

GitHub Fundamentals BootCamp Labs

Learn the complete GitHub – from code management to Copilot

Revision 1.7 – 01/01/25

Tech Skills Transformations LLC / Brent Laster

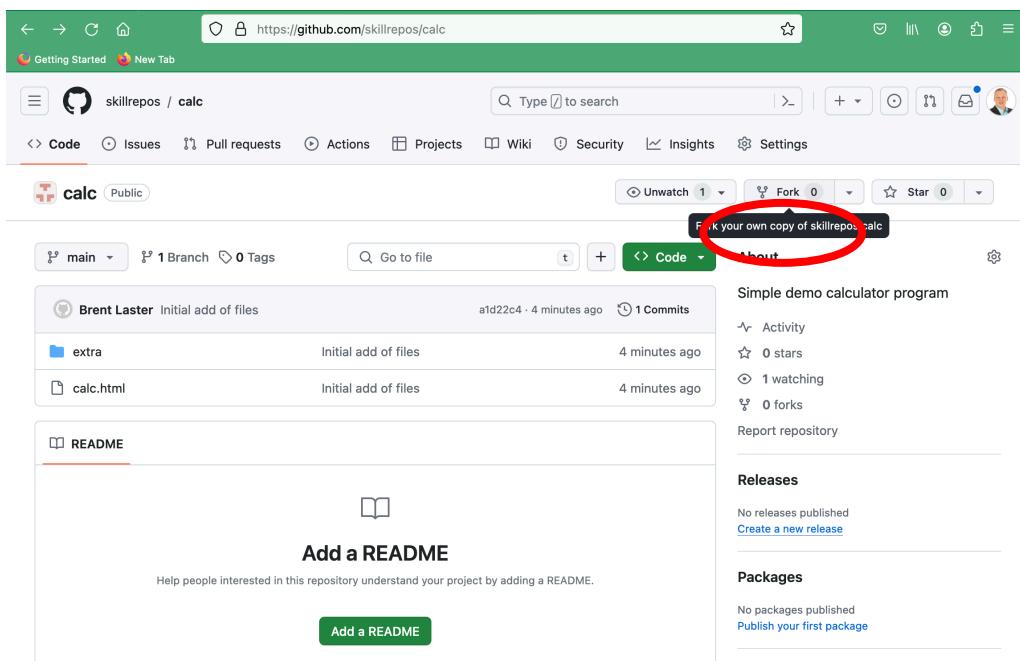
Setup and prerequisites

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. (Note: If you cannot use the Fork functionality, see alternatives in **Appendix 2** at the end of this document.)



The screenshot shows the GitHub interface for creating a new fork of a repository named 'calc'. The 'Code' tab is selected. The 'Owner' dropdown is set to 'brentlaster' and the 'Repository name' field contains 'calc'. A green message says 'calc is available.' Below the fields, it states that forks are named the same as upstream. A yellow callout bubble with the text 'uncheck' points to the 'Copy the main branch only' checkbox, which is circled in red. The 'Description' field contains 'Simple demo calculator program'. A note below the checkbox says 'Contribute back to skillrepos/calc by adding your own branch. [Learn more.](#)' A message indicates the fork is in a personal account. At the bottom right is a large green 'Create fork' button, also circled in red.

- 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 repository page for 'brentlaster/calc'. The 'Code' tab is selected. The repository is public and was forked from 'skillrepos/calc'. The main branch is 'main'. A modal window is open in the center, showing a list of files: 'extra' (initial add of files), 'calc.html' (initial add of files), and 'README'. At the top of the modal, there are tabs for 'Local' (circled in red with number 3) and 'Codespaces' (circled in red with number 2). Below these are 'HTTPS', 'SSH', and 'GitHub CLI' buttons. The 'HTTPS' button is highlighted and has a red circle around it with number 4. To its right is a 'Copy url to clipboard' button. Below the buttons is a URL: 'https://github.com/brentlaster/calc.git'. A note below the URL says 'Use Git or checkout with SVN using the web URL' (circled in red with number 5). At the bottom of the modal are 'Open with GitHub Desktop' and 'Download ZIP' buttons.

- Open a terminal on your system and clone down the repository from GitHub. You can use the following command – just paste (or type) the URL you copied from the step above and hit Enter/Return. Then change into the subdirectory that was created from the clone.

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

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

```
$ git config user.name "your name"
$ git config user.email <same email as you're using on GitHub>
```

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

```
$ git remote -v
```

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

Edit calc.html and change

<title>Calc</title>

to

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

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

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

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

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

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

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

(Shortcut to token page is <https://github.com/settings/tokens/new>)

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

use.

The screenshot shows the GitHub settings interface for creating a new personal access token. The left sidebar has links for GitHub Apps, OAuth Apps, Personal access tokens (selected), Fine-grained tokens (Beta), and Tokens (classic). The main area is titled "New personal access token (classic)". It explains that personal access tokens function like OAuth tokens and can be used for Git over HTTPS or API authentication. A note section says the token is for a "GitHub Fundamentals class". The "Expiration" field is set to 30 days, with a note that it will expire on Mon, Jan 22 2024. The "Select scopes" section lists various permissions with checkboxes. The selected scopes are: repo (checked), workflow (checked), write:packages (checked), and delete:packages (checked). Other available scopes include repo:status, repo_deployment, public_repo, repo:invite, security_events, and read:packages.

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

A modal dialog box is shown with the following content:

- read:ssh_signing_key**
- Read public user SSH signing keys
- Generate token** (green button)
- Cancel**

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

The screenshot shows the GitHub 'Personal access tokens (classic)' page. On the left, there's a sidebar with options like 'GitHub Apps', 'OAuth Apps', 'Personal access tokens' (which is selected and has a 'Beta' badge), 'Fine-grained tokens', and 'Tokens (classic)'. The main area displays a message: 'Make sure to copy your personal access token now. You won't be able to see it again.' Below this, a token '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' is listed with a green checkmark and a 'Copied!' message bubble.

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 "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':
```

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.

1. After the push is complete, you can switch back to the GitHub repo in the browser, change the branch to **dev** and click on the calc.html file to see the change. (If you don't see **dev** listed in the branch

dropdown list, click on the **3 Branches** button next to the dropdown and you should be able to see it there. Alternatively, you can go to github.com/<github userid>/calc/tree/dev in the browser.)

The screenshot shows a GitHub repository page for 'calc'. At the top, there are navigation buttons: 'dev' (selected), '3 Branches' (highlighted in green), and '0 Tags'. On the right, there are buttons for 'Go to file', 'Add file', and 'Code'. A modal window titled 'Switch branches/tags' is open on the left. It contains a search bar with 'Find or create a branch...', a tab for 'Branches' (selected), and a list of branches: 'main' (marked as 'default'), 'cspace', and 'dev'. Below this is a link 'View all branches'. The main repository area shows a commit for 'Updating title' by 'Brent Laster' from 8 hours ago. A 'Sync fork' button is also visible.

The screenshot shows the same GitHub repository page for 'calc'. The 'Branches' tab is selected in the top navigation. A message at the top says 'This branch is 1 commit ahead of skillrepos/calc:main'. Below this, a commit for 'Updating title' is listed. Underneath the commit, there are links for 'calc.html' and 'README'. The 'calc.html' link is highlighted with a blue border. The 'README' link is also present. The commit details show it was made by 'Brent Laster' 8 hours ago.

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.

The screenshot shows the 'Blame' view for the 'calc.html' file. The 'Blame' button in the top navigation bar is circled in red. The blame output shows the history of changes to the file. It includes a header with 'Code' and 'Blame' buttons, the number of lines (46 loc), and the file size (1.29 KB). Below this, a color-coded timeline shows changes from 'Older' to 'Newer'. The blame output lists commits from 'last week' and '22 minutes ago'. The commit from '22 minutes ago' is for 'Updating title'. The blame output shows the HTML code for the title and a script block containing JavaScript for arithmetic operations.

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

The screenshot shows a GitHub commit history for the file `calc / calc.html` on the `dev` branch. At the top, there are navigation links: Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below that is a search bar with the placeholder "Go to file" and a dropdown menu showing "dev". The commit list starts with a recent update by Brent Laster:

- Brent Laster** Updating title · d9e79db · 24 minutes ago · History

Below the commit, there are tabs for "Code" and "Blame", and a summary: 59 lines (46 loc) · 1.29 KB. There are also "Raw", "Download", "Edit", and "More" buttons. At the bottom, there are "Older" and "Newer" navigation arrows.

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.

The screenshot shows the commit history for `calc / calc.html` on the `dev` branch. The commits listed are:

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

Updating title

Brent Laster committed 28 minutes ago

1 parent a1d22c4 commit d9e79db

Showing 1 changed file with 2 additions and 2 deletions.

Whitespace Ignore whitespace Split Unified

