

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

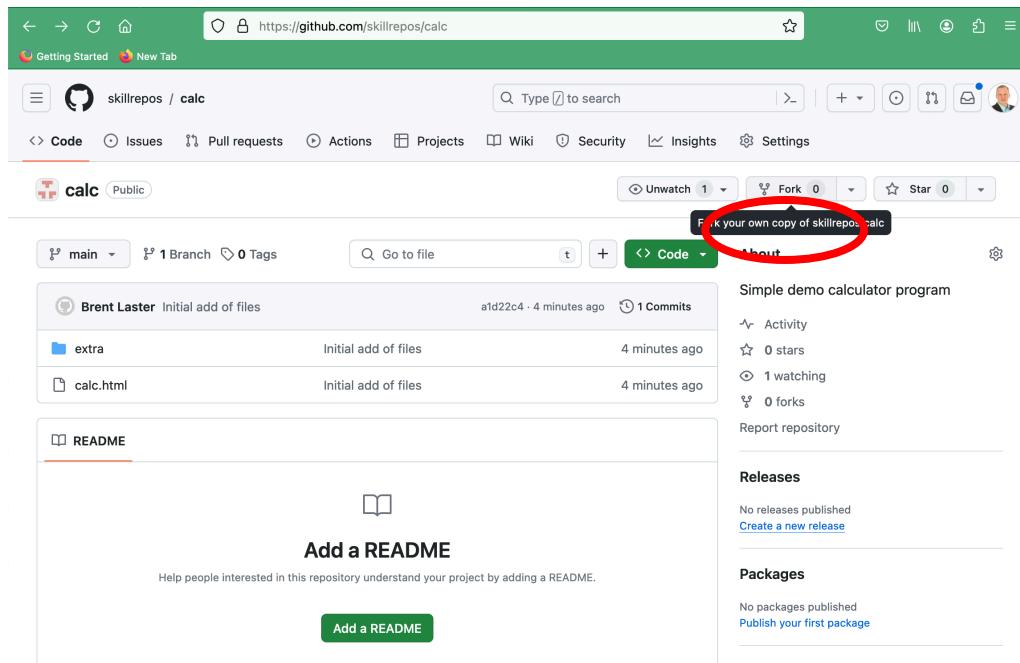
Setup and prerequisites

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.)



Create a new fork

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

Required fields are marked with an asterisk (*).

Owner * Repository name *

brentlaster / calc

calc is available.

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

Description (optional)

Simple demo calculator program

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

You are creating a fork in your personal account.

Create fork

- Now you'll be on your fork of the repo. Next, let's clone your repo down to your local system so we can make changes there. In your project, ensure you are on the **Code** tab, then click on the large green **<> Code** button. In the **Local** tab, select **HTTPS** under Clone and then click on the **copy icon** to copy your project's URL.

brentlaster / calc

Code Pull requests Actions Projects Wiki Security Insights Settings

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

main 1 Branch 0 Tags Go to file

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

Brent Laster Initial add of files

extra Initial commit
calc.html Initial commit

README

Local Codespaces

Clone

HTTPS SSH GitHub CLI

4 https://github.com/brentlaster/calc.git
5 Use Git or checkout with SVN using the web URL

Copy url to clipboard

Open with GitHub Desktop

Download ZIP

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

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

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

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

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

```
$ git remote -v
```

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

Edit calc.html and change

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

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

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

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

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

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

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

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

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

New personal access token (classic)

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

Note

GitHub Fundamentals class

What's this token for?

Expiration *

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

Select scopes

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

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

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

read:ssh_signing_key Read public user SSH signing keys

Generate token **Cancel**

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

Personal access tokens (classic)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again.

Copied!

✓ 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 ✓

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

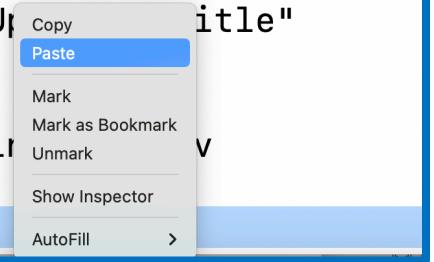
```
$ git push -u origin main:dev
```

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



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

```
developer@Bs-MacBook-Pro calc % vi calc.html
developer@Bs-MacBook-Pro calc % git commit -am "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':
```



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 interface. On the left, a sidebar titled "Switch branches/tags" has a search bar "Find or create a branch..." and tabs for "Branches" (selected) and "Tags". Under "Branches", there are three entries: "main" (labeled "default"), "cspace", and "dev" (marked with a checkmark). Below this is a link "View all branches". On the right, the main area shows a commit history for the "calc:main" branch. The first commit, "Updating title", was made by "Brent Laster" at 61e42da · 8 hours ago, with 3 Commits. A "Sync fork" button is visible above the commit list.

This screenshot shows a GitHub repository interface focusing on the "calc.html" file. The top navigation bar includes "Code", "Pull requests", "Actions", "Projects", "Wiki", "Security", "Insights", and "Settings". The main area displays the file content with a "Blame" button highlighted with a red circle. The blame history shows one commit from "Brent Laster" updating the title. The commit details are: "Updating title" at 61e42da · 8 hours ago, with 3 Commits. The blame history table shows the commit details and the corresponding code changes.

Date	Author	Message	Time Ago	Commits
last week	Brent Laster	Initial add of files	22 minutes ago	1 Commit
22 minutes ago	Brent Laster	Updating title	22 minutes ago	1 Commit
last week	Brent Laster	Initial add of files	22 minutes ago	1 Commit

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.

This screenshot shows the blame history for the "calc.html" file. The "Blame" button is circled in red. The blame history table shows the following details:

Date	Author	Message	Time Ago	Commits
last week	Brent Laster	Initial add of files	22 minutes ago	1 Commit
22 minutes ago	Brent Laster	Updating title	22 minutes ago	1 Commit
last week	Brent Laster	Initial add of files	22 minutes ago	1 Commit

The blame history table shows the commit details and the corresponding code changes. The blame history table shows the commit details and the corresponding code changes.

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` in the `dev` branch. The commit message is `Updating title`, made by `Brent Laster` 24 minutes ago. The commit hash is `d9e79db`. The interface includes standard GitHub navigation links like Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A red circle highlights the `History` button in the top right corner of the commit card.

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 expanded commit history for `Updating title`. The commit was made by `Brent Laster` 26 minutes ago. The commit hash is `d9e79db`. The commit message is `Updating title`. The commit is dated Dec 31, 2023. The interface includes filters for All users and All time. The commit is shown in a detailed view with a copy and share icon.

The screenshot shows a GitHub commit page for a file named 'calc.html'. The commit was made by 'Brent Laster' 28 minutes ago. It has 2 additions and 2 deletions. The diff shows the following changes:

```

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

```

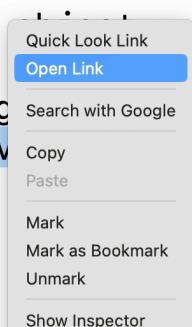
5. Let's now merge our change from the dev branch to main via a pull request. **Switch back to the terminal where you did the commit and push.**

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

```

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

```



-- OR --

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

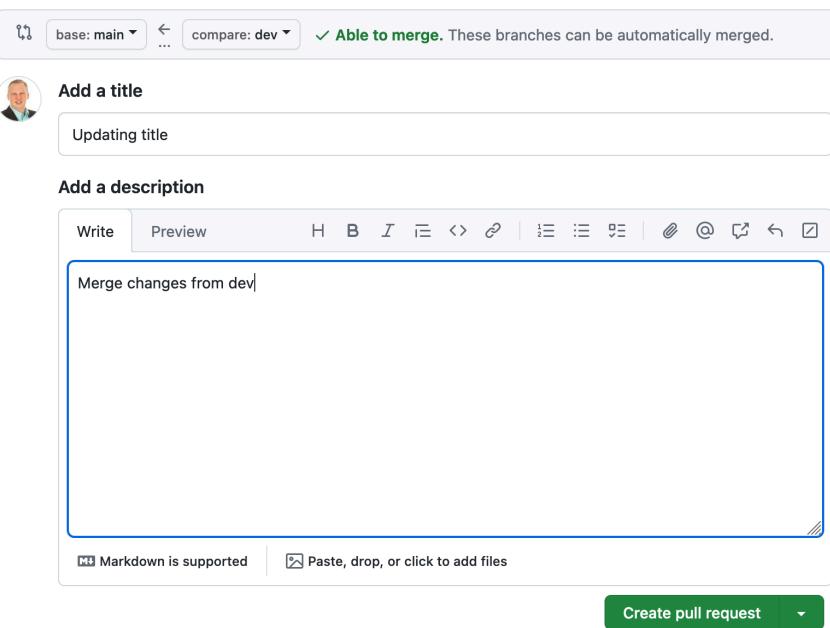
Comparing changes

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

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](#)

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



9. At this point, you have created a new pull request. (Note that the *Pull Requests* tab at the top shows 1 pull request in the repo.) It will check for any conflicts for merging.

We haven't set up any CI processes or reviewers so there is nothing for those sections. Note the check in the middle section that says *This branch has no conflicts with the base branch*. You can look at the *Commits* or *File Changed* tabs if you want to see more details on the changes.

10. When you're ready, switch back to the ***Conversation*** tab. Then click on the ***Merge pull request*** button and then the ***Confirm merge*** button to complete the pull request. After that, the pull request will be completed and closed (shown in second screenshot). Afterwards, you can click on the button to delete the ***dev*** branch if you want.

