

GitHub Collaboration Cheat Sheet

Bootcamp Hackathon 1 | 10-12 January 2024



Part 1

- ✓ Branching vs Forking
- ✓ Creating The 'Upstream' Repository
- ✓ Creating A Project Board

Stand down call
Mon-Thur

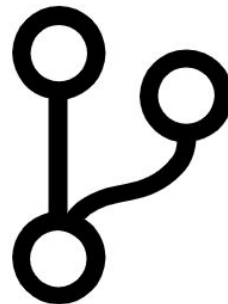


Branching vs Forking

Before we begin, here is a basic overview on the most common methods we use for working collaboratively using Git and GitHub.

Branching in an individual repository:

- In a single repository, branching allows you to create separate lines of development. You can work on a new feature, bug fix, or experiment without affecting the main or master branch.
- It's a way to keep your changes isolated until you're ready to merge them back into the main codebase.
- Great for personal projects or small teams where everyone has access to the same repository.
- Branches can be created via git commands in the Terminal.
- Throughout the first Hackathon, we'll just use the single 'main' branch of our repositories.

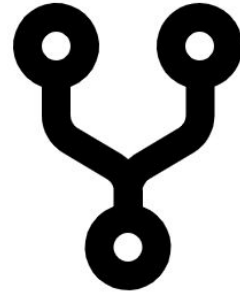


Branching vs Forking (Continued)

Now let's take a look at using forks to work collaboratively.

Upstream repository and Forking

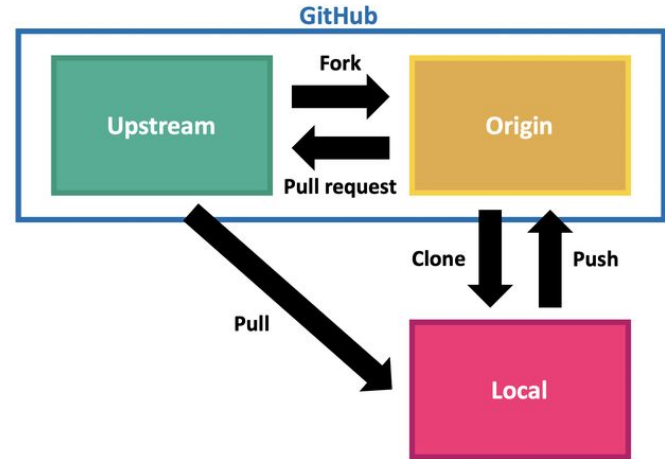
- An upstream repository is usually the original repository from which your project was forked. Forking creates a copy of the repository under your account.
- Forking is commonly used in open-source projects where contributors want to make changes without directly altering the original project.
- You can then create a GitPod Workspace from your forked repository, make changes, and create a pull request to propose those changes to the upstream repository.
- You Can continue to contribute to the upstream repository via the forked repository as long as you keep your fork up to date by pulling the latest upstream version, using the command: ***git pull upstream main***



Upstream Repository & Forking Workflow

For our first Hackathon, we'll use the forking collaboration method. Let's examine the accompanying image. As we can see, we have an upstream repository, a forked (origin) repository and a local (GitPod) workspace. Here's a breakdown of this process:

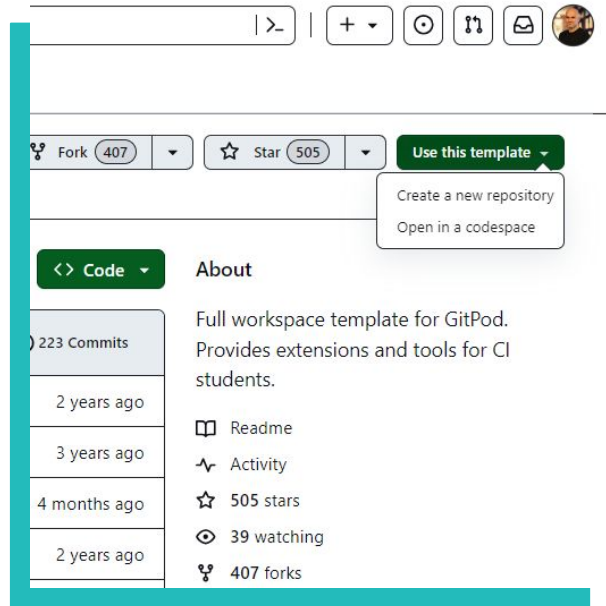
- Create an upstream repository
- Create duplicates of the upstream repository by forking
- Create a new GitPod workspace from the forked repository
- Create, edit and push our changes to the forked repository.
- Request our code be “pulled “ into the upstream repository via pull requests.
- As we can see in the image, we can then refresh the code in our GitPod workspace by pulling directly from the upstream repository to our GitPod Workspace:
(git pull upstream main)



