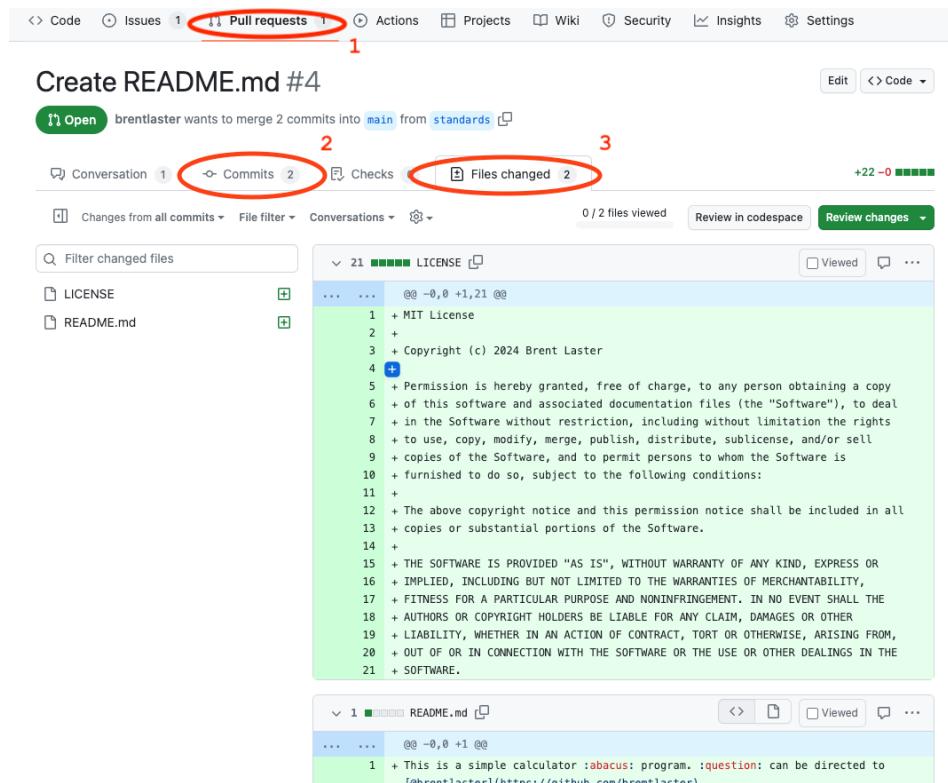
7. On the next screen, you’ll be able to pick the license you want. You can select the “MIT License” or another one if you prefer. Once done, click the “Review and submit” button on the right.



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 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 'Issues' tab is selected. A single issue titled 'Needs README #2' is shown, which is now closed. The issue was opened by brentlaster 19 hours ago. The conversation log includes:

- 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

On the right side of the issue page, there are sections for Assignees (brentlaster), Labels (documentation), Projects (None yet), Milestone (No milestone), Development (Successfully merging a pull request may close this issue), and Notifications (Customize, Unsubscribe). It also notes that notifications are being received because the issue was modified.

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.

1. In order to prepare for publishing a page, let's create a new branch in our repo. In the **Code** tab, click on the branch dropdown that says **main**. Then 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 interface in the 'Code' tab. The 'main' branch is selected. A modal window titled 'Switch branches/tags' is open, showing a search bar with 'pages' typed in. Below the search bar are tabs for 'Branches' and 'Tags'. At the bottom of the modal, a button is circled in red, labeled 'Create branch pages from main'.

- 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 <> item. You can optionally add a description if you want.

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

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 **Repository name *** 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.

