A Crash Course for SU2 code Developers

Tim Albring, Developer Director

1st SU2 Conference 2020

SU2 foundation

So, you want to be an SU2 developer?

It's easier than you might think.

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

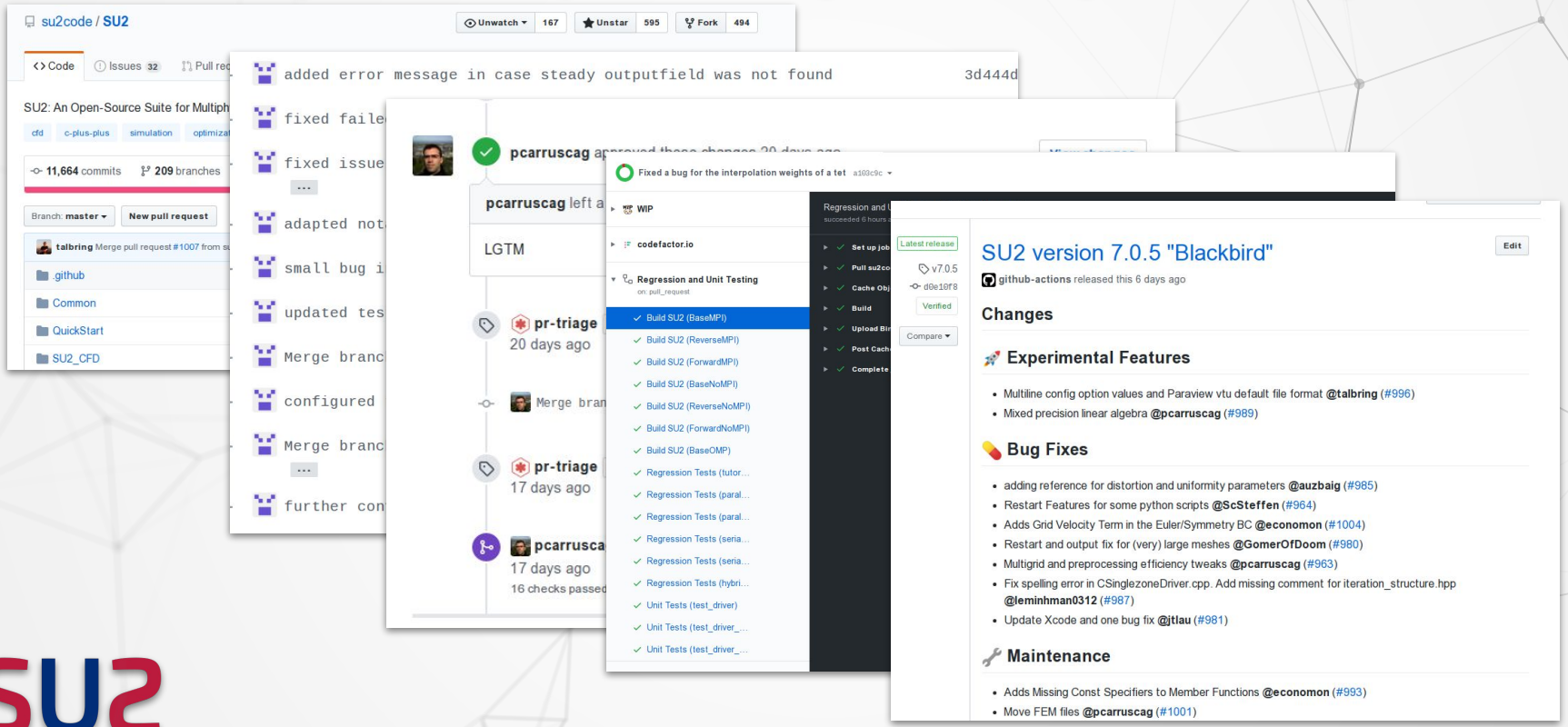
SU2 foundation

Disclaimer - What this talk is not

- *A programming tutorial*
- *A guide on how to install and build SU2*
- *A source-code walkthrough*
- *A professional software engineering lecture*
- *Only for experienced developers*
- *Only for new developers*

