

GitHub Fundamentals BootCamp Labs

Learn the complete GitHub – from code management to Copilot

Revision 1.5 – 05/09/24

Tech Skills Transformations LLC / Brent Laster

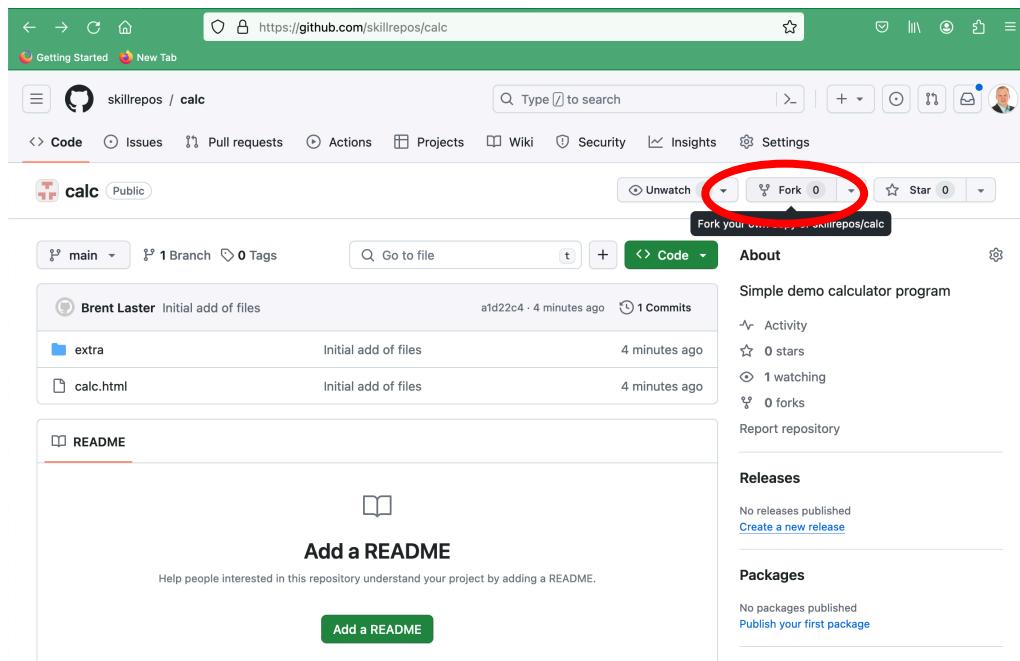
Setup and prerequisites

1. In order to do some of the labs in this class, you will need to have a personal access token (PAT) setup and also two separate GitHub userids, as well as a version of Git installed.
2. Git can be installed by going to <https://git-scm.org> and following the instructions there for your OS.
3. To create the second GitHub userid, just select another email address and sign up for the free tier at GitHub.com.
4. You can set up the PAT in advance by following the instructions [here](#) or do it as part of the first lab.
5. If you are doing the labs on Windows, it is recommended to use the Git Bash shell that can be installed with Git for Windows.

Lab 1 – Getting Started

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

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



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 yellow callout bubble points to the 'Owner' dropdown with the text 'uncheck'. A red circle highlights the checkbox labeled 'Copy the main branch only'. A green circle highlights the 'Create fork' button.

Create a new fork

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

Required fields are marked with an asterisk (*).

Owner * Repository name *

brentlaster / calc

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 'brentlaster/calc'. 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 highlighted with a red circle 5.

brentlaster / calc

1

2

3 Local

4

5

This branch is up to date with skillrepos/calc:main

Brent Laster Initial add of files

extra

calc.html

README

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. Then change into the local working directory.

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

```
$ cd calc
```

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

```
$ git config user.name "your name"
```

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

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

```
$ git remote -v
```

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

Edit calc.html and change

<title>Calc</title>i

to

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

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

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

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

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

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

(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.**

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

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

<input type="checkbox"/> read:ssh_signing_key Generate token Cancel	Read public user SSH signing keys
--	---

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

The screenshot shows the GitHub 'Personal access tokens (classic)' page. On the left, there's a sidebar with 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens' (which is expanded, showing 'Fine-grained tokens' and 'Tokens (classic)'). A 'Beta' button is next to 'Personal access tokens'. Below the sidebar, a message says 'Tokens you have generated that can be used to access the [GitHub API](#)'. A blue box contains the text 'Make sure to copy your personal access token now. You won't be able to see it again!'. Below this, a green box shows a copied token: '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' with a checkmark and a 'Copied!' message.

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

\$ git push -u origin main:dev

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



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

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

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

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

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

END OF LAB

Lab 2 – Pull requests

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

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

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

This screenshot continues from the previous one. The 'dev' branch is still selected. A modal message says 'This branch is 1 commit ahead of skillrepos/calc:main'. The main repository view shows a commit from 'Brent Laster' updating the title, dated '61e42da · 8 hours ago' with '3 Commits'. Below this, a file named 'calc.html' is listed with the status 'Updating title' and the date '8 hours ago'. At the bottom, there are links for 'calc.html' and 'README'.

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

Brent Laster Updating title d9e79db · 22 minutes ago History

Code Blame 59 lines (46 loc) · 1.29 KB

Older Newer

```

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

```

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

Brent Laster Updating title d9e79db · 24 minutes ago History

Code Blame 59 lines (46 loc) · 1.29 KB

Older Newer

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.

History for calc / calc.html on dev All users All time

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

Updating title

Brent Laster committed 28 minutes ago

1 parent a1d22c4 commit d9e79db

Showing 1 changed file with 2 additions and 2 deletions.

Whitespace Ignore whitespace Split Unified

```

diff --git a/calc.html b/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>

```

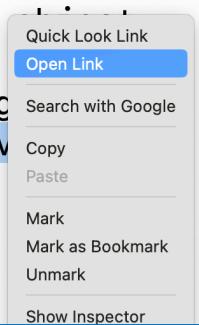
5. Let's now merge our change from the dev branch to main via a pull request. **Switch back to the terminal where you did the commit and push.**

In the output from the push, you should see a link (*highlighted in the screenshot below*). 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 --

brentlaster / calc

Type ⌘ to search

Code Pull requests Actions Projects Wiki Security Insights Settings

calc Public

forked from [skillrepos/calc](#)

Pin Watch 0

dev had recent pushes 26 minutes ago

Compare & pull request

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

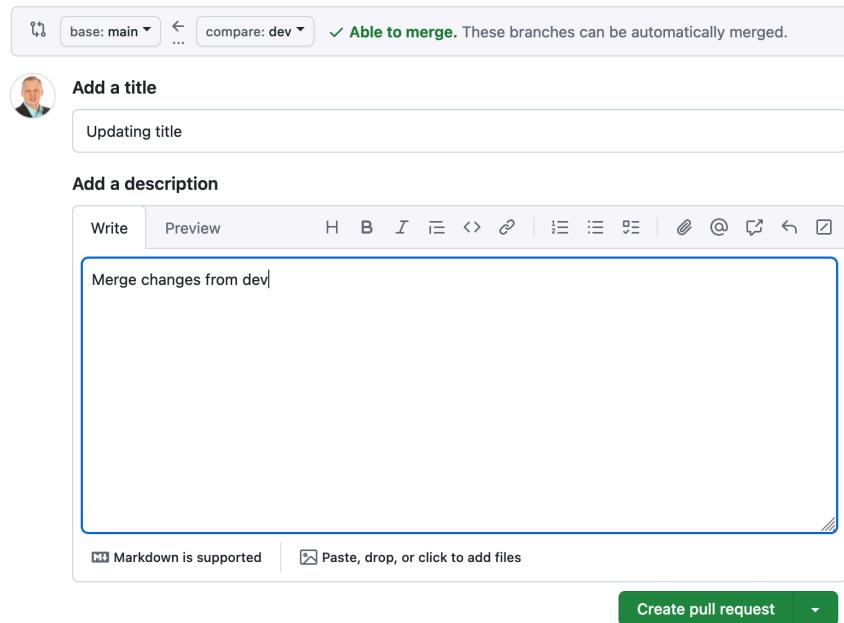
Write Preview

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.



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" with a "Updating title #1" message. Below it, a commit history shows a single commit from "brentlaster" merging "dev" into "main". A tooltip provides instructions: "Add more commits by pushing to the `dev` branch on [brentlaster/calc](#)". A modal window is open, containing a "Merge pull request #1 from brentlaster/dev" button and an "Updating title" field. It also notes: "This commit will be authored by bclaster@nclasters.org" and has "Confirm merge" and "Cancel" buttons. To the right of the main area, there are collapsed sections for Labels, Project, Milestone, and Notifications.

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

Merge changes from dev

Updating title

brentlaster merged commit `dc98b08` into `main` now

Pull request successfully merged and closed
You're all set—the `dev` branch can be safely deleted.