```

v 4 calc.html
...
@@ -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 @@ <h2>Calc</h2>
 56 56
 57 57
 58 58 </body>
 59 - </html>
 59 + </html>

```

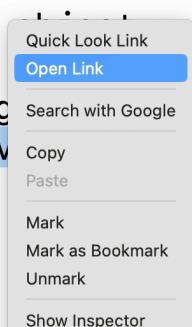
5. Let's now merge our change from the dev branch to main via a pull request. **Switch back to the terminal where you did the commit and push.**

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

- 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

Reviewers

Assignees

No reviews

No one—assign yourself

Write Preview

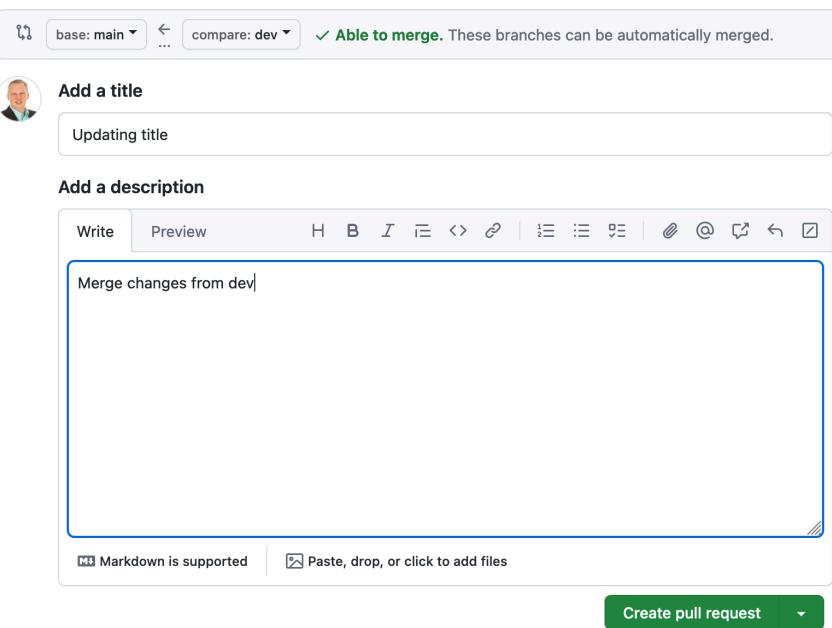
- After making that change, the gray bar showing the base and compare should look like the screenshot below.

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.

- 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 a single commit "Updating title" with hash `d9e79db`. To the right, there are labels (None), projects (None), milestones (None), and notifications (None).

In the main area, a message says "Add more commits by pushing to the `dev` branch on [brentlaster/calc](#)". Below this, a modal window is open:

- Merge pull request #1 from brentlaster/dev**
- Updating title**
- This commit will be authored by `bclaster@nclasters.org`
- Confirm merge** (green button) and **Cancel** (white button)

Below the modal, there's a link to "Add a comment".

After confirming the merge, the status changes to "Merged". The title now says "Updating title #1" and the message is "brentlaster merged 1 commit into `main` from `dev` now". The commit history shows "Merge changes from dev" with a smiley face emoji. The hash is still `d9e79db`. A "Revert" button is available next to the merge commit.

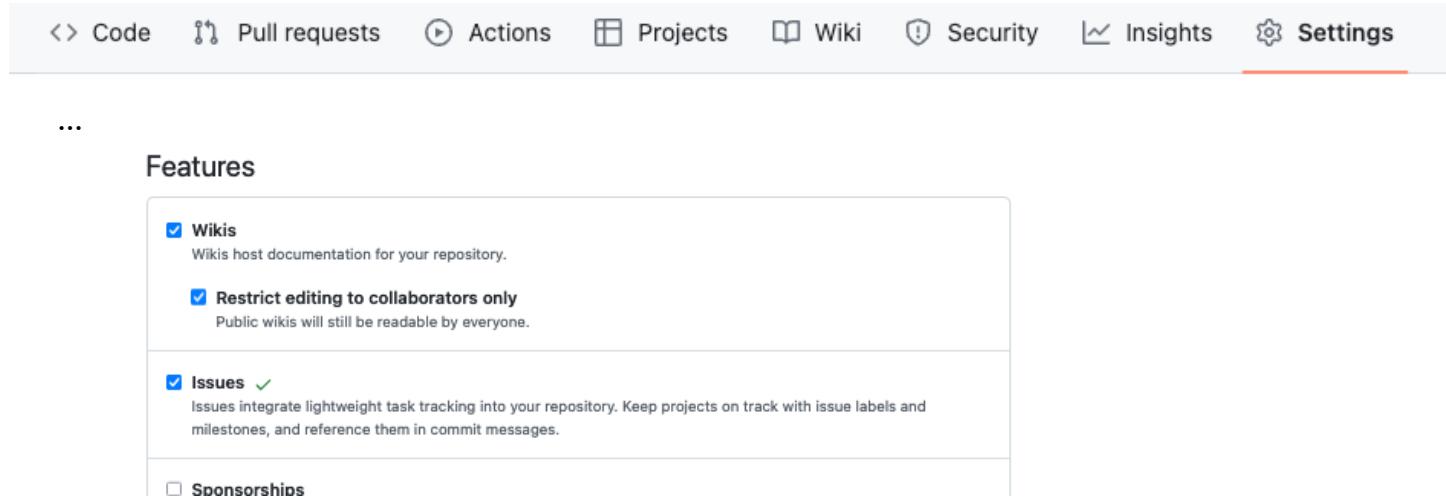
At the bottom, a purple box contains the message "Pull request successfully merged and closed" and "You're all set—the `dev` branch can be safely deleted." A "Delete branch" button is also present.

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.

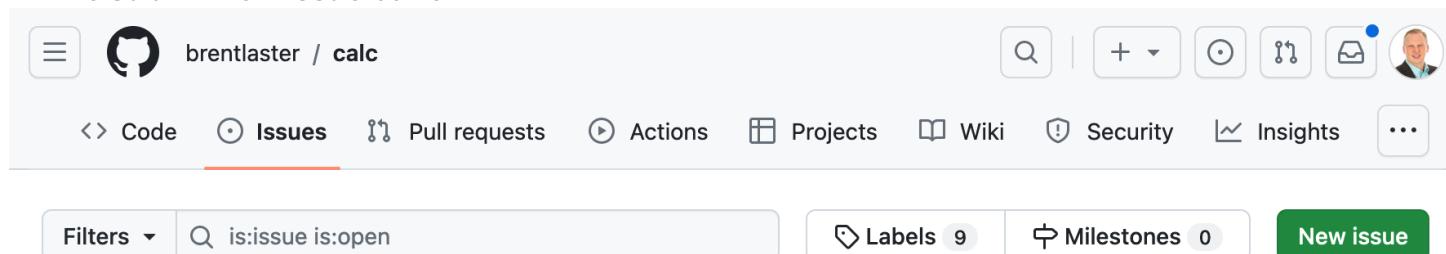
1. We'd like to have a *README* file in our project to make it more standard. So, let's create an issue to document that. First, ensure that the repository has the *Issues* feature turned on. On the main repo page, go to the repository's **Settings** tab, and then scroll down until you see the **Features**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



The screenshot shows the 'Features' section of the GitHub repository settings. It includes options for Wikis, Restrict editing to collaborators only, Issues (which is checked and highlighted in green), and Sponsorships.

- 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**: This option is not checked.

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 repository issues page. The 'Issues' tab is selected. At the top right, there is a green 'New issue' button. Below the tabs, there are filters for 'Labels' (9) and 'Milestones' (0).

Add a title

Needs README

Add a description

Please add README file :book:

Write Preview H B I E ↵ | ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ...

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

Submit new issue

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

- 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

Add a comment

Write Preview H B I E ↵ | ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ...

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 portion of a Jira interface. At the top, there are navigation links: Wiki, Security, Insights, and a three-dot menu. Below these are two buttons: 'Edit' and 'New issue'. A horizontal line separates the header from the main content area. In the main area, there is a light blue sidebar with a three-dot icon. To its right, the word 'Assignees' is centered above a gear icon. Below this, the text 'No one—[assign yourself](#)' is displayed.

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

The screenshot shows a 'Apply labels to this issue' dialog box. On the left, there's a sidebar with a three-dot icon, 'Assignees' (listing 'brentlaster'), and 'Labels'. The main area has a 'Filter labels' input field containing 'Something isn't working'. Below it, a list shows a checked item '✓ documentation' with the description 'Improvements or additions to documentation' and an unchecked item 'duplicate' with the description 'This issue or pull request already exists'.

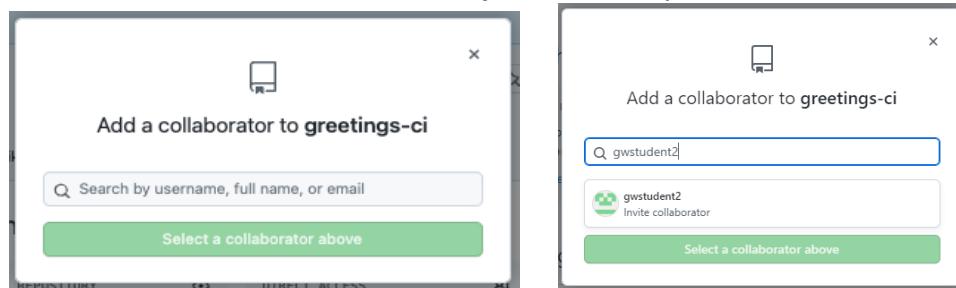
6. After this, if you click on the **Issues** tab at the top, and look at your issue, it should look like the following.

The screenshot shows a GitHub Issues page for the repository 'brentlaster / calc'. The top navigation bar includes links for Code, Issues (with 1 open issue), Pull requests, Actions, Projects, Wiki, Security, and Insights. The Issues tab is active. Below the navigation is a search bar and filter options: 'Filters' (set to 'is:issue is:open'), 'Labels 9' (including 'documentation'), 'Milestones 0', and 'New issue'. The main list shows one open issue: '#2 opened 15 hours ago by brentlaster' with the title 'Needs README documentation' and the label 'documentation' highlighted.

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 'Who has access' section of the GitHub repository settings. On the left sidebar, 'Collaborators' is selected under the 'Access' category. The main area displays that it is a 'PUBLIC REPOSITORY' with 'DIRECT ACCESS'. It shows 0 collaborators have access. A 'Manage' button is available. Below this is the 'Manage access' section, which displays a message: 'You haven't invited any collaborator' and features 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](#)

Watch 0

@brentlaster has invited you to collaborate on this repository

[View invitation](#)

[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](#)

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.

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

The screenshot shows the GitHub interface for the 'calc' repository. The top navigation bar includes tabs for Code, Issues (1), Pull requests, Actions, Projects, Wiki, and Settings. Below the header, it shows the repository is public and forked from 'skillrepos/calc'. A message indicates the branch is 2 commits ahead of 'skillrepos/calc:main'. The commit history shows two recent changes: 'Merge pull request #1 from br...' by brentlaster (dc98b08 · yesterday) and 'Initial add of files' by brentlaster (last week). The 'extra' folder was added, and 'calc.html' was updated. The 'README' section is open, showing a placeholder text area with a 'Add a README' button at the bottom, which is circled in red.

2. This will bring up the editor in GitHub. Enter the text below in the new file text input area for README.md. Fill in your github userid in both places instead of `github-userid`. (Notes: Do this on a single line. Also, there is no space between the “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.)

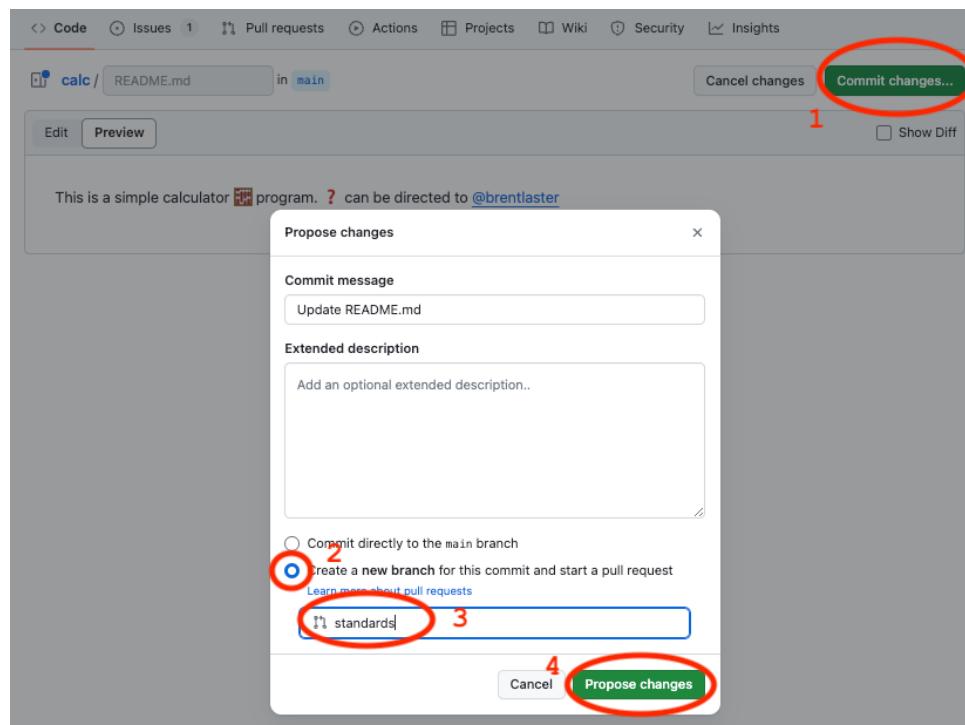
This is a simple calculator :abacus: program. :question: can be directed to [@github-userid](<https://github.com/github-userid>)

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

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

This is a simple calculator :abacus: program. ? can be directed to [@brentlaster](#)