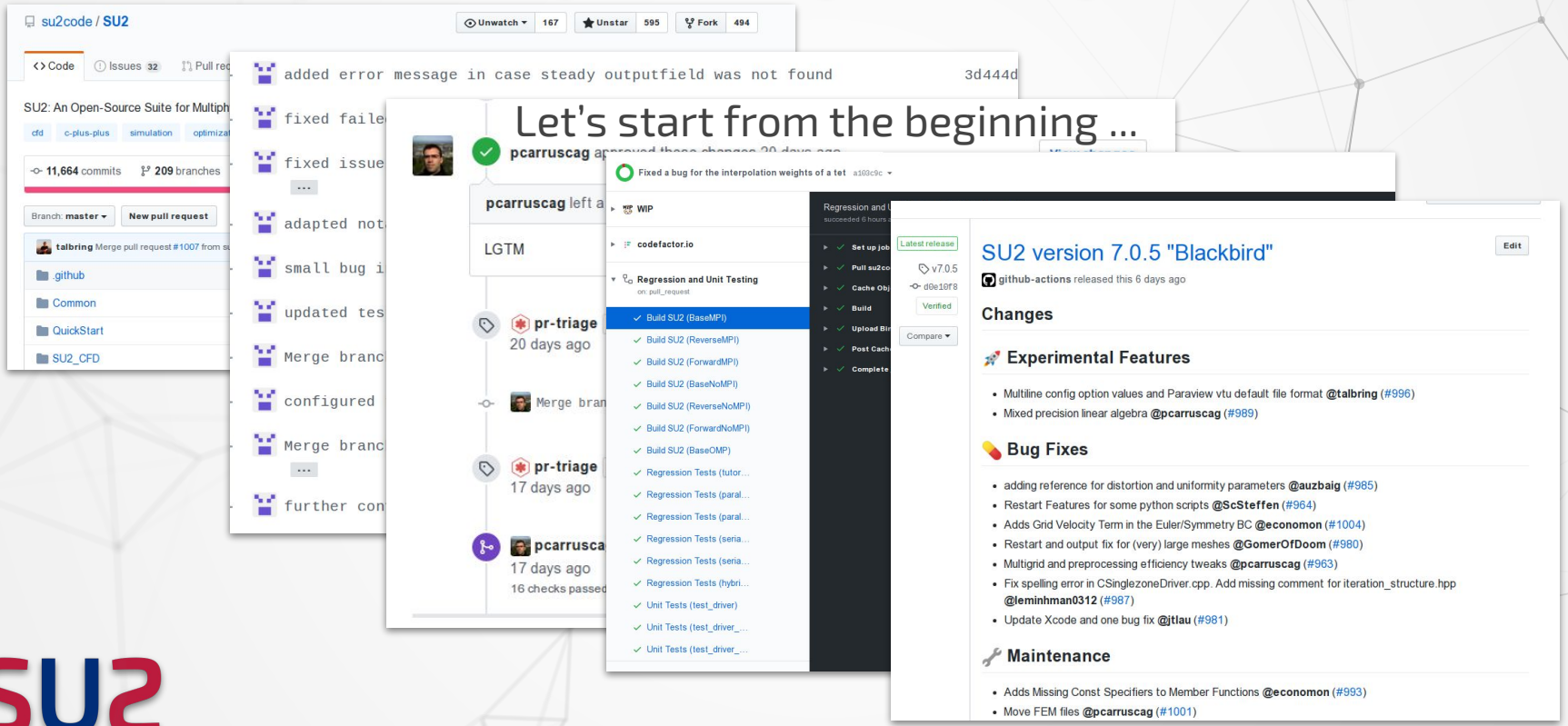
SU2 foundation

Git/Branching Development Pull Request Testing Release



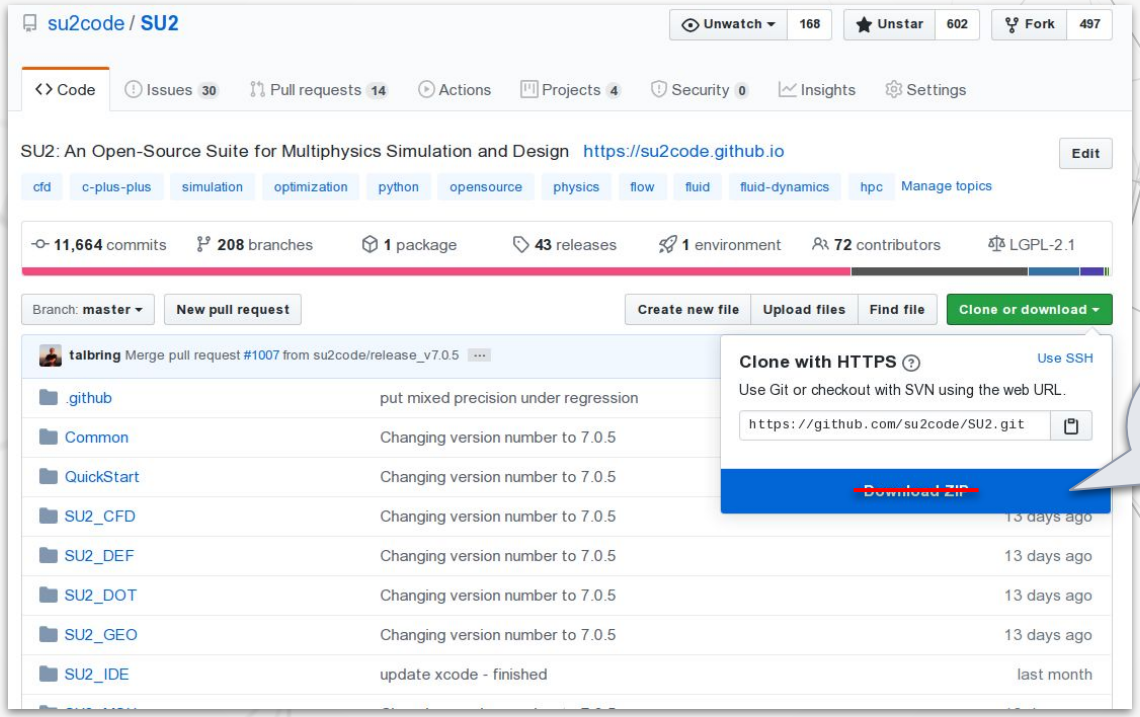
SU2 foundation

Git/Branching Development Pull Request Testing Release



SU2 foundation

Git/Branching Development Pull Request Testing Release



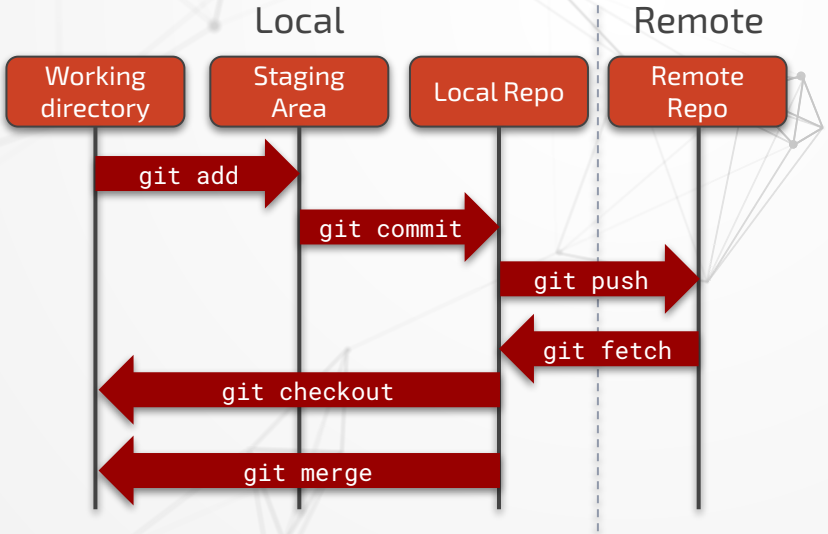
Please don't

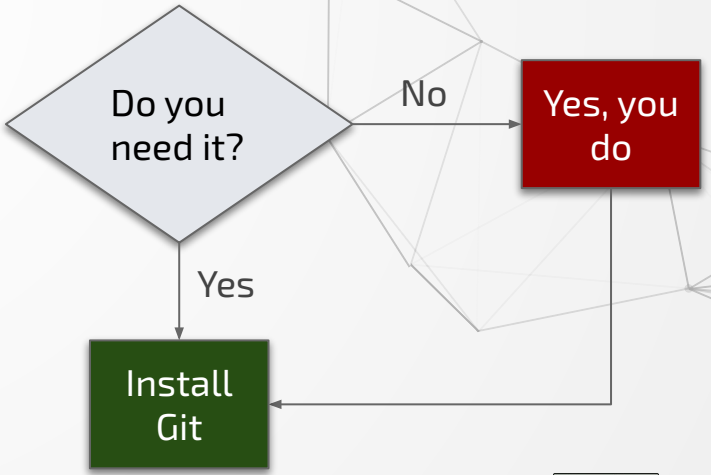
git clone https://github.com/su2code/SU2

SU2 foundation

What are Git and Github?

- Git is a **version control system** to keep track of changes to files over time
- Github is a website that hosts Git **repositories** online
- Each change made to the repository code is a '**commit**', often with a supporting comment about the change that was made

