EPIC GitHub Tutorial (Part 2): Contributing to UFS/EPIC Repositories

Natalie Perlin, NOAA/EPIC, natalie.perlin@noaa.gov

https://github.com/NOAA-EPIC/training-github

EARTH PREDICTION INNOVATION CENTER (EPIC)

Welcome!

- This is Part 2 of the EPIC GitHub Tutorial on Contributing to UFS/EPIC Repositories
- View Part 1: https://epic.noaa.gov/tutorials/?playlist=afde205&video=9e65c0d
- All the materials and tutorial presentations (Part 1 and Part 2) could be found at https://github.com/NOAA-EPIC/training-github
- Get all the materials and tutorial presentations (Part 1, Part 2):

```
git clone https://github.com/NOAA-EPIC/training-github.git
cd ./training-github
```

- The directory ./training-github/ will contain all of the materials for the tutorial.

Recap of the Part 1: Git and GitHub Basics

- Basic Git and GitHub terms and concepts introduced
- Basic Git configurations and commands discussed
- Example given on setting up a SSH key pair for GitHub authentication
- A new local repository was created, initialized, pushed to GitHub
- Modifications made, committed, local changes tracked, logs viewed
- A new branch made, different branches compared
- Started topic: public GitHub repositories, forks and clones

GitHub Tutorial Part 2: Working with Remote Repositories

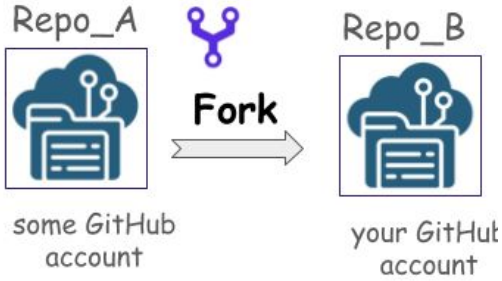
- UFS/EPIC Public Repositories, Forks and Clones
- Branches and Tags
- Git Workflow: from Local Spaces to Remote Repos
- Fetching and Merging Remote Branches
- Resolve Merge conflicts
- Making Pull Requests
- GitHub Issues and Discussions

Common GitHub repositories used by UFS community

- https://github.com/ufs-community/ufs-weather-model
- https://github.com/ufs-community/ufs-srweather-app
- https://github.com/ufs-community/land-DA_workflow
- https://github.com/ufs-community/uwtools
- https://github.com/ufs-community/UFS_UTILS
- https://github.com/ufs-community/ccpp-physics (forked from NCAR/ccpp-physics)
- https://github.com/hafs-community/HAFS (forked from hafs-communityHAFS)
- https://github.com/ufs-community/global-workflow-AR (NOAA-EMC/global-workflow)
- https://github.com/NOAA-EMC/UPP
- https://github.com/noaa-oar-arl/NEXUS
- https://github.com/NOAA-EMC/AQM-utils
- https://github.com/JCSDA/spack-stack

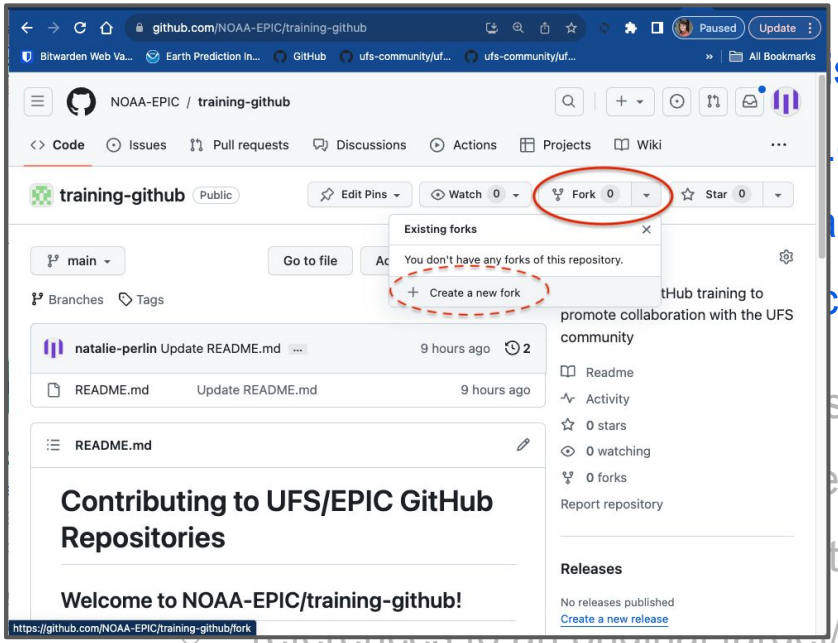
GitHub Repositories: Forks and Clones

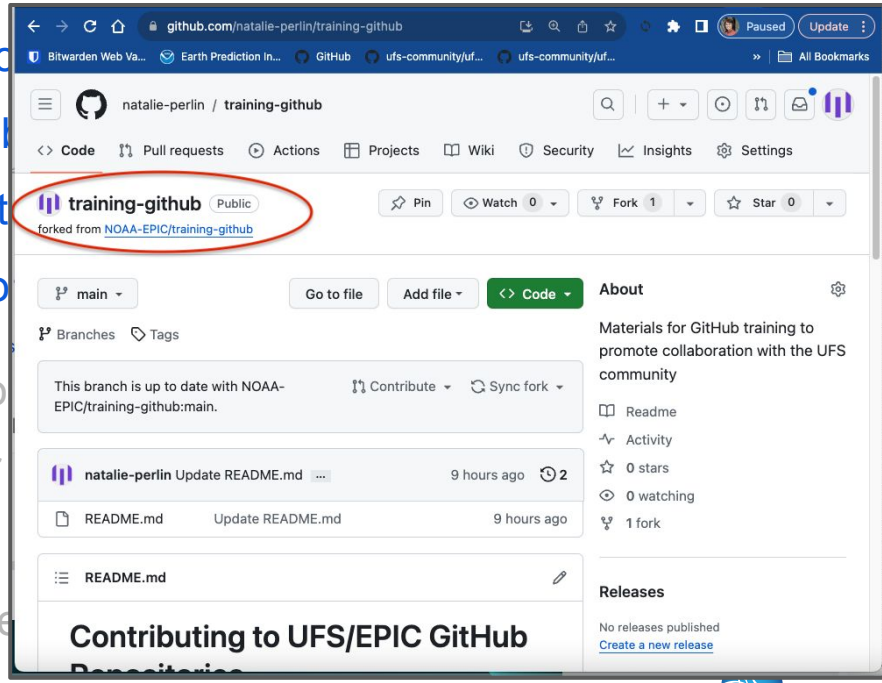
- **fork**: a separate copy of another existing repository
  - created in a remote location, e.g., GitHub account
  - could be synced with the primary repository
  - contribute to your fork not affecting the original repo



- **clone**: a linked local copy of an existing repository
  - a repository code is downloaded to your local machine
  - done through the command 'git clone'
  - references to an original target/remote repository

GitHub Repositories: Forks and Clones





GitHub Repositories: Forks and Clones

- **fork**: a separate copy of another existing repository
  - created in a remote location, e.g., GitHub account
  - could be synced with the primary repository
  - contribute to your fork not affecting the original repo

- **clone**: a linked local copy of an existing repository
  - a repository code is downloaded to your local machine
  - done through the command 'git clone'
  - references to an original target/remote repository

GitHub Repositories: Forks and Clones

- **fork**: a separate copy of another existing repository
  - crea
  - cou
  - con
- **clone**: a
  - a re
  - don
  - refe

```
[Natalie@MacbookPro:~]$ git clone https://github.com/NOAA-EPIC/training-github.git
Cloning into 'training-github'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 16 (delta 3), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), 9.95 MiB | 1.52 MiB/s, done.
Resolving deltas: 100% (3/3), done.
[Natalie@MacbookPro:~]$ cd training-github
[Natalie@MacbookPro:~/training-github]$ git remote -v
origin  https://github.com/NOAA-EPIC/training-github.git (fetch)
origin  https://github.com/NOAA-EPIC/training-github.git (push)
[Natalie@MacbookPro:~/training-github]$ git branch
* main
[Natalie@MacbookPro:~/training-github]$
```

Contributing to an open-source GitHub repository

1. **FORK** the original repository to your GitHub account
2. **CLONE** your repository to a local machine
3. Make changes to your cloned repository
4. **PUSH** them to your GitHub repository: it then will be synced to your changes
5. Create a **PULL REQUEST** (PR) to merge your changes with the original repository: it means making a request to the repository owner and telling them "Please check my changes and merge them if you like it"

Contributing to an open-source GitHub repository

Repo_A

some GitHub account

pull request

fork

Repo_B

your GitHub account

clone

Repo_B

push

your local system

MAKE CHANGES

GitHub Repositories: Branches and Tags

Branches:

- used for concurrent development, bux fixes, implementation *features*
- can be moved or deleted
- often expected to be merged back into original branch
- *feature branches* could have a long span

Tags:

- mark specific points in history to capture software versions
- typically used to mark stable points that correspond to specific releases
- *permanent* and don't change after they are created
- used to label a specific *commit*

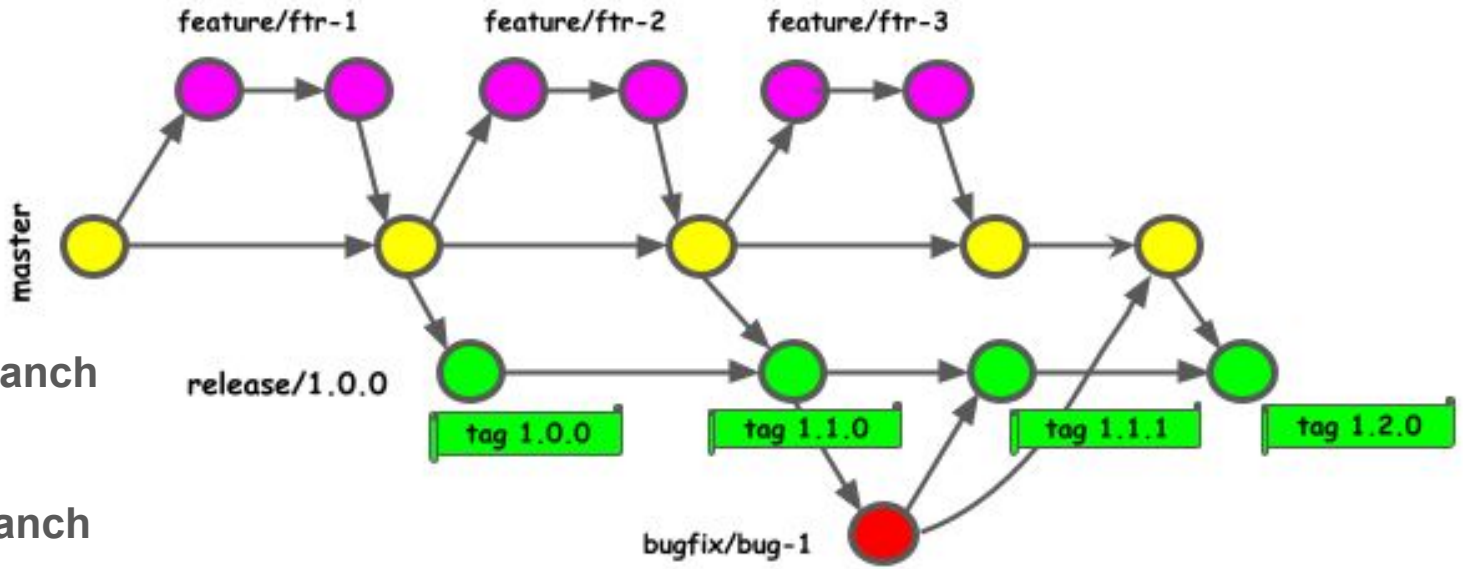
GitHub Repositories: Branches and Tags

feature branch

release branch

tag

bug fix branch

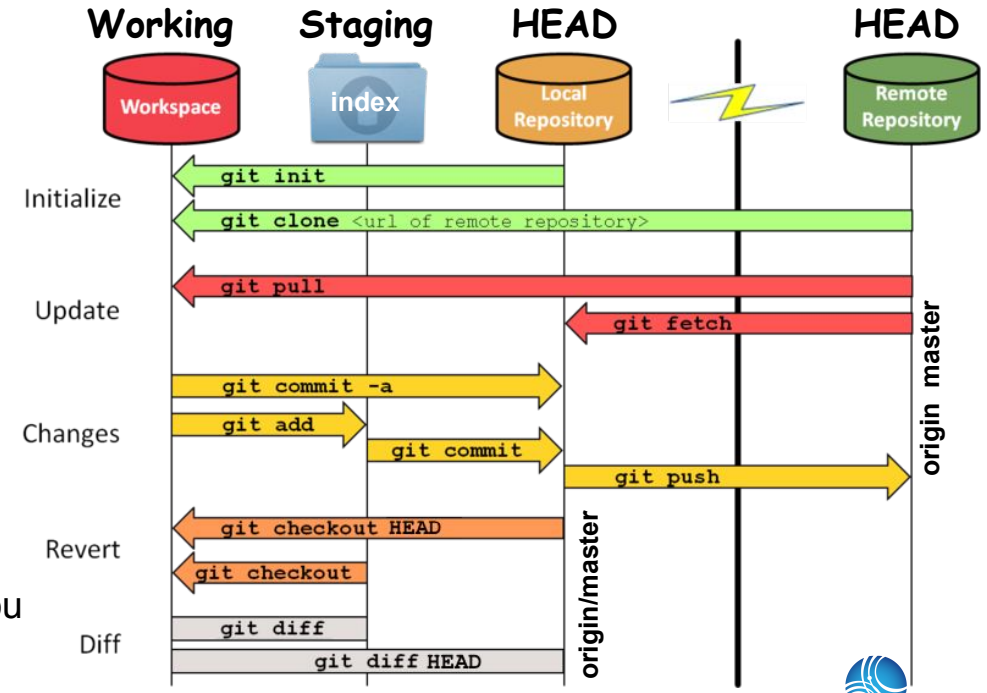


Git workflow: from local spaces to remote repos

Local Git has **three** stages:

- **Working area** -
  your local directory,
  all untracked, unstaged,
  locally deleted files
- **Staging area:**
  what needs to be committed,
  does not need to be committed,
  any new files to be added/removed
- **Local repository** (HEAD*):
  committed changes, ,
  latest checked out version of the
  remote repository

(*) HEAD - symbolic reference to the branch you are working on, or rather, most recent commit

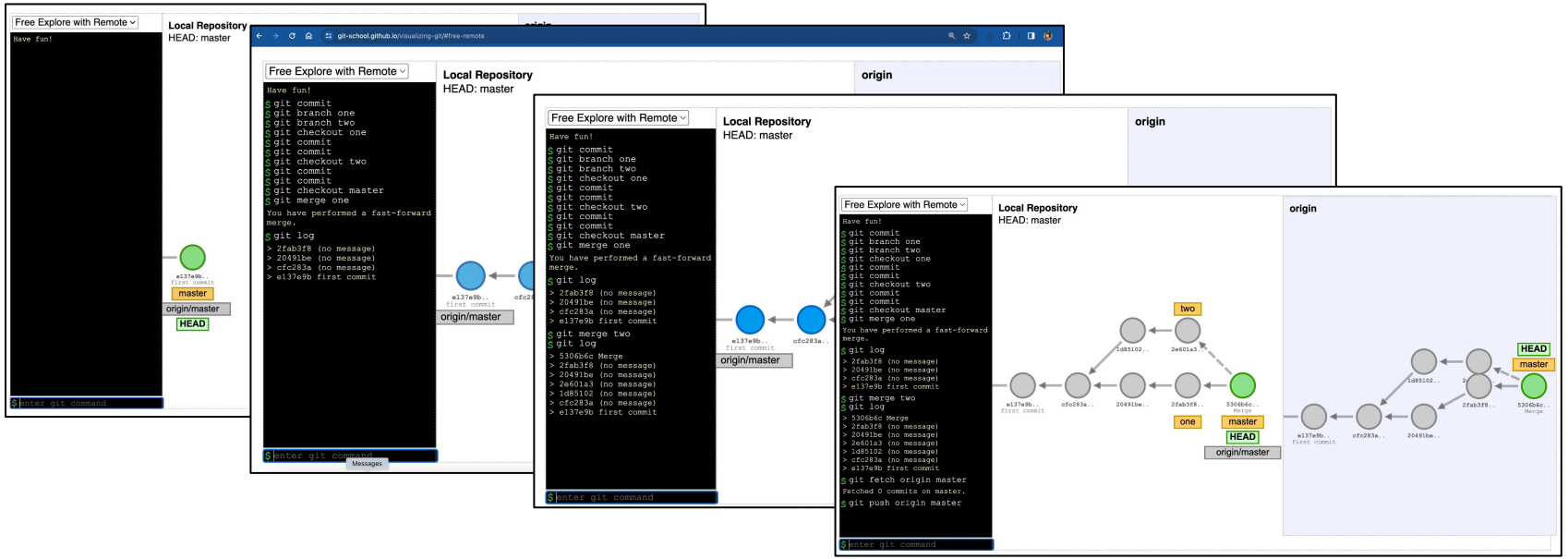


Git workflow: from local spaces to remote repos

- Use an online workflow visualizer tool: https://git-school.github.io/visualizing-git:
  - **case 1:** divergent local branches, merge, update the remote repo
  - **case 2:** divergent local branches, detached HEAD, rebase, merge, bring upstream branch changes, update the remote repo
- Use an existing authoritative repository to test real-case scenarios
  - fetch and merge remote branches, update a fork repository
  - introduce your changes, merge, resolve merge conflicts
  - make a pull request (PR) into the authoritative repository
- Test out a PR, give feedback, make a PR into another developer's code

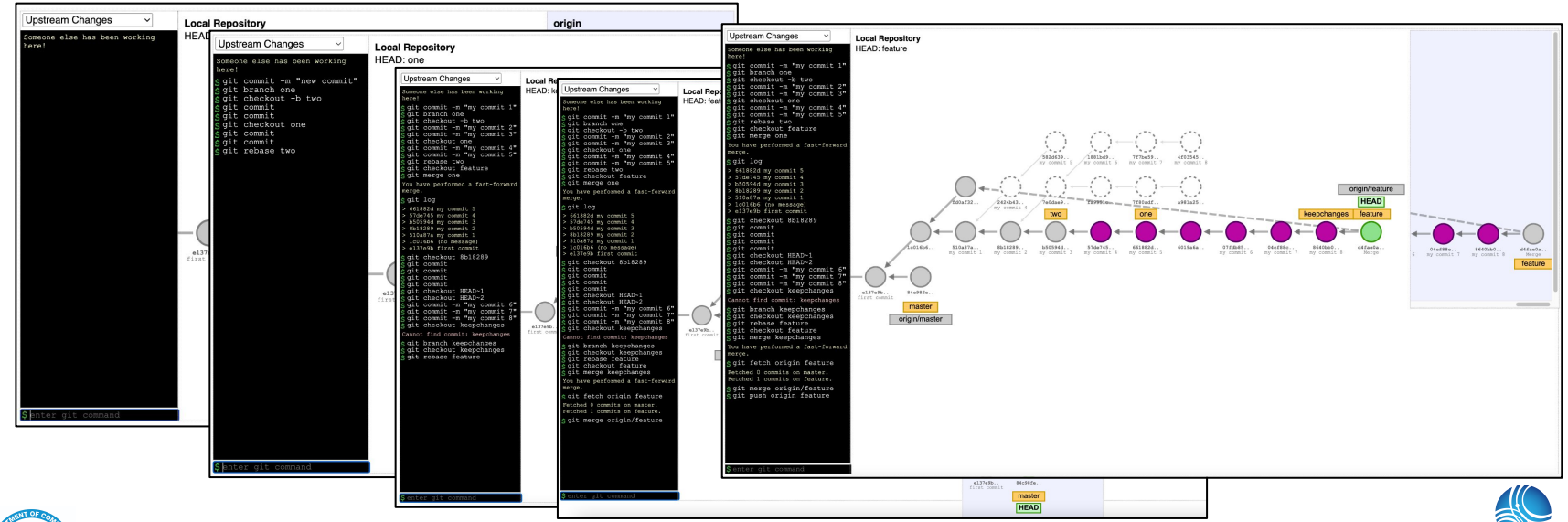
Visualization cases: case 1 (https://git-school.github.io/visualizing-git)

case 1: divergent local branches, merge, update the remote repo



Visualization cases: case 2 (https://git-school.github.io/visualizing-git)

case 2: divergent local branches, rebase, detached HEAD, merge, bring upstream branch changes, update the remote repo



General steps to contribute to remote repos

1. Create a fork of the authoritative repo in your GitHub account (dashboard)
2. Clone the repository into your local machine

   `git clone <url-of-the-repo-fork>`
3. Add an upstream authoritative repository for the reference and updates

   `git remote add upstream  <url-of-the-authoritative-repo>`
4. Checkout your repository's main branch: `git checkout master`
5. Fetch any changes from the authoritative repository:

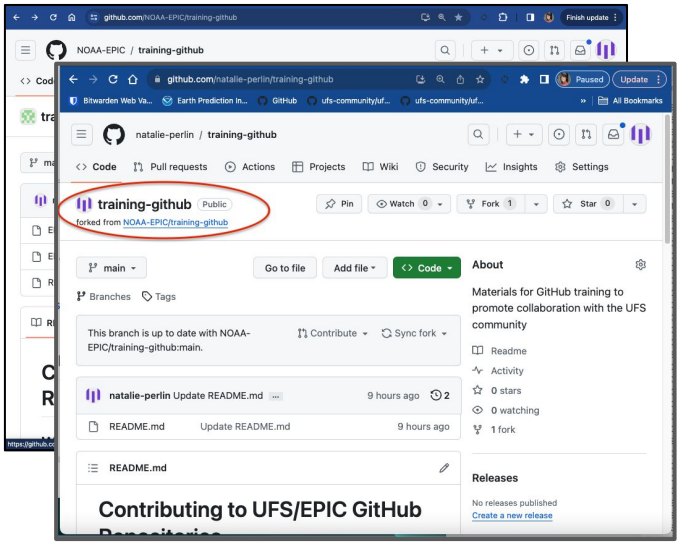
   `git fetch upstream`
6. Merge in the changes to the master branch into your working reference

   `git merge upstream/master`
7. Push your changes back to GitHub: `git push origin master`
8. Create a pull request (PR) into the authoritative repository (dashboard)

Git workflow example with an existing authoritative repo

An existing repository: https://github.com/NOAA-EPIC/training-github

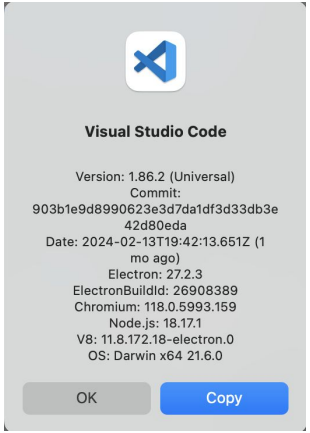
- fetch and merge remote branches, update a fork repository

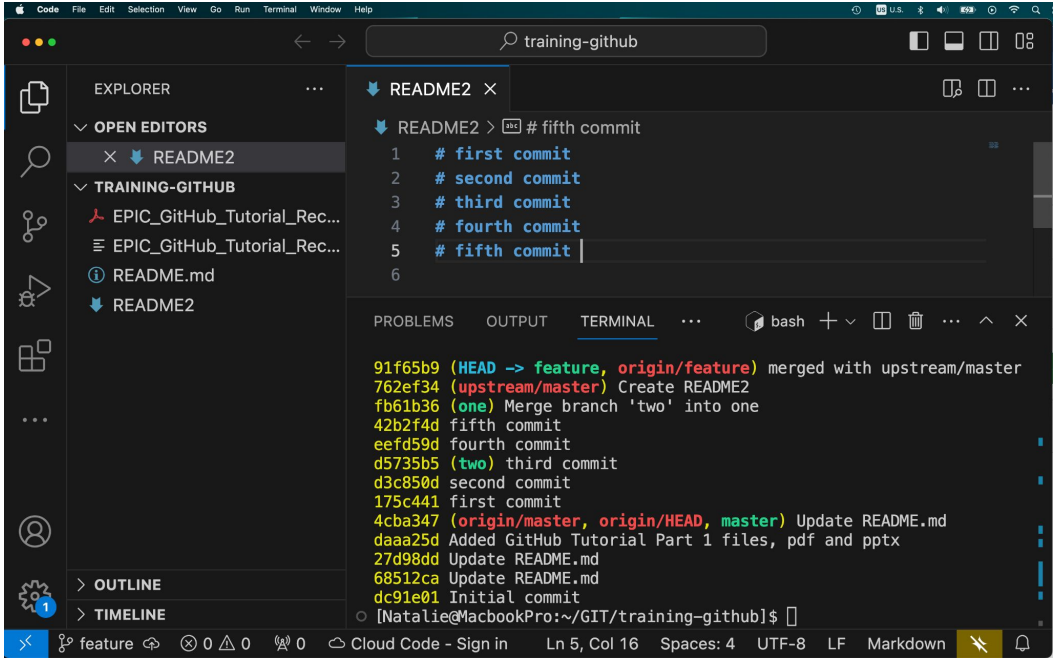



```
git clone git@github.com:<repo-fork>
cd <training-github>
git remote add upstream <authoritative-repo>
git remote -v
git checkout master
# make changes, commit them to a master branch
# fetch any updates from the authoritative repository:
git fetch upstream
git merge upstream/master
git log --oneline
git push origin master
```

Git workflow example with an existing authoritative repo

We could use a GUI code editor, such as Visual Studio Code





Git workflow example 2 with an existing authoritative repo

- An existing authoritative repository: https://github.com/NOAA-EPIC/training-github
  - introduce your changes, merge, resolve merge conflicts

```
git clone git@github.com:<repo-fork>
git remote add upstream <authoritative-repo>
git remote -v
git checkout master
git checkout -b feature
# create a file README2, line: # first commit
git add README2
git commit -m "first commit"
git branch one
git checkout -b two
```

Git workflow example 2 (continued)

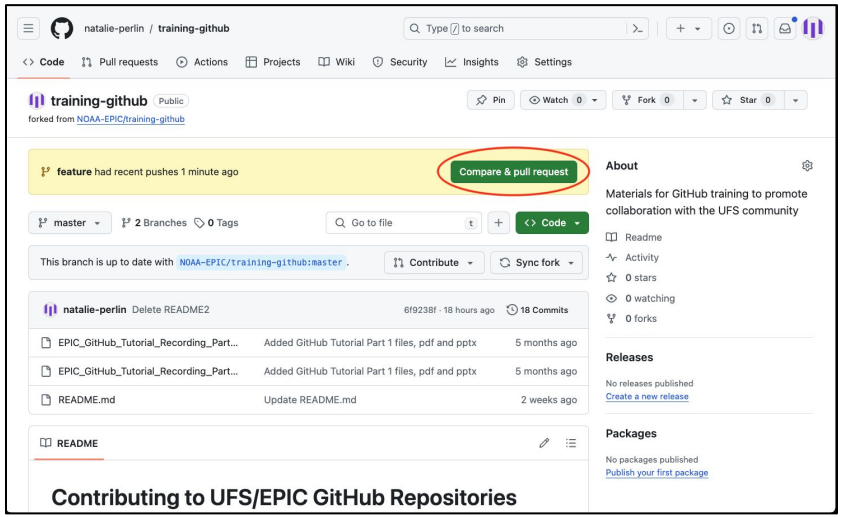
- An existing authoritative repository: https://github.com/NOAA-EPIC/training-github
  - introduce your changes, merge, resolve merge conflicts

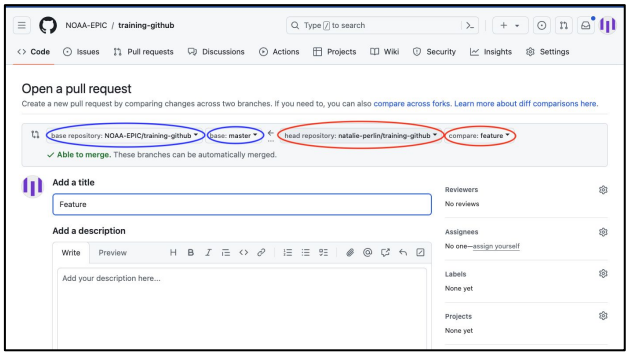
```
git branch
# Add a line to README2 file: # second commit
git commit -am "second commit"
# Add a line to README2 file: # third commit
git commit -am "third commit"
git checkout one
# Add a line to README2 file: # fourth commit
git commit -am "fourth commit"
# Add a line to README2 file: # fifth commit
git commit -am "fifth commit"
git branch
git merge two
```

```
# solve merge conflicts using a text editor!!
git commit -am "merge with two"
git branch
git checkout feature
git merge one
git log --oneline
git fetch upstream master
git merge upstream/master
git checkout upstream/master
git commit -am "merge with upstream"
git push origin feature
```

Git workflow example: submit a pull request

- An existing authoritative repository: https://github.com/NOAA-EPIC/training-github
  - make a pull request (PR) into the authoritative repository (*)

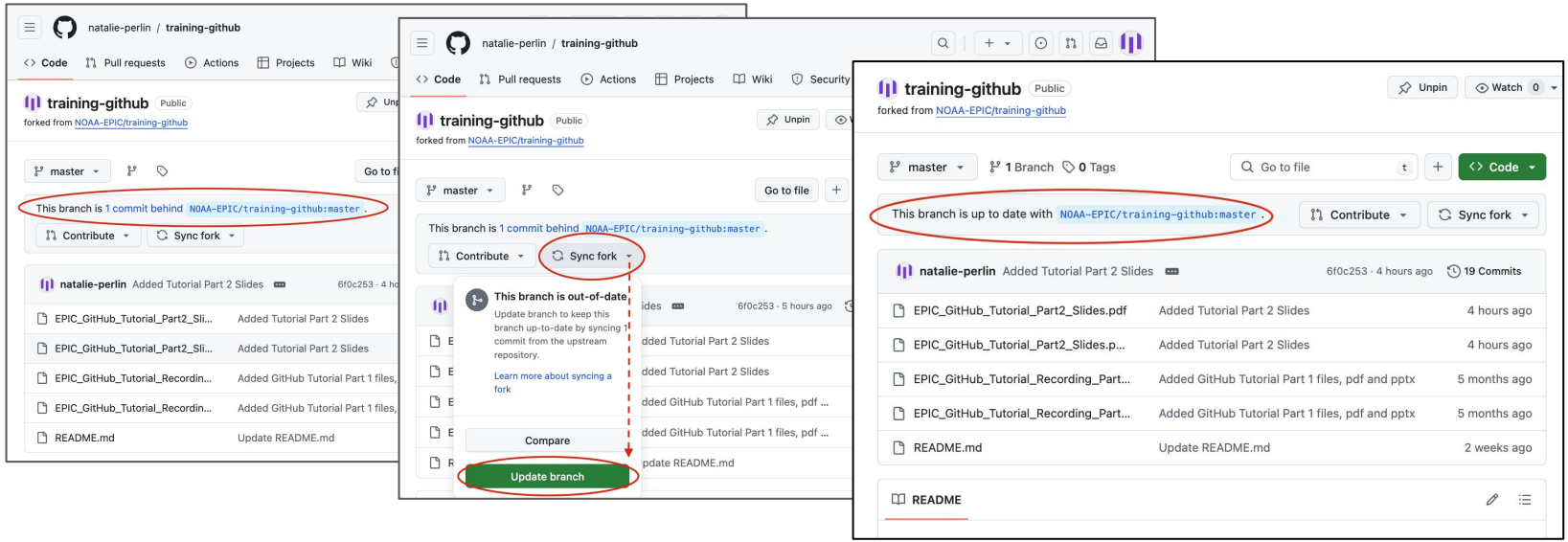




(*) - you may need to address any comments, resolve conversations, complete requested changes by PR Reviewers!

Local repository: keep it synched with the remote!

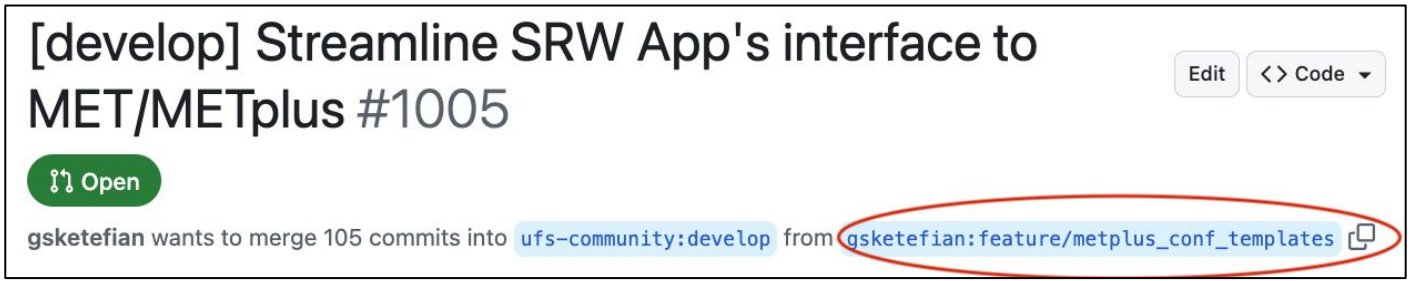
- An existing authoritative repository: https://github.com/NOAA-EPIC/training-github
- Remember to sync your primary (master) branch with that from the original repo!



Git workflow example: testing another user's pull request

- Test out a PR, give feedback, make a PR into another developer's code

```
git clone git@github.com:ufs-community/ufs-srweather-app.git
cd ufs-srweather-app
git pull origin pull/<PR#>/head:PRtest
git checkout PRtest
# test the PR, build, run the code, comment on the PR on specific
# issues, files, or lines
# Make your own PR into original PR author's repository:
# get author's repository location and branch from the PR's page, i.e:
# https://github.com/ufs-community/ufs-srweather-app/pull/1005
```



Perusing Public Repositories

- Check out a few different repositories with submodules:

```
git clone --recursive <url-repo>

git clone --recurse-submodules <url-repo>

git clone <url-repo>
cd <repo>
git submodule update –remote
```

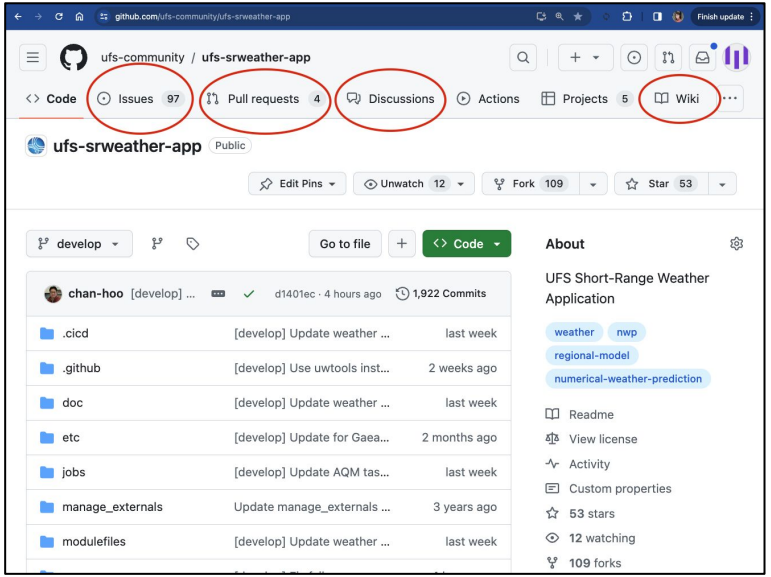
git clone --recurse-submodules https://github.com/jcsda/spack-stack.git

git clone --recursive https://github.com/ufs-community/ufs-weather-model.git

Perusing Public Repositories

- Submitting issues - “**Issues**”
- Reaching out for help through discussions - “**Discussions**”
- Checking out and testing existing PRs - “**Pull Request**s”
- Navigating documentation - “**Wiki**”, look for links to **readthedocs.io**, such as

  https://ufs-weather-model.readthedocs.io/
  https://ufs-srweather-app.readthedocs.io/



Perusing Public Repositories