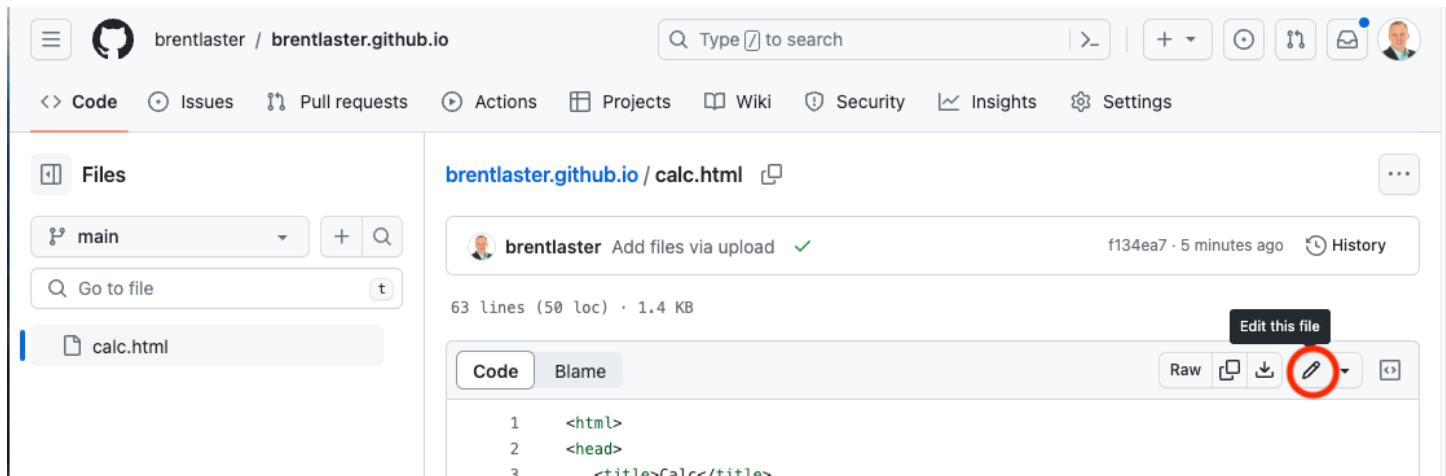
- So that we have content to publish, we'll grab the code from the calc.html file in your local repository from Lab 1 and add it here. On the screen with the **Quick setup – if you've done this kind of thing before** instructions, click on the link in the big blue bar for uploading an existing file.

The screenshot shows the GitHub repository setup page for `brentlaster/brentlaster.github.io`. It features sections for 'Start coding with Codespaces', 'Add collaborators to this repository', and a 'Quick setup' section. The 'Quick setup' section includes links for 'Set up in Desktop' (disabled), 'HTTPS' (selected), 'SSH', and a URL. A note says 'Get started by creating a new file or uploading an existing file.' The 'uploading an existing file' link is circled in red.

- 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. Then click on the **Commit changes** button to add the file to the repo.

The screenshot shows the GitHub file upload interface. Step 1 highlights the area where files can be dragged or selected. Step 2 highlights the file `calc.html` in the file list. Step 3 highlights the **Commit changes** button.

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.

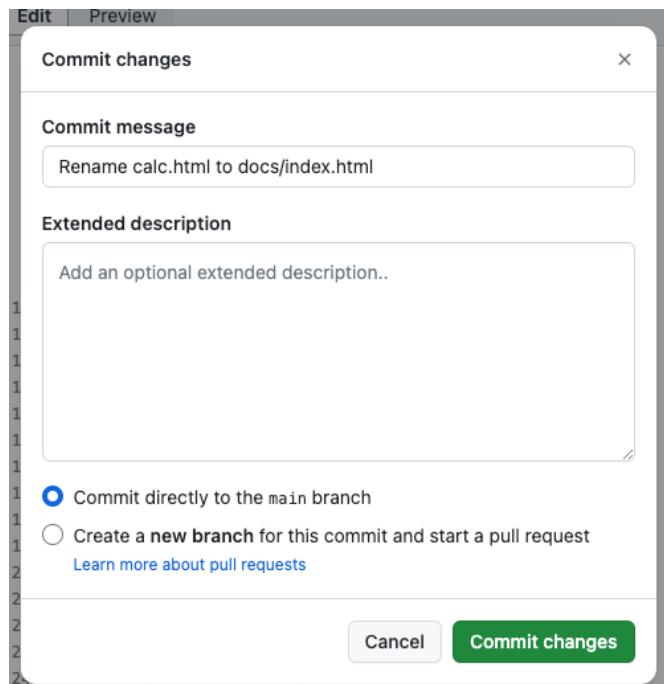


5. In the edit dialog, in the text entry box for the name, type over the "calc.html" text with the replacement text of "docs/index.html". The screenshots show this in 2 parts for clarity.

The image contains two screenshots of the GitHub Code tab, illustrating the renaming process:

- Screenshot 1:** Shows the 'Edit' dialog for the file 'calc.html'. The file path 'brentlaster.github.io / docs' is highlighted with a red oval. The 'Edit' button is visible at the bottom left of the dialog.
- Screenshot 2:** Shows the 'Edit' dialog after the file has been renamed to 'index.html'. The file path 'brentlaster.github.io / docs / index.html' is highlighted with a red oval. The file content remains the same as in the first screenshot.