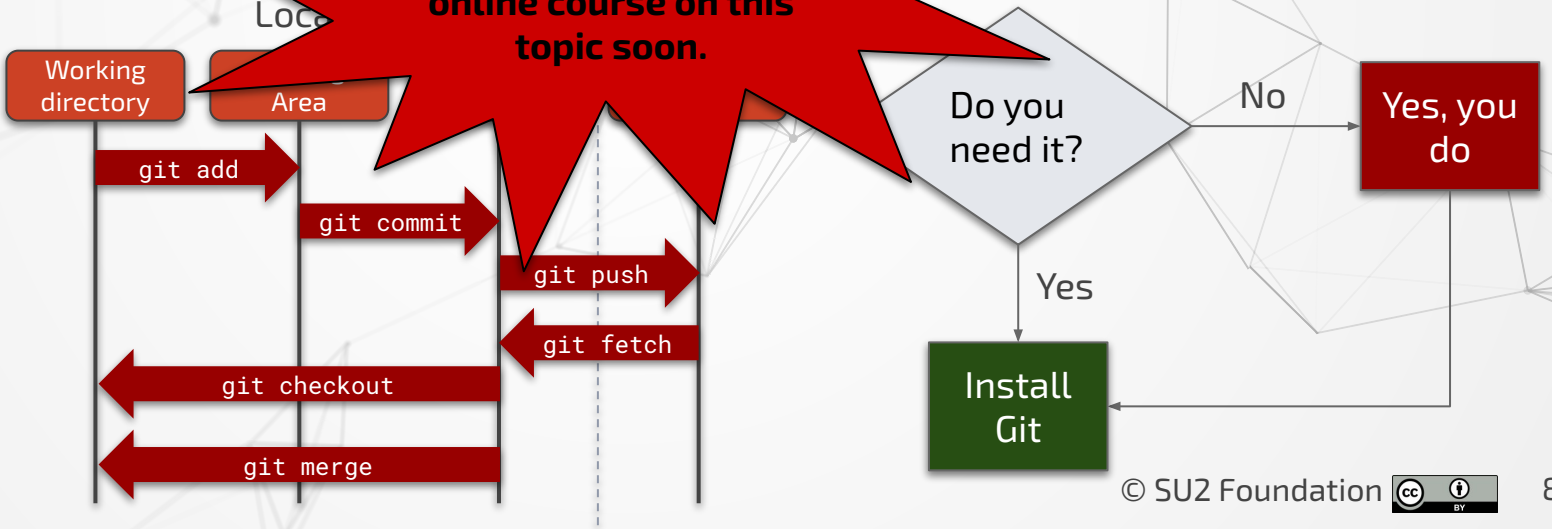




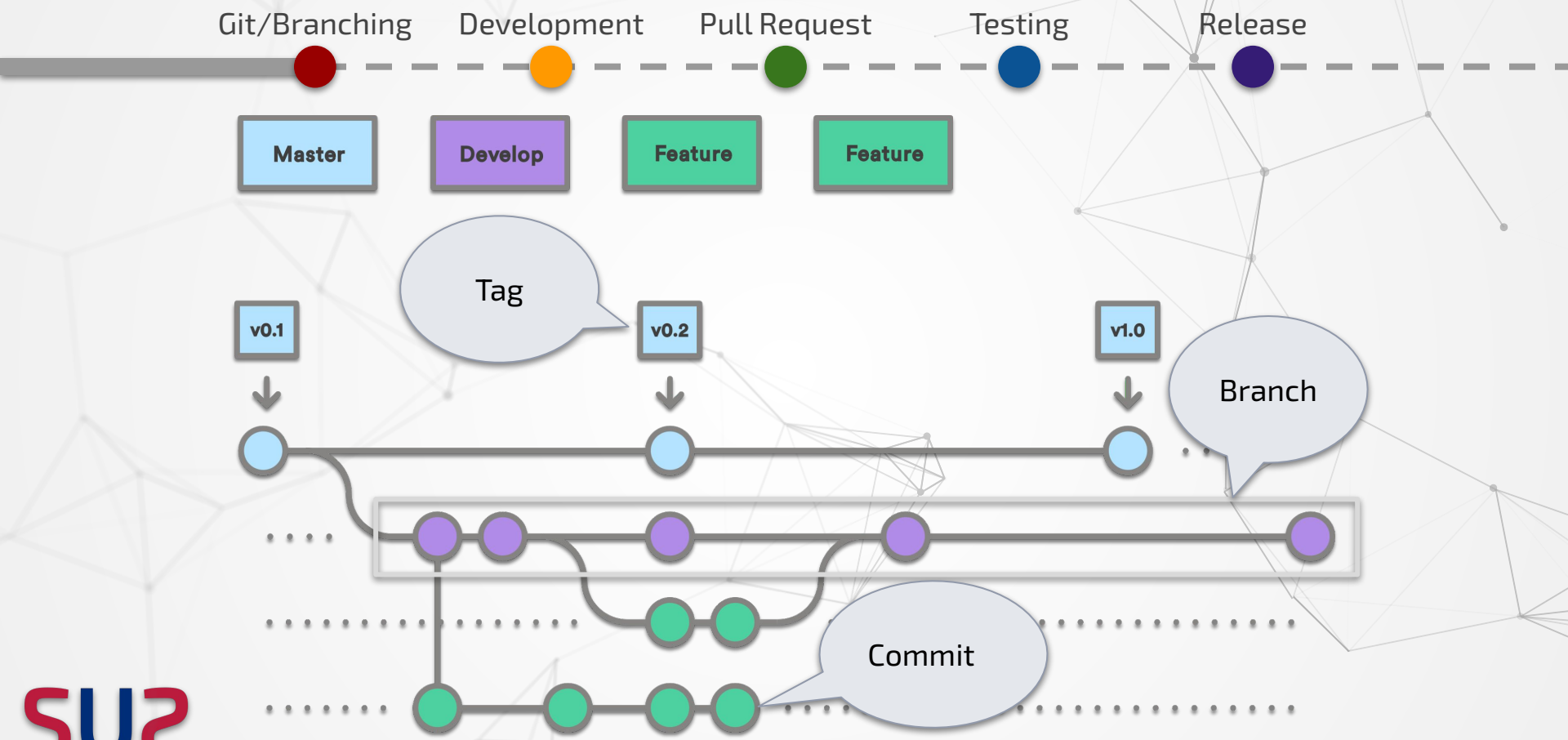
What are Git and Github?

- Git is a **version control system** to keep track of chang to files over time
- Github is a website that host **reposit** onl
- Each change made to the reposit supporting comment about the change that was made

If there is enough interest, we can offer an online course on this topic soon.

