

# GitHub Fundamentals BootCamp Labs

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

**Revision 2.0 – 01/09/25**

Tech Skills Transformations LLC / Brent Laster

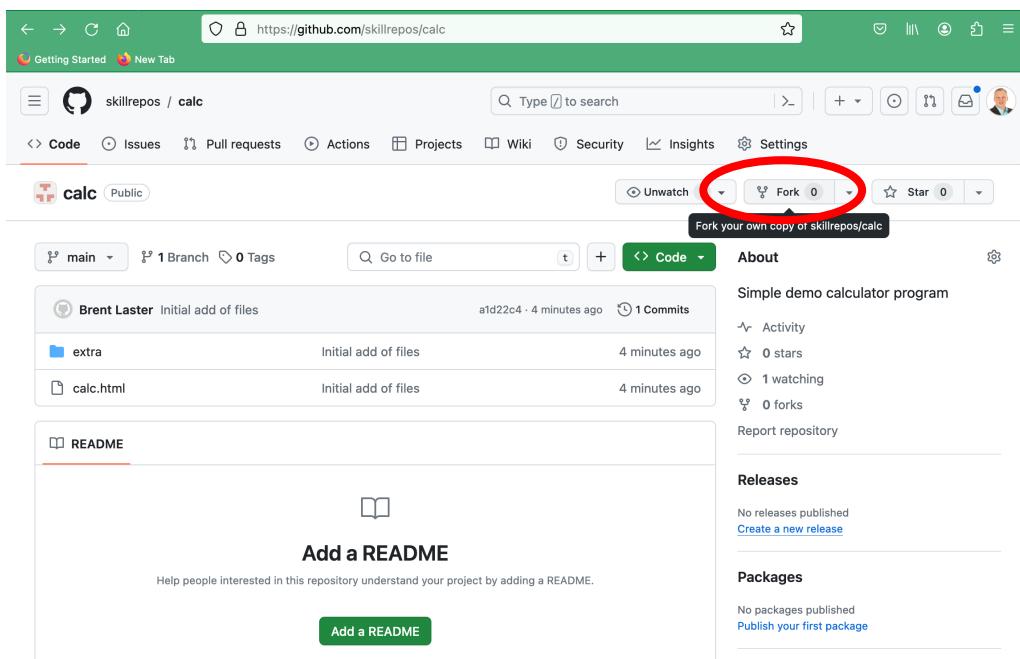
## Setup and prerequisites

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

## Lab 1 – Getting Started

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

1. Log in to GitHub with your primary GitHub account.
2. Go to <https://github.com/skillrepos/calc> and fork that project into your own GitHub space. Do this by clicking on the **Fork** button. On the next screen, **make sure to uncheck** the box next to **Copy the main branch only**. Then click the **Create Fork** button. (Note: If you cannot use the Fork functionality, see alternatives in **Appendix 2** at the end of this document.)



The screenshot shows the GitHub 'Create a new fork' page for the repository 'skillrepos / calc'. A yellow speech bubble with the text 'uncheck' points to the 'Copilot' checkbox, which is circled in red. Another red 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.

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

**Owner \*** brentlaster    **Repository name \*** 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

**Copilot 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 repository page for 'brentlaster / calc'. A red circle highlights the 'Code' tab. Another red circle highlights the 'Local' tab. A third red circle highlights the 'Clone' button. A fourth red circle highlights the 'HTTPS' option in the dropdown menu. A fifth red circle highlights the 'Copy url to clipboard' button.

**brentlaster / calc**

**Code**    **Pull requests**    **Actions**    **Projects**    **Wiki**    **Security**    **Insights**    **Settings**

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

**main**    1 Branch    0 Tags

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

**Local**

**Clone**

**HTTPS**    SSH    GitHub CLI

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

[Copy url to clipboard](#)

Brent Laster Initial add of files

extra    Initial a

calc.html    Initial a

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 hit Enter/Return. Then change into the subdirectory that was created from the clone.

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

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

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

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

```
$ git remote -v
```

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

#### Edit calc.html and change

<title>Calc</title>

to

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

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

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

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

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

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

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

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

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

**New personal access token (classic)**

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

**Note**

GitHub Fundamentals class

What's this token for?

**Expiration \***

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

**Select scopes**

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

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> 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"/> <b>workflow</b>	Update GitHub Action workflows
<input checked="" type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input checked="" type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry

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

**read:ssh\_signing\_key**      Read public user SSH signing keys

**Generate token**      **Cancel**

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

The screenshot shows the GitHub 'Personal access tokens (classic)' page. On the left, there's a sidebar with 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens' (which is expanded). Under 'Personal access tokens', there are 'Fine-grained tokens' and 'Tokens (classic)', with 'Tokens (classic)' being the active tab. A 'Beta' button is visible next to 'Tokens (classic)'. The main area displays a message: 'Make sure to copy your personal access token now. You won't be able to see it again.' Below this, a green box contains a copied token: 'ghp\_Kmdx72enC8FGSUoYQbc4W7gnMJBo3X39avRk'. A 'Copied!' message bubble is shown above the token.

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':
```

A screenshot of a terminal window showing a context menu with 'Paste' selected. The menu also includes options like 'Copy', 'Mark', 'Show Inspector', and 'AutoFill'.

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

The screenshot shows a GitHub repository interface. At the top, there are navigation buttons for 'dev' (selected), '3 Branches' (which shows 'main' as the default), and '0 Tags'. On the right, there are buttons for 'Go to file', 'Add file', and 'Code'. Below these are buttons for 'Contribute' and 'Sync fork'. The main area displays a commit history for the 'main' branch. The first commit is by 'Brent Laster' with the message 'Updating title', timestamped '8 hours ago'. To the left of the commit list, a modal window titled 'Switch branches/tags' is open, showing a search bar and a list of branches ('main', 'cspace', 'dev') with 'dev' checked. There is also a link 'View all branches' at the bottom of the modal.

The screenshot shows the same GitHub repository interface, but the branch has been switched to 'dev'. A message at the top states 'This branch is 1 commit ahead of skillrepos/main'. The commit history now shows a single commit by 'Brent Laster' titled 'Updating title' made 8 hours ago. The 'calc.html' file is selected, and a modal window titled 'Blame' is open, showing the blame history for this file. The modal lists two files: 'calc.html' and 'README'.

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

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

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.

Commits

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

- o- Commits on Dec 31, 2023
  - [Updating title](#) Brent Laster committed 26 minutes ago 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

```

v 4 calc.html
...
@@ -1,6 +1,6 @@
1 1 <html>
2 2 <head>
3 - <title>Calc</title>
3 + <title>brentlaster's Calc</title>
4 4
5 5 <script language=javascript type="text/javascript">
6 6
@@ -56,4 +56,4 @@ <h2>Calc</h2>
56 56
57 57
58 58 </body>
59 - </html>
59 + </html>

```

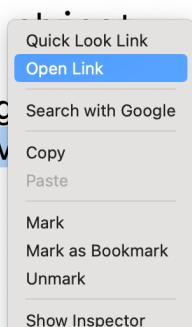
5. Let's now merge our change from the dev branch to main via a pull request. **Switch back to the terminal where you did the commit and the subsequent push.**

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

```

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

```



-- OR --

brentlaster / calc

Type ⌘ to search

Code Pull requests Actions Projects Wiki Security Insights Settings

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

Pin Watch 0

dev had recent pushes 26 minutes ago

Compare & pull request

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

### Comparing changes

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

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

Choose a Base Repository

Filter repos

skillrepos/calc

brentlaster/calc

Reviewers

No reviews

Assignees

No one—assign yourself

Write Preview

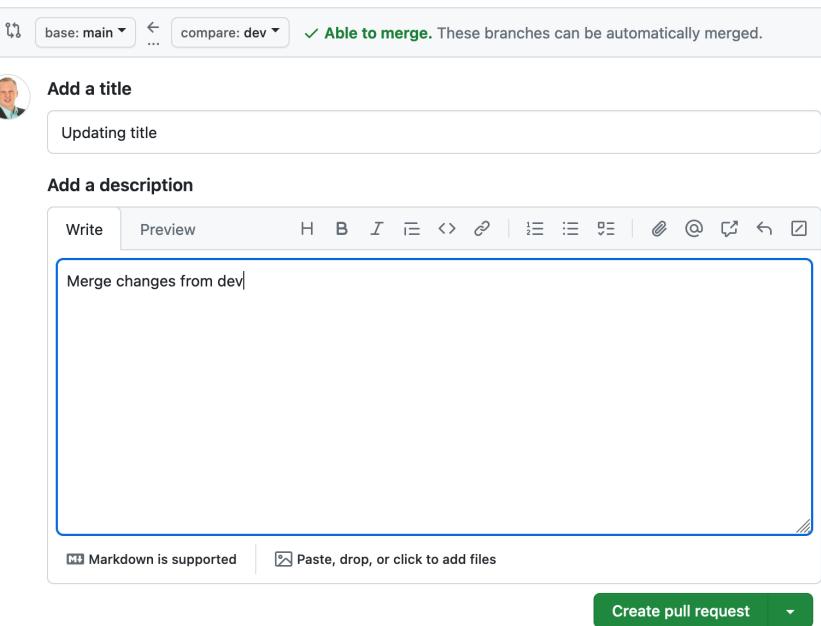
- After making that change, the gray bar showing the base and compare should look like the screenshot below.

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

base: main ▾ compare: dev ▾

Able to merge. These branches can be automatically merged.

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



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

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

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

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

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

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

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

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

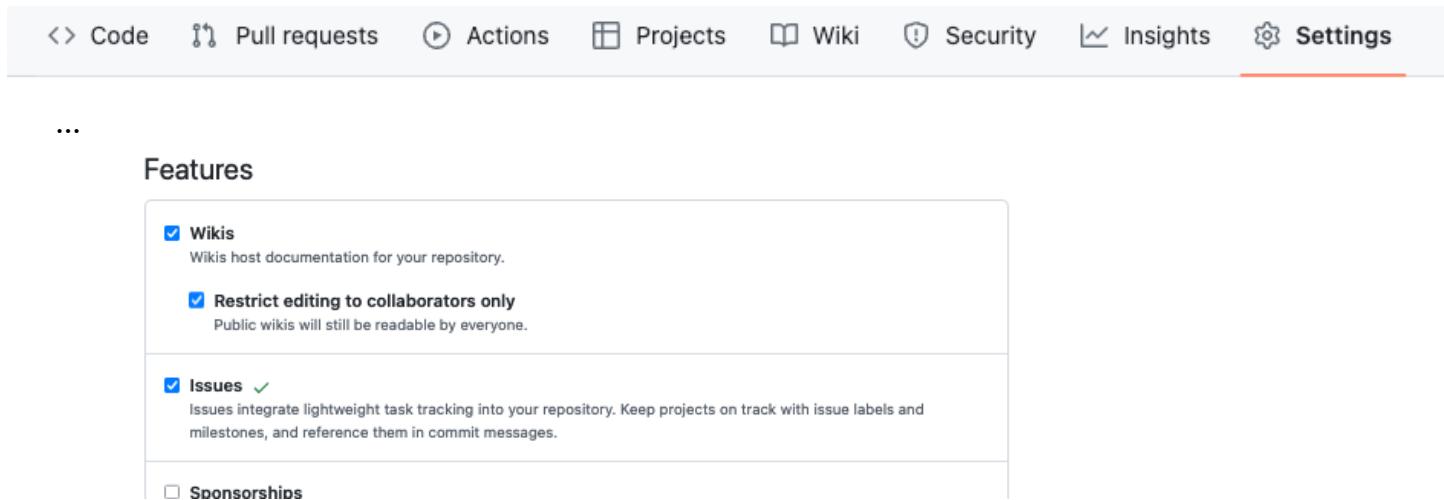
At the bottom, a purple box contains the message "Pull request successfully merged and closed" and "You're all set—the `dev` branch can be safely deleted." A "Delete branch" button is also present.

END OF LAB

## Lab 3: Creating GitHub issues

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

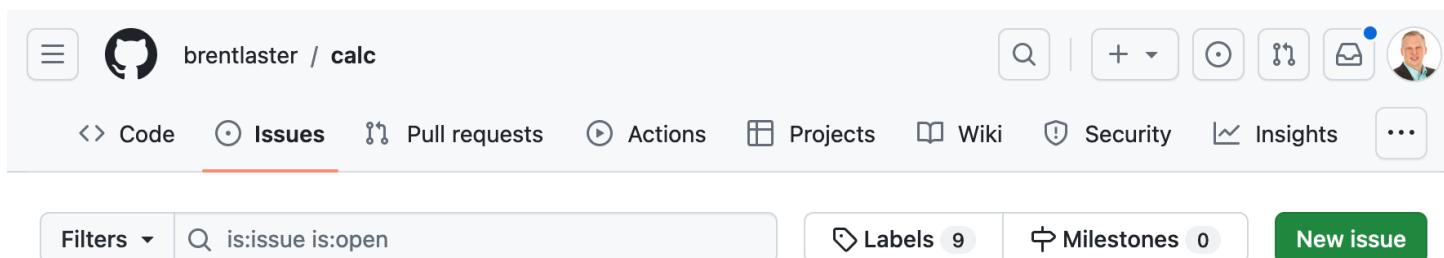
1. We'd like to have a *README* file in our project to make it more standard. So, let's create an issue to document that. First, ensure that the repository has the *Issues* feature turned on. On the main repo page, go to the repository's **Settings** tab, and then scroll down until you see the **Features**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



The screenshot shows the 'Features' section of the GitHub repository settings. The 'Issues' checkbox is checked, indicating that lightweight task tracking is enabled. Other features like Wikis and Restrict editing to collaborators only are also listed.

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 page with the 'Issues' tab selected. The 'New issue' button is highlighted in green. The search bar contains the query 'is:issue is:open'. There are filters for 'Labels' (9) and 'Milestones' (0), and a 'New issue' button.

Add a title

Needs README

Add a description

Please add README file :book:

Write Preview H B I E ↵ | E E E ...

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

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)

Code Issues 1 Pull requests Actions Projects Wiki

## Needs README #2

**brentlaster** commented now · 0 comments

brentlaster commented now

Please add README file

Add a comment

Write Preview H B I E ↵ | E E 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 navigation links for Wiki, Security, and Insights. Below that, there are two buttons: 'Edit' and 'New issue'. The main area is titled 'Assignees' with a gear icon for settings. It displays the message 'No one—[assign yourself](#)'. There is also a three-dot menu icon on the left.

5. Add the documentation label to the issue by clicking on *Labels* and selecting the *documentation* one. (After selecting it, just click anywhere else outside of the box to apply it.)

The screenshot shows the 'Apply labels to this issue' modal. It has a 'Filter labels' input field containing 'Something isn't working'. Below it, there are two label options: 'documentation' (selected, indicated by a checked checkbox) and 'duplicate' (unchecked). The 'documentation' label has a description 'Improvements or additions to documentation' below it. A note at the bottom says '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 the GitHub Issues page for the repository 'brentlaster/calc'. The 'Issues' tab is selected, showing 1 open issue. The search bar contains 'is:issue is:open'. The filter bar shows '1 Open' and 'Needs README documentation' (which is highlighted in blue). The results table shows one issue: '#2 opened 15 hours ago by brentlaster'. The issue title is 'Needs README documentation'.

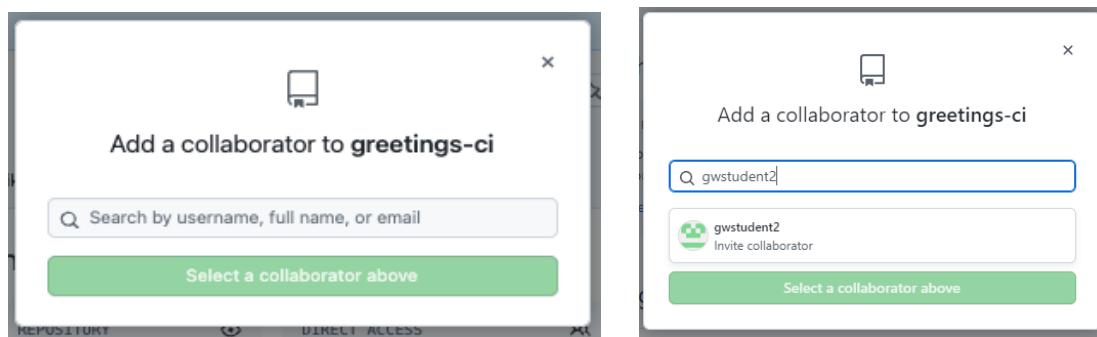
7. In preparation for the next lab, we need to add your second GitHub userid as a *collaborator* to this repository.

Go to the repository's **Settings** tab and then select **Collaborators** on the left under **Access**.

Then click the **Add people** button.

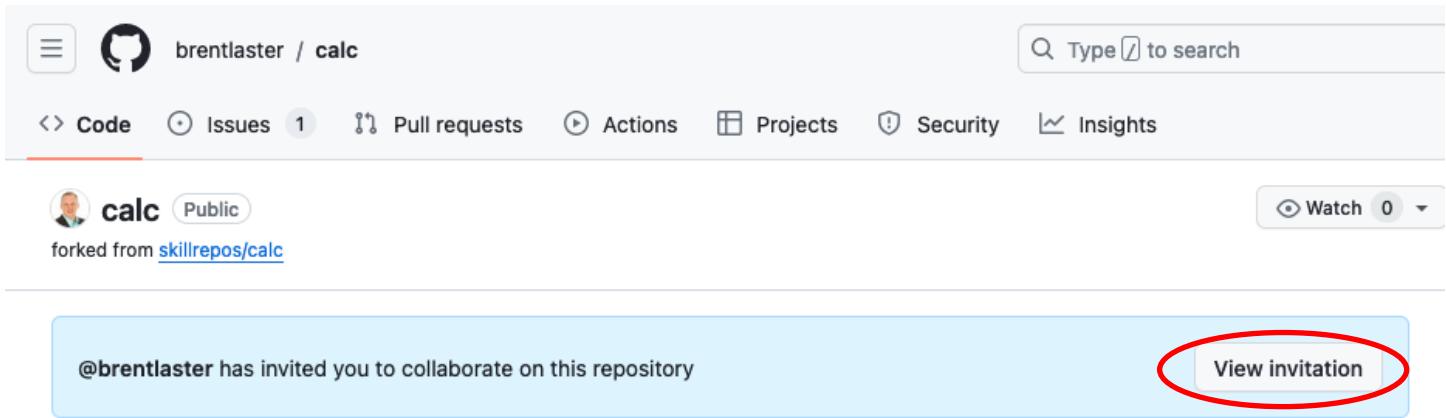
The screenshot shows the GitHub repository settings interface. The top navigation bar includes Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Settings tab is active. On the left, a sidebar menu is open under the 'Access' heading, with 'Collaborators' selected. Other options in the sidebar include Moderation options, Code and automation (Branches, Tags, Actions, Webhooks, Environments, Pages), Security (Code security and analysis, Deploy keys, Secrets), and a 'More' section. The main content area is titled 'Who has access' and shows two sections: 'PUBLIC REPOSITORY' (describing a public repository visible to anyone) and 'DIRECT ACCESS' (showing 0 collaborators). Below this is a large 'Manage access' section with a header 'Manage access' and a sub-section 'You haven't invited any collaborator' containing an 'Add people' button.

8. In the dialog box that pops up, type in 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.)

You can get to the invitation through the email link -or- 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 and navigation links for Code, Issues (1), Pull requests, Actions, Projects, Security, and Insights. Below the header, it says 'calc' (Public) and 'forked from skillrepos/calc'. On the right, there's a 'Watch' button with a count of 0. A blue banner at the top of the main content area states '@brentlaster has invited you to collaborate on this repository' with a 'View invitation' button circled in red.

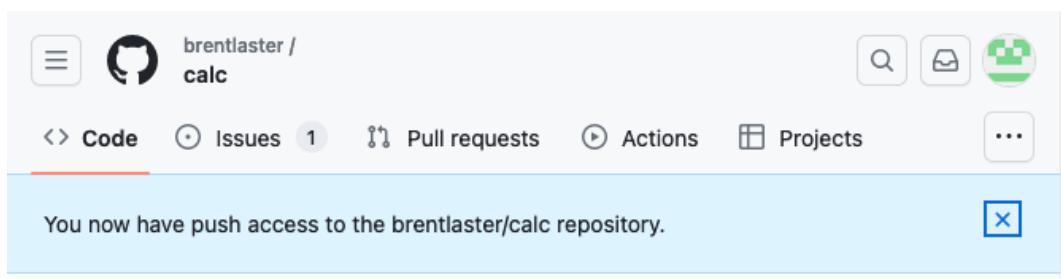
**brentlaster invited you to collaborate**

**Accept invitation** (button circled in red) | Decline

**Owners** of calc will be able to see:

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

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



The screenshot shows the same GitHub repository page after accepting the invitation. The banner now says 'You now have push access to the brentlaster/calc repository.' with a close button.

END OF LAB

## Lab 4: Setting up a pull request with reviewers

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

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

The screenshot shows the GitHub interface for the 'calc' repository. The top navigation bar has tabs for Code, Issues (1), Pull requests, Actions, Projects, Wiki, and Settings. Below the header, it says 'forked from skillrepos/calc'. The main area shows a branch status: 'This branch is 2 commits ahead of skillrepos/calc:main'. Below this, a list of commits is shown:

- brentlaster Merge pull request #1 from br... dc98b08 · yesterday 3 Commits
- extra Initial add of files last week
- calc.html Updating title yesterday

Under the 'README' section, there is a button labeled 'Add a README' which is circled in red.

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

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

Code Issues 1 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](https://github.com/brentlaster)
2
```

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

Code Issues 1 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 📈 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**.

Code Issues 1 Pull requests Actions Projects Wiki Security Insights

**calc / README.md** in main

Cancel changes Commit changes... 1

Edit Preview Show Diff

This is a simple calculator 📈 program. ? can be directed to @brentlaster

Propose changes

Commit message  
Update README.md

Extended description  
Add an optional extended description...

Commit directly to the main branch  
 Create a new branch for this commit and start a pull request  
Learn more about pull requests

standards 3

4 Cancel 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, it says 'base: main' and 'compare: standards'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' Below this, there's a summary: '1 commit', '1 file changed', and '1 contributor'. A red circle highlights the 'Create pull request' button. The commit details show 'Create README.md' by 'brentlaster committed now'. The diff view shows a single addition to README.md: '+ This is a simple calculator :abacus: program. :question: can be directed to [@brentlaster] (https://github.com/brentlaster)'. There are 'Split' and 'Unified' options for the diff view.

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 click outside of the field to apply it.) Make sure your other userid shows up in the Reviewers section now.

The screenshot shows the 'Open a pull request' page. In the top right, there's a 'Reviewers' section with a count of '1'. A red circle highlights the 'Reviewers' link. Below it, a dropdown menu shows 'Type or choose a user' with 'gwstudent2' selected, indicated by a red circle. Other fields include 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Development' (using closing keywords). At the bottom, there's a 'Create pull request' button.

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.

The screenshot shows the GitHub pull request creation interface. At the top, it says "base: main" and "compare: standards". A green checkmark indicates "Able to merge. These branches can be automatically merged." In the "Add a title" field, there is a placeholder "Create README.md". The "Add a description" field contains the text "Resolves #2 1" followed by a blue button with the text "#2 Needs README". A large red circle highlights this entire section. At the bottom of the description field, there is a note "Markdown is supported" and a "Paste, drop, or click to add files" button. Below the description field is a green "Create pull request" button, which is also highlighted with a red circle. To the right of the description field, there are sections for "Reviewers" (gwstudent2), "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Development" (Use Closing keywords in the description to automatically close issues). There is also a "Helpful resources" section.

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 userid.

Create README.md #3

**Open** brentlaster wants to merge 1 commit into [main](#) from [standards](#)

Conversation 0 Commits 1 Checks 0 Files changed 1

brentlaster commented 1 minute ago

Owner ...

Resolves #2

Create README.md Verified fb69983

brentlaster requested a review from gwstudent2 1 minute ago

Add more commits by pushing to the [standards](#) branch on [brentlaster/calculator](#).

**Review requested** Review has been requested on this pull request. It is not required to merge. [Learn more about requesting a pull request review.](#)

1 pending reviewer

Require approval from specific reviewers before merging Add rule

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 go to <https://github.com/pulls/review-requested>. Then click on the commit message for the pull request.

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

Getting Started New Tab

Pull Requests

Created Assigned Mentioned **Review requests**

is:open is:pr review-requested:gwstudent2 archived

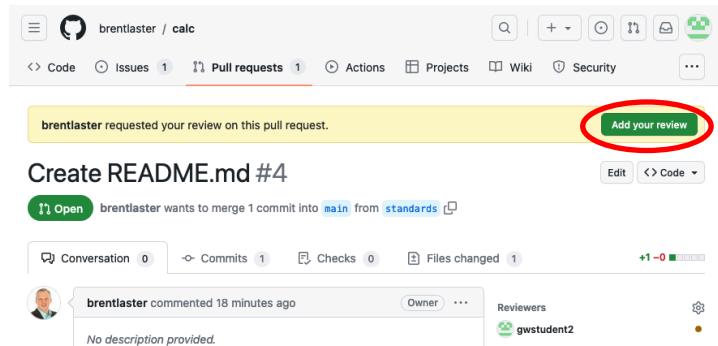
1 Open ✓ 1 Closed

brentlaster/calculator Create README.md

#4 opened 14 minutes ago by brendlaster

ProTip! Add [no:assignee](#) to see everything that's not assigned.

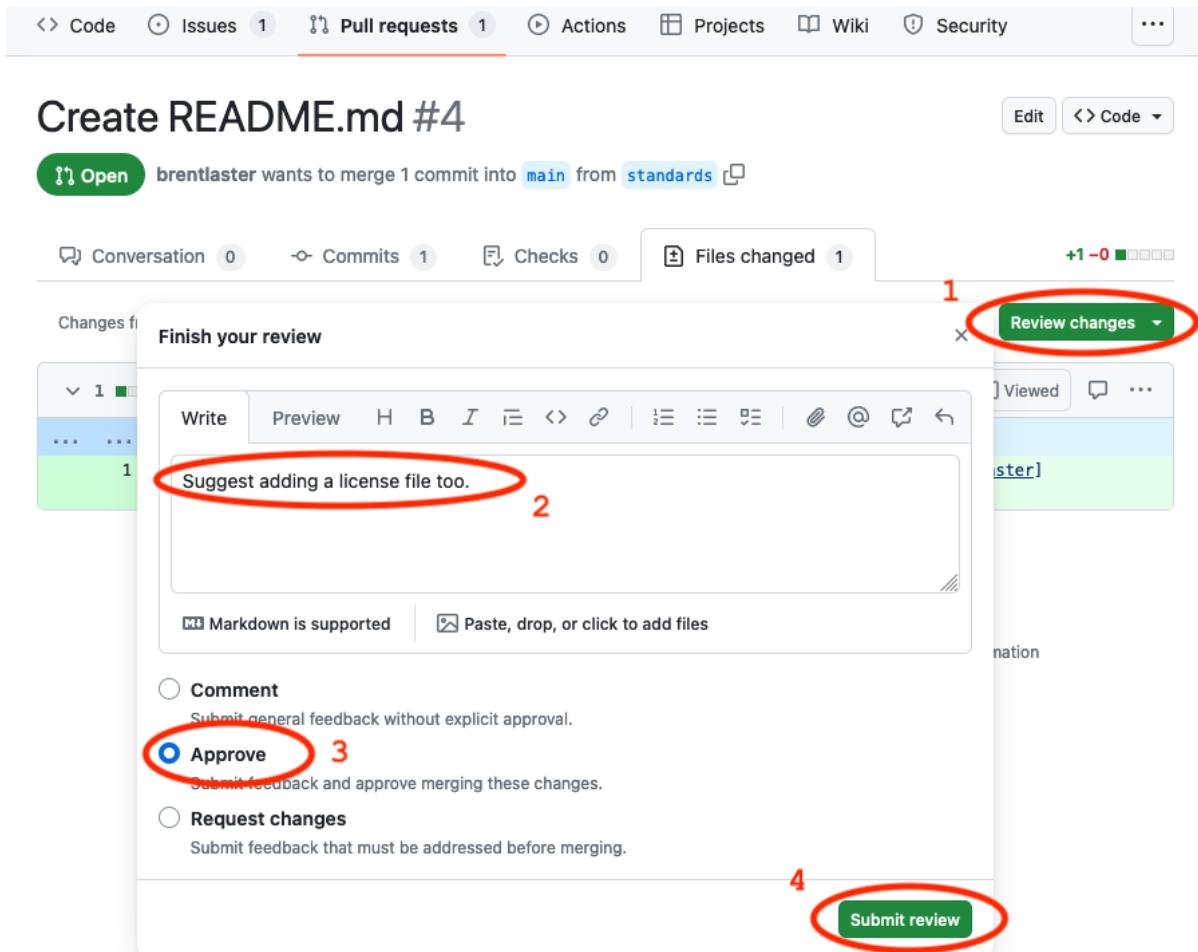
2. This will open up the pull request. There is a button at the top to “Add your review”. Click on that.



3. We could click on any of the lines and add a comment if we wanted, but since this is simply adding a README file, it looks ok. However, since this is about standards, let's make a suggestion to also add a license for the repo.

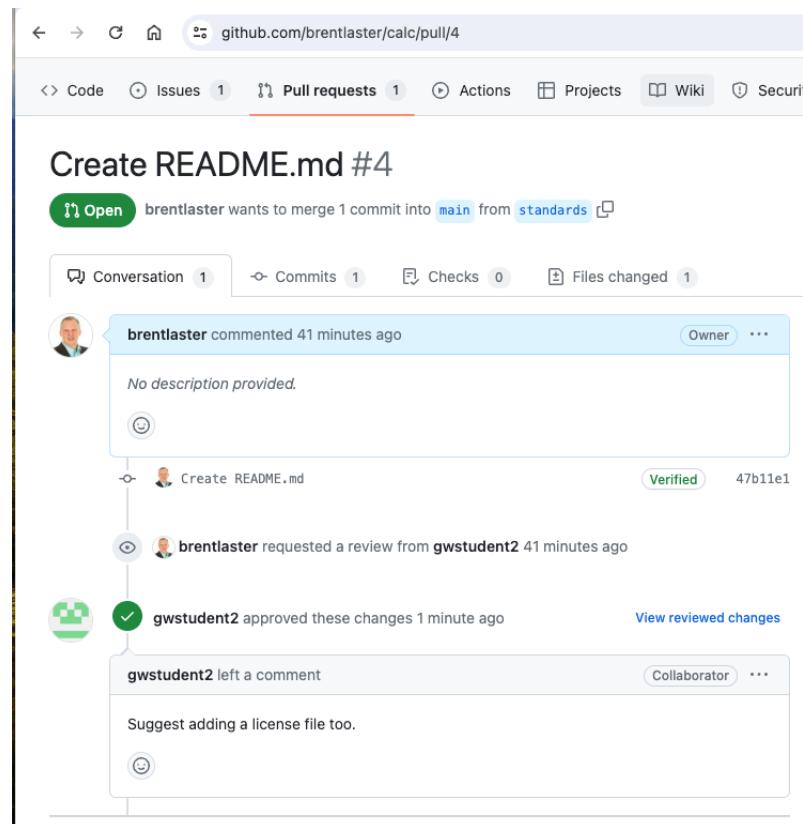
Select the “Review changes” button and add a comment to that effect.

Then select the “Approve” option, and then “Submit review”.



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. (Some icons and wording may look different.)



5. Since there was a suggestion to add a license file, that sounds like a good idea, so let's do that.

Click on the **Code** tab at the top, then select the **standards** branch (or whatever name you gave the new branch) from the branch dropdown.

Then select the “+” sign (or "Add file" option) and the option to **+ Create new file**.

The screenshot shows a GitHub repository named 'calc'. Step 1 highlights the 'Code' tab. Step 2 highlights the 'standards' dropdown menu. Step 3 highlights the '+' button in the top right of the code editor area. Step 4 highlights the 'Create new file' option in the dropdown menu.

**1** Code

**2** standards

**3** +

**4** 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”.

The screenshot shows a GitHub file creation dialog for a file named 'LICENSE'. Step 1 highlights the file name 'LICENSE'. Step 2 highlights the 'Choose a license template' button.

**1** LICENSE

**2** Choose a license template

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.	
<b>MIT License</b> <span style="color: red;">1</span>	<b>Permissions</b> ✓ Commercial use ✓ Modification ✓ Distribution ✓ Private use	<b>Limitations</b> ✗ Liability ✗ Warranty	<b>Conditions</b> ⓘ License and copyright notice
BSD 2-Clause "Simplified" License	This is not legal advice. <a href="#">Learn more about repository licenses.</a>		
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 1

Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

**Commit changes** 2

8. Go back to the pull request by selecting **Pull requests** at the top and selecting the one open pull request.

You can look at the changes currently in the pull request by clicking on the **Commits** tab and also the **Files changed** tab.

The screenshot shows a GitHub pull request interface. At the top, there is a navigation bar with tabs: Code, Issues (1), Pull requests (1), Actions, Projects, Wiki, Security, Insights, and Settings. The 'Pull requests' tab is highlighted with a red circle and the number '1'. Below the navigation bar, the title 'Create README.md #4' is displayed, followed by a green 'Open' button and a message from 'brentlaster' about merging two commits into 'main' from 'standards'. The main content area shows 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)'. At the bottom of the interface, there are tabs for 'Changes from all commits', 'File filter', 'Conversations', and 'Review changes'. The 'Files changed' tab is also circled in red. The status bar at the bottom right shows '+22 -0' and a progress bar.

9. Click back on the **Conversation** tab in the pull request and go ahead and merge (*Merge pull request*) and close (*Confirm merge*) the pull request.

After completing the merge, you should be able to click on the **Issues** tab and see that your issue has been automatically closed. You can click on the **Closed** list and then open the issue to see the automatically generated log of comments and actions if you want.

The screenshot shows a GitHub issue page for a repository named 'brentlaster / calc'. The issue is titled 'Needs README #2' and is marked as 'Closed'. A comment from 'brentlaster' 19 hours ago says 'Please add README file'. This comment was self-assigned and added the 'documentation' label. It was mentioned in another issue (#4) and linked to a pull request (#4) that was merged. The issue was then updated with 'Update README.md' and closed as completed. The right sidebar shows details like assignees ('brentlaster'), labels ('documentation'), projects ('None yet'), milestones ('None'), and notifications ('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.

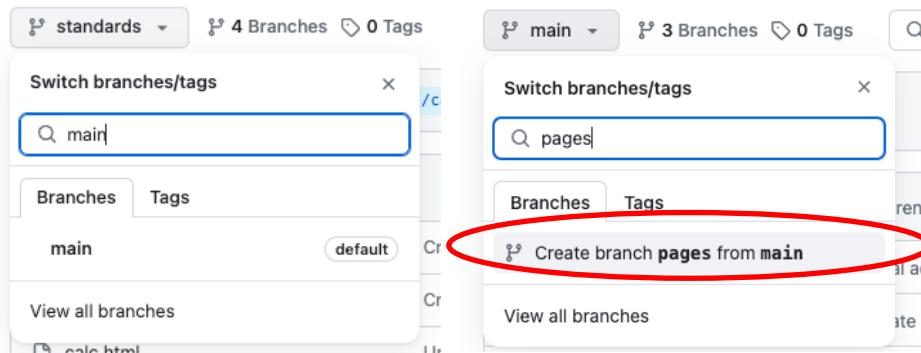
1. This lab is done with your primary GitHub id.

In order to prepare for publishing a page, let's create a new branch in our repo.

In the **Code** tab, if not on the **main** branch, click on the branch dropdown, type in **main** and select it.

Click in the branch dropdown again, and, in the text area that says, **Find or create a branch...**, enter the text **pages**.

Then click on the “Create branch: **pages** from **main**” link.



## 2. Now create a new repository to use for the "pages" repo.

Go to <https://github.com/new> to create a new repository. (Alternatively, you could go to your home page, then to **Repositories**, then to **New**.)

Refer to the picture below. 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 \*

brentlaster ▾

Repository name \*

brentlaster.github.io

brentlaster.github.io is available.

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

#### Description (optional)

Code repo for the web page for the calc code

Public

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

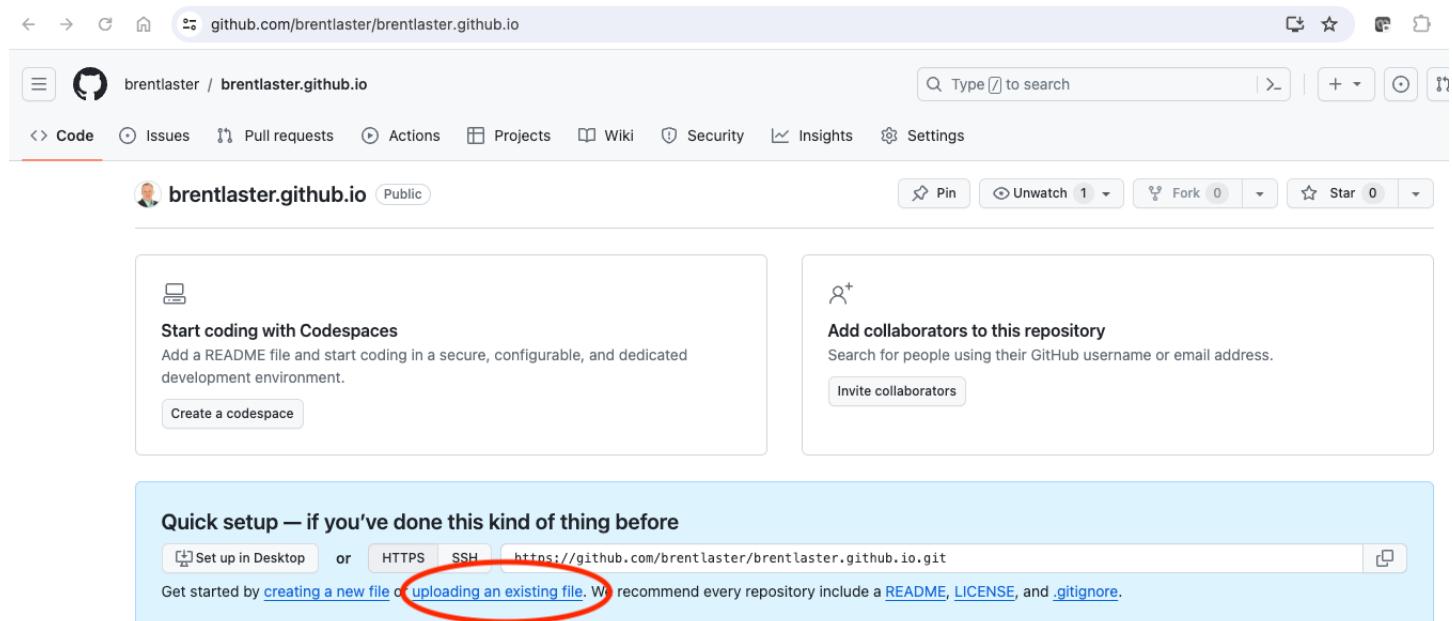
Private

You choose who can see and commit to this repository.

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

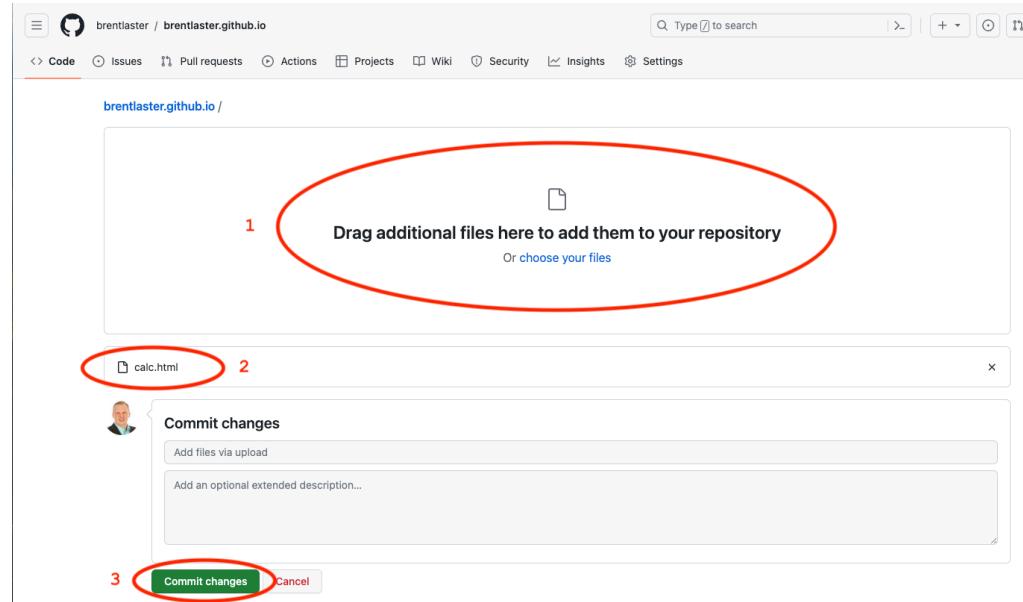
**Create repository**

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



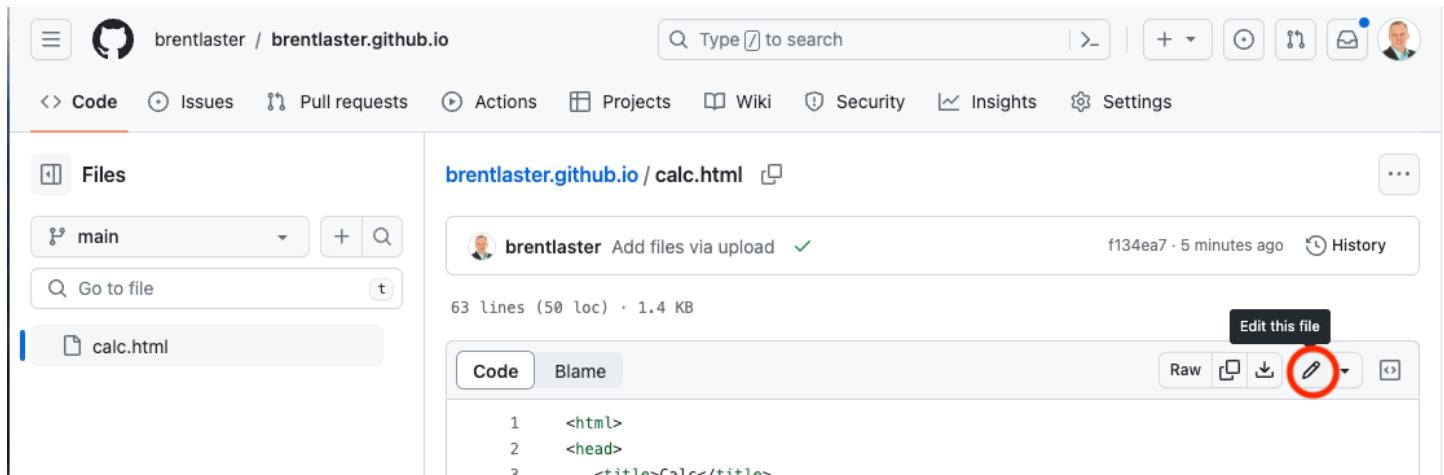
The screenshot shows a GitHub repository setup page for 'brentlaster/brentlaster.github.io'. The 'Code' tab is selected. In the center, there's a 'Quick setup – if you've done this kind of thing before' section. It includes links for 'Set up in Desktop', 'HTTPS', and 'SSH', and a URL 'https://github.com/brentlaster/brentlaster.github.io.git'.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

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



The screenshot shows the 'Commit changes' interface. At the top, there's a large input field with a red oval around it, labeled 'Drag additional files here to add them to your repository' and 'Or choose your files'. Below this, a file named 'calc.html' is listed with a red oval around it and the number '2'. At the bottom, there's a 'Commit changes' button with a red oval around it and 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.



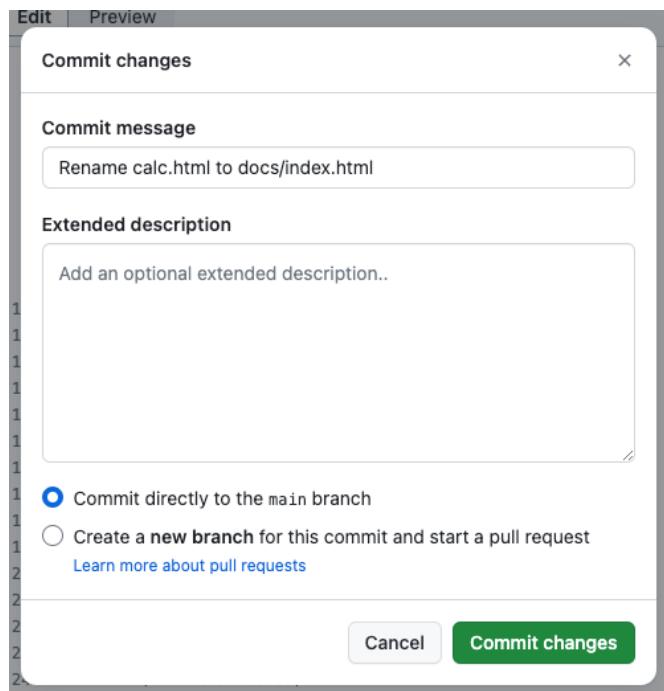
5. In the edit dialog, in the text entry box for the name, type over the "calc.html" text with the replacement text of "docs/index.html". Note you are adding the directory *docs* to the path. The screenshots show this in 2 parts for clarity.

This block contains two screenshots of the GitHub Code tab illustrating the renaming process.

**Screenshot 1:** The first screenshot shows the 'Edit' dialog for the file 'calc.html'. The file path 'brentlaster.github.io / calc.html' is highlighted with a red oval. The 'Edit' button is visible at the bottom of the dialog.

**Screenshot 2:** The second screenshot shows the result of the rename. The file path has been changed to 'brentlaster.github.io / docs / index.html', which is also highlighted with a red oval. The 'Commit changes...' button is visible at the bottom of the dialog.

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 **main** on the left and then **/docs** entry on the right 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.

+  =

### 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

4 Commits

[Code](#)

About

Code repo for the web page for the calc code

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

Description  
Simple web calculator program

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

Use your GitHub Pages website

Topics (separate with spaces)  
calculator example-code

Include in the home page  
 Releases  
 Packages  
 Deployments

About  
Simple web calculator program  
[brentlaster.github.io/](http://brentlaster.github.io/)  
calculator example-code

Activity  
0 stars  
1 watching  
0 forks

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

END OF LAB

## Lab 7: Learning about GitHub Actions

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

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

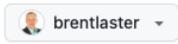
The screenshot shows two GitHub pages side-by-side. The top page is the original repository at <https://github.com/skillrepos/greetings-ci>. It displays the repository details, including the main branch, 2 branches in total, and 0 tags. The bottom page is a fork of this repository at <https://github.com/skillrepos/greetings-ci/fork>. Both pages have similar navigation bars and search bars. The forked repository page includes an 'About' section describing it as a 'Simple starter repo for CI/CD with GitHub Actions'.

### Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks](#).

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

Owner \*



Repository name \*

greetings-ci

greetings-ci is available.

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

Description (optional)

Simple starter repo for CI/CD with GitHub Actions

uncheck

Copy the `main` branch only

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

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

**Create fork**

3. We have a simple java source file named `echoMsg.java` in the subdirectory `src/main/java`, a Gradle build file in the root directory named `build.gradle`, and some other supporting files.

We could clone this repository and build it manually via running Gradle locally.

But let's set this to build with an automatic CI process specified via a text file.

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

The screenshot shows the GitHub repository page for 'gwstudent/greetings-ci'. The 'Code' tab is selected. The 'Actions' button in the top navigation bar is circled in red. Below the navigation bar, there are buttons for 'main', '1 branch', '0 tags', 'Go to file', 'Add file', and 'Code'. The main content area displays a list of commits from 'Brent Laster' with the commit message 'add extra dir'. The commits are dated '7 minutes ago' and show 2 commits. To the right, there is an 'About' section with details about the repository, including 'Simple starter repo for CI/CD with GitHub Actions', 0 stars, 0 watching, and 1 fork. There are also sections for '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**.

#### Automation

The screenshot shows the 'Automation' section with four cards:

- Greetings**: By GitHub Actions. Greets users who are first time contributors to the repo. Includes 'Configure' and 'Automation' buttons.
- Stale**: By GitHub Actions. Checks for stale issues and pull requests. Includes 'Configure' and 'Automation' buttons.
- Manual workflow**: By GitHub Actions. Simple workflow that is manually triggered. Includes 'Configure' and 'Automation' buttons.
- Labeler**: By GitHub Actions. Labels pull requests based on the files changed. Includes 'Configure' and 'Automation' buttons.

#### Browse all categories

Automation  
Continuous integration

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

The screenshot shows the GitHub Actions search results for the repository 'gwstudent / greetings-ci'. A red circle highlights the search bar containing the text 'Gradle'. Below the search bar, the text 'Found 52 workflows' is displayed. The results are categorized into three columns:

- Android CI**: By GitHub Actions. Build an Android project with Gradle. **Configure** button.
- Java with Ant**: By GitHub Actions. Build and test a Java project with Apache Ant. **Configure** button.
- Clojure**: By GitHub Actions. Build and test a Clojure project with Leiningen. **Configure** button.
- Publish Java Package**: By GitHub Actions. Publish a Java package to a Maven repository. **Configure** button.
- Java with Gradle**: By GitHub Actions. Build and test a Java project using Gradle. **Configure** button.
- Publish Java Package**: By GitHub Actions. Publish a Java package to GitHub Packages. **Configure** button.

## Get started with GitHub Actions

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

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

Categories

Automation

**Continuous integration**

Deployment

Security

Q Gradle

Found 52 workflows

**Android CI**  
By GitHub Actions  
Build an Android project with Gradle.  
**Configure** Java

**Java with Ant**  
By GitHub Actions  
Build and test a Java project with Apache Ant.  
**Configure** Java

**Clojure**  
By GitHub Actions  
Build and test a Clojure project with Leiningen.  
**Configure** Clojure

**Publish Java Package**

**Java with Gradle**

**Publish Java Package**

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

The screenshot shows the GitHub Actions search results for the repository 'gwstudent / greetings-ci'. The search bar contains 'Gradle'. The results are categorized into three columns. The third column, 'Java with Gradle', has its 'Configure' button highlighted with a red circle.

## Get started with GitHub Actions

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

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

Categories

Automation

**Continuous integration**

Deployment

Security

Q Gradle

Found 3 workflows

**Android CI**  
By GitHub Actions  
Build an Android project with Gradle.  
**Configure** Java

**Publish Java Package with Gradle**  
By GitHub Actions  
Build a Java package using Gradle and publish to GitHub Packages.  
**Configure** Java

**Java with Gradle**  
By GitHub Actions  
Build and test a Java project using Gradle wrapper script.  
**Configure** Java

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 two GitHub repository pages side-by-side. The top page is for the file `gradle.yml` in the `greetings-ci/.github/workflows` directory. The bottom page is for the file `pipeline.yml` in the same directory. Both pages have a header with navigation links: Code, Pull requests, Actions, Projects, Wiki, Security, and a status bar indicating the file is in `main`. Below the header, there are buttons for "Edit new file" and "Preview". The main content area is empty for both files.

TO

8. The second edit is to remove the second job in this workflow since it would require some additional setup. To do this we will just highlight/select the code from line 50 on and hit delete. (*If you have trouble just selecting that code, try starting at the bottom and selecting/highlighting from the bottom up.*) The code to be deleted is highlighted in the next screenshot.

The screenshot shows the `pipeline.yml` file in the `greetings-ci/.github/workflows` directory. The file content is as follows:

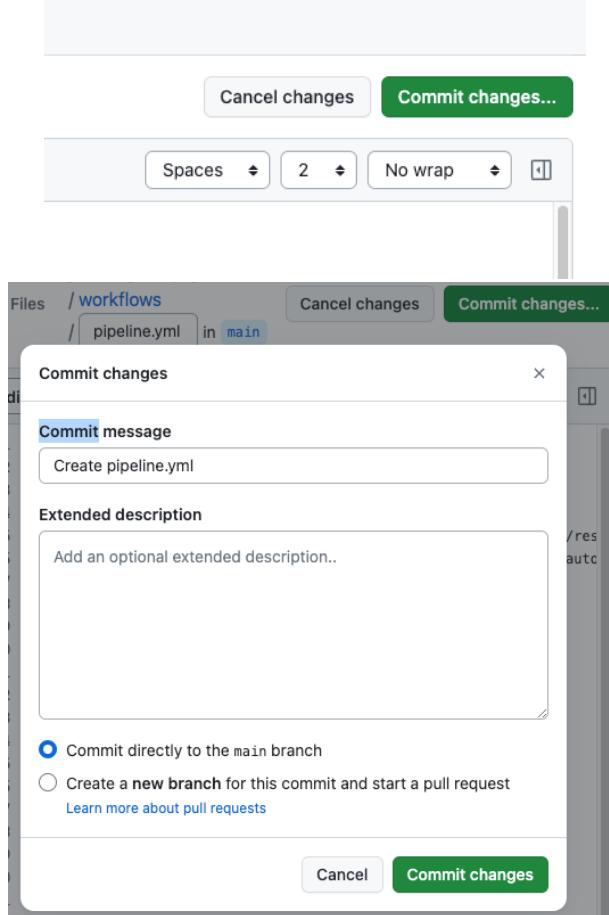
```

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
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
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

```

A red box highlights the code from line 50 to the end of the file, specifically the `dependency-submission:` section and the final three steps. The text "highlight/select, and delete" is overlaid in red on the highlighted area.

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



10. Since we've committed a new file and this workflow is now in place, the `on: push:` event is triggered and the CI automation kicks in.

Click on the **Actions** menu again to see the automated processing happening. (You may have to wait a moment for it to start.)

The screenshot shows the GitHub Actions interface. The top navigation bar includes links for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Actions' tab is currently selected. On the left, there's a sidebar with sections for All workflows, Java CI with Gradle, Management, Caches, and Runners (marked as Beta). The main content area displays 'All workflows' with a search bar for 'Filter workflow runs'. A callout box encourages users to 'Help us improve GitHub Actions' with three quick questions, with a 'Give feedback' button and an 'X' to close it. Below this, a section for '1 workflow run' is shown, listing the 'Create pipeline.yml' run. The run details include the event (push), status (In progress), branch (main), actor (brentlaster), and timestamp (now).

11. 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', 'All workflows', 'Java CI with Gradle' (which is selected and highlighted in blue), 'Management', 'Caches', 'Runners', and a 'Beta' button. The main area is titled 'Java CI with Gradle' and 'pipeline.yml'. It displays a 'Help us improve GitHub Actions' card and a '1 workflow run' section. The run is labeled 'Create pipeline.yml' with a green checkmark. Below the run, it says 'Java CI with Gradle #1: Commit cc93266 pushed by brentlaster'. The run has a status of 'main', was created '1 minute ago', and has been running for '36s'. A red circle highlights the 'Create pipeline.yml' link in the run list.

12. 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. At the top, there are navigation links: Code, Pull requests, Actions (which is underlined in red), Projects, Wiki, Security, Insights, and Settings. Below that, it shows the commit '← Java CI with Gradle' and the workflow name 'Create pipeline.yml #1'. There are buttons for 'Re-run all jobs' and three dots. The main area is titled 'Summary' and shows a list of jobs: 'build' (selected and highlighted in blue) and 'Run details', 'Usage', and 'Workflow file'. The 'build' job has a green checkmark and is circled in red. When expanded, it shows the 'Set up job' step and the 'Run actions/checkout@v3' step, which is also circled in red. The log for this step shows several lines of command-line output, including repository syncing and Git version info.

END OF LAB

## Lab 8: Creating packages

**Purpose:** In this lab, we'll see how to create GitHub packages.

1. We'll continue working in your fork of the **greetings-ci** repo under your primary userid.

In a separate branch named **package**, we have an updated *build.gradle* file and a new Actions workflow file - *.github/workflows/publish-package.yml*. You can switch to the *package* branch and look at those if you want. (You don't need to make any changes.)

```

name: Publish package to GitHub Packages
on:
  release:
    types: [created]
  workflow_dispatch:
jobs:
  publish:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
  
```

2. Let's create a pull request to merge those into *main*.

Go the Pull Requests menu and open a new pull request (via the button) to merge the *package* branch into the *main* branch - **on your fork NOT skillrepos/greetings-ci**. Make sure to set the **base repository = <your repo> main** and **compare = package** in the gray bar so you are merging in the same repo and **NOT** into skillrepos/greeting-ci.

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

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

Choose a Base Repository

Filter repos

Discussions

skillrepos/greetings-ci

gwstudent/greetings-ci

Create pull request

## Comparing changes

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

base: main ▾ ... compare: package ▾

Choose a head ref

Find a branch

Branches Tags

main default

✓ package

There isn't anything to compare.

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

## Comparing changes

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

base: main ▾ ... compare: package ▾ Mergeable. These branches can be automatically merged.

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

Create pull request

-o 1 commit

2 files changed

1 contributor

Commits on Jan 17, 2024

Initial add on package  
Brent Laster committed on Jan 17

321b6bb

Showing 2 changed files with 52 additions and 0 deletions.

Split Unified

.github/workflows/publish-package.yml

```

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

```

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

The screenshot shows a GitHub pull request interface. At the top, there are navigation links: Code, Pull requests (1), Actions, Projects, Wiki, Security, Insights, and Settings. Below this, the title "Initial add on package #1" is displayed, along with a green "Open" button and a message from gwstudent. The main area shows a list of changes: Conversation (0), Commits (1), Checks (1), and Files changed (2). The "Files changed" tab is selected, showing a single file: ".github/workflows/publish-package.yml". The file content is as follows:

```

25 .github/workflows/publish-package.yml
...
1 + name: Publish package to GitHub Packages
2 + on:
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.)

The screenshot shows the GitHub Actions page for the "greetings-ci" repository. On the left, there is a sidebar with "Files" and a dropdown menu set to "main". The main area displays the ".github/workflows/publish-package.yml" file. The file content is:

```

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

```

6. Switch to the **Actions** menu, then select the **Publish package to GitHub Packages** workflow on the left and select the **Run workflow** button that shows up in the blue bar.

The screenshot shows the GitHub Actions interface. At the top, the 'Actions' tab is selected (1). In the main area, there's a workflow named 'Publish package to GitHub Packages' (2). A modal window is open for this workflow, containing a survey for GitHub Actions improvement (3). Below the survey, it says '0 workflow runs'. A tooltip indicates the workflow has a 'workflow\_dispatch' event trigger. On the right, there's a 'Run workflow' button (4) which is highlighted with a red circle.

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 the GitHub repository 'greetings-ci' (1). The 'Code' tab is selected. In the bottom right corner of the repository page, there is a 'Packages' section (2) which is highlighted with a red circle. It lists one package: 'org.gradle.sample.greetings-ci'.

7. You can also see the new package in your profile area. Click on your picture in the upper right, then select **Your profile** and then the **Packages** tab.

The screenshot shows a GitHub profile page for 'brentlaster'. In the top navigation bar, the 'Packages' tab is highlighted with a red circle and labeled '2'. On the right, a sidebar menu is open, showing a list of options. The 'Your profile' option is highlighted with a red circle and labeled '1'. Below it, the 'Packages' option is also highlighted with a red circle and labeled '3'. The main content area shows a circular profile picture of a man, Brent Laster, and a list of packages under the 'Packages' tab, with one package 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.

The screenshot shows the GitHub package page for 'org.gradle.sample.greetings-ci 1.1'. The page includes sections for 'Code', 'Pull requests', 'Actions', 'Projects', 'Wiki', and 'Details'. The 'Details' section shows the package was published on January 05, 2024. The 'Assets' section lists several files, with 'greetings-ci-1.1.jar' circled in red.

END OF LAB

## Lab 9: Creating a release

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

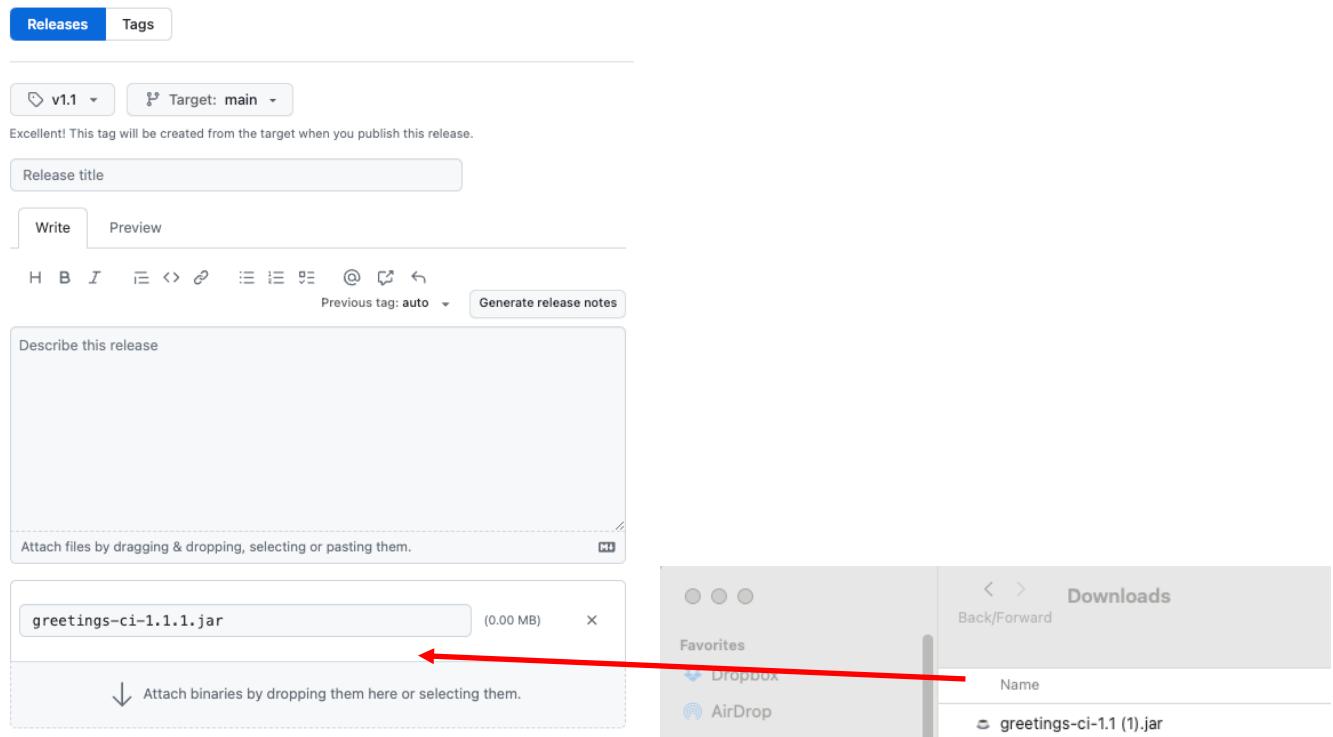
1. On the **Code** tab of the `greetings-ci` repo, on the right-hand side, find the **Releases** section and click on the **Create a new release** link. (You can also go directly to the page by going to <https://github.com/<github-userid>/greetings-ci/releases/new>.)

The screenshot shows the GitHub repository page for 'greetings-ci'. The 'Code' tab is active. On the right side, there is a 'Releases' section. Inside this section, there is a button labeled 'Create a new release' which is highlighted with a red circle.

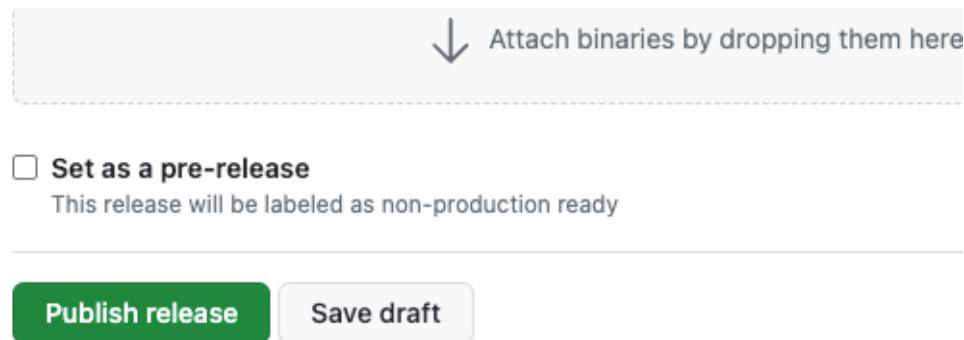
2. We need to create a tag on the repo before we create a release. Click on the **Choose a tag** dropdown and enter `v1.1` (or some other name if you prefer) for the tag name. Then click on the **+ Create new tag: v1.1 on publish** line.

The screenshot shows the GitHub repository page for 'greetings-ci' with the 'Releases' tab selected. A dropdown menu is open for 'Choose a tag', showing the option 'v1.1'. Below the dropdown, there is a button labeled '+ Create new tag: v1.1 on publish'.

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.



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 a GitHub release 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 title 'Initial release' is displayed with a 'Latest' badge. A message from 'brentlaster' states 'released this now'. The description area contains the text 'Initial release of content for 1.1.'. Under the 'Assets' section, there is a table listing three items:

	<a href="#">greetings-ci-1.1.jar</a>	1006 Bytes 2 minutes ago
	<a href="#">Source code (zip)</a>	21 minutes ago
	<a href="#">Source code (tar.gz)</a>	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 main page of the GitHub repository 'greetings-ci'. The top navigation bar includes Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository card for 'greetings-ci' (Public) shows it was forked from 'skillrepos/greetings-ci'. The 'About' section describes it as a 'Simple starter repo for CI/CD with GitHub Actions'. The 'Activity' section shows 0 stars, 0 watching, and 140 forks. On the right side, there is a sidebar with sections for 'About', 'Releases', and 'Packages'. The 'Releases' section shows one release: 'Initial release' (Latest), which was released 1 minute ago. This release is circled with a red oval.

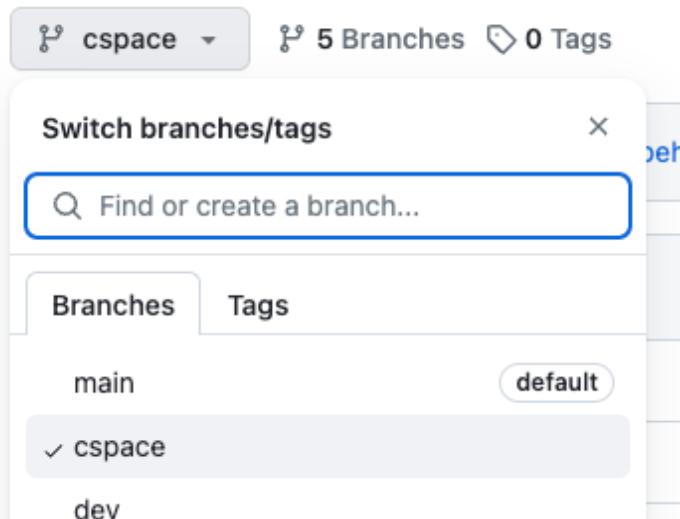
7. You can click on that if you want to see details about the release (same as output in step 5).

END OF LAB

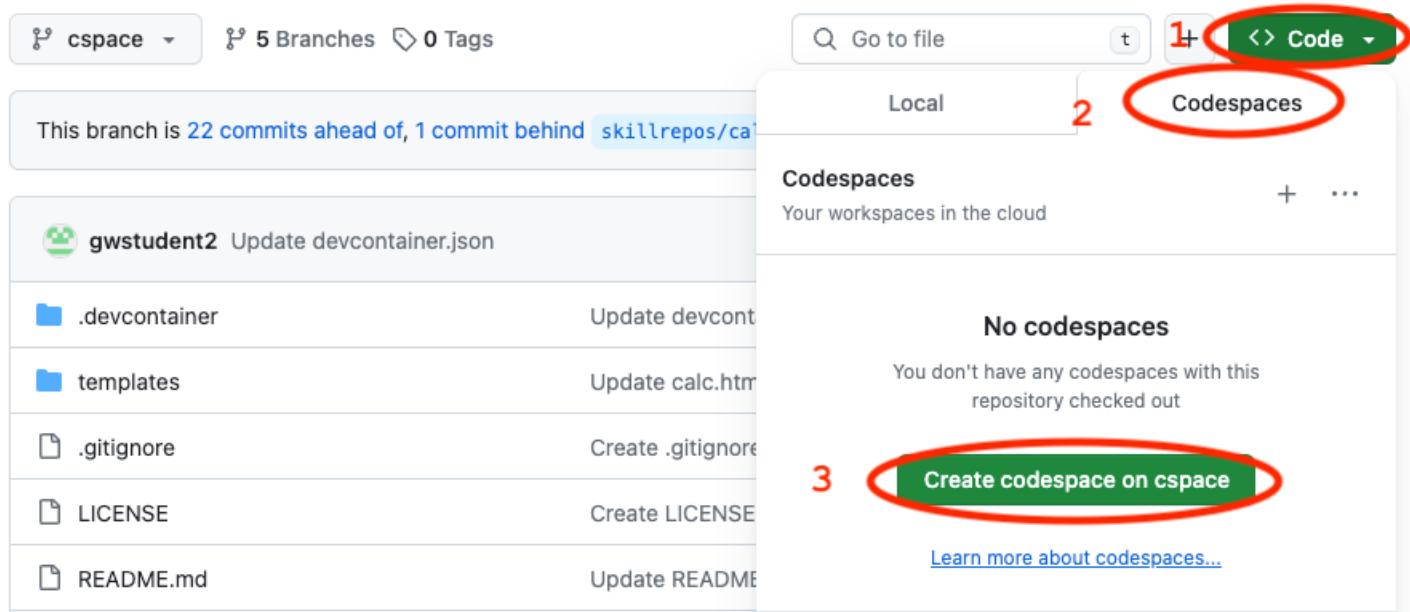
## Lab 10: Working with Codespaces

**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.



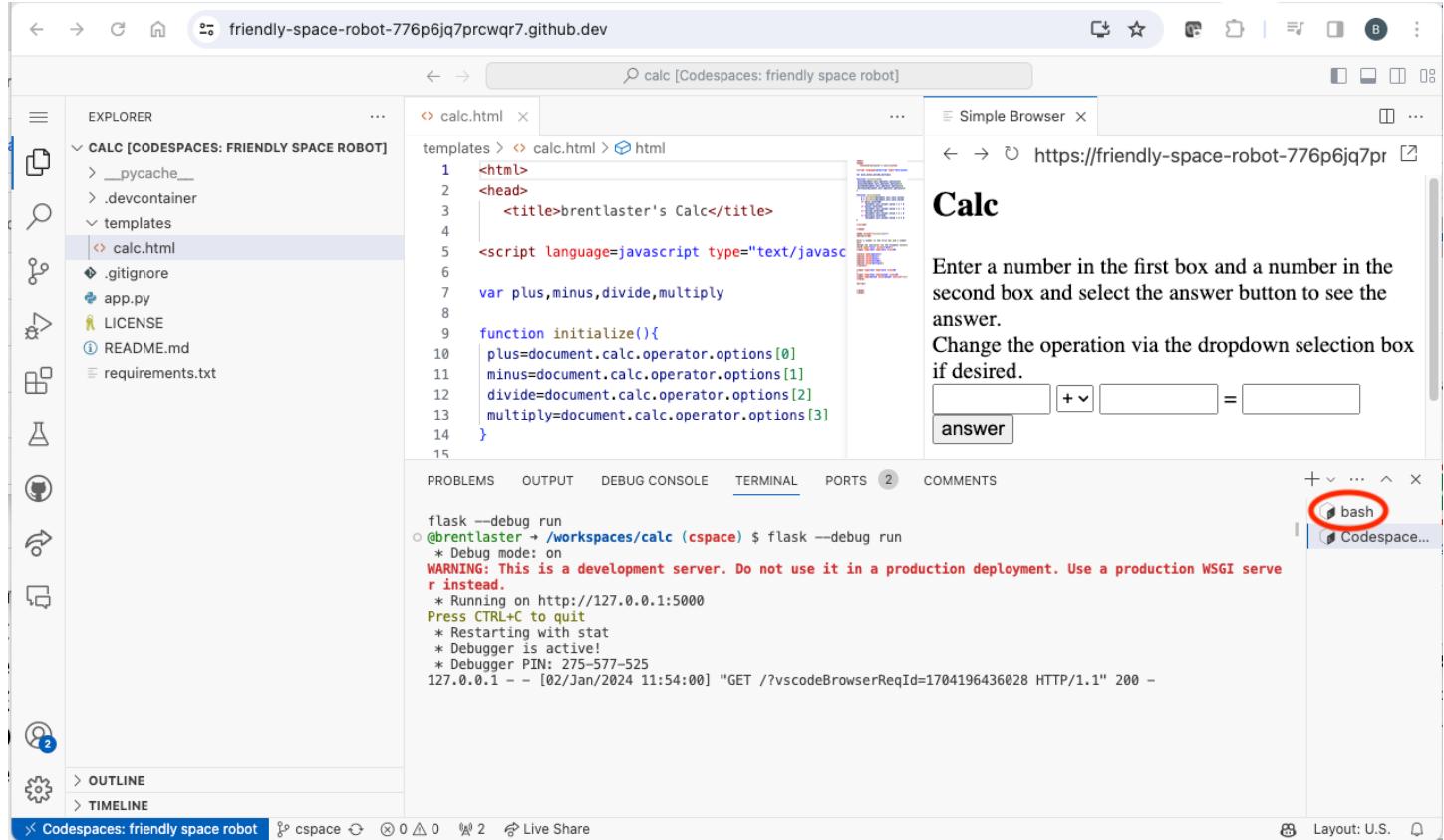
2. Click on the **<> Code** button, then select the **Codespaces** tab, and then select **Create codespace on cspace**.



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.

The screenshot shows the PACE Robotic Platform interface. On the left, there's a sidebar with "PACE ROBOT" and a "M" icon. The main area has tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", "PORTS" (with a value of 2), and "COMMENTS". A search bar at the top right contains the text "Simple Brow".

**calc.html**

```

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

```

**Calc**

Enter a number in the first box and a number  
Change the operation via the dropdown selecti  
answer

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 PACE Robotic Platform interface. The "Simple Brow" tab is active. The browser window displays the URL <https://friendly-space-rob>. The page title is "Brent's Calc". The content of the page includes instructions: "Enter a number in the first box and a second box and select the answer button", and "Change the operation via the dropdown selecti".

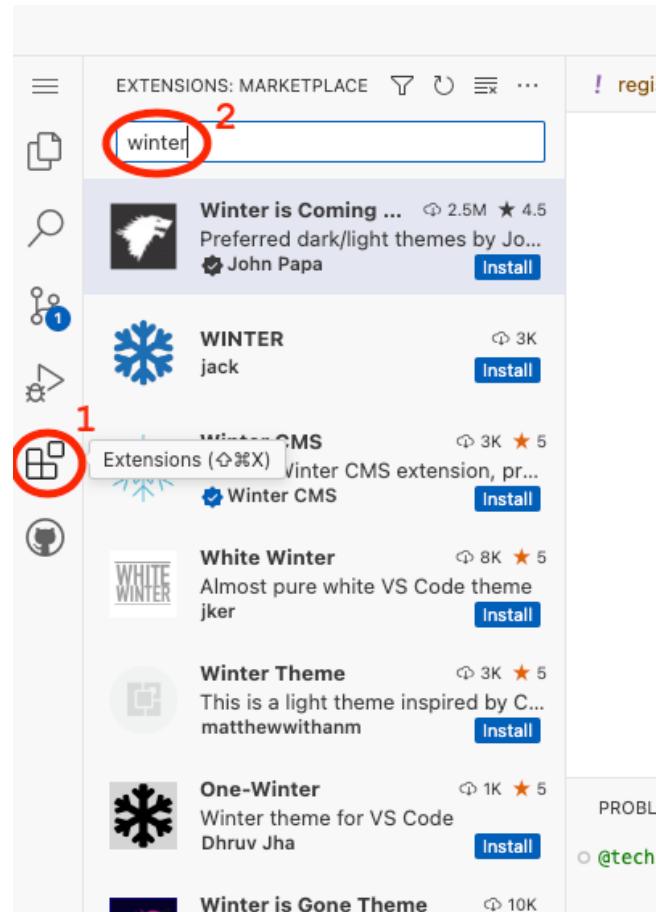
**calc.html**

```

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

```

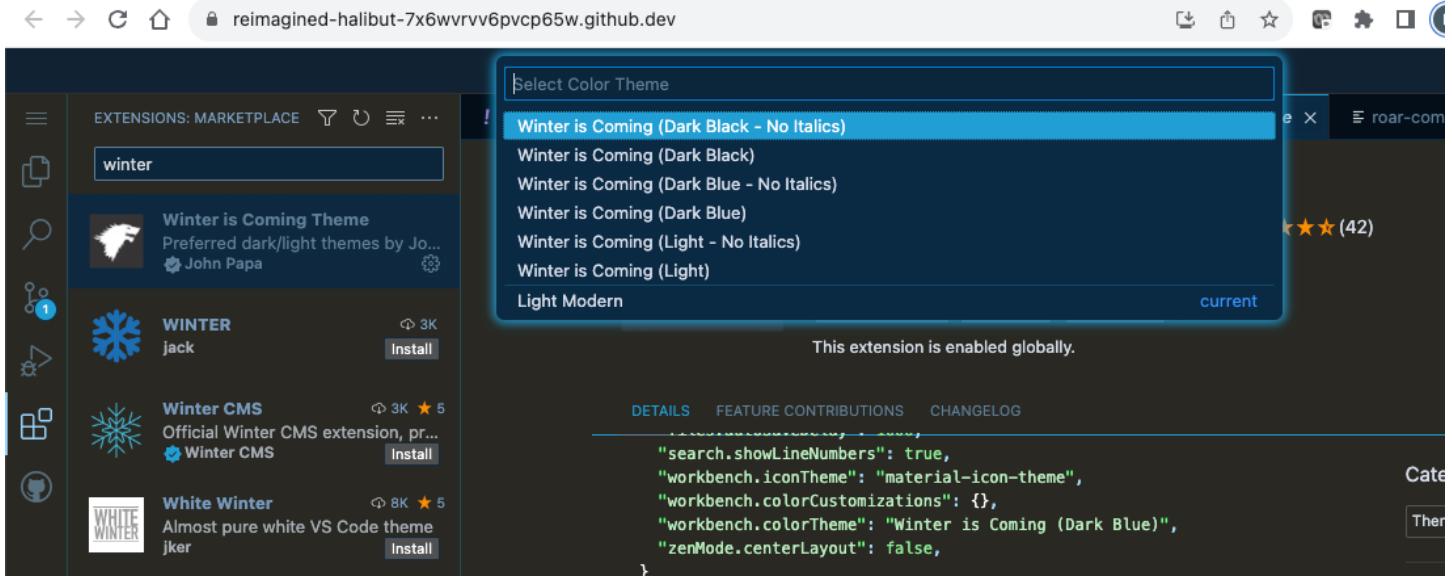
6. Next, let's see how to modify our codespace environment. We'll install an extension for the color theme. For practice, we'll use the "Winter is Coming" theme. Click on the *Extensions icon* (#1 in figure below), then in the *search bar* type in "winter" to quickly find the extension (#2).



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.

The screenshot shows the VS Code Marketplace interface. On the left, the search results for 'winter' are displayed. In the center, the details page for the 'Winter is Coming Theme' extension is shown. The extension has a rating of 4.5 stars and over 2.5 million installs. The details page includes tabs for DETAILS, FEATURE CONTRIBUTIONS, and CHANGELOG, with the DETAILS tab currently selected. Below the tabs, there's a section for 'Usage' and 'Installation' with step-by-step instructions. To the right, there are sections for 'Categories' (Themes), 'Extension Resources' (Marketplace, Repository, License, John Papa), and 'More Info' (Published: 2017-10-18).

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.

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.

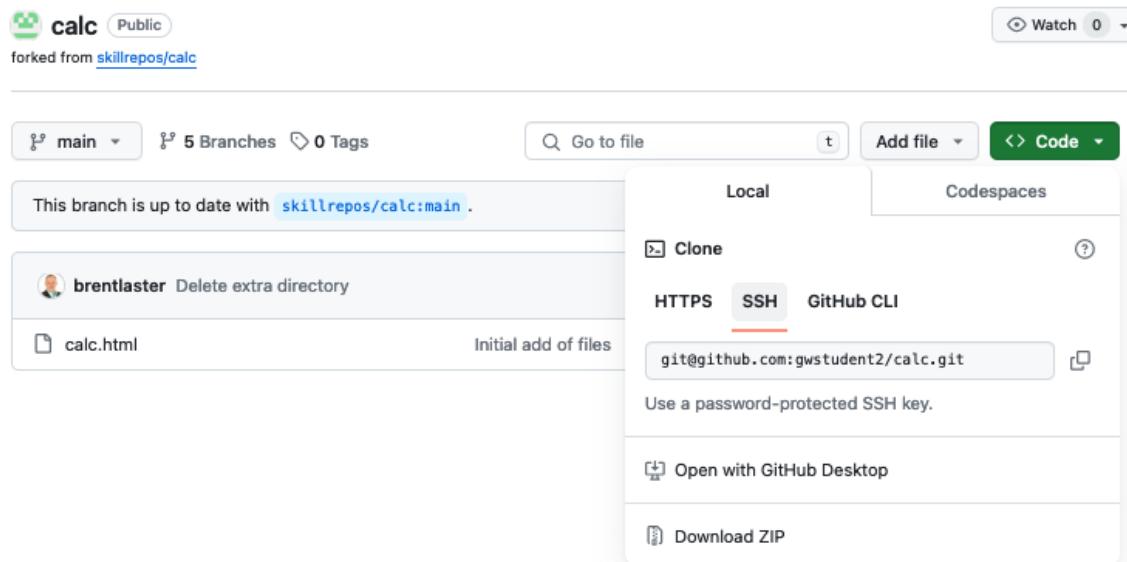
(Note: If you prefer to disable the global credentials helper entirely, you can try

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

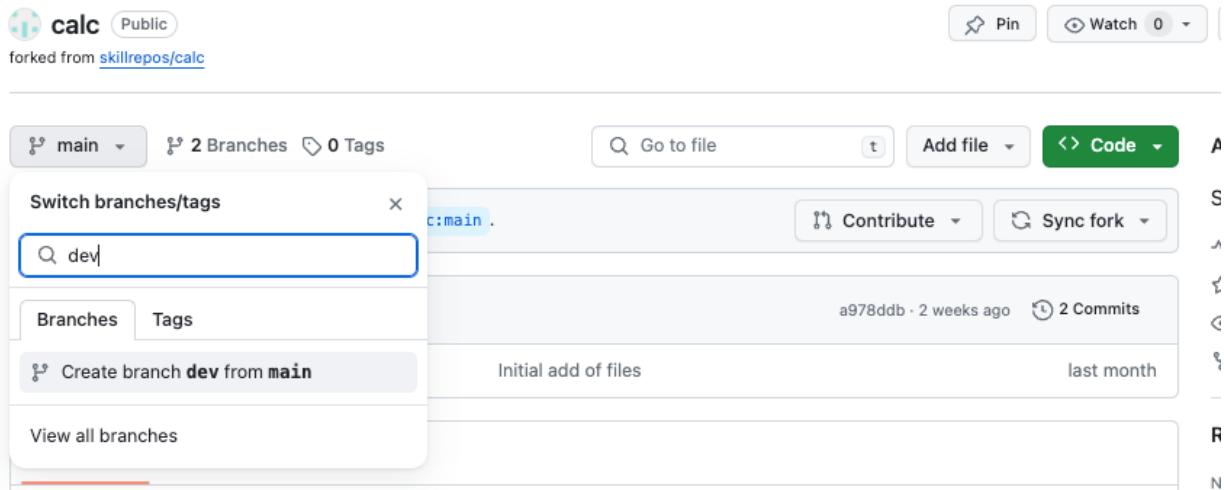
This may or may not work depending on if you have access to do this.)

**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.

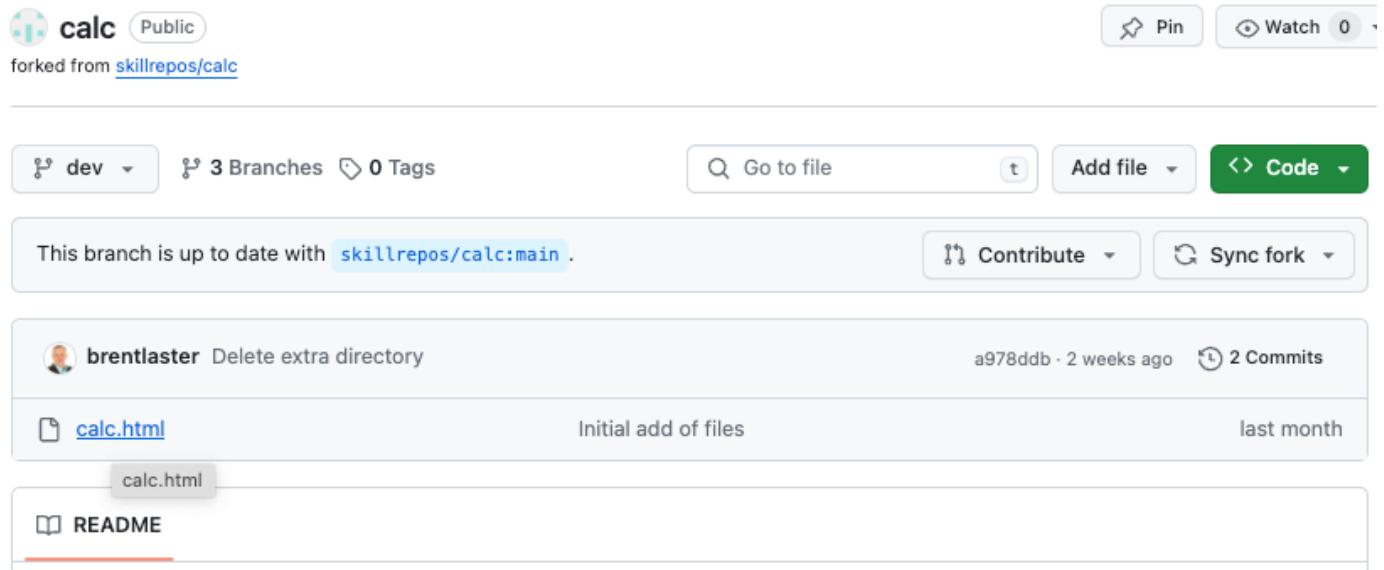
If you go this route, when you get the remote URL from the browser, select the SSH tab.



**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.



Click on the pencil icon to edit the file.

Brent Laster Initial add of files

a1d22c4 · last month History

**Code** Blame 59 lines (46 loc) · 1.27 KB Code 55% faster with GitHub Copilot

```

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

```

Make the changes noted in Lab 1 in the file.

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

github.com/gwstudent/calc/edit/dev/calc.html

gwstudent / calc

Code Pull requests Actions Projects Wiki Security Insights Settings

calc / calc.html in dev

Cancel changes Commit changes...

Edit Preview

Commit changes

Commit message: Update calc.htm|

Extended description: Add an optional extended description..

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

Learn more about pull requests

Cancel Commit changes

```

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.an);
18       b = parseInt(document.calc.bn);
19       if (plus.selected)
20         document.calc.an=b+a;
21       if (minus.selected)
22         document.calc.an=a-b;
23       if (divide.selected)
24         document.calc.an=a/b;
25       if (multiply.selected)
26         document.calc.an=a*b;
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/import'. The main heading is 'Preparing your new repository' with the 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 bar with the text 'Your import will begin shortly...' underneath it.

## 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.

[github.com/new](https://github.com/new)

New repository

Type  to search

**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 \*** sec-demo **1**

sec-demo is available.

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

**Description (optional)**

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

**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

ⓘ 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 search bar and navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, there's a 'sec-demo' button and a 'Public' link. On the left, there's a 'Start coding with Codespaces' section with a 'Create a codespace' button. On the right, there's a 'Add collaborators to this repository' section with a 'Invite collaborators' button. The main content area has a heading 'Quick setup — if you've done this kind of thing before'. It shows three connection options: 'Set up in Desktop' (disabled), 'HTTPS' (selected and highlighted with a red box), and 'SSH'. Below this, it says 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)'. There's also a '...or create a new repository on the command line' section with a block of Git commands. A red box highlights the 'git remote add origin git@github.com:brentlaster/sec-demo.git' line. Another red box highlights the entire '...or push an existing repository from the command line' section, which contains the same Git command.

```

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

```

```

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.