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.

1

2

3

4

Optional

8. After these changes, you can visit the site at <https://<github-userid>.github.io> and see the automatic web page.

9. 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**.

10. 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.

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 to uncheck the box next to *Copy the main branch only***, so that both branches will be included in the fork.

The top screenshot shows the original repository 'greetings-ci' by 'skillrepos'. The bottom screenshot shows the forked repository 'greetings-ci' by 'brentlaster'.

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 *



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](#).

(i) You are creating a fork in your personal account.

Create fork

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 a GitHub repository page for `gwstudent/greetings-ci`. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. Below the repository name, it says "forked from skillrepos/greetings-ci". The main navigation bar has tabs for Code, Pull requests, Actions (which is circled in red), Projects, Wiki, Security, Insights, and Settings. Under the Code tab, there are dropdowns for main branch, 1 branch, and 0 tags. A green "Code" button is highlighted. Below this, a message states "This branch is up to date with skillrepos/greetings-ci:main.". A list of commits is shown, all made by "Brent Laster" and pushed 7 minutes ago, with commit IDs like 97df205 and 2 commits. The right sidebar contains sections for About (repo description), Releases (no releases), and Packages (no 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 GitHub Actions category page. It features four cards under the "Automation" section: Greetings, Stale, Manual workflow, and Labeler. Below this, a "Browse all categories" section lists Automation, Continuous integration (which is circled in red), Deployment, and 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 search bar at the top has 'Gradle' typed into it and is circled in red. Below the search bar, there's a sidebar with categories: Automation, Continuous Integration (which is selected and highlighted in blue), Deployment, and Security. The main area displays search results for 'Found 52 workflows'. There are several cards for different workflows, including 'Android CI', 'Java with Ant', 'Clojure', 'Publish Java Package', 'Java with Gradle', and another 'Publish Java Package'. 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](#)

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

This screenshot is similar to the previous one, showing the search results for 'Gradle'. However, the 'Java with Gradle' workflow is now highlighted with a red circle around its 'Configure' button, indicating it has been selected.

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](#)

Categories

Automation
Continuous integration (selected)
Deployment
Security

Q Gradle

Found 3 workflows

This screenshot shows the search results again, but the 'Java with Gradle' workflow is now circled in red around its 'Configure' button, indicating it is the current target for configuration.

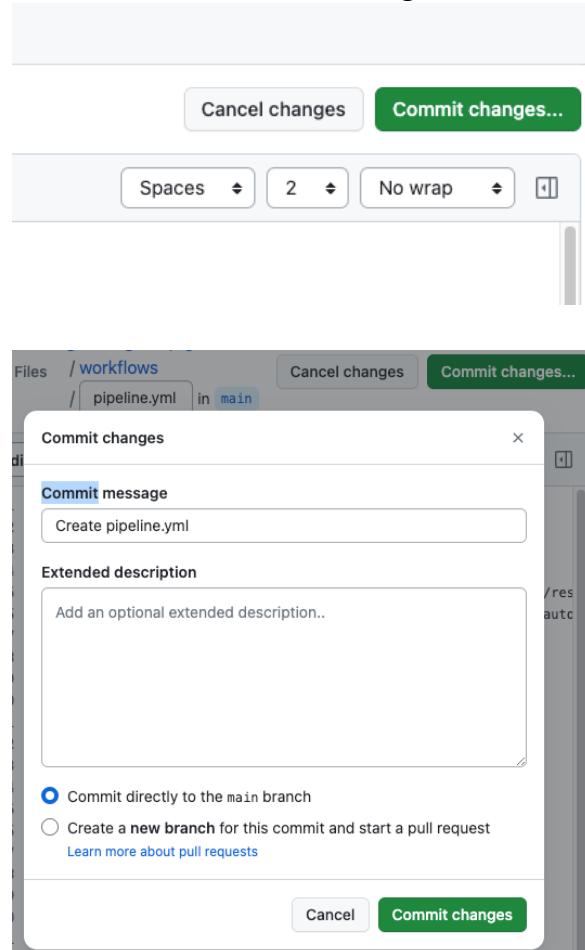
- This will bring up a page with a starter workflow for CI that we can edit as needed. The only edit we want to make here right now is to change the name. In the top section where the path is, notice that there is a text entry box around "gradle.yml". This is the current name of the workflow. Click in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

The screenshot shows the GitHub Actions pipeline configuration page for the 'gradle.yml' file. At the top, there's a navigation bar with links for Code, Pull requests, Actions, Projects, Wiki, and Security. Below that, the file path 'greetings-ci/.github/workflows/gradle.yml' is shown, with 'gradle.yml' highlighted in blue and 'in main' indicating the branch. There are two tabs at the bottom: 'Edit new file' (selected) and 'Preview'.

