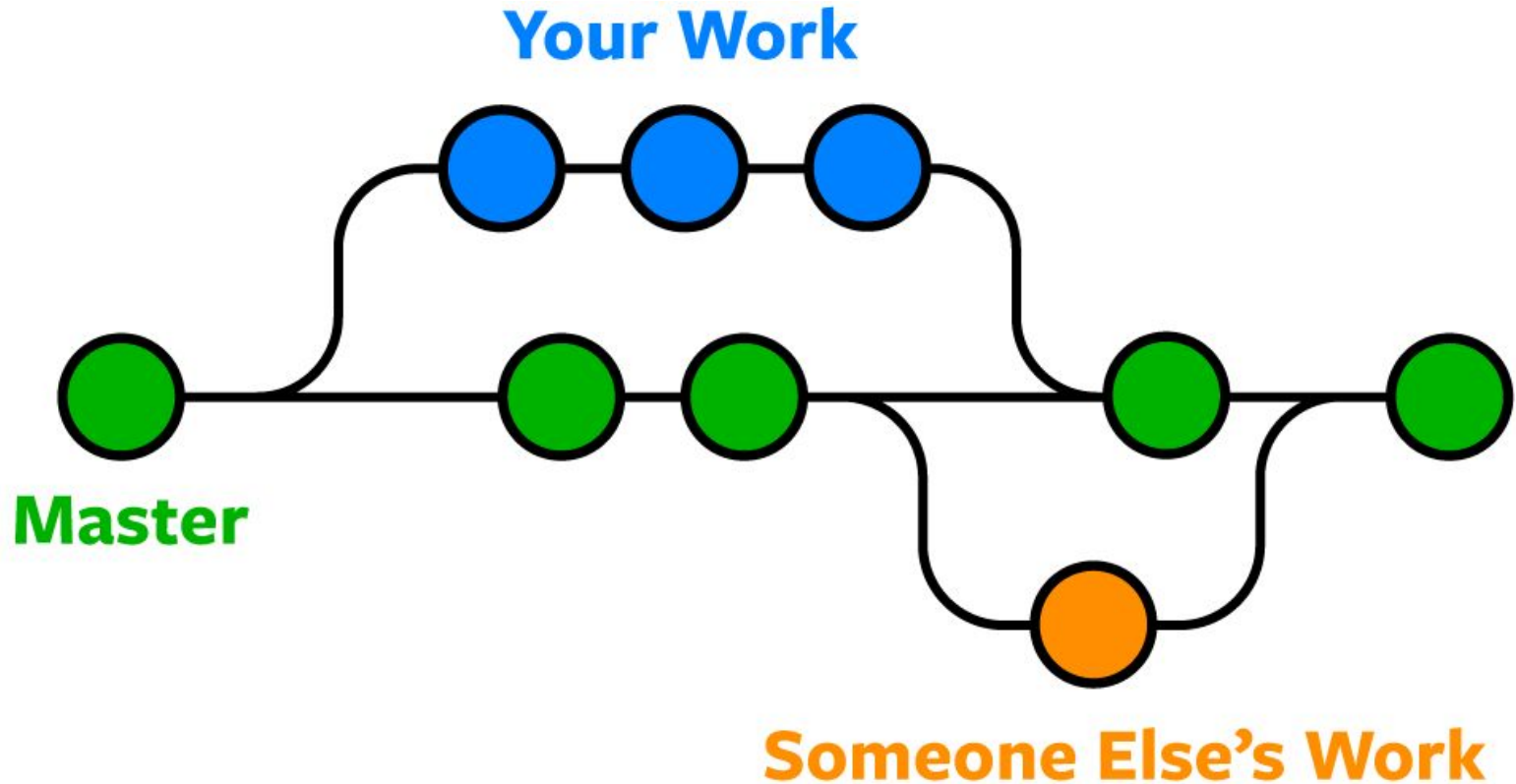


141B Lecture 2

Git and github

James Sharpnack

Versioning Systems (git)



Overview

- maintaining versions of human readable files
- facilitate collaboration
- **git** is distributed - as opposed to cvs and subversion
- **.git** folder contains history of changes made to “versioned” files
- **github.com** is a service that provides servers with repo and some other features: issues, project boards, pull requests, renders ipynb & md
- Some IDEs (pycharm, RStudio) have built in git (I don't use this)

What should be tracked?

Definitely: code, markdown documentation, bash scripts/makefiles, ...

Possibly: logs, jupyter notebooks, images (that won't change), ...

Questionable: processed data, static pdfs, ...

Definitely not: full data, continually updated pdfs (other things compiled from source code), ...

Versioning commands

`git init` : initialize empty `.git`

`git add ____` : track the changes to file (put in staging area called the “index”)

`git commit -m “message here”` : commit the changes made

`git status` : see what you tracked and untracked changes since commit

`git log` : see history

`git diff` : see differences within files

Messages

- concise but descriptive, add body if more explanation is needed
- 50 char limit recommended
- imperative mode
- don't overthink it, but try

remove offending gh workflows



dajmcdon committed 13 days ago

add strawman and template (#6) ...



jsharpna and **dajmcdon** committed 13 days ago

Collaboration commands

`git clone` : copy repo

`git pull repo-name branch-name`: fetch changes (puts them into `.git`) and merges changes to branch

`git remote add repo-name http://github.com/path...` : `repo-name` is now a shorthand for the full path

`git merge` : merge two branches (e.g. `origin/main` and `main`)

`git push` : push changes to remote repository (typically github), will complain if can't be merged

Resolving conflicts

```
<<<<<< HEAD:mergetest
```

```
This is my third line (your commit)
```

```
=====
```

```
This is a fourth line I am adding (commit from merge)
```

```
>>>>>> 4e2b407f501b68f8588aa645acafffa0224b9b78:mergetest
```


Daily operations

```
git add code.py
```

```
git commit -m "add header flag to df reader"
```

```
git pull or git fetch; git merge
```

(resolve conflicts if any and commit again)

```
git push
```

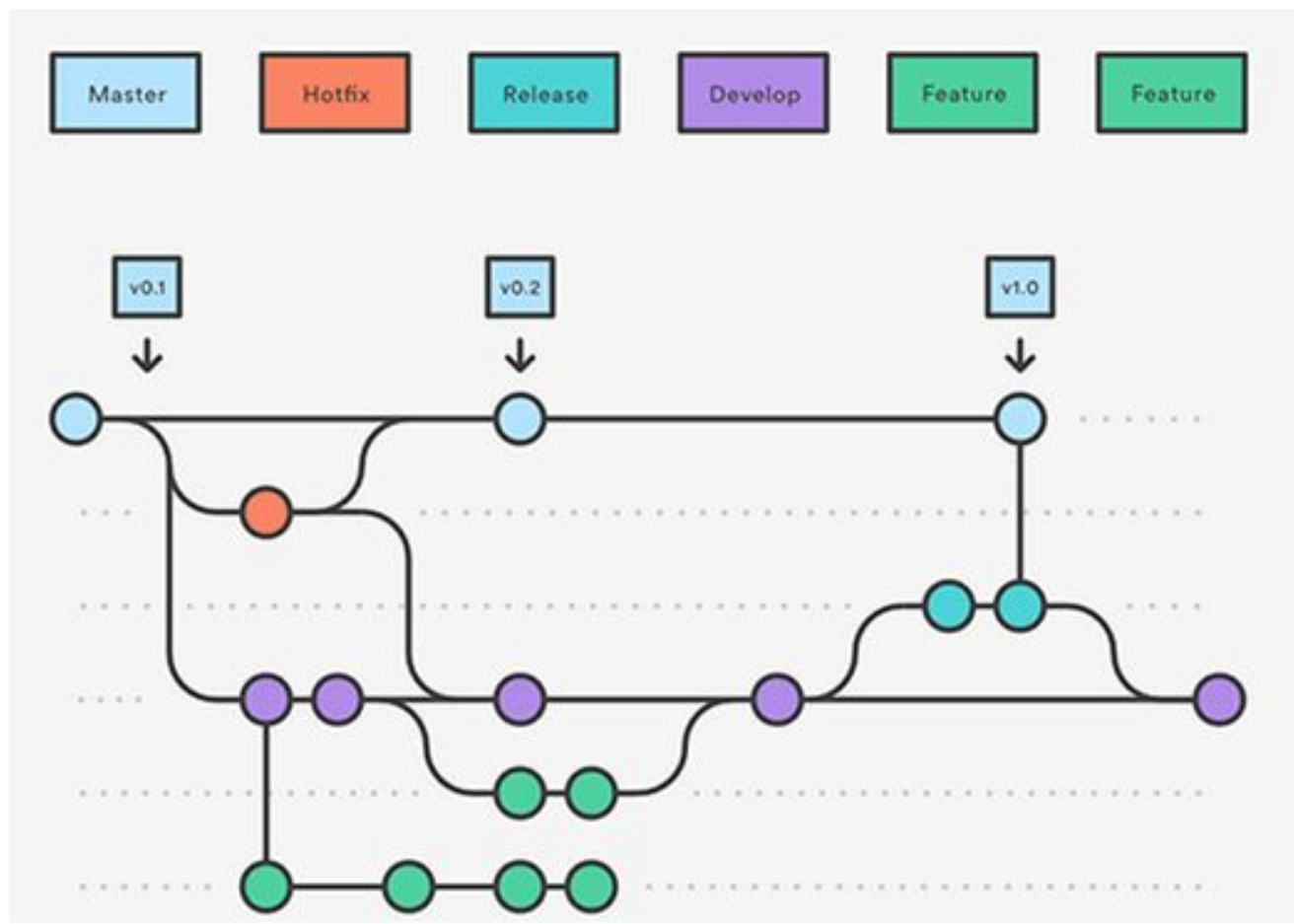
- commit often (small “atomic” commits)
- take messages (your own, and others) seriously
- commit complete, tested code

.gitignore

`.gitignore` : tells git what to ignore during `git add`

- git status will not mention ignored files
- git commit -a will not add them
- easy mistake is to delete `.gitignore`
- <https://git-scm.com/docs/gitignore>

Gitflow



Merging branches without a Pull Request

```
git checkout -b myfeature dev-aardvark : create branch
```

```
... do some work ...
```

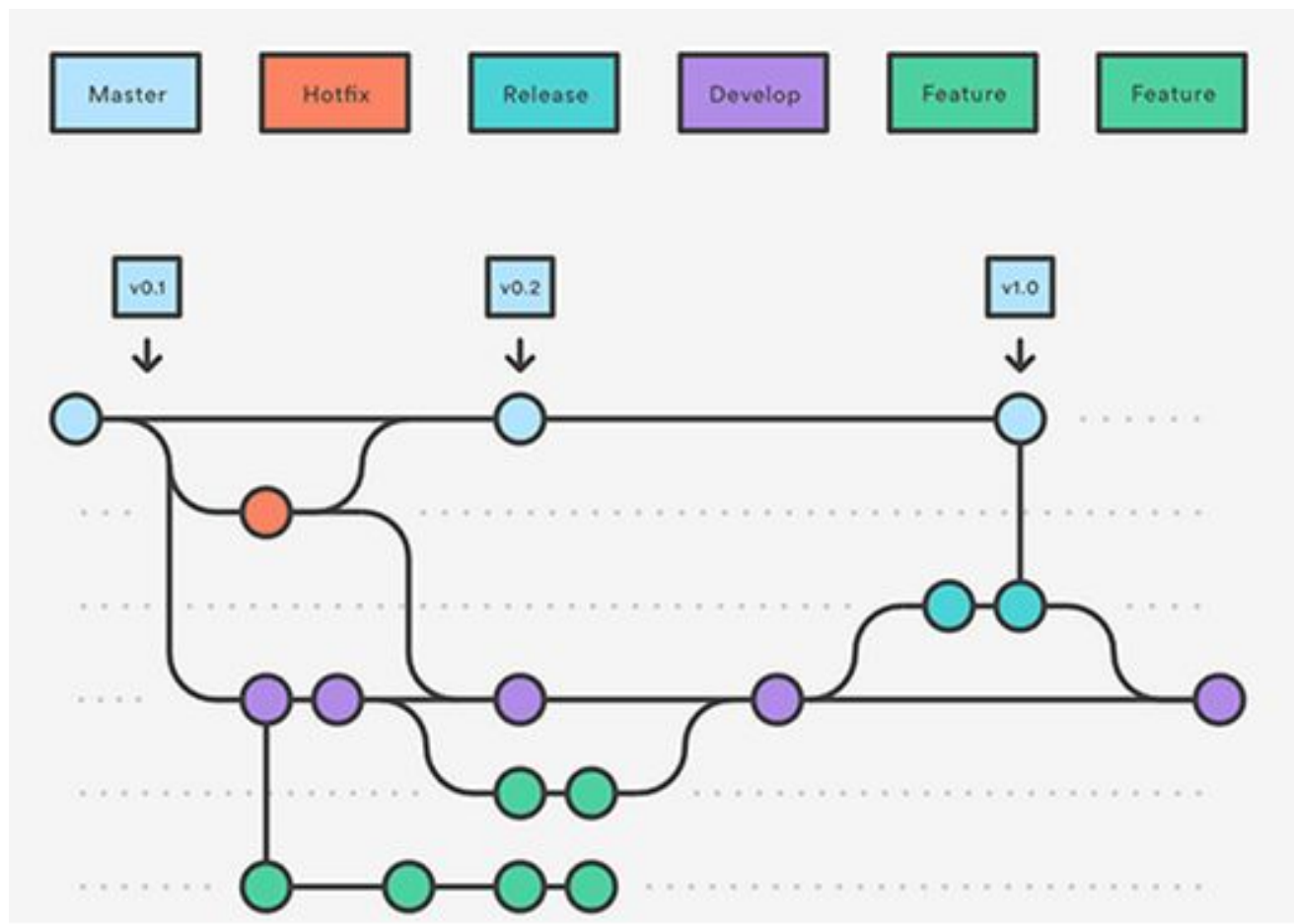
```
git checkout dev-aardvark
```

```
git merge myfeature : to be replaced with pull request for develop  
and main
```

```
git branch -d myfeature : delete feature branch
```

```
git push origin dev-aardvark : push develop changes to github
```

Gitflow



🔗 main ▾

🔗 5 branches

📦 0 tags

Switch branches/tags



Find or create a branch...

Branches

Tags

✓ main

default

dev-aardvark

develop

fcast-utils

organization

[View all branches](#)

template.yaml

Create check-release-fcast-1

test CI on strawman

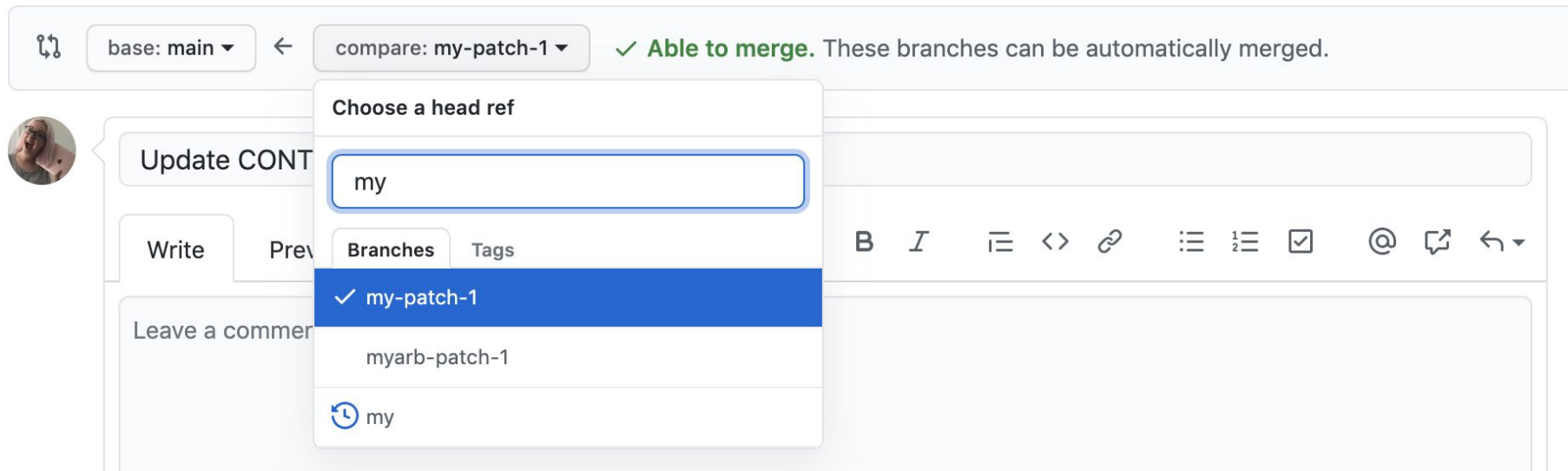
Update README.md

Public repository for CMU Delphi forecasting efforts for SARS-C

github.com stuff : pull requests

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](#).



The screenshot displays the GitHub pull request interface. At the top, there's a status bar with a merge icon, a dropdown for 'base: main', a left arrow, a dropdown for 'compare: my-patch-1', and a green checkmark with the text 'Able to merge. These branches can be automatically merged.' Below this, on the left, is a user profile picture and a comment box with the text 'Update CONT'. To the right of the comment box is a rich text editor with a toolbar containing icons for bold (B), italic (I), list (≡), code (<>), link (🔗), table (≡), table of contents (1/2 ≡), checkmark (☑), mention (@), quote (🗨), and undo (↶). A dropdown menu titled 'Choose a head ref' is open, showing a search bar with 'my' entered. Below the search bar are two tabs: 'Branches' and 'Tags'. Under the 'Branches' tab, there are three options: '✓ my-patch-1' (highlighted in blue), 'myarb-patch-1', and a circular arrow icon next to 'my'.

github.com stuff : issues

◀ The no-conflict mode should be the default behaviour #12395

[Edit](#)[New Issue](#)

thewebdreamer opened this issue 3 days ago · 10 comments



thewebdreamer commented 3 days ago

The no-conflict mode should be the default behaviour. Why would a Bootstrap client need to implement this?



cvrebert commented 3 days ago

I believe no-conflict-is-not-the-default is the norm for jQuery plugins?



thewebdreamer commented 3 days ago

It is true that it is the norm for jQuery plugins.

Couldn't there be a clash with other jQuery plugins with the current implementation of Bootstrap though?

Labels



js

Milestone



No milestone

Assignee



No one assigned

Notifications



Subscribe

3 participants



github.com stuff : project boards

The screenshot displays a GitHub Project Board for the organization 'octo-org'. The top navigation bar includes links for Repositories (12), People (14), Teams (15), Projects (6), and Settings. The board is titled 'octo-org projects for Q4' and was updated just now.

The board is organized into three columns: 'To do' (3 items), 'In progress' (0 items), and 'Done' (0 items). A '+ Add column' button is located on the right side of the board.

The 'To do' column contains three cards:

- Welcome to GitHub Projects**: A card with a checklist of tasks:
 - ☒ Create a new project
 - ☒ Give your project a name
 - ☐ Press the `?` key to see available keyboard shortcuts
 - ☐ Add a new column
 - ☐ Drag and drop this card to the new column
 - ☐ Search for and add issues or PRs to your project
 - ☐ Manage automation on columnsAdded by jmarlena
- Cards**: A card explaining that cards can be added to track the progress of issues and pull requests. You can also add note cards, like this one! Added by jmarlena
- Automation**: A card with a link to 'Automatically move your cards' to the right place based on the status and activity of your issues and pull requests. Added by jmarlena

At the bottom of each column, there is a 'Manage' link. The 'To do' column is automated as 'To do', the 'In progress' column is automated as 'In progress', and the 'Done' column is automated as 'Done'.

good repo git standards

1. **Commit often with concise descriptive messages**, code bite-sized features, debugged and tested
2. **Develop and feature branches**: Add new features (forecasters, utilities) by branching off of develop, use naming scheme dev-*, merge with a PR and assign reviewer
3. **Main is for releases**, merging into main should be for hotfix, naming scheme is hotfix-*, or new release from develop (maintainer will make a PR against main)
4. **Issues** should be generated for desired features or bugs in release candidate (dev-*), bugs in main (hotfix-*), etc.
5. **Project board** tracks issues to be resolved in each release