Creating The Upstream Repository

The **Hackathon Team Lead/Scrum Master** will create a **new** repository (Not a Codespace) using the [Code Institute Gitpod Template](#).

This repository will be the one your team will submit to the judging panel for review and will contain the code of your successfully deployed Hackathon project.



Creating The Upstream Repository (Continued)

Something to keep in mind when naming your repositories is that the name should reflect the title of your application or business. For example: *"campbells-cafe"*, *"reacher-creatures"*, *"precious-paws"* etc.

We should **try to avoid** naming our repositories in a functional way eg: *"ci-hackathon-project-1"* or *"Individual Project for Assessment"* Remember. You are building a portfolio of work that you can show off to potential employers and clients.

Add a short description that includes the purpose of your site and it's target audience.

Make sure your repository is set to "Public" **This is extremely important.**

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

 Code-Institute-Org/gitpod-full-template ▾

Start your repository with a template repository's contents.

☐ Include all branches

Copy all branches from Code-Institute-Org/gitpod-full-template and not just the default branch.

Owner *

 davidcalikes-codeinstitute ▾

Repository name *

reacher-creatures

✓ reacher-creatures is available.

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

Description (optional)

The Official Website of The Lee Child Appreciation Society

☒

 Public

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

☐

 Private

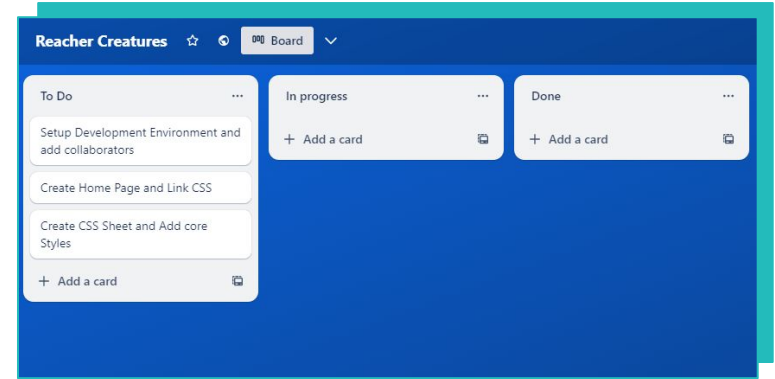
You choose who can see and commit to this repository.

Creating A Project Board

For Hackathon 1 we should create a project board to manage the user stories and development tasks required to build our applications.

We can use **GitHub Projects** or **Trello**

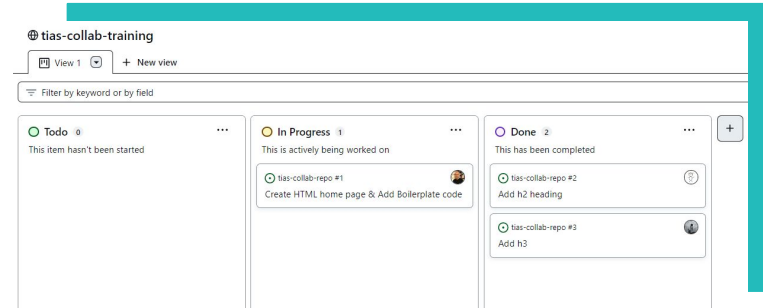
We must make sure our boards are **visible to the public** (Anyone with the link can view) and that they contain at least three sections. **Make sure every team member has editor access to the board.**



To Do

In Progress

Done



Part 2

- ✓ Forking The Upstream Repository
- ✓ Creating The GitPod Workspace
- ✓ Committing Changes to the Fork