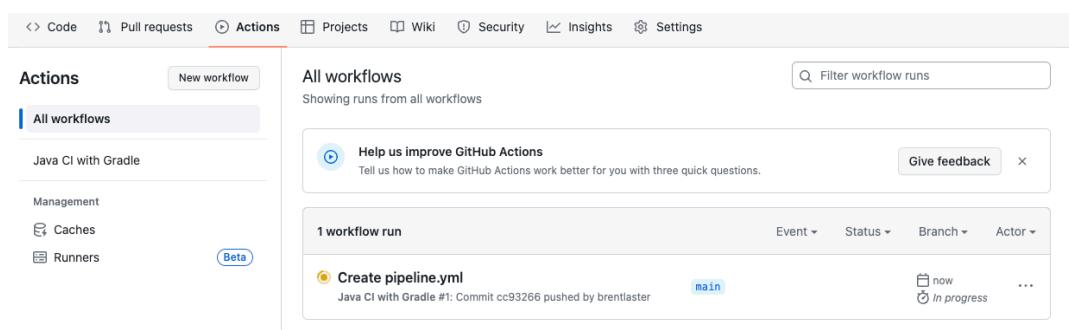
TO

The screenshot shows the GitHub Actions pipeline configuration page after renaming the file to 'pipeline.yaml'. The URL now shows 'greetings-ci/.github/workflows/pipeline.yaml'. The rest of the interface is identical to the previous screenshot, with the 'Edit new file' tab selected.

8. 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.



9. 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.



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', 'New workflow', 'All workflows' (listing 'Java CI with Gradle'), 'Management', 'Caches', 'Runners', and a 'Beta' button. The main area is titled 'Java CI with Gradle' and shows 'pipeline.yml'. It includes a 'Help us improve GitHub Actions' card and a '1 workflow run' section. The run is labeled 'Create pipeline.yml' with a green checkmark, indicating success. The run was triggered by a push from 'brentlaster' to the 'main' branch, 1 minute ago, with 36s duration. A red oval highlights the 'Create pipeline.yml' link in the run list.

11. 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.

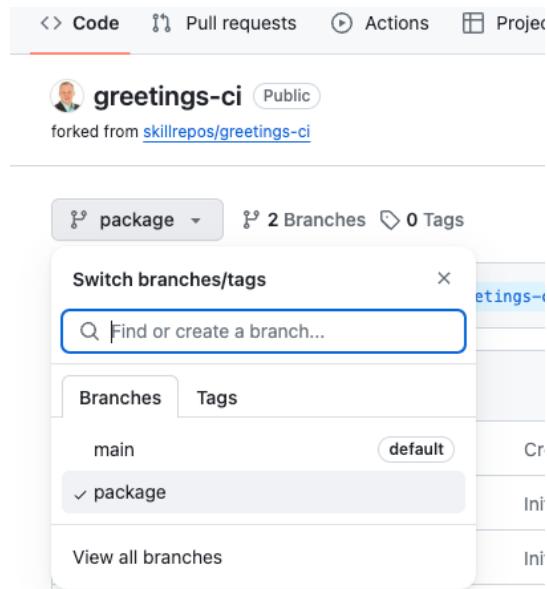
The screenshot shows the detailed view of the 'Create pipeline.yml' workflow. The left sidebar has 'Actions' selected, showing 'Java CI with Gradle' and 'Create pipeline.yml #1'. The main area shows the 'Summary' tab. Under 'Jobs', 'build' is selected, indicated by a red oval. The 'build' job summary shows it succeeded 3 minutes ago in 25s. The job graph shows two steps: 'Set up job' and 'Run actions/checkout@v3', both with green checkmarks. The 'Run actions/checkout@v3' step is expanded, showing its sub-steps: 'Run actions/checkout@v3', 'Syncing repository: brentlaster/greetings-ci', 'Getting Git version info', and 'Temporarily overriding HOME=/home/runner/work/_temp/77610fae-ac20-4aea-98ae-8803530bce6e before making global git config changes'. A red oval highlights the 'Run actions/checkout@v3' step in the 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.