4. Now let's commit these changes to a new branch and open a pull request to merge them. click on the green **Commit changes...** button in the upper right corner. In the dialog, enter a 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 branches. At the top, there's a header with links for Code, Issues (1), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, a section titled "Comparing changes" asks to choose two branches to see what's changed or to start a new pull request. It notes that the branches can be automatically merged. A green box highlights the "Able to merge" status. To the right, a red circle highlights the "Create pull request" button. Below the main area, there's a summary: 1 commit, 1 file changed, and 1 contributor. The commit details show a user named "brentlaster" committing "Create README.md". A diff view shows one addition to README.md: "This is a simple calculator :abacus: program. :question: can be directed to [brentlaster] (<https://github.com/brentlaster>)".

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

The screenshot shows the "Open a pull request" form. It includes fields for "Add a title" (with a placeholder "Create README.md") and "Add a description" (with a rich text editor and a note that Markdown is supported). On the right, there's a sidebar for "Reviewers" which says "Request up to 15 reviewers" and has a search bar. A user named "gwstudent2" is selected, indicated by a blue background and a checkmark. Other sections in the sidebar include "Labels" (2), "Projects", "Milestone", "Development" (with a note about closing keywords), and "Helpful resources".

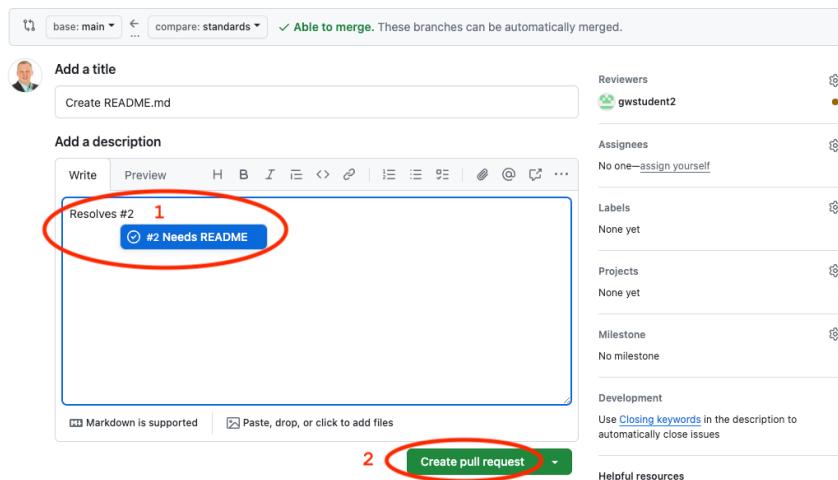
7. Also, we can add in a description that will automatically close the associated issue when we resolve this pull request. Click in the “Add your description here...” field and enter **Resolves #2**

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

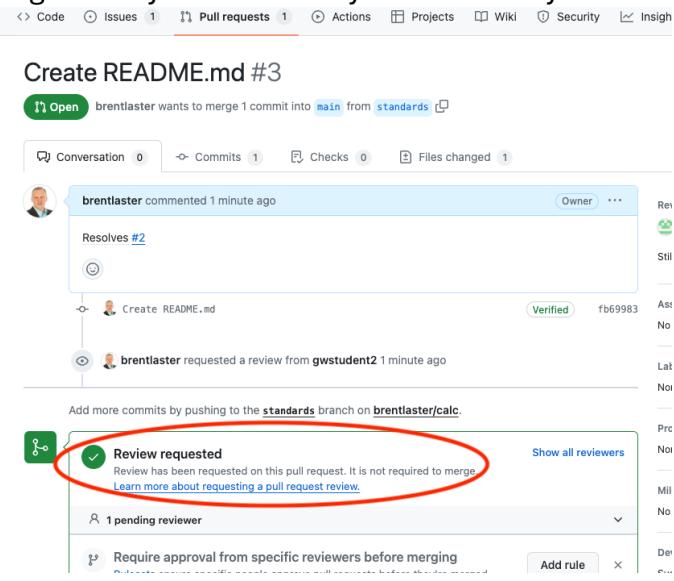
Then click on the “Create pull request” button.

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](#).



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



END OF LAB

Lab 5: Completing a pull request with reviewers

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

- 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 inbox. At the top right, there is a red circle around the envelope icon with the number '1' below it. The inbox lists notifications under various filters like 'Assigned', 'Participating', 'Mentioned', 'Team mentioned', and 'Review requested'. A specific notification for 'brentlaster/calc #4 Create README.md' is highlighted with a red circle and the number '2' below it. This notification is marked as '+1 review requested'. Below the inbox, there is a 'ProTip!' message about creating custom filters.

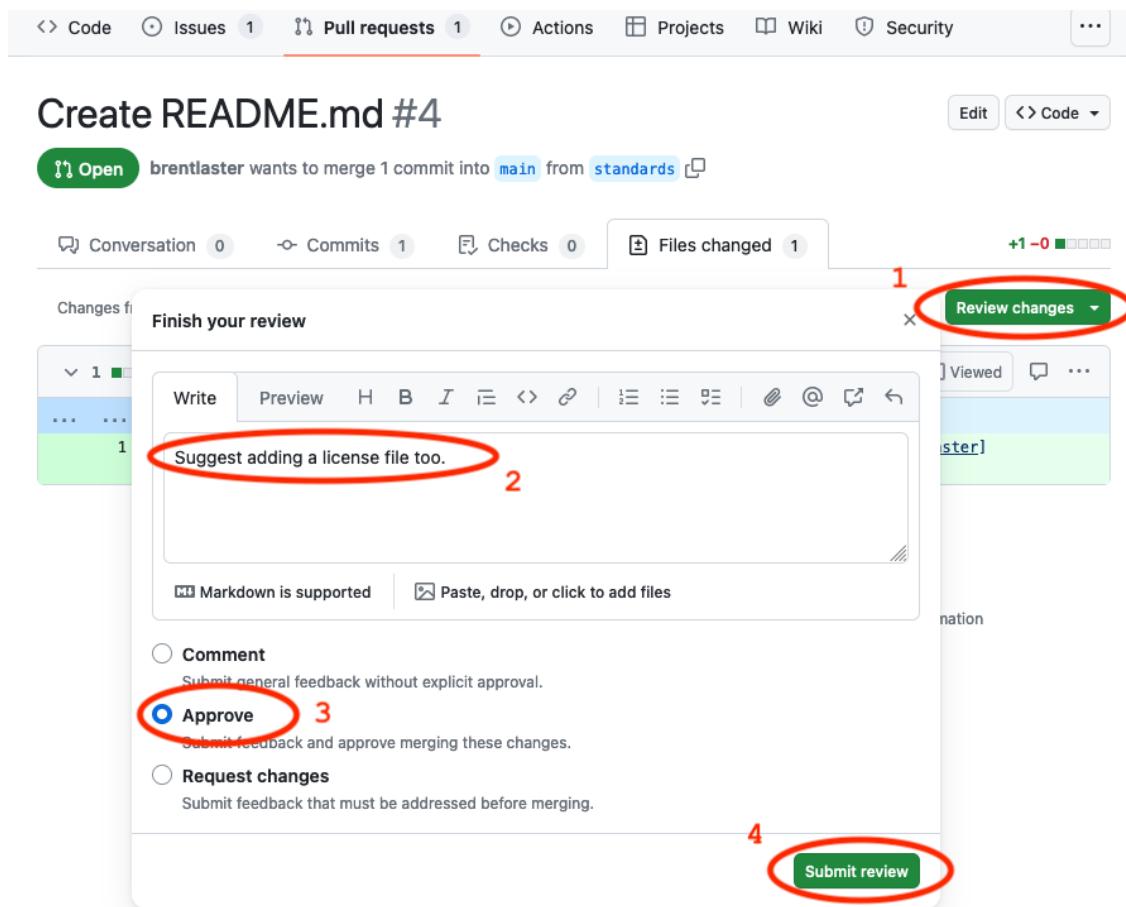
- OR -

The screenshot shows a browser window with the URL <https://github.com/pulls/review-requested> circled in red. The page title is 'Pull Requests'. The main content area shows a single open pull request for 'brentlaster/calc Create README.md'. A red circle highlights this pull request line. A 'ProTip!' message at the bottom suggests using `no:assignee` to see unassigned items.

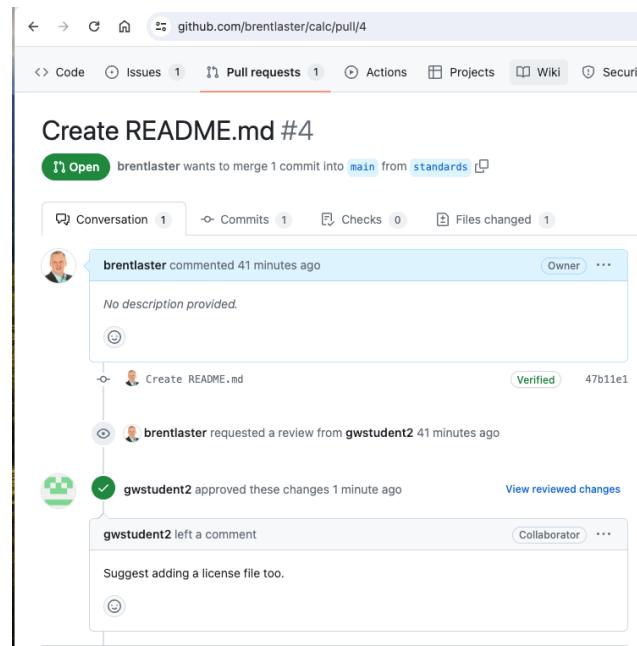
2. This will open up the pull request. There is a button at the top to “Add your review”. Click on that.

The screenshot shows a GitHub pull request page for 'Create README.md #4'. At the top, a yellow banner says 'brentlaster requested your review on this pull request.' A red circle highlights the green 'Add your review' button. Below the banner, the pull request details show it is 'Open' and ready to merge into 'main' from 'standards'. The conversation tab shows a comment from 'brentlaster' and the reviewers section shows 'gwstudent2'.

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”.

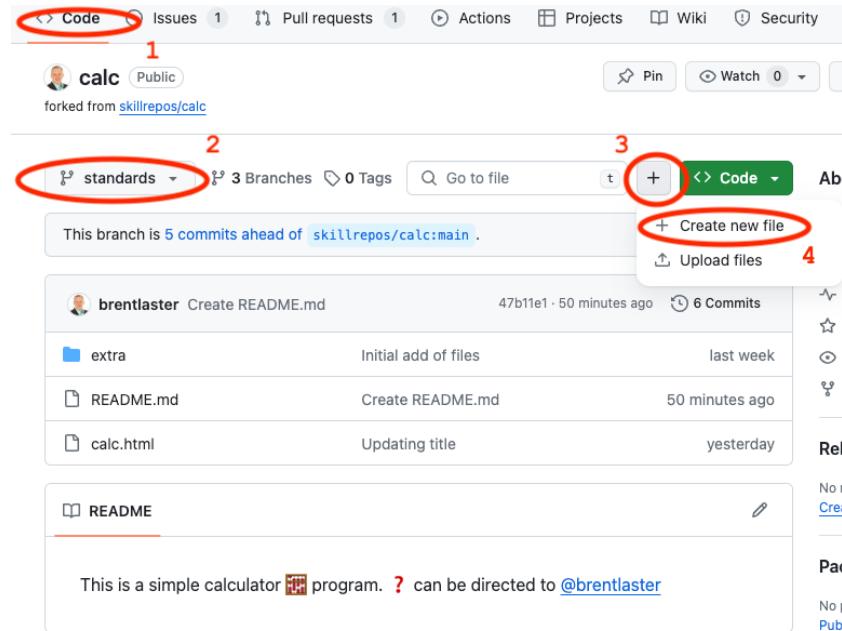


4. Go to the session with your original GitHub userid or log out of the other one and log back in if you need to. Go to the **Pull requests** menu at the top, find the pull request and click on the commit message. Then you should see a screen like below.

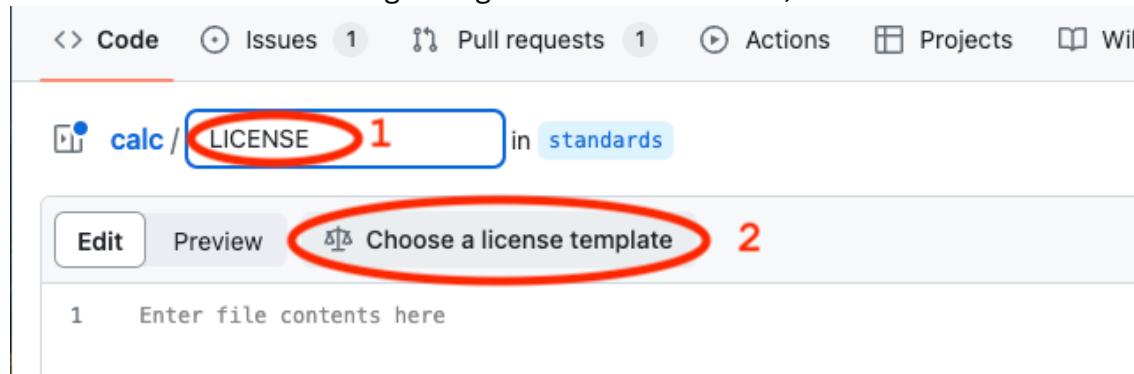


5. Since there was a suggestion to add a license file, that sounds like a good idea, so let's do that. Click on the **Code** tab at the top, then select the **standards** branch (or whatever name you gave

the new branch) from the branch dropdown, then select the “+” sign (or "Add file" option) and the option to + **Create new file**.



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.

Add a license to your project

Apache License 2.0	A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.
GNU General Public License v3.0	
MIT License 1	Permissions ✓ Commercial use ✓ Modification ✓ Distribution ✓ Private use
BSD 2-Clause "Simplified" License	Limitations ✗ Liability ✗ Warranty
BSD 3-Clause "New" or "Revised" License	Conditions ⓘ License and copyright notice
Boost Software License 1.0	This is not legal advice. Learn more about repository licenses.
	MIT License

To adopt MIT License, enter your details. You'll have a chance to review before committing a LICENSE file to a new branch or the root of your project.

Year ⓘ
2024

Full name ⓘ
Brent Laster

Review and submit 2

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.

Your license is ready. Please review it below and either commit it to the main branch or to a new branch.

calc / LICENSE in standards

Commit changes

Commit message
Create LICENSE

Extended description
Add an optional extended description..

Commit directly to the standards branch 1
 Create a new branch for this commit and start a pull request

Cancel 2 Commit changes 2

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.

The screenshot shows the GitHub interface for a pull request titled "Create README.md #4". The top navigation bar includes "Code", "Issues 1", "Pull requests 1", "Actions", "Projects", "Wiki", "Security", "Insights", and "Settings". A red circle highlights the "Pull requests 1" button. Below the navigation, there's an "Open" button and a message: "brentlaster wants to merge 2 commits into main from standards". The "Conversation" tab is selected, indicated by a red circle around the "1" icon. Other tabs shown are "Commits 2", "Checks 3", and "Files changed 2". A search bar for "Filter changed files" is present. The main area displays two commits. Commit 1 is for "LICENSE" and commit 2 is for "README.md". The "README.md" commit contains the text: "+ This is a simple calculator :abacus: program. :question: can be directed to (@brentlaster) (<https://github.com/brentlaster>)".

- 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 the GitHub Issues page for a repository named "brentlaster / calc". The "Issues" tab is selected. A single issue titled "Needs README #2" is listed, with a status of "Closed". The issue was opened by "brentlaster" 19 hours ago, with a note: "brentlaster opened this issue 19 hours ago · 0 comments · Fixed by #4 or #5 · May be fixed by #6". The issue has the following activity log:

- 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 [Create README.md #4](#) 22 minutes ago that will close this issue
- brentlaster mentioned this issue 15 minutes ago: "Update README.md #5" (status: Merged)
- brentlaster closed this as completed in #5 14 minutes ago

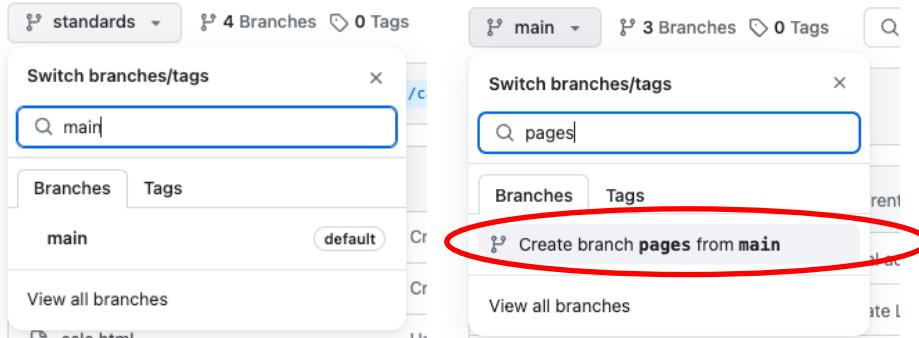
On the right side of the issue card, there are sections for "Assignees" (brentlaster), "Labels" (documentation), "Projects" (None yet), "Milestone" (No milestone), "Development" (Successfully merging a pull request may close this issue), and "Notifications" (Customize, Unsubscribe). It also states: "You're receiving notifications because you modified this issue".

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. 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.



