Ensuring reasonable scopes for your PRs

Childhood Cancer Data Lab x Alex's Lemonade Stand

Recall, a **good pull request** should be…

Focused on a single task

Manageable for review without overly fatiguing your reviewer

Here, we'll offer tips and strategies that will help to achieve these goals

Plan ahead, but it's ok to revise your plan

Update the SingleR workflow to take as input a single model and output a single set of annotations using label.ont #365

Closed allyhawkins opened this issue on Jun 30 · 5 comments

allyhawkins commented on Jun 30 Member

We have decided to update our usage of `SingleR` so that we annotate each SCE object with a single reference and single set of labels. The output would be a single column with cell type annotations. Currently, the workflow is designed to take as input a list of reference models to use for annotation. We should make the necessary changes so the workflow expects

Pipelines
S sci-sprint
Closed

...After some discussion...

sjspielman commented 3 weeks ago Member

> I think this would simplify everything so that we only work with the `label.ont` moving forward, which is what we want.

👍

Thanks for the feedback here! I'm going to split this issue up into 2 and assign myself to them -

You can break out smaller issues from the big one

Use only `label.ont` reference label for SingleR celltyping #380

Closed **sjspielman** opened this issue 3 weeks ago · 1 comment · Fixed by #382

sjspielman commented 3 weeks ago • edited Member

Broken out of #365

The cell type annotation workflow should be updated to only deal with *one* reference label. In our case, we'll default to `label.ont`.

- The `build-celltype-ref.nf` workflow will need to be updated to only return the label defined by `params.label_type` (`params.label_name`?) which will be assigned `"label.ont"`

Use only one reference model per project #381

Closed **sjspielman** opened this issue 3 weeks ago · 1 comment

sjspielman commented 3 weeks ago Member

Broken out of #365

The cell type annotation workflow should be updated to only take one model.

Thoughts to this end:

- `groupTuple()` no longer needed

Don't commit what you shouldn't commit

Everybody, say it with me! Never add/commit anything without running your two new favorite commands, `git status` & `git diff`

- You may find an accidentally modified file that needs to be restored (`git restore <file>`)

When opening the PR, but *before filing the PR*, make sure the "Files Changed" view looks as you expect

- This is worth taking the extra time to do!
- Most of the time, I find something to tweak before actually filing the PR. This helps my reviewer out, which helps the overall process move forward smoothly.

And conversely, be sure to commit what you *should* commit

Before filing the PR, go back to the issue and make sure you have stayed in scope

- Are there more tasks you should have done but forgot to do?
- Are there extra tasks you did that are beyond the scope of the associated issue?

Remember, you can "undo" commits (but still preserve the commit history) with `git revert <commit-to-revert>`

- This is another reason it helps to have small commits with informative messages that actually match the work you did

Creating stacked pull requests

Setting up for a stacked pull request

- Rather than branching off of `main`, you create a new branch off of your feature branch
- Below **feature** is the *base branch* for **feature**
  - In other words, feature is "stacked on" feature
  - When feature is ready to go, it is merged into feature
  - Then only feature will need to be merged into main

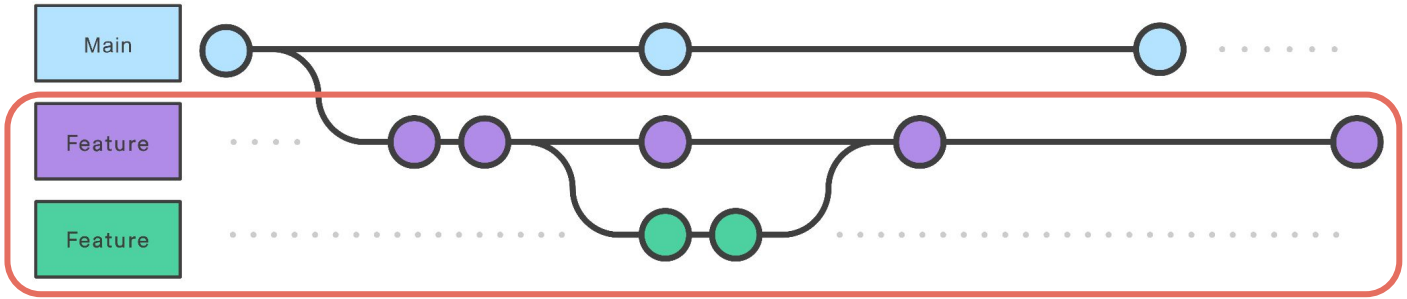


Image adapted from Atlassian

Why should you stack your branches?

Recall, a **good pull request** should be…

- **Focused** on a single task
- **Manageable** for review without overly tiring your reviewer

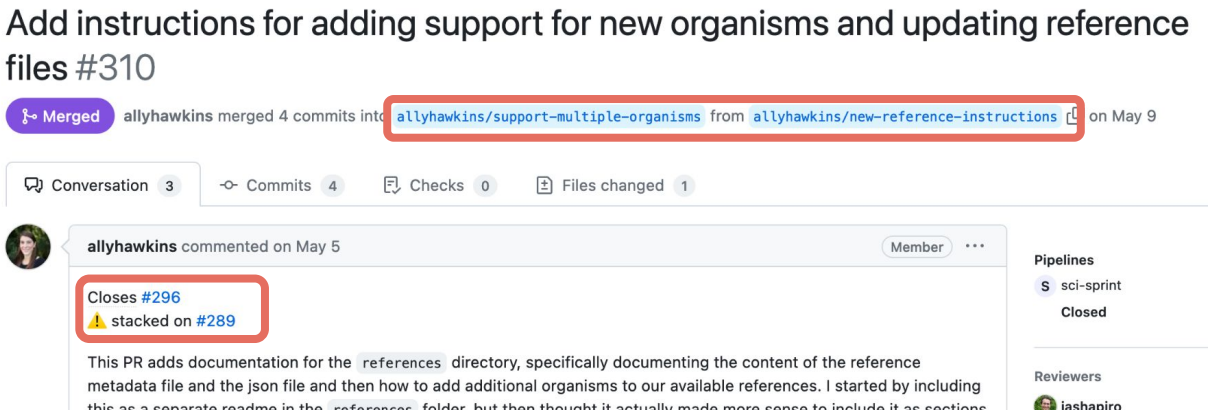
Sometimes, what seems like a "single task" actually has several moving parts that, if combined into one PR, can lead to review fatigue. E.g., "analyze dataset" might seem like a single task, but it's probably not!

- The data might need cleaning first
- Some custom functions may need to be written
- There may be several stages to the analysis itself
- Perhaps those custom functions need tests and/or documentation (hint: they do)
- The analysis may need documentation (hint: it does)

Stacking can help move the project forward, faster

Rather than waiting to file PR #2 until PR #1 is merged, you can get PR #2 going now!

But, you need to communicate with your reviewers!



Stacking can help team members work together on the same code while avoiding merge conflicts

Add test functions for ASW #187

Merged **allyhawkins** merged 12 commits into `cbethell/calculate_batch_asw` from `allyhawkins/asw-test-functions` 2 weel

Conversation 20 | Commits 12 | Checks 0 | Files changed 1

allyhawkins commented 2 weeks ago Member

Closes #182
Stacked on #181

This PR adds in test functions for all of the functions being added in #181. To set up actually running the test function I

The merge order matters

Below, feature is merged into feature first

As a consequence, feature now contains work from two branches, which makes the remaining PR less focused and possibly confusing for your reviewer!

Tip: It will help to wait for reviews on *both PRs* to come in before doing any merging.

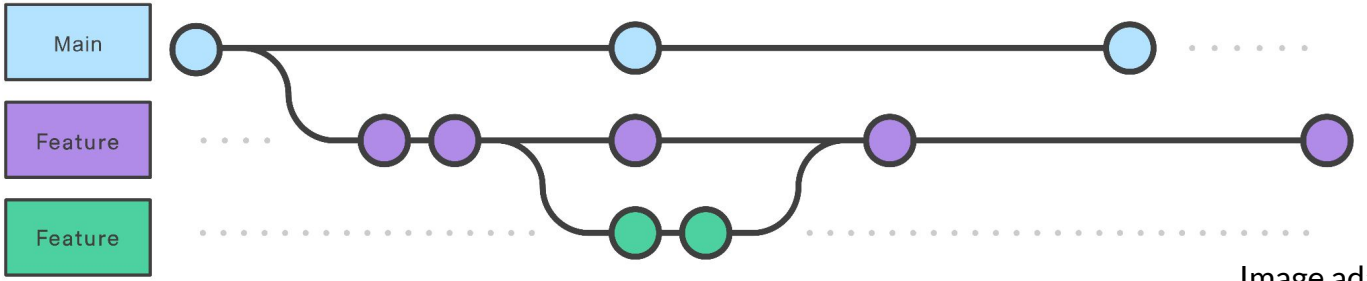


Image adapted from Atlassian

The merge order matters

Unlike the image below, we could have merged feature into main first

- In this case, each PR retains its original focus/scope
- The base branch for feature will automagically 🪄 get changed to main in the PR, *if and only if the* feature branch is deleted
- We'll see this in action soon via live demo!

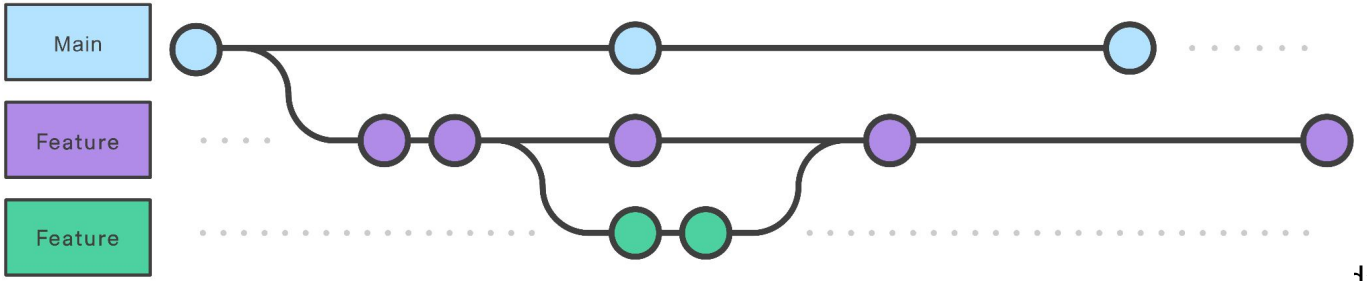
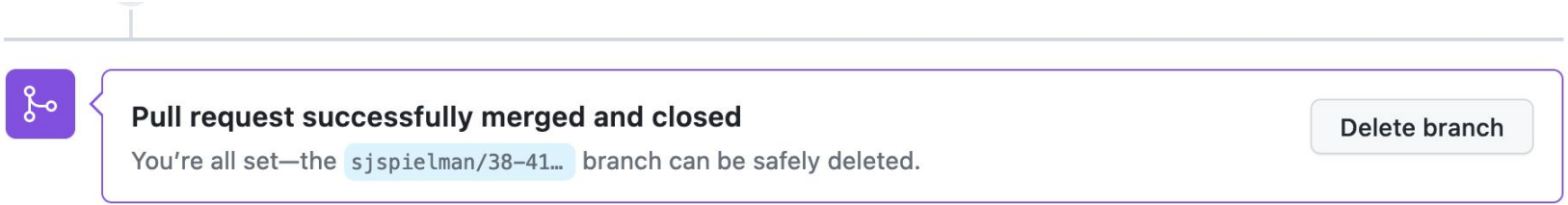


Image adapted from Atlassian

What to expect when you're expecting some stacking

Don't forget to **delete your branch** after it's merged, so the next branch in the stack heads to the right base



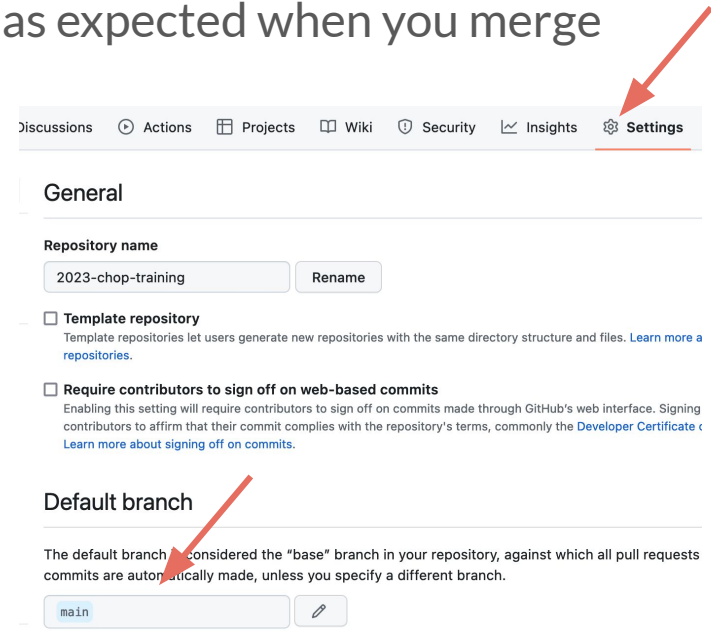
What to expect when you're expecting stacking

Some GitHub systems you have in place may not work as expected when you merge into a *non-default branch*

Writing `Closes #<issue-number>` in the PR comment will not automatically close the PR (ask me how I know 😭)

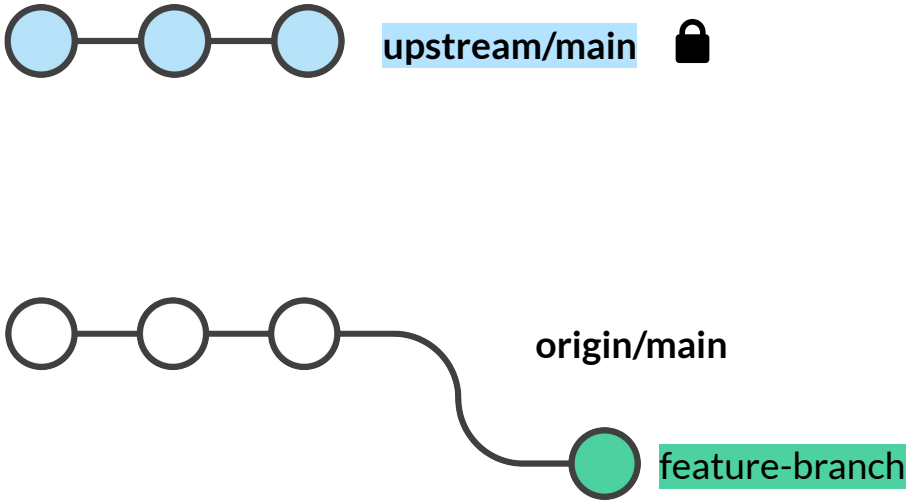
Your GitHub actions may not get run, depending on how you set them up

- They will hopefully get run eventually once you hit the branch that *is* being merged into `main`, but not necessarily at every step along the way



Stacking branches when working in a fork

Recall the **forking** workflow



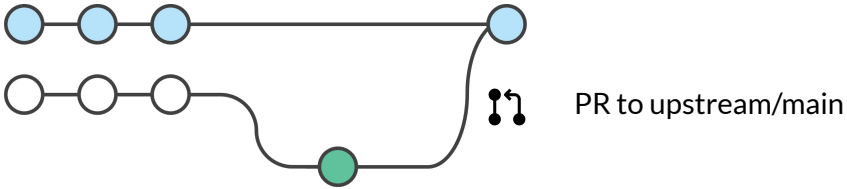
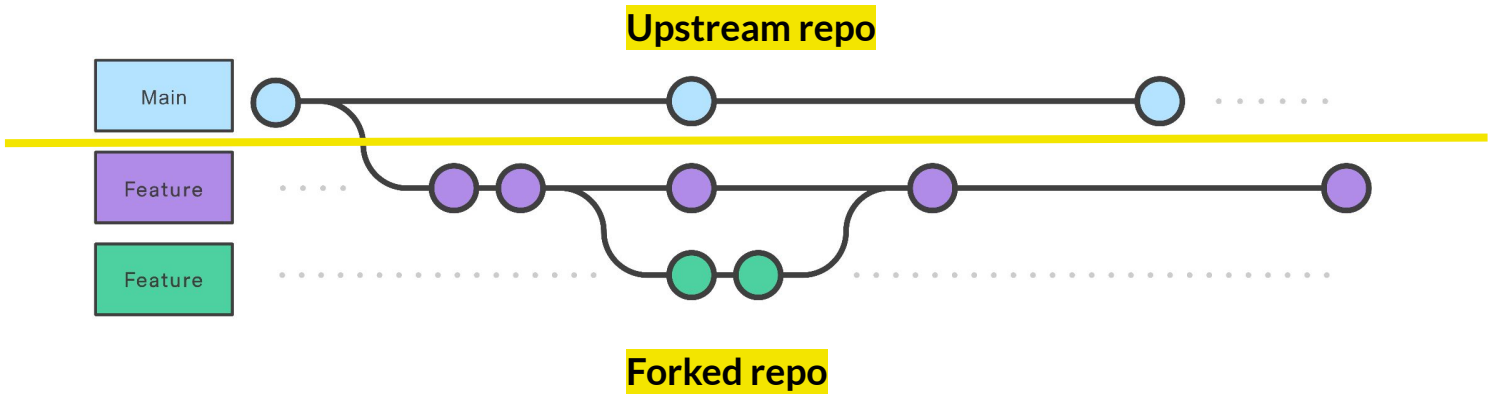


Image adapted from https://www.atlassian.com/git/tutorials/using-branches

Stacking and forking require a little extra thought

If you stack in your fork, then stacked PR for the stacked branch has to also be filed *in your fork*

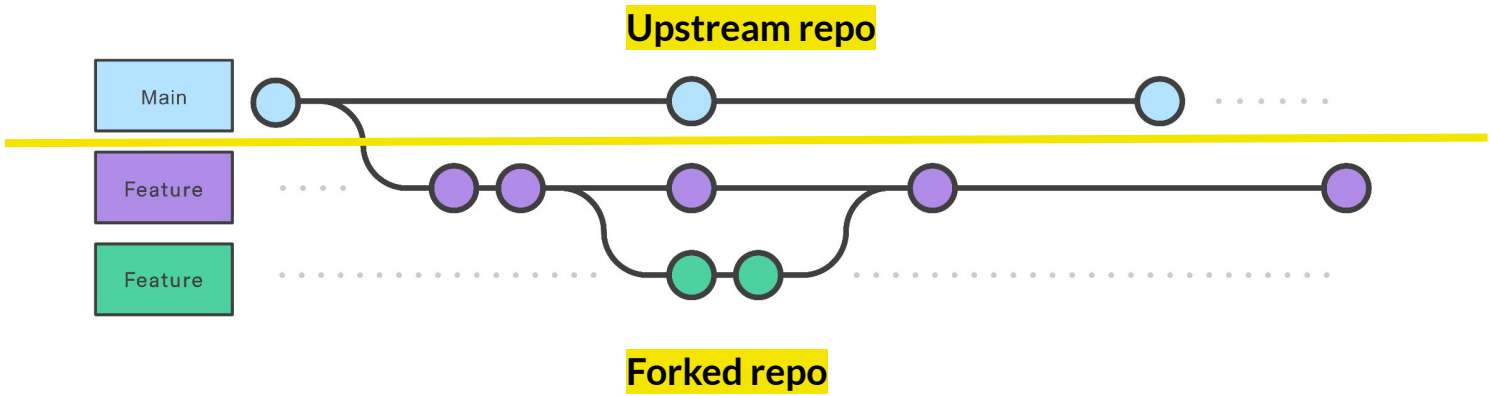
- The initial PR still goes into the upstream repository
- Merge order matters here a lot more! We recommend merging at the "top" of the stack (aka the youngest branch) *first*



Stacking PRs in forks may or may not meet your needs

This will also cause the overall project history to be spread across multiple repositories. You'll have to decide if this is a reasonable choice for a given project.

- In the live demo, we'll see an approach using `git cherry-pick` that can *approximate* stacking in forks, without actually stacking but still helping your reviewer avoid fatigue



To the demo!