Hands-on-Training:
Use CumulusCI for Open Source App Development

Philadelphia Sprint 2019

Jason Lantz, Sr. Director - Release Engineering - Salesforce.org
jlantz@salesforce.com, @jasontlantz

Presenters

Jason Lantz

Senior Director,
Release Engineering

David Reed

Senior Member of Technical Staff,
Release Engineering

Did you complete the CCI Setup?

To follow along with today's training, you must have successfully completed the Get Ready to Use CumulusCI setup instructions.

As a test, you can confirm you have your terminal connected to GitHub properly:

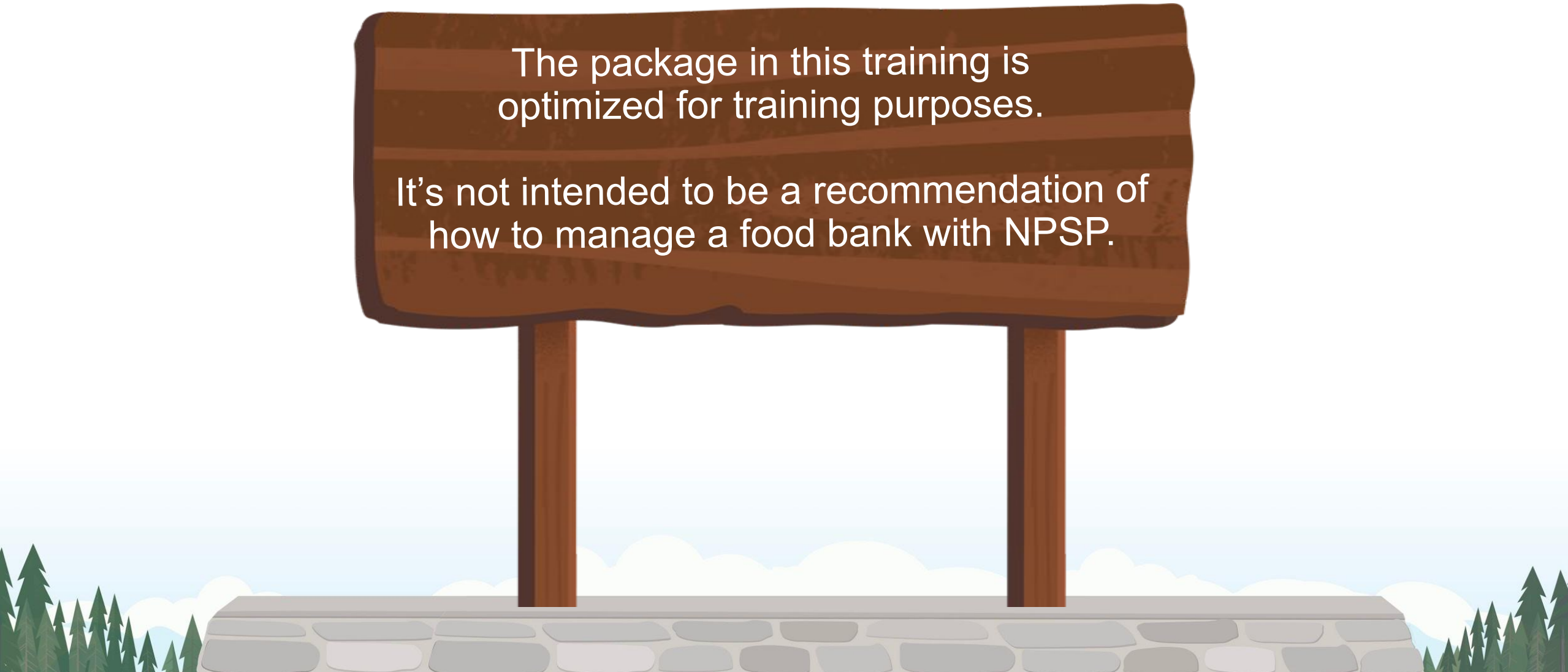
1. Open Terminal
2. Run the following command:

```
cci service list
```

3. You should see a green checkmark for the github service showing it's configured

Disclaimer

Don’t do this in production!



Goals for the Food Bank Project Development

A package of Salesforce customizations for food banks to track food deliveries!

In the CumulusCI setup instructions, we read about Tasha and her Food Bank App team challenges:

- Shared place for the project development
- Automated way to create an org with NPSP
- Share work with the team to get feedback
- Peer-review of customizations
- Way for SFDO Community to see project progress and roadmap

Training Agenda

Lesson 1

Set Up the Project

Lesson 2

Make and capture development changes

Lesson 3

Share customizations with the Team

Lesson 4

Review a Pull Request from a Collaborator

Stuck or Confused?

How to get help during the training

1. **Raise your hand!** Let us know if you're having issues and one of our helpers will be right over.

2. **Refer to the digital copy of this deck.** If you get behind, use it to catch up.

3. **Partner with your neighbor.** If you weren't able to complete the CumulusCI setup instructions on your computer, watch your neighbor's work rather than following along on your own computer.

Helpers please raise your hand and say hi!

Goal: Shared workplace

Section 1:

Set Up the Project

Section 1: Set Up the Project

What

- Create a git repository (in GitHub) to manage the project's development.
- Configure the project to use CumulusCI.
- Commit the configuration to version control (GitHub).

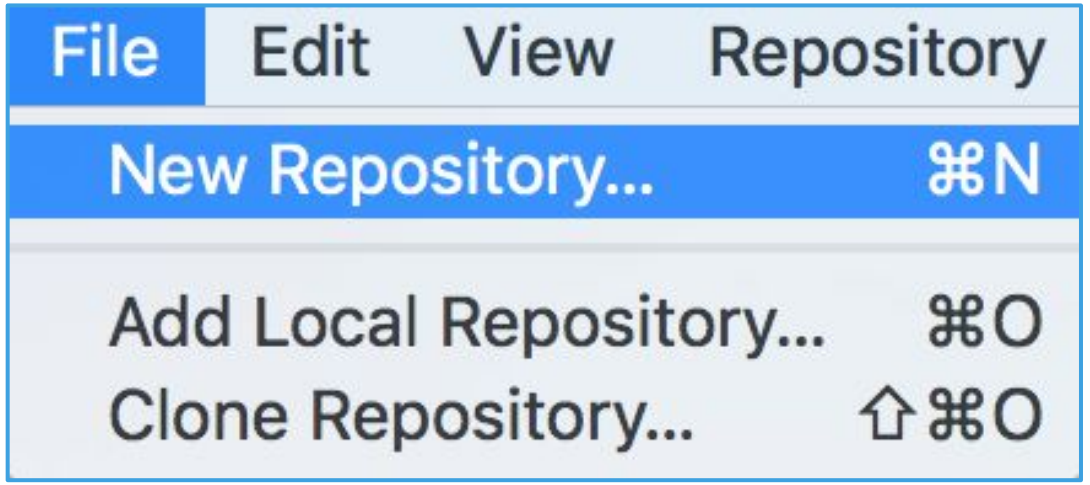
Why

- Version control (git) helps keep track of changes to the project.
- CumulusCI provides customizable automation for everyone on the team to:
  - Create development or test environments in scratch orgs
  - Automatically install the latest NPSP
- Creates a central workspace for your team to start building.

Creating the Git Repository

Use GitHub Desktop

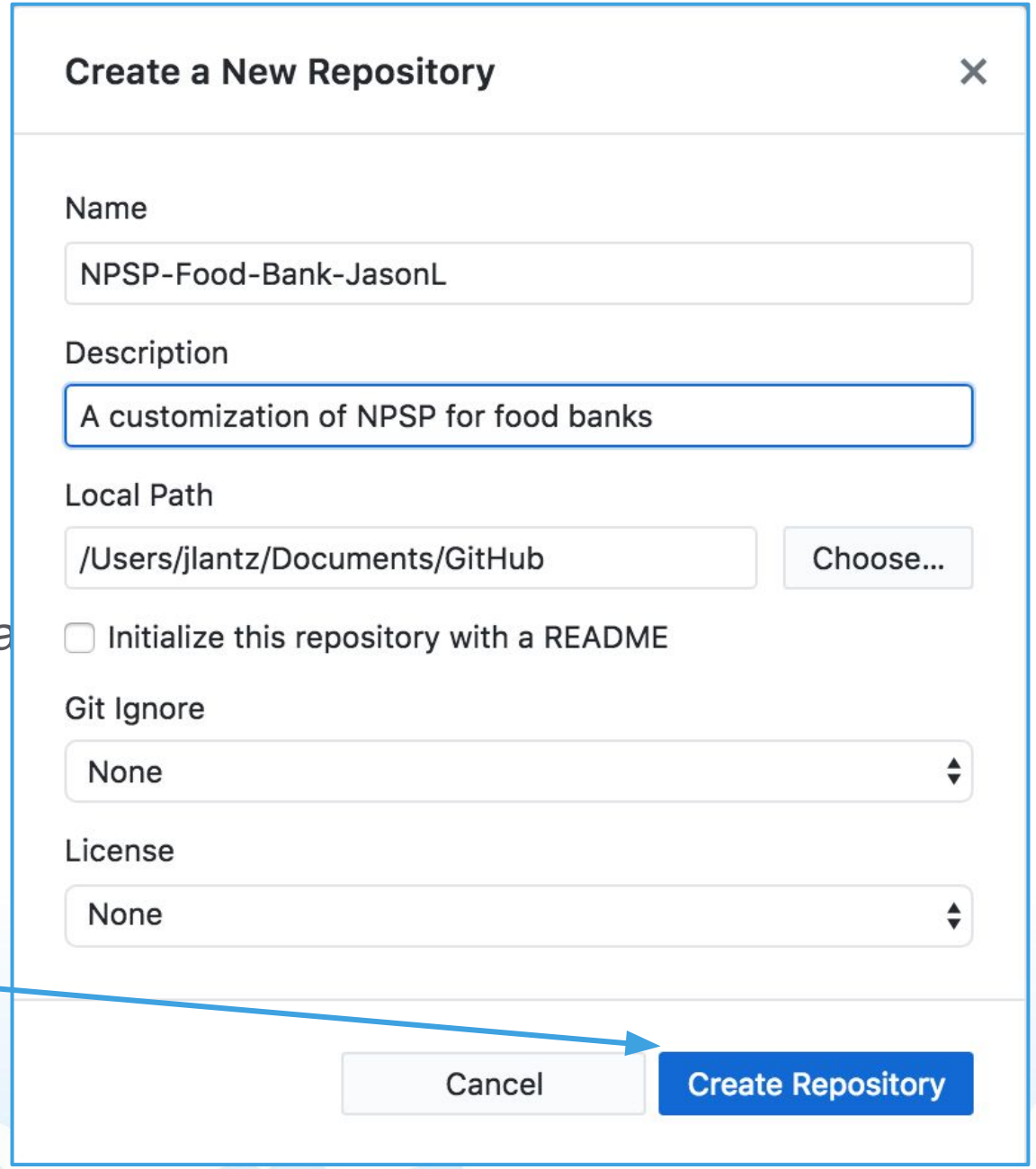
1. Open GitHub Desktop.
2. Select **File -> New Repository** from the menu.