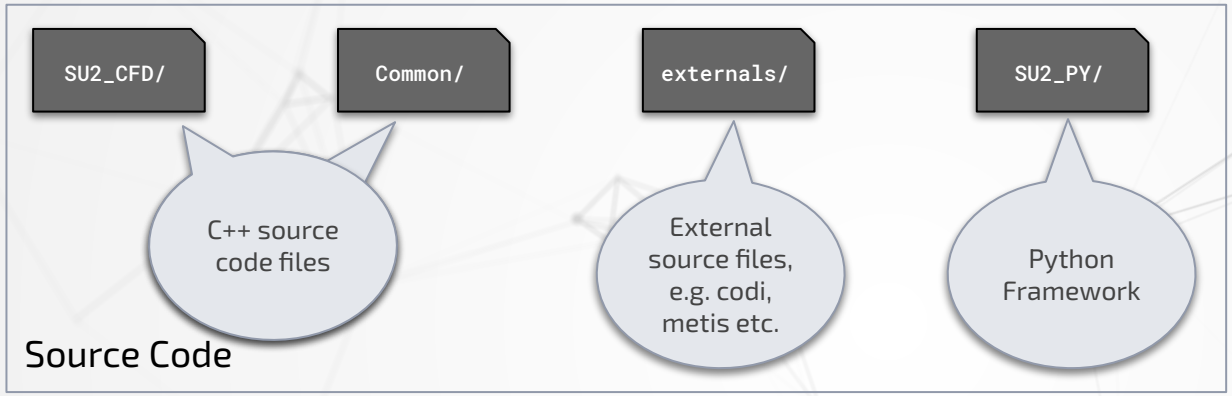


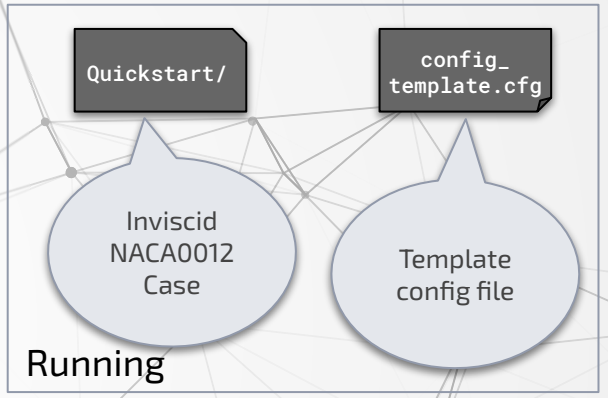
SU2 foundation

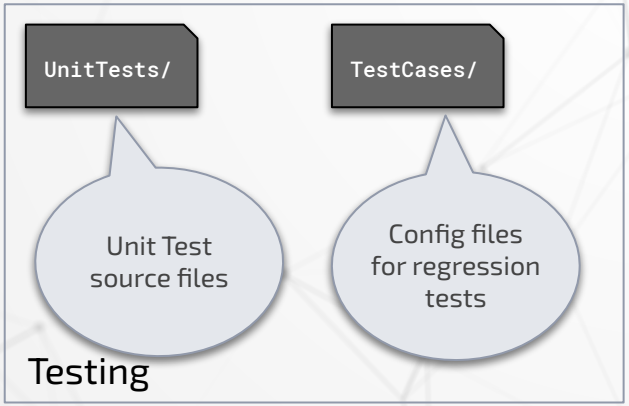


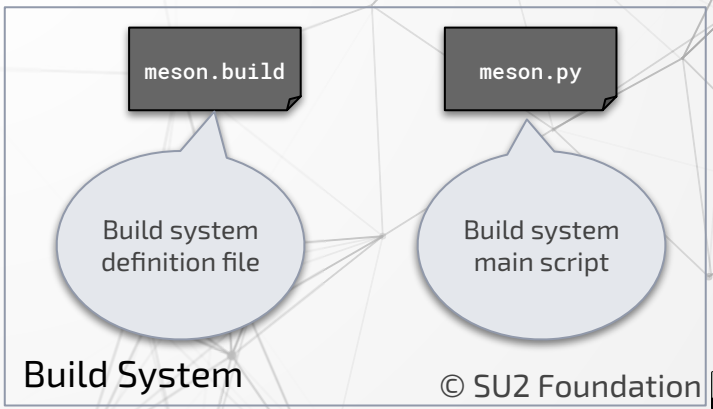
SU2 foundation

Git/Branching Development Pull Request Testing Release



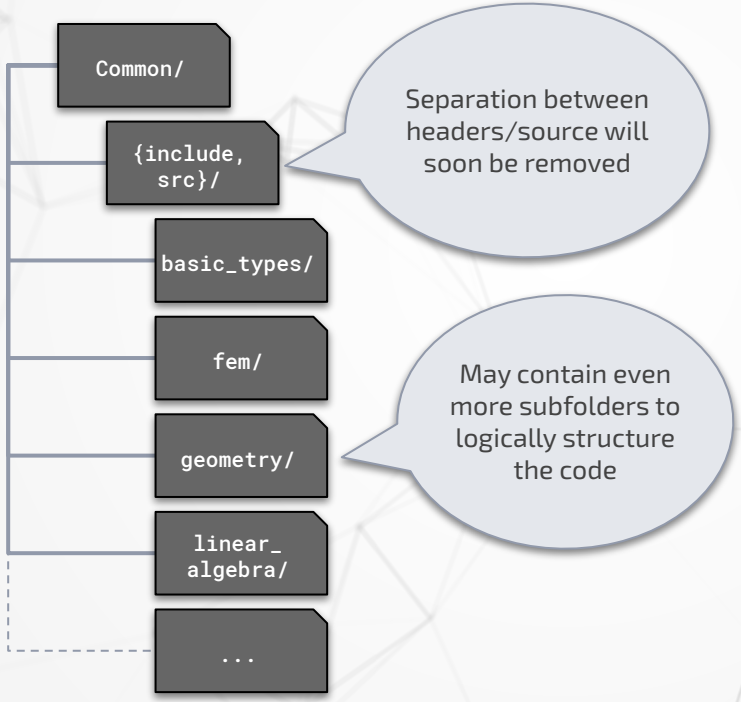


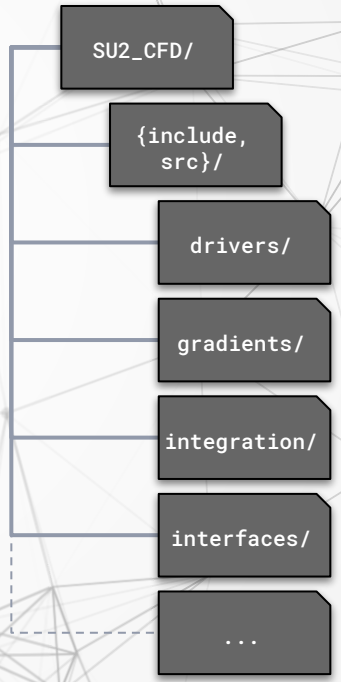




SU2 foundation

Structure of the Source Code





SU2 foundation

Before starting to code - these are the topics you should get familiar with in C++:

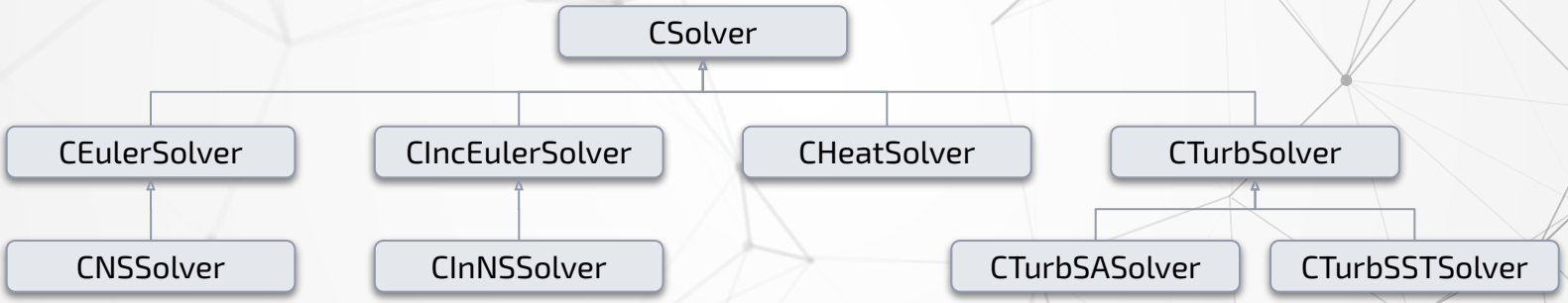
- Classes and polymorphism
- Virtual functions
- Pointers
- *Message Passing Interface (MPI)*

Plus, some C++11 features like **auto**, **range-based loops, lambda functions** can help

SU2 foundation

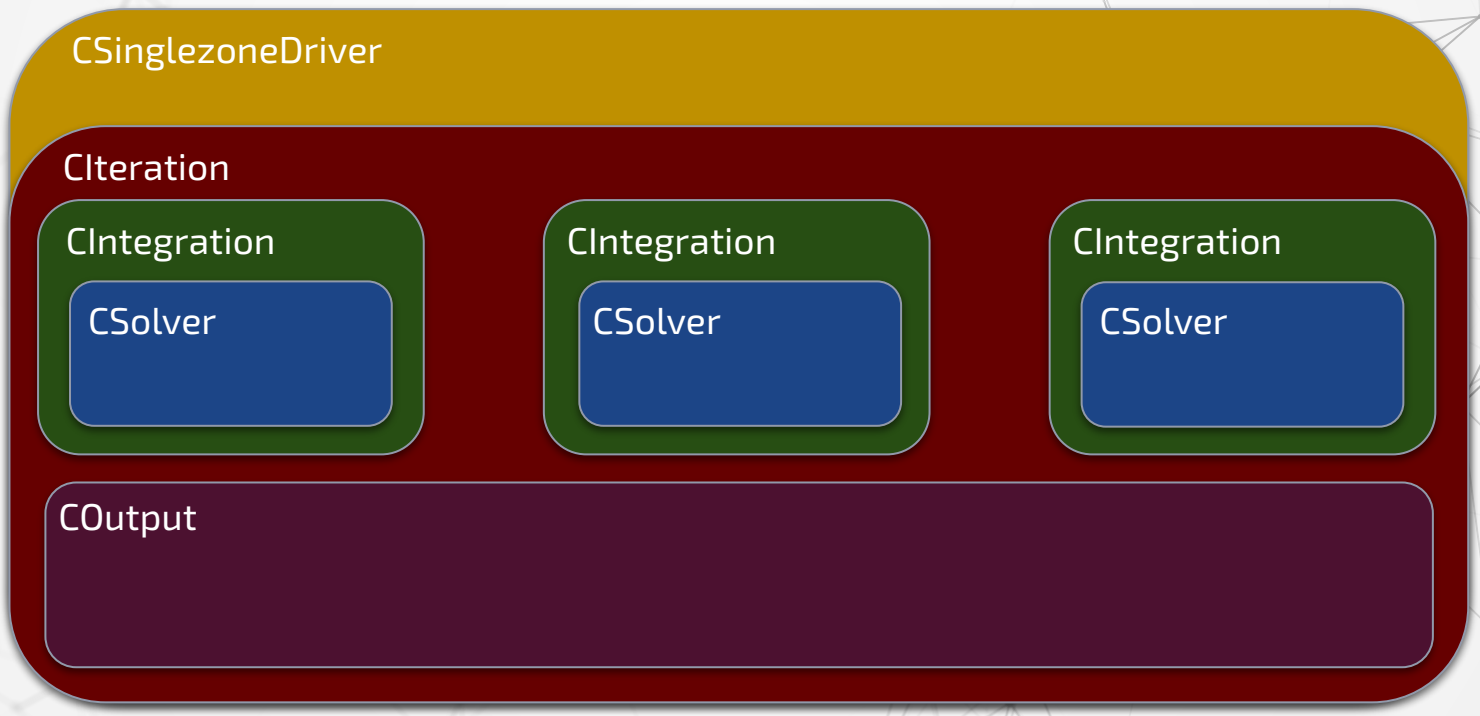
C++ class abstractions encourage code reuse and data encapsulation ensures you can make localized changes easily.