2. Let's create a pull request to merge those into *main*. Go the Pull Requests menu and open a new pull request to merge the “package” branch into the “main” branch - **on your fork** NOT skillrepos/greetings-ci. **Make sure to set *base = main* and *compare = package*** in the gray bar so you are merging in the same repo and NOT into skillrepos/greeting-ci. Go ahead and create the pull request.

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

Merge package into main #24

Open brentlaster wants to merge 3 commits into `main` from `package`

Conversation 0 Commits 3 Checks 0 Files changed 11

Changes from all commits File filter Conversations

`Filter changed files`

`.github/workflows/publish-package.yml`

```

    ...
    ...
    @@ -0,0 +1,25 @@
    1 + name: Publish package to GitHub Packages
    2 + on:
    3 +   release:
    4 +     types: [created]
    5 +     workflow_dispatch:
    6 +       jobs:

```

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. Notice that it has a `workflow_dispatch` trigger. This allows the workflow to be invoked manually. 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.

Code Pull requests Actions Projects Wiki Security Insights Settings

Actions New workflow

All workflows

Java CI with Gradle

Publish package to GitHub Packages

Management Caches Runners

Help us improve GitHub Actions Tell us how to make GitHub Actions work better for you with three quick questions. Give feedback

0 workflow runs Event Status Branch Actor

This workflow has a `workflow_dispatch` event trigger.

Run workflow

Use workflow from Branch: main

Run workflow

0 workflow runs Event Status Branch Actor

This workflow has a `workflow_dispatch` event trigger.

Run workflow

Publish package to GitHub Packages Publish package to GitHub Packages #1: Manually run by brentlaster main 1 minute ago 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 top right, there's a 'Packages' section with one entry: 'org.gradle.sample.greetings-ci'. This entry is circled in red.

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 user profile for 'brentlaster'. The 'Packages' tab is selected. In the top right, there's a 'Your profile' section with one package: 'org.gradle.sample.greetings-ci'. This entry is circled in red.

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. Try this for the **greetings-ci-1.1 jar** file.

The screenshot shows a GitHub repository page for 'org.gradle.sample.greetings-ci'. On the right side, under the 'Assets' section, a file named 'greetings-ci-1.1.jar' is listed with a download count of 1,006 B and a timestamp of 1 minute ago. This file is circled in red.