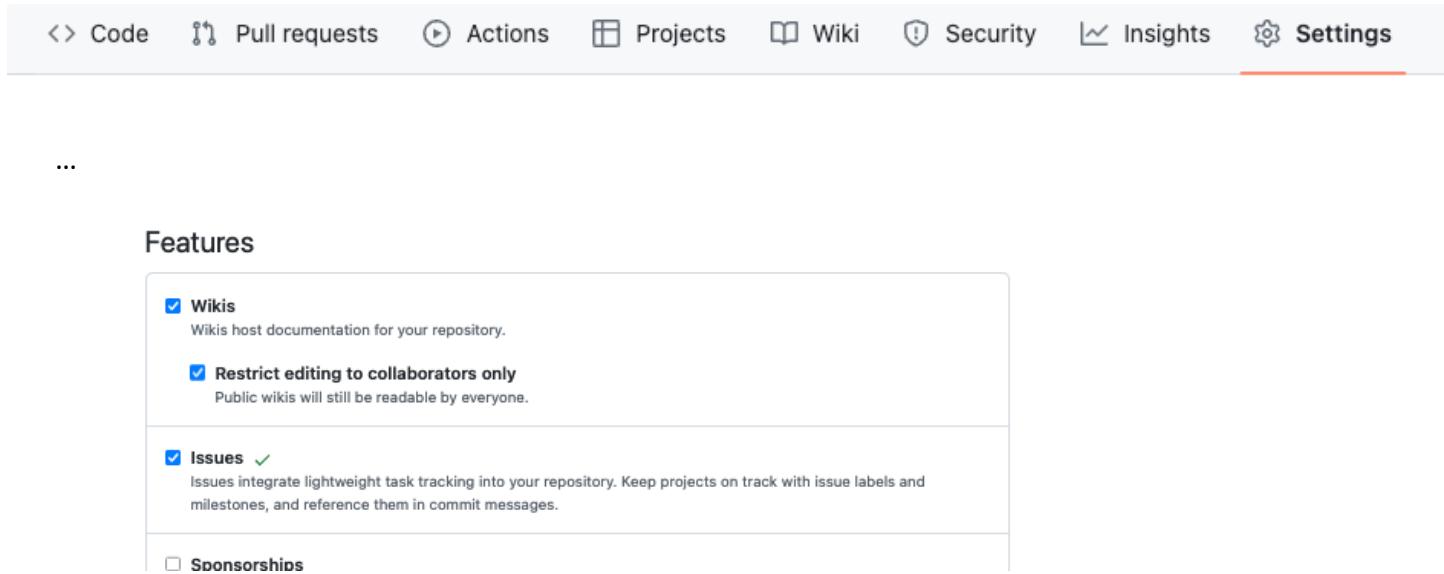
Delete branch

END OF LAB

Lab 3: Creating GitHub issues

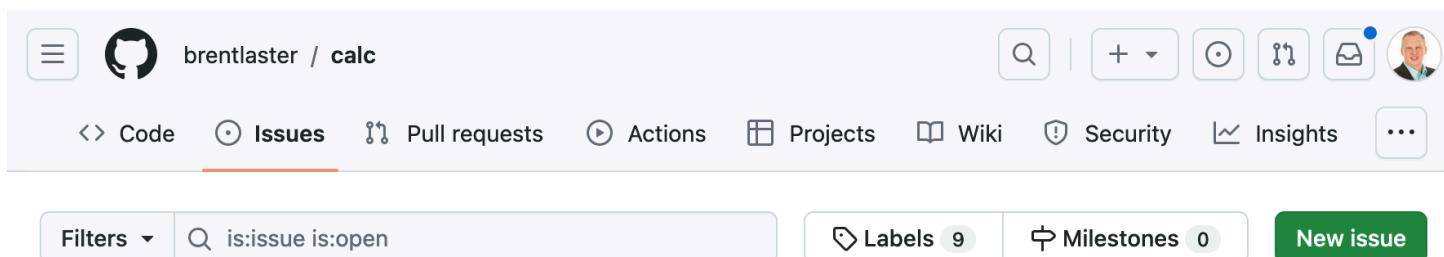
Purpose: In this lab, you'll create an issue, assign it to a user, and add labels for it.

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**.



The screenshot shows the GitHub repository settings page. At the top, there are tabs for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Settings tab is selected. Below the tabs, there is a section titled "Features". Inside this section, there are several checkboxes with descriptions. The "Issues" checkbox is checked and highlighted in green, indicating it is enabled. Other checkboxes include "Wikis", "Restrict editing to collaborators only", and "Sponsorships".

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



The screenshot shows the GitHub repository issues page. At the top, there are tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and ... The Issues tab is selected. Below the tabs, there is a search bar, a filter dropdown set to "Filters", a search input with the query "is:issue is:open", a labels counter (9), a milestones counter (0), and a prominent green "New issue" button. The interface is clean and modern, typical of GitHub's design.

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

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 user interface with a top navigation bar containing 'Wiki', 'Security', 'Insights', and a three-dot menu. Below this is a toolbar with 'Edit' and a green 'New issue' button. The main area features a light blue sidebar with a three-dot menu icon. On the right, there's a section titled 'Assignees' with a gear icon. A message says 'No one—[assign yourself](#)' with 'assign yourself' underlined in blue.

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

The screenshot shows a user interface with a sidebar on the left containing sections for 'Assignees' (with 'brentlaster' listed) and 'Labels'. A modal window titled 'Apply labels to this issue' is open, showing a 'Filter labels' input field and a list of labels. The 'documentation' label is checked with a blue circle and has a tooltip 'Improvements or additions to documentation'. The 'duplicate' label is also listed with a grey circle and a tooltip 'This issue or pull request already exists'.

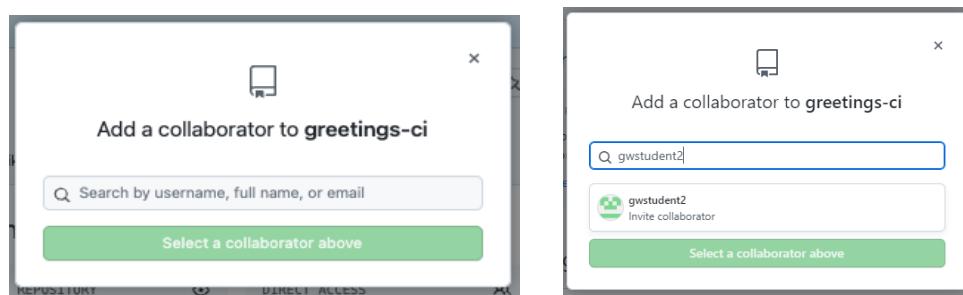
6. After this, if you click on the **Issues** tab at the top, and look at your issue, it should look like the following.

The screenshot shows a browser window for 'github.com/brentlaster/calc/issues'. The top navigation bar includes 'Code', 'Issues 1', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', and 'Insights'. The 'Issues' tab is active. Below the navigation is a search bar and filter options. The main content area shows a single open issue with the title 'Needs README documentation' and a note '#2 opened 15 hours ago by brentlaster'. The issue is highlighted with a red border.

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

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

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



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

The screenshot shows a GitHub repository page for 'brentlaster / calc'. At the top, there's a search bar with the placeholder 'Type ⌘ to search'. Below the search bar is a navigation bar with links: Code (selected), Issues (1), Pull requests, Actions, Projects, Security, and Insights.

The repository details show the owner 'brentlaster' with a profile picture, the repository name 'calc' (Public), and a note that it was forked from 'skillrepos/calc'. To the right is a 'Watch' button with a count of 0.

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

Below the message box is another navigation bar with links: Code, Issues (1), Pull requests, Actions, and Projects.

The main content area shows two user profile pictures: 'brentlaster' and another user. A message reads 'brentlaster invited you to collaborate' with 'Accept invitation' and 'Decline' buttons. The 'Accept invitation' button is circled in red. Below this, a note states that 'Owners of calc will be able to see:' followed by a list of items: Your public profile information, Certain activity within this repository, Country of request origin, Your access level for this repository, and Your IP address.

At the bottom of the content area, 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 for 'brentlaster / calc'. The navigation bar at the top is identical.

The main content area now displays a message: '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. At the top, there are tabs for Code, Issues (1), Pull requests, Actions, Projects, Wiki, and Settings. Below the tabs, the repository details are shown: 'calc' (Public, forked from skillrepos/calc). A 'Watch' button is present. The main area shows a message: 'This branch is 2 commits ahead of skillrepos/calc:main'. Below this are commit details for 'brentlaster': 'Merge pull request #1 from br...' (dc98b08 · yesterday), '3 Commits'. A list of files includes 'extra' (Initial add of files, last week) and 'calc.html' (Updating title, yesterday). A large box at the bottom is titled 'README' and contains a 'Add a README' button, which is highlighted with a red oval.

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

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

A screenshot of a GitHub repository named 'calc'. The file 'README.md' is being edited in the 'main' branch. The 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' and 'Soft wrap'.

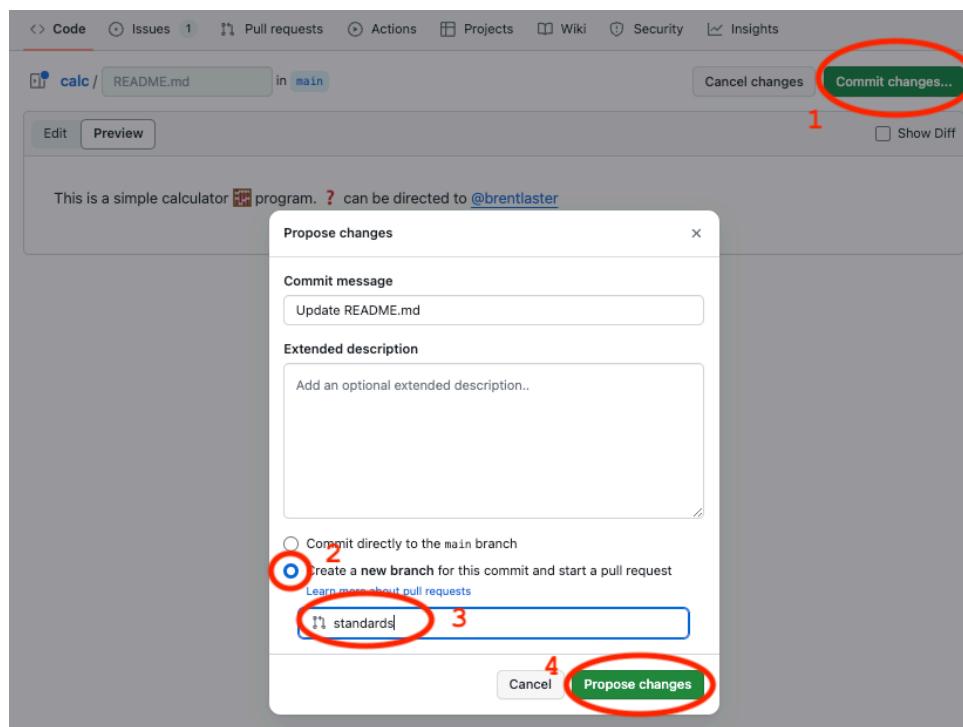
3. Click on the Preview tab (next to Edit) to see how this will render once committed.