Example:

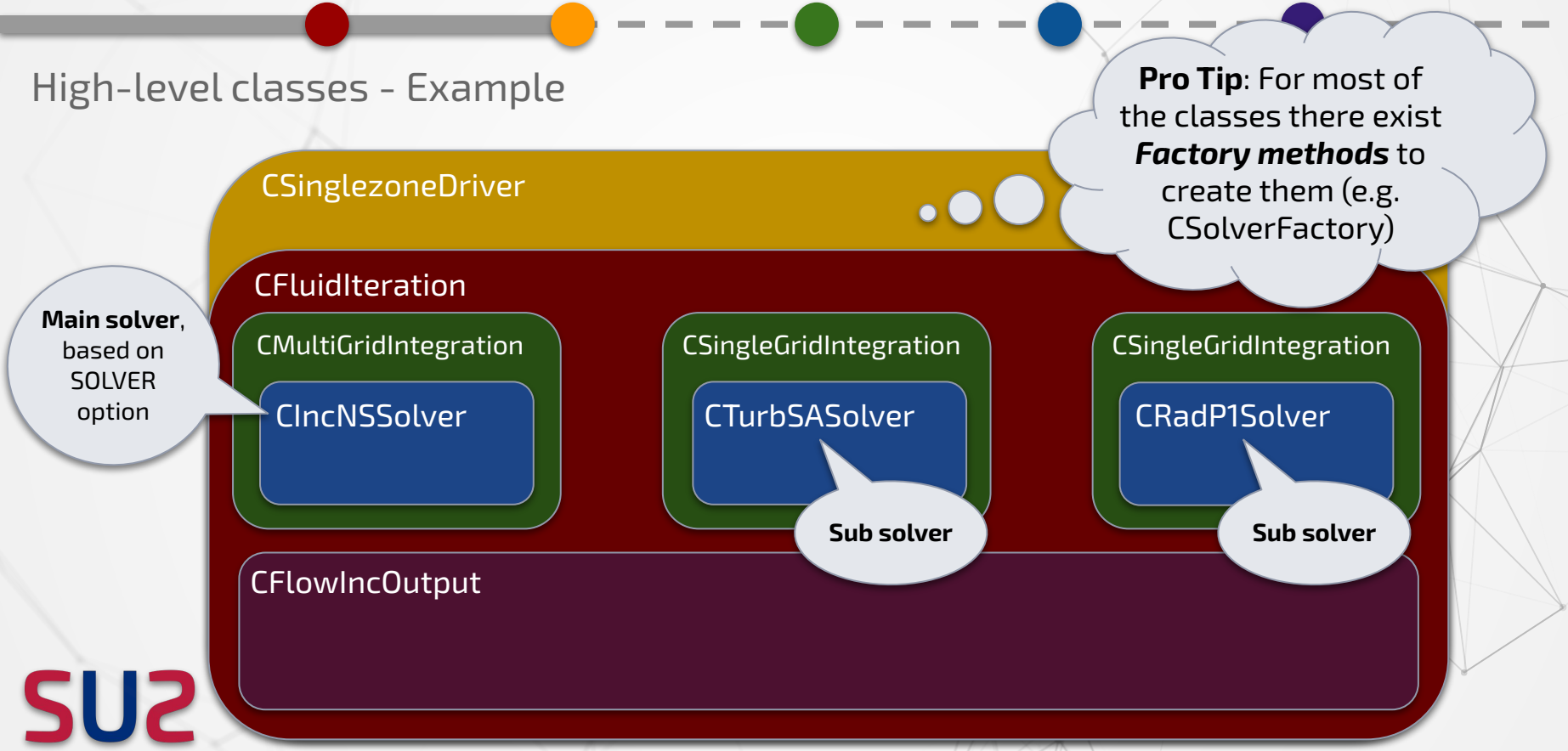


SU2 foundation

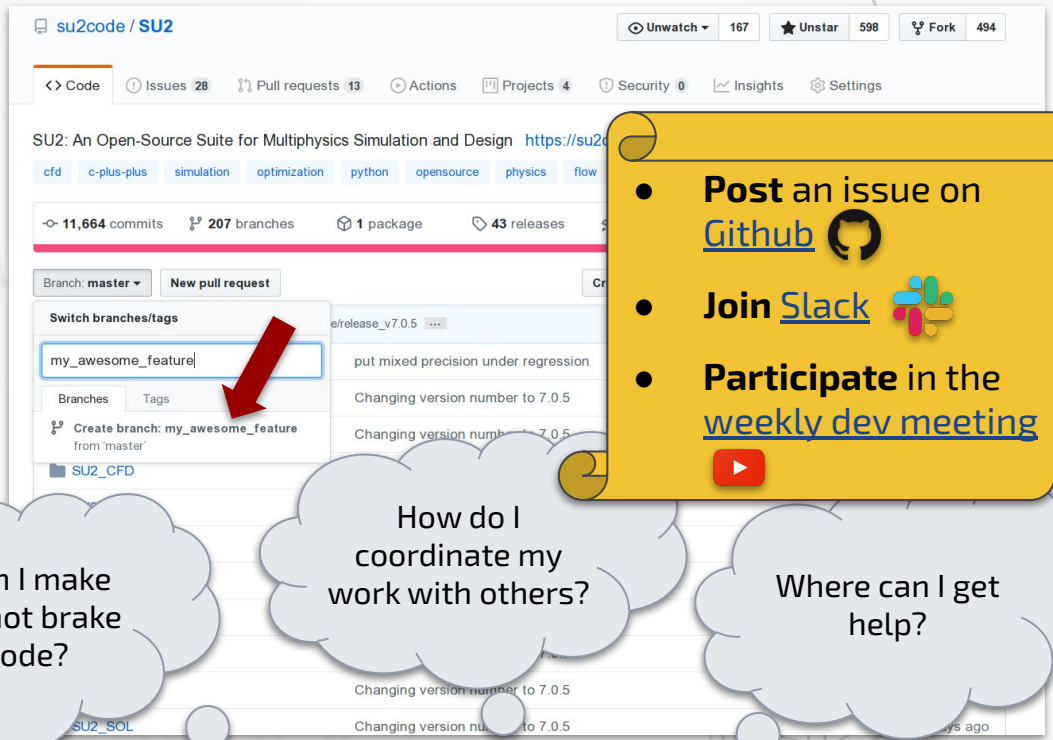
High-level classes



High-level classes - Example



Git/Branching Development Pull Request Testing Release



- **Post** an issue on Github
- **Join** Slack
- **Participate** in the weekly dev meeting

How can I make sure to not brake the code?

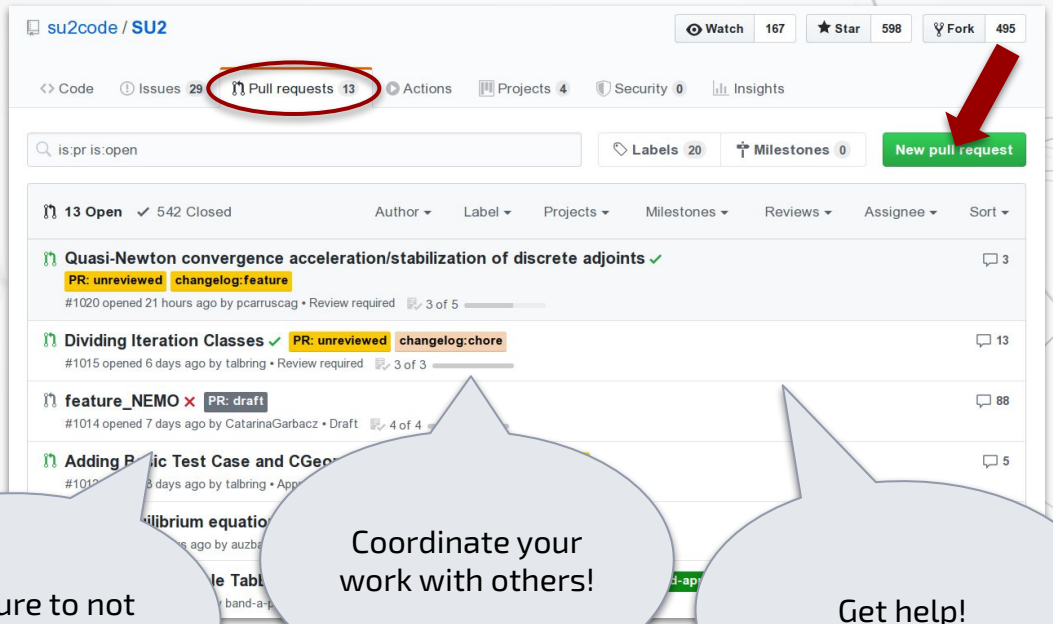
How do I coordinate my work with others?

Where can I get help?

Create your branch!

SU2 foundation

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



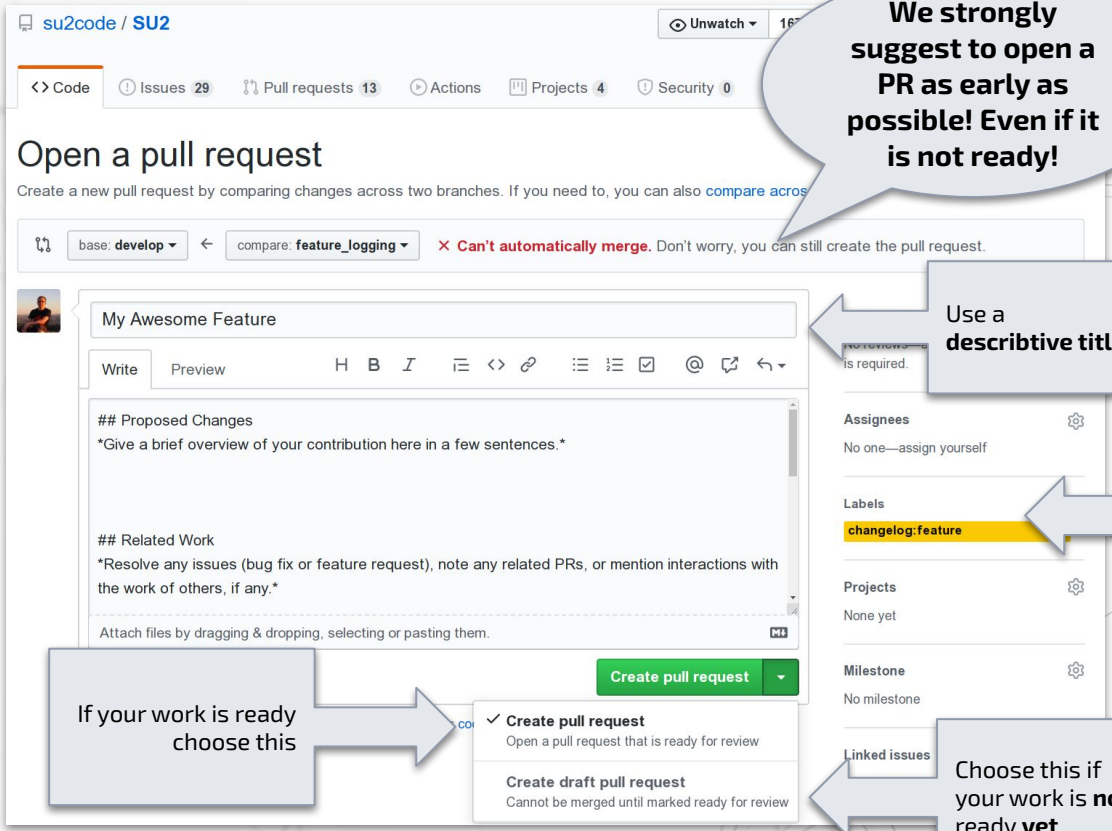
Make sure to not brake the code!

Coordinate your work with others!

Get help!

And create a pull request!

SU2 foundation



We strongly suggest to open a PR as early as possible! Even if it is not ready!

Submit the PR to the **develop** branch

Use a **describtive title**

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

Choose a **label** so that your feature is added to the changes for the release

If your work is ready choose this

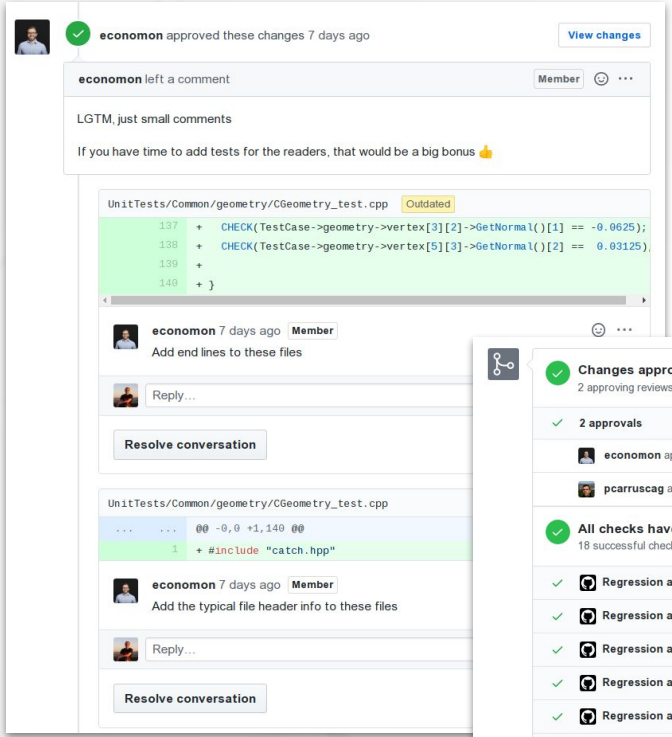
Choose this if your work is **not** ready **yet**

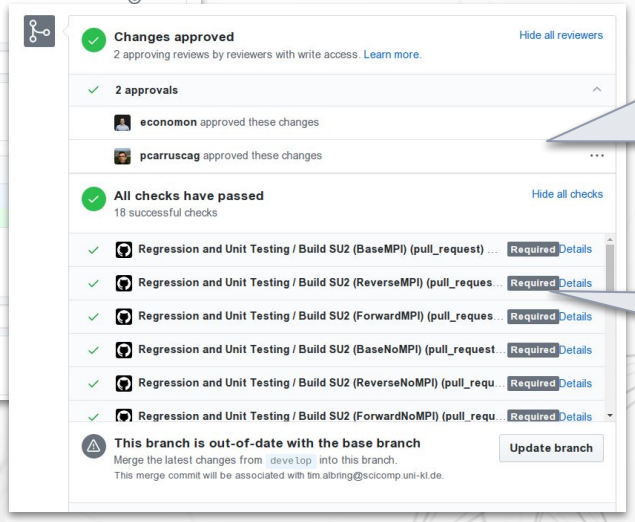
SU2 foundation

Benefits of opening a (draft) Pull Request

- ✓ **Coordination:** Let others know what you are working on
- ✓ **Feedback and help**: Experienced developers can help you and can guide you during the development process
- ✓ **Automated Building and Testing**: You instantly know if something is broken or results have changed in a commit
- ✓ **Faster review process**: Your PR is likely to be merged in sooner

SU2 foundation





Don't see review comments as a criticism of your work. It's to help you and us to maintain a great product!

And trust me, you'll learn a lot in that process.

PR needs approval from **at least two** people from the community

All checks must pass

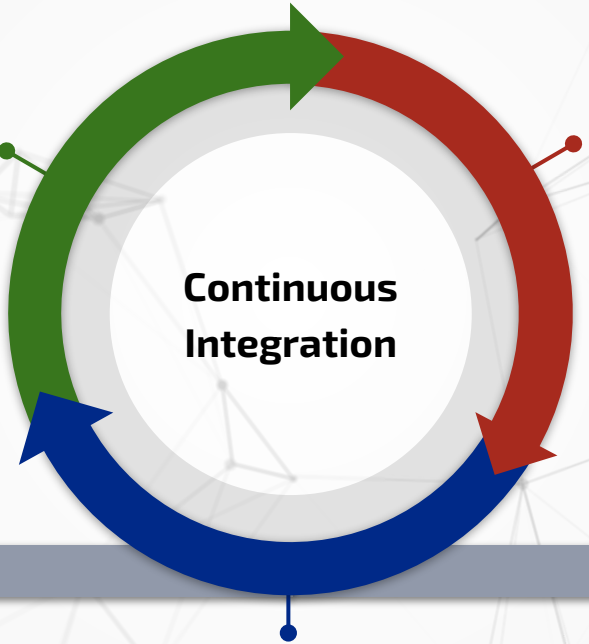
SU2 foundation

Unit Testing

Testing of individual units in the source code More info on that in next talk by C. Pederson

Regression Testing

Comparing results against stored values



Build

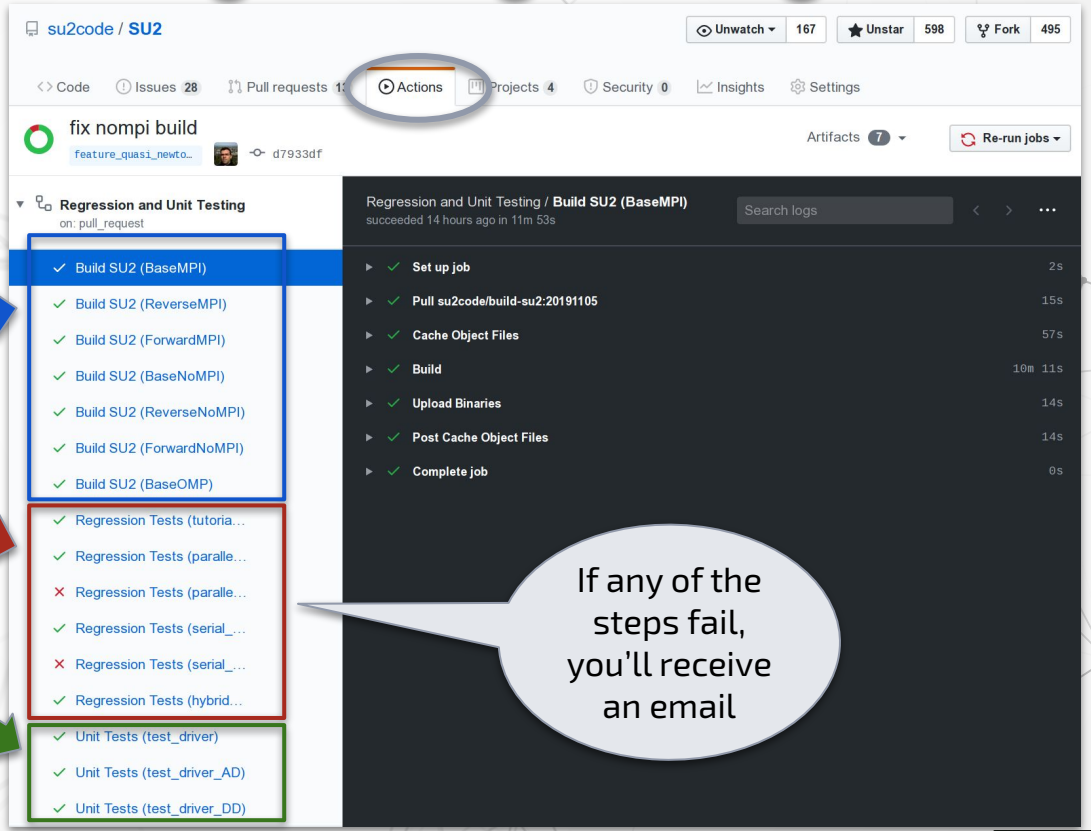
Automated building on Linux, Mac, Windows

SU2 foundation

GitHub Actions

On **every push** to a **branch** opened as a (draft) **Pull Request**:

1. The code is built
2. Regression tests run
3. Unit tests are executed



SU2 foundation

Add a regression test to protect your features

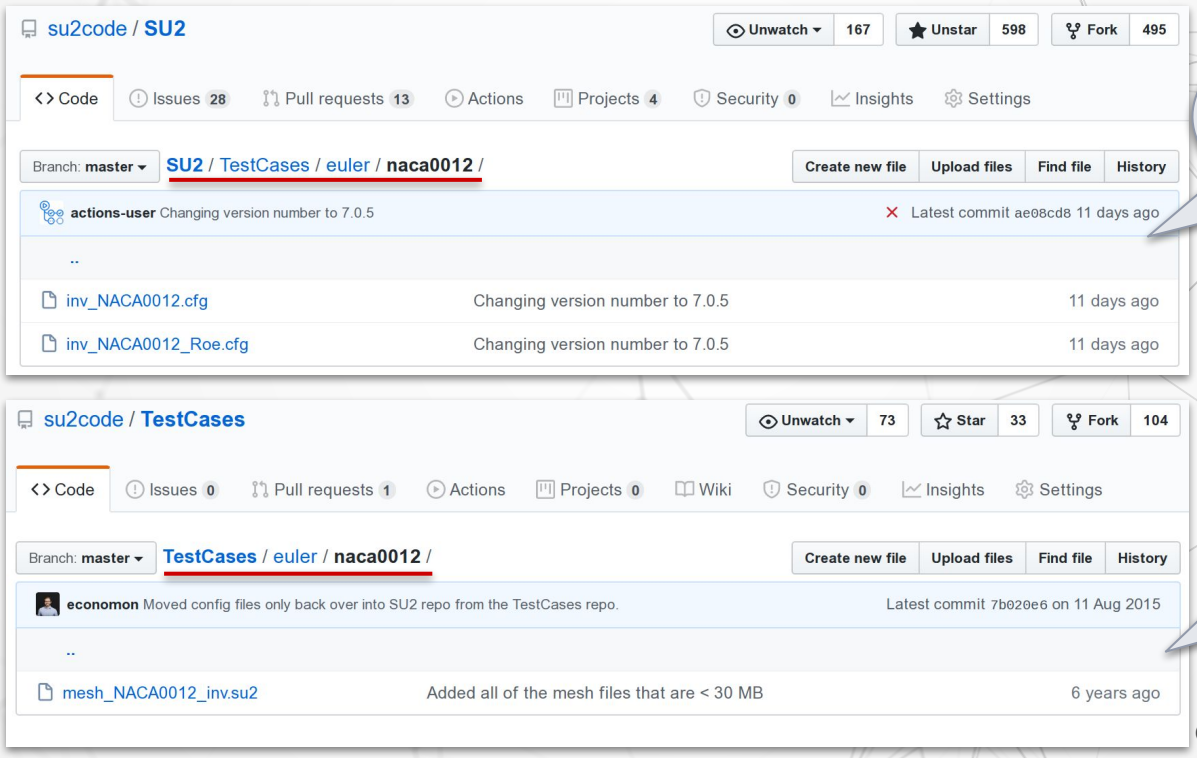
```
# NACA0012
naca0012            = TestCase('naca0012')
naca0012.cfg_dir    = "euler/naca0012"
naca0012.cfg_file   = "inv_NACA0012_Roe.cfg"
naca0012.test_iter  = 20
naca0012.test_vals  = [-4.047448, -3.538057, 0.338691, 0.023131] #last 4 columns
naca0012.su2_exec   = "SU2_CFD"
naca0012.timeout    =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

If the computed values after your commit don't match these values, the test will fail

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

SU2 foundation

2. Put the config file and any supporting data in the corresponding locations.

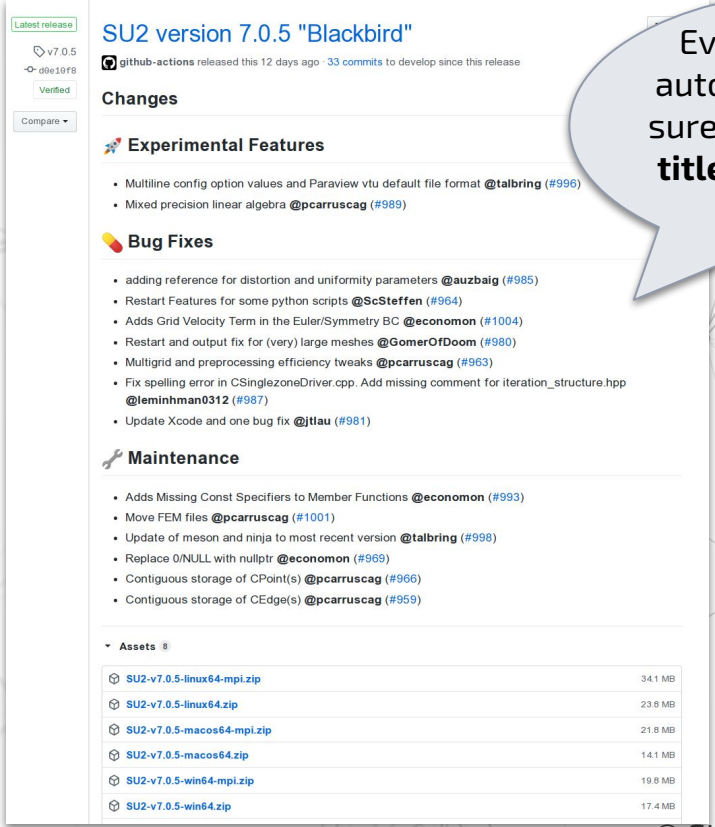


Lighter weight, more frequently updated files go in code repo.

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

Frequently a new version **with your contribution** will be released, which means

1. develop branch is moved to master
2. A new tag is created
3. Binaries are created
4. Release notes are published
5. New version is advertised



Latest release

v7.0.5

d0e10f8

Verified

Compare

SU2 version 7.0.5 "Blackbird"

github-actions released this 12 days ago · 33 commits to develop since this release

Changes

Experimental Features

- Multiline config option values and Paraview vtu default file format **@talbring** (#996)
- Mixed precision linear algebra **@pcarruscag** (#989)

Bug Fixes

- adding reference for distortion and uniformity parameters **@auzbaig** (#985)
- Restart Features for some python scripts **@ScSteffen** (#964)
- Adds Grid Velocity Term in the Euler/Symmetry BC **@economon** (#1004)
- Restart and output fix for (very) large meshes **@GomerOfDoom** (#980)
- Multigrid and preprocessing efficiency tweaks **@pcarruscag** (#963)
- Fix spelling error in CSinglezoneDriver.cpp. Add missing comment for iteration_structure.hpp **@leminhman0312** (#987)
- Update Xcode and one bug fix **@jtlau** (#981)

Maintenance

- Adds Missing Const Specifiers to Member Functions **@economon** (#993)
- Move FEM files **@pcarruscag** (#1001)
- Update of meson and ninja to most recent version **@talbring** (#998)
- Replace 0/NULL with nullptr **@economon** (#969)
- Contiguous storage of CPoint(s) **@pcarruscag** (#966)
- Contiguous storage of CEdge(s) **@pcarruscag** (#959)

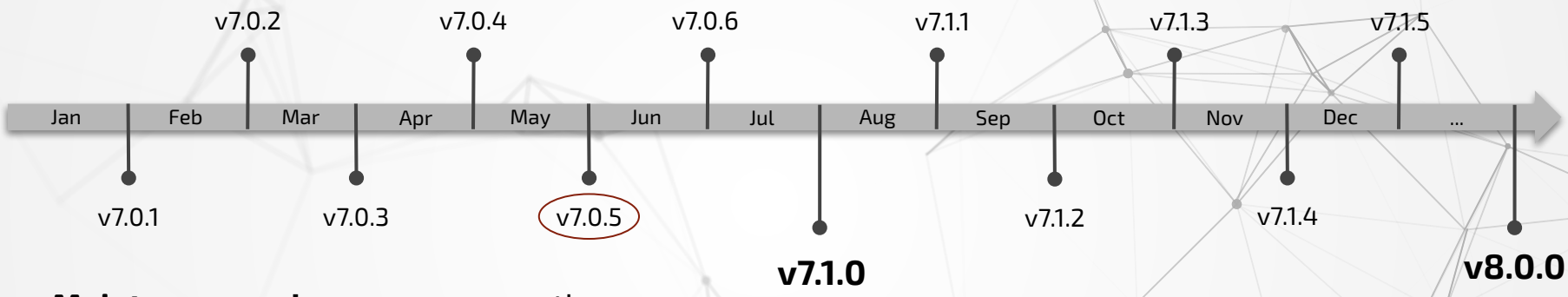
Assets 8

SU2-v7.0.5-linux64-mpi.zip	34.1 MB
SU2-v7.0.5-linux64.zip	23.8 MB
SU2-v7.0.5-macos64-mpi.zip	21.8 MB
SU2-v7.0.5-macos64.zip	14.1 MB
SU2-v7.0.5-win64-mpi.zip	19.8 MB
SU2-v7.0.5-win64.zip	17.4 MB

Everything happens automatically, so make sure to choose a **proper title** and **label** for your PR

SU2 foundation

Release Schedule



Maintenance releases every month

Minor releases every 6 months (may vary)

Major releases every x year(s) (depends)

Goes along with updates in **documentation** and may break **backward compatibility** of config files

SU2 foundation

SU2 Development Survival Guide

1. **Clone main repository**: `$ git clone https://github.com/su2code/SU2.git`
2. **Create new feature branch** for your development work. Work on this branch in the repo.
3. **Open a draft PR** so that other developers immediately see what you are working on. Also you can leverage the testing system! Note that any new commits will appear right there in the open PR and will kick-off the tests again.
4. **Work on your feature!** Please mind white space issues, compiler warnings, and match the SU2 style -> Make use of clang-format.
5. **Merge in the remote develop branch into your branch as frequent as possible.** This ensures that, when the time comes and your feature is finished, it will be easy to merge your contribution into develop, as you will have solved any conflicts on your side.
6. Once you feel your feature is finished, **mark your PR as "Ready to Review"**.
7. **Get reviews and engage with the community concerning your contribution**. Fix problems in your branch or address any feedback on the message boards.
8. Once the reviewers approve and all tests 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, right?

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.

SU2 foundation