END OF LAB

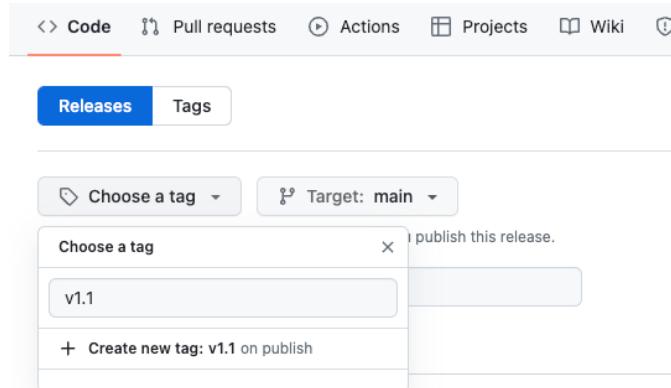
Lab 9: Creating a release

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

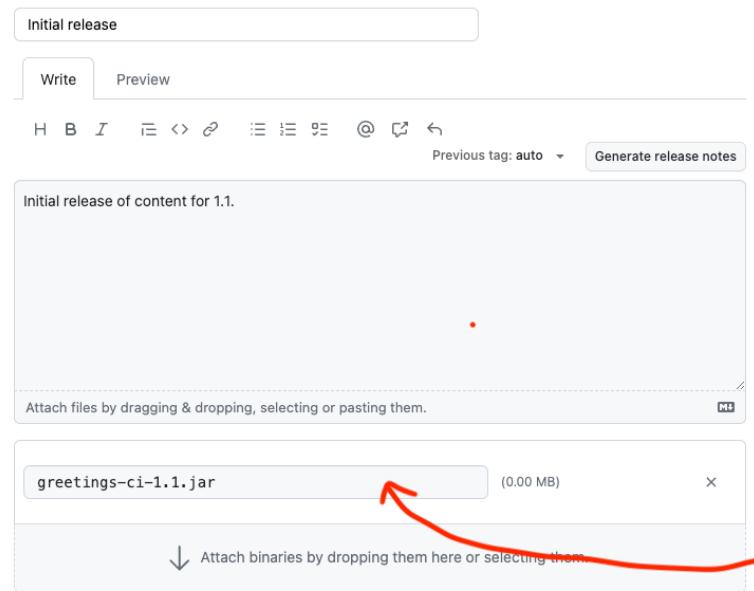
- On the **Code** tab of the *greetings-ci* repo, on the right-hand side, find the **Releases** section and click on the **Create a new release** link. (You can also go directly to the page by going to <https://github.com/<github-userid>/greetings-ci/releases/new>)

The screenshot shows the GitHub repository page for 'greetings-ci'. On the right side, under the 'Releases' section, there is a message 'No releases published' followed by a blue link 'Create a new release'. This link is circled in red.

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. Drag and drop from the local download you did at the end of the last lab to add the greetings-ci.jar file to the release.



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



Set as a pre-release
This release will be labeled as non-production ready

Publish release **Save draft**

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

The screenshot shows the GitHub Releases page for a repository. 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 has a title 'Initial release' with a 'Latest' button. It shows a profile picture of 'brentlaster' and the message 'released this now'. A description below reads 'Initial release of content for 1.1.'. Under the 'Assets' section, there is a table with three rows: 'greetings-ci-1.1.jar' (1006 Bytes, 2 minutes ago), 'Source code (zip)' (21 minutes ago), and 'Source code (tar.gz)' (21 minutes ago).

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 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. It shows the repository is public and forked from 'skillrepos/greetings-ci'. Below this, there are buttons for Pin, Watch (0), Fork (140), and Star (0). The main content area shows a branch status: 'main' is 2 commits ahead of 'skillrepos/greetings-ci:main'. There is a list of recent commits by 'brentlaster' and other contributors. On the right side, there are sections for 'About' (Simple starter repo for CI/CD with GitHub Actions), 'Activity' (0 stars, 0 watching, 140 forks), 'Releases' (1 release, circled in red), 'Packages' (1 package), and 'Skills' (org.gradle.sample.greetings-ci).

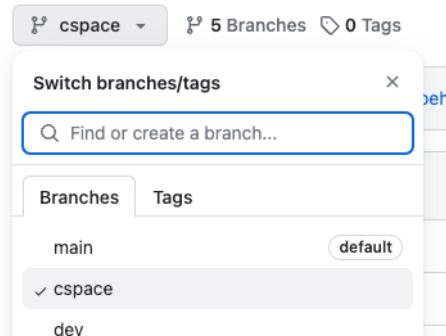
7. You can click on that if you want to see details about the release (same as output in step 5).

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**.

A screenshot of a GitHub repository page for 'skillrepos/calc'. At the top left is a dropdown menu with 'cspac'. To its right are buttons for '5 Branches' and '0 Tags'. On the far right is a green 'Code' button with a white icon and a red number '1+' above it. A red circle highlights this button. To the right of the 'Code' button is a tab labeled 'Codespaces' with a red number '2' above it. A red circle highlights this tab. Below the tabs, a section titled 'Codespaces' shows the message 'Your workspaces in the cloud'. At the bottom of this section is a green button with the text 'Create codespace on cspace' and a red number '3' to its left. A red circle highlights this button. To the right of the 'Create codespace' button is a link 'Learn more about codespaces...'.

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 are also terminal at the bottom. To get back to the main terminal, click on the `bash` selection at the far right side.

The screenshot shows the VS Code interface for a Codespace named "friendly space robot".

- EXPLORER:** Shows the project structure with a file named "calc.html" selected.
- CODEVIEW:** Displays the content of "calc.html". The code includes HTML, CSS, and JavaScript. A specific line of CSS is highlighted: `document.calc.answer.value = a * b`.
- BROWSER:** A preview window titled "Simple Browser" shows the word "Calc". Below it is a form with input fields and a button labeled "answer".
- TERMINAL:** Shows the command `flask --debug run` being executed, along with a warning message about using a development server in production.
- STATUS BAR:** Shows the path "Codespaces: friendly space robot", the number of files (0), and the layout setting "Layout: U.S.".