A screenshot of the same GitHub repository and file ('README.md' in 'main'). The 'Preview' tab is selected, showing the rendered content:

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

The interface includes tabs for 'Edit' and 'Preview', and a checkbox for 'Show Diff'. A 'Commit changes...' button is visible at the top right.

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



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

The screenshot shows the GitHub '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, followed by a prominent green 'Create pull request' button, which is circled in red. The main content area shows a commit history from 'Commits on Jan 1, 2024', featuring a single commit titled 'Create README.md' by 'brentlaster' (verified). The commit details show a file named 'README.md' with one addition and zero deletions. The diff view highlights the added content: '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.

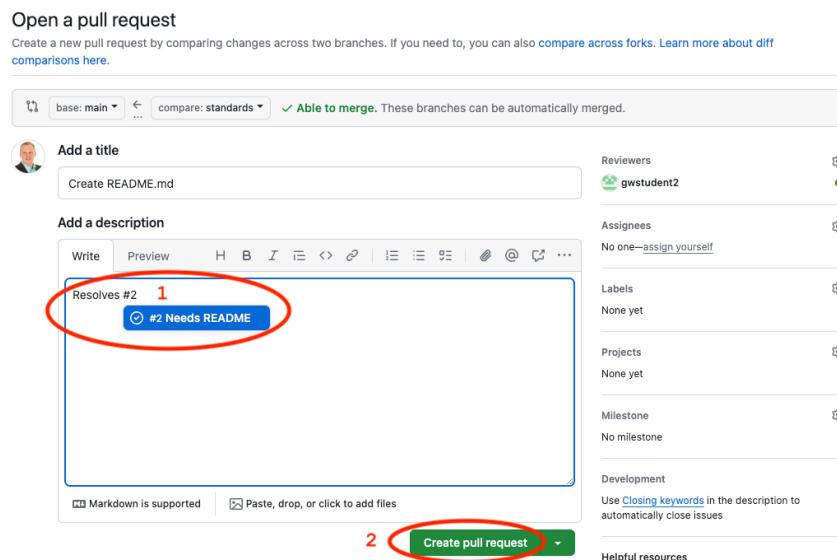
The screenshot shows the 'Open a pull request' page. The top bar includes dropdowns for 'base: main' and 'compare: standards', and a green 'Able to merge' status. On the left, there are fields for 'Add a title' (with a placeholder 'Create README.md') and 'Add a description' (with a text area containing 'Add your description here...'). On the right, the 'Reviewers' section is highlighted, showing a list with one user selected: 'gwstudent2'. A red circle with the number '1' is placed over the 'Reviewers' link, and another red circle with the number '2' is placed over the selected user 'gwstudent2' in the list. Other sections visible include 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Development' (using closing keywords to automatically close issues).

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

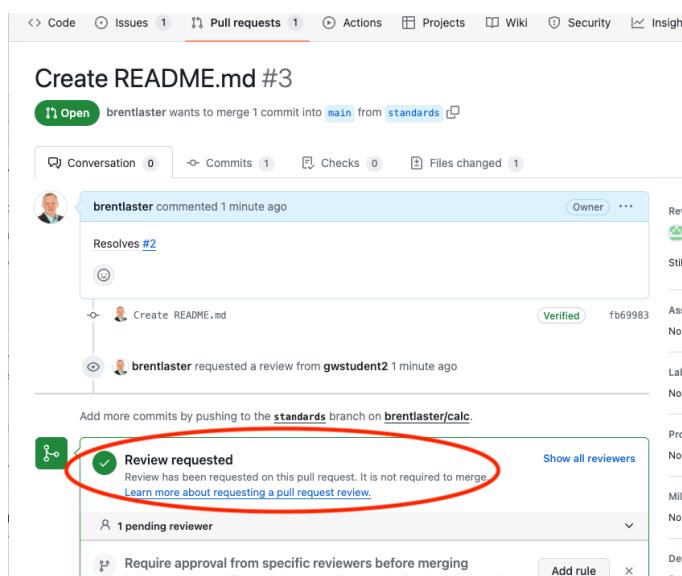
Resolves #2

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

Then click on the “Create pull request” button.



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



END OF LAB

Lab 5: Completing a pull request with reviewers

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

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

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

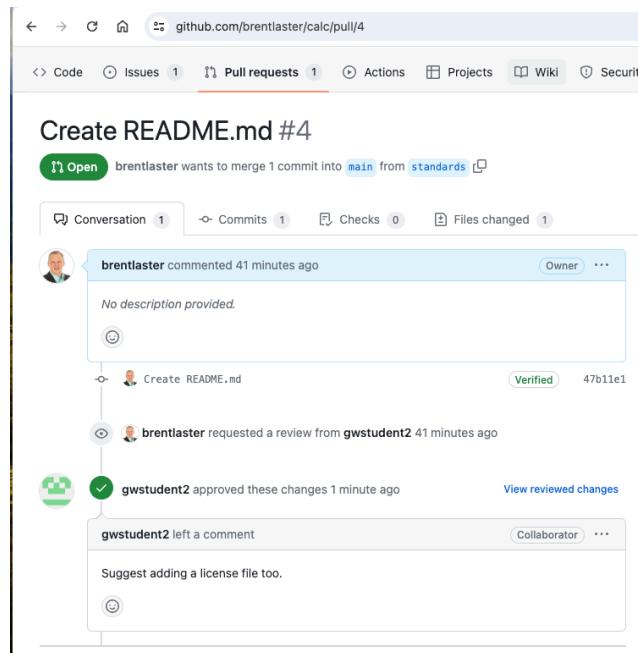
- OR -

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

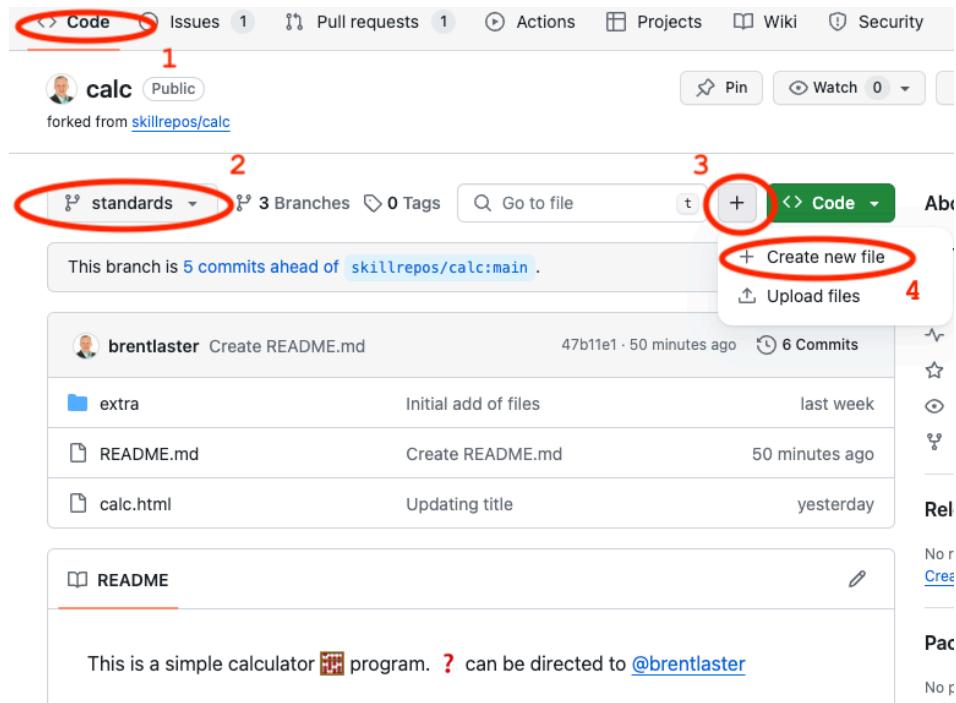
2. This will open up the pull request. There is a button at the top to "Add your review". Click on that.

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

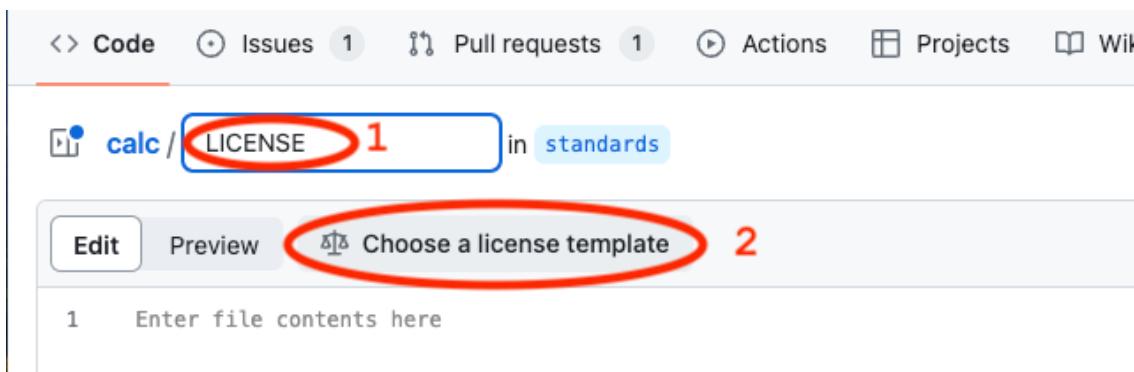
4. Go to the session with your original GitHub userid or log out of the other one and log back in if you need to. Go to the **Pull requests** menu at the top, find the pull request and click on the commit message. Then you should see a screen like below.