Creating the Git Repository

Use GitHub Desktop

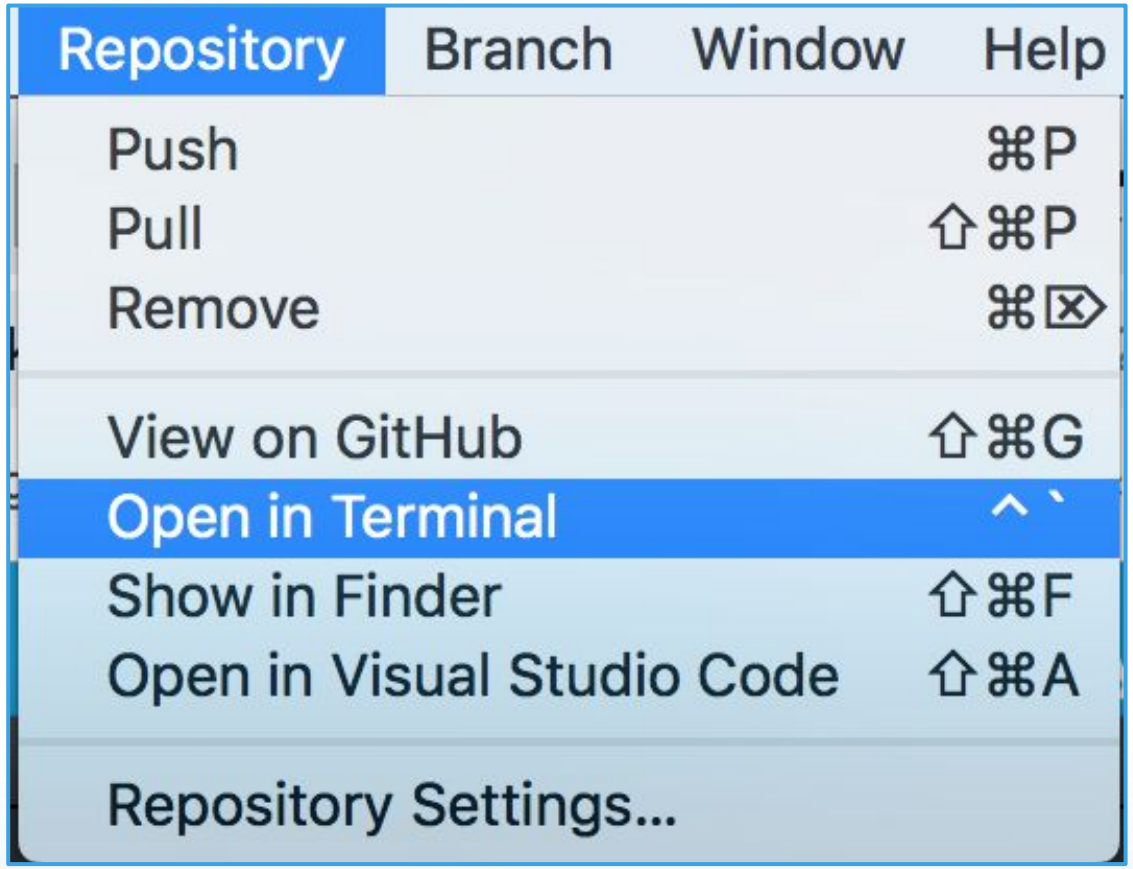
3. Enter the information for the repository below:
    - Name: **NPSP-Food-Bank-YOURFIRSTNAME**
    - Description: **Customization of NPSP for food banks**
    - Local Path: **Choose GitHub** folder, or crea a new one if you see a warning
    - Git Ignore: **None**
    - Licence: **None**
4. Click **Create Repository.**



Open in Terminal

Use GitHub Desktop

Open a terminal inside the GitHub repository:
Repository > Open in Terminal



Why Extend NPSP?

An **Extension Package** is a package that contains references to metadata in another package.

- When building a solution for food banks, there is likely to be a fair amount of overlap with the functionality of NPSP such as householding and fundraising.
- In order to work on an extension of NPSP, you need to have NPSP installed into your org so your metadata can be deployed
- NOTE: For this training, we won't be making any metadata references to NPSP but we want to show how to configure the project to extend NPSP

Configure for CumulusCI

In your terminal application

In the Terminal window, type the command:

```
cci project init
```

1. Use default Project Name (**hit Enter**)
2. Customize the Package Name to be **NPSP for Food Banks (YourName)**
3. Answer **n** for managed package project
4. Use default API Version (**hit Enter**)
5. Enter **mdapi** for the source format
6. Answer **y** to extend another CumulusCI project. Enter **2** to extend NPSP
7. **Hit Enter** to accept defaults for remaining prompts

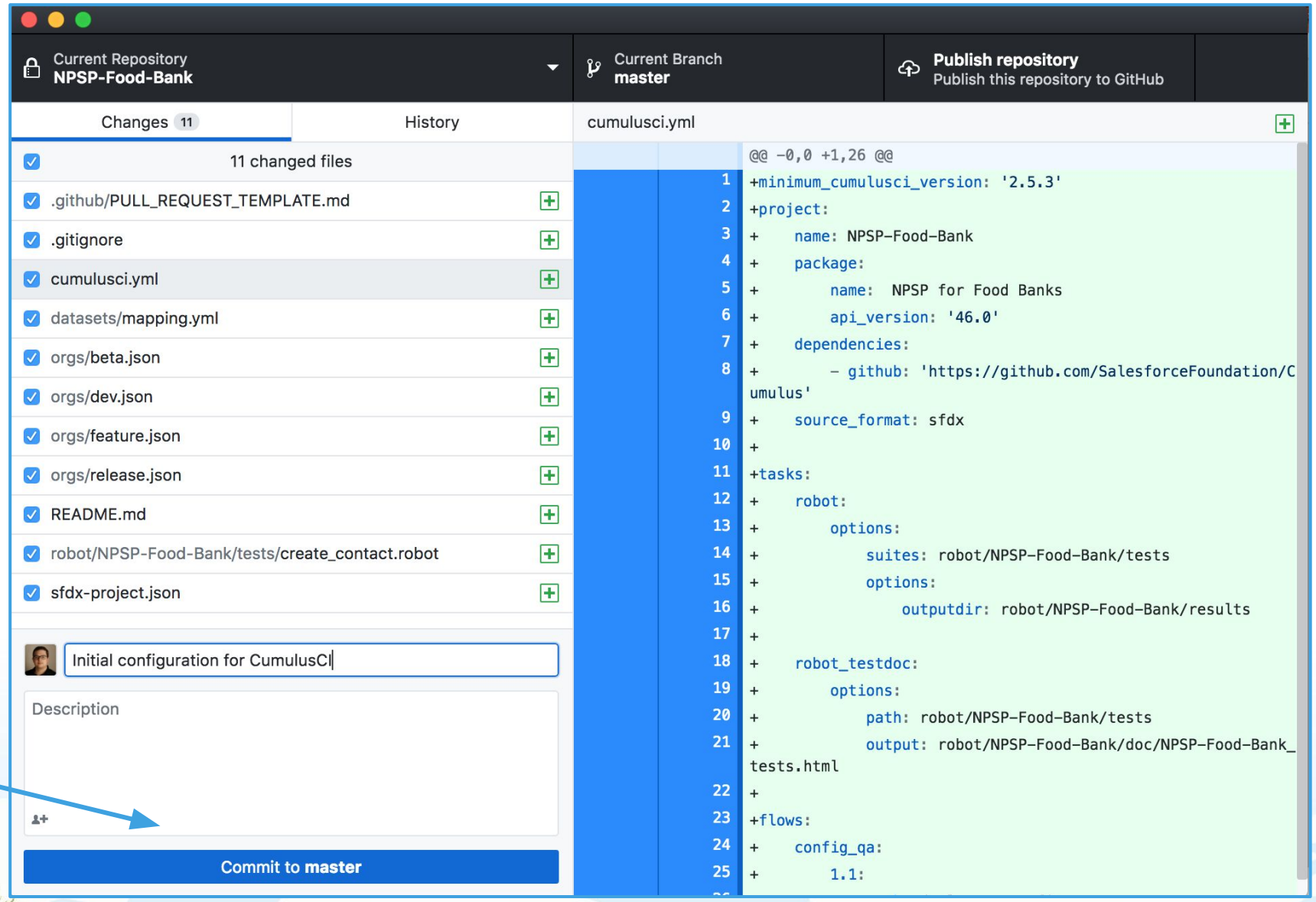
Pro Tip: If you enter something wrong, press Ctrl and C at the same time to exit and then re-run the cci project init command

Commit CumulusCI Configuration

Use GitHub Desktop

1. Enter a brief **description** of your changes.

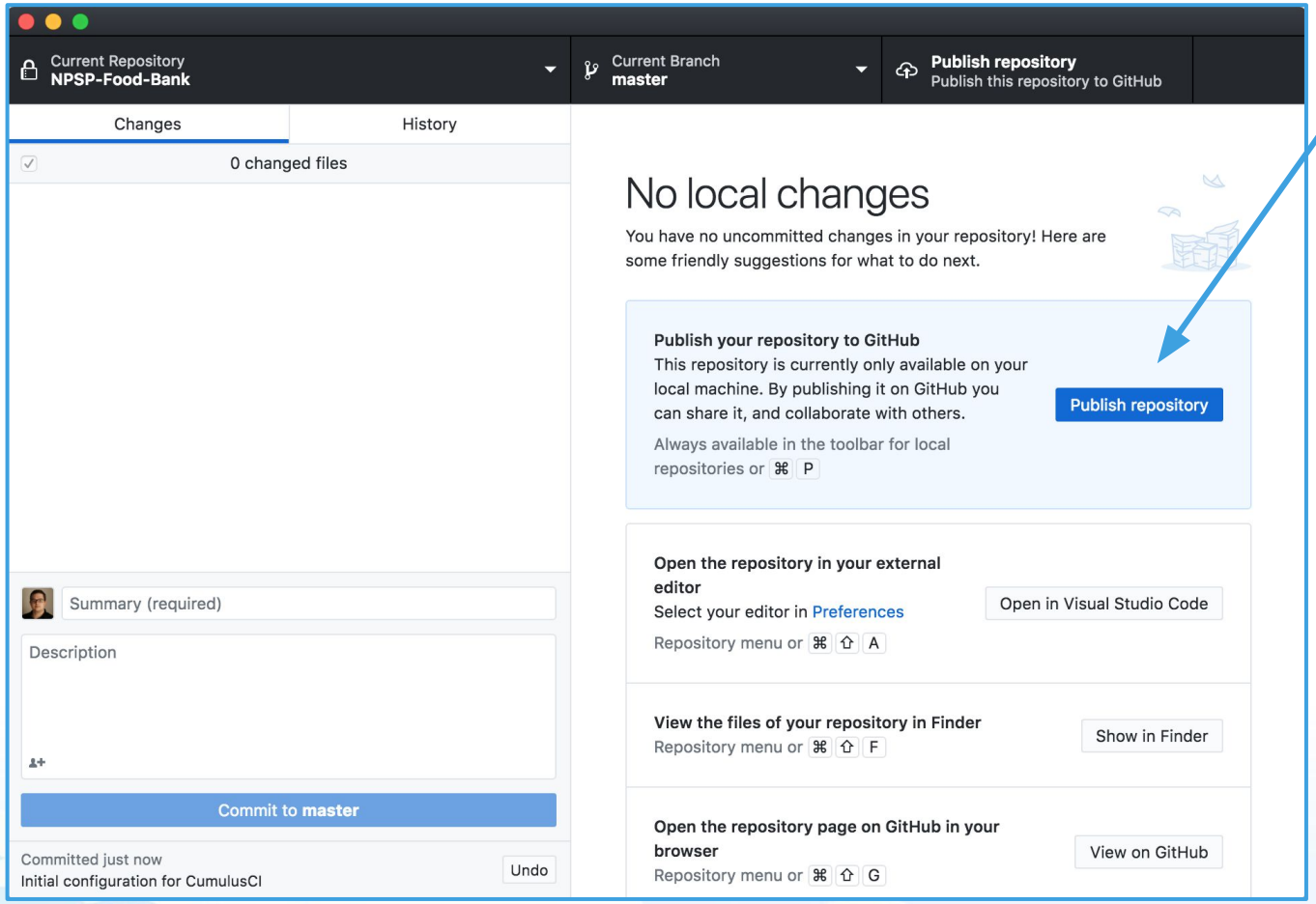
2. **Click Commit** to master.