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.

Create a new repository

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

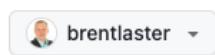
Required fields are marked with an asterisk ().*

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *



Repository name *

brentlaster.github.io

brentlaster.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 repository

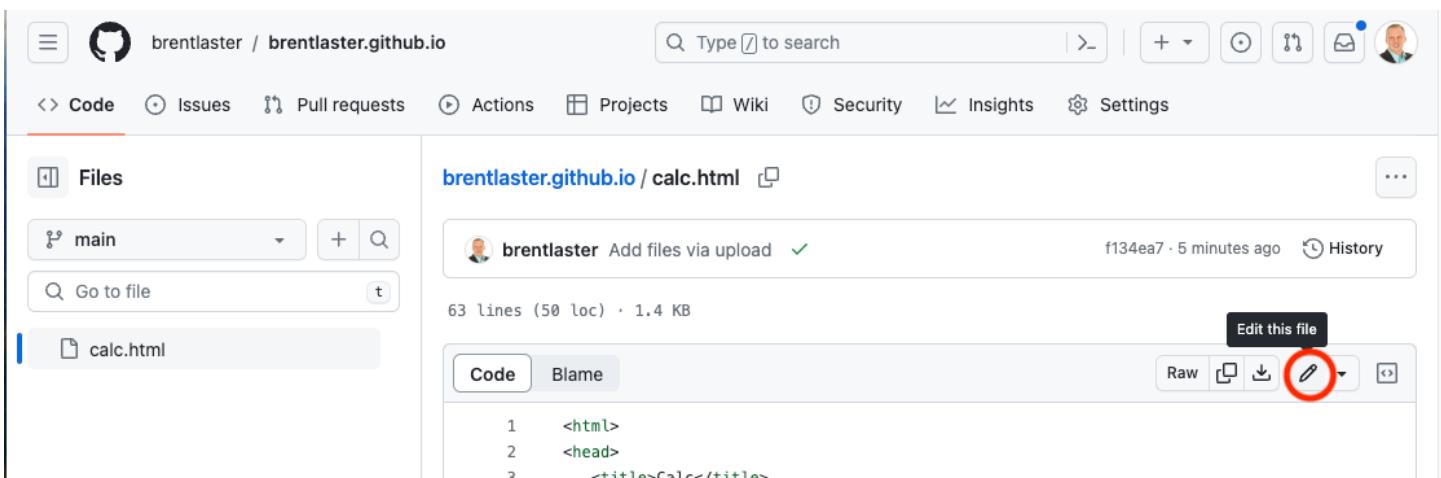
3. So that we have content to publish, we'll grab the code from the calc.html file in your local repository from Lab 1 and add it here. On the screen with the **Quick setup – if you've done this kind of thing before** instructions, click on the link in the big blue bar for **uploading an existing file**.

The screenshot shows the GitHub repository setup page for `brentlaster/brentlaster.github.io`. It includes sections for 'Start coding with Codespaces', 'Add collaborators to this repository', and a 'Quick setup' section. The 'Quick setup' section contains instructions to 'Get started by creating a new file or uploading an existing file.' A red oval highlights the 'uploading an existing file' link.

- 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 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.

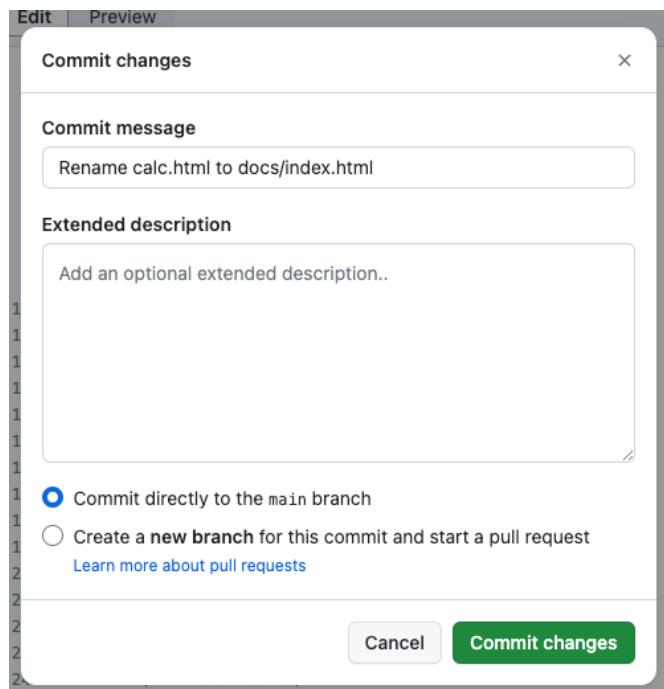
- You should now be back in the new repo’s **Code** tab. Let’s change the name and location of this file to make it more consistent for GitHub Pages. Click on the `calc.html` file and then click on the “pencil” icon to edit it.



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”. Note you are adding the directory `docs` to the path. The screenshots show this in 2 parts for clarity.

The image contains two stacked screenshots of the GitHub interface. Both screenshots show a file named 'calc.html' in the 'main' branch. The top screenshot shows the file being renamed. A red oval highlights the URL bar which now reads 'brentlaster.github.io / docs'. The bottom screenshot shows the file has been successfully renamed to 'index.html', and the URL bar now reads 'brentlaster.github.io / docs / index.html'. Both screenshots include a red oval highlighting the URL bar.

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



7. 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.

8. 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

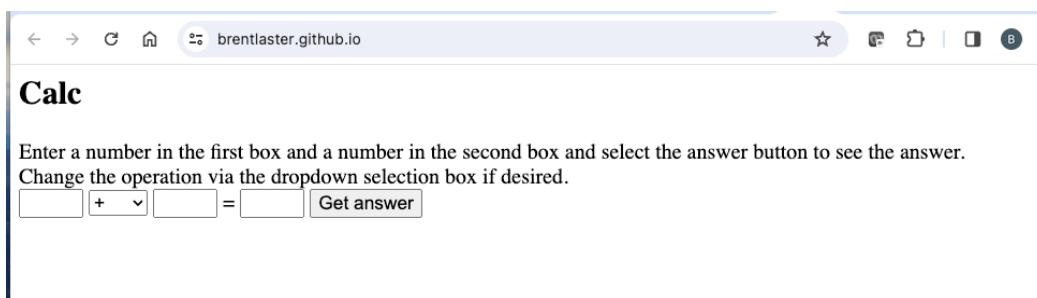
[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://gwstudent.github.io/>

Last deployed by  gwstudent 4 minutes ago

[Visit site](#)

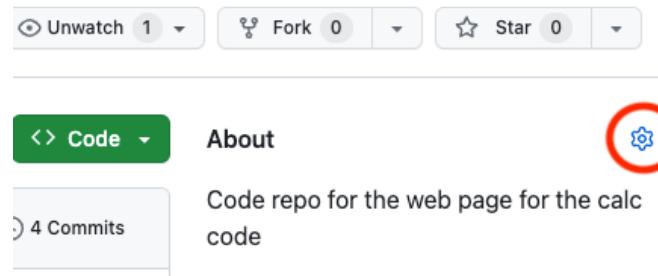
...



The screenshot shows a web browser window with the URL <https://brentlaster.github.io/>. The page title is "Calc". Below the title, there is a text instruction: "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 instruction is a form with two input fields, a dropdown menu for operations (+, -, ×, ÷), another input field for the result, and a "Get answer" button.

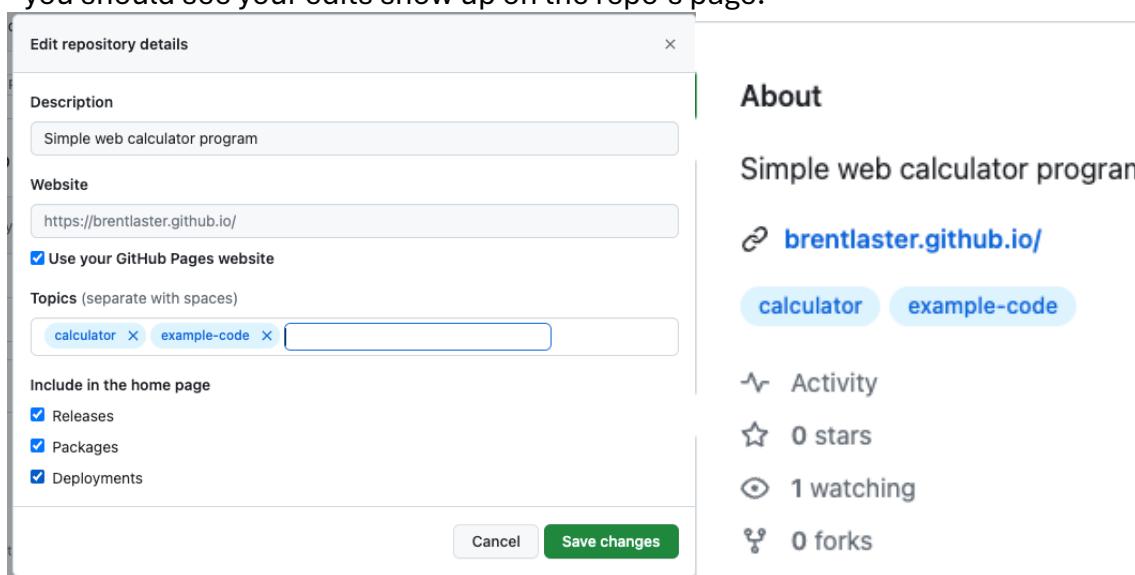
OPTIONAL:

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**.



The screenshot shows the "About" section of a GitHub repository. It includes a "Code" button, a "About" button (circled in red), and a "4 Commits" button. The "About" section contains the text: "Code repo for the web page for the calc code".

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.



The screenshot shows the "Edit repository details" dialog and the resulting repository page. The dialog contains fields for Description (Simple web calculator program), Website (<https://brentlaster.github.io/>), and Topics (calculator, example-code). The resulting repository page shows the updated information: "About" (Simple web calculator program), "Website" (<https://brentlaster.github.io/>), "Topics" (calculator, example-code), "Activity" (0 stars, 1 watching, 0 forks), and "Releases", "Packages", and "Deployments" sections.

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.

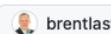
The screenshot shows two GitHub pages side-by-side. The top page is for the repository `greetings-ci` owned by `skillrepos`. It displays the main branch, two branches in total, and zero tags. The bottom page is for the forked repository `greetings-ci/fork` owned by `brentlaster`. Both pages include a search bar, code editor, and a list of recent commits. A yellow callout box labeled "uncheck" points to the `Copy the main branch only` checkbox in the fork creation form.

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 the GitHub repository page for `gwstudent/greetings-ci`. The top navigation bar includes links for Search or jump to..., Pull requests, Issues, Marketplace, Explore, Pin, Watch, Fork, Star, and a gear icon. Below the navigation bar, the repository name is displayed along with its status as Public and a note that it is forked from `skillrepos/greetings-ci`. The main navigation tabs are Code (which is active), Pull requests, Actions (which is highlighted with a red circle), Projects, Wiki, Security, Insights, and Settings. Under the Code tab, there are buttons for Go to file, Add file, and Code. The repository details section shows the branch is up to date with the upstream. A list of recent commits is shown, all made by Brent Laster. The commits include adding extra files like `extra`, `gradle/wrapper`, and `src/main/java`, as well as `build.gradle` and `gradlew`. The commit times range from 7 minutes ago to 14 minutes ago. To the right of the commits, there is an About section with a brief description of the repo as a simple starter for CI/CD with GitHub Actions, and sections for Releases, Packages, and 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**.

The screenshot shows the 'Browse all categories' page on GitHub. It lists several categories of actions: Automation, Continuous integration (which is highlighted with a red circle), Deployment, and Security. Each category has a description and two buttons: 'Configure' and 'Automation'. The 'Continuous integration' category is described as containing actions for GitHub Actions that check for stale issues and pull requests, trigger manual workflows, and label pull requests based on file changes.

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](#)

Categories

Automation

Continuous Integration

Deployment

Security

Q Gradle

Found 52 workflows



Android CI

By GitHub Actions

Build an Android project with Gradle.

Configure

Java



Java with Ant

By GitHub Actions

Build and test a Java project with Apache Ant.

Configure

Java



Clojure

By GitHub Actions

Build and test a Clojure project with Leiningen.

Configure

Clojure

Publish Java Package

Java with Gradle

Publish Java Package

- 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](#)

Categories

Automation

Continuous Integration

Deployment

Security

Q Gradle

Found 3 workflows



Android CI

By GitHub Actions

Build an Android project with Gradle.

Configure

Java



Publish Java Package with Gradle

By GitHub Actions

Build a Java Package using Gradle and publish to GitHub Packages.

Configure

Java



Java with Gradle

By GitHub Actions

Build and test a Java project using a Gradle wrapper script.

Configure

Java

- 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.)

<> Code Pull requests Actions Projects Wiki Security

greetings-ci / .github / workflows / gradle.yml

in main

<> Edit new file Preview

TO

<> Code Pull requests Actions Projects Wiki Security

greetings-ci / .github / workflows / pipeline.yml

in main

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.

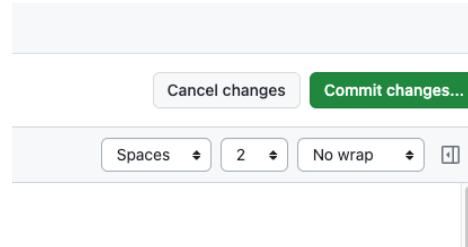
```