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.	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.	
GNU General Public License v3.0			
MIT License 1	Permissions ✓ Commercial use ✓ Modification ✓ Distribution ✓ Private use	Limitations ✗ Liability ✗ Warranty	Conditions ⓘ License and copyright notice
BSD 2-Clause "Simplified" License	This is not legal advice. Learn more about repository licenses.		
BSD 3-Clause "New" or "Revised" License			
Boost Software License 1.0	MIT License		

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.

LICENSE in standards

Commit changes

Commit directly to the standards branch

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

[Learn more about pull requests](#)

Commit changes

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 a GitHub pull request interface for a repository named 'Create README.md #4'. At the top, there's a navigation bar with tabs for Code, Issues (1), Pull requests (1), Actions, Projects, Wiki, Security, Insights, and Settings. The 'Pull requests' tab is highlighted with a red circle and has a red number '1' below it. Below the navigation bar, there's a summary: 'Open brentlaster wants to merge 2 commits into main from standards'. The 'Commits' tab (2) and 'Files changed' tab (3) are both circled in red. The 'Files changed' tab is active, showing two files: LICENSE and README.md. The LICENSE file contains the MIT License text. The README.md file contains a single line of text: '+ This is a simple calculator :abacus: program. :question: can be directed to @brentlaster (<https://github.com/brentlaster>)'.

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 issue is titled 'Needs README #2' and is marked as 'Closed'. The timeline shows the following activity:

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

Labels: documentation

Assignees: brentlaster

Projects: None yet

Milestone: No milestone

Development: Successfully merging a pull request may close this issue.

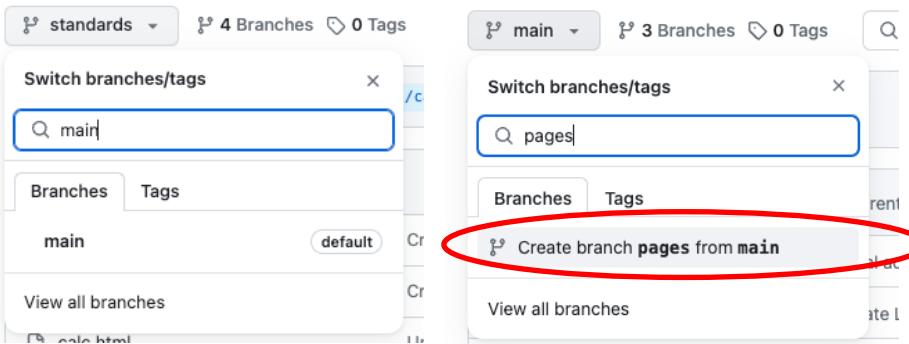
Notifications: Customize, Unsubscribe

END OF LAB

Lab 6: Adding a GitHub Pages website for your repository

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

1. 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 * 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

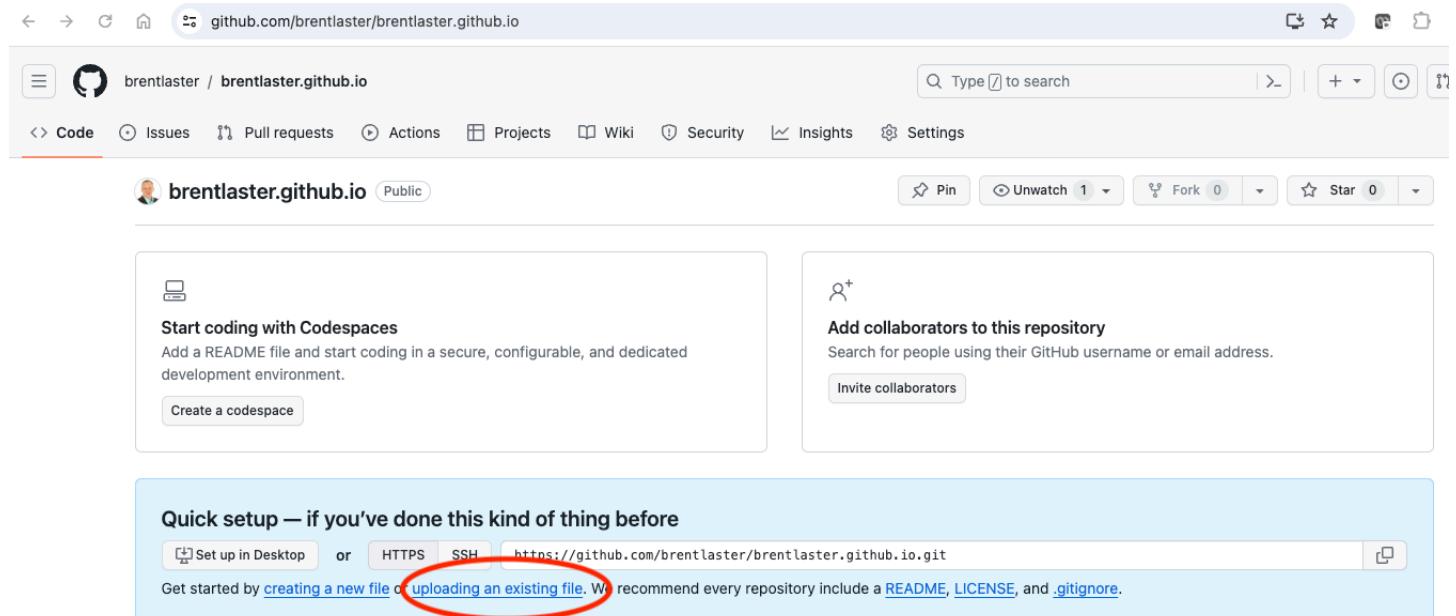
Public
Anyone on the internet can see this repository. You choose who can commit.

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

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

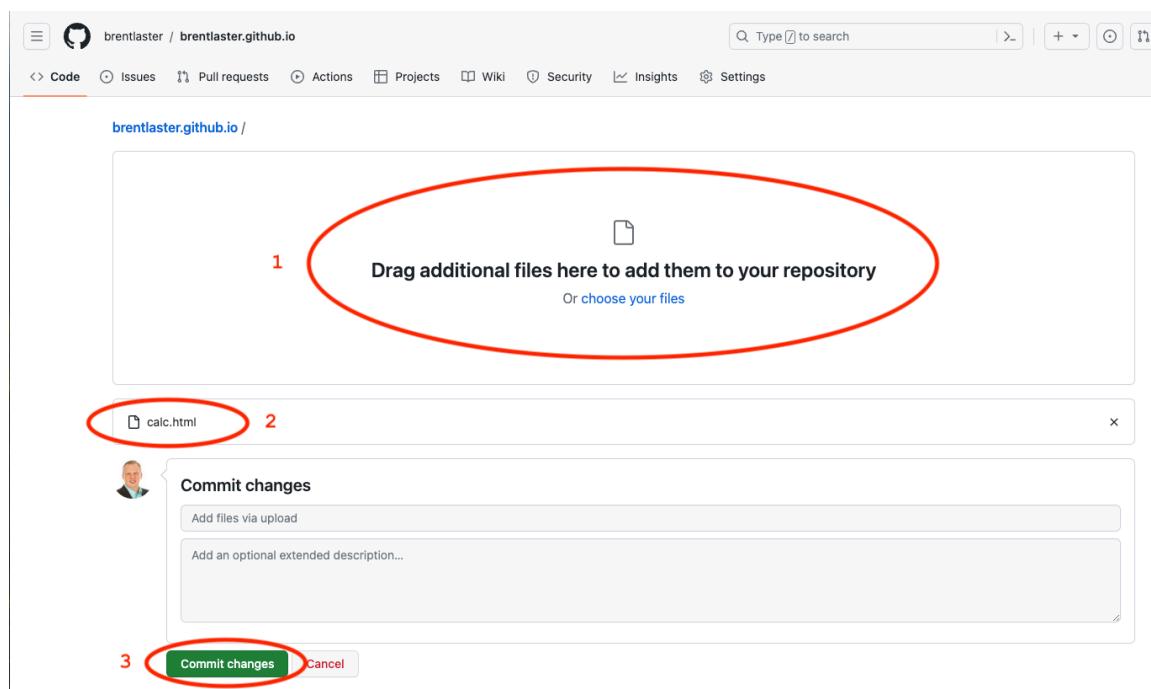
Create repository

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 a GitHub repository page for 'brentlaster/brentlaster.github.io'. The 'Code' tab is selected. In the center, there's a 'Quick setup – if you've done this kind of thing before' section. It contains links for 'Set up in Desktop' (disabled), 'HTTPS' (selected), 'SSH', and a URL 'https://github.com/brentlaster/brentlaster.github.io.git'. Below these links is a note: 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)'. The link 'uploading an existing file' is circled in red.

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



The screenshot shows the 'Commit changes' dialog box. At the top, there's a large input field with a red circle containing the number '1' and the text 'Drag additional files here to add them to your repository' followed by 'Or [choose your files](#)'. Below this, a file named 'calc.html' is listed with a red circle containing the number '2'. At the bottom, there's a green 'Commit changes' button with a red circle containing the number '3'.