The screenshot shows a GitHub pull request interface. At the top, there's a green button labeled "Open" and a title "Updating title #1". Below the title, it says "brentlaster wants to merge 1 commit into `main` from `dev`". A commit message "Updating title" is shown with a diff preview icon. The commit hash is `d9e79db` and there are "No changes" listed. To the right, there are columns for "Labels" (None), "Project" (None), "Milestone" (None), and "Notifications" (None).

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

Below the modal, there's a "Add a comment" button and a purple "Merged" button. The commit message "Updating title" is shown again with a smiley face icon. The commit hash is `d9e79db`. Below the commit, a message from "brentlaster" says "merged commit `dc98b08` into `main` now". There's a "Revert" button next to this message.

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." There's also a "Delete branch" button.

END OF LAB

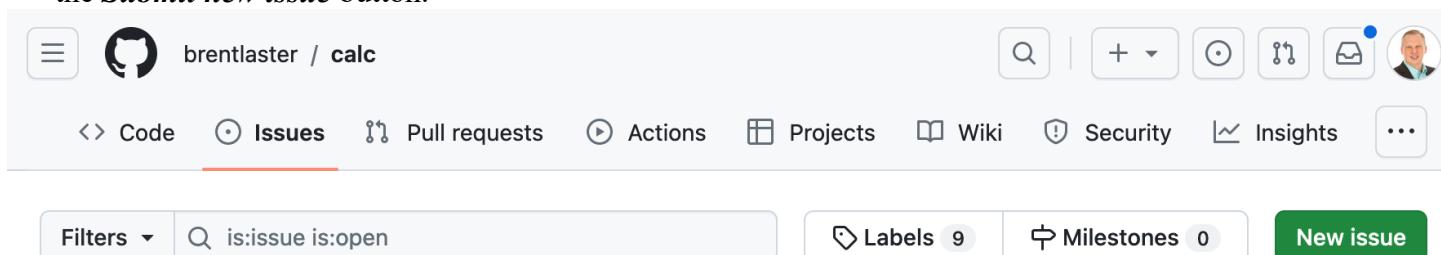
Lab 3: Creating GitHub issues

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**.



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



Add a title

Needs README

Add a description

Please add README file :book:

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)

Code Issues 1 Pull requests Actions Projects Wiki

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 the GitHub interface for creating a new issue. At the top, there are tabs for Wiki, Security, Insights, and a three-dot menu. Below the tabs are two buttons: 'Edit' and 'New issue'. The main area has a title 'Issue' and a description field. To the left, there's a sidebar with sections for 'Assignees' and 'Labels'. The 'Assignees' section shows a placeholder 'No one—[assign yourself](#)'. There are three dots at the top of the sidebar.

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

This screenshot shows the 'Labels' section of the GitHub issue editor. It includes a 'Labels' heading, a 'Filter labels' input field, and a list of available labels. One label, 'documentation', is selected and highlighted with a blue circle and a checkmark. Other labels listed are 'duplicate' (unchecked) and 'Something isn't working' (unchecked). A note below the 'duplicate' label states: 'This issue or pull request already exists.'

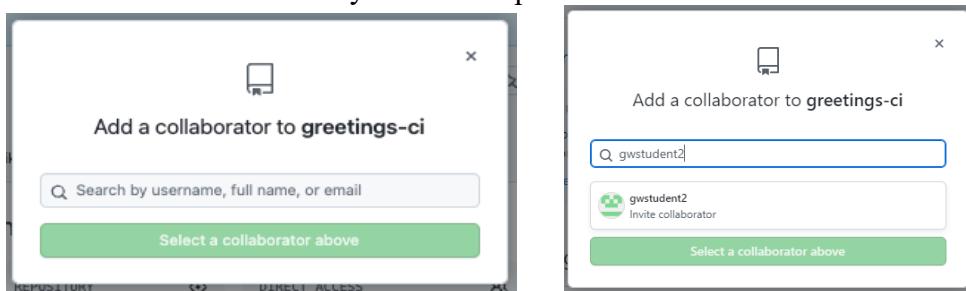
6. After this, if you click on the **Issues** tab at the top, and look at your issue, it should look like the following.

This screenshot shows the 'Issues' tab of a GitHub repository named 'brentlaster / calc'. The repository has 9 labels and 0 milestones. The search bar contains 'Type ⌘ to search'. The filter bar shows 'is:issue is:open' and '1 Open'. The list of issues includes one item: '#2 opened 15 hours ago by brentlaster' with the label 'Needs README documentation'.

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 a GitHub repository settings page. On the left, there's a sidebar with sections like General, Access (selected), Collaborators, Moderation options, Code and automation, Security, and others. The main area shows two boxes: 'PUBLIC REPOSITORY' (This repository is public and visible to anyone.) and 'DIRECT ACCESS' (0 collaborators have access to this repository. Only you can contribute to this repository.). A 'Manage' button is located below the PUBLIC REPOSITORY box.

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



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

The screenshot shows a GitHub repository page for 'brentlaster / calc'. The top navigation bar includes Code, Issues (1), Pull requests, Actions, Projects, Security, Insights, and a Watch 0 button. The repository details show it's Public and forked from 'skillrepos/calculator'. A message at the bottom says '@brentlaster has invited you to collaborate on this repository' with a 'View invitation' button circled in red.

The screenshot shows a GitHub invitation interface and a repository view for the 'calc' repository.

Invitation Interface:

- At the top, there are two user icons: a man in a suit and a green GitHub logo.
- The text "brentlaster invited you to collaborate" is displayed.
- A red oval highlights the "Accept invitation" button, which is green with white text.
- Below the button, a note states: "Owners of calc will be able to see:" followed by a bulleted list:
 - Your public profile information
 - Certain activity within this repository
 - Country of request origin
 - Your access level for this repository
 - Your IP address
- A link "Is this user sending spam or malicious content? Block brentlaster" is present.

Repository View:

- The repository name is "brentlaster / calc".
- The navigation bar includes "Code", "Issues 1", "Pull requests", "Actions", and "Projects".
- A message at the bottom of the repository view says "You now have push access to the brentlaster/calc repository." with a blue "X" icon.

END OF LAB

Lab 4: Setting up a pull request with reviewers

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

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

The screenshot shows the GitHub interface for the 'calc' repository. The top navigation bar includes tabs for Code, Issues (1), Pull requests, Actions, Projects, Wiki, and Settings. Below the header, it displays the repository details: 'calc' (Public, forked from skillrepos/calc). The main content area shows a list of commits: 'brentlaster Merge pull request #1 from br...' (dc98b08 · yesterday, 3 Commits), 'extra' (Initial add of files, last week), and 'calc.html' (Updating title, yesterday). The 'README' section is visible, showing a placeholder icon and the text 'Add a README'. A red oval highlights the green 'Add a README' button.

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

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

Code Issues Pull requests Actions Projects Wiki Security Insights

calc / README.md in main

Cancel changes Commit changes...

Edit Preview Spaces 2 Soft wrap

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

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

Code Issues Pull requests Actions Projects Wiki Security Insights

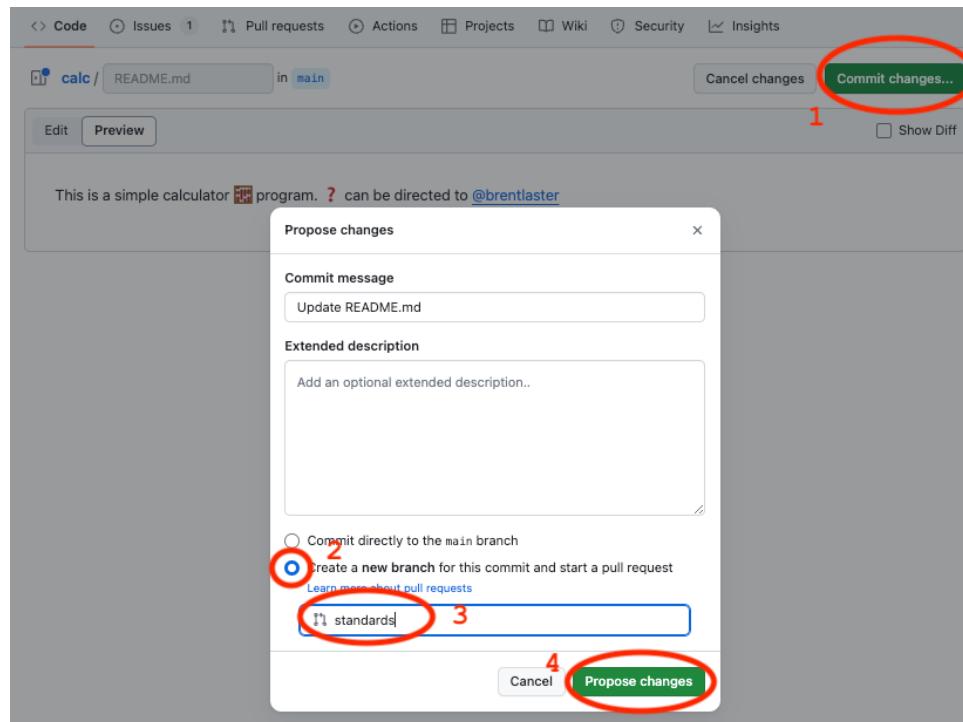
calc / README.md in main

Cancel changes Commit changes...