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 for a Codespace named "PACE ROBOT".

- EXPLORER:** Shows the project structure with a file named "calc.html" selected.
- CODEVIEW:** Displays the content of "calc.html". The title "Brent's Calc" has been added to line 34.
- BROWSER:** A preview window titled "Simple Br" shows the word "Calc" replaced by "Brent's Calc". Below it is a form with input fields and a button labeled "answer".
- TERMINAL:** Shows the command `flask --debug run` being executed, along with a warning message about using a development server in production.
- STATUS BAR:** Shows the path "Codespaces: PACE ROBOT", the number of files (0), and the layout setting "Layout: U.S.".

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 the Codespace interface with two panes. On the left is the 'calc.html' file editor, displaying the following code:

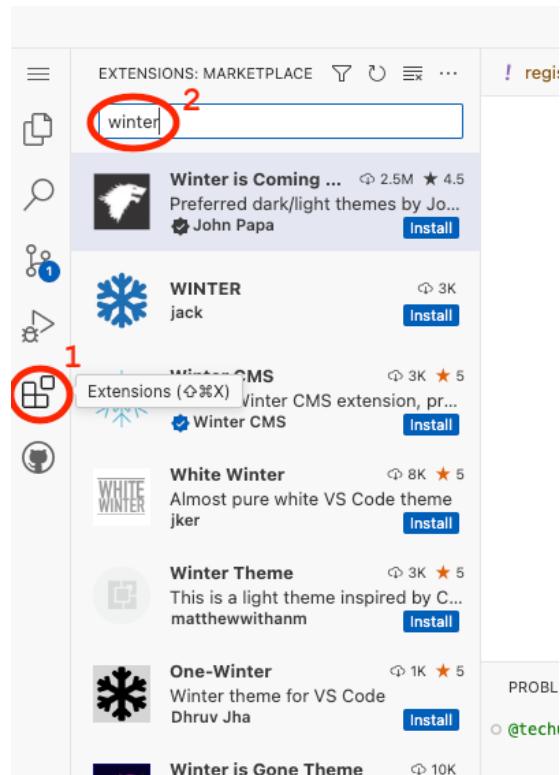
```

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>

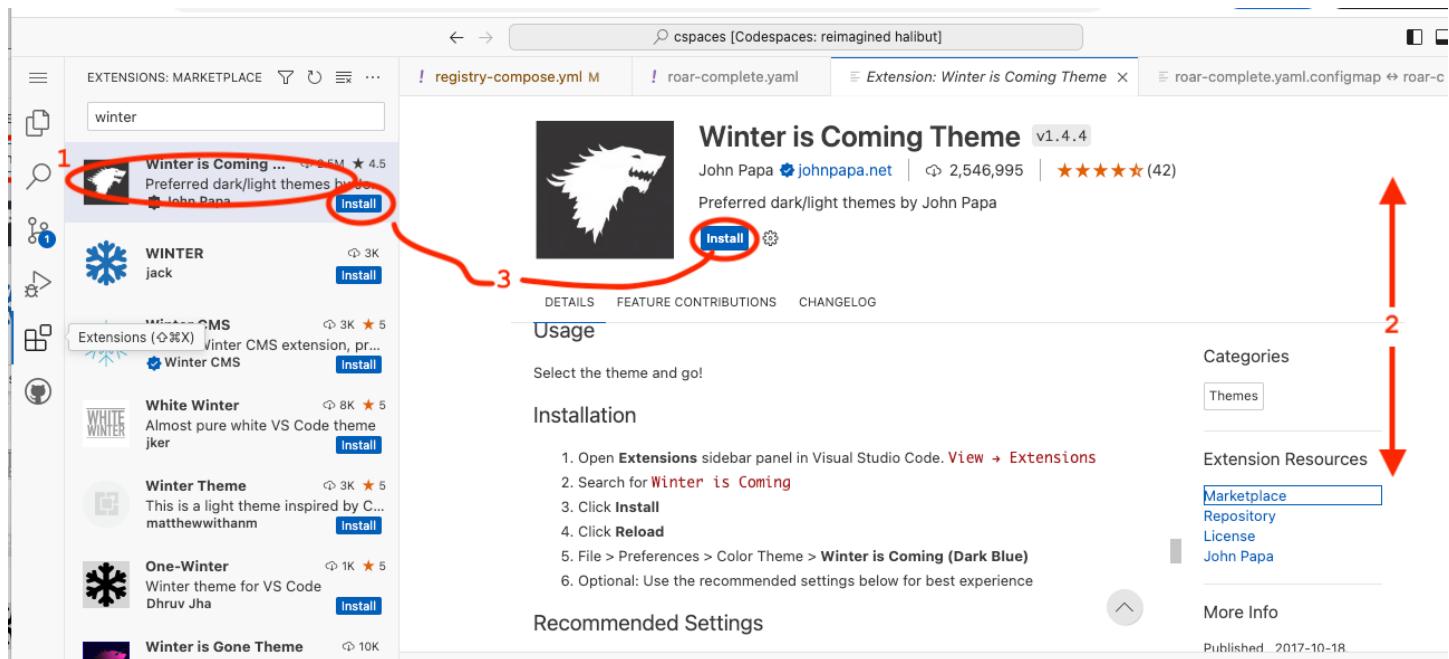
```