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

brentlaster / brentlaster.github.io

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

main calc.html

brentlaster.github.io / calc.html

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

63 lines (50 loc) · 1.4 KB

Edit this file

Code Blame

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

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

brentlaster / brentlaster.github.io

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

brentlaster.github.io / docs

Cancel changes Commit changes...

main

Edit Preview

brentlaster / brentlaster.github.io

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

brentlaster.github.io / docs / index.html

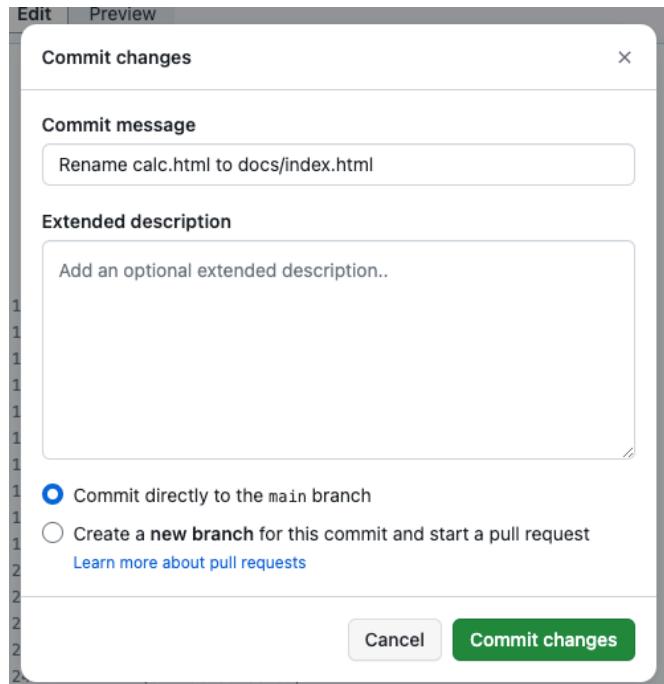
Cancel changes Commit changes...

main

Edit Preview

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

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



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

1 GitHub Pages

2

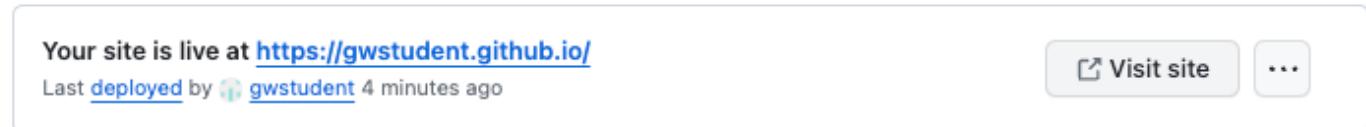
3

4

- 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.



brentlaster.github.io

Calc

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.

[] + [] = [] Get answer

OPTIONAL:

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

Unwatch 1 ▾ Fork 0 ▾ Star 0 ▾

Code ▾ About 

4 Commits

Code repo for the web page for the calc code

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

The image shows two side-by-side screenshots. On the left is a 'Edit repository details' modal window. It contains fields for 'Description' (Simple web calculator program), 'Website' (https://brentlaster.github.io/), and 'Topics' (calculator, example-code). Below these are checkboxes for 'Include in the home page' (Releases, Packages, Deployments) which are all checked. At the bottom are 'Cancel' and 'Save changes' buttons. On the right is a GitHub user profile page for 'brentlaster'. It shows the name 'Simple web calculator program', a link to 'brentlaster.github.io/' with a 'Copy' icon, and two topic tags: 'calculator' and 'example-code'. Below this is an 'Activity' section showing 0 stars, 1 watching, and 0 forks.

END OF LAB

Lab 7: Learning about GitHub Actions

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

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

The image shows a screenshot of a web browser displaying the GitHub repository 'greetings-ci' owned by 'skillrepos'. The repository is public and has 2 branches and 0 tags. A pull request from 'brentlaster' titled 'Delete extra directory' was merged 3 days ago. The repository description is 'Simple starter repo for CI/CD with GitHub Actions'. On the right side, there is a 'Fork your own copy of skillrepos/greetings-ci' button. The GitHub interface includes standard navigation bars like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

The screenshot shows the GitHub fork creation interface for the repository `skillrepos/greetings-ci`. A yellow callout box points to the `Owner` dropdown, which is set to `brentlaster`. A red circle highlights the `Copy the main branch only` checkbox, which is checked. A green circle highlights the `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. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

brentlaster / greetings-ci greetings-ci is available.

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

Description (optional)

Simple starter repo for CI/CD with GitHub Actions

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

ⓘ You are creating a fork in your personal account.

Create fork

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 `Actions` tab is highlighted with a red circle. The page displays a list of recent commits and their details. The sidebar on the right provides repository statistics: 0 stars, 0 watching, and 1 fork.

gwstudent / greetings-ci Public

forked from [skillrepos/greetings-ci](#)

Code Pull requests Actions Projects Wiki Security Insights Settings

This branch is up to date with skillrepos/greetings-ci:main.

Commit	Author	Date	Commits
Brent Laster add extra dir	97df205 7 minutes ago	2 commits	extra gradle/wrapper src/main/java build.gradle gradlew gradlew.bat
		7 minutes ago	
		14 minutes ago	
		14 minutes ago	
		14 minutes ago	
		14 minutes ago	

About

Simple starter repo for CI/CD with GitHub Actions

0 stars
0 watching
1 fork

Releases

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

Packages

No packages published

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

Automation

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

Browse all categories

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

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

gwstudent / greetings-ci (Public)
forked from skillrepos/greetings-ci

Code Pull requests Actions Projects Wiki Security Insights Settings

Get started with GitHub Actions

Categories: Automation, Continuous integration (selected), 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

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

gwstudent / greetings-ci (Public)
forked from skillrepos/greetings-ci

Code Pull requests Actions Projects Wiki Security Insights Settings

Get started with GitHub Actions

Categories: Automation, Continuous integration (selected), 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 ● (circled in red)

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

The screenshot shows a GitHub repository interface. The top navigation bar includes 'Code', 'Pull requests', 'Actions', 'Projects', 'Wiki', and 'Security'. Below the navigation is a search bar with the text 'greetings-ci/.github/workflows/ gradle.yml' and a dropdown indicating it's in the 'main' branch. At the bottom of the search bar are 'Edit new file' and 'Preview' buttons.

TO

The screenshot shows the same GitHub repository interface after the name change. The search bar now displays 'greetings-ci/.github/workflows/ pipeline.yml' and is also in the 'main' branch. The 'Edit new file' and 'Preview' buttons are visible at the bottom.

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.

The screenshot shows the 'pipeline.yml' workflow file. A large red box highlights the entire code block starting from line 50. The code is as follows:

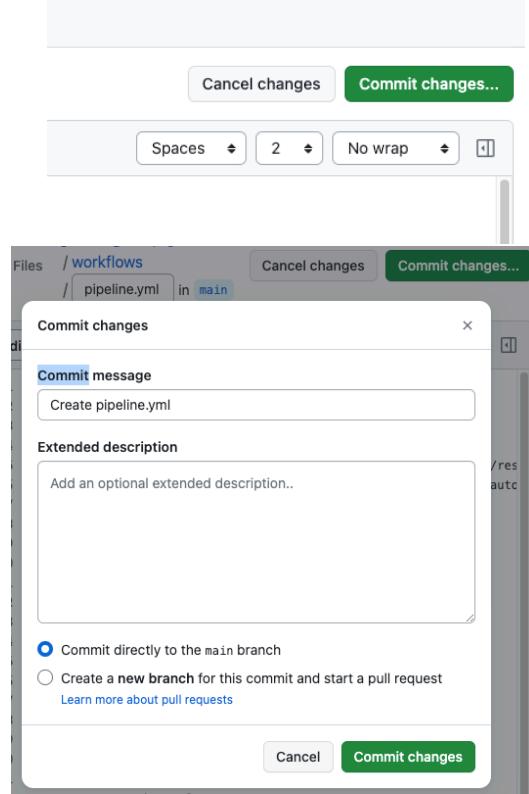
```

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

```

A red annotation 'highlight/select, and delete' is placed over the code block starting at line 50.

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



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

The screenshot shows the GitHub Actions page. The top navigation bar includes 'Code', 'Pull requests', 'Actions' (which is highlighted), 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Actions' sidebar lists 'All workflows' (Java CI with Gradle) and 'Management' (Caches, Runners, Beta). The main content area shows 'All workflows' with a 'Help us improve GitHub Actions' survey and '1 workflow run' for 'Create pipeline.yml'. The run details show it was triggered 'now' and is currently 'In progress'. Filter options for 'Event', 'Status', 'Branch', and 'Actor' are available.

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

The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with 'Actions' selected, followed by 'New workflow', 'All workflows', 'Java CI with Gradle' (which is highlighted), 'Management', 'Caches', 'Runners', and a 'Beta' button. The main area is titled 'Java CI with Gradle' and 'pipeline.yml'. It features a 'Help us improve GitHub Actions' card with a 'Give feedback' button. Below that is a table showing '1 workflow run' for the 'Create pipeline.yml' job. The job status is green, indicating success. The commit information is: 'Java CI with Gradle #1: Commit cc93266 pushed by brentlaster'. The run was completed 1 minute ago and took 36s. There are dropdown menus for Event, Status, Branch, and Actor.

- 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 #1' workflow run. The top navigation bar includes 'Code', 'Pull requests', 'Actions' (which is selected), 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The main title is 'Create pipeline.yml #1'. On the left, there's a sidebar with 'Summary', 'Jobs' (where 'build' is selected and circled in red), 'Run details', 'Usage', and 'Workflow file'. The main content area shows the 'build' job which succeeded 3 minutes ago in 25s. It has a 'Search logs' bar and a refresh/cog icon. The job graph shows two steps: 'Set up job' and 'Run actions/checkout@v3' (which is also circled in red). The 'Run actions/checkout@v3' step expanded to show its sub-steps: 1. Run actions/checkout@v3, 14. Syncing repository: brentlaster/greetings-ci, 15. Getting Git version info, and 19. Temporarily overriding HOME='/home/runner/work/_temp/77610fae-ac20-4aea-8803530bce6e' before making global git config changes.

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 interface for the **greetings-ci** repository. On the left, there's a sidebar for switching branches/tags with a search bar and tabs for **Branches** and **Tags**. The **main** branch is the default. Below it, the **package** branch is selected. On the right, the repository structure is shown with a tree view. Under **.github/workflows**, the **publish-package.yml** file is highlighted. The code editor on the right displays the contents of this workflow file:

```

1 name: Publish package to GitHub Packages
2 on:
3   release:
4     types: [created]
5   workflow_dispatch:
6 jobs:
7   publish:
8     runs-on: ubuntu-latest
9     permissions:
10       contents: read
11       packages: write

```

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

The screenshot shows the GitHub interface for creating a pull request. At the top, the **Pull requests** tab is selected. In the center, a modal window is open for creating a new pull request. The modal shows the configuration: **base repository: skillrepos/greetings-ci**, **base: main**, **head repository: gwstudent/greetings-ci**, and **compare: main**. The **Create pull request** button is at the bottom right of the modal.