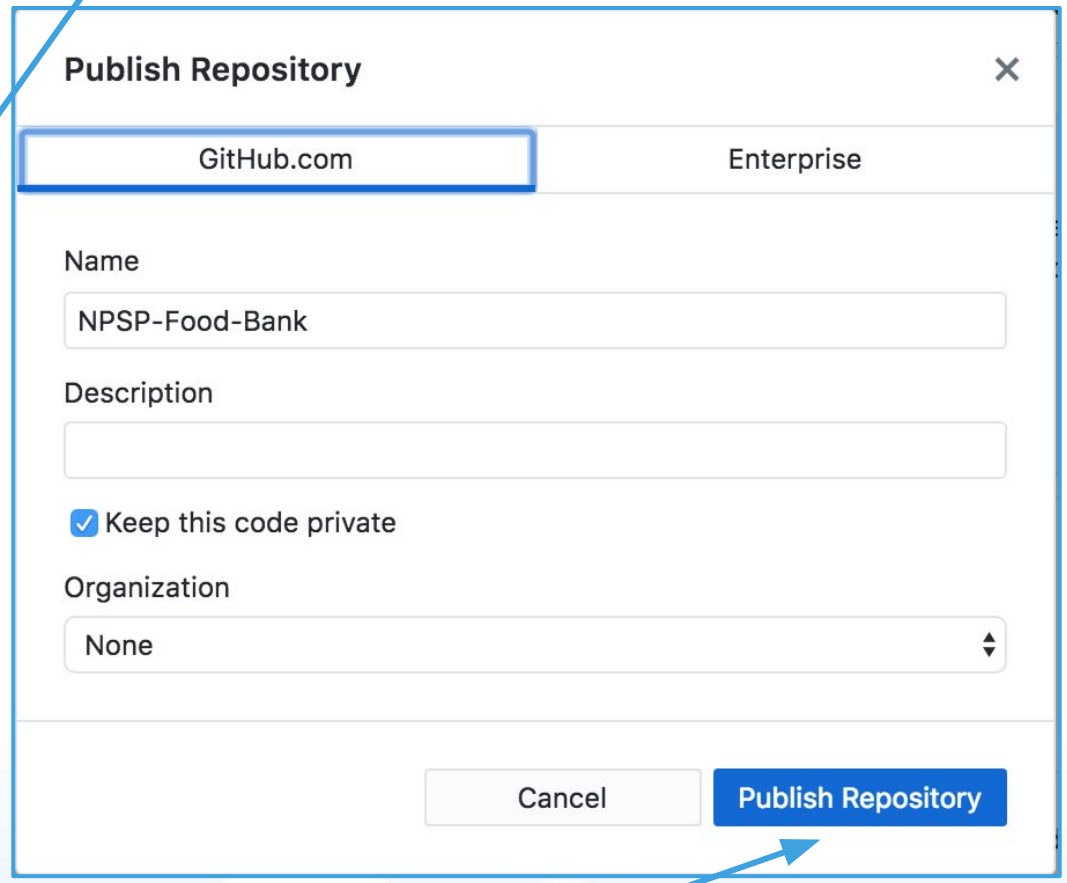


Push Changes to GitHub

Use GitHub Desktop

1. Click **Publish Repository.**

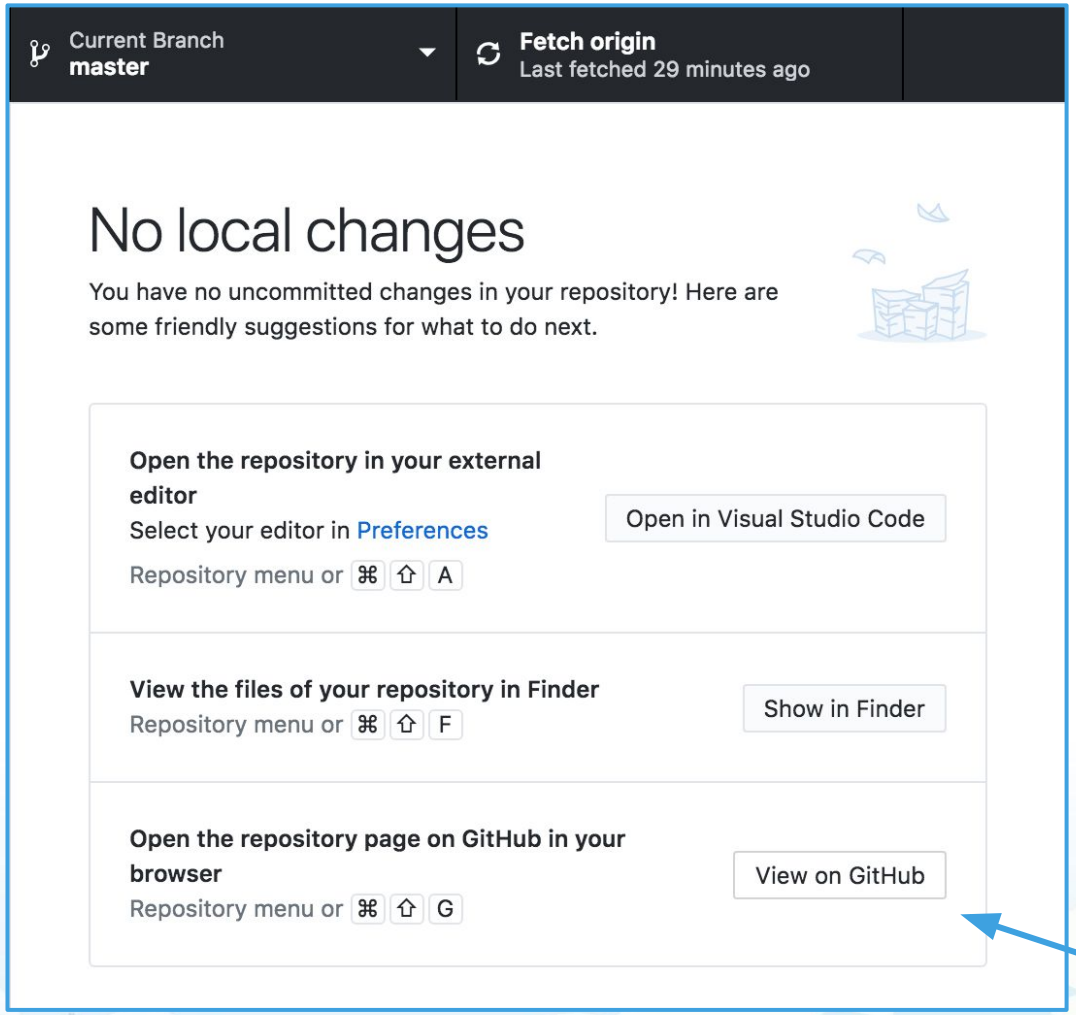


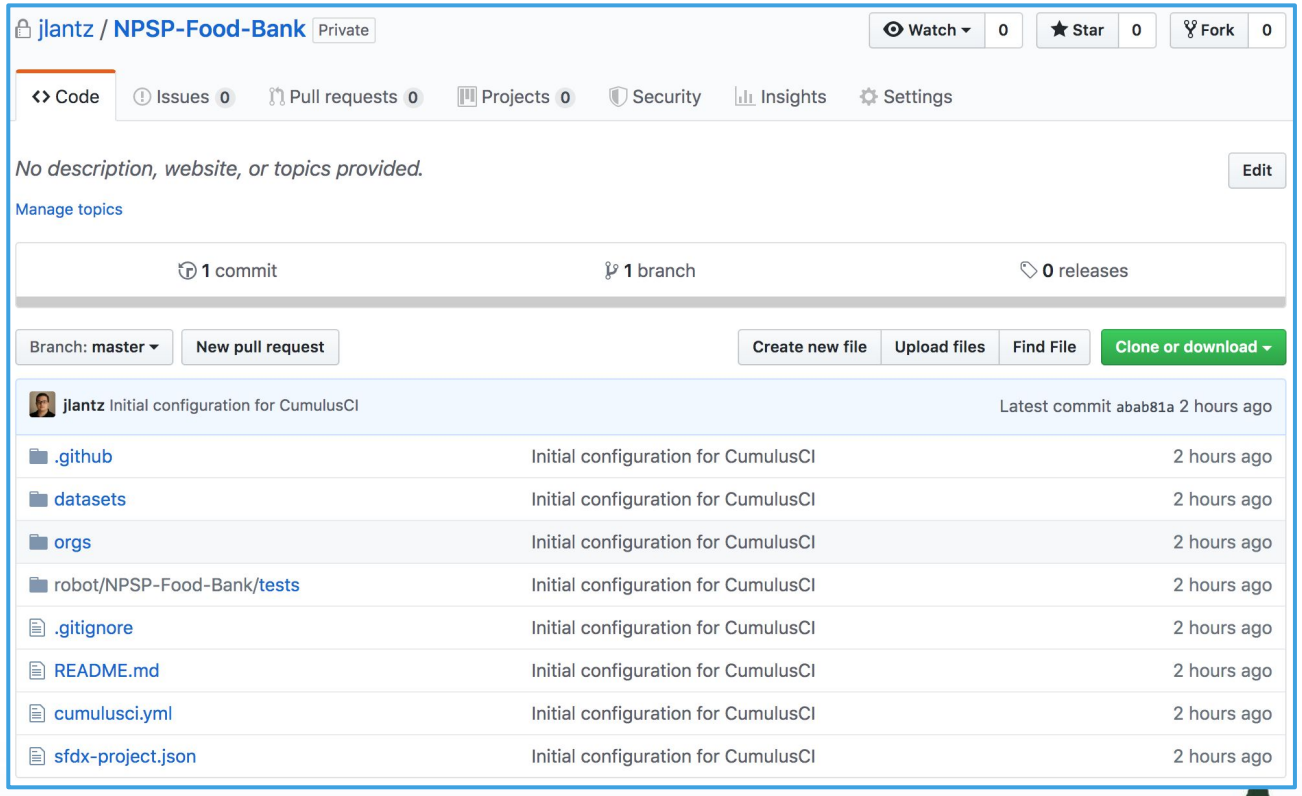


2. Click **Publish Repository.**

Project Repository on GitHub

Use GitHub Desktop to open GitHub in browser





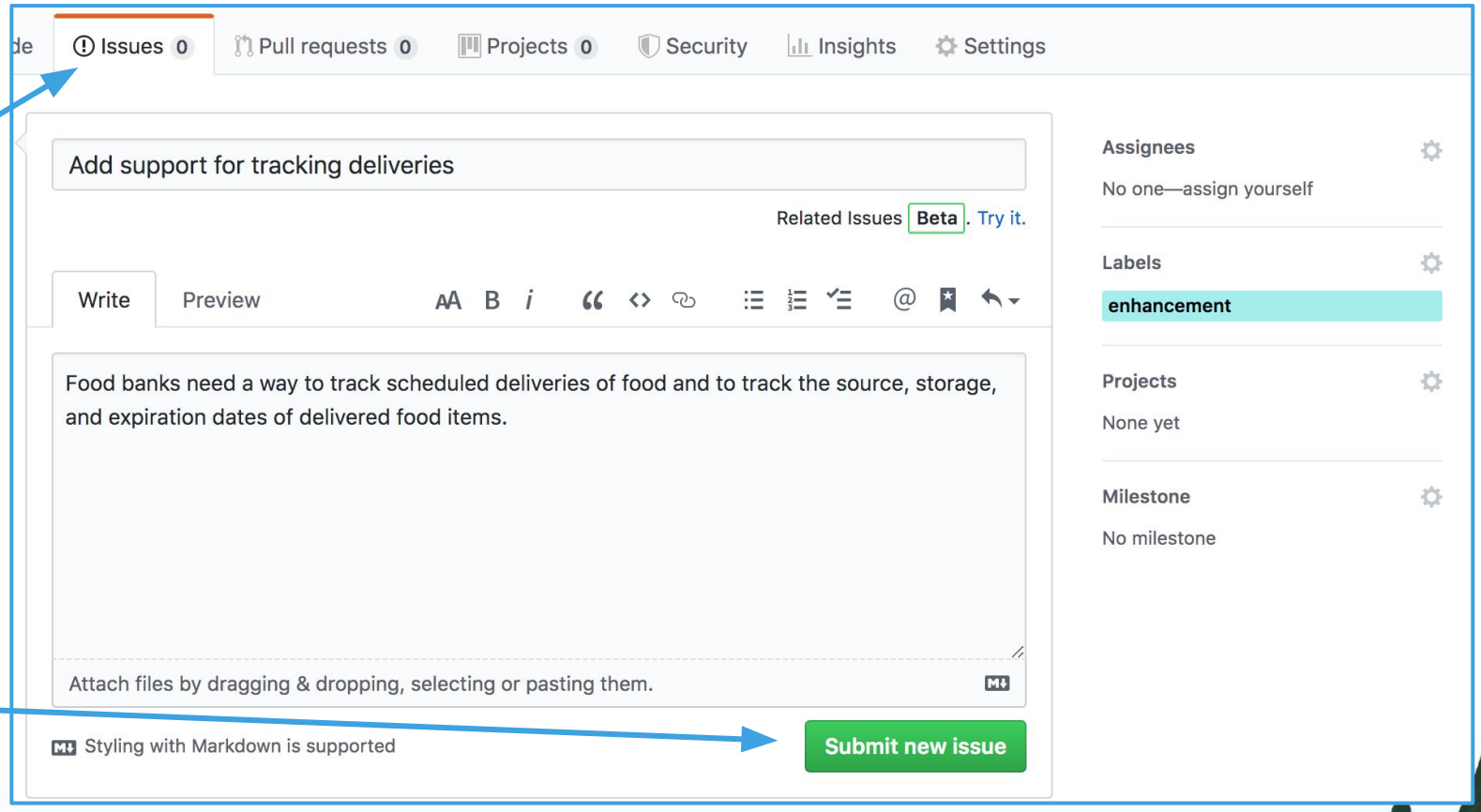
Click **View on GitHub**.