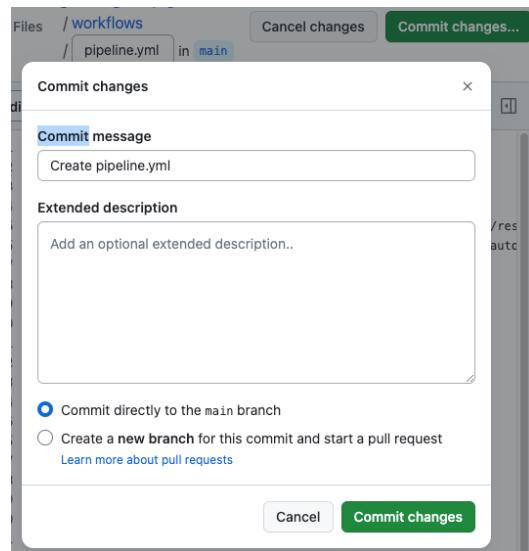
greetings-ci/.github/workflows/pipeline.yml in main
Edit Preview

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
52   runs-on: ubuntu-latest
53   permissions:
54     contents: write
55
56   steps:
57     - uses: actions/checkout@v4
58     - name: Set up JDK 17
59       uses: actions/setup-java@v4
59       with:
60         java-version: '17'
61         distribution: 'temurin'
62
63       # Generates and submits a dependency graph, enabling Dependabot Alerts for all project dependencies.
64       # See: https://github.com/gradle/actions/blob/main/dependency-submission/README.md
65     - name: Generate and submit dependency graph
66       uses: gradle/actions/dependency-submission@417ae3ccd767c252f5661f1ace9f835f9654f2b5 # v3.1.0
67
68

```

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





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

Workflow	Status	Time	Actor
Create pipeline.yml	In progress	now	brentlaster

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.

Workflow	Status	Time	Actor
Create pipeline.yml	Success	1 minute ago	brentlaster

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 a GitHub Actions pipeline named "Create pipeline.yml #1". The "build" job has succeeded. The log output for the "Run actions/checkout@v3" step is shown, indicating it synchronized the repository and got the Git version info. A red circle highlights the "build" job in the sidebar and the "Run actions/checkout@v3" step in the log.

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.)

The screenshot shows the GitHub repository "greetings-ci". It has 2 branches and 0 tags. The "package" branch is selected. The repository is public and forked from "skillrepos/greetings-ci". On the right, the "publish-package.yml" workflow file is displayed, which defines a workflow for publishing packages to GitHub Packages.

```

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/greetings-ci. After you make those changes, go ahead and create the pull request (by clicking through the **Create pull request** buttons on the screens).

The screenshot shows the GitHub Pull Requests page. At the top, there are navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation is a search bar with filters for 'is:pr is:open' and buttons for Labels (9), Milestones (0), and New pull request. A summary table shows 0 Open and 23 Closed pull requests. The main area displays a pull request card with the following details:

- base repository: skillrepos/greetings-ci
- base: main
- head repository: gwstudent/greetings-ci
- compare: main
- Choose a Base Repository: automatically merged.
- Filter repos: skillrepos/greetings-ci, gwstudent/greetings-ci
- Create pull request button

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 GitHub Comparing changes page. At the top, it shows 'base: main' and 'compare: package'. A modal window titled 'Choose a head ref' is open, showing a dropdown menu with 'Find a branch' and two tabs: 'Branches' (selected) and 'Tags'. Under 'Branches', 'main' is listed as the default branch, and 'package' is selected. The main content area displays the following message: 'There isn't anything to compare. Commit from package. Try switching the base for your comparison.' Below this, there is a 'Create pull request' button.

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 Able to merge. These branches can be automatically merged.

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

Initial add on package Brent Laster committed on Jan 17

Showing 2 changed files with 52 additions and 0 deletions.

25 .github/workflows/publish-package.yml

```

@@ -0,0 +1,25 @@
+ name: Publish package to GitHub Packages
+ ...