Stand down call
Mon-Thur



Forking The Upstream Repository

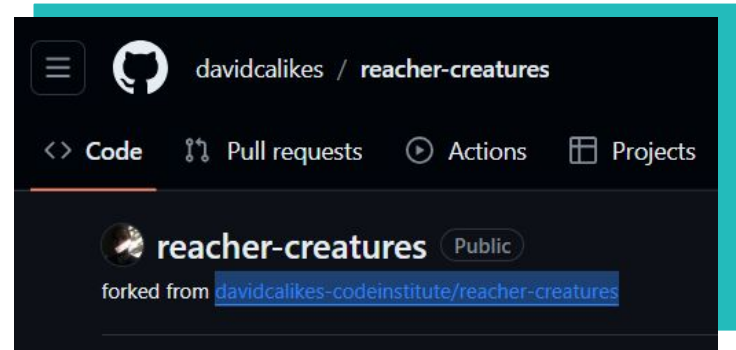
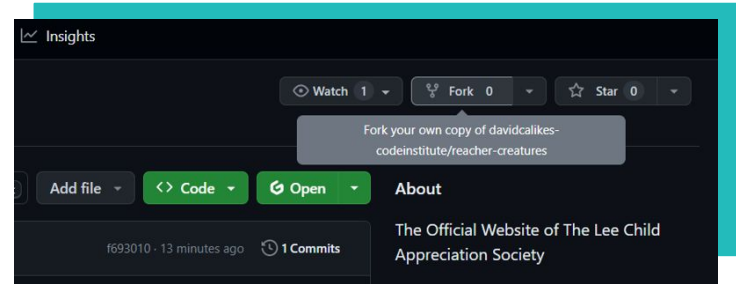
The Hackathon Team Members/Collaborators will duplicate the 'Upstream' repository by forking via the "Fork" tab in GitHub (See Top Image)

This repository will now be assigned to the team members individual GitHub account and it will be among their repositories.

If you have forked correctly, you will see the name of your fork and the upstream repository will be linked directly underneath. (See Below Image)

You can now create a GitHub workspace (see next page) and start working on any issues/tickets that have been assigned to you on the project board.

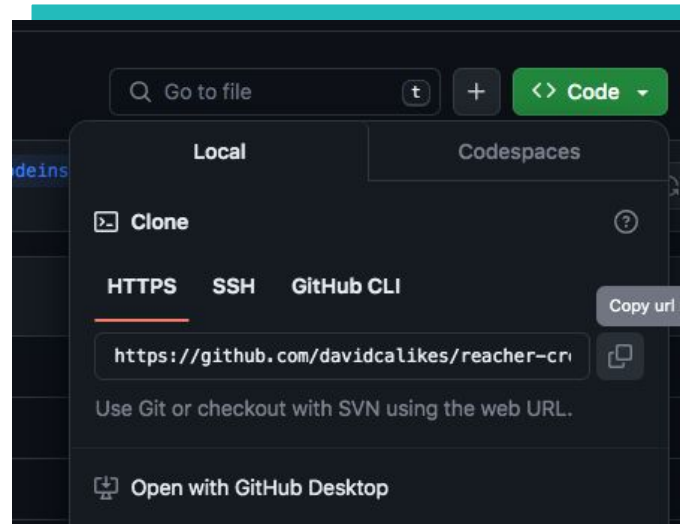
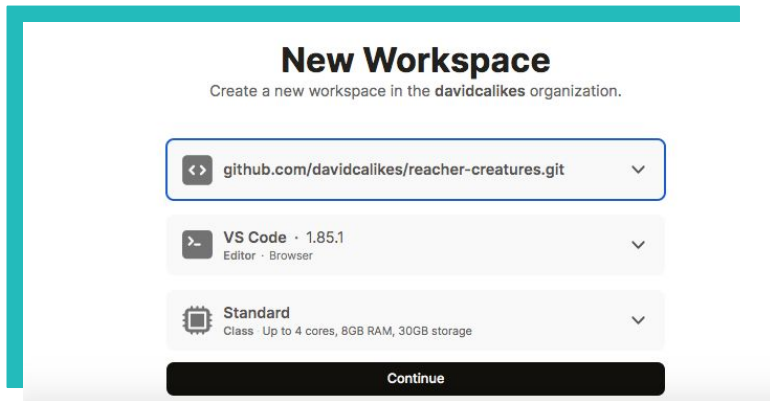
Make sure you are in constant communication with your teammates throughout the project.