Create a GitHub Issue

In GitHub

Log a change you'd like to see:

1. Click the **Issues** tab to create a new issue.
2. Select **enhancement** from the **Labels** on the right to categorize the issue.
3. Add a **Subject** and enter details in the **Description**.
4. Click **Submit new issue.**



Create a Development Environment

In your terminal application

To create your development scratch org, run the following command in your Terminal:

```
cci flow run dev_org --org dev
```

(If you need to open your Terminal again, go to **Repository -> Open in Terminal** from the menu in GitHub Desktop.)

What is CumulusCI Doing? (This may take a few mins)

What happens when you type cci task run dev_org --org dev

- dev_org is a flow—or series of tasks—that configures a scratch org that you'll to use when developing your app. Tasks include:
  - Installs dependencies—in this case the latest version of NPSP and all its dependencies (Contacts & Organizations, Households, Recurring Donations, etc).
  - Completes the NPSP post-install steps automatically!
- --org dev tells CumulusCI in which scratch org to run the flow. In this case ours is called dev.

Section 1: Recap

Great job! In this session you:

- Created a Git Repository for your Food Bank project.
- Configured the repo to use CumulusCI.
- Created a scratch org with NPSP installed.
- Submitted a GitHub Issue to suggest new development.
- Started the process to create a new Scratch Org.

Get ready to use your new Scratch Org, and add new Objects and Fields in Session 2 after the break!

Goal: Automatically create org with NPSP!

Section 2:

Make and Capture Development Changes

Section 2: Make and Capture Development Changes

What

- In a scratch org, create objects and fields for tracking Deliveries.
- Capture your changes in a centralized shared environment.

Why

- All changes to the project will go through this same cycle. Like riding a bicycle: the same process is used on any CumulusCI project. Learn once and reuse everywhere.
- Implement your design, and share the changes you made with your team.

Open the Scratch Org in a Browser

Use terminal application

Open the scratch org (named dev) in a new browser window, run the following command:

```
cci org browser dev
```

Create the Deliveries Schema

Create customizations in your Salesforce scratch org

New Object: **Delivery**

- **Name Field** (Auto Number)
- **Scheduled Date** (DateTime)
- **Status** (Picklist)
  - Requested
  - Scheduled
  - Completed
  - Cancelled
- **Supplier** (Lookup to Account)

New Object: **Delivery Item**

- **Delivery** (Lookup)
- **Food Storage Type** (Picklist)
  - Non-refrigerated
  - Frozen
  - Refrigerated
- **Food Expiration Date** (Date)

Tip! Great Starting Point for Community Projects

Schema-only packages

One of the easiest starting points for sharing a community project is to create and publish a managed package containing only custom objects and fields (aka schema).

- It's easy to retrieve schema from an instance.
- Schema is easy to package in a managed package.
- Objects and fields can be extended by users with different implementations.
- Schema in a managed package is namespaced (ex: npspfoodb__) which ensures that all organizations have the same schema. This common namespaced schema allow for extension packages such as implementations for a specific state's regulatory requirements. It also provides a common schema for data scientists to help many food banks make sense of their data.

View Metadata Changes

Use CumulusCI's list_changes task in terminal application

To create a list of all the changes (metadata) detected in your dev scratch org, type the following command in your Terminal:

```
cci task run list_changes --org dev
```

It's best to leave profiles out of a managed package. Modify the command to exclude profiles:

```
cci task run list_changes --org dev -o exclude "Profile:"
```

Pro Tip: You can use the up arrow on your keyboard to recall the last command you used! Next, edit it and click enter to run the command!

Retrieving Changes

Use CumulusCI's retrieve_changes task in terminal application

Next, retrieve all changes except those profile changes by running the command:

```
cci task run retrieve_changes --org dev -o exclude "Profile:"
```

Pro Tip: You can use the up arrow on your keyboard to recall the last command you used! Next, edit it and click enter to run the command!

View a Diff of Metadata Changes

Your changes should now appear in GitHub Desktop!

Current Repository
NPSP-Food-Bank-JLantz

Current Branch
master

Publish repository
Publish this repository to GitHub

An updated version of GitHub Desktop is available and will be installed at the next launch. See what's new or restart GitHub Desktop.

Changes 5 | History

5 changed files

- src/layouts/Delivery__c-Delivery Layout.layout
- src/layouts/Delivery_Item__c-Delivery Item Layout.layout
- src/objects/Delivery__c.object
- src/objects/Delivery_Item__c.object
- src/package.xml

Summary (required)

Description

Commit to **master**

Committed an hour ago
Configure for CumulusCI

Undo

src/layouts/Delivery__c-Delivery Layout.layout

```
@@ -0,0 +1,90 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
+    <layoutSections>
+        <customLabel>false</customLabel>
+        <detailHeading>false</detailHeading>
+        <editHeading>true</editHeading>
+        <label>Information</label>
+        <layoutColumns>
+            <layoutItems>
+                <behavior>Readonly</behavior>
+                <field>Name</field>
+            </layoutItems>
+            <layoutItems>
+                <behavior>Edit</behavior>
+                <field>Scheduled_Date__c</field>
+            </layoutItems>
+            <layoutItems>
+                <behavior>Edit</behavior>
+                <field>Status__c</field>
+            </layoutItems>
+            <layoutItems>
+                <behavior>Edit</behavior>
+                <field>Supplier__c</field>
+            </layoutItems>
+        </layoutColumns>
```

Section 2: Recap

Great job! In this session you:

- Used a Scratch Org for the first time!
- Added customizations via two new Objects and their fields.
- Used Terminal to show you all the changes you made.

Get ready to create to share your changes with your Team in Session 3 after the break!

Section 3:

Share Customizations with the Team

Section 3: Share Customizations with the Team

What

- Create a feature branch and add your changes to it.
- Push the changes to GitHub & create a Pull Request.
- Add team members to your GitHub repository.

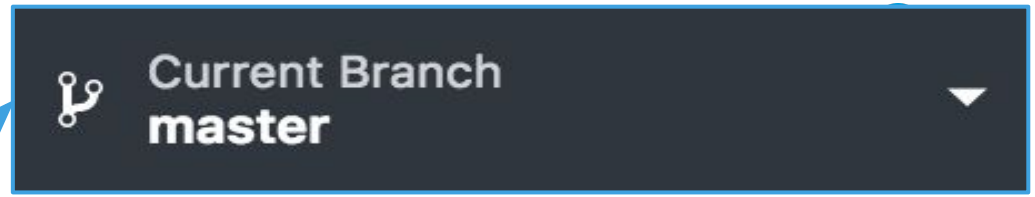
Why

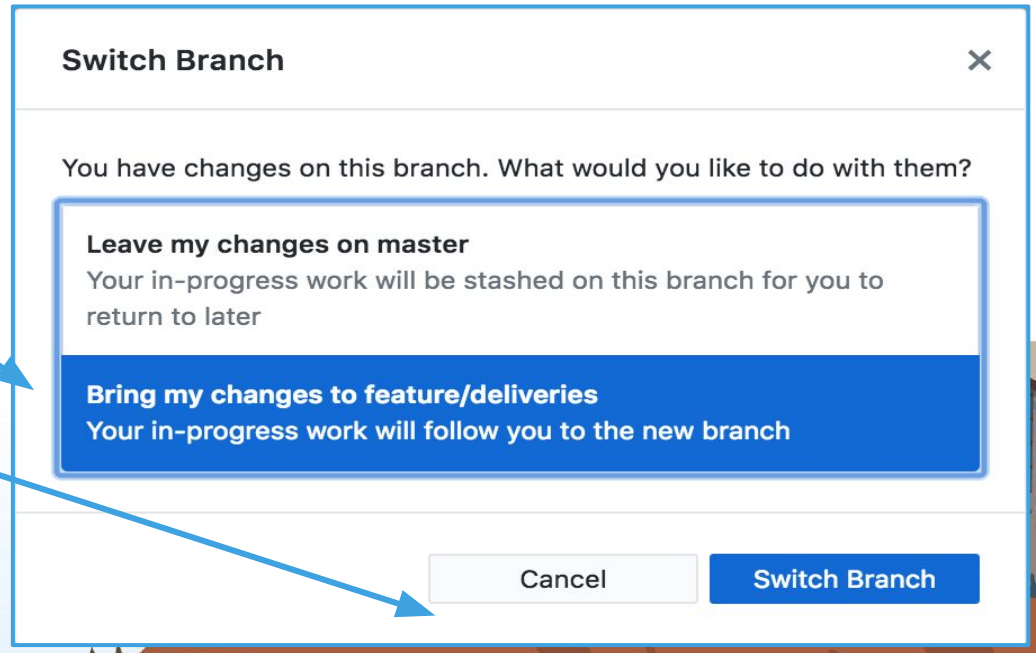
- Branches are used to isolate your development work, i.e. independent of other team members changes.
- Pull Requests provide a way for teams to track, review, and combine their work.
- A second set of eyes reviewing your work is a vital part of the quality assurance process.