```

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

The screenshot shows the GitHub interface for a pull request titled "Initial add on package #1". The "Code" tab is active, displaying the contents of the .github/workflows/publish-package.yml file. The file contains the following YAML code:

```

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

```

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 the main branch. The ".github/workflows/publish-package.yml" file is open, showing its contents:

```

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 shows the GitHub Actions interface for the "Publish package to GitHub Packages" workflow. The workflow has 0 runs. A red circle highlights the "Run workflow" button in the blue bar at the bottom right. The "Run workflow" button is also circled in red in the screenshot.

Management
Caches
Runners

0 workflow runs

This workflow has a `workflow_dispatch` event trigger.

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.

brentlaster / greetings-ci

Code Pull requests Actions Projects Wiki Security Insights Settings

greetings-ci Public forked from skillrepos/greetings-ci

main 1 Branch 0 Tags Go to file + <> Code

This branch is 2 commits ahead of, 3 commits behind skillrepos/greetings-ci:main.

brentlaster Merge pull request #1 from brentlaster/package 6a09cd0 · 3 minutes ago 27 Commits

- .github/workflows Merge pull request #1 from brentlaster/package 3 minutes ago
- gradle/wrapper Initial add 2 years ago
- src/main/java Initial add 2 years ago
- build.gradle Update build.gradle 4 days ago
- gradlew Initial add 2 years ago
- gradlew.bat Initial add 2 years ago

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

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.

github.com/brentlaster?tab=packages

Overview Repositories 217 Projects Packages Stars 1

1 package org.gradle.sample.greetings-ci Published 1 minute ago by Brent Laster in brentlaster/greetings-ci

Your profile 1

Set status

Switch account

Your repositories Your projects Your organizations Your enterprises Your stars Your sponsors Your aists

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.

The screenshot shows the GitHub repository page for `org.gradle.sample.greetings-ci`. On the right side, under the **Releases** section, there is a release labeled **greetings-ci-1.1**. Below it, the **Assets** section lists several files, with `greetings-ci-1.1.jar` circled in red.

END OF LAB

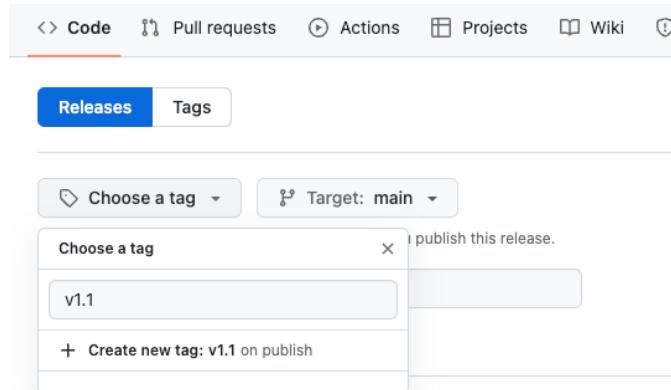
Lab 9: Creating a release

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

1. 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 `org.gradle.sample.greetings-ci`. On the right side, under the **Releases** section, there is a message **No releases published** followed by a blue **Create a new release** link, which 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. 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 page for a repository. At the top, there are tabs for 'Releases' (selected) and 'Tags'. Below the tabs, 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.' There is a 'Release title' input field and a rich text editor toolbar. Below the editor is a text area for 'Describe this release'. At the bottom, there is a file upload section with a placeholder 'Attach files by dragging & dropping, selecting or pasting them.' and a 'greetings-ci-1.1.1.jar' file listed. A red arrow points from this file to the 'Downloads' sidebar on the right, which lists the same file. The sidebar also includes sections for 'Favorites' (Dropbox, AirDrop), 'Back/Forward', and a 'Name' field.

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

The screenshot shows the 'Publish release' dialog box. It has a large input field for attaching files with the placeholder 'Attach binaries by dropping them here'. Below it is a checkbox for 'Set as a pre-release' with the note 'This release will be labeled as non-production ready'. At the bottom are two buttons: 'Publish release' (green) and '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 three dots. Below these, it says 'Releases / Initial release'. The main content area has a title 'Initial release' with a 'Latest' button. It shows a message from 'brentlaster' released now. The description reads 'Initial release of content for 1.1.' Below this is a section for 'Assets' with one item: 'greetings-ci-1.1.jar' (1006 Bytes, 2 minutes ago). There are also links for 'Source code (zip)' and 'Source code (tar.gz)' (both 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.

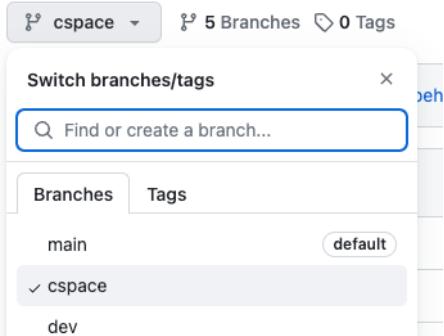
The screenshot shows the GitHub repository page for 'greetings-ci'. At the top, there are navigation links: Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below these, it shows the repository details: 'greetings-ci' (Public), forked from 'skillrepos/greetings-ci'. The main content area includes a branch status bar ('main', '2 commits ahead of 3 commits behind skillrepos/greetings-ci:main'), a code editor, and a file browser. On the right side, there is an 'About' section with a description: 'Simple starter repo for CI/CD with GitHub Actions', and sections for 'Activity', '0 stars', '0 watching', and '140 forks'. The 'Releases' section is highlighted with a red circle around the 'Initial release' entry. This entry shows the same details as the previous screenshot: 'Initial release' (Latest, 1 minute ago).

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

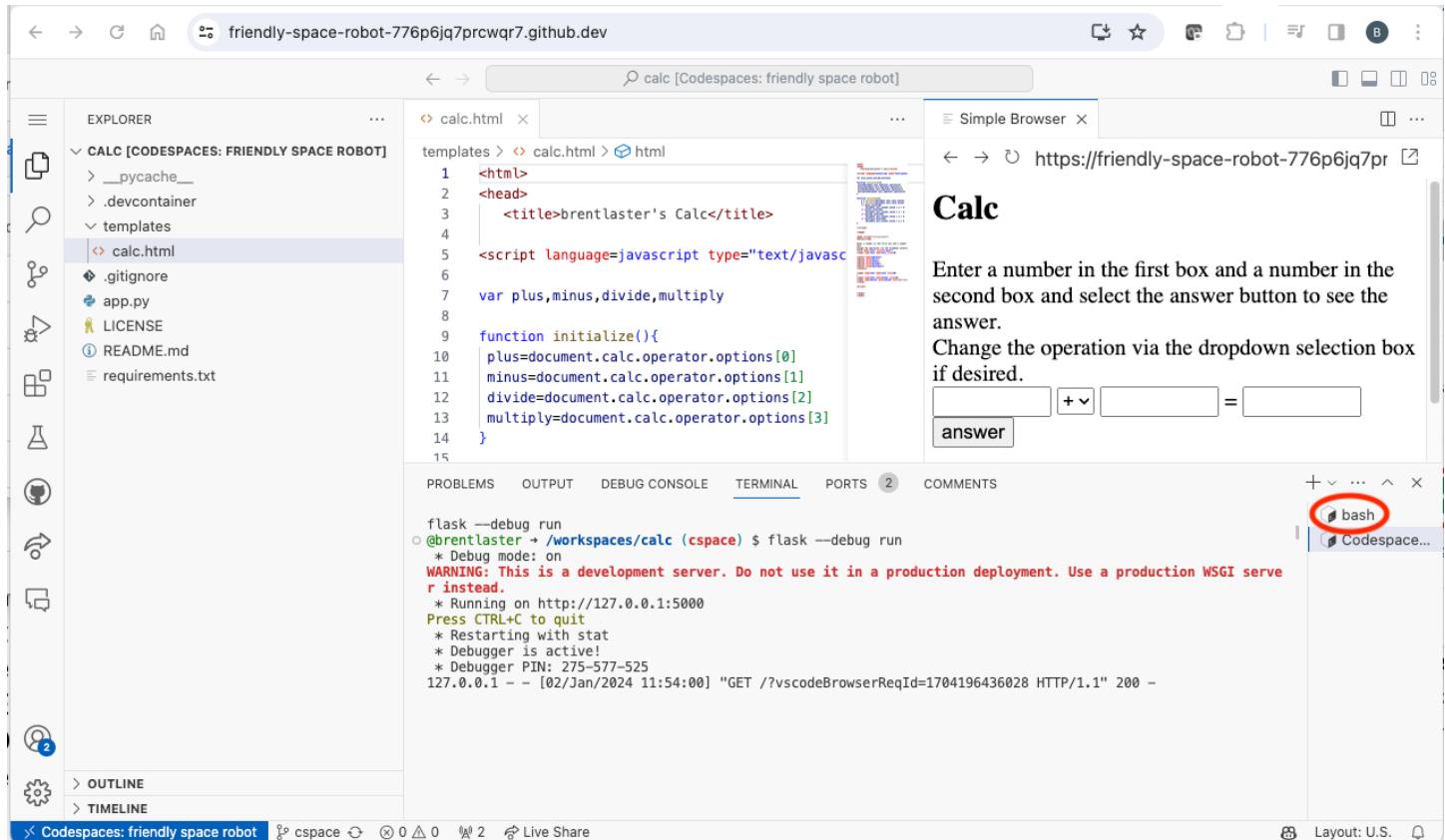
1. Go back to the `github.com/<github-userid>/calc` project. In that project, select the **cspac** branch.



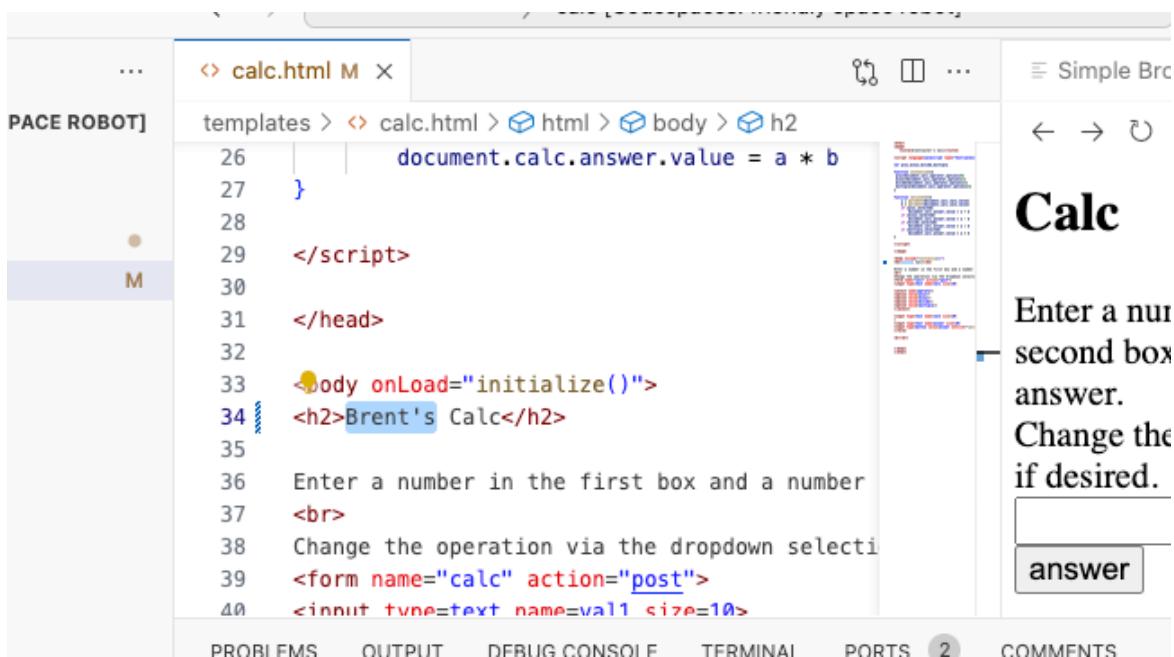
2. Click on the **<> Code** button, then select the **Codespaces** tab, and then select **Create codespace on cspac**.

A screenshot of the same GitHub repository page. The 'cspac' branch is still selected. On the right side, there's a 'Code' button with a green background and white text, which has a red circle around it. Next to it is a 'Codespaces' tab with a red circle around it, labeled with the number '2'. Below that, a section titled 'Codespaces' shows a message 'No codespaces' and a link 'Create codespace on cspac' which is also highlighted with a red circle and labeled with the number '3'.

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.)



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.



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 two panes. On the left is the code editor for 'calc.html' with the following content:

```

<h2>Brent's Calc</h2>

```

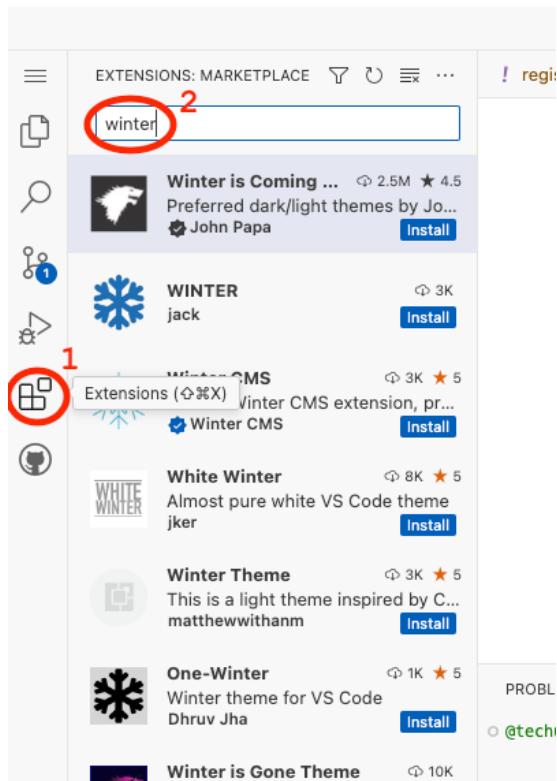
On the right is the 'Simple Browser' pane showing the result:

Brent's Calc

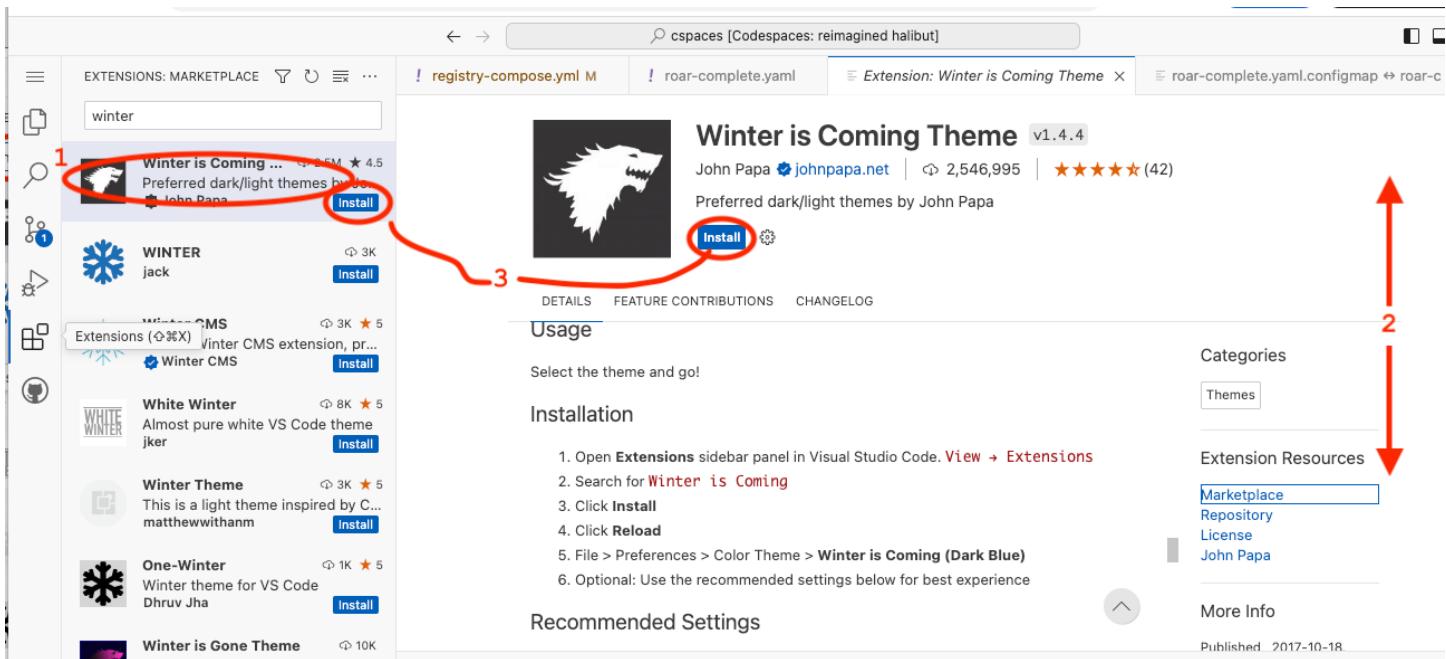
Enter a number in the first box and a second box and select the answer button.

Change the operation via the dropdown.

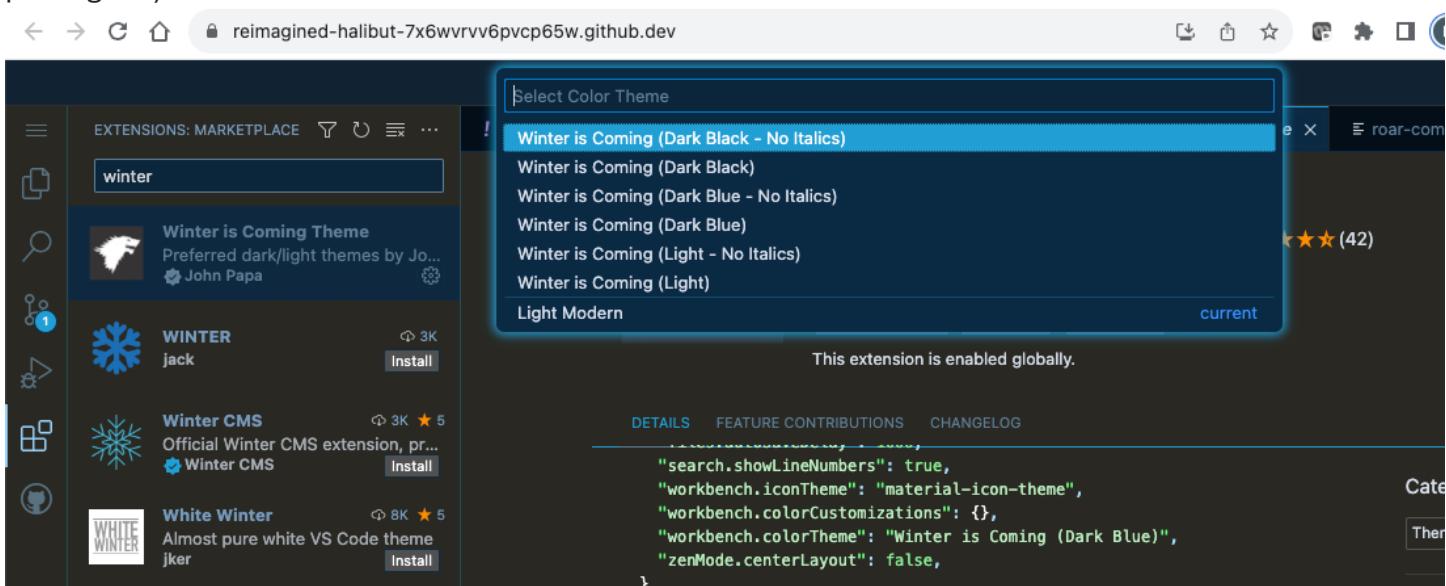
6. Next, let's see how to modify our codespace environment. We'll install an extension for the color theme. For practice, we'll use the "Winter is Coming" theme. Click on the *Extensions icon* (#1 in figure below), then in the *search bar* type in "winter" to quickly find the extension (#2).



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.)



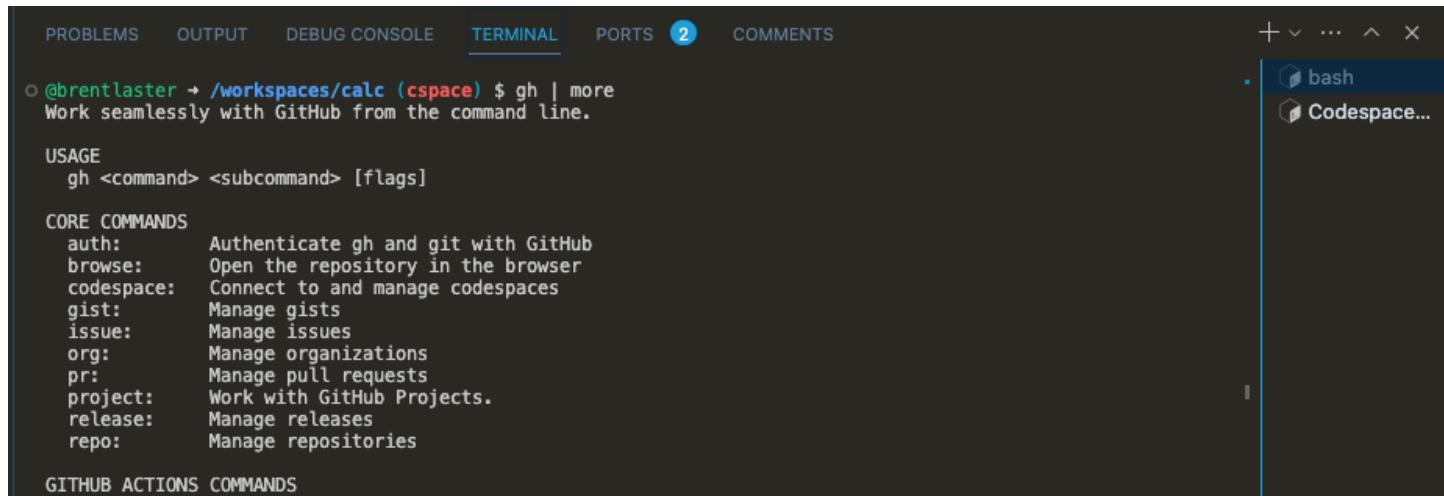
END OF LAB

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



The screenshot shows a terminal window within a code editor interface. The tab bar at the top includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), 'PORTS' (with a '2' notification), and 'COMMENTS'. The terminal content displays the output of the command '\$ gh | more'. It starts with a welcome message: '@brentlaster + /workspaces/calc (cspace) \$ gh | more Work seamlessly with GitHub from the command line.' Below this is the 'USAGE' section: 'gh <command> <subcommand> [flags]'. The 'CORE COMMANDS' section lists various commands with their descriptions: 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), and repo (Manage repositories). At the bottom, there is a section for 'GITHUB ACTIONS COMMANDS'.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS  2    COMMENTS
+ v ... ^ x
○ @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
```