Edit Preview Show Diff

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

4. Now let's commit these changes to a new branch and open a pull request to merge them. click on the green **Commit changes...** button in the upper right corner. In the dialog, enter a 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 are navigation links: Code, Issues (1), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, 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 header, it says "Discuss and review the changes in this comparison with others. Learn about pull requests". Underneath, it shows "1 commit", "1 file changed", and "1 contributor". A detailed view of the commit "Create README.md" by "brentlaster committed now" is shown, with a "Verified" badge and a link to the commit hash "47b11e1". A diff viewer shows a single addition to README.md: "+ This is a simple calculator :abacus: program. :question: can be directed to [@brentlaster] (<https://github.com/brentlaster>)".

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. It has 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 is a "Reviewers" section with a "Request up to 15 reviewers" input field and a "Type or choose a user" dropdown. A user named "gwstudent2" is selected and highlighted with a red circle. Other sections include "Labels" (with a note that none are yet), "Projects" (none yet), "Milestone" (no milestone), and "Development" (a note about closing keywords). At the bottom, there is a "Create pull request" button and a "Helpful resources" section.

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

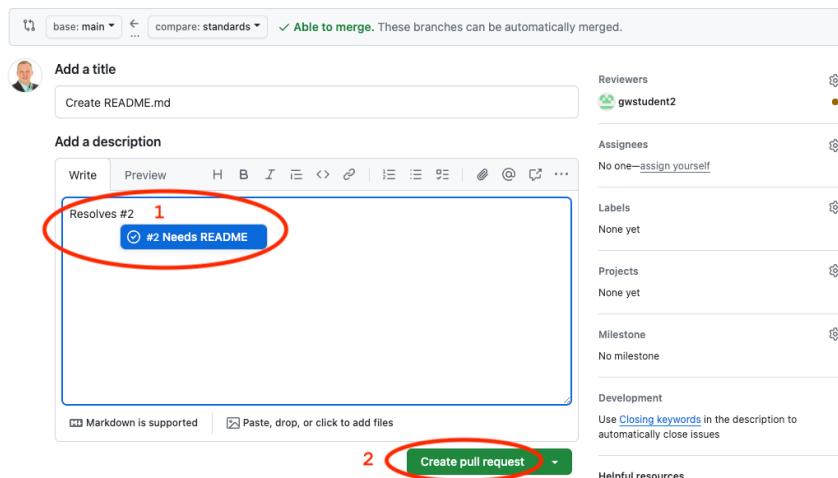
Resolves #2

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.

Inbox 1

All Unread is:unread Group by: Date

Saved Done

Filters Assigned Participating Mentioned Team mentioned Review requested

brentlaster/calc Create README.md +1 review requested brentlaster ✓

Select all brentlaster/calc #4 Create README.md

ProTip! Create custom filters to quickly access your most important notifications.

1-1 of 1 Prev Next

Repositories brentlaster/calc 1

- OR -

https://github.com/pulls/review-requested

Pull Requests

Created Assigned Mentioned Review requests

1 Open 1 Closed

brentlaster/calc Create README.md #4 opened 14 minutes ago by brentlaster

ProTip! Add no:assignee to see everything that's not assigned.

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

brentlaster / calc

Pull requests 1

brentlaster requested your review on this pull request. Add your review

Create README.md #4

1 Open brentlaster wants to merge 1 commit into main from standards

Conversation 0 Commits 1 Checks 0 Files changed 1

brentlaster commented 18 minutes ago Owner ...

Reviewers 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”.

The screenshot shows the GitHub pull request review interface for a pull request titled "Create README.md #4". The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and more. The pull request summary indicates "brentlaster wants to merge 1 commit into main from standards". Below this, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A red circle labeled "1" highlights the "Review changes" button. The main area is titled "Finish your review" and contains a text editor with a toolbar. A red circle labeled "2" highlights the comment "Suggest adding a license file too.". Below the editor, there are three radio button options: "Comment", "Approve", and "Request changes". A red circle labeled "3" highlights the "Approve" option. At the bottom right, a red circle labeled "4" highlights the "Submit review" button.

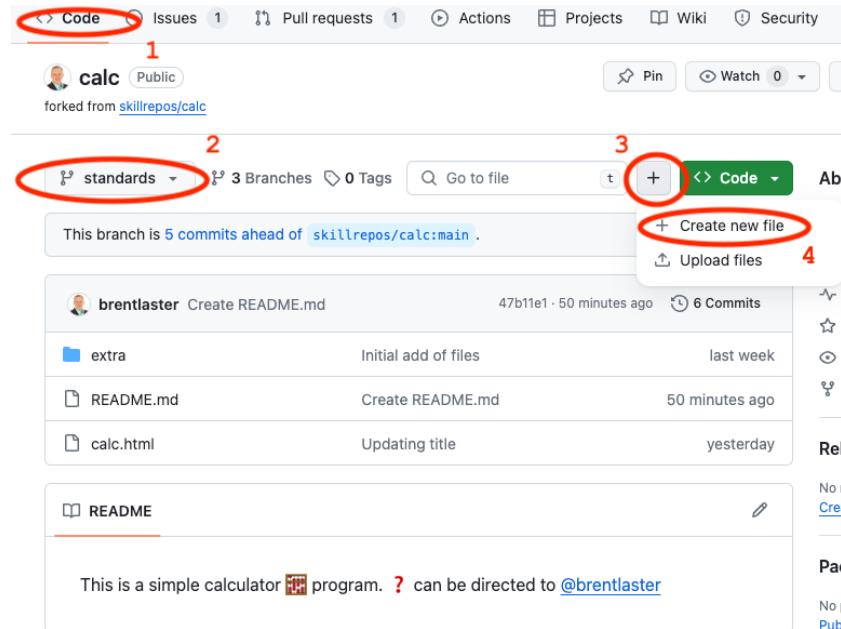
4. Go to the session with your original GitHub userid or log out of the other one and log back in if you need to. Go to the **Pull requests** menu at the top, find the pull request and click on the commit message. Then you should see a screen like below.

The screenshot shows the GitHub pull request commit history for the same pull request. It lists the following commits:

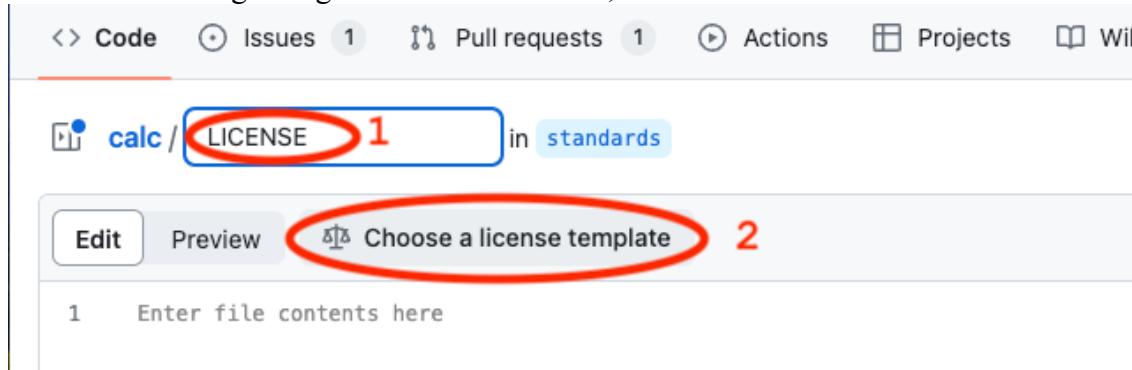
- brentlaster commented 41 minutes ago: No description provided.
- Create README.md (Verified) 47b11e1
- brentlaster requested a review from gwstudent2 41 minutes ago
- gwstudent2 approved these changes 1 minute ago View reviewed changes
- gwstudent2 left a comment Suggest adding a license file too.

5. Since there was a suggestion to add a license file, that sounds like a good idea, so let's do that. Click on the **Code** tab at the top, then select the **standards** branch (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			
GNU General Public License v3.0	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.	
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		

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

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

Cancel 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. At the top, there's a navigation bar with tabs like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Pull requests' tab is highlighted with a red circle. Below the navigation bar, there's a summary message: 'brentlaster wants to merge 2 commits into main from standards'. Underneath this, there are three tabs: 'Conversation' (with 1 item), 'Commits' (with 2 items, circled in red), 'Checks' (with 1 item), and 'Files changed' (with 2 items, 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 note about a calculator program. At the bottom right of the main content area, there are buttons for 'Review in codespace' and 'Review changes'.

- Click back on the **Conversation** tab in the pull request and go ahead and merge (*Merge pull request*) and close (*Confirm merge*) the pull request. After completing the merge, you should be able to click on the **Issues** tab and see that your issue has been automatically closed. You can click on the **Closed** list and then open the issue to see the automatically generated log of comments and actions if you want.

