A Crash Course in SU2 Development

Dr. Thomas D. Economon

Joint NIA & SU2 Foundation Workshop
Hampton, Virginia
August 9, 2019

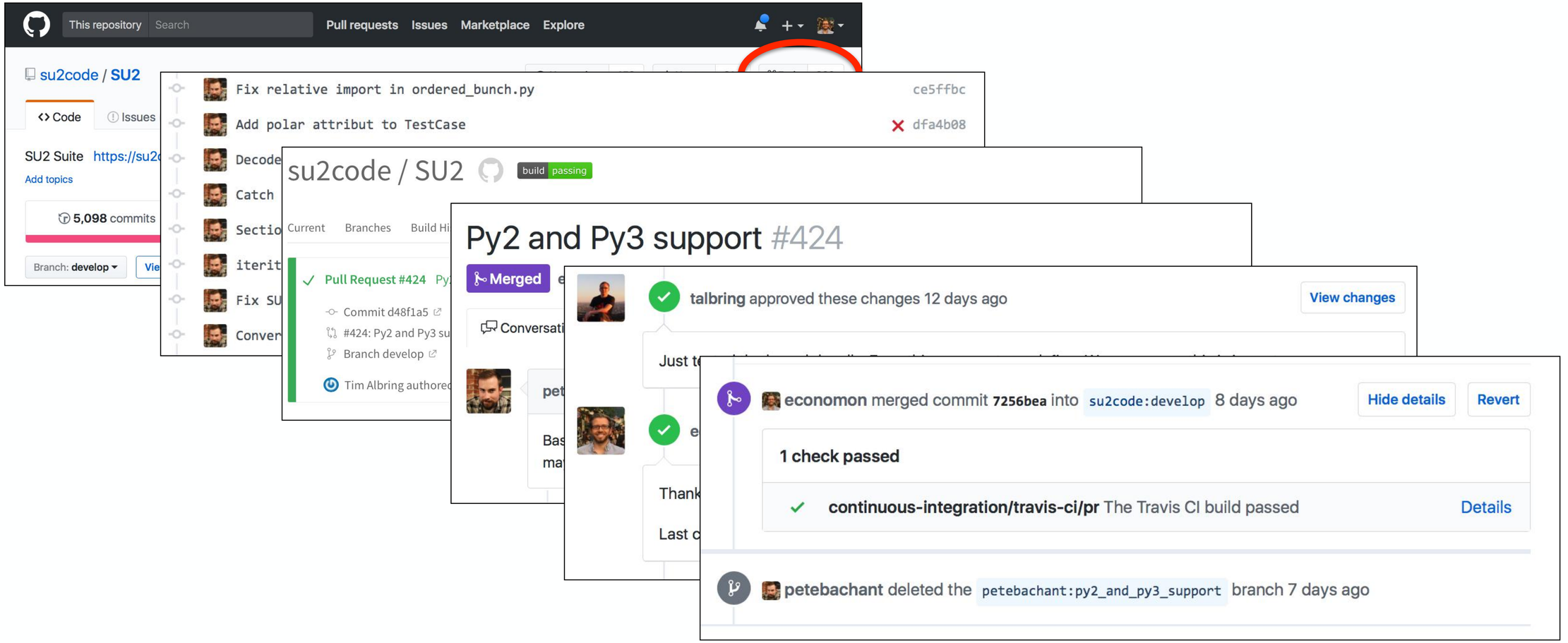
So, you want to be an SU2 developer?

Good news: it's easy.

We leverage standard development processes and the latest tools for open-source projects.

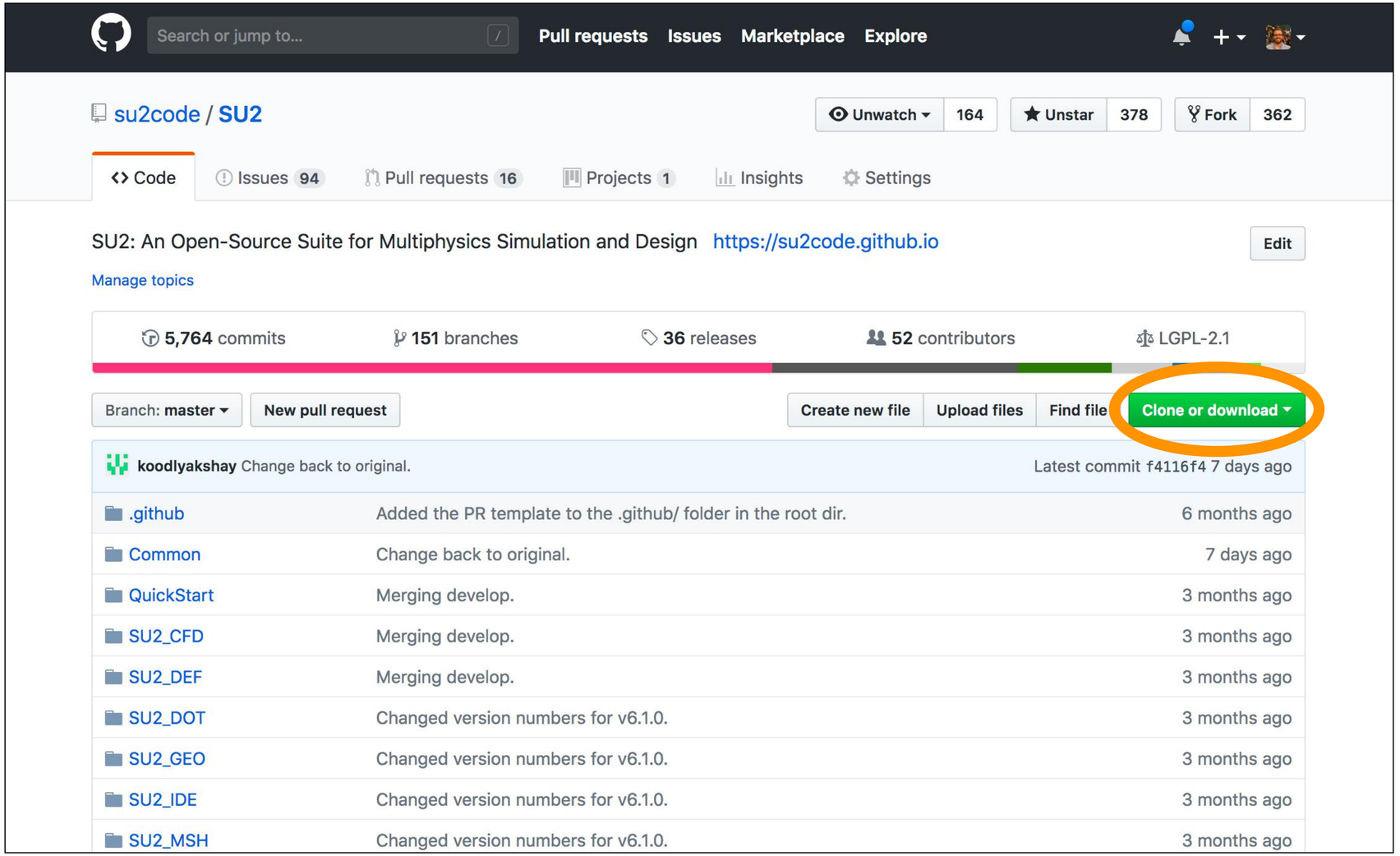
You will be ready to hack at the end of this talk.

Git/Branching Development Regressions Pull Request Release

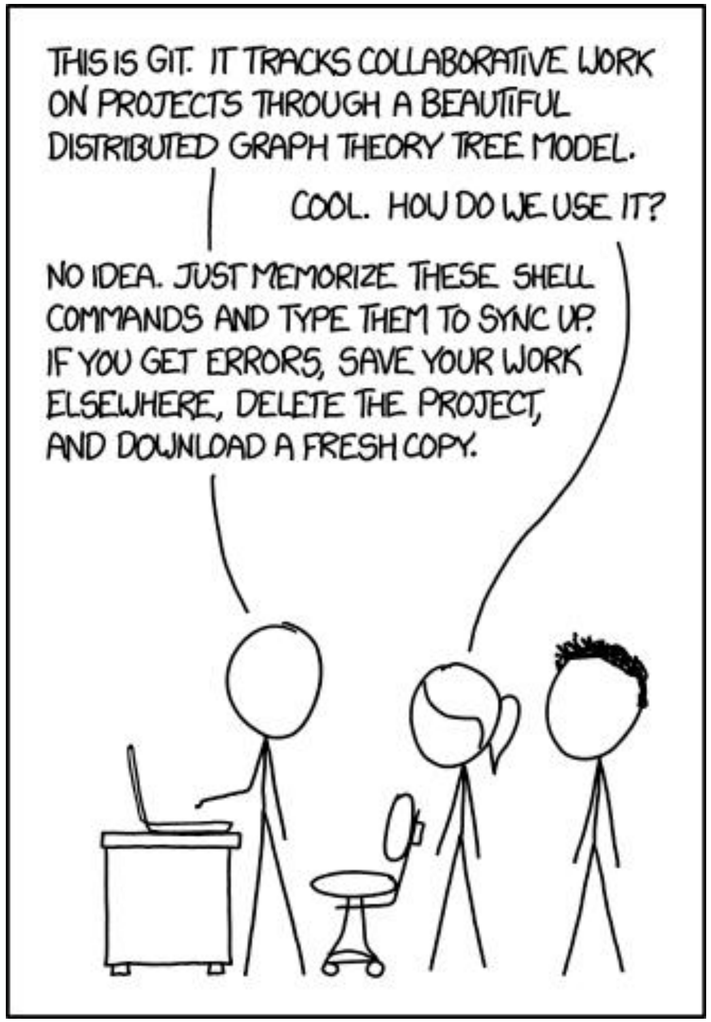


Anyone can be an SU2 developer.

Git/Branching — Development — Regressions — Pull Request — Release



Your starting point: https://github.com/su2code/SU2.



http://xkcd.com/1597/

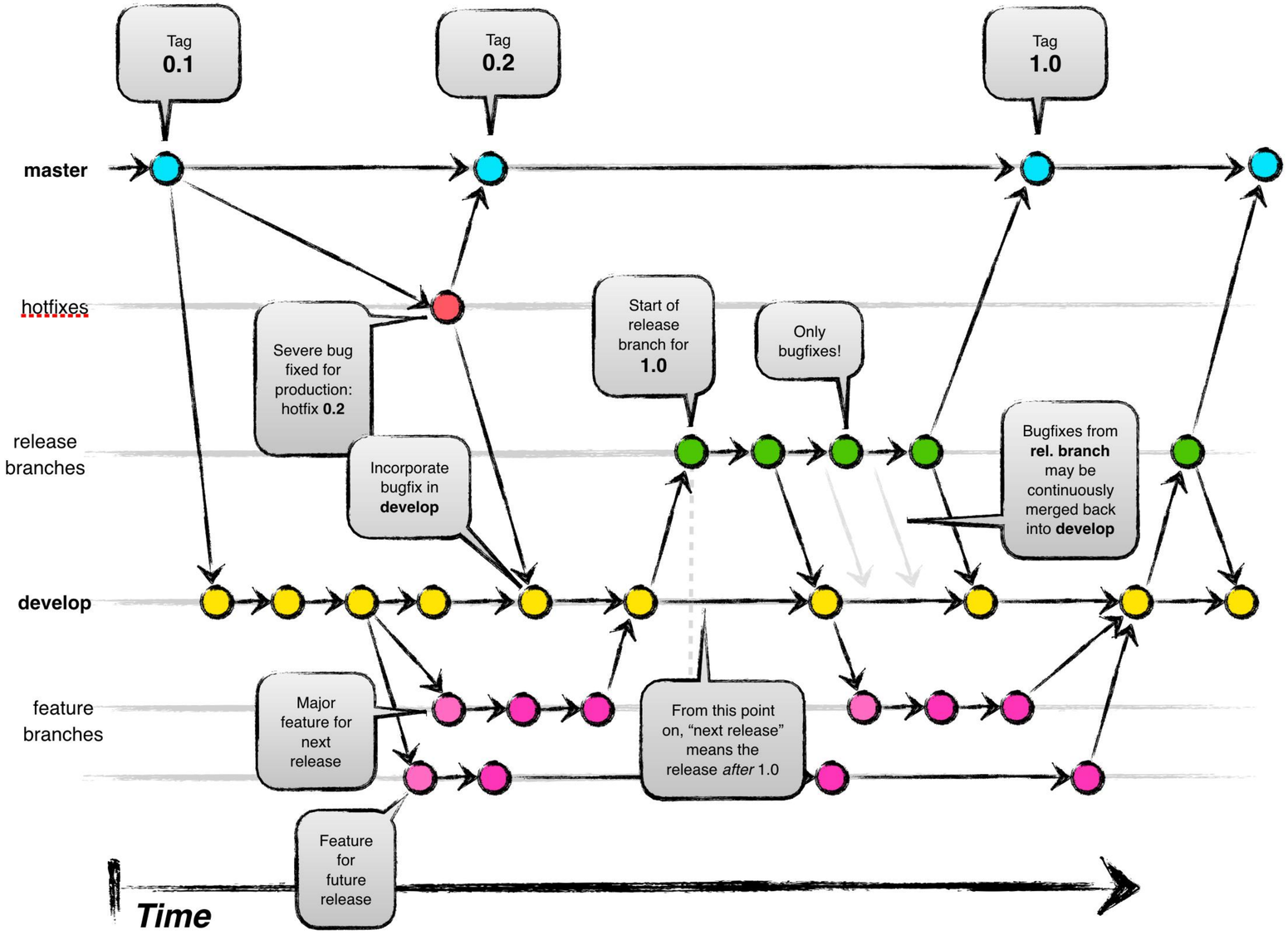
Here's that list of shell commands you should memorize:

- $ git clone https://github.com/su2code/SU2.git
- $ git branch
- $ git checkout -b feature_awesome origin/feature_awesome
- $ git status
- $ git diff
- $ git commit -am "This is an awesome commit."
- $ git push origin feature_awesome
- $ git checkout develop
- $ git pull origin develop
- $ git merge develop

And their translations:

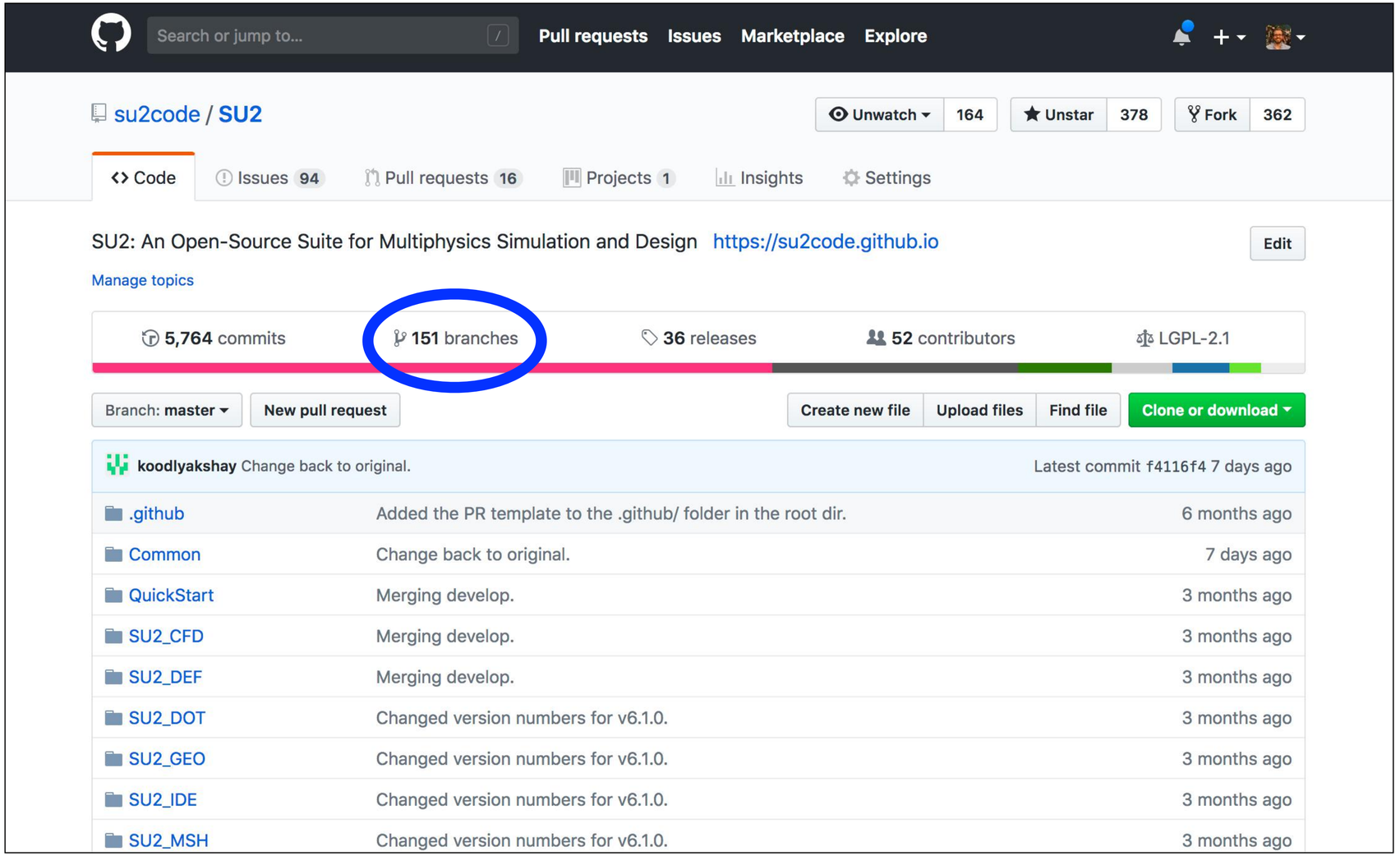
- Get a fresh copy of the entire repo (master branch to start)
- Check which branches I have locally
- Check out my feature branch that is already on the remote
- Check which files have changed since last commit
- Detailed diff of code changes since last commit
- While working, make commits frequently with messages
- Regularly push to the remote on GitHub
- Switch to the develop branch (assuming you have it locally)
- Merge the changes in the remote develop into local develop
- Merge the changes from local develop into current local branch

We use the popular Gitflow branching model.



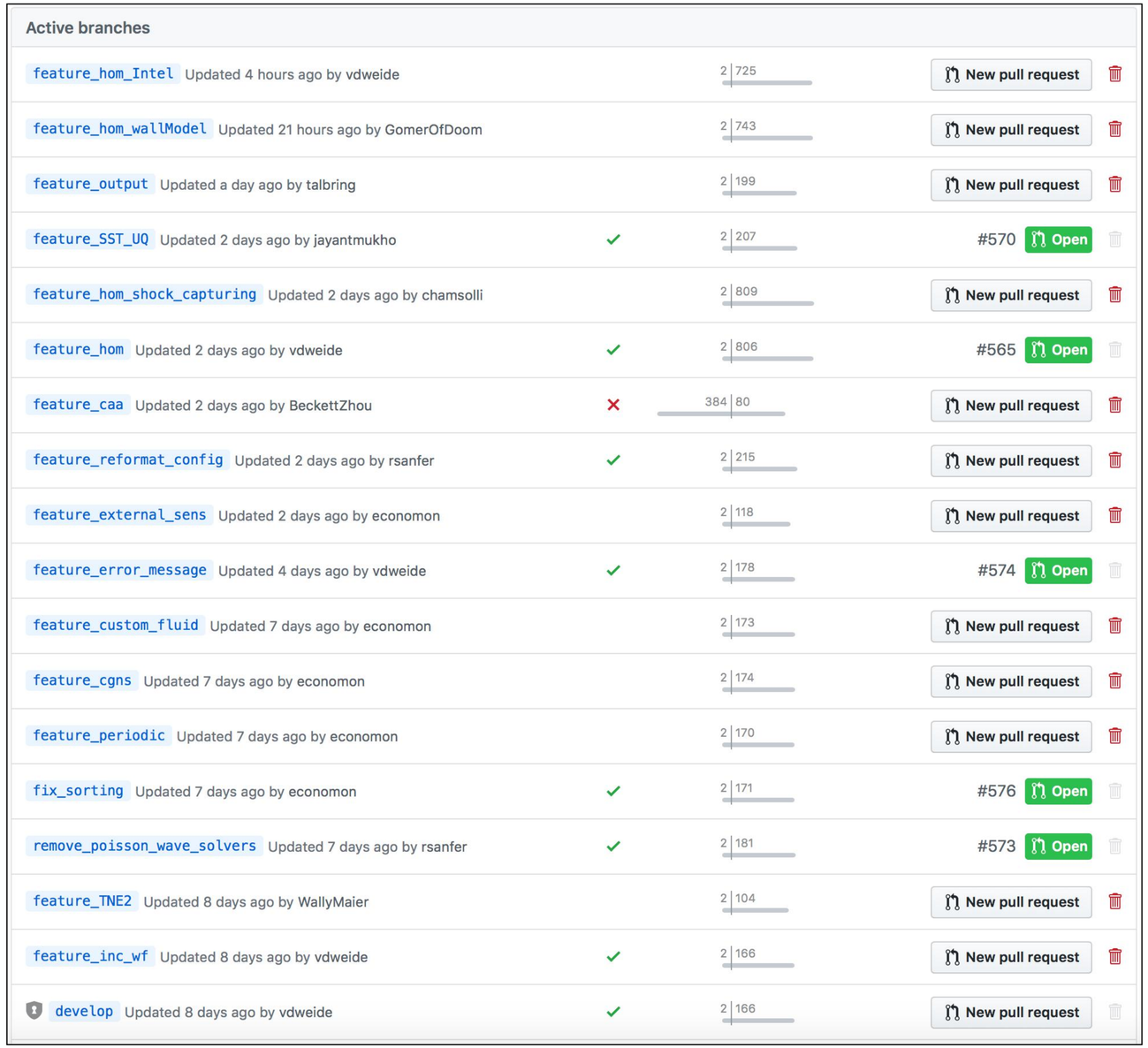
Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License: Creative Commons

Git/Branching — Development — Regressions — Pull Request — Release



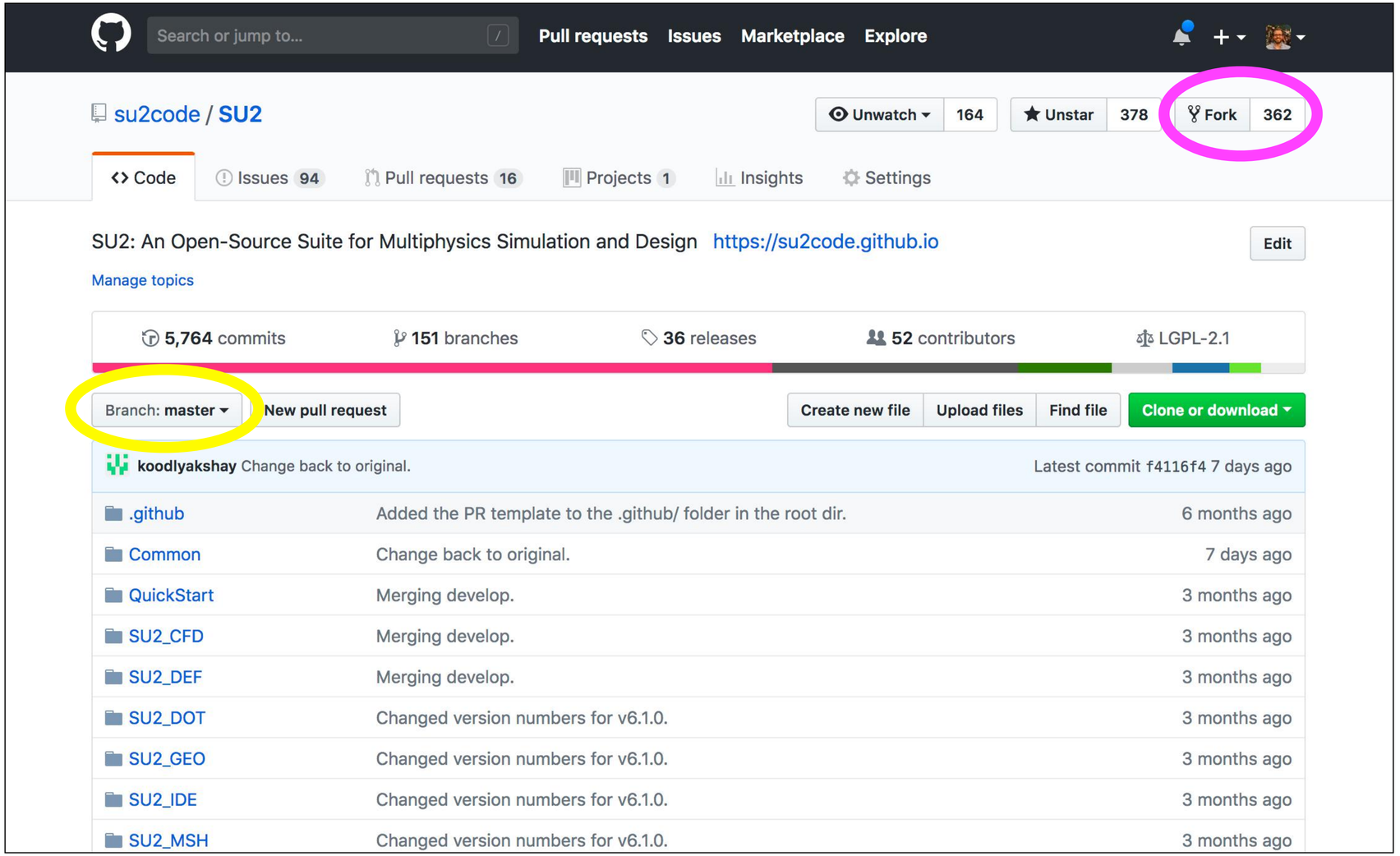
See all of our public repo branches here.

Git/Branching — Development — Regressions — Pull Request — Release



A current snapshot of active branches

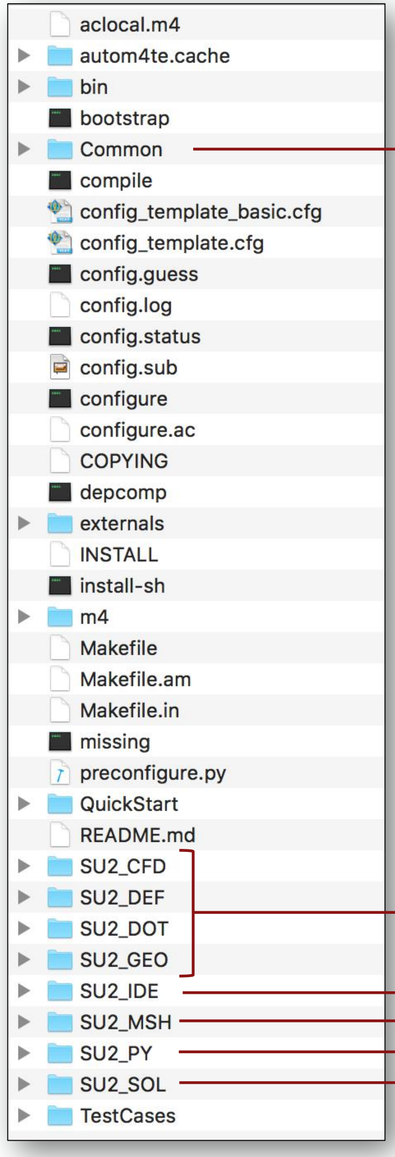
Note that develop is a protected branch



New branches can be made in the browser interface here or by pushing local branches to the remote with git.

Git/Branching — Development — Regressions — Pull Request — Release

Here is what you see inside the SU2/ repo.



Run ./bootstrap to reset autotools

Template config file with all options

External source files, e.g., ParMETIS

Inviscid NACA 0012

Config files for tests

C++ Source Code in SU2_*/src/, majority of lines in Common/src/ & SU2_CFD/src

IDE project files, e.g., Xcode

Python Scripts

C++ Executables

- SU2_CFD -> Primary multiphysics PDE solver for primal and adjoint
- SU2_SOL -> Solution export code
- SU2_DEF -> Mesh deformation
- SU2_DOT -> Gradient projection
- SU2_GEO -> Geometry definition
- SU2_MSH -> Mesh adaptation

Python Scripts (just a subset of them)

- parallel_computation.py
- mesh_deformation.py
- shape_optimization.py
- continuous_adjoint.py
- discrete_adjoint.py
- finite_differences.py
- direct_differentiation.py

- C++ class abstractions encourage code reuse and data encapsulation ensures you can make localized changes easily.

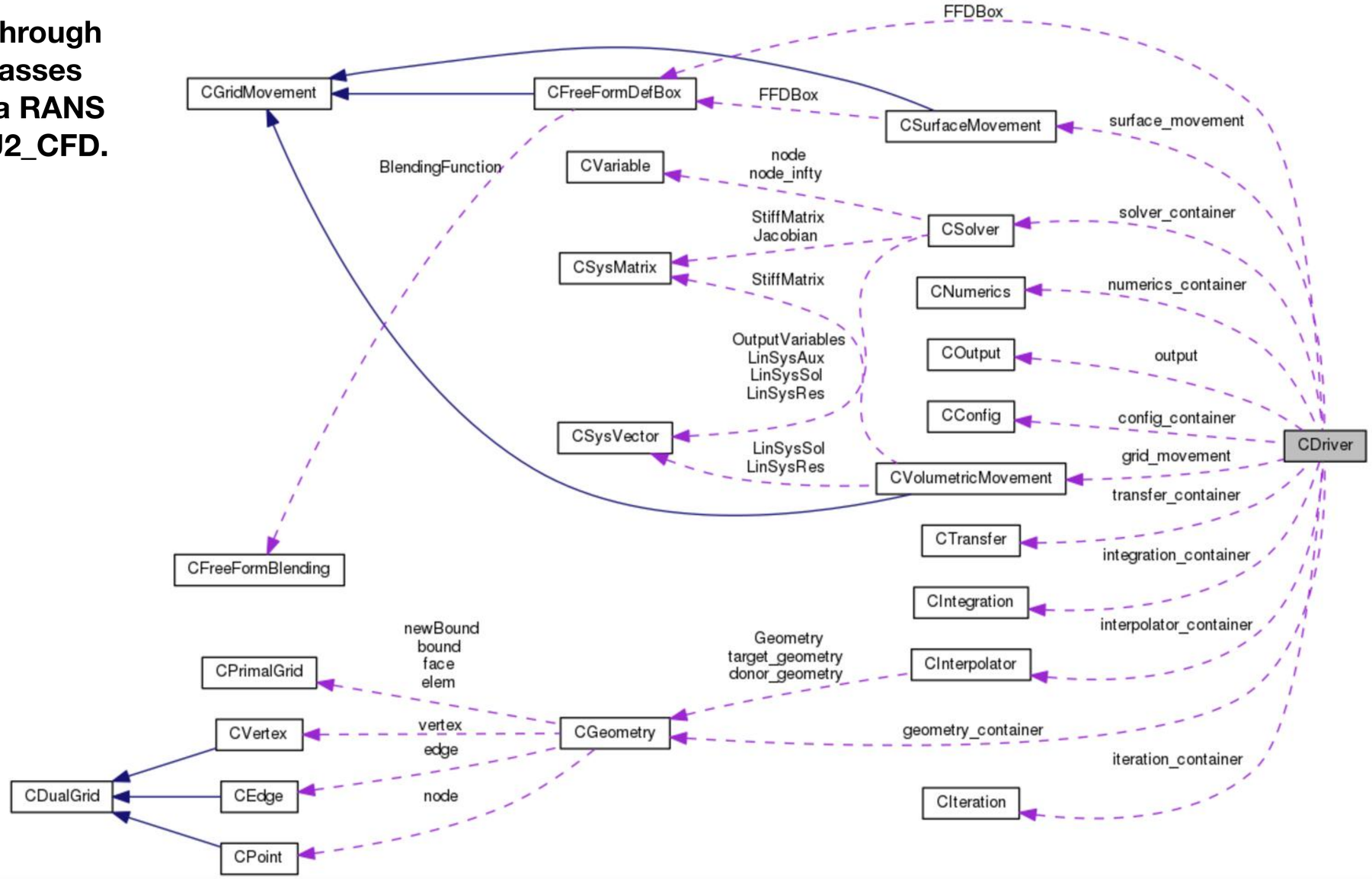
- Common base classes/methods (grid, linear solvers, output, etc.) are reused for many sets of physical governing equations.

- For a particular PDE, we define iteration, numerics, solver, and variable classes that are customized for the particular methods and algorithms.

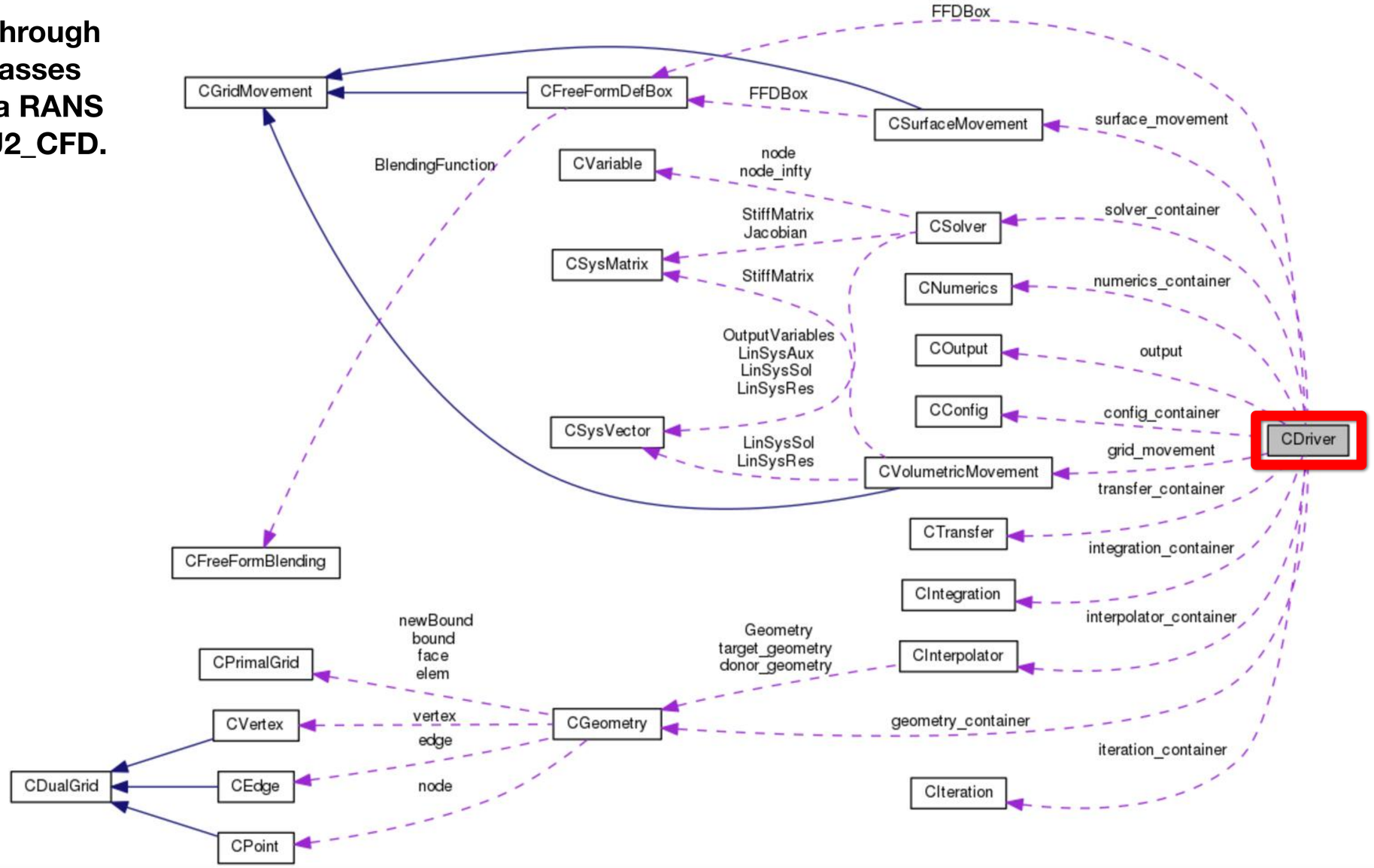
  - Legacy file division: solver_*.cpp, variable_*.cpp, numerics_*.cpp, contain child classes for a particular PDE, e.g., solver_direct_mean.cpp for mean flow.

  - We are in the process of moving to a single class per file convention!

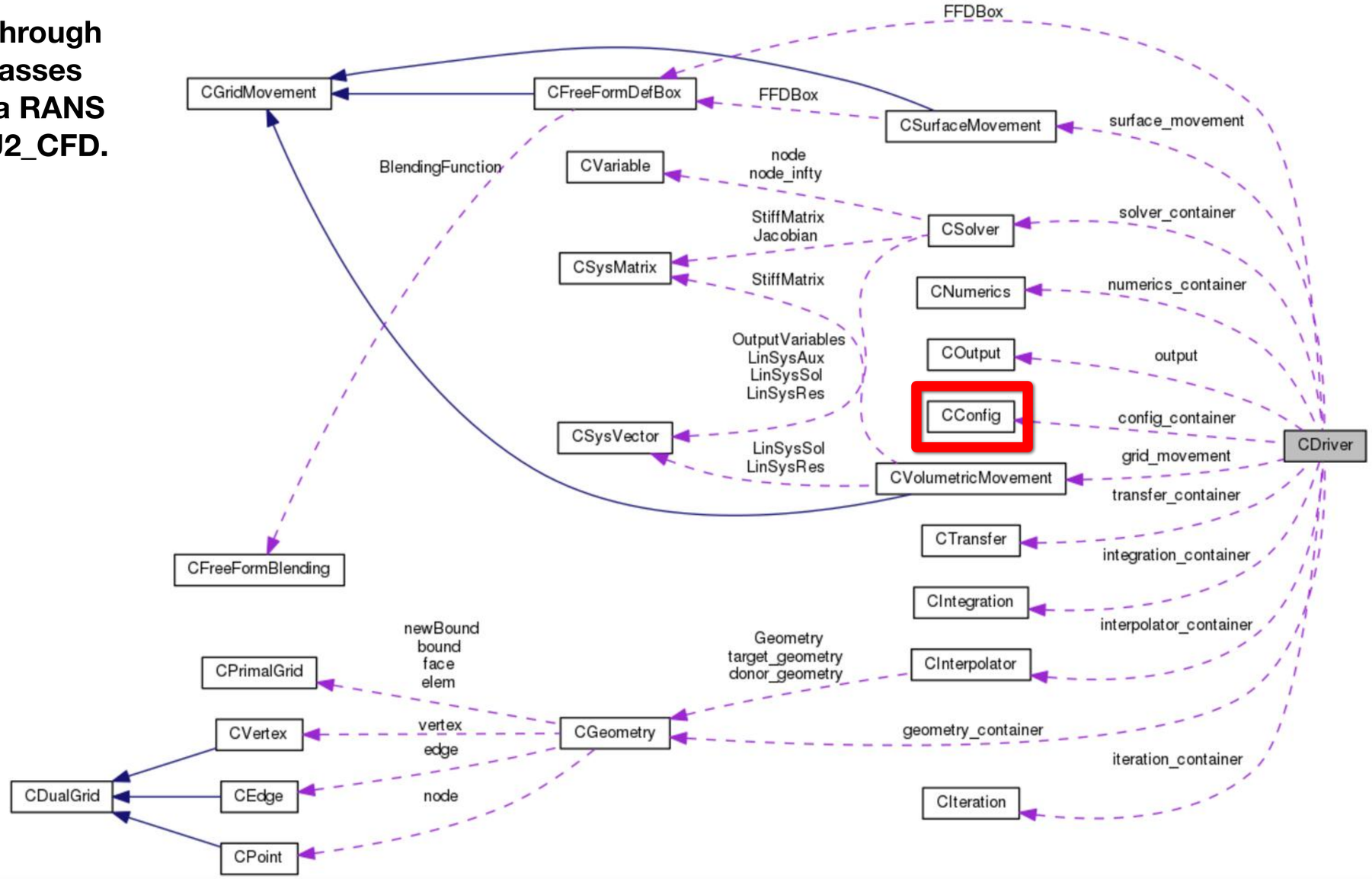
Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.



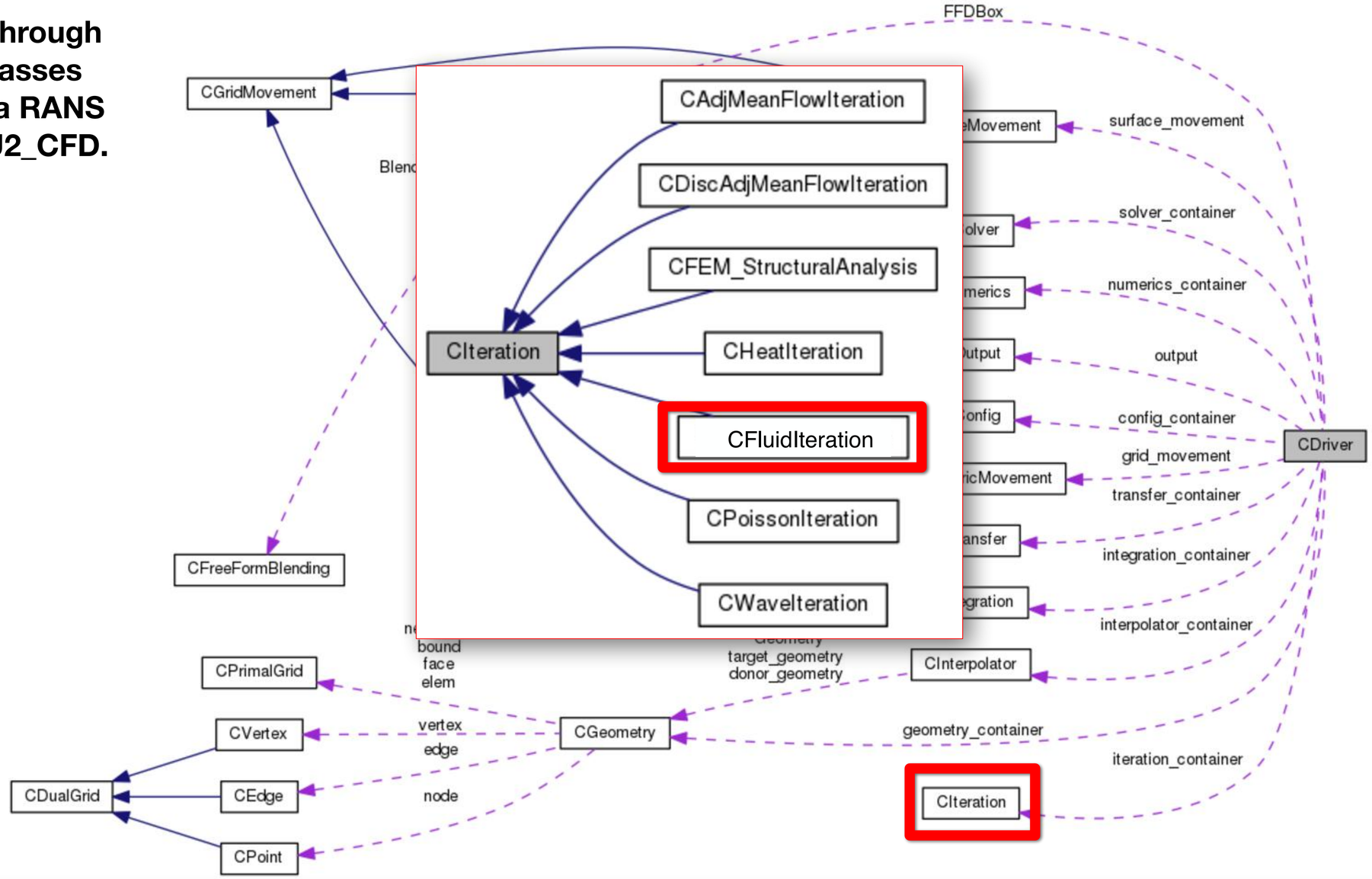
Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.



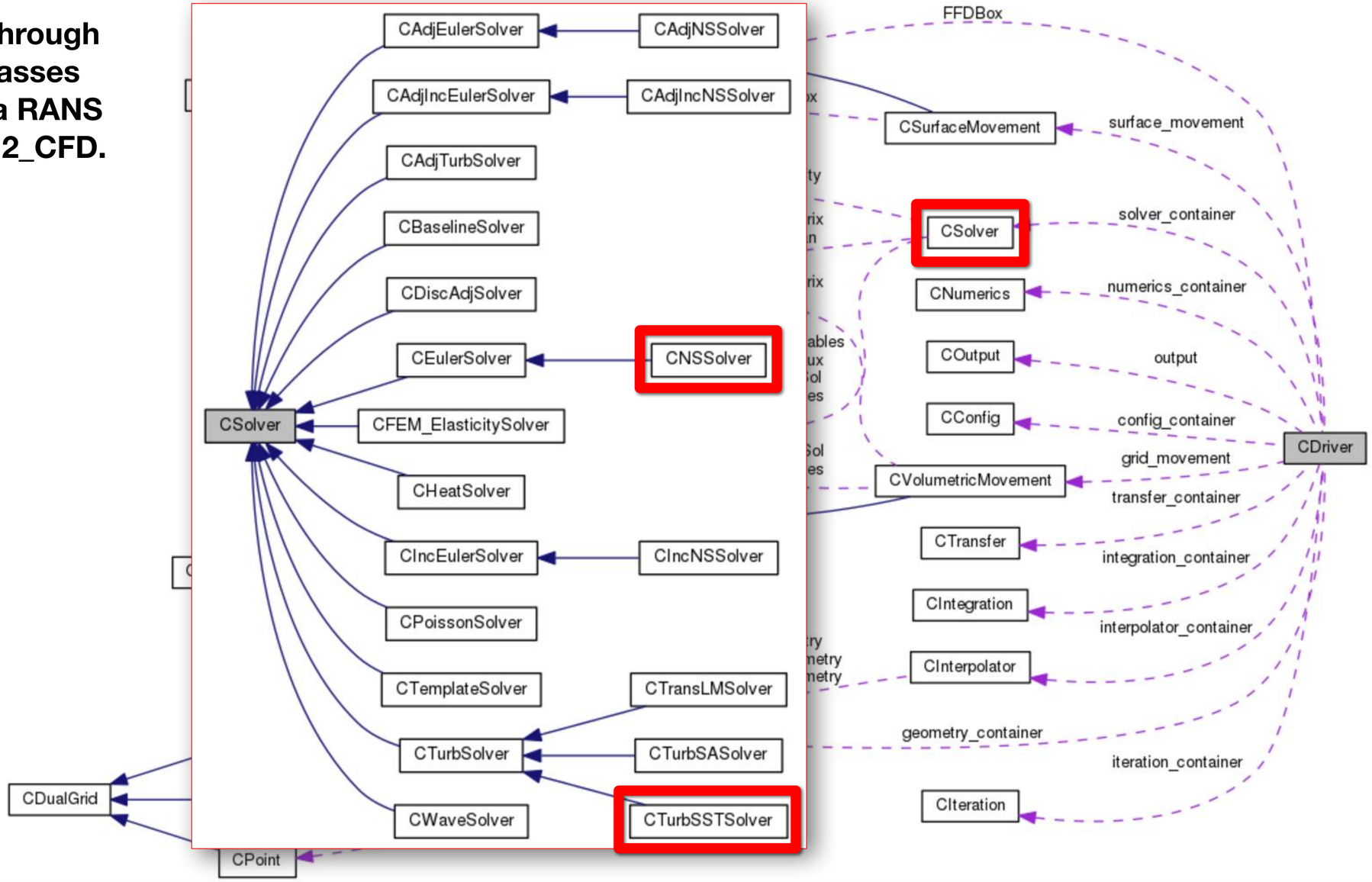
Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.



Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.

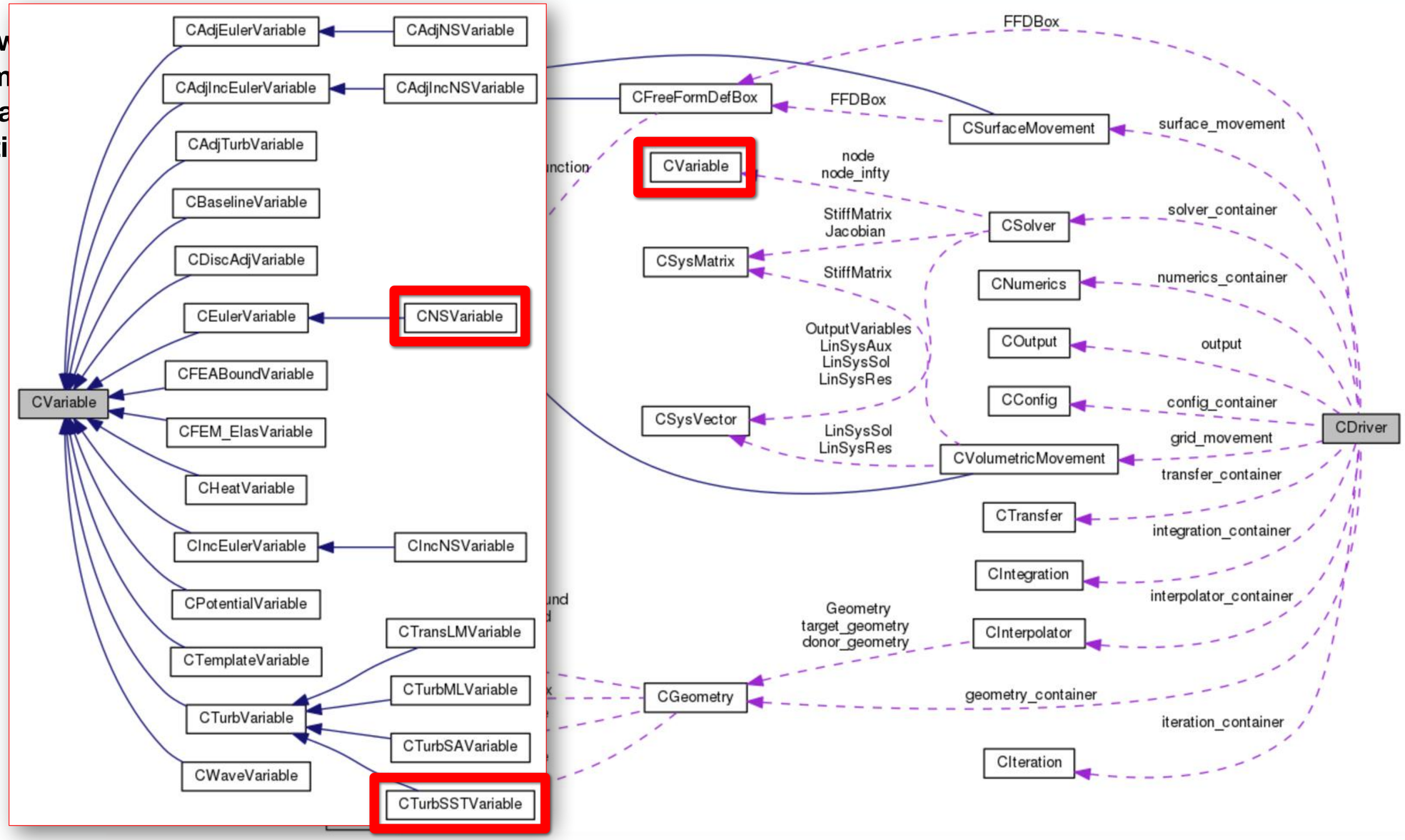


Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.

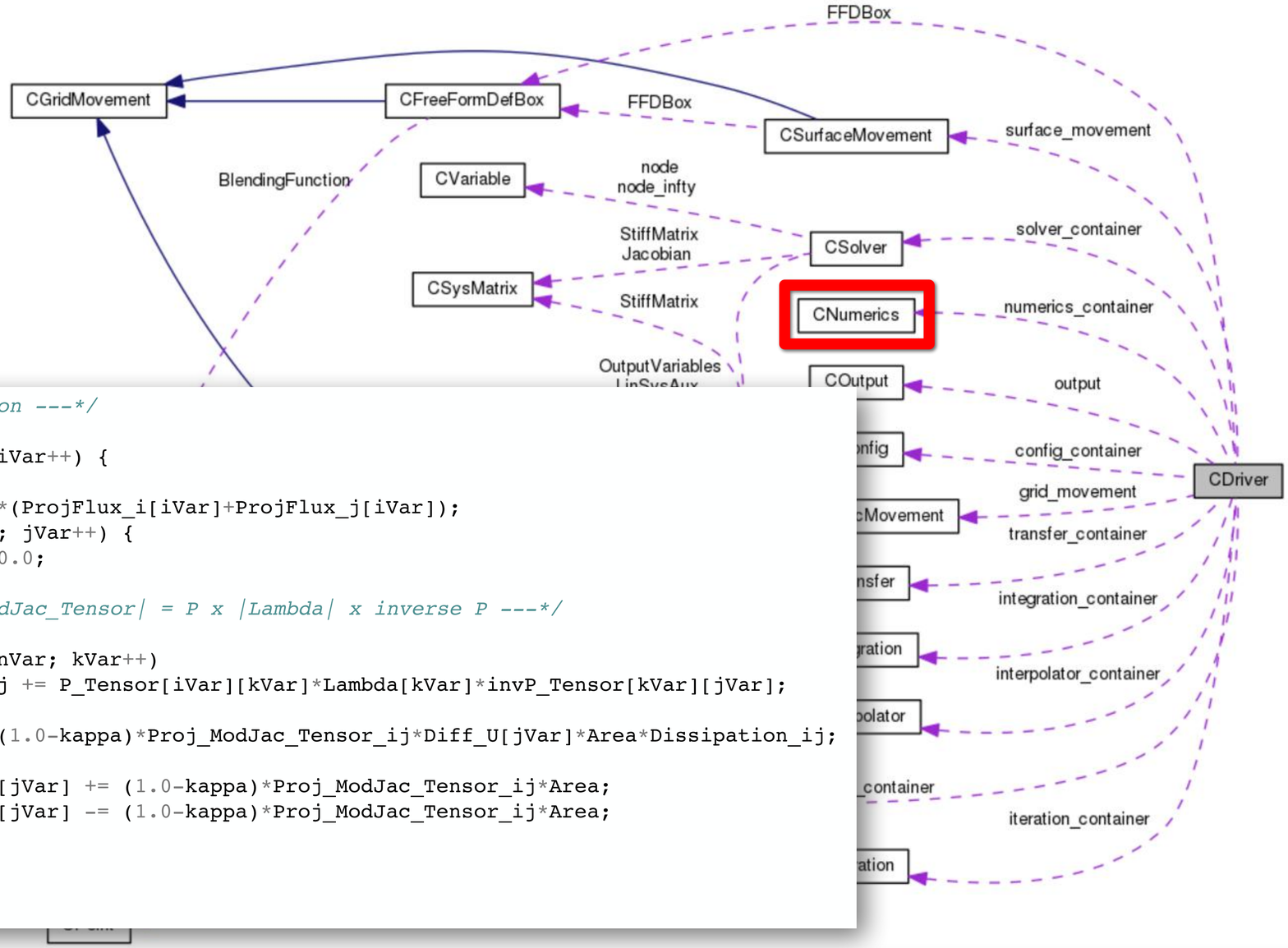


Git/Branching Development Regressions Pull Request Release

Top-dov
of som
instantia
calculati

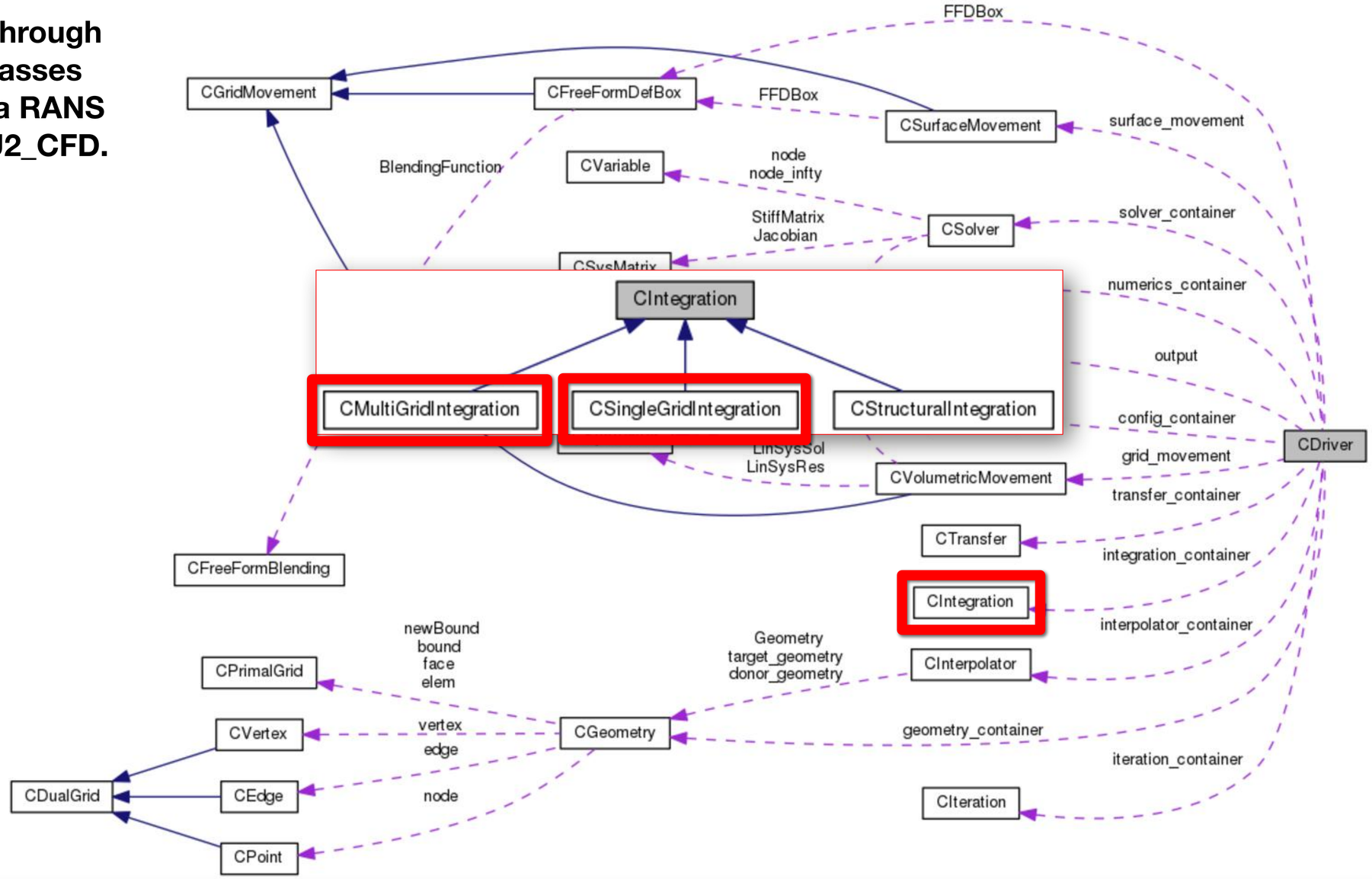


Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.



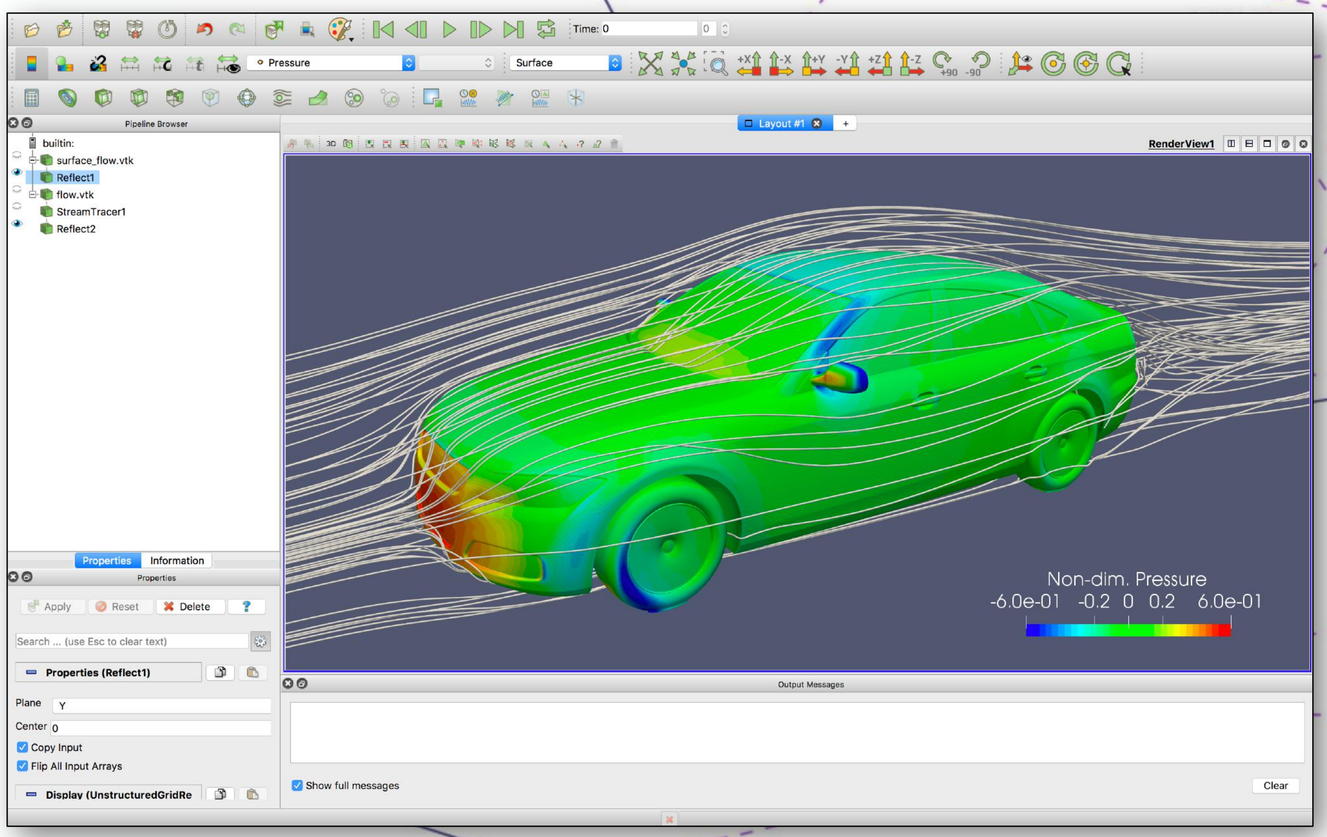

```
/*--- Roe's Flux approximation ---*/

for (iVar = 0; iVar < nVar; iVar++) {

  val_residual[iVar] = kappa*(ProjFlux_i[iVar]+ProjFlux_j[iVar]);
  for (jVar = 0; jVar < nVar; jVar++) {
    Proj_ModJac_Tensor_ij = 0.0;

      /*--- Compute |Proj_ModJac_Tensor| = P x |Lambda| x inverse P ---*/

      for (kVar = 0; kVar < nVar; kVar++)
        Proj_ModJac_Tensor_ij += P_Tensor[iVar][kVar]*Lambda[kVar]*invP_Tensor[kVar][jVar];

      val_residual[iVar] -= (1.0-kappa)*Proj_ModJac_Tensor_ij*Diff_U[jVar]*Area*Dissipation_ij;
      if(implicit){
        val_Jacobian_i[iVar][jVar] += (1.0-kappa)*Proj_ModJac_Tensor_ij*Area;
        val_Jacobian_j[iVar][jVar] -= (1.0-kappa)*Proj_ModJac_Tensor_ij*Area;
      }
  }
}
```

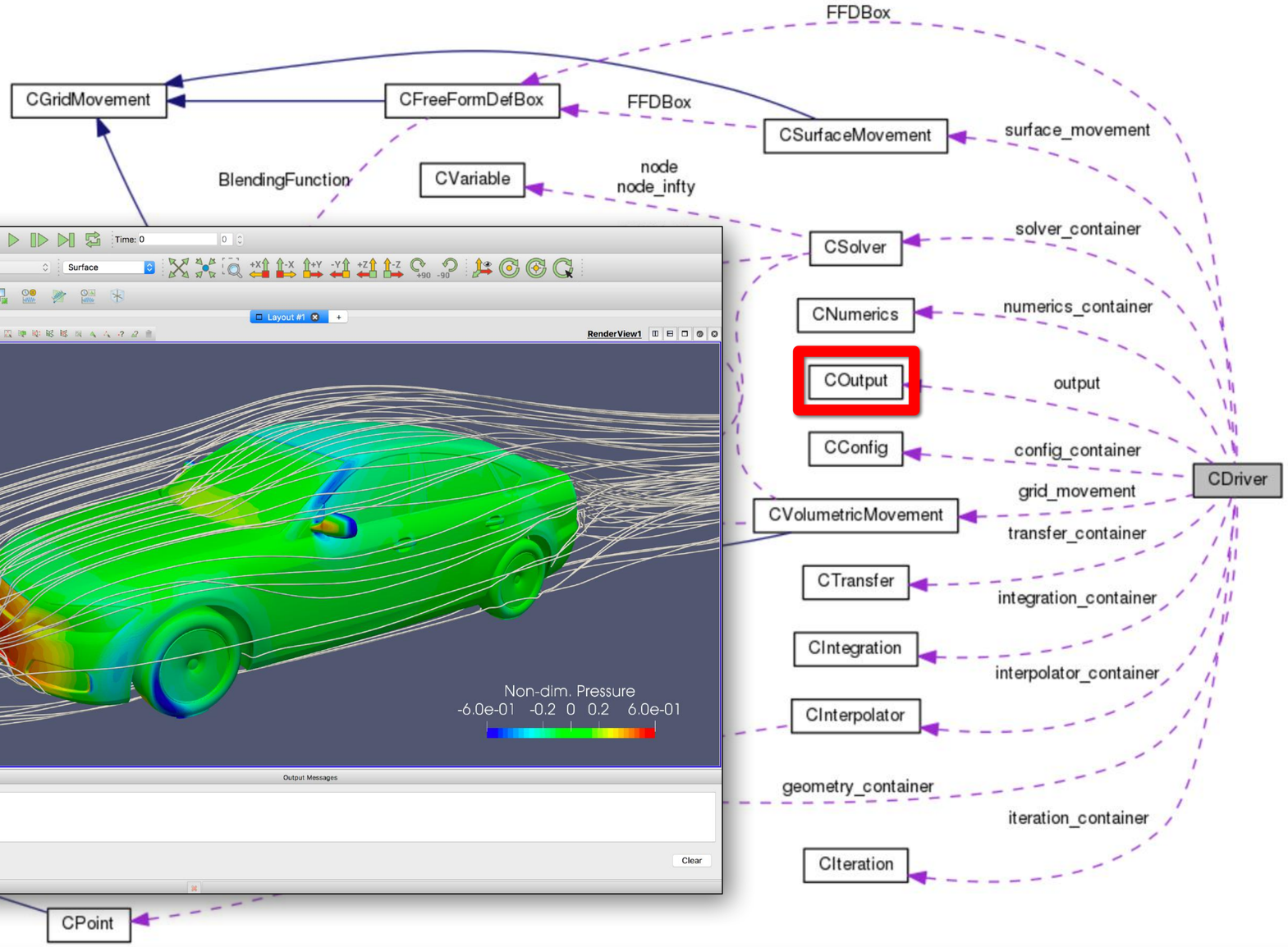
Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.



Git/Branching Development Regressions Pull Request Release

Top-down walkthrough of some key classes instantiated for a RANS calculation in SU2_CFD.





- Now that you know the basics, you are ready to create a new branch for your awesome feature (feature_awesome) and start hacking.

- But you might say, "Wait, how do I coordinate my contribution with other ongoing work in the repository?”

  - **Posting to GitHub in issues and projects or interacting with the SU2 Foundation technical Working Groups** are great ways to discuss potential developments and coordinate among other developers in the open.

- And then maybe you’ll ask, “How can I make sure that my work doesn’t ‘break’ other capabilities that already exist in SU2?”

  - **Continuous integration will save your bacon.** Travis CI is free for open-source!

Git/Branching — Development — Regressions — Pull Request — Release

SU2 (ver. 6.1.0 "Falcon"): The C

Computational analysis tools have revolutioniz
proprietary, unavailable, or prohibitively expen
analysis and design freely available as open-s

For an overview of the technical details in SU2

"SU2: An open-source suite for multiphysics s
http://arc.aiaa.org/doi/10.2514/1.J053813

build passing

Our security blanket: a comprehensive suite of ~200 regression test cases for serial, parallel, physics, AD, python, etc.

su2code / SU2 build unknown

Travis CI

Current Branches Build History Pull Requests Build #2312 More options

develop Merge pull request #575 from pecos-hybrid/bugfix_viscosity_in_FD

Change dynamic viscosity in FD hybrid central / upwind blending to kinematic viscosity

Commit d00c3b3
Compare a0ecddf..d00c3b3
Branch develop

Edwin van der Weide authored GitHub committed

#2312 passed

Ran for 2 hrs 5 min 54 sec
Total time 8 hrs 9 min 37 sec

8 days ago

Restart build

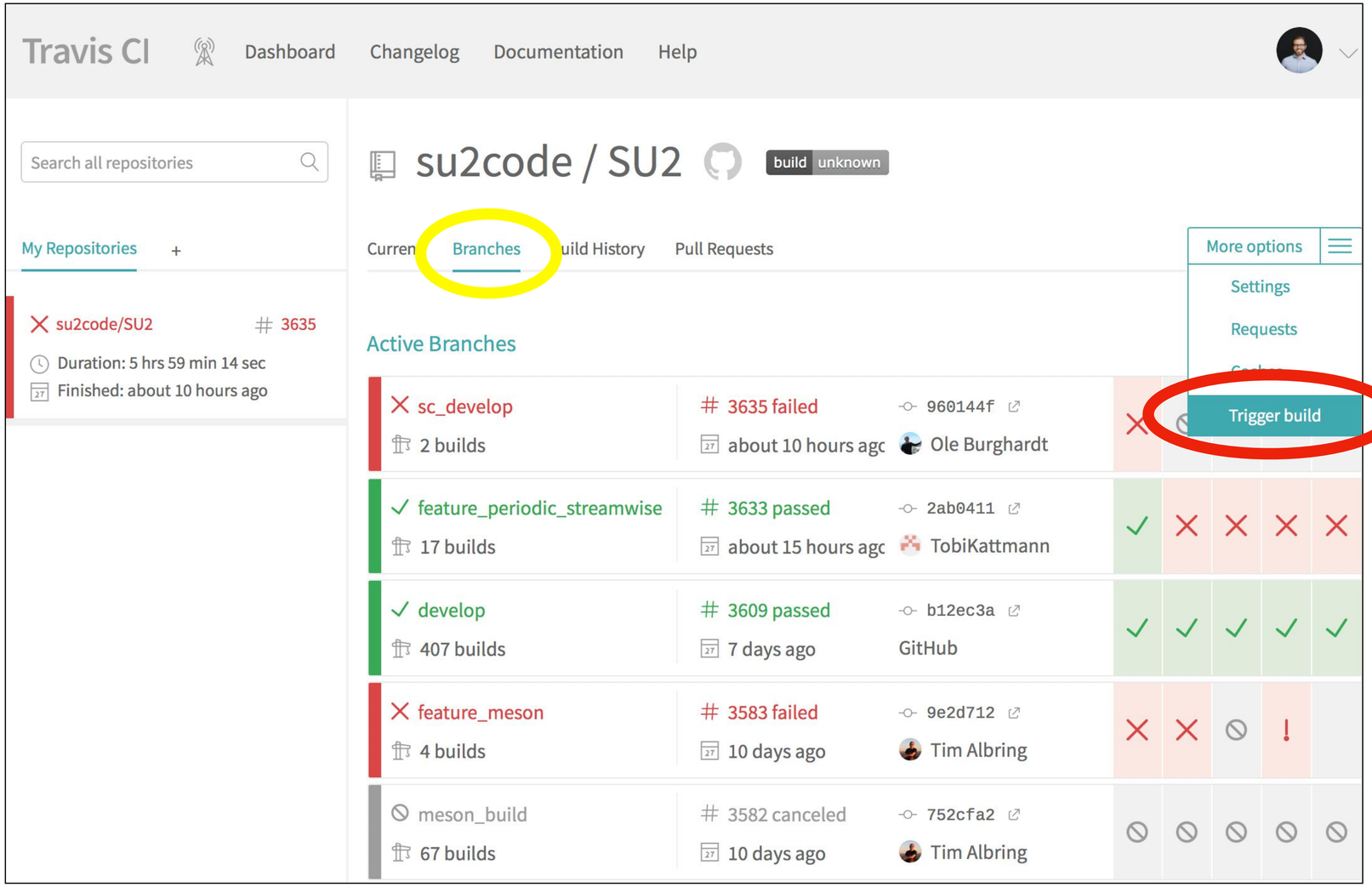
Build Jobs

✓	# 2312.1	Python: 2.7	CONFIGURE_COMMAND="./preconfigure.py --prefix=$TRAVIS_BUI...	1 hr 3 min 49 sec
✓	# 2312.2	Python: 3.6	CONFIGURE_COMMAND="./preconfigure.py --prefix=$TRAVIS_BUI...	1 hr 3 min 25 sec
✓	# 2312.3	Python: 2.7	CONFIGURE_COMMAND="./preconfigure.py --enable-mpi --with-cc...	58 min 49 sec
✓	# 2312.4	Python: 3.6	CONFIGURE_COMMAND="./preconfigure.py --enable-mpi --with-cc...	58 min 31 sec
✓	# 2312.5	Python: 2.7	CONFIGURE_COMMAND="./preconfigure.py --with-cc=gcc --with-c...	1 hr 3 sec
✓	# 2312.6	Python: 3.6	CONFIGURE_COMMAND="./preconfigure.py --with-cc=gcc --with-c...	59 min 52 sec
✓	# 2312.7	Python: 2.7	CONFIGURE_COMMAND="./preconfigure.py --enable-mpi --with-cc...	1 hr 36 sec
✓	# 2312.8	Python: 3.6	CONFIGURE_COMMAND="./preconfigure.py --enable-mpi --with-cc...	1 hr 4 min 32 sec

Git/Branching — Development — Regressions — Pull Request — Release

Use continuous integration to guide your development by activating Travis CI for your own branches! Update SU2/.travis.yml with your own email and branch.

```
# Continuous Integration setup for SU2.

dist: trusty
sudo: required

language: python

compiler:
    - gcc

notifications:
    email:
        recipients:
            - your.email@here.com

branches:
    only:
        - feature_awesome
```



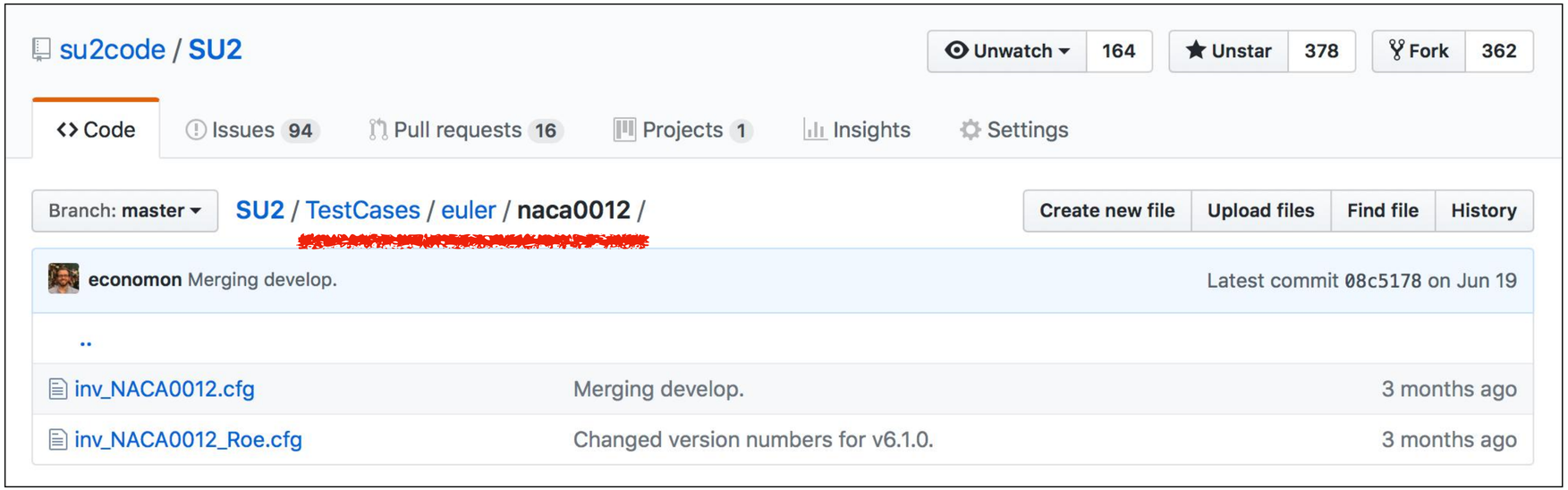
New capabilities in your feature branch should also have a test case to protect them in the future.

```
# NACA0012
naca0012             = TestCase('naca0012')
naca0012.cfg_dir     = "euler/naca0012"
naca0012.cfg_file    = "inv_NACA0012_Roe.cfg"
naca0012.test_iter   = 20
naca0012.test_vals   = [-4.047448, -3.538057, 0.338691, 0.023131] #last 4 columns
naca0012.su2_exec    = "SU2_CFD"
naca0012.timeout     = 1600
naca0012.tol         = 0.00001
test_list.append(naca0012)
```

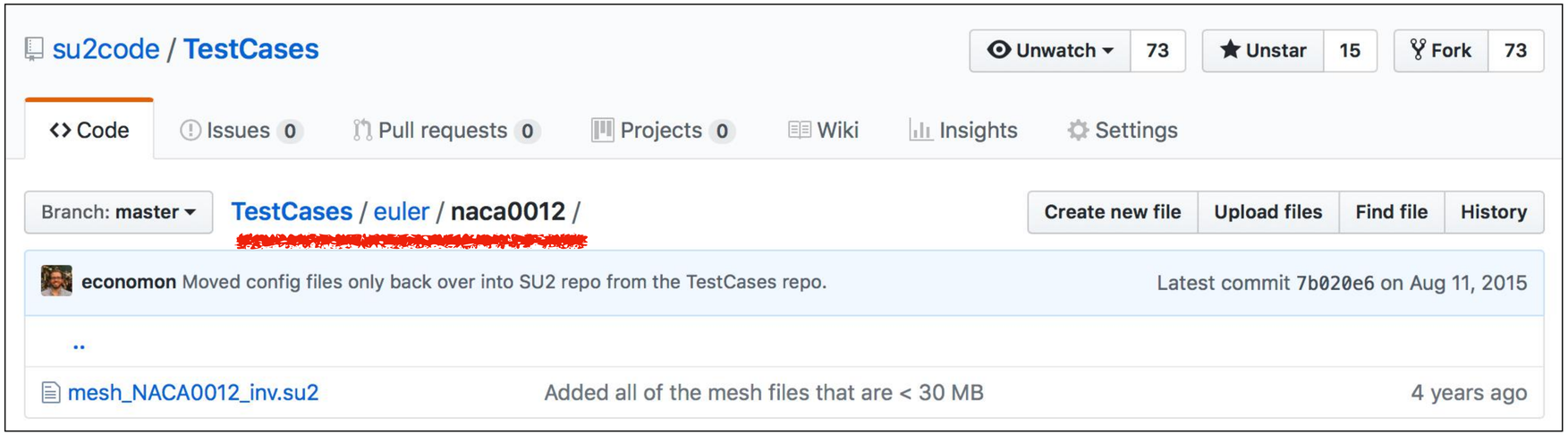
If the computed values after one of your commits don't match these values, you will get an email with details of the failed cases. Investigate it!

1. Add a new test case to serial_regression.py, parallel_regression.py, etc. Use others as a guide. See NACA 0012 example.

2. Put the config file and any supporting data in the corresponding locations. Travis CI combines the complementary sets.



Lighter weight, more frequently updated files go in code repo.



Larger, more static files that support the tests go in the TestCases repo.

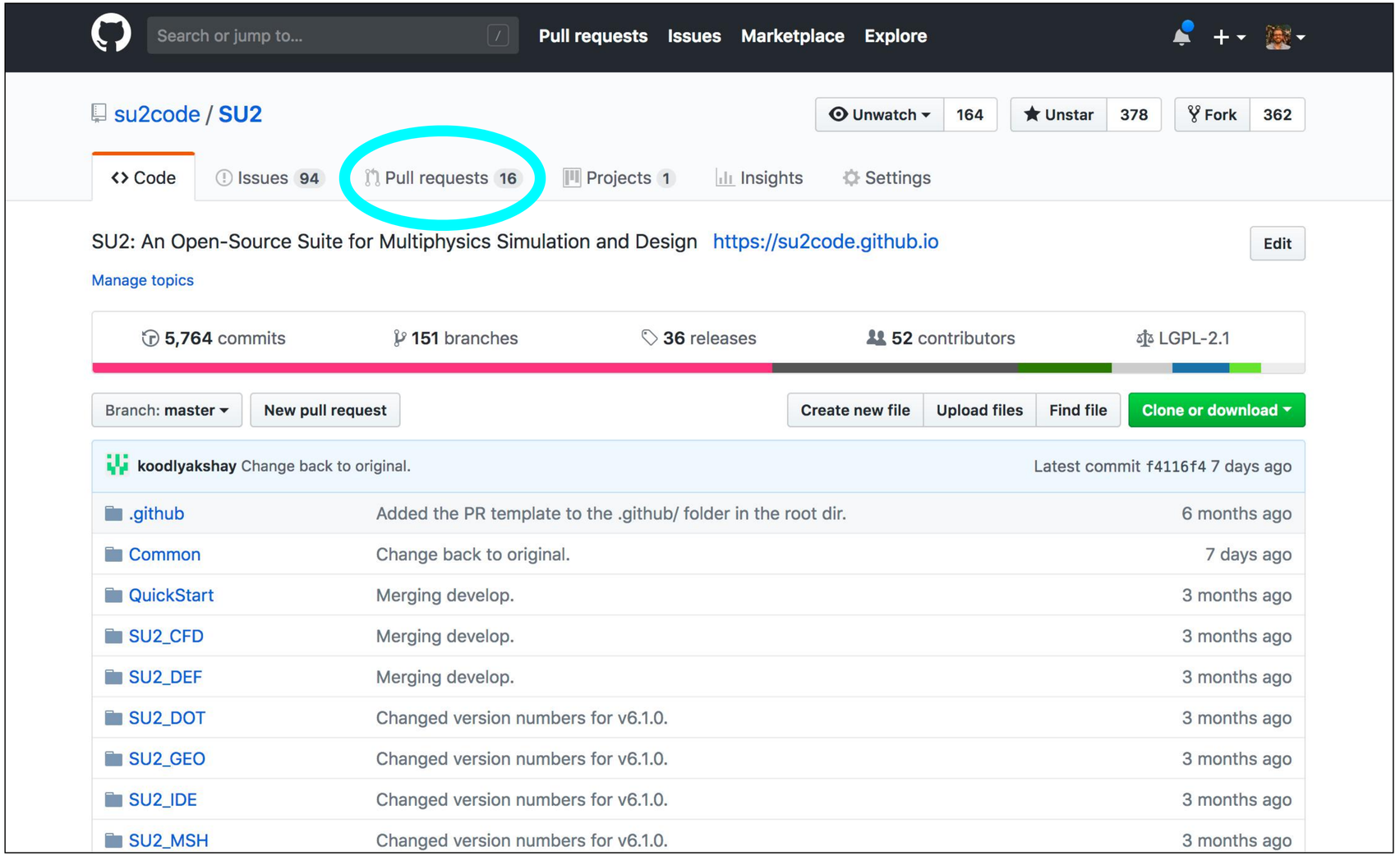
- So, you've finished your awesome feature and the tests are passing. You've even added your own regression test (or two), and you checked that there are no new compiler warnings and the style conforms to the SU2 standard.

- At this point you are wondering, "I would like to contribute my feature to the open source, but how do I do that?"

- To get your work into an official open-source release of SU2, the modifications have to first be staged in the develop branch.

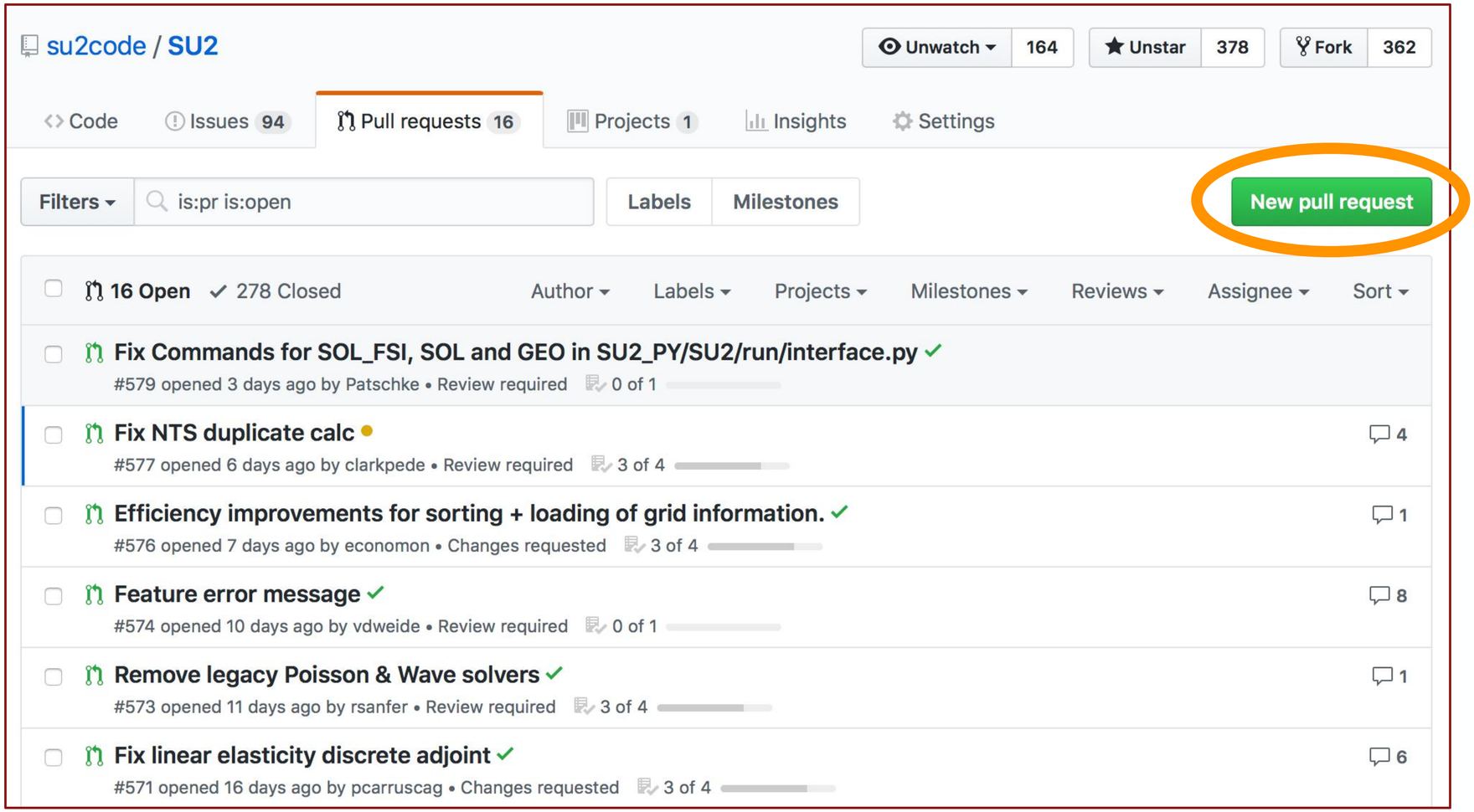
- To do so, we use the standard **Pull Request (PR)** approach.

Git/Branching — Development — Regressions — Pull Request — Release



Once you're ready to contribute, it's PR time.

Git/Branching — Development — Regressions — Pull Request — Release

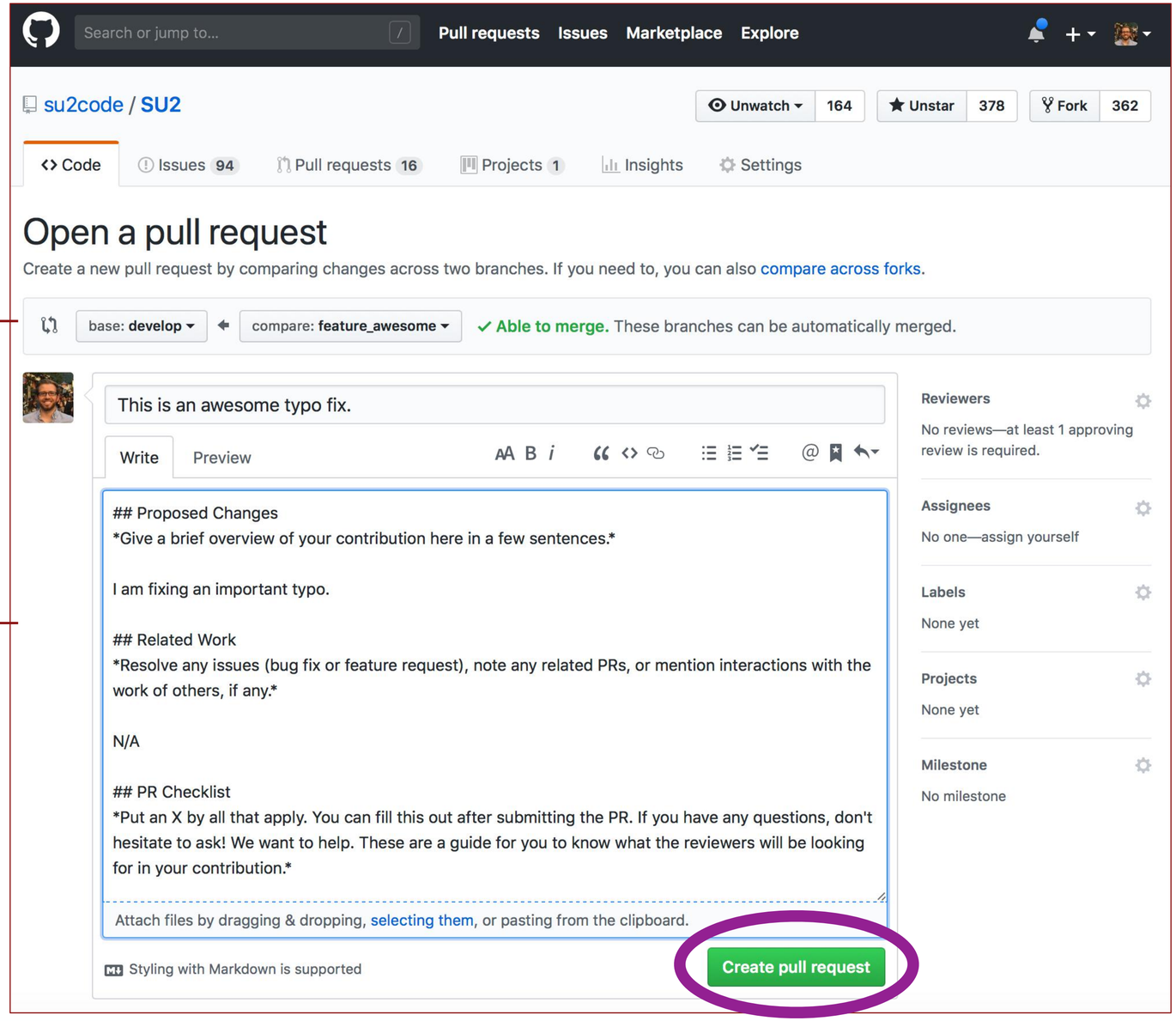


A PR is a request to the project to pull in your contribution. Can be from an internal branch or from an external fork.

Git/Branching — Development — Regressions — Pull Request — Release

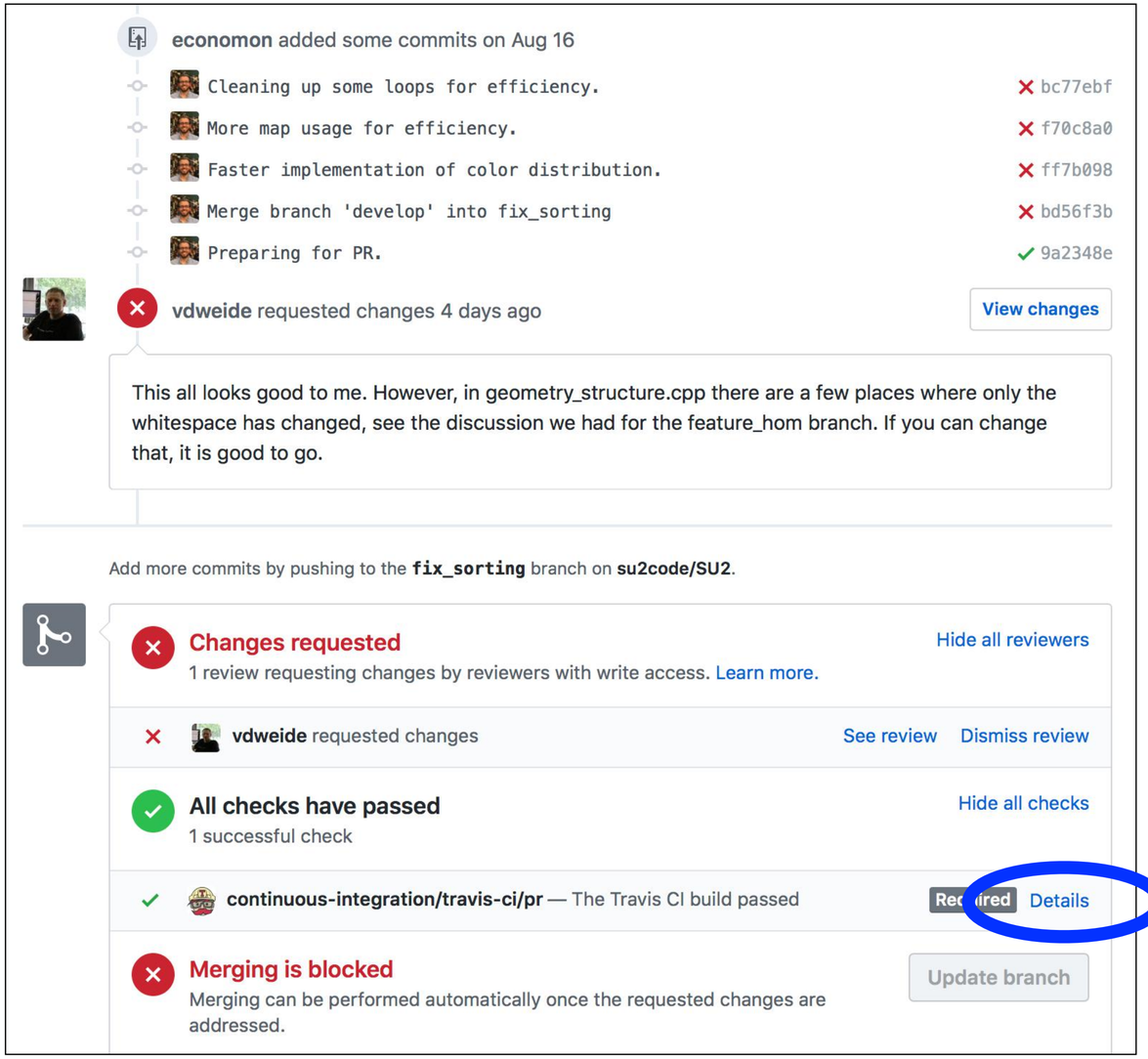
Submit the PR to the develop branch

Fill out the PR template questions that guide you along your way.



Git/Branching Development Regressions Pull Request Release

You can now open draft PRs on GitHub to work in the open - get feedback and continuously run regressions on your branch!

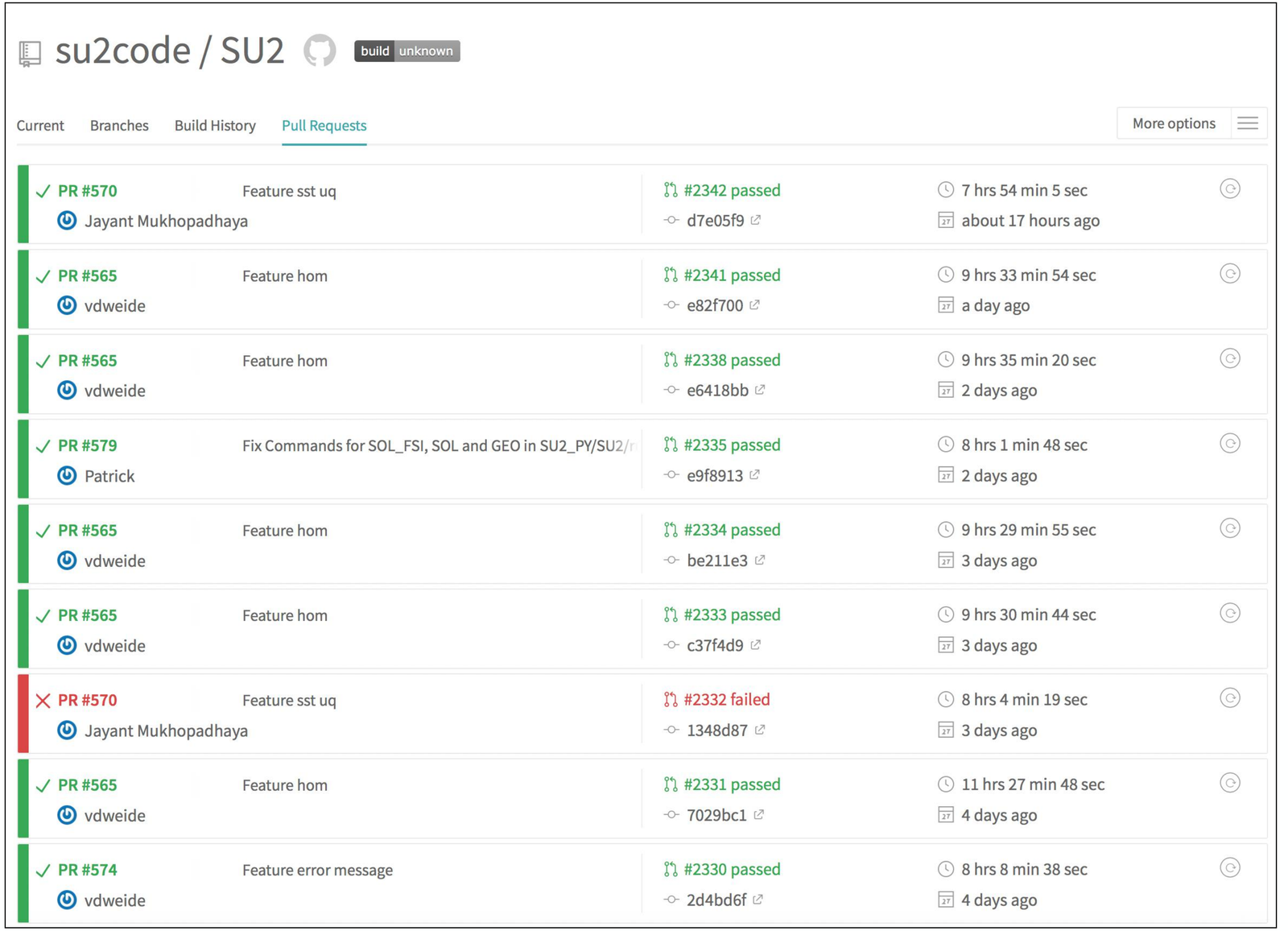


All regression tests must pass with your code integrated. Travis CI again takes care of this for us. Merging is blocked until passage.

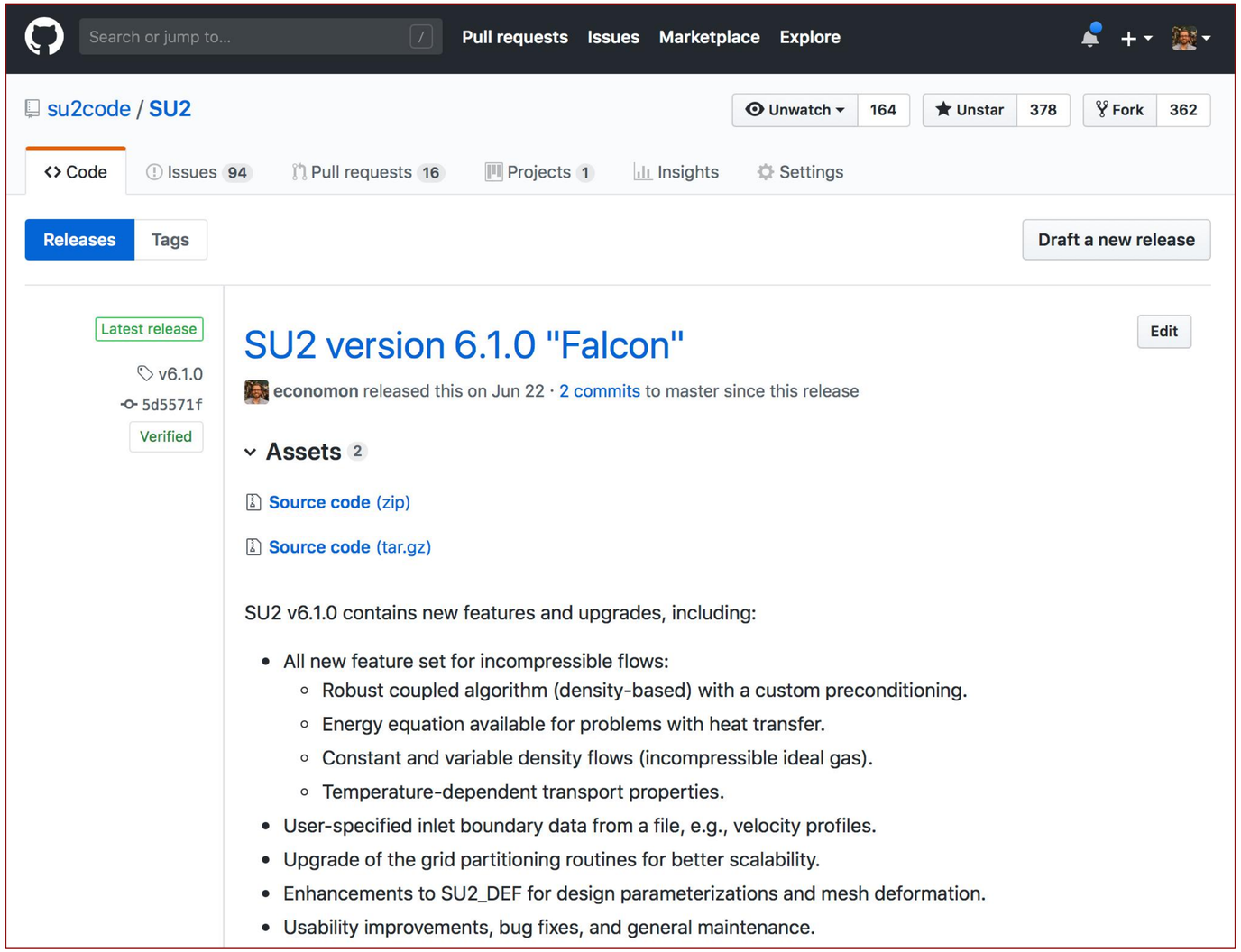
Code is reviewed by fellow developers for content, organization, and style. PR is blocked until at least one approval! Our convention is 2 reviewer approvals.

PRs keep community informed, offer opportunity for discussion, and are a *controlled gate* for quality assurance of contributions.

Details of the tests for all PRs can be found over in Travis CI.



Git/Branching Development Regressions Pull Request Release



Releases: we move develop to master, create tags, binaries, and advertise. Your awesome feature is released!

Git/Branching Development Regressions Pull Request Release

SU2 Downloa Docs Tutorials Forum Develop IDS News

- Introduction to SU2
  - SU2, the Open-Source CFD Code
  - Contribute
- Quick Start
- Installation
- Input Files
- Running SU2
- Developer Docs
- FAQ
- Contact

SU2, the Open-Source CFD C

SU2

The Open-Source CFD

The SU2 suite is an open-source collection of C++ based software tools
Partial Differential Equation (PDE) analysis and solving PDE-constrained
problems. The toolset is designed with Computational Fluid Dynamics (
aerodynamic shape optimization in mind, but is extensible to treat arbi
governing equations such as potential flow, elasticity, electrodynamics,
reacting flows, and many others. SU2 is under active development by in
around the world on GitHub and is released under an open-source licer

For a detailed description of the code philosophy, components, and imp
please see our SU2 AIAA Journal article.

Improve this page

https://su2code.github.io

Search or jump to... / Pull requests Issues Marketplace Explore

su2code / su2code.github.io

Unwatch 58 Star 1 Fork 16

Code Issues 0 Pull requests 1 Projects 0 Wiki Insights Settings

Branch: master su2code.github.io / _docs / index.md Find file Copy path

economon New website 8766c49 on Feb 13

1 contributor

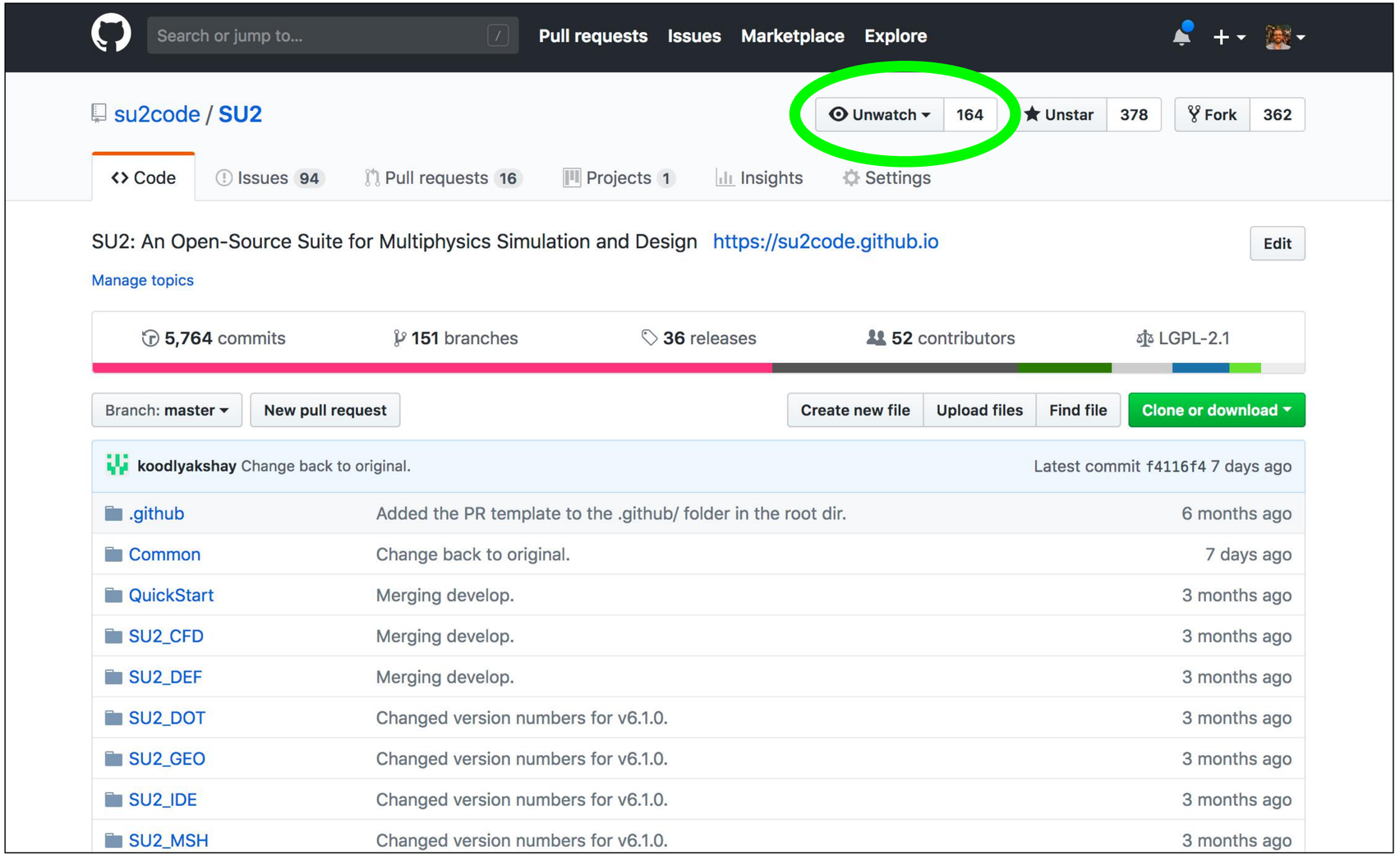
12 lines (8 sloc) 934 Bytes Raw Blame History

title	permalink	redirect_from
SU2, the Open-Source CFD Code	/docs/home/	/docs/index.html

The SU2 suite is an open-source collection of C++ based software tools for performing Partial Differential Equation (PDE) analysis and solving PDE-constrained optimization problems. The toolset is designed with Computational Fluid Dynamics (CFD) and aerodynamic shape optimization in mind, but is extensible to treat arbitrary sets of governing equations such as potential flow, elasticity, electrodynamics, chemically-reacting flows, and many others. SU2 is under active development by individuals all around the world on GitHub and is released under an open-source license.

For a detailed description of the code philosophy, components, and implementations, please see our SU2 AIAA Journal article.

Documentation and tutorials are critical for amplifying the impact of your work. Good news: it's the same process to create it.



Keep up-to-date via email with all of the activity in the repo by “watching”

SU2 Development Survival Guide

A Best Practice Workflow

1. Clone main repository: $ git clone https://github.com/su2code/SU2.git

2. Create new feature branch (in remote and locally) for your development work. Work on this branch in the repo.

3. Activate the regressions for your branch by changing to your branch name and email in .travis.yml (trigger builds manually in Travis CI interface). Use this to guide development and correct any failures along the way that you will be informed of by email.

4. Work on your feature! Please mind white space issues, compiler warnings, and match SU2 style.

5. If you are working on a single branch for an extended amount of time, merge the remote develop branch into your own branch at regular, frequent intervals. This ensures that, when the time comes, it will be easy to merge your contribution into develop, as you will have resolved any conflicts on your side before a PR.

6. Once you feel your feature is ready, submit a PR. Fill out the PR template that is provided for you. Consider opening a draft PR earlier in the process to get feedback sooner and have your branch tested by Travis CI while you work.

7. Get reviews and engage with the community concerning your contribution. Fix problems in your branch or address any feedback on the message boards. Note that any new commits will appear right there in the open PR and will kick-off the regressions again.

8. Once the reviewers approve and the regressions pass, the community will merge in your work.

9. Celebrate your contribution and proudly introduce yourself as an SU2 developer at your social engagements.

Pretty easy to be a developer, huh?

We have set up safety nets and removed overhead wherever possible.

So, try crazy ideas in your branches. Don’t be afraid to make big changes that push the boundaries of the code. The community and infrastructure will be there to help you. This is how we make progress.