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 interface for comparing branches. At the top, there are dropdown menus for 'base: main' and 'compare: package'. A modal window titled 'Choose a head ref' is open, showing a search bar and a list of branches. The 'package' branch is selected. Below the modal, a message says 'There isn't anything to compare.' because both branches have the same commits.

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](#).

This screenshot shows a GitHub pull request. The 'base: main' and 'compare: package' dropdowns at the top are circled in red. The commit details below show 1 commit, 2 files changed, and 1 contributor. The commit log shows an addition to the file '.github/workflows/publish-package.yml' by Brent Laster on Jan 17. The diff view shows the added code:

```

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

```

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

This screenshot shows a GitHub pull request interface. The 'Pull requests' tab is active, showing one pull request. The commit details show 1 commit, 2 files changed, and 52 additions. The commit log shows an addition to the file '.github/workflows/publish-package.yml' by gwstudent. The 'Files changed' tab is selected, showing the diff for the added workflow file:

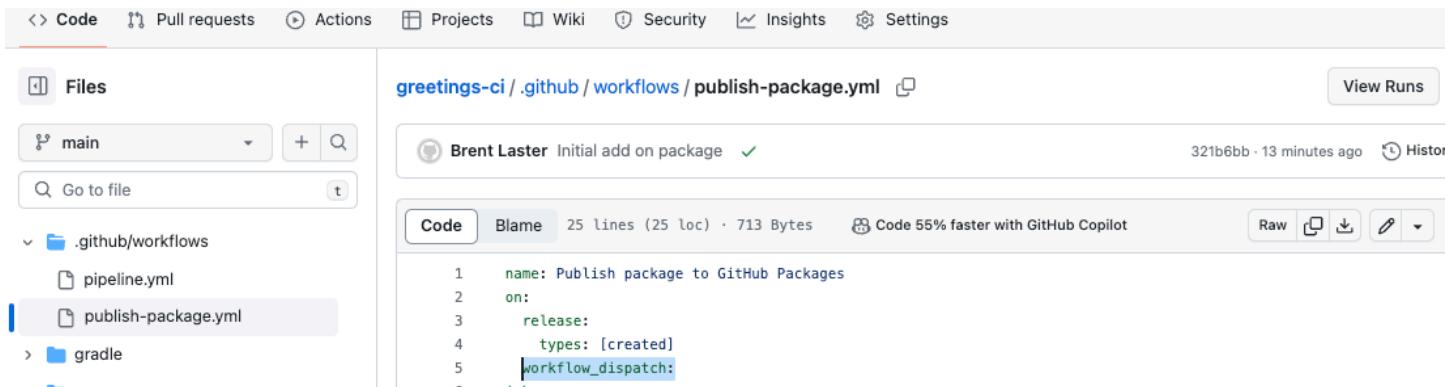
```

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

```

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. (No changes need to be made.)

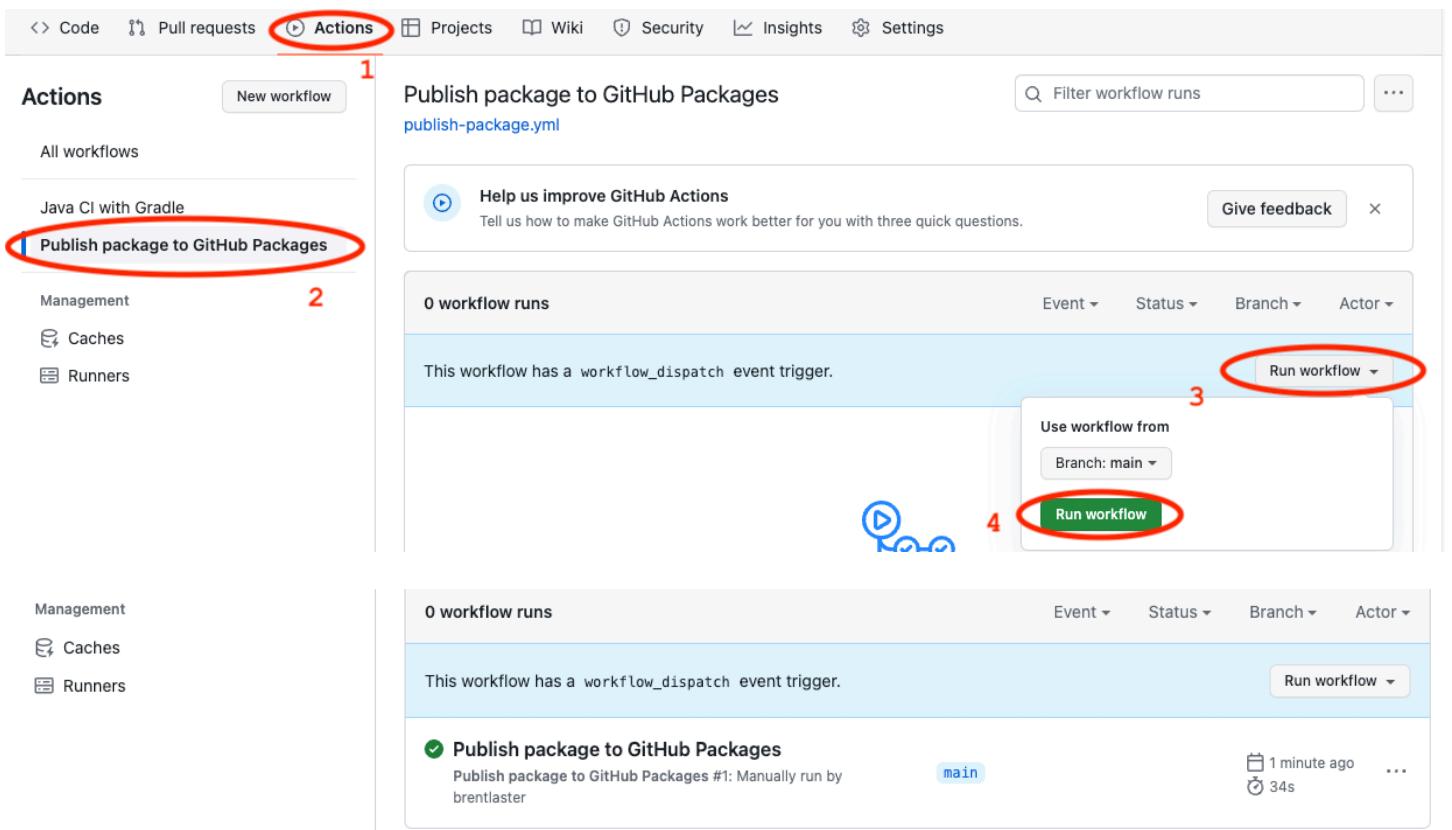


```

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.



Actions

1 Publish package to GitHub Packages

2 Management

3 Run workflow

4 Run workflow

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

The screenshot shows a GitHub repository page for 'greetings-ci'. The 'Code' tab is selected. In the bottom right corner of the main content area, there is a 'Packages' section with a single item: 'org.gradle.sample.greetings-ci'. This item 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 profile page for 'brentlaster'. The 'Packages' tab is selected. On the right, a dropdown menu titled 'Your profile' is open, showing a list of options. The 'Your packages' option is highlighted with a red circle and contains a single item: 'org.gradle.sample.greetings-ci'. This item is also 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. Do this for the **greetings-ci-1.1.jar** file.

org.gradle.sample.greetings-ci 1.1

1,006 B • Done

greetings-ci-1.1.jar
1,006 B • 1 minute ago

Full Download History

Details

brentlaster
January 05, 2024

Assets

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

END OF LAB

Lab 9: Creating a release

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

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

About

Simple starter repo for CI/CD with GitHub Actions

Activity

0 stars

0 watching

140 forks

Releases

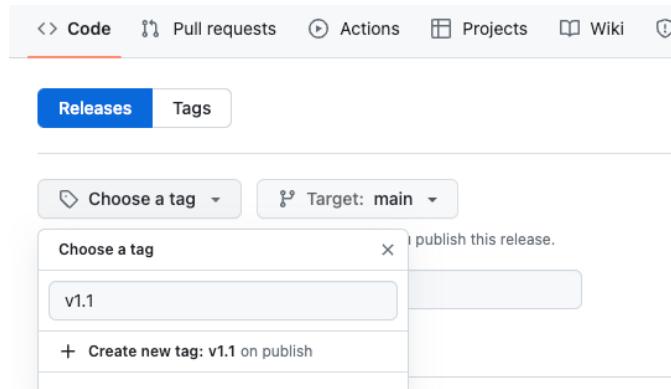
No releases published

Create a new release

Packages 1

org.gradle.sample.greetings-ci

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



3. 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.

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, a breadcrumb trail reads 'Releases / Initial release'. The main content area has a title 'Initial release' with a 'Latest' button. It shows a profile picture of the user 'brentlaster' and the message 'released this now'. A description below says 'Initial release of content for 1.1.'. Under the heading 'Assets', there is a table with three rows:

	greetings-ci-1.1.jar	1006 Bytes 2 minutes ago
	Source code (zip)	21 minutes ago
	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. The repository information shows it's 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 message: 'This branch is 2 commits ahead of, 3 commits behind skillrepos/greetings-ci:main.' It also lists recent commits and files. On the right side, there are sections for About, Activity, Stars, Forks, and a circled 'Releases' section. The 'Releases' section shows one entry: 'Initial release' (Latest) from 'brentlaster' 1 minute ago. This entry is circled in red.

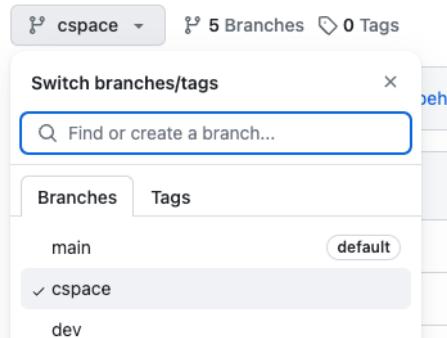
7. You can click on that if you want to see details about the release (same as output in step 5).

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 the same GitHub repository page. On the right side, there's a navigation bar with a 'Go to file' search bar, a 't' button, and a '1+' button followed by a red-highlighted '`<> Code`' button. Below this is a tab bar with 'Local' (selected) and 'Codespaces' (highlighted with a red circle). The main area shows a list of files: '.devcontainer', 'templates', '.gitignore', 'LICENSE', and 'README.md'. To the right, under the 'Codespaces' tab, it says 'No codespaces' and 'You don't have any codespaces with this repository checked out'. A red-highlighted button labeled 'Create codespace on cspace' is at the bottom of this section. A red number '3' is placed near this button.

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. (You can also dismiss any dialogs about linting.)

`calc.html`

```

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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2 COMMENTS

```

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

```

CODESPACES: friendly space robot cspace 0 0 2 Live Share 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.