On the right is the 'Simple Browser' pane, showing the result of the code execution. The title of the browser window is 'Simple Browser'. The URL in the address bar is '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'.

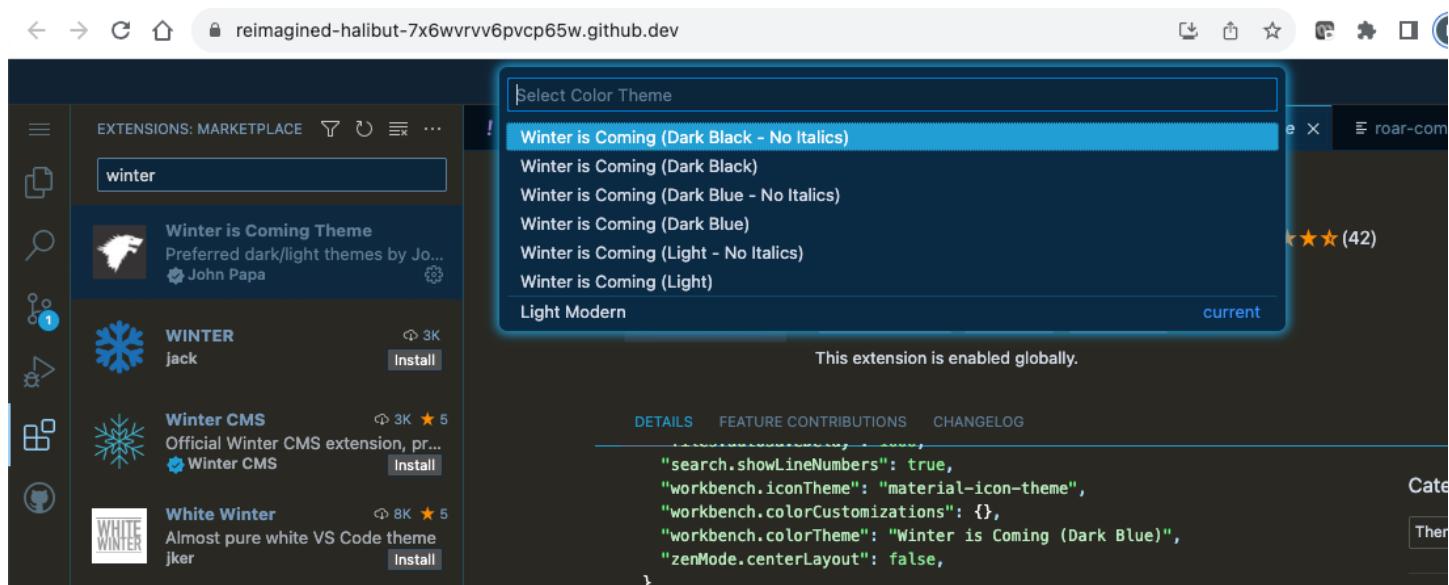
6. Next, let's see how to modify our codespace environment. We'll install an extension for the color theme in the codespace. For practice, we'll use the "Winter is Coming" theme. Click on the *Extensions icon* (#1 in figure below), then in the *search bar* type in "winter" to quickly find the extension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



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
```

END OF LAB

Demo: Copilot

That's all - THANKS!