Creating a Feature Branch

Use GitHub Desktop

1. Create a new branch by clicking on **Current Branch: master.**
2. Enter **feature/deliveries** for the branch name and click **New Branch.**
3. Select **Bring my Changes to feature/deliveries.**
4. Click **Switch Branch.**

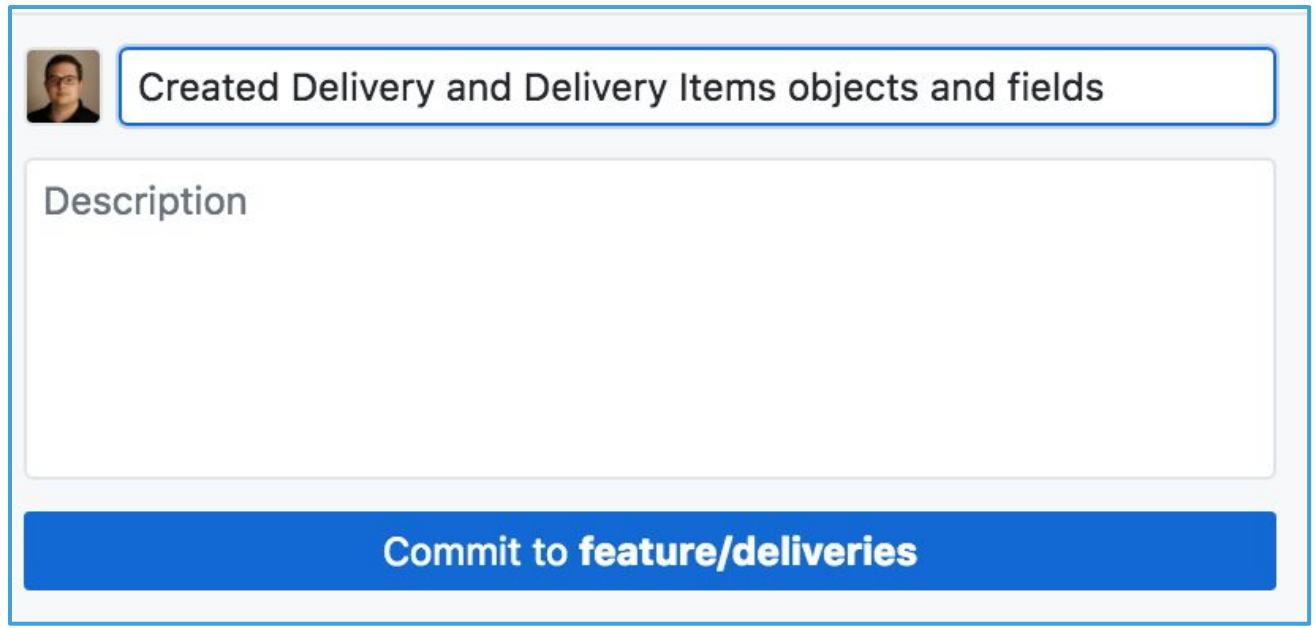




Commit Changes to Feature Branch

Use GitHub Desktop

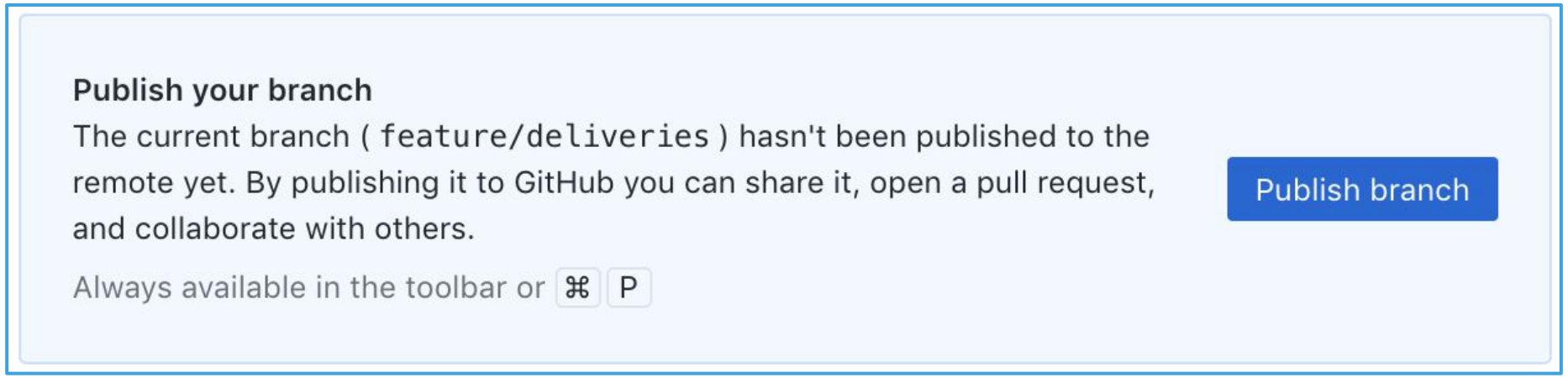
1. Enter a **Commit Message** and **Description** indicating what you've changed.
2. Click **Commit to feature/deliveries**.



Push Feature Branch to GitHub

Use GitHub Desktop

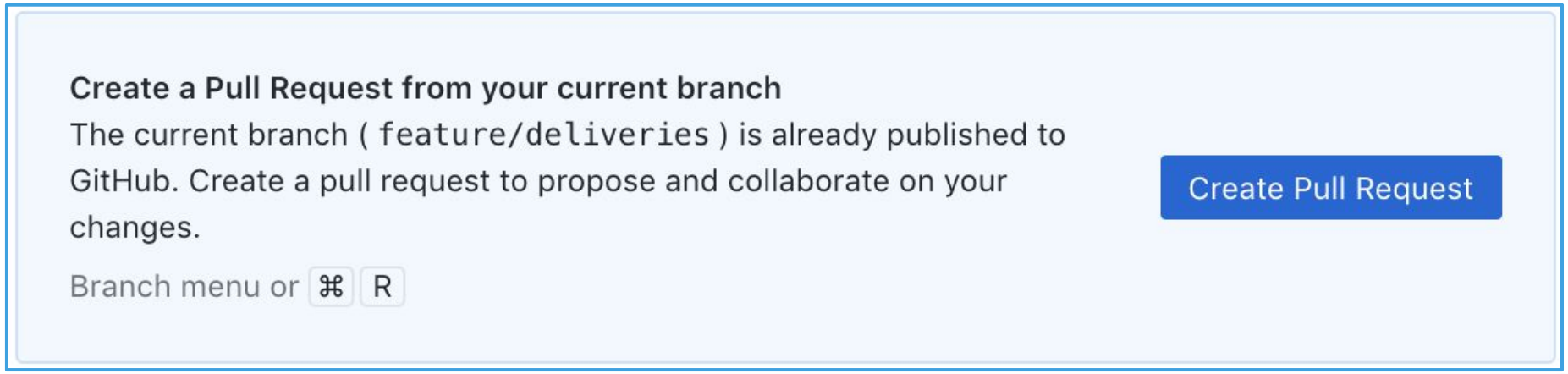
Next, click the **Publish branch** button.

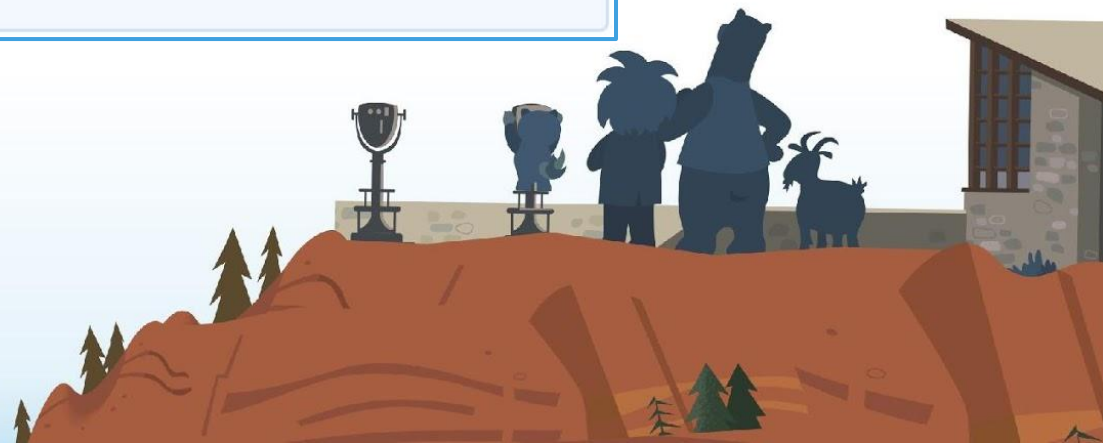


Create a Pull Request

Use GitHub Desktop

- Pull Requests track a request to merge one branch into another branch. In this case, we want to merge **feature/dependencies** into **master**.
- Click the **Create Pull Request** button.





Pull Requests & CumulusCI Release Notes

In GitHub

CumulusCI handles generating cumulative release notes for your project's releases using content entered in the body of pull requests. There are 3 main sections:

- **Critical Changes**: Add bullet points for any changes that might break existing functionality and describe any action users might need to take in their orgs.

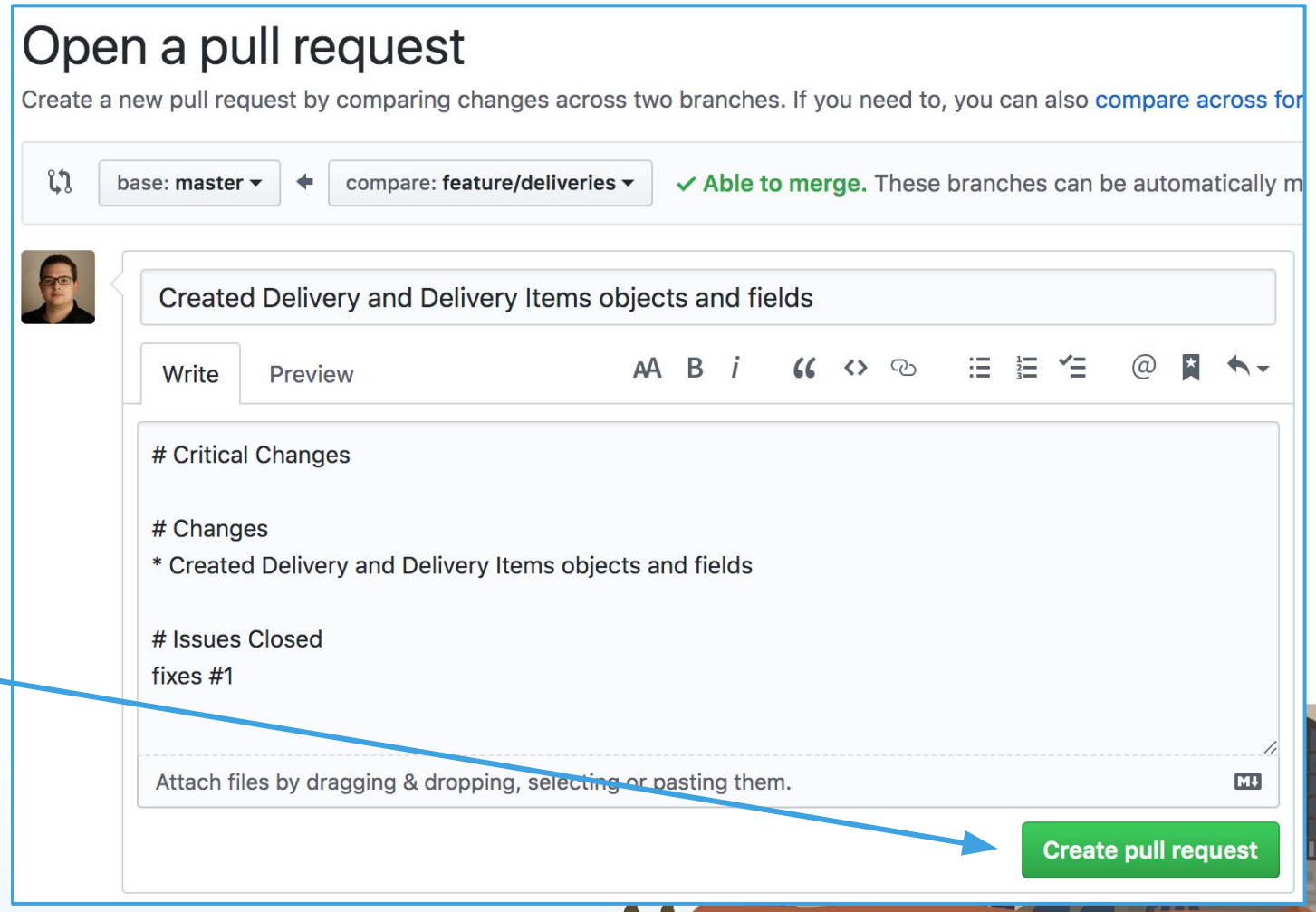
- **Changes**: Add bullet points for notable changes to the package including any new feature that requires user action to enable.

- **Issues Closed**: Link to any GitHub issues that are resolved by the pull request using the format **Fixes #N** where N is the issue number.

Create a Pull Request

In GitHub

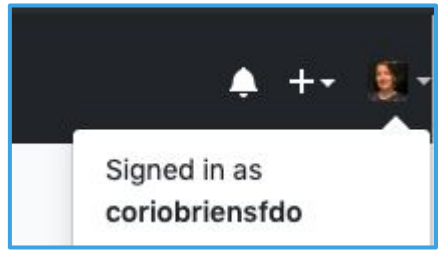
1. Edit the release notes to include the following details:
   a. In **Changes**, enter Created Delivery and Delivery Items objects and fields
   b. In **Issues Closed**, enter Fixes #N where N is your issue number
2. Click **Create pull request**

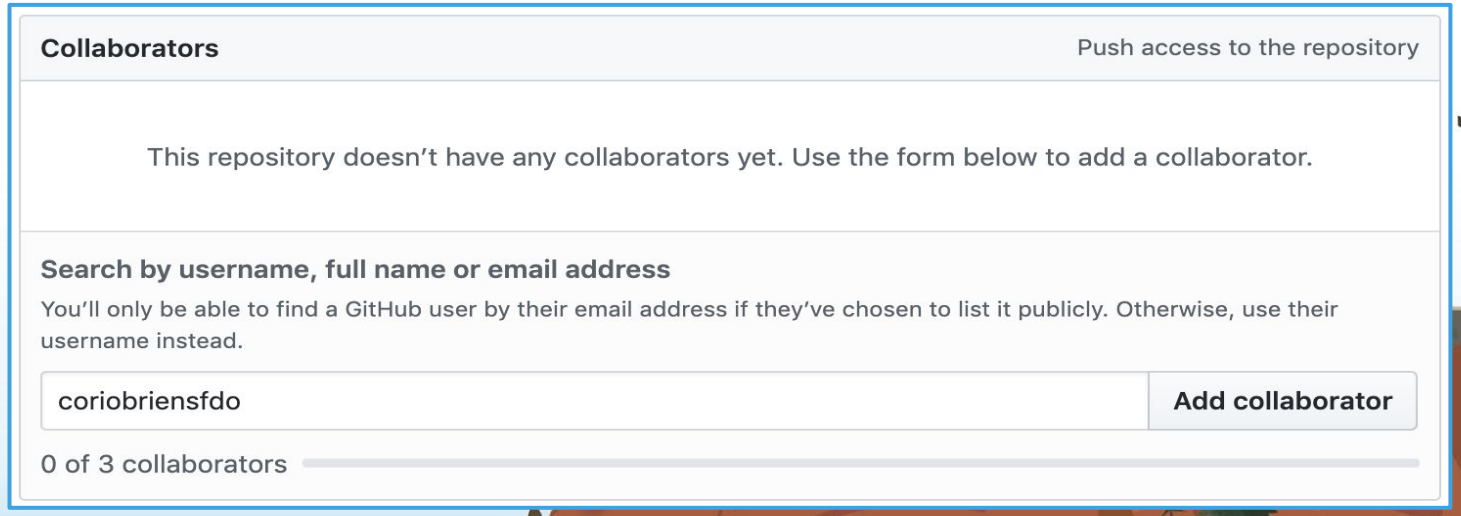


Add a Neighbor as a Collaborator

And get yourself set up as a collaborator for them!

1. Find a partner and get their GitHub username. It should similar to “coriobriensfdo”.
2. In GitHub, click on the **Settings** tab.
3. Click **Collaborators** on the left.
4. Enter your **GitHub password** to confirm your access.
5. Type in your partner’s **GitHub username** and click the **Add collaborator** button.
6. You and your partner should get an email with an invitation. Click the **View Invitation** button, then **Accept Invitation**.





Section 3: Recap

Great job! In this session you:

- Created your first feature branch and pull request.
- Added a collaborator to your GitHub repo.

Get ready to create to review your partners work and suggest changes in Session 4 after the break!

Goal: Peer review of work!

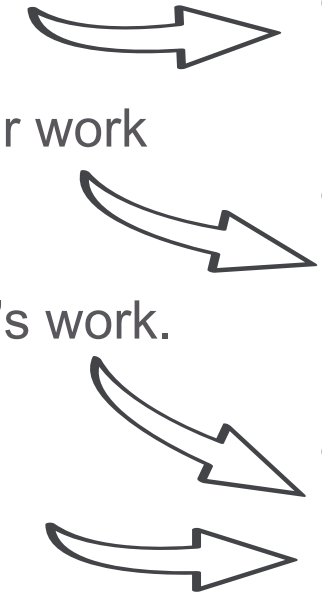
Section 4:

Review a Pull Request from a Collaborator

Section 4: Review a Pull Request from a Collaborator

What

- Clone your partner's repository.
- Create a new scratch org with their work included.
- Propose a change to your partner's work.
- Review and accept the proposed changes to your work.



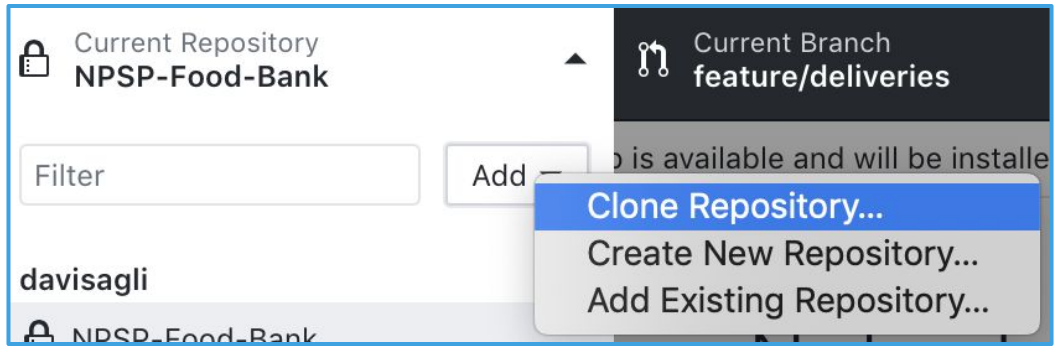
Why

- This'll allow you to review the work your partner shared in their GitHub repo.
- Allows you to create a clean org with their work and completes any post-install steps!
- It's best practice to suggest changes to others' work which allows them to revise before accepting them.

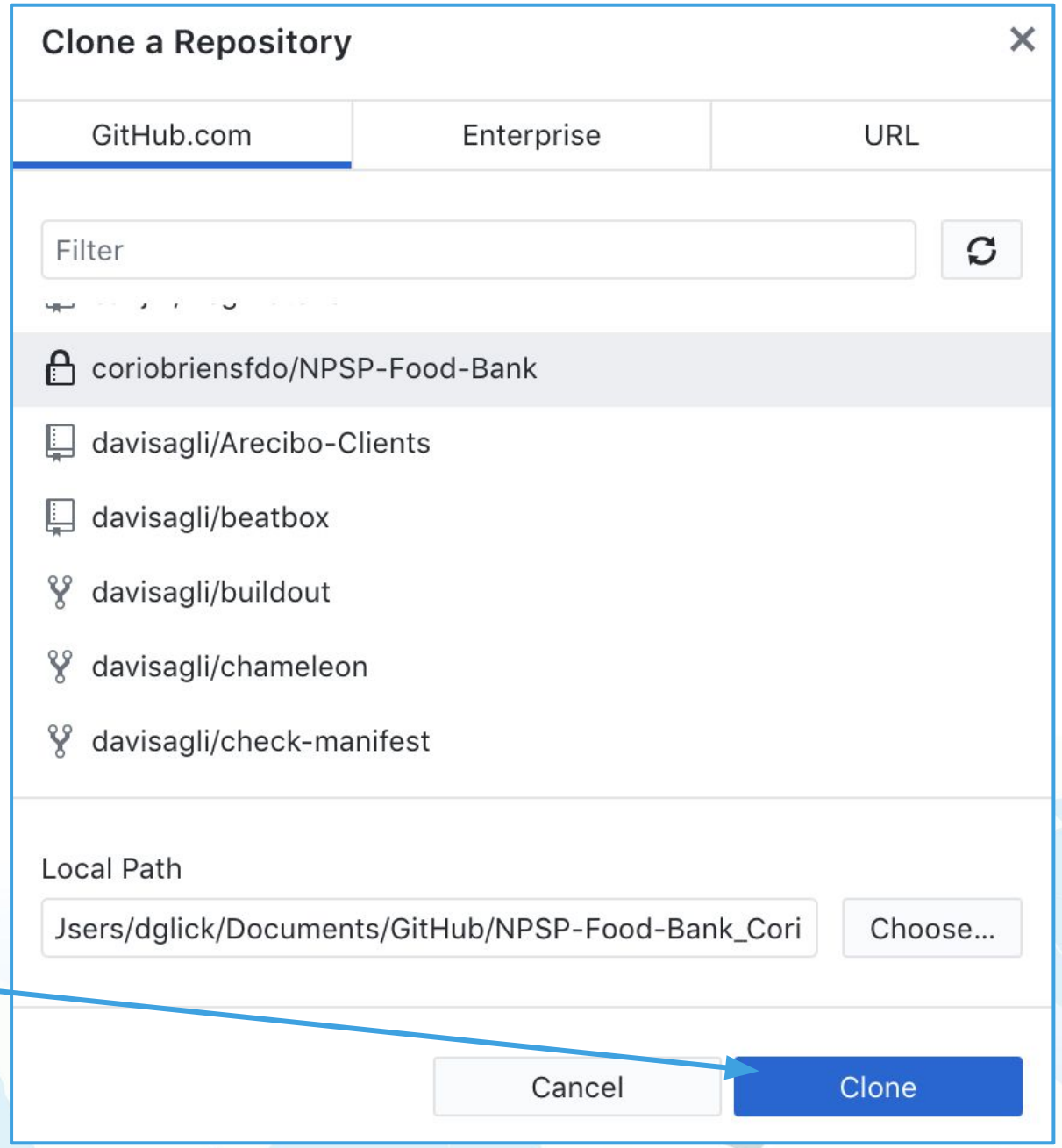
Clone Your Partner's Repository

Use GitHub Desktop

1. Click **Current Repository**, then **Clone Repository.**



2. Choose your partner's repository on the GitHub.com tab.
3. Click **Clone.**

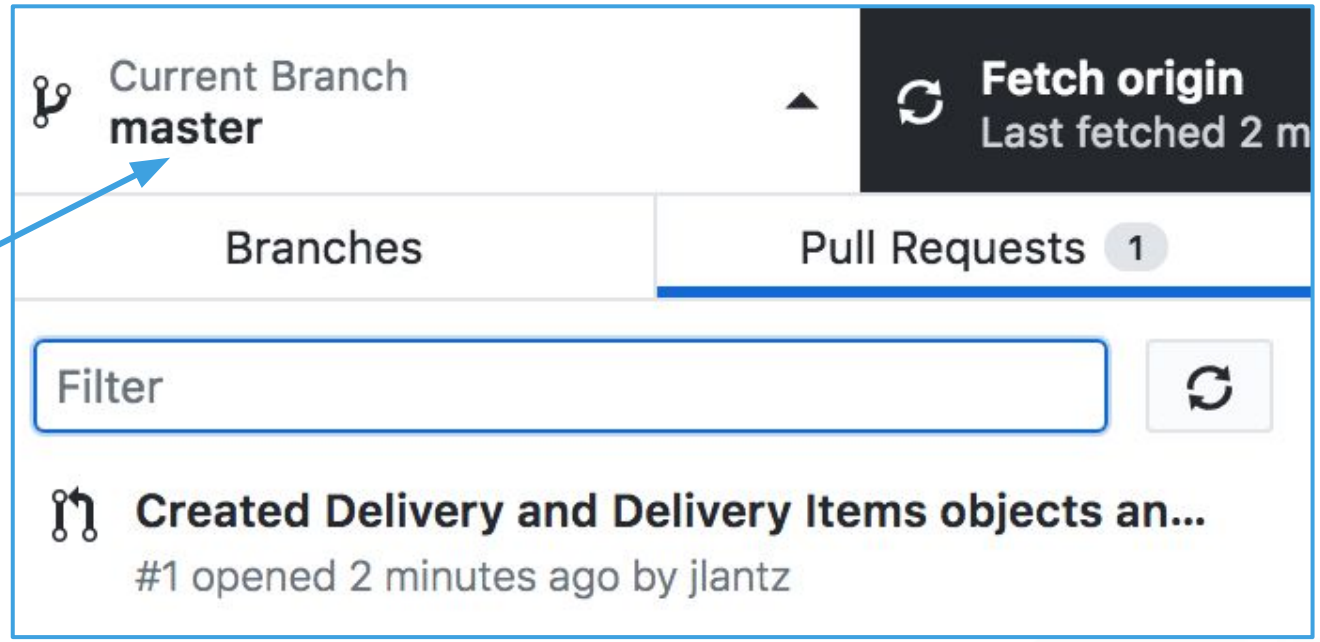


Switch to the Feature Branch

Use GitHub Desktop

A feature branch is the place where your partner's work lives, before it's combined with the "master" version.

1. Click **Current Branch**.
2. Make sure the **Pull Requests** tab is selected.
3. Click your partner's pull request to select it.



Get a Scratch Org Using the QA Org Flow

Start in GitHub Desktop

In the **Repository** menu, click **Open in Terminal** to open a terminal window in the folder for your partner's repository.

Enter the following command to create a new scratch org for doing QA:

```
cci flow run qa_org --org dev
```

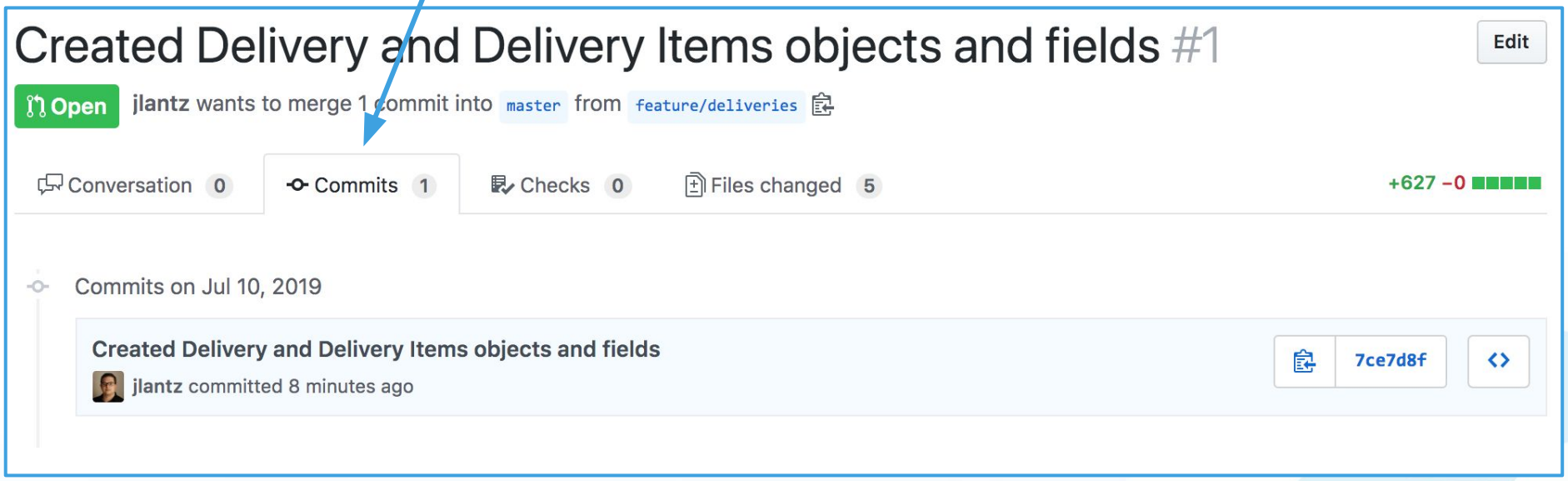
When it's done, enter the following command to open the org:

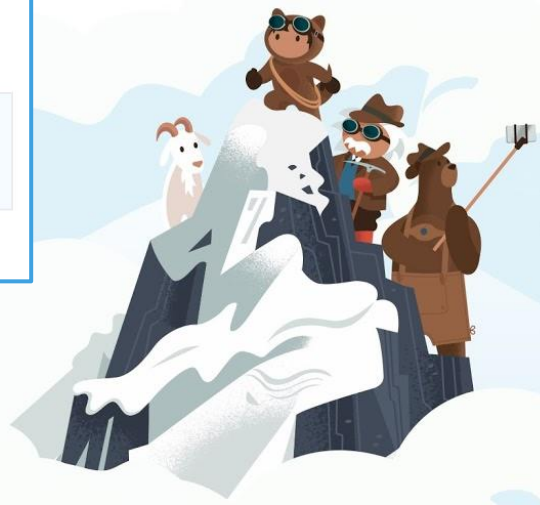
```
cci org browser dev
```

View Commit History for Pull Request

Open GitHub in browser from GitHub Desktop

1. On the **Branch** menu, click **Show Pull Request** to open the pull request in GitHub in your browser.
2. Click on the **Commits** tab of the new pull request to see a history of the changes.

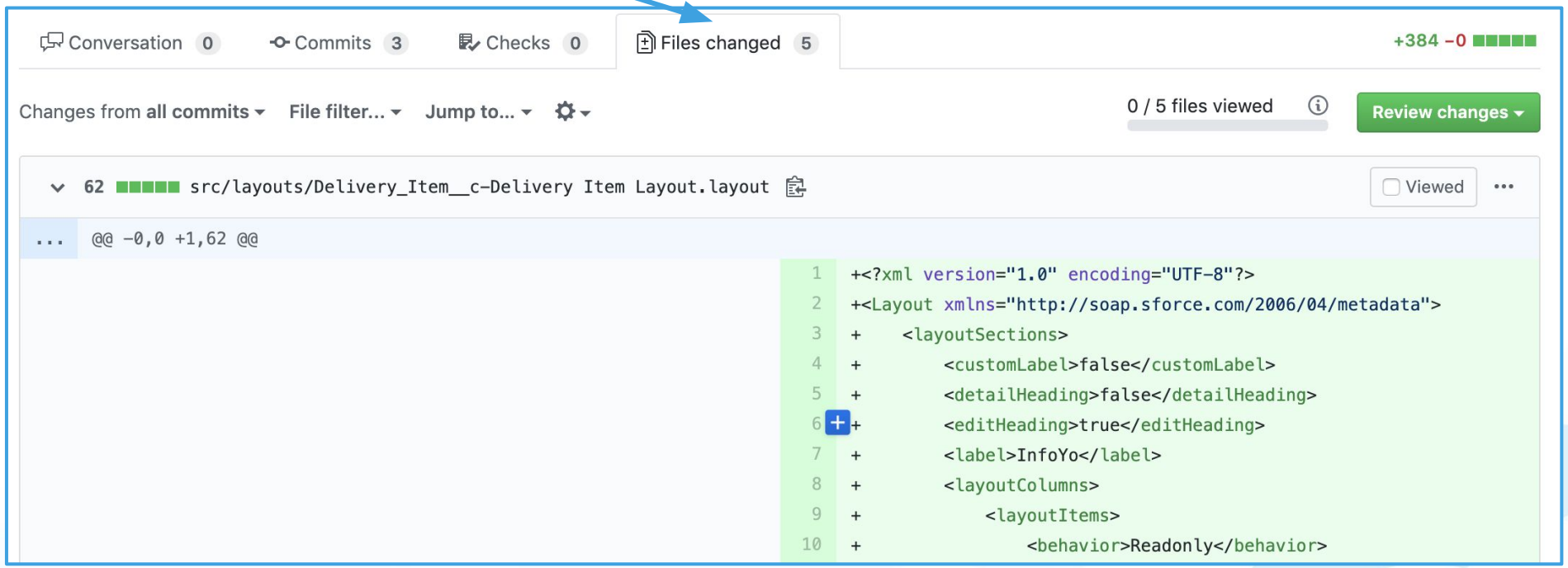


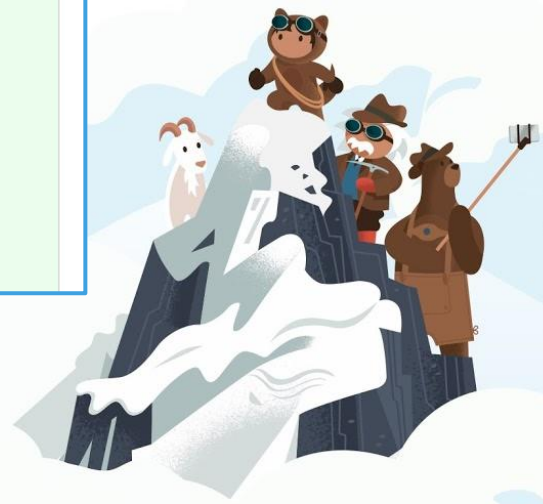


View a Diff of Changes in the Pull Request

Use GitHub

Click on the **Files Changed** tab of the pull request to see the changes made to the project's files.

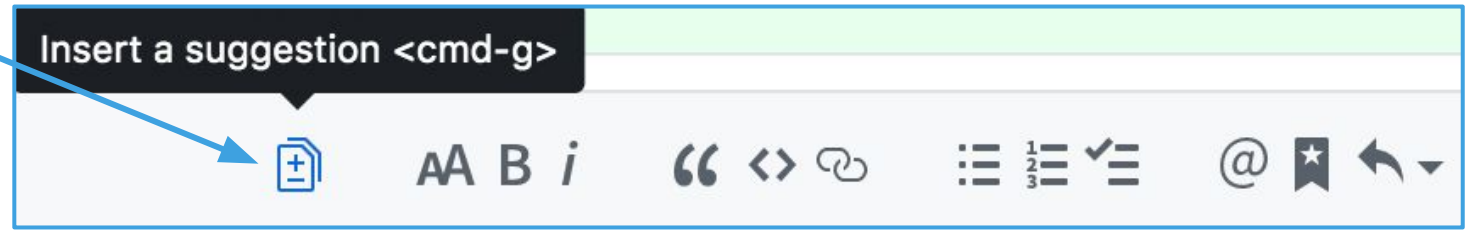




Propose a Change

Use GitHub

1. Find the line containing **<label>Information</label>** and click the small plus sign to the left of the line.
2. Click the **Insert a suggestion icon** to suggest a change:



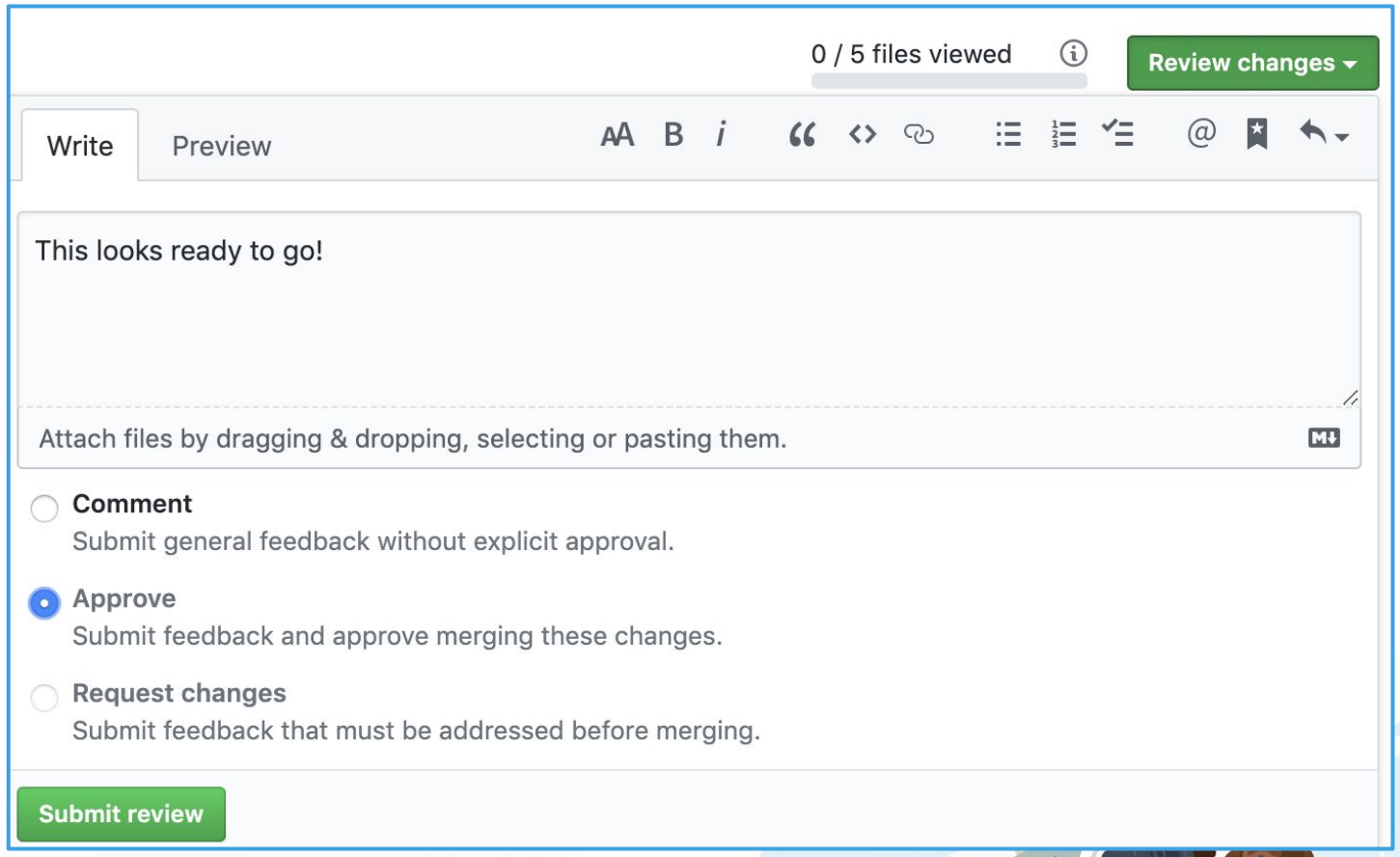
3. Type in your suggested change and click **Comment** to save the comment.
4. Click **Submit Review** to submit your review.

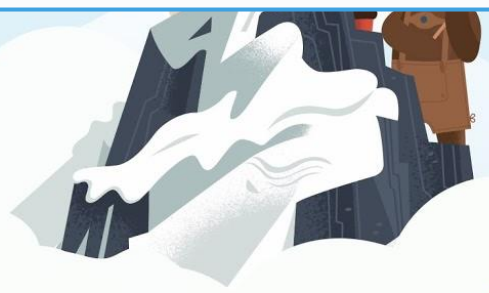
NOTE: Normally the review would be done by another member of the team

Accept the Proposed Change

Use GitHub

1. Click the **Review Changes** button.
2. Enter **a message** for the person who submitted the pull request.
3. Click the **Approve** radio button if you are satisfied with the changes.
4. Click the **Submit review** button.

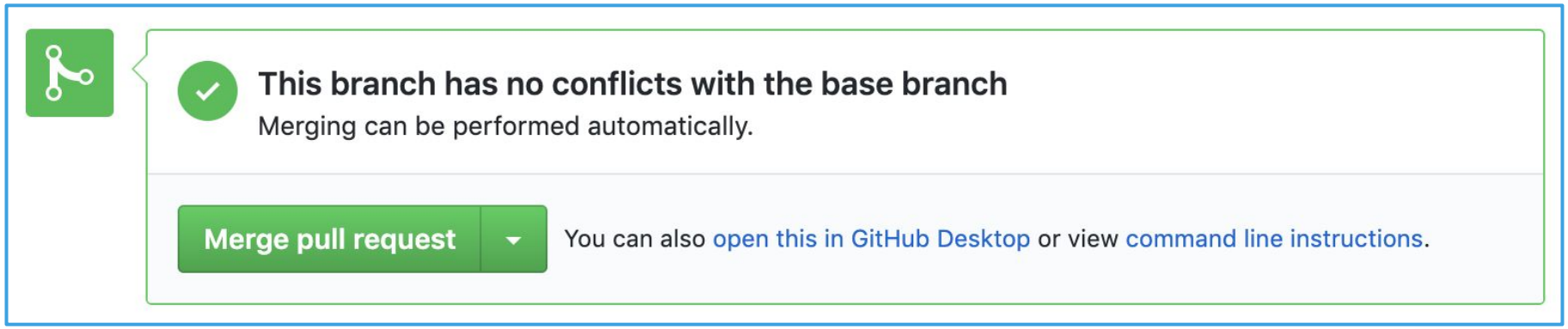




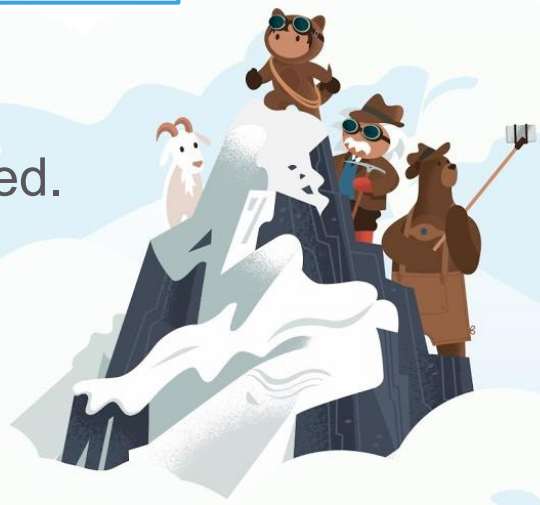
Merge the Pull Request

Use GitHub

On the **Conversation tab** of the pull request, use the **Merge Pull Request** button to merge your feature branch into the master branch.



Next use the **Delete Branch** button to delete the feature branch. Feature branches are designed to be temporary and are no longer needed once merged.



Section 4: Recap

Great job! In this session you:

- Cloned your partners GitHub Repo.
- Reviewed their work in a nifty new scratch org.
- Proposed a change to their work.
- Accepted changes to your own work.

Additional Resources

CumulusCI Documentation
https://cumulusci.readthedocs.io

DevOps Group in the Hub
https://powerofus.force.com/s/group/0F980000000PSRHCA4/salesforceorg-devops-tooling

CumulusCI on GitHub
https://github.com/SFDO-Tooling/CumulusCI


salesforce
Quality Engineering
At Salesforce.org

Presenter

Jess Ingrassellino

Director, Quality Assurance

Life of a Quality Engineer

A salesforce sprint story

This is Jo.

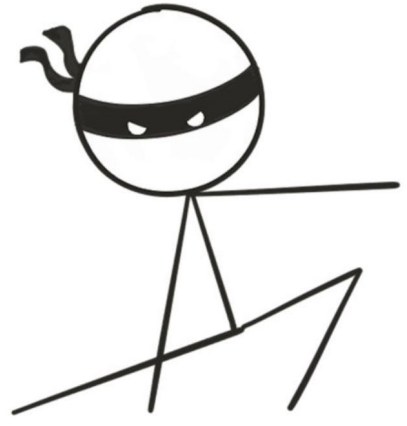
Jo is a QE at Salesforce.org

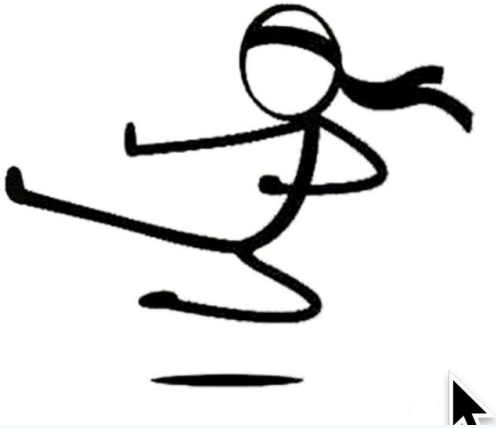
Jo is on a scrum team, and does exploratory testing.

There are other kinds of QEs at Salesforce.org

Accessibility testers focus on testing accessibility issues.

Automation testers develop code that tests our UI.



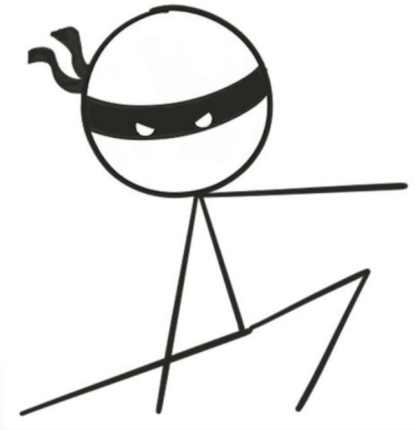


Life of a Quality Engineer: Before a Feature is Built

A salesforce sprint story

Jo loves to learn from their peers who are project managers, developers, designers, doc writers, and on the CCE team. Jo collaborates with all team members to learn about the feature!

Automation testers and accessibility testers are also learning about features, getting test cases ready for the automation suite, and bringing up risk-areas for accessibility.



Life of a Quality Engineer: Beginning of Sprint

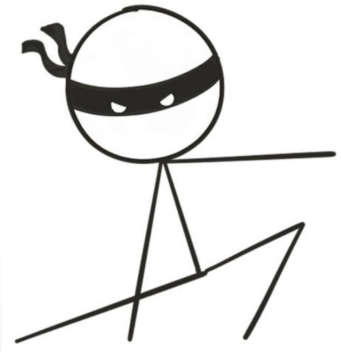
A salesforce sprint story

When a sprint starts, QEs plan something called a Test Charter. This document contains all the test plans, and the QE works tirelessly reviewing documents and meeting notes to get the test charter done BEFORE development work is done.

WHAT IS A TEST CHARTER?

A Test Charter is a collaborative template for planning and recording test efforts.

EVERYONE contributes to the document.

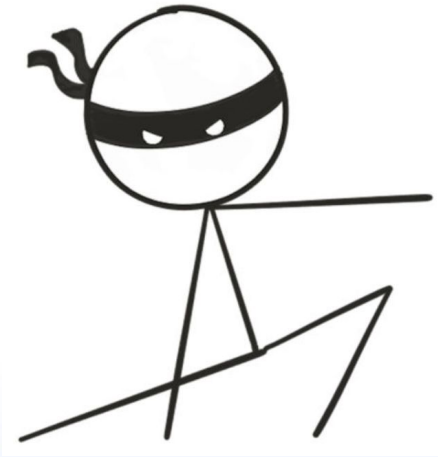


Life of a Quality Engineer: Mid-Sprint

A salesforce sprint story

Fun Fact!

Testers for a single scrum team are working on charters for multiple developers and features! Sometimes, they are also testing bug fixes at the same time.



Life of a Quality Engineer: Sprint Close

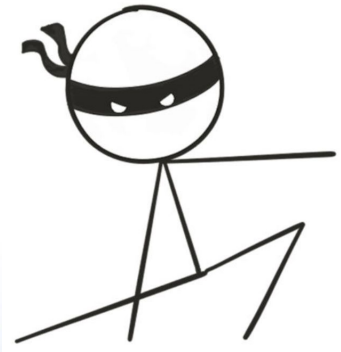
A salesforce sprint story

Prioritization

We use risk-based testing techniques to help us determine what parts of our highly complex system need the most attention.

Regression

We have regression test cases for every product, and once the merge freeze happens, we run regression on every product to make sure that we didn't break existing functionality.



So You Want to Test in Your Sandbox...

When the sprint ends… the release comes to you!

Where you are	Your Approach
Just starting out	Learn about our regression process. Create your testing plan.
We do some testing, but……...	Analyze and formalize with approaches like risk-based and tiered testing. Get your regression time under 2 hours.
We test well, but want to apply automation and “level up”	Familiarize yourself with CCI, Python, and Robot Framework. There are some changes that you may need to make to fork and run our automation suite on your product, but it is a great next step for teams with automation support.


thank you