Creating A GitPod Workspace From Your Fork & Committing Changes

Create a GitPod Workspace from your **Forked** repository.

Firstly, Copy the URL of your Forked repository via the tab in image 1



Then simply create a new workspace by pasting the URL into the “Create New Workspace” input on your GitPod Dashboard. We can commit changes to this fork by using the following terminal commands:

```
git add .  git commit -m “message”  git push
```

Part 3

- ✓ Creating a Pull Request
- ✓ Merging Code
- ✓ Resolving Merge Conflicts

Stand down call
Mon-Thur

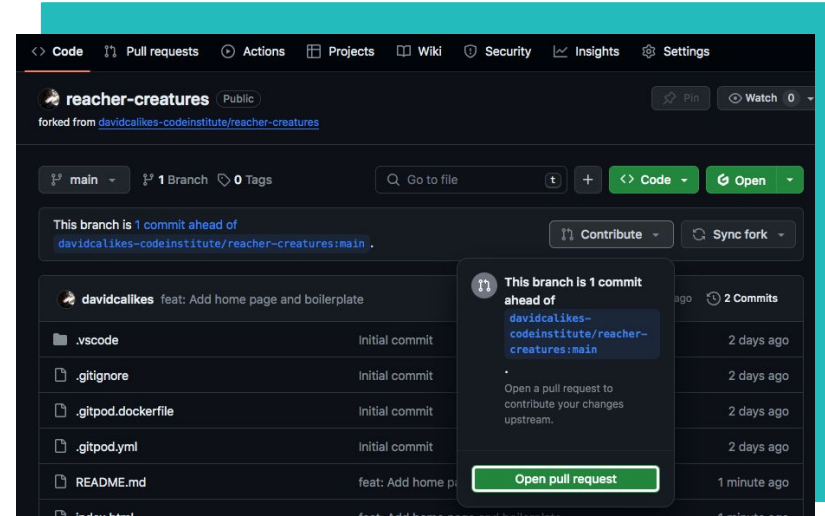


Creating A Pull Request

The Hackathon Team Members/Collaborators

will request that the owner of the upstream project repository, pull their code into the main branch of the upstream repository we do this by creating what's called a 'pull request'.

When using the forking collaboration method, we simply have to navigate back to our **forked** repository, and click on the “**contribute**” tab and then “**Open pull request**”

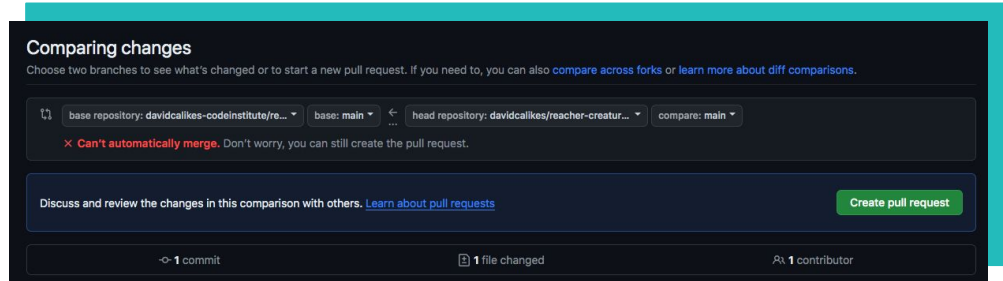
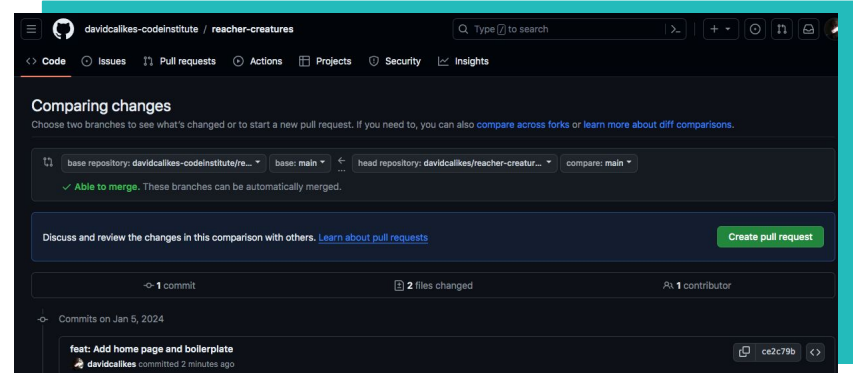


Creating A Pull Request

This button will bring us to the upstream repo and let us know if there are any potential issues with merging our code.

Click “Create Pull Request”

The GUI will inform you if there are potential issues with merging this code.



Avoiding Merge Conflicts

Even if the User Interface in GitHub indicates that a pull request may cause an issue when merged upstream, the request can still be submitted and reviewed by the Team Lead anyway.

The easiest way to avoid merge conflicts is to:

Constantly communicate as a team.

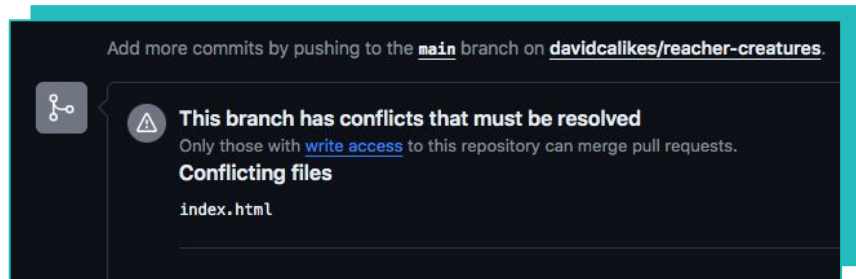
Remind your team mates to Pull from the upstream before working on each new addition/issue. Let them know when you've completed a task.

Plan & Prepare

Planning is key. Make a plan and stick to it. If you deviate from the plan. Inform your team.

Separation of Concerns

Separate each development task so that developers are unlikely to be working on the same document/ area of each document at the same time.

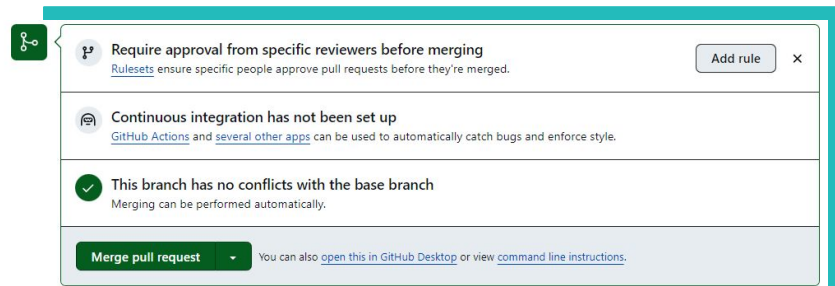
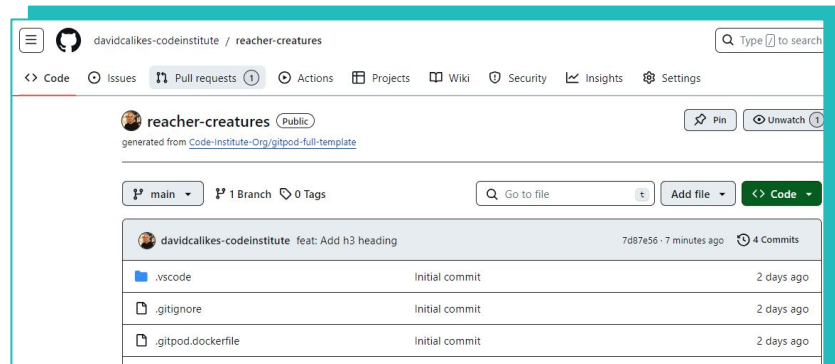


Merging Code

When a request has been submitted the Team Lead will handle the request from the upstream project repository.

Click on the “**Pull requests**” tab to see any open requests.

If the code can be merged without any conflicts the Team lead can “**Merge Pull Request**” into the upstream repository.



Resolving Merge Conflicts

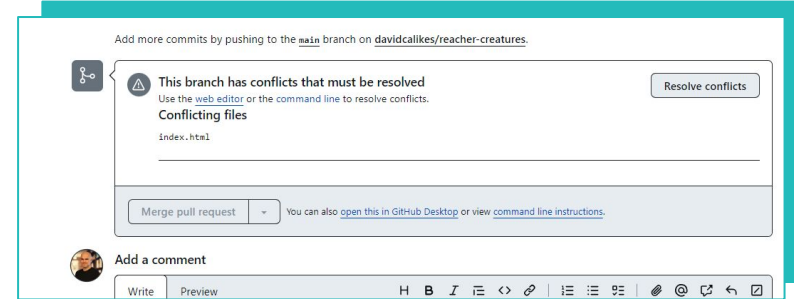
If a pull request results in a merge conflict, we can resolve this via the web editor in the GitHub GUI.

Click on “**Web Editor**” to open the GitHub GUI Web Editor.

Edit/Remove the Conflicting Code

Click “**Commit Merge**” at the top right of the Web Editor Page.

You can now successfully merge the code in the pull requests page.



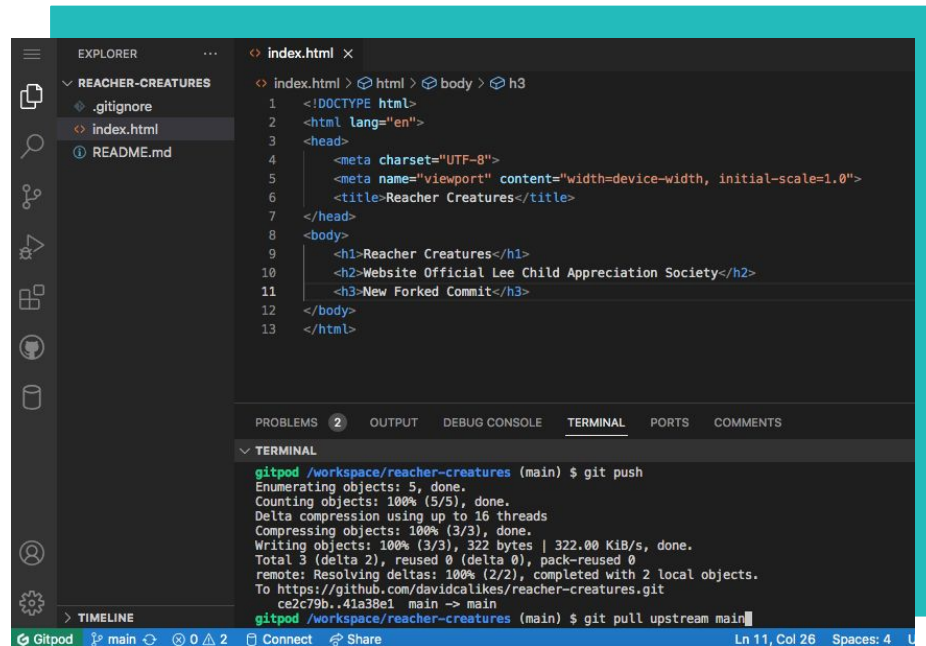
Updating Our Forks

After each successful merge, we can reuse our original forked repository to make more changes/updates but before we do we must make sure we pull all of the latest code into our forked repository. Remember, the other developers are also committing code to the project, so we need to make sure we have access to that code too.

To make sure our **forked** repository has the latest version of the upstream codebase we use the terminal command:

git pull upstream main

***before** we begin every meaningful coding contribution*



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'REACHER-CREATURES' with files '.gitignore', 'index.html', and 'README.md'. The main editor area displays the 'index.html' file, which contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Reacher Creatures</title>
7 </head>
8 <body>
9   <h1>Reacher Creatures</h1>
10  <h2>Website Official Lee Child Appreciation Society</h2>
11  <h3>New Forked Commit</h3>
12 </body>
13 </html>
```

Below the editor, the TERMINAL panel is open, showing the output of the following commands:

```
gitpod /workspace/reacher-creatures (main) $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/davidcalikes/reacher-creatures.git
ce2c79b..41a38e1 main -> main
gitpod /workspace/reacher-creatures (main) $ git pull upstream main
```

The status bar at the bottom indicates the current file is 'Ln 11, Col 26' with 'Spaces: 4'.