CCPP Code Management and Testing

Dustin Swales

NOAA GLOBAL SYSTEMS LABORATORY & DEVELOPMENTAL TESTBED CENTER

August 15, 2023 – CCPP Visioning Workshop

Developmental Testbed Center

GitHub Repositories

There are ***three*** authoritative CCPP repositories (public):

- https://github.com/NCAR/ccpp-scm
- https://github.com/NCAR/ccpp-physics
- https://github.com/NCAR/ccpp-framework

All use **main** as the default branch.

The **main** branch of *ccpp-scm* points to the **main** branches in both *ccpp-physics* and *ccpp-framework*

Code Management (forks)

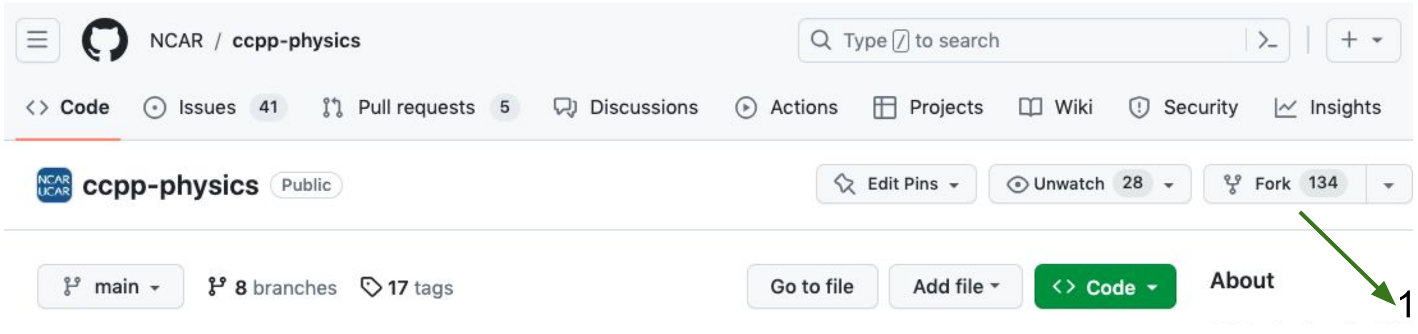
The CCPP development practices make use of the GitHub forking workflow. (See this website for some background information.)

The following steps describe how to create a fork for CCPP development.

1. Go to the repository you wish to fork, and make sure you are signed in to your GitHub account.
    - For CCPP-SCM changes, this should be the authoritative repository (https://github.com/NCAR/ccpp-scm)
    - For CCPP-Framework changes, this should be the authoritative repository (https://github.com/NCAR/ccpp-framework)
    - For CCPP-Physics changes, this could be the authoritative repository **OR** the Application Fork corresponding to your host model of interest
        - UFS Fork* (https://github.com/ufs-community/ccpp-physics)
        - CCPP-SCM (https://github.com/NCAR/ccpp-physics)
2. Select the "fork" button in the upper right corner.

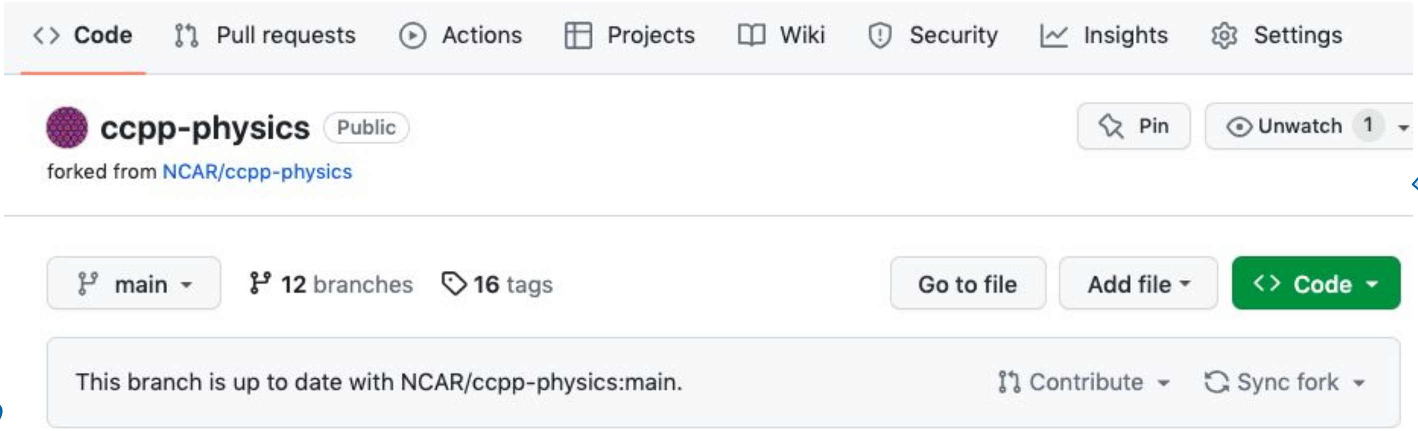
*More on the UFS fork of the ccpp-physics later…

Code Management (forks)



Authoritative

134 forks



forked from NCAR/ccpp-physics

This branch is up to date with NCAR/ccpp-physics:main.

Personal Fork

Code Management (branches)

Code development occurs on **branches** in your personal **forked** repository.

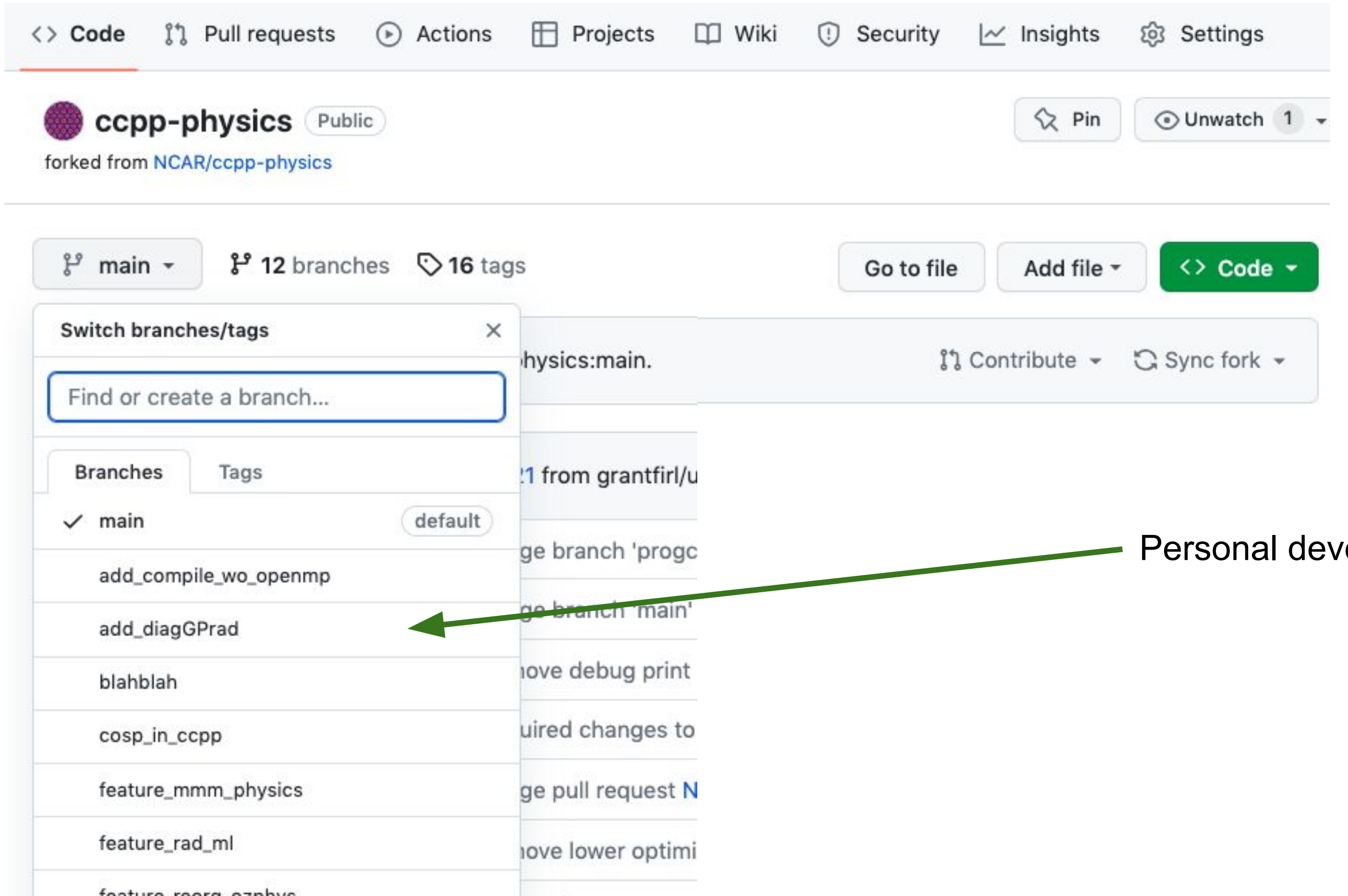
When innovations are ready, a **pull-request** is created from your branch/fork into the authoritative repositories to incorporate these changes.

For example, using the CCPP-SCM, say I want to change some source code to fix a bug in a physics parameterization:

```
> git clone --recursive https://github.com/NCAR/ccpp-scm.git                          - Clone ccpp-scm repository, and dependencies
> cd ccpp-scm/ccpp/physics                                                            - Go to physics directory
> git remote add dustinswales https://github.com/dustinswales/ccpp-physics.git        - Add personal forked repository as remote
> git checkout -b bugfix_for_somefile                                                 - Create branch in ccpp-physics.
> vi physics/somefile.F90                                                             - Modify source file
> git add physics/somefile.F90                                                        - Add changes to be committed.
> git commit -m "Address bug found in physics/somefile.F90"                           - Commit changes
> git push dustinswales bugfix_for_somefile                                           - Push changes to forked ccpp-physics repository.
```

In this simple example I made some changes to a source file, created a new branch on my forked ccpp-physics repository, and pushed these changes to GitHub.

Code Management (branches)



Code Pull requests Actions Projects Wiki Security Insights Settings

ccpp-physics Public

forked from NCAR/ccpp-physics

Pin Unwatch 1

main 12 branches 16 tags Go to file Add file Code

Switch branches/tags

Find or create a branch...

Branches Tags

- ✓ main default
- add_compile_wo_openmp
- add_diagGPrad
- blahblah
- cosp_in_ccpp
- feature_mmm_physics
- feature_rad_ml

hysics:main. Contribute Sync fork

1 from grantfirl/u

ge branch 'progc

ge branch 'main'

nove debug print

uired changes to

ge pull request N

nove lower optimi

Personal development branches

Code Management (pull-requests)

Once changes are ready to be included in the authoritative repositories, GitHub **pull requests** (PRs) are opened into the impacted repositories.

To create a PR, go to the github.com web interface, and navigate to your repository fork and branch. In most cases, this will be in the ccpp-physics repository, hence the following example:

- Navigate to: https://github.com/<yourusername>/ccpp-physics
- Use the drop-down menu on the left-side to select a branch to view your development branch
- Use the button just right of the branch menu, to start a "New Pull Request"
- Fill in a short title (one line)
- Fill in a detailed description, including reporting on any testing you did
- Click on "Create pull request"

If your development also requires changes in other repositories, you must open PRs in those repositories as well.

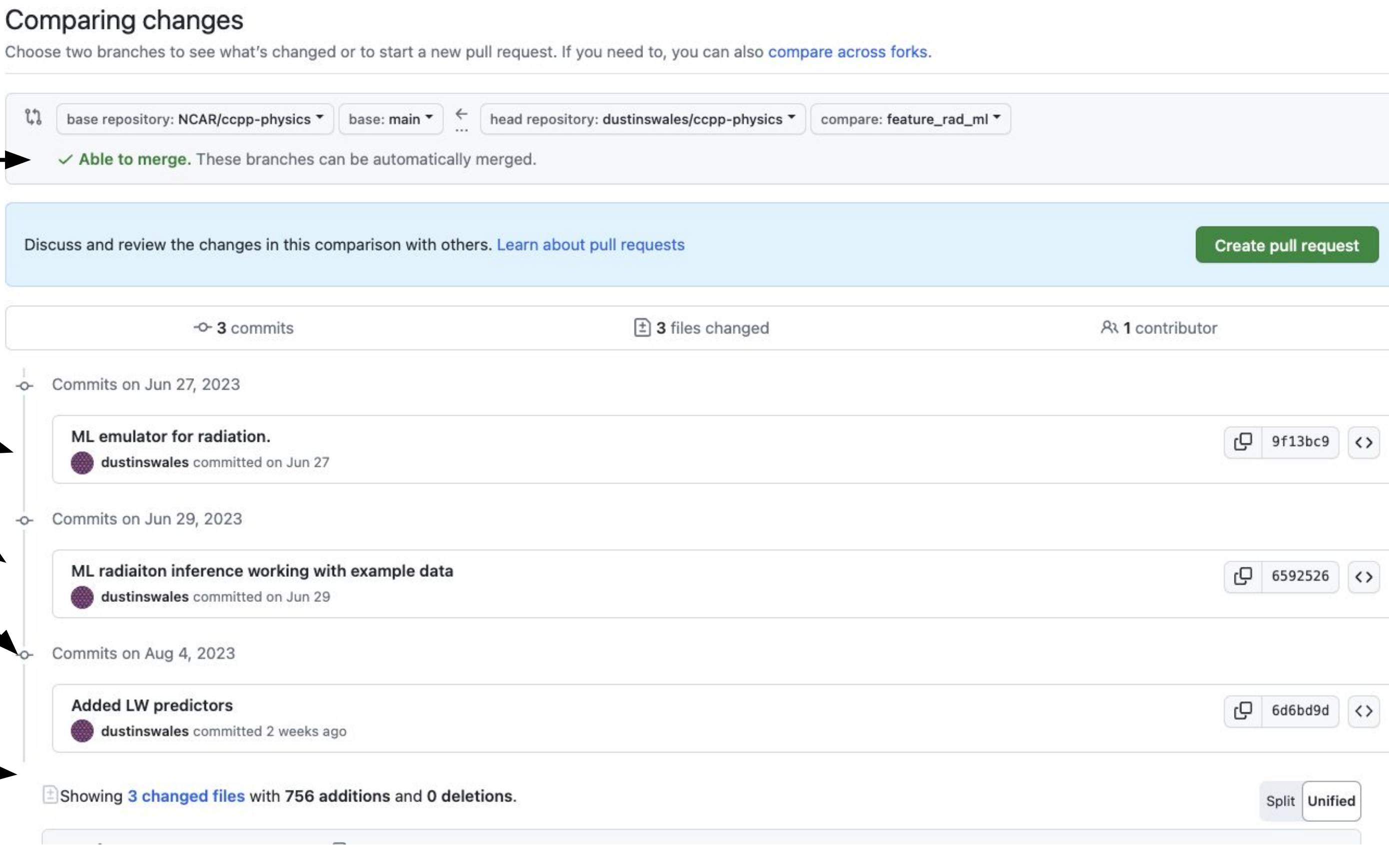
Several people (aka CODEOWNERS) are automatically added to the list of reviewers on the right hand side.

Code Management (pull-requests)

Is personal fork up-to-date with merge target?

List of commits included in pull-request

Differences (highlighted) in files.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: NCAR/ccpp-physics | base: main ← head repository: dustinswales/ccpp-physics | compare: feature_rad_ml

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. Learn about pull requests

3 commits | 3 files changed | 1 contributor

Commits on Jun 27, 2023

ML emulator for radiation.
dustinswales committed on Jun 27 — 9f13bc9

Commits on Jun 29, 2023

ML radiaiton inference working with example data
dustinswales committed on Jun 29 — 6592526

Commits on Aug 4, 2023

Added LW predictors
dustinswales committed 2 weeks ago — 6d6bd9d

Showing 3 changed files with 756 additions and 0 deletions.

Code Management (CI)

Continuous-Integration and Continuous-Deployment (CI/CD) are used throughout the authoritative repositories to streamline pull-requests.

Each time a PR is opened, a set of tests are automatically run to ensure the code changes are working as expected.

For the ccpp-physics, we have CI tests to inform us when corresponding host model changes are needed. For example, in the case when a new interstitial variable is added to ccpp-physics, we have a CI test will alert us that host-model changes are also needed.

For the ccpp-scm we have a wide variety of CI tests that are triggered when a PR is created:

- *Build SCM w/ GNU fortran 10/11/12 and python 3.7/3.9 for DEBUG/RELEASE modes*	(12 tests)
- *Build SCM and run regression tests for DEBUG/RELEASE modes*	(2 tests)
- *Test CCPP prebuild step for supported hosts*	(1 test)
- *Build and Run SCM using cases from DEPHY repository*	(1 test)
- *Test SCM with UFS-replay scripts using staged UFS output*	(1 test)
	17 tests

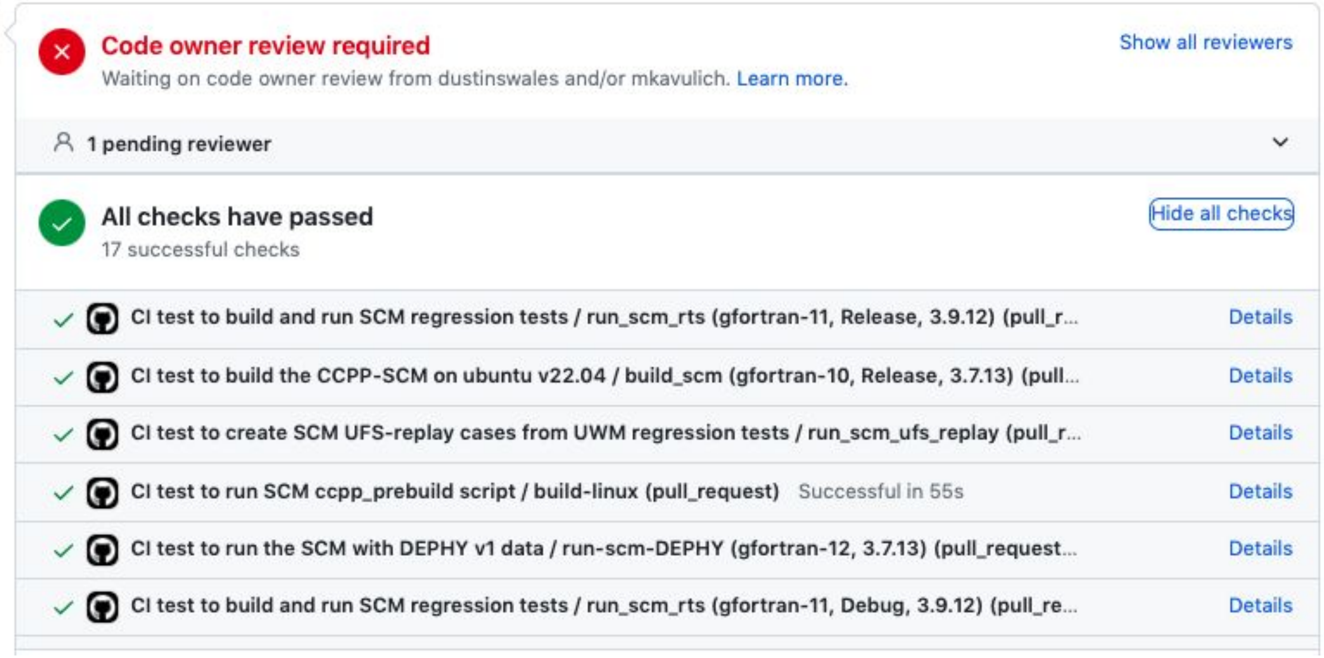
At the moment we don’t have any CI testing in the ccpp-framework repository, but this will change in the near future.

Code Management (CI)

Testing triggered when pull request opened.

Upon successful completion of the tests.

Individual test results.



Code owner review required Show all reviewers
Waiting on code owner review from dustinswales and/or mkavulich. Learn more.

1 pending reviewer

All checks have passed Hide all checks
17 successful checks

✓ CI test to build and run SCM regression tests / run_scm_rts (gfortran-11, Release, 3.9.12) (pull_r... Details
✓ CI test to build the CCPP-SCM on ubuntu v22.04 / build_scm (gfortran-10, Release, 3.7.13) (pull... Details
✓ CI test to create SCM UFS-replay cases from UWM regression tests / run_scm_ufs_replay (pull_r... Details
✓ CI test to run SCM ccpp_prebuild script / build-linux (pull_request) Successful in 55s Details
✓ CI test to run the SCM with DEPHY v1 data / run-scm-DEPHY (gfortran-12, 3.7.13) (pull_request... Details
✓ CI test to build and run SCM regression tests / run_scm_rts (gfortran-11, Debug, 3.9.12) (pull_re... Details

Screenshot from pull-request into ccpp-scm.

Code Management (UFS fork)

The CCPP-SCM uses the authoritative ccpp-physics repository, whereas the UFS Weather Model (UWM) uses a forked copy of the ccpp-physics repository, https://github.com/ufs-community/ccpp-physics.

Contributions to the ccpp-physics can come from many sources:

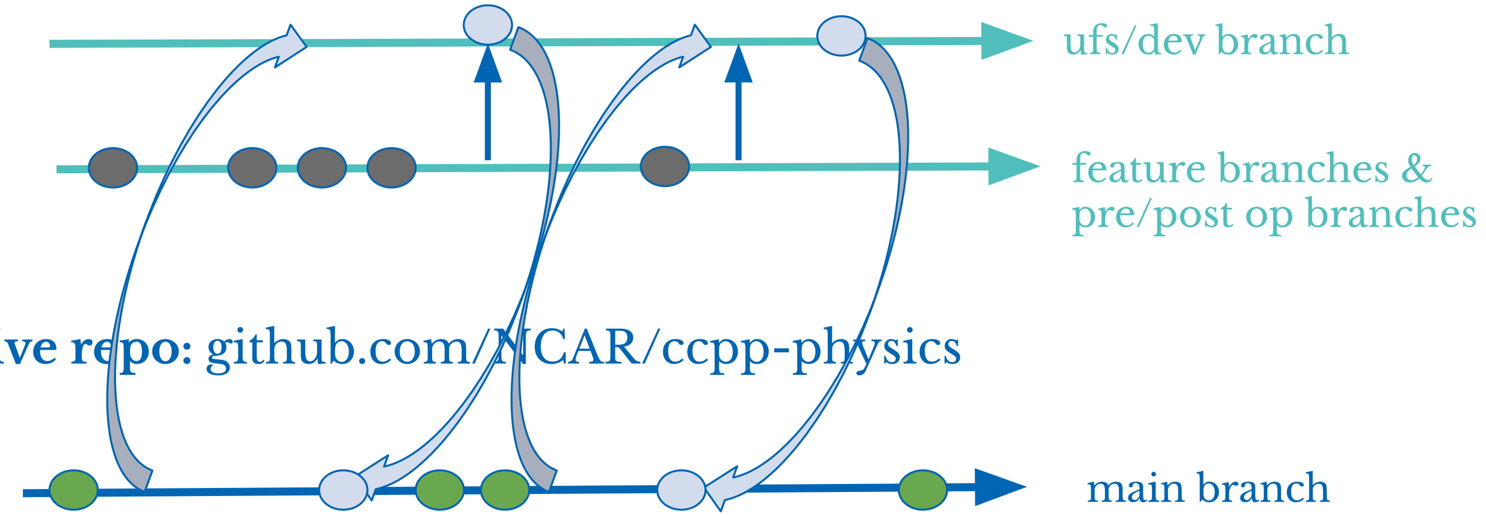
- If the development is targeted for UFS applications, then it is suggested that PRs should be opened into the ufs-community ccpp-physics fork.
- For physics development coming from outside the UFS, the authoritative NCAR repository should be used.

ccpp-physics code managers are responsible for keeping the authoritative NCAR ccpp-physics repository up-to-date with development occurring in the UFS fork. PRs are created into NCAR:ccpp-physics each time something is merged into ufs-community:ccpp-physics.

Going the other way, if contributions are added to NCAR:ccpp-physics, and are of interest to the UWM, the code managers will open up PRs into ufs-community:ccpp-physics.

Code Management (UFS fork)

UFS fork: github.com/ufs-community/ccpp-physics



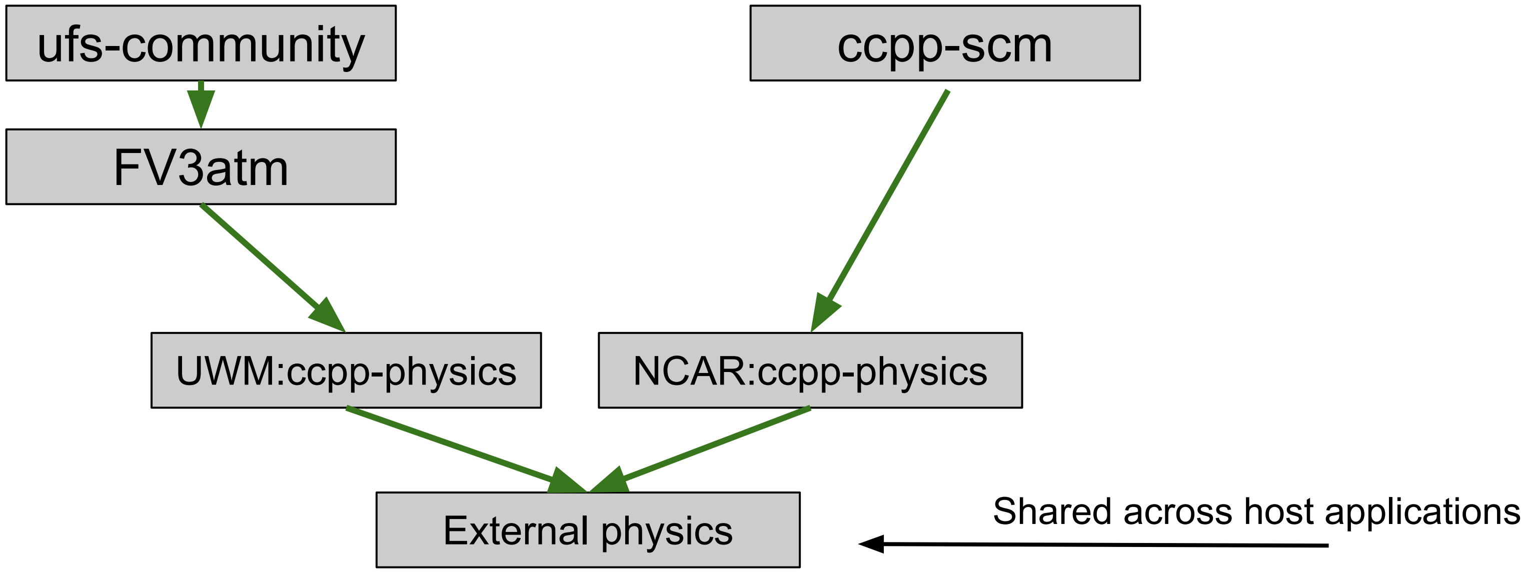
Authoritative repo: github.com/NCAR/ccpp-physics

Responsibility: EMC+DTC, DTC

Submodules in ccpp-physics

Much like other NWP/GSM applications, it’s possible for schemes to have external dependencies in the ccpp-physics. These dependencies are imported into the ccpp-physics using “submodules”

In some cases the entire parametrization could be external to the ccpp, and only an interface to the scheme, or driver, exists in the ccpp-physics repository (e.g. rte-rrtmgp).



Submodules in ccpp-physics

Using submodules can be useful from a code management perspective:

- There are clear lines of responsibility (e.g. this belongs to the host and this is part of the scheme).
- Distributed development. Parameterization and ccpp development are decoupled.
- Scheme advancements are added as needed.

Adding a submodule to the ccpp-physics repository is straightforward, see https://git-scm.com/book/en/v2/Git-Tools-Submodules

Testing

Responsibility: Model team, CCPP team, developer

	Code	Who	Test
MODEL	Model X	Model X team	RTs for Model X
CCPP	ccpp-physics main	CCPP team	RTs for all models
	ccpp-physics model X fork/branch	Model X team*	RTs for Model X
SCHEME	scheme main	Developer	Simple tests
	scheme model X fork/branch	Model X team	RTs for Model X

Simple tests: Unit tests that verify software integrity using CCPP single-column model: answers do not change unless supposed to, portability (works with needed compilers), threading, etc.

Model Regression Tests (RTs): Integration tests using relevant configurations and platforms

*As a special case, DTC will co-manage the UFS fork/branch

Testing

In general, physics contributions **need to be tested** by their respective host-model before opening a pull-request into the authoritative repository.

For the ccpp-scm, since it is so lightweight, this can all be handled via CI (described above).

The UWM requires access to HPC for its testing, and runs many tests, making automation more challenging. A hundred or so(?) simulations are run using several compilers across several supported platforms.

For changes that come into the authoritative ccpp-physics repository from the UWM fork, the DTC also runs the UWM regression tests with the authoritative code base. **The DTC receives support to run these tests*.

Code Releases

CCPP releases include up-to-date documentation for the ***ccpp-scm*** and ***ccpp-physics***.

These releases occur about ~1/year (Last one was June 2022).

The ***ccpp-physics*** and ***ccpp-scm*** are updated often throughout the year, but not the documentation.

If there is a UFS application release, we will tag the code base for posterity, but the DTC does not (always) update documentation for these releases.

Tagging

Tags are used to reference a stable/static code base on github.

The authoritative ccpp-physics and ccpp-scm repositories apply tags to the following events:

- CCPP releases (e.g. v6.0.0)
- UFS application releases (e.g. GFSv17.HR1)
  - When a UFS application has an official release and applies a tag, the code managers for the CCPP will propagate this from ufs-community:ccpp-physics to NCAR:ccpp-physics, and NCAR:ccpp-scm.

CCPP-SCM tags: https://github.com/NCAR/ccpp-scm/tags

CCPP-Physics tags:

- NCAR	https://github.com/NCAR/ccpp-physics/tags
- ufs-community	https://github.com/ufs-community/ccpp-physics/tags

There are no obstacles, or objections, to adding more tags to the codebase.

Scheme Versioning

Currently the individual parameterizations are **NOT** versioned within the CCPP.

However, there's nothing preventing scheme developers to use scheme versioning.

One possibility is to use ***semantic versioning***, where given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes
2. MINOR version when you add functionality in a backward compatible manner
3. PATCH version when you make backward compatible bug fixes