`calc.html M`

```

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

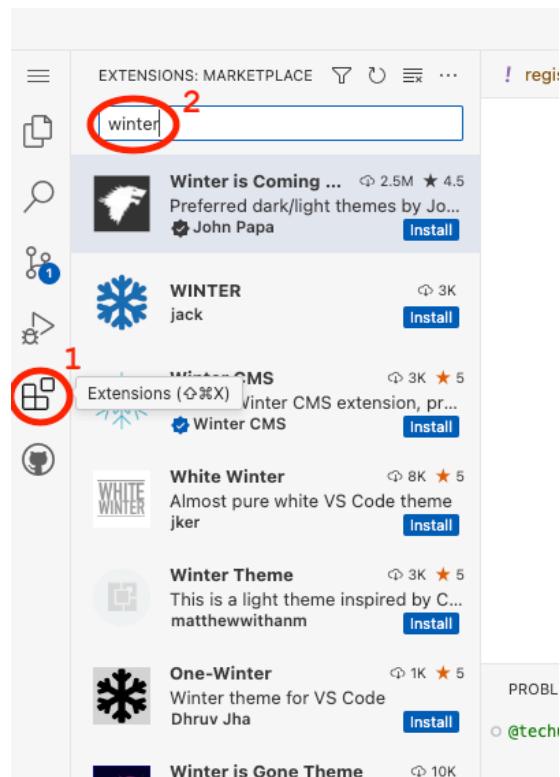
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2 COMMENTS

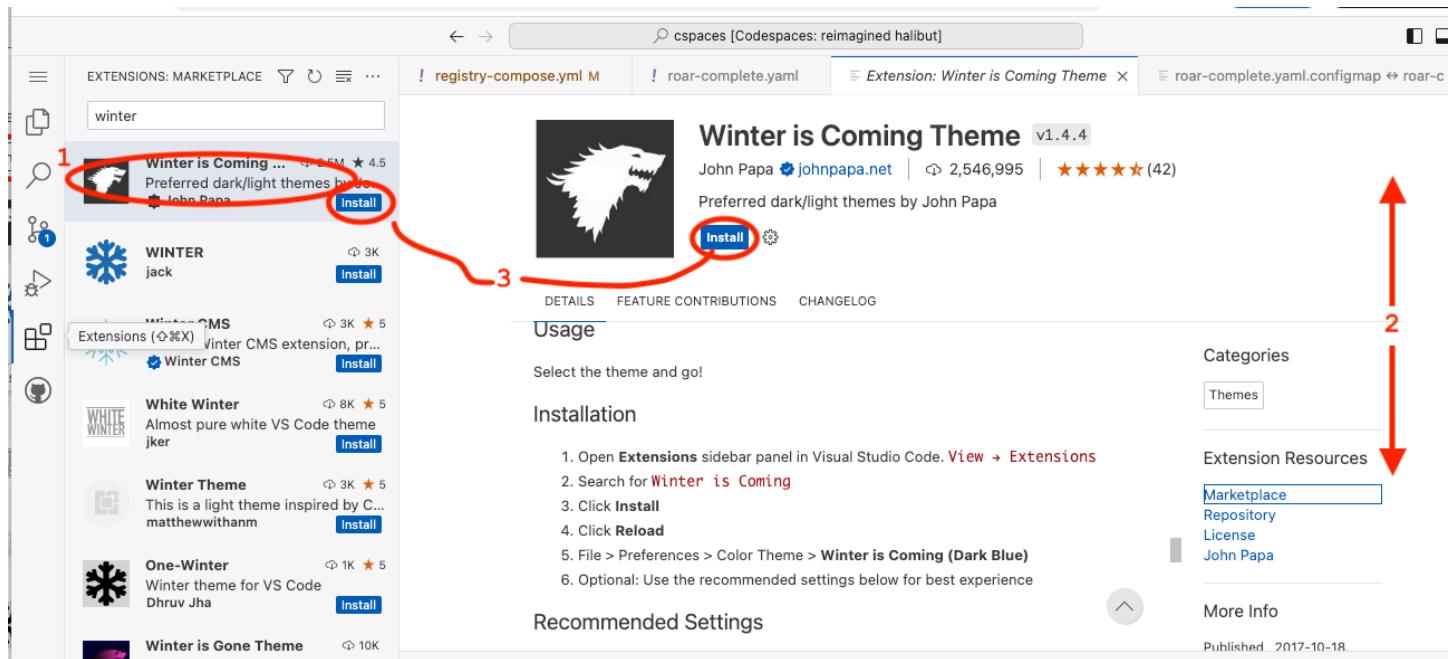
5. Now, in the Simple Browser pane, click the circular arrow icon to reload the webapp. You should see your change being displayed.

The screenshot shows the DevTools interface with two panes. The left pane displays the code for `calc.html`, specifically the `<body>` section which includes the line `<h2>Brent's Calc</h2>`. The right pane shows a browser window titled "Simple Browser" displaying the application "Brent's Calc". The page content includes instructions to enter numbers and select an operation, and a note about changing the theme via a dropdown.

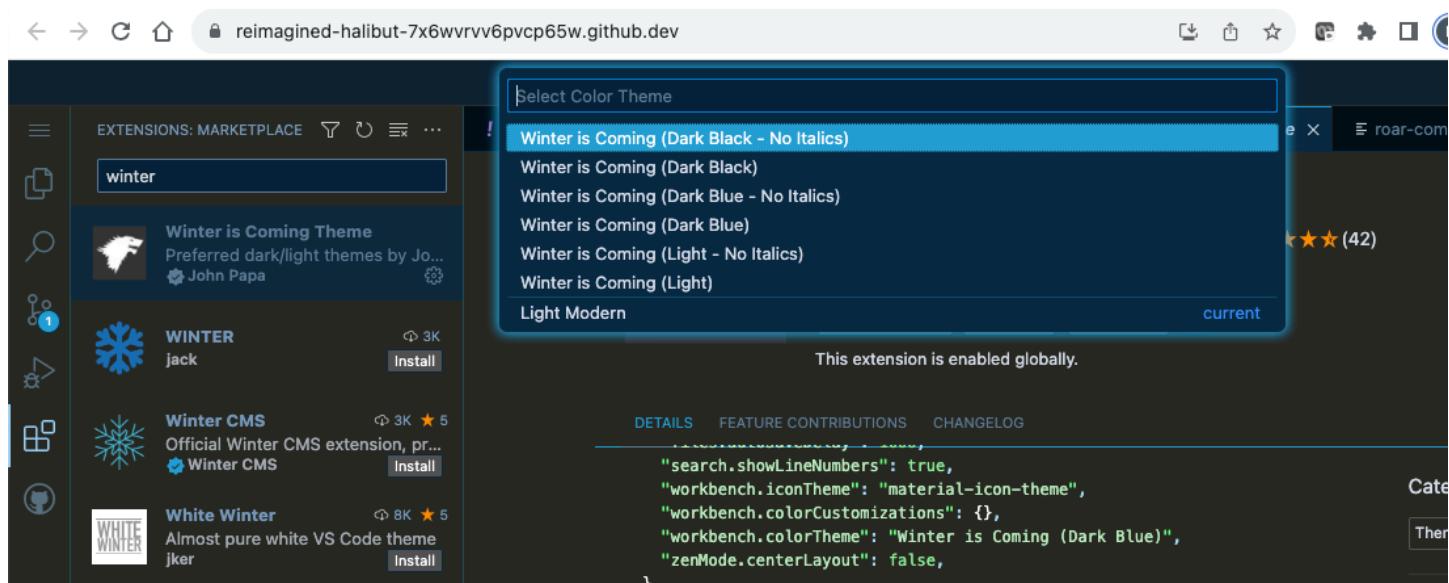
6. Next, let's see how to modify our codespace environment. We'll install an extension for the color theme. For practice, we'll use the "Winter is Coming" theme. Click on the *Extensions icon* (#1 in figure below), then in the *search bar* type in "winter" to quickly find the extension (#2).



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



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



END OF LAB

Lab 11: GitHub Command Line

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

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

\$ gh | more

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2 COMMENTS

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

USAGE
  gh <command> <subcommand> [flags]

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

GITHUB ACTIONS COMMANDS
  
```

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

\$ gh codespace list

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

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

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

\$ gh issue view 1

\$ gh issue view 1 --web

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

```
$ gh pr view 1
```

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

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

```
$ cd ..
```

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

OPTIONAL:

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

The screenshot shows the GitHub Codespaces interface. At the top, there's a search bar and various navigation icons. Below that, a sidebar on the left lists repositories: 'All' (1) and 'Templates' (1). Under 'By repository', there's a entry for 'gwstudent/calc'. The main area is titled 'Your codespaces' and shows 'Explore quick start templates' with options for 'Blank', 'React', and 'Jupyter Notebook'. Below that, a section titled 'Owned by gwstudent' shows a single codespace named 'friendly goldfish'. To the right of this list is a context menu for the 'friendly goldfish' codespace. The menu includes options like 'Rename', 'Export changes to a branch', 'Change machine type', 'Stop codespace', 'Auto-delete codespace' (with a checkmark), 'Open in Browser', 'Open in Visual Studio Code', 'Open in JetBrains Gateway', 'Open in JupyterLab', and 'Delete'. The 'Auto-delete codespace' option is highlighted with a red circle.

END OF LAB

Demo: Copilot

That's all - THANKS!

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

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

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

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

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

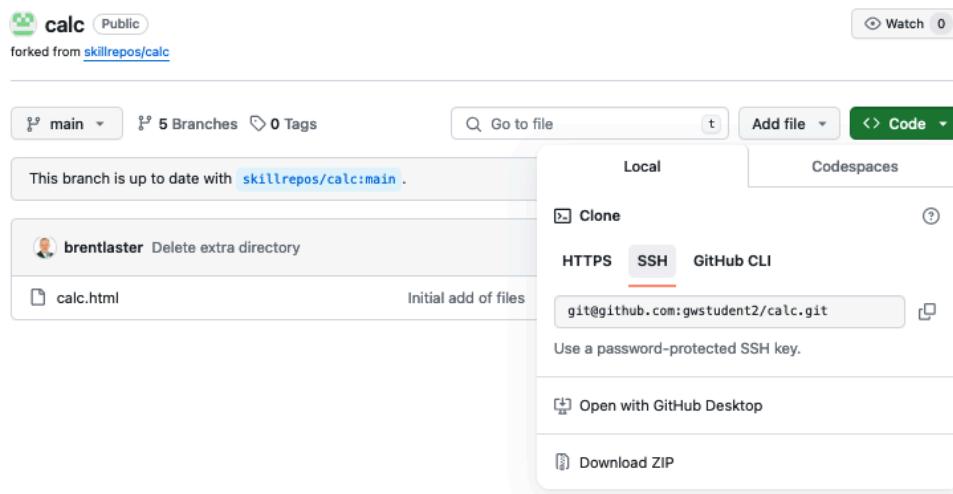
(Note: If you prefer to disable the global credentials helper entirely, you can try

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

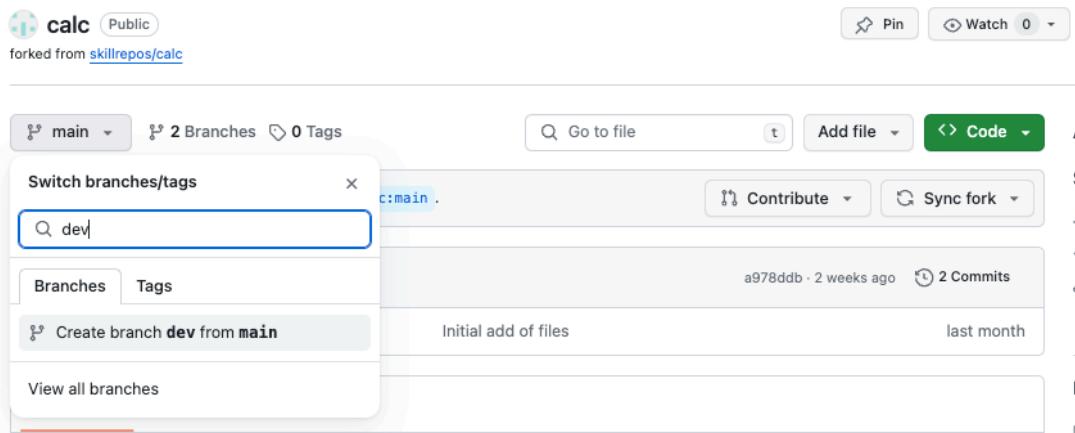
This may or may not work depending on if you have access to do this.)

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

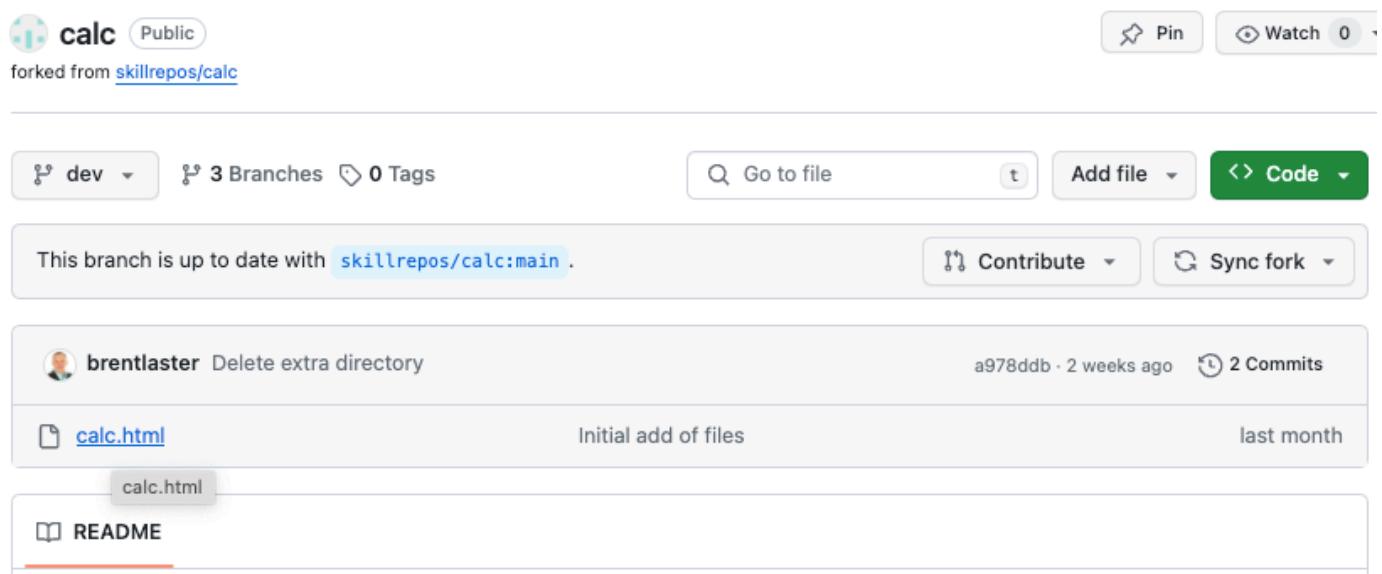
If you go this route, when you get the remote URL from the browser, select the SSH tab.



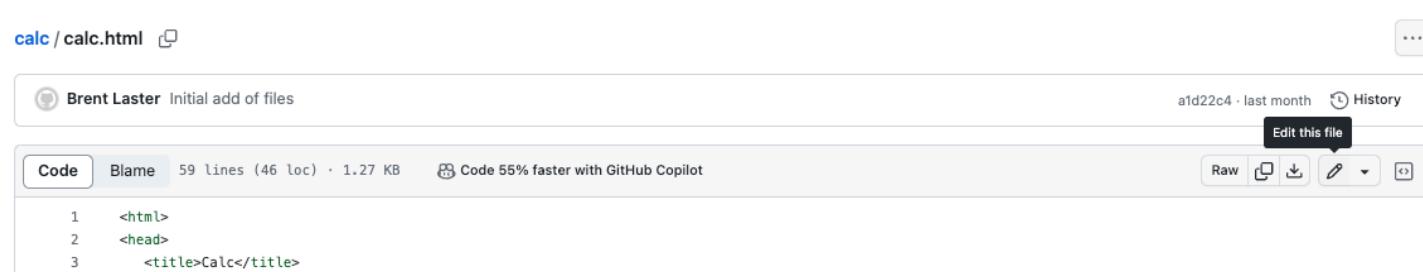
C. Commit directly in GitHub: Another option is to commit directly to GitHub in the browser. To do this, first create a *dev* branch in the repo. Click on the branch dropdown under the title of the repo. In the *Find or create a branch* field, type **dev**. Then click on **Create branch dev from main**.



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



Click on the pencil icon to edit the file.



Make the changes noted in Lab 1 in the file.

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

The screenshot shows a GitHub repository named 'gwstudent / calc'. The 'Code' tab is selected, showing the file 'calc.html' in the 'dev' branch. The code editor displays the following HTML and JavaScript:

```
1 <html>
2 <head>
3     <title>Brent's Calc</title>
4
5     <script language="javascript">
6
7         var plus,minus,divide,multiply;
8
9         function initialize(){
10             plus=document.calc.operate[0];
11             minus=document.calc.operate[1];
12             divide=document.calc.operate[2];
13             multiply=document.calc.operate[3];
14         }
15
16         function calculate(){
17             a = parseInt(document.calc.value[0]);
18             b = parseInt(document.calc.value[1]);
19             if (plus.selected)
20                 document.calc.answer.value=a+b;
21             if (minus.selected)
22                 document.calc.answer.value=a-b;
23             if (divide.selected)
24                 document.calc.answer.value=a/b;
25             if (multiply.selected)
26                 document.calc.answer.value=a*b;
27         }

```

A modal dialog titled 'Commit changes' is open over the code editor. It contains the following fields:

- Commit message:** A text input field containing the placeholder 'Update calc.html'.
- Extended description:** A large text area with the placeholder 'Add an optional extended description..'
- Branch selection:** Two radio buttons:
 - Commit directly to the dev branch
 - Create a new branch for this commit and start a pull request
Learn more about pull requests
- Buttons:** 'Cancel' and 'Commit changes' (the latter is a green button).