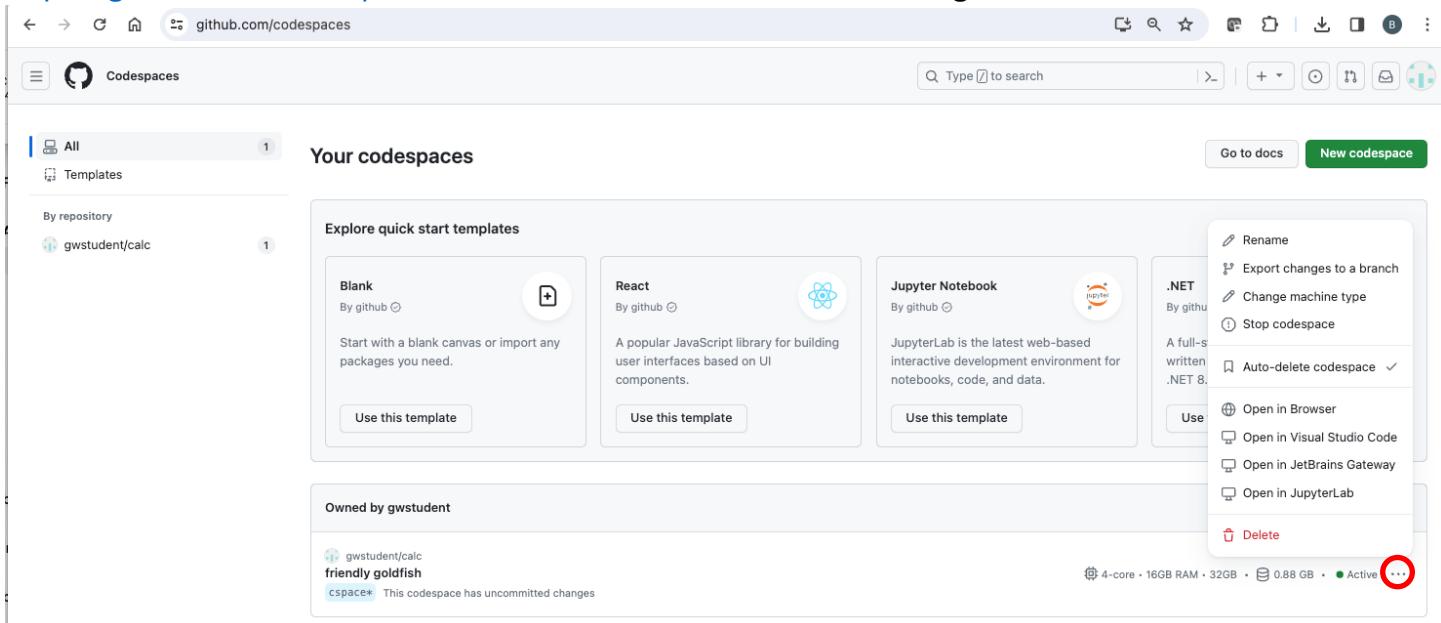
6. You can also clone repos easily with the command line. Change up one directory to not clone within the current directory. Then run the command to clone down your other repo.

```
$ 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.



END OF LAB

Demo: Copilot

That's all - THANKS!

APPENDIX 1

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

A. Resetting credential helpers: Especially on Windows, if you are pasting in your token for the password, but still getting an error message referencing password authentication, you may be running into issues because you have previous credentials stored in the *credential helper*. One of the things you can try in this case is resetting the stored credentials via:

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

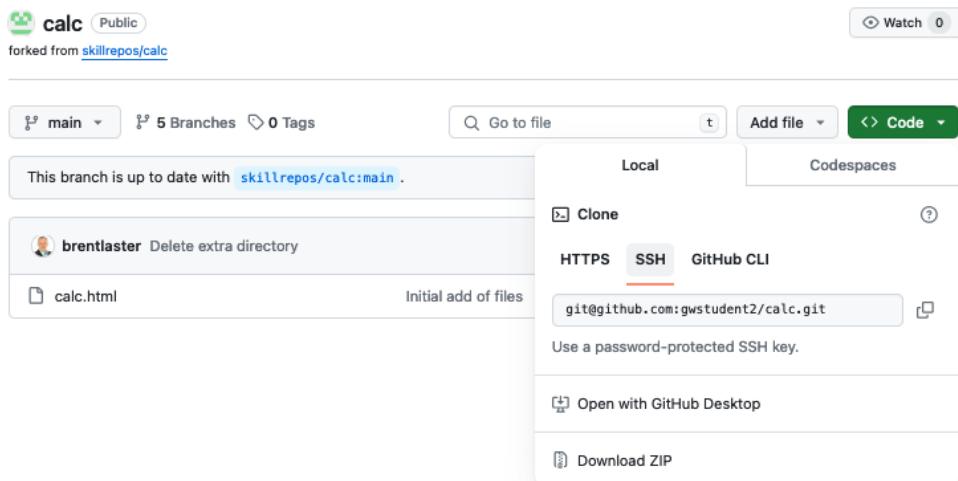
Then you do your push as per the lab. It will probably pop up a text entry box for you to add your username in and another to paste in your password (PAT) and then will replace your credentials with those and complete the push.

(Note: If you prefer to disable the global credentials helper entirely, you can try
\$ git config --unset --system credentials.helper

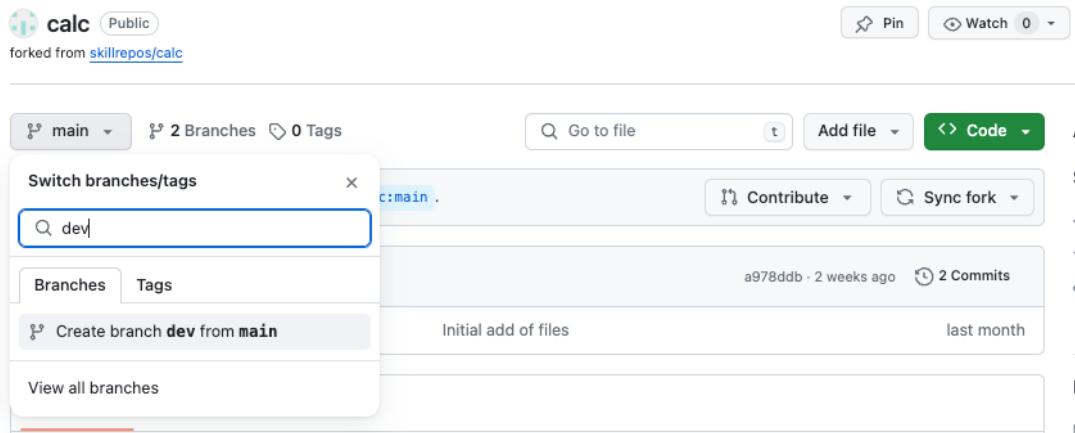
This may or may not work depending on if you have access to do this.)

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

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



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



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

The screenshot shows a GitHub repository page for a project named 'calc'. At the top, there's a green 'Public' button and a 'forked from skillrepos/calc' link. On the right, there are 'Pin' and 'Watch' buttons. Below the header, there are navigation links for 'dev' (selected), '3 Branches', '0 Tags', and search fields for 'Go to file' and 'Add file'. A green 'Code' button is also present. A message at the top states 'This branch is up to date with skillrepos/calc:main.' Below this, a commit by 'brentlaster' is shown, which deleted an extra directory. The commit was made 2 weeks ago and has 2 commits. The commit details show files 'calc.html' and 'README' were added initially, and 'calc.html' was modified later. The 'calc.html' file is currently selected.

Click on the pencil icon to edit the file.

The screenshot shows the 'calc.html' file editor on GitHub. The file contains the following code:

```

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

```

At the top, there's a breadcrumb navigation 'calc / calc.html'. Below it, a commit by 'Brent Laster' is shown with the message 'Initial add of files'. The commit hash is 'a1d22c4' and it was made 'last month'. To the right, there's a 'History' link and a prominent 'Edit this file' button, which is highlighted with a black box. Below the commit, there are buttons for 'Raw', 'Copy', 'Download', 'Blame', and 'Code 55% faster with GitHub Copilot'.

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.

← → ⌂ ⌂ github.com/gwstudent/calc/edit/dev/calc.html

gwstudent / calc

Code Pull requests Actions Projects Wiki Security Insights Settings

calc / calc.html in dev Cancel changes Commit changes...

Edit Preview Spaces 2 No wrap

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

Commit changes

Commit message

Update calc.html

Extended description

Add an optional extended description..

Commit directly to the dev branch

Create a new branch for this commit and start a pull request

[Learn more about pull requests](#)

Cancel Commit changes

APPENDIX 2

Alternative approaches to forking a repo if you can't use the actual Fork button.

IMPORTANT NOTE: In these examples, the repository “sec-demo” is used. Change the name to whatever repository you are trying to fork.

Option 1 – Using import

1. Sign into GitHub if not already signed in.
2. Go to <https://github.com/new/import>
3. On that page, fill out the form as follows:

In “Your source repository details”, in the “The URL for your source repository *” field, enter

<https://github.com/skillrepos/<repo-name>>

Under “Your new repository details”, make sure your userid shows up in the “Owner *” field and enter
the desired <repo-name> in the “Repository name *” field.

The visibility field should be set to “Public”.

Import your project to GitHub

Import all the files, including revision history, from another version control system.

Required fields are marked with an asterisk ().*

Support for importing Mercurial, Subversion and Team Foundation Version Control (TFVC) repositories ended on April 12, 2024. For more details, see the [changelog](#).

Your source repository details

The URL for your source repository *

`https://github.com/skillrepos/sec-demo`

1

Learn more about [importing git repositories](#).

Please enter your credentials if required for cloning your remote repository.

Your username for your source repository

Your access token or password for your source repository

Your new repository details

Owner *



Repository name *

2

sec-demo is available.

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.

You are creating a public repository in your personal account.

Cancel

Begin import

3

4. If you haven't already, click on the green "Begin import" button.

- After this, you should see the import processing...

The screenshot shows a GitHub repository page for 'brentlaster/sec-demo/import'. The main heading is 'Preparing your new repository' with a sub-instruction: 'There is no need to keep this window open. We'll email you when the import is done.' Below this, there's a large circular progress indicator and the text 'Your import will begin shortly...'. The GitHub interface includes a search bar, navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings, along with user profile icons.

Option 2 – Using clone and push

- Sign into GitHub if not already signed in.
- If you don't already have one, create a GitHub token or SSH key. If you are familiar with SSH keys, you can add your public key at <https://github.com/settings/keys>. Otherwise, you can just create a "classic" token by following the instructions at <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-personal-access-token-classic>. If you use a GitHub token, make sure to save a copy of it to use in the push step.
- Clone down the [skillrepos/sec-demo](#) repository.

`git clone https://github.com/skillrepos/sec-demo (if using token)`

or

`git clone git@github.com:skillrepos/sec-demo (if using ssh)`

- Create a new repository in your GitHub space named `sec-demo`. Go to <https://github.com/new>. Fill in the "repo name" field with "sec-demo" and then click on the "Create repository" button.

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 * brentlaster **Repository name *** sec-demo **1**
 brentlaster sec-demo is available.

Great repository names are short and memorable. Need inspiration? How about [animated-octo-barnacle](#) ?

Description (optional)

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

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

Initialize this repository with:

Add a README file
 This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

Grant your Marketplace apps access to this repository

You are subscribed to 1 Marketplace app.

Docker for GitHub Copilot (auto-installed)
 Learn about containerization, generate Docker assets and analyze project vulnerabilities in GitHub Copilot

You are creating a public repository in your personal account.

Create repository **2**

5. On the page that comes up after that, select the appropriate protocol (https or ssh) and then follow the instructions for "...or push an existing repository from the command line" to push your content back to the GitHub repository. If you're using https you will be prompted for a password at push time. Just paste in the classic token. (Note that for security reasons, you will not see the token displayed.)

The screenshot shows a GitHub repository page for 'sec-demo' owned by 'brentlaster'. The page includes sections for 'Start coding with Codespaces', 'Add collaborators to this repository', 'Quick setup', command-line instructions, and a 'ProTip!' note.

Start coding with Codespaces
Add a README file and start coding in a secure, configurable, and dedicated development environment.
[Create a codespace](#)

Add collaborators to this repository
Search for people using their GitHub username or email address.
[Invite collaborators](#)

Quick setup — if you've done this kind of thing before

Set up in Desktop [HTTPS](#) **SSH** [git@github.com:brentlaster/sec-demo.git](#)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# sec-demo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:brentlaster/sec-demo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:brentlaster/sec-demo.git
git branch -M main
git push -u origin main
```

ProTip! Use the URL for this page when adding GitHub as a remote.