The screenshot shows a GitHub 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. The activity log on the left shows the following events:

- brentlaster commented 19 hours ago: 'Please add README file :wrench:'
- 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' (status: Merged)
- brentlaster mentioned this issue 15 minutes ago: 'Update README.md #5' (status: Merged)
- brentlaster closed this as completed in #5 14 minutes ago

On the right side of the screen, there are several sections for managing the issue:

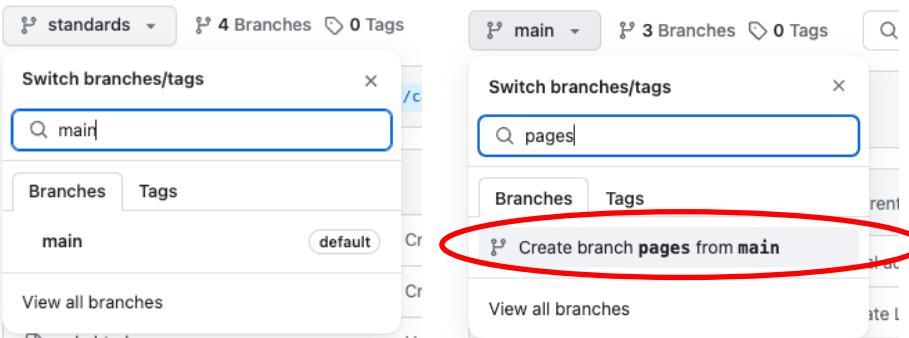
- Assignees:** brentlaster
- Labels:** documentation
- Projects:** None yet
- Milestone:** No milestone
- Development:** Successfully merging a pull request may close this issue.
 - Create README.md: brentlaster/calc
 - Update README.md: brentlaster/calc
 - Create README.md: brentlaster/calc
- Notifications:** Customize, Unsubscribe (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.

- 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.



- 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](#)

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

The screenshot shows the GitHub repository setup interface. At the top, there's a green button labeled "Create repository". Below it, the repository name "brentlaster / brentlaster.github.io" is shown. 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 two main boxes: "Start coding with Codespaces" (with a "Create a codespace" button) and "Add collaborators to this repository" (with an "Invite collaborators" button). Below this, a blue banner provides instructions: "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.

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

The screenshot shows the GitHub "Commit changes" dialog. At the top, there's a large area with a red circle labeled "1" containing the text "Drag additional files here to add them to your repository" and "Or choose your files". Below this, a file named "calc.html" is listed with a red circle labeled "2" around its icon. At the bottom, there's a "Commit changes" button with a red circle labeled "3" around it. A "Cancel" button is also visible.

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.

The screenshot shows the GitHub Code tab for the repository `brentlaster / brentlaster.github.io`. The left sidebar shows a tree view with `main` selected. In the main area, a file named `calc.html` is listed under `brentlaster.github.io / calc.html`. A commit message from `brentlaster` is shown: "Add files via upload". Below the commit, the file content is displayed:

```

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

```

The "Edit this file" button in the toolbar is circled in red.

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.

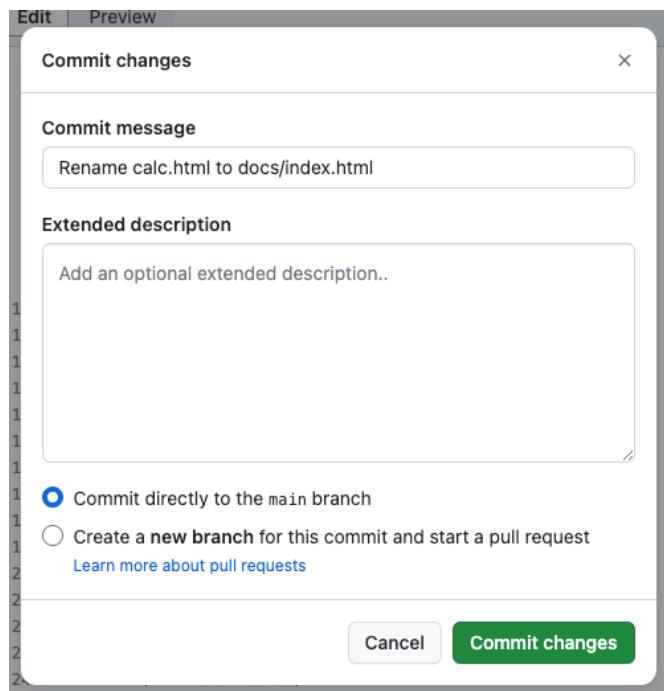
The image contains two screenshots of the GitHub edit dialog. The top screenshot shows the file path `brentlaster.github.io / calc.html` highlighted with a red oval. The bottom screenshot shows the file path `brentlaster.github.io / docs / index.html` highlighted with a red oval. Both screenshots show the "Edit" tab selected and the file content:

```

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.



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](#) [...](#)

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:

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**.

[Unwatch 1](#) [Fork 0](#) [Star 0](#)

[Code](#) [About](#)

Code repo for the web page for the calc code
4 Commits

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.

Edit repository details

Description
Simple web calculator program

Website
<https://brentlaster.github.io/>

Use your GitHub Pages website

Topics (separate with spaces)
calculator example-code

Include in the home page
 Releases
 Packages
 Deployments

[Cancel](#) [Save changes](#)

About

Simple web calculator program

[brentlaster.github.io/](#)

calculator example-code

Activity

0 stars
1 watching
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 screenshot shows the GitHub repository 'greetings-ci' with the following details:

- Branches:** main (selected), 2 Branches, 0 Tags
- Commits:** brentlaster Delete extra directory (c32ca8a · 3 days ago), 22 Commits
- Activity:** Initial add (2 years ago)
- Actions:** Fork 139, Star 0
- About:** Simple starter repo for CI/CD with GitHub Actions

The screenshot shows the 'Create a new fork' page for the 'greetings-ci' repository:

- Owner:** brentlaster
- Repository name:** greetings-ci (highlighted with a yellow box)
- Description (optional):** Simple starter repo for CI/CD with GitHub Actions
- Checkboxes:**
 - Copy the main branch only** (highlighted with a red circle)
 - Contribute back to skillrepos/greetings-ci by adding your own branch. [Learn more.](#)
- Information:** You are creating a fork in your personal account.
- Create fork button:** A green button with a red oval around it.

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 main content area displays a list of recent commits from 'Brent Laster' with commit messages like 'add extra dir', 'Initial add', and 'Initial add'. The right sidebar provides repository statistics: 0 stars, 0 watching, and 1 fork. It also includes sections for 'About', 'Releases', and 'Packages'.

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

The screenshot shows the 'Browse all categories' section of the GitHub Actions catalog. The 'Continuous integration' category is circled in red. Other categories visible include 'Automation', 'Stale', 'Manual workflow', and 'Labeler'. Below this, there is a search bar and a link to 'View all categories'.

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

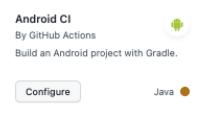
Continuous integration

Deployment

Security

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

By GitHub Actions

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

[Configure](#)

[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

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 with Gradle

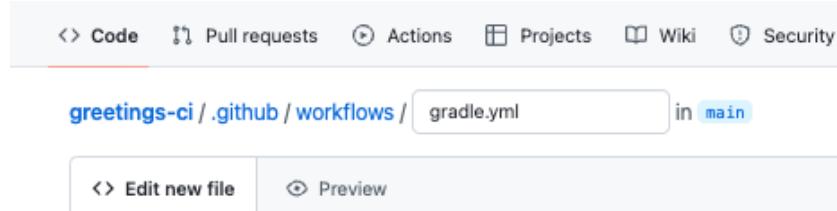
By GitHub Actions

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

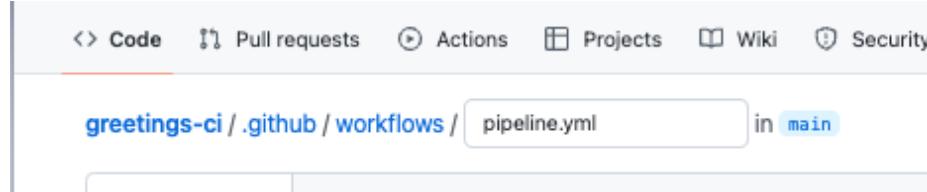
[Configure](#)

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.)



TO



- 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

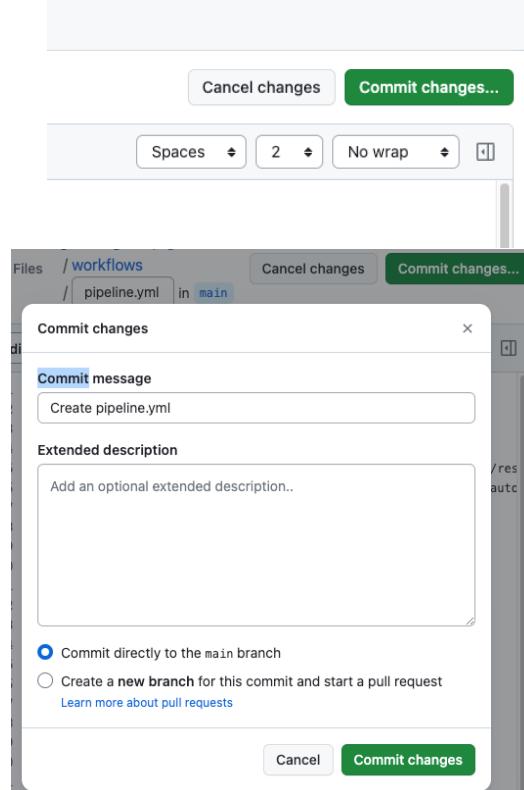
trouble just selecting that code, try starting at the bottom and selecting/highlighting from the bottom up.) The code to be deleted is highlighted in the next screenshot.

```

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       highlight/select, and delete
60       with:
61         java-version: '17'
62         distribution: 'temurin'
63
64   # Generates and submits a dependency graph, enabling Dependabot Alerts for all project dependencies.
65   # See: https://github.com/gradle/actions/blob/main/dependency-submission/README.md
66   - name: Generate and submit dependency graph
67     uses: gradle/actions/dependency-submission@417ae3ccd767c252f5661f1ace9f835f9654f2b5 # v3.1.0
68

```

- 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 interface. The top navigation bar has tabs for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Actions tab is selected. On the left, there's a sidebar with sections for Actions, All workflows, Java CI with Gradle, Management, Caches, Runners, and a Beta button. The main area is titled 'All workflows' and shows '1 workflow run'. A card for 'Create pipeline.yml' is displayed, indicating it was triggered by a commit from the 'main' branch, pushed by brentlaster, and is currently 'In progress'.

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.

This screenshot shows the same GitHub Actions interface after the workflow has completed successfully. The 'Create pipeline.yml' run is now marked as 'Succeeded' with a green checkmark icon. The time of completion is listed as '1 minute ago'.

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

The screenshot shows the detailed view of the 'Create pipeline.yml #1' job. The left sidebar lists 'Summary', 'Jobs' (with 'build' selected), 'Run details', 'Usage', and 'Workflow file'. The main area is titled 'build' and shows a log of the build process. The log includes steps like 'Set up job' and 'Run actions/checkout@v3'. The 'Run actions/checkout@v3' step is expanded, showing sub-steps 1 through 19. A red circle highlights the 'build' link in the sidebar, and another red circle highlights the expanded '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 interface for the **greetings-ci** repository. The repository is public and forked from [skillrepos/greetings-ci](#). The main branch is **main**, and there is a **package** branch. The **package** branch has an updated **build.gradle** file and a new Actions workflow file (**.github/workflows/publish-package.yml**). The workflow file content is as follows:

```

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. 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 interface. A new pull request is being created to merge the **package** branch into the **main** branch. The base repository is set to **skillrepos/greetings-ci** and the head repository is set to **gwstudent/greetings-ci**. The compare dropdown is set to **main**. The pull request summary indicates it is automatically merged. The **Create pull request** button is visible at the bottom right.

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](#).

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](#).

Able to merge. These branches can be automatically merged.

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

Commits on Jan 17, 2024

Initial add on package
Brent Laster committed on Jan 17

Showing 2 changed files with 52 additions and 0 deletions.

25 .github/workflows/publish-package.yml

```
@@ -0,0 +1,25 @@
 1 + name: Publish package to GitHub Packages
 2 +
 3 +   release:
 4 +     types: [created]
```

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

Initial add on package #1

Open gwstudent wants to merge 1 commit into **main** from **package**

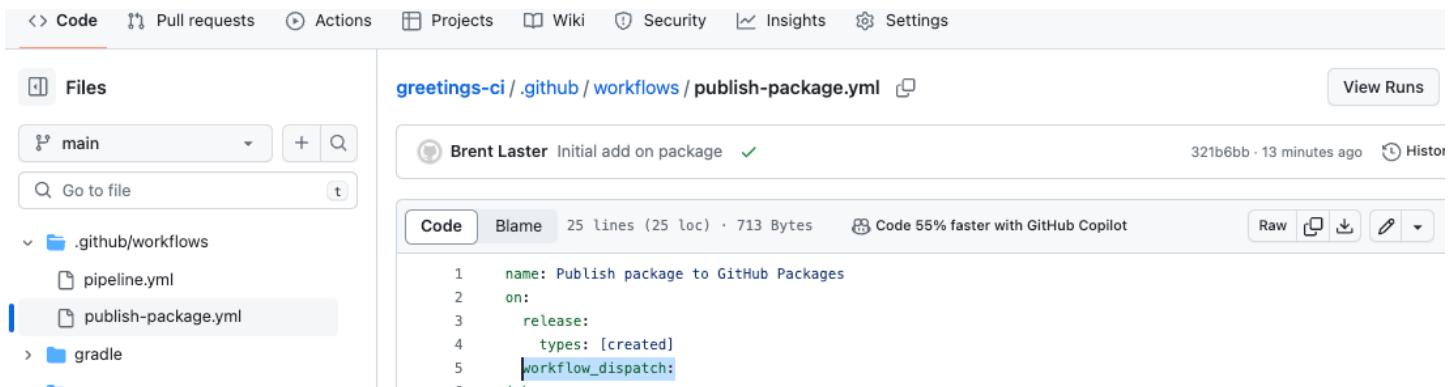
Commits 1 **Checks** 1 **Files changed** 2

.github/workflows/publish-package.yml

```
@@ -0,0 +1,25 @@
 1 + name: Publish package to GitHub Packages
 2 +
 3 +   release:
 4 +     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 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.)

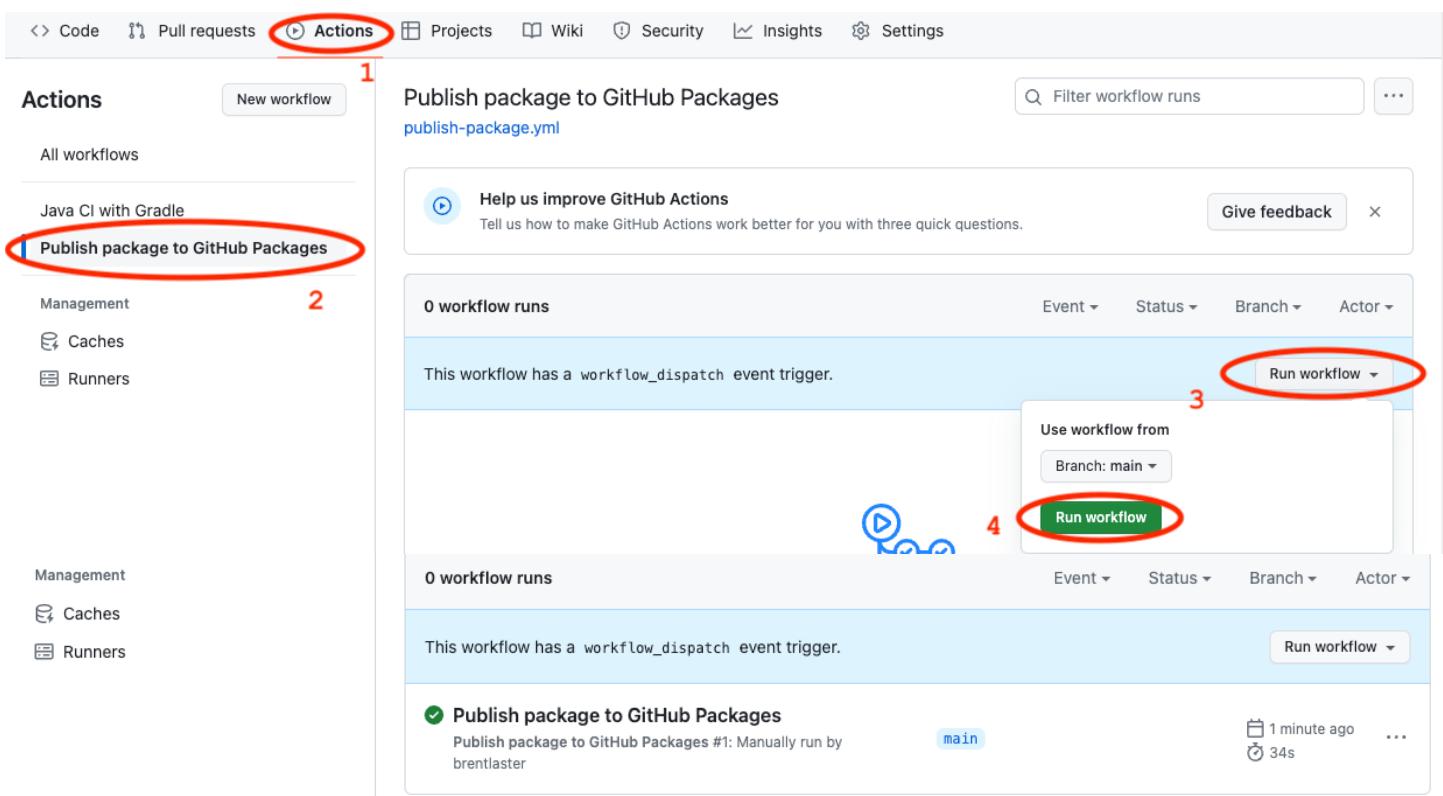


```

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.



1

Actions

2

Java CI with Gradle

Publish package to GitHub Packages

3

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.

This screenshot shows the GitHub repository page for 'greetings-ci'. At the top, there's a navigation bar with links for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the repository name 'greetings-ci' is displayed, along with a public status badge, a fork from 'skillrepos/greetings-ci', and a star count of 0. There are also buttons for Pin, Watch, Fork (140), and Star. The main content area shows a branch summary: 'main' (1 Branch, 0 Tags), a commit message 'This branch is 2 commits ahead of, 3 commits behind skillrepos/greetings-ci:main.', and a list of 27 commits. On the right side, there's an 'About' section with a description: 'Simple starter repo for CI/CD with GitHub Actions', activity metrics (0 stars, 0 watching, 140 forks), releases (none published), and a packages section. The package 'org.gradle.sample.greetings-ci' is listed here and highlighted with a red circle.

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.

This screenshot shows the GitHub user profile for 'brentlaster'. At the top, there's a navigation bar with links for Overview, Repositories (217), Projects, Packages (highlighted with a red circle labeled '2'), Stars (1), and a search bar. Below the navigation bar, there's a large circular profile picture of a man. To the right of the profile picture, there's a sidebar with options: Set status, Your profile (highlighted with a red circle labeled '1'), and Switch account. The main content area shows a summary: 1 package, the package 'org.gradle.sample.greetings-ci' (highlighted with a red circle labeled '3'), and a note that it was published 4 minutes ago by Brent Laster in brentlaster/greetings-ci. A list of other profile items is also shown.

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.

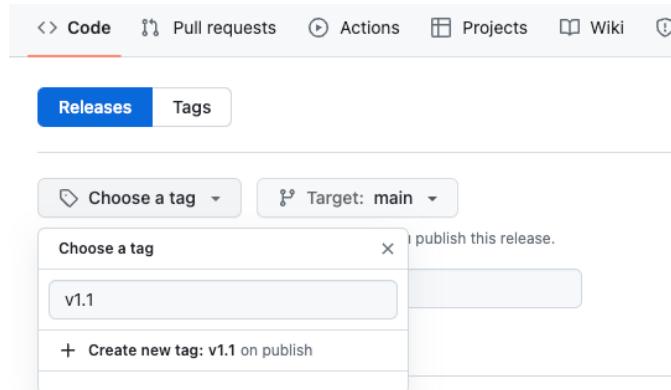
END OF LAB

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>.)

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



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

The screenshot shows the GitHub 'Releases' interface. At the top, there are tabs for 'Releases' (selected) and 'Tags'. Below that, a dropdown for 'Choose a tag' is set to 'v1.1'. A note 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 with 'Write' and 'Preview' tabs. A note below the editor says 'Previous tag: auto'. A large text area for 'Describe this release' is empty. Below it, a section for attachments says 'Attach files by dragging & dropping, selecting or pasting them.' A file named 'greetings-ci-1.1.1.jar' is listed with '(0.00 MB)' and a delete 'X' button. A red arrow points from this attachment area to a separate 'Downloads' window on the right. The 'Downloads' window shows a list of files, with 'greetings-ci-1.1.1.jar' being the last item in the list.

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

A screenshot of a modal dialog box. It has a large 'Attach binaries by dropping them here' area with a downward arrow icon. 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 'Initial release' page. At the top, there's a navigation bar with links like 'Code', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and a three-dot menu. Below the navigation is a breadcrumb trail: 'Releases / Initial release'. The main content area has a title 'Initial release' with a 'Latest' button. It shows a message from 'brentlaster' that was released 'now'. A note below says 'Initial release of content for 1.1.' Under the heading 'Assets', there is a list with one item: 'greetings-ci-1.1.jar' (1006 Bytes, 2 minutes ago), followed by two 'Source code' options (zip and tar.gz, both 21 minutes ago). There are also 'Compare', 'Edit', and 'Delete' buttons at the top right.

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.

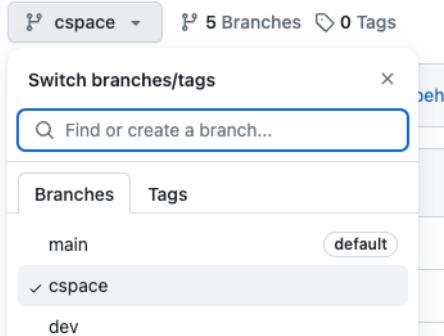
The screenshot shows the GitHub repository page for 'greetings-ci'. The top navigation bar includes 'Code', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below the navigation, it shows the repository owner 'brentlaster' and the fact that it's a 'Public' fork of 'skillrepos/greetings-ci'. The main content area displays a list of files and a commit history. On the right side, there's an 'About' section with a description: 'Simple starter repo for CI/CD with GitHub Actions'. Below that is an 'Activity' section showing 0 stars, 0 watching, and 140 forks. The most prominent feature on the right is the 'Releases' section, which lists '1' release. The first release is titled 'Initial release' with a 'Latest' button, and it was released '1 minute ago'. This release is circled in red. Below the releases is a 'Packages' section with one entry: 'org.gradle.sample.greetings-ci'.

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

Lab 10: Working with Codespaces

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

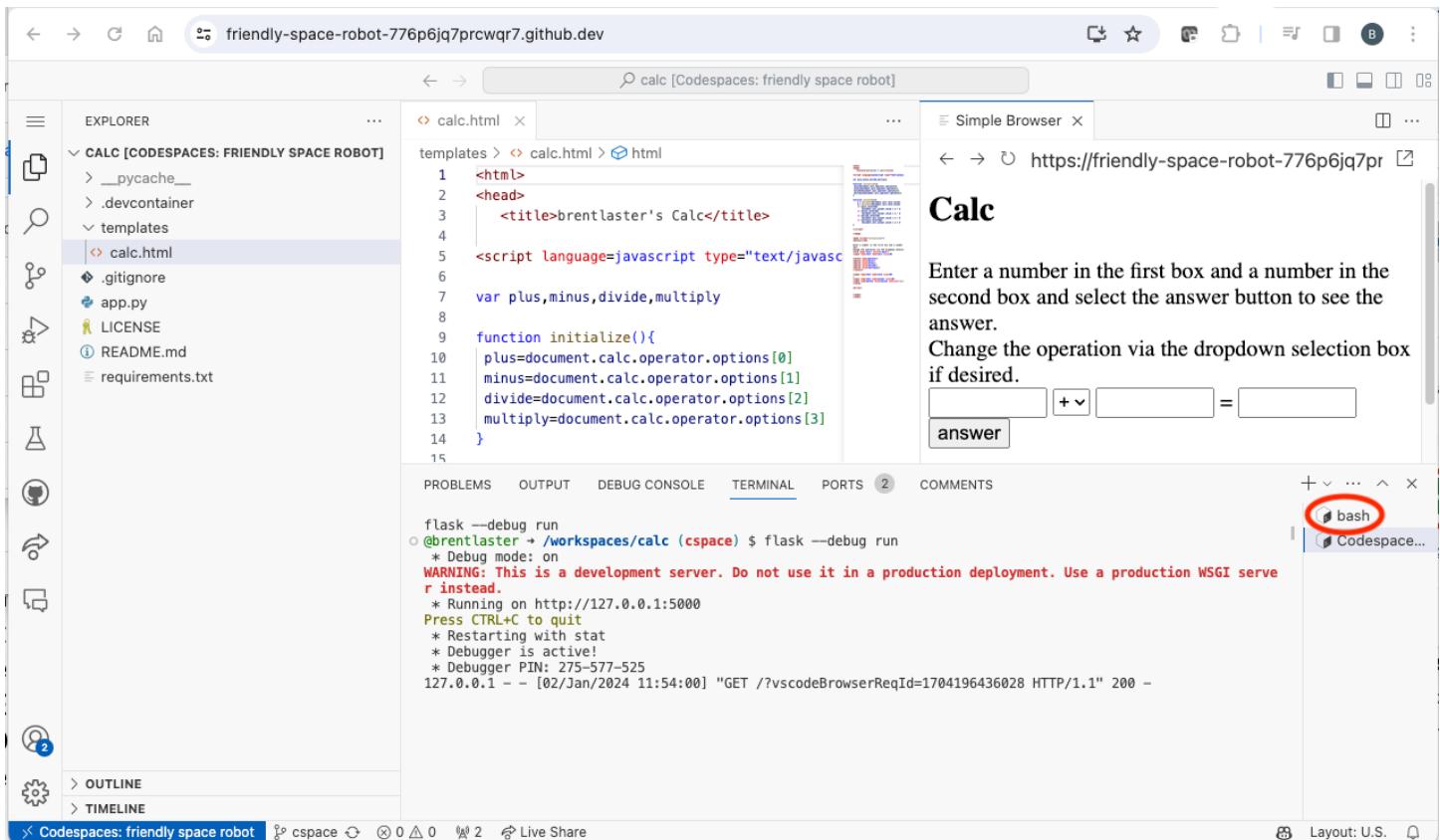
1. Go back to the `github.com/<github-userid>/calc` project. In that project, select the **cspac** branch.



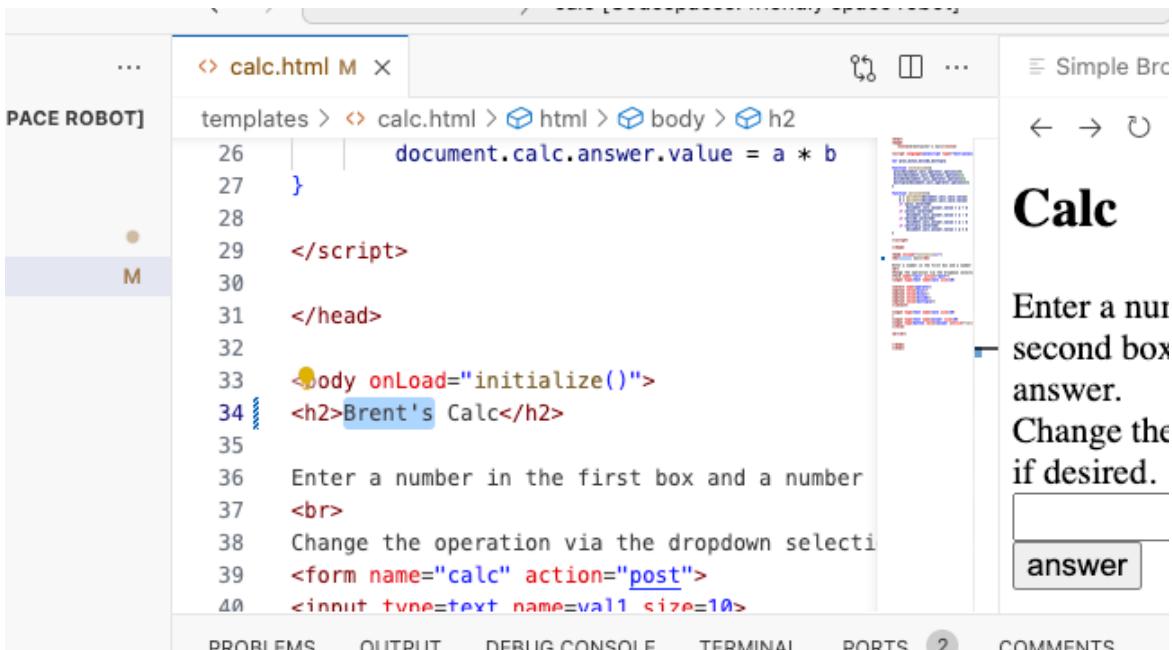
2. Click on the **<> Code** button, then select the **Codespaces** tab, and then select **Create codespace on cspace**.

A screenshot of the GitHub Codespaces interface. At the top, there's a 'Go to file' search bar, a '+' button (circled in red with '1'), a '**<> Code**' button (circled in red with '1'), and tabs for 'Local' (circled in red with '2') and 'Codespaces'. The 'Codespaces' tab is selected, showing the message 'No codespaces' and 'You don't have any codespaces with this repository checked out'. A large green button at the bottom right is circled in red with '3' and labeled 'Create codespace on cspace'. There's also a link 'Learn more about codespaces...'.

3. Creating the codespace will take a few minutes to complete. When it's done, you'll now have a new codespace with this repo checked out and the calculator webpage open and running. There 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:

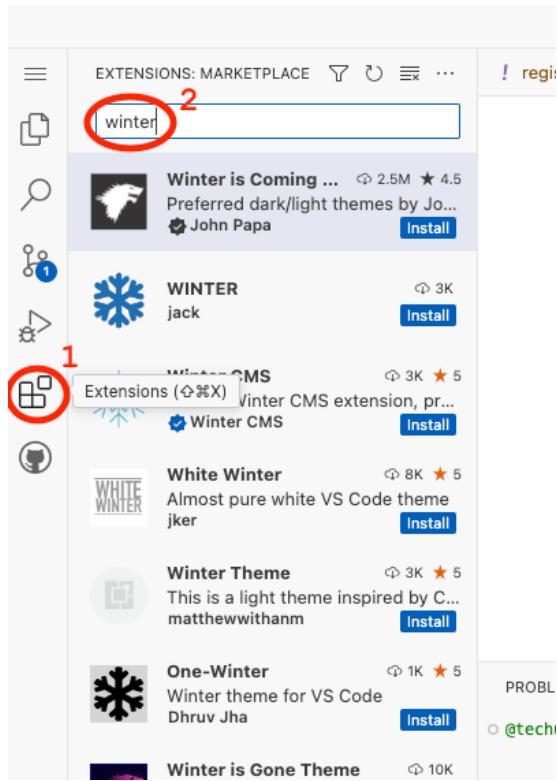
```

<h1>Brent's Calc</h1>
<input type="text" id="a">
<input type="text" id="b">
<button type="button" value="+" id="add">+
<button type="button" value="-" id="subtract">-
<button type="button" value="*" id="multiply">*
<button type="button" value="/" id="divide">/
<script>
    document.getElementById('add').addEventListener('click', function() {
        let a = document.getElementById('a').value;
        let b = document.getElementById('b').value;
        let result = Number(a) + Number(b);
        document.getElementById('result').value = result;
    });
    document.getElementById('subtract').addEventListener('click', function() {
        let a = document.getElementById('a').value;
        let b = document.getElementById('b').value;
        let result = Number(a) - Number(b);
        document.getElementById('result').value = result;
    });
    document.getElementById('multiply').addEventListener('click', function() {
        let a = document.getElementById('a').value;
        let b = document.getElementById('b').value;
        let result = Number(a) * Number(b);
        document.getElementById('result').value = result;
    });
    document.getElementById('divide').addEventListener('click', function() {
        let a = document.getElementById('a').value;
        let b = document.getElementById('b').value;
        let result = Number(a) / Number(b);
        document.getElementById('result').value = result;
    });
</script>

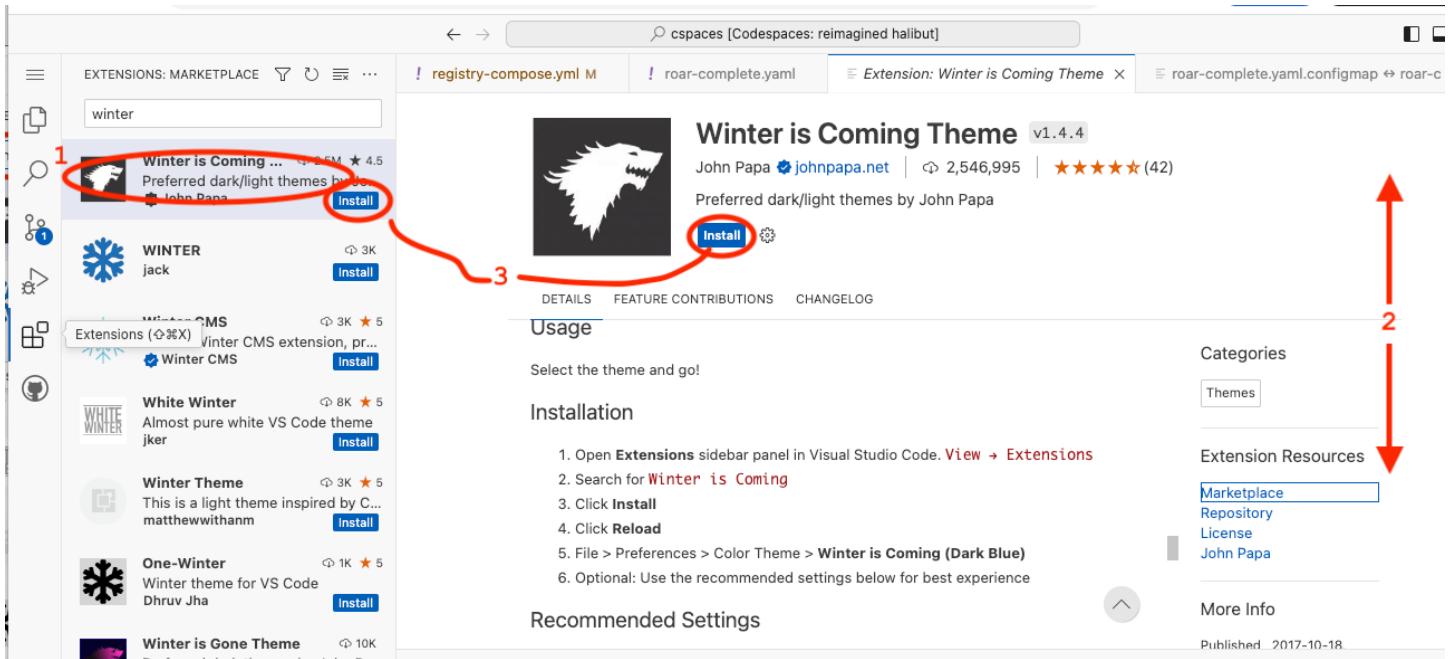
```

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

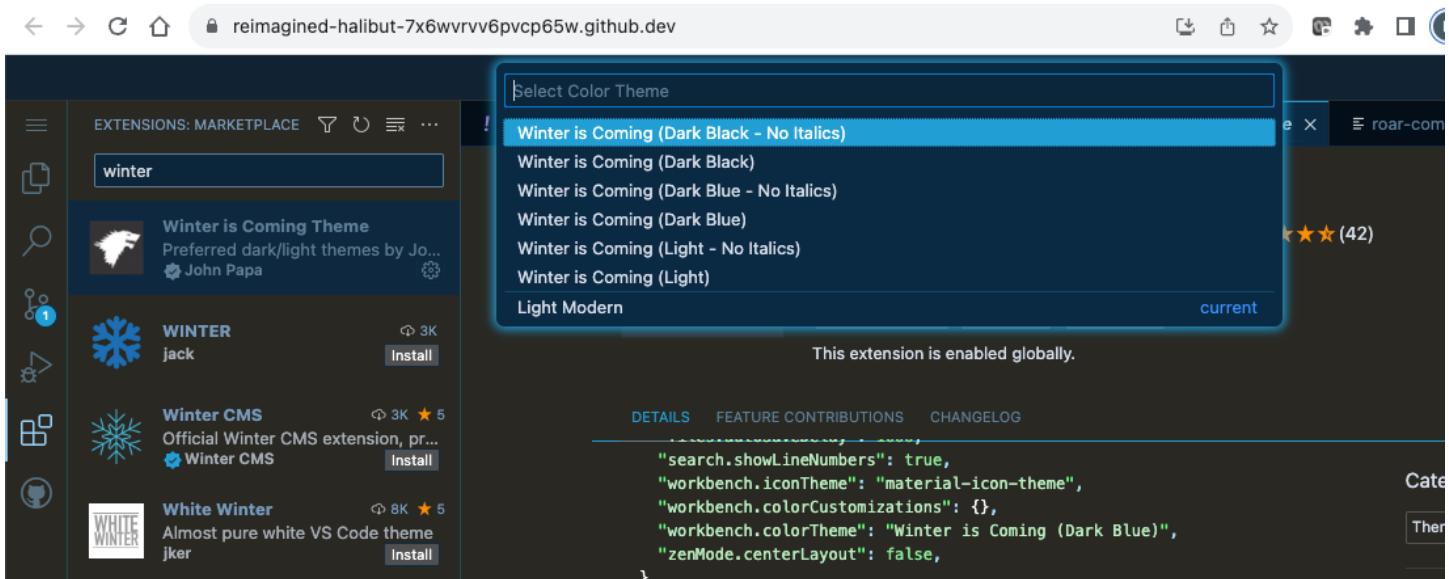
6. Next, let's see how to modify our codespace environment. We'll install an extension for the color theme. For practice, we'll use the "Winter is Coming" theme. Click on the *Extensions icon* (#1 in figure below), then in the *search bar* type in "winter" to quickly find the extension (#2).



