The Julia CI ecosystem

Spoiling scientific developers for good

Michael Schlottke-Lakemper

Applied and Computational Mathematics, RWTH Aachen University
High-Performance Computing Center Stuttgart (HLRS), University of Stuttgart

deRSE24, Würzburg, Germany, 6th March 2024

Back in the days (10 years ago)

- Numerical simulation framework for CFD (∼250k SLOC, C++)
- License: no (closed source)
- Version control: Subversion (hosted locally)
- Continuous testing: no
- Documentation: somewhat
- Code reviews/commit guidelines: no

Back in the days (10 years ago)

- Numerical simulation framework for CFD (∼250k SLOC, C++)
- License: no (closed source)
- Version control: Subversion (hosted locally)
- Continuous testing: no
- Documentation: somewhat
- Code reviews/commit guidelines: no

Commit procedure for `trunk`

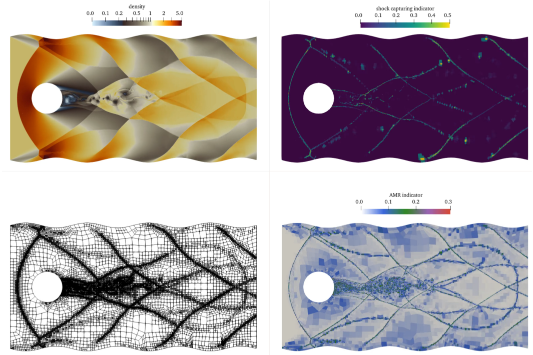
1) Manually run testcases against `trunk` branch.
2) Manually run testcases against branch.
2) Merge if no additional tests fail.

Fast forward to 2024

- Numerical simulation framework for CFD (∼50k SLOC, Julia)
- License: MIT
- Version control: Git (hosted on GitHub)
- Continuous testing: >20 jobs (Linux/macOS/Windows, serial/parallel, ...)
- Documentation: yes
- Code reviews/commit guidelines: yes

Some CI workflows in Trixi.jl

1. Testing and code coverage
2. Documentation
3. Compatibility bounds
4. Code formatting
5. Spelling
6. Release process
7. Review checklist
8. Downstream tests

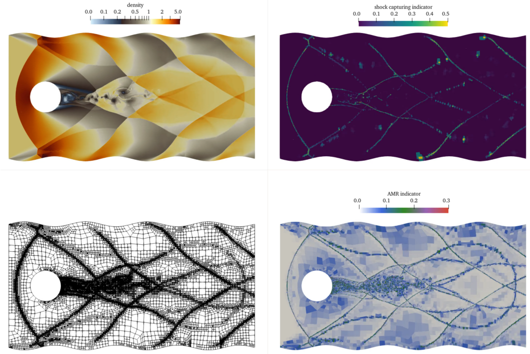


`https://github.com/trixi-framework/Trixi.jl`

Some CI workflows in Trixi.jl

1. Testing and code coverage
2. Documentation
3. Compatibility bounds
4. Code formatting
5. Spelling
6. Release process
7. Review checklist
8. