A guide on contributing to SideFX EDU

If you're reading this, you might be thinking about submitting a tool to SideFX EDU, or even better, you've submitted something already and you're looking for a refresher.

If you are in doubt as to how to proceed please contact the Education & Training Department (education@sidefx.com) and we can help you out.

Before we look at what it means to contribute to SideFX EDU, we want to make it clear that it is not an experimental graveyard. This means that if you intend on contributing as SideFX staff you intend on maintaining the tool (or delegate the responsibility) if issues arise due to updates to Houdini and so forth. If you are not SideFX staff but still want to contribute, we recommend you get in touch with us before going through this guide.

We also want to make sure that SideFX EDU provides quality experiences to the instructors and learners; and user experience is key to this. Many of the tools wrap up common concepts and workflows into high level tools so learners find them valuable as-is for learning and experimenting.

Depending on the type of contribution, there are different types of requirements. This document will try to explain most of them. If what you are trying to do is not described, please contact us.

Finally, SideFXEDU is open source. This means we expect contributors to provide us the source code / unlocked assets to share with the community. You should not expect any direct returns from your contribution, except for being credited in the release journal on the forum.

Contribution Checklist

HDAs

1) The tool definition must use the following formatting with the **'edu' namespace**, i.e.:

`edu::my_tool_name::1.0`

2) The tool must be put in the right **tab submenu**. You can set this in the *Type Properties > Interactive > Shelf Tools > Context* tab. Valid values are Education/<category_name>, i.e.:

Education/Rendering, Education/Math, Education/Utility...

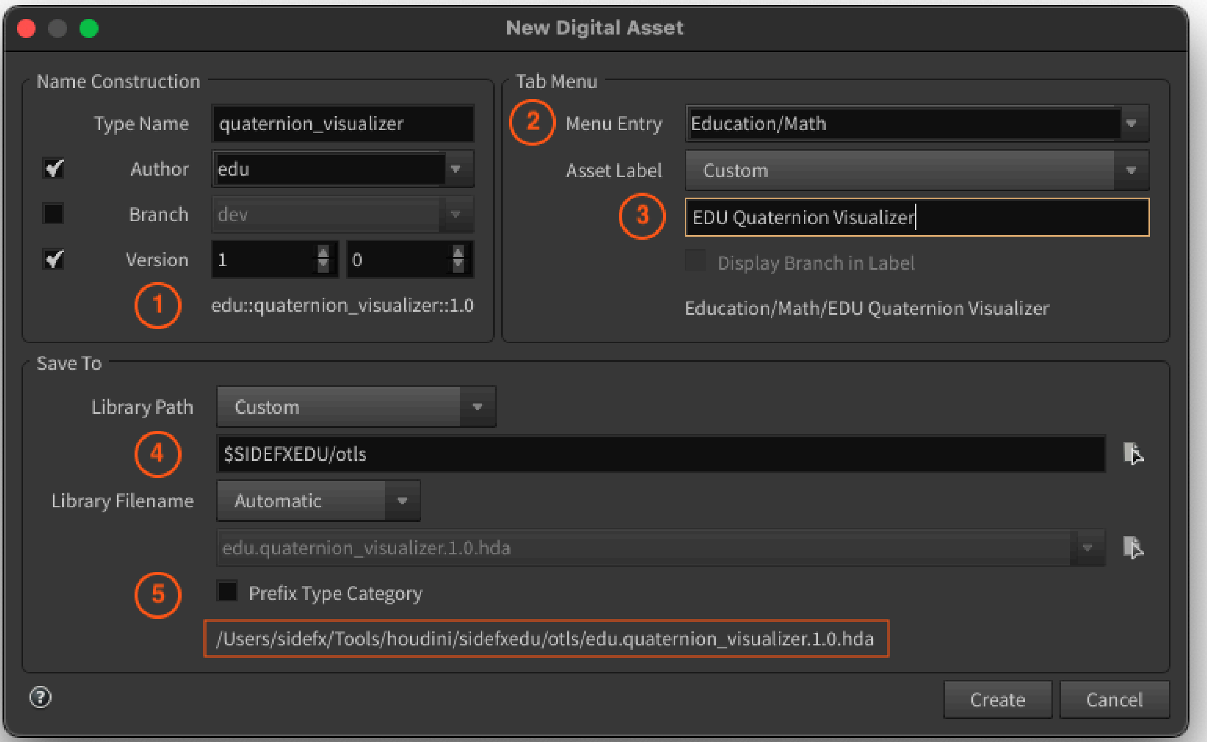
3) The tool label must use **EDU as a prefix**, i.e.

EDU My Tool Label

4) You can store the definition of the HDA anywhere on disk, but if you want it to be loaded with the SideFX EDU package, you should put it inside the /otls folder, inside the SideFX EDU package folder, i.e.:

`$SIDEFXEDU/otls`

5) The definition must **not be prefixed** with the type category


New Digital Asset

Name Construction
- Type Name: quaternion_visualizer
- [✓] Author: edu
- [ ] Branch: dev
- [✓] Version: 1 0
- 1 edu::quaternion_visualizer::1.0

Tab Menu
- 2 Menu Entry: Education/Math
- Asset Label: Custom
- 3 EDU Quaternion Visualizer
- [ ] Display Branch in Label
- Education/Math/EDU Quaternion Visualizer

Save To
- Library Path: Custom
- 4 $SIDEFXEDU/otls
- Library Filename: Automatic
- edu.quaternion_visualizer.1.0.hda
- 5 [ ] Prefix Type Category
- /Users/sidefx/Tools/houdini/sidefxedu/otls/edu.quaternion_visualizer.1.0.hda

- The tool **must have an icon**; the icon must look similar in style and color to the icons used in other tools found in EDU. This icon must also be of **type SVG**, and not a rasterized image to ensure scalability. Refer to the Icon section for some examples and the color palette.

- The inputs and outputs of the tool have been given a **proper tooltip**. You can set these in the *Type Properties > Input/Output* tab of the tool.

- If your HDA is a SOP or a TOP, you need to make sure you **have an output node** at the bottom of your network. This ensures that regardless of where the display flag is set that it produces the same output.

- On the Scripts tab, add the OnCreated() function and paste the following code in it:

```
# This code changes the name of the node and adds a comment
# So the user is well aware this is an Educational tool
node = kwargs['node']
name = node.name()
node.setName('edu_'+name, unique_name=True)
node.setComment("Education tool")
node.setGenericFlag(hou.nodeFlag.DisplayComment, True)
```

- The tool needs documentation. To create documentation the right way, please refer to the Documentation section below.

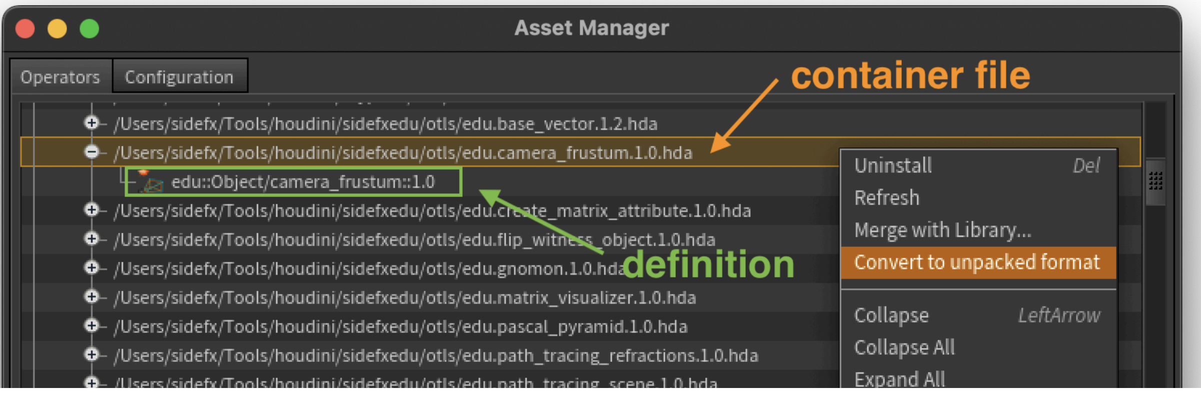
- The tool needs a **simple demo scene inside the *\/hip* folder** where the functionality of the tool is demonstrated. See examples inside the SideFX EDU repo, under the */hip* folder: `$SIDEFXEDU/hip/`.
Naming convention is as follows:

`edu_my_tool_name_demoscene.hip`

- [Optional but appreciated] **A short video inside the */help/movies* folder** where the functionality of the tool is demonstrated, using the demoscene mentioned above. See examples inside the SideFX EDU repo, under the */help/movies* folder: `$SIDEFXEDU/help/movies`.
Naming convention is as follows:

`edu_my_tool_name_movie.mp4`

- All tools need to be in the **expanded format** for diffing on Github.
  - Right-click on an instance of your HDA and choose *"Show in Asset Manager"*.
  - The Asset Manager opens with the HDA definition selected.
  - Select the container (i.e. the file containing the definition) and right-click on it.
  - Select *"Convert to unpacked format"*.



/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.base_vector.1.2.hda
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.camera_frustum.1.0.hda
edu::Object/camera_frustum::1.0
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.create_matrix_attribute.1.0.hda
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.flip_witness_object.1.0.hda
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.gnomon.1.0.hda
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.matrix_visualizer.1.0.hda
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.pascal_pyramid.1.0.hda
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.path_tracing_refractions.1.0.hda
/Users/sidefx/Tools/houdini/sidefxedu/otls/edu.path_tracing_scene.1.0.hda

Please refer to the following page for more information on how to expand your HDAs: Working with files and assets as text.

Misc tips

If you are making significant changes to an HDA, please bump the version of the tool, i.e. `edu::gnomon::1.0` becomes `edu::gnomon::2.0`. This ensures backwards compatibility, and ensures we dont break any automatic scripts users have made. It is recommended to use the new HDA tool to create and manage HDAs. This makes namespaces and labeling very easy.

If you don't have Github access, you should send the following in a .zip to the education@sidefx.com:

- [ ] HDA compliant with the above checklist
- [ ] Icon in SVG format
- [ ] Help documentation in txt file
- [ ] Demoscene .hip (as simple as possible, with comments and stickies where appropriate)
- [ ] A video tutorial no longer than 5 mins (Optional but much appreciated!)

Icons

- An Adobe Illustrator template file is available inside `$SIDEFXEDU/docs/sidefxedu_icons.ai`.

- The .svg file icon must be of **ratio 1:1**. Houdini expects icons to have a 1:1 aspect ratio, so make sure your SVG has that to prevent any weird distortion.

- The colors we typically use are: Black `#000000` / White `#ffffff` / Orange `#ff6600`

Documentation

HDAs

Any tool submitted should have a basic pass of tooltips added to the helpcard. Assuming you've installed SideFXEDU, you should be able to generate a help card template with python in Houdini.

Tip: set the icon before running the script below, so it can add it automatically to the page.

- Select an instance of your HDA and run this code in the Houdini Python Shell to create a template page:

```
from edu_utils import create_node_help_auto
create_node_help_auto(hou.selectedNodes()[0])
```

- A file named after the HDA should be created inside `$SIDEFXEDU/help/nodes/<HDA_TYPE>/`.
For instance for the `edu::gnomon::1.0` HDA, the help file will be created here:

```
$SIDEFXEDU/help/nodes/sop/edu--gnomon-1.0.txt
```

- Make sure to add the `#sidefxedu` tag at the beginning of the page, as well as any other tag you deem useful. These tags can be used on the SideFX EDU help page to help the user filter and search for nodes:

```
#tags:   education, rendering, math, utility
```

- Make sure the title of your help page also prefixes the title with EDU.

- Use this Wiki Markup Reference page as a reference on how to write the node's help page.

- Of course, once you’ve finished writing the help page, check that it works by pressing the help button on the parameter interface of the tool (or hover the parameters to see the tooltips).

Note: Currently the icon will not display on the help page. This is a bug that will be addressed in the near future.

Github

Write Access

If you have writing permissions to the SideFX EDU Github repository, there are a **few rules** we have defined for commit messages to **ensure our automatic forum updates** work properly, and for everyone's benefit.

Do NOT commit to the repo if you have access simply because you are staff. Before adding any new tools or making changes, **get in touch with the** SideFX Education team. The preferred method of submitting content to SideFX EDU as an external (non-SideFX Education team) contributor is to submit a **PullRequest**. This video explains how.

For community users we recommend you get in touch with us instead, since the repo is writable only for SideFX staff. The reason for this is to ensure we ship EULA compliant and commercially licensed Houdini assets only. This ensures everyone is able to use it. We will upgrade any non-commercial or Indie assets for free for you, should we decide to ship your contribution.

Commit Messages

All commit messages should start with one of the following:
[NEWTOOL] ****tool_name**:** - The first time a new tool is added to the repo.
[UPDATE] ****tool_name**:** - Any changes to a tool.
[BUGFIX] ****tool_name**:** - If the change is specifically a bugfix
[IGNORE] ****tool_name**:** - Any commits that don't need to be seen by the general public.

The commit message should also **use asterisks around the name** of the thing you're describing. This will ensure that it will print in bold on the forum post.

Examples for commit messages would be

*[NEWTOOL] *Oscilloscope*: this node helps visualizing a math equation in the viewport*

*[UPDATE] *Oscilloscope*: added new equation presets*

*[BUGFIX] *Oscilloscope*: fixed switch selection on type menu*

*[IGNORE] *Oscilloscope*: cleaned and reorganized the network inside the HDA*

[IGNORE] Added the SideFX sidefxEDU tag to all the docs

Working with forks and pull requests

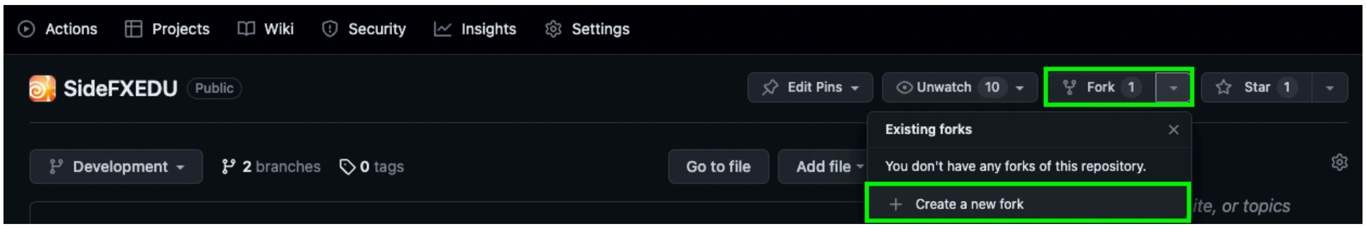
If you do not have writing permissions to the SideFX EDU Github repository, do not worry, you can still contribute to the project, thanks to pull requests, provided you first fork our repository.
For a general overview of the process: Collaborating with pull requests - GitHub Docs

Forking the SideFX EDU repository

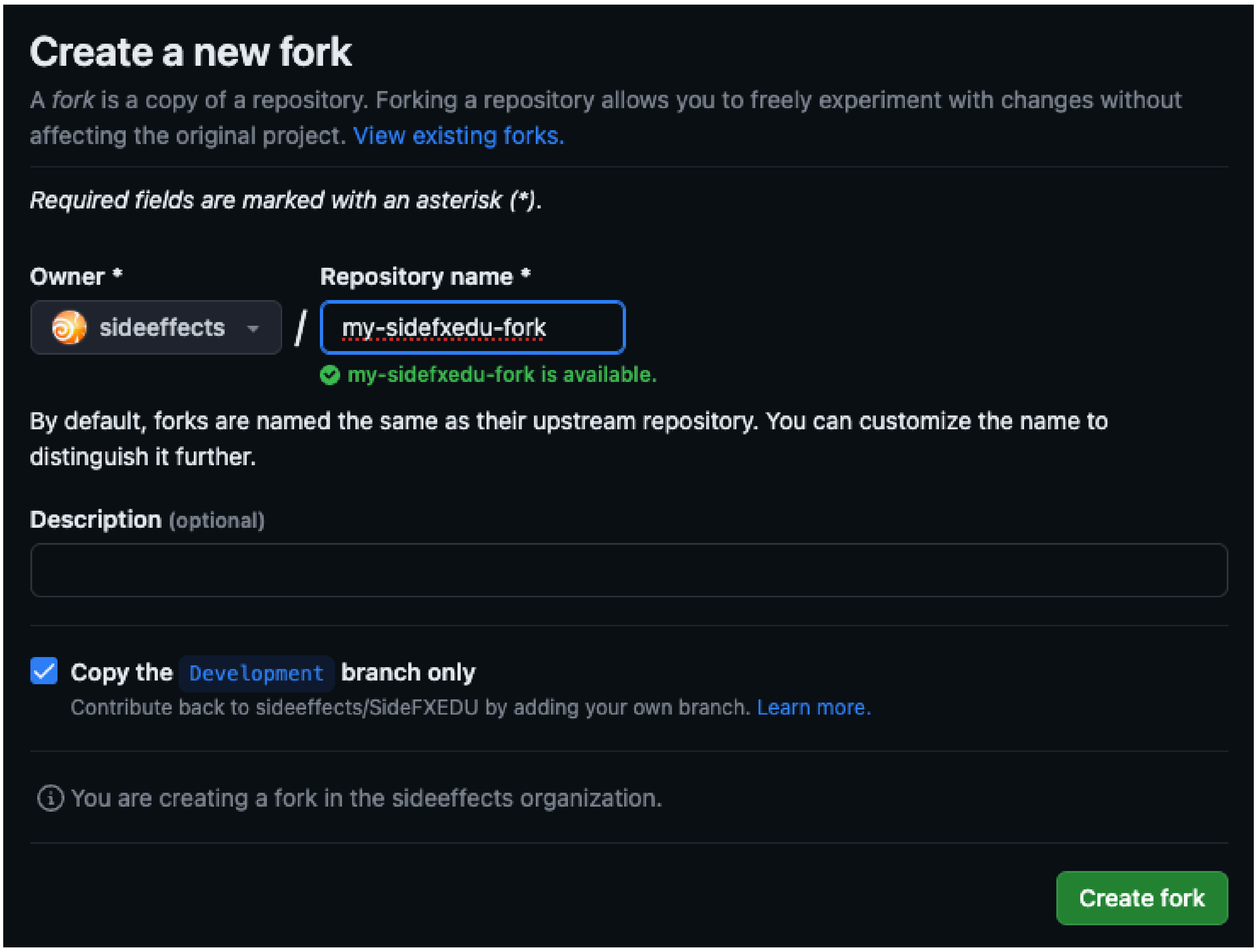
A fork is a new repository that shares code and visibility settings with the original "upstream" repository. Forks are often used to iterate on ideas or changes before they are proposed back to the upstream repository, such as in open source projects or when a user does not have write access to the upstream repository.

For more information: Working with forks - GitHub Docs

First go to the main SideFXEDU github page, and clik the Fork



Then:



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 * sideeffects / Repository name * my-sidefxedu-fork

my-sidefxedu-fork is available.

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

Description (optional)

[✓] Copy the `Development` branch only
Contribute back to sideeffects/SideFXEDU by adding your own branch. Learn more.

You are creating a fork in the sideeffects organization.

Branching

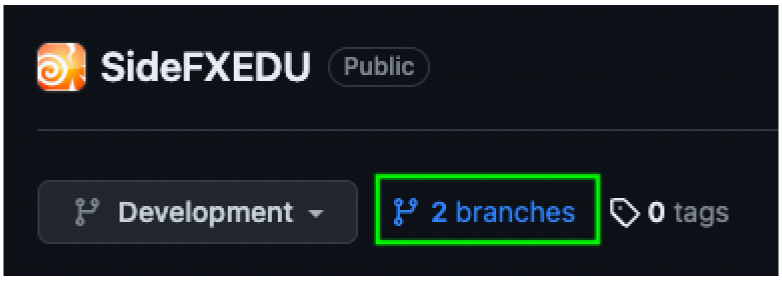
Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository.

Use a branch to isolate development work without affecting other branches in the repository. Each repository has one default branch, and can have multiple other branches. You can merge a branch into another branch using a pull request.

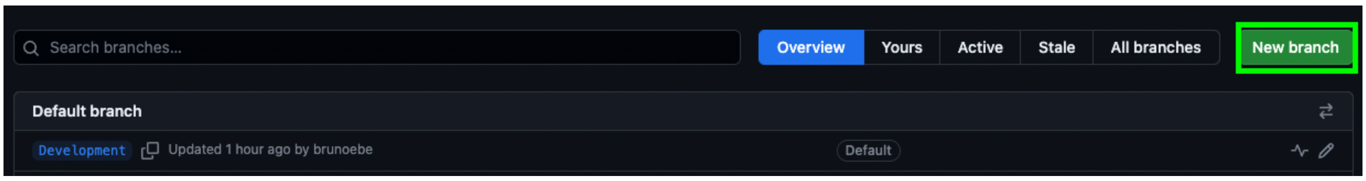
The SideFX EDU repository has several branches, including a "main", which is the current version of Houdini, some older versions, and various others. If you are actively developing your tool but do not want it to be live yet, you can use branches for this purpose. This allows you to make a copy of the repo and make your changes without having to worry about things breaking. Once complete, an admin of the repo can merge your branch into Main once approved.

For more information: About branches - GitHub Docs

Click on the 'x branches' link



Then click on the 'New branch' button



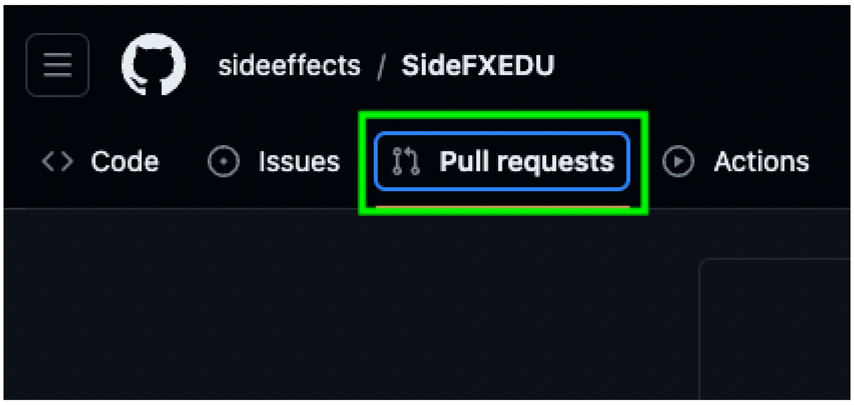
Pull requests

You can create a pull request to propose changes you've made to a fork of an upstream repository.

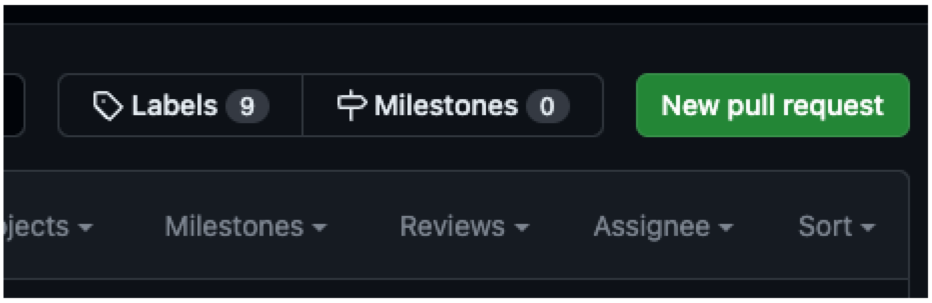
Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

For more information: Creating a pull request from a fork - GitHub Docs

Click on 'Pull requests'



Then click on 'New pull request'


Labels 9
Milestones 0
New pull request
jects
Milestones
Reviews
Assignee
Sort