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 (cspacel) $ 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
```

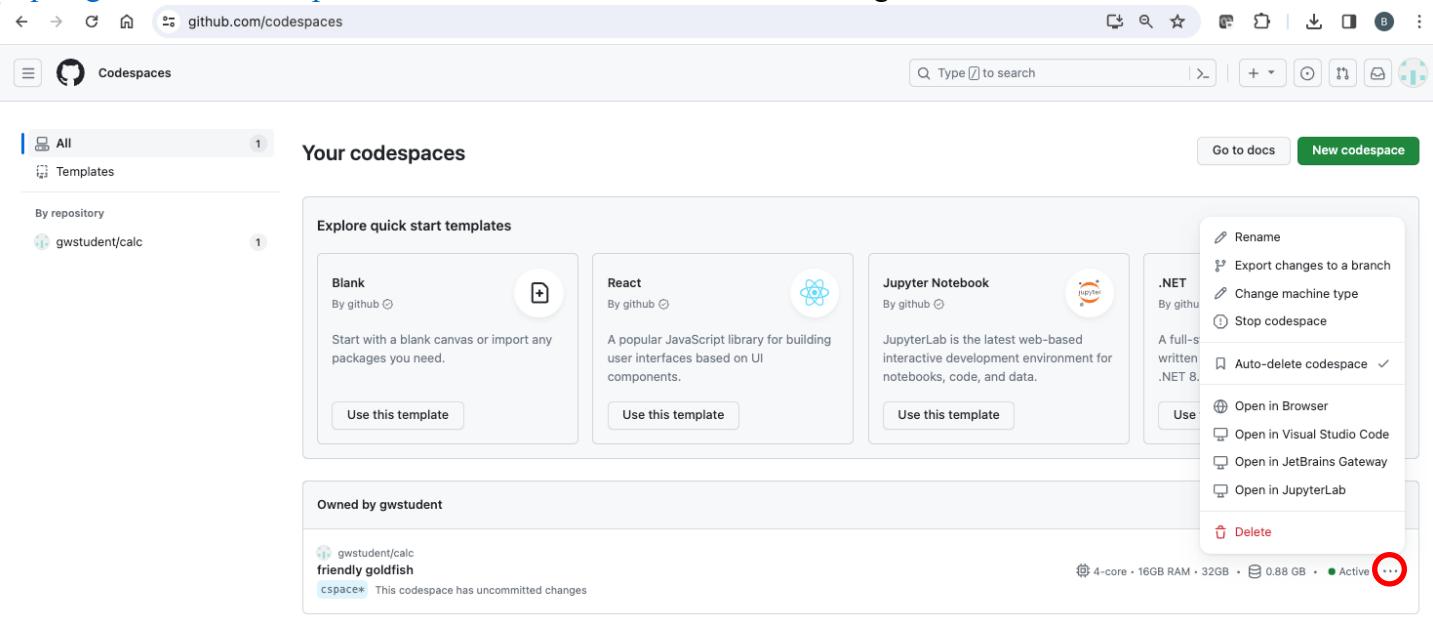
6. You can also clone repos easily with the command line. Change up one directory to not clone within the current directory. Then run the command to clone down your other repo.

```
$ 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

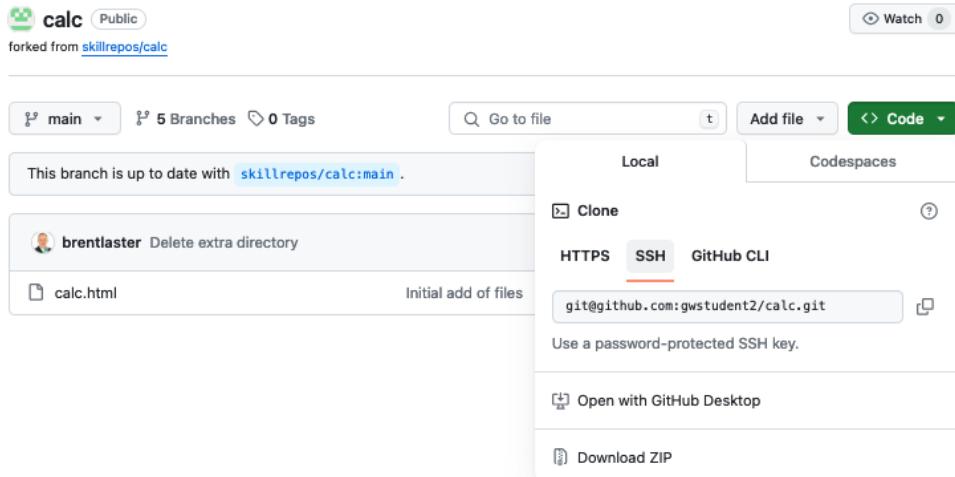
```
$ git config --unset --system credentials.helper
```

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

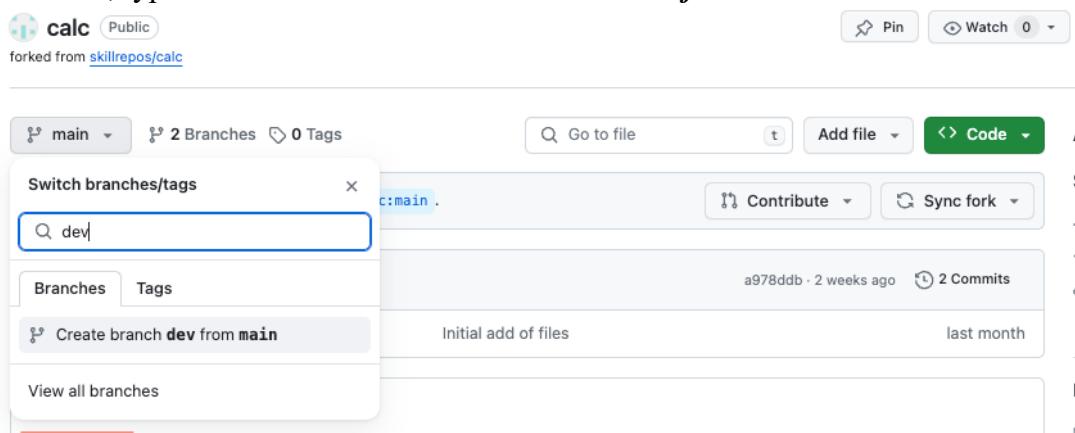
B. SSH keys: If you are familiar with using ssh and have keys, you can add them into GitHub and use those.

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'. The repository is public and has been forked from 'skillrepos/calc'. At the top right, there are 'Pin' and 'Watch' buttons. Below the repository name, it shows 'dev' as the active branch, '3 Branches', and '0 Tags'. A search bar and a 'Code' dropdown menu are also present. A message indicates that the branch is up to date with 'skillrepos/calc:main'. On the left, a commit by 'brentlaster' is listed, which deleted an extra directory. This commit was made 2 weeks ago and includes 2 commits. The commit details show files 'calc.html' and 'README' were added initially. 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 right, there is an 'Edit this file' button. Below the code, there are buttons for 'Raw', 'Copy', 'Download', 'Edit', and 'Diff'.

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, and the file 'calc.html' is being edited in the 'dev' branch. A modal dialog titled 'Commit changes' is open over the code editor. The 'Commit message' field contains the text 'Update calc.html'. Below it, the 'Extended description' field has the placeholder 'Add an optional extended description..'. At the bottom of the dialog, there are two radio button options: 'Commit directly to the dev branch' (selected) and 'Create a new branch for this commit and start a pull request', with a link to 'Learn more about pull requests'. The 'Commit changes' button is highlighted in green.

```
1 <html>
2 <head>
3     <title>Brent's Calc</title>
4
5     <script language=javascr...
6
7     var plus,minus,divide,mul...
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    }

```

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'. 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 is 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, as well as 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** sec-demo is available.

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

Description (optional)

Visibility

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.

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

(1) 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'. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, there's a search bar and a user profile icon for 'brentlaster'. The repository name 'sec-demo' is displayed, along with a 'Public' badge. On the right side, there are buttons for Pin, Unwatch (1), Fork (0), and Star (0). A 'Create a codespace' button is located in the top-left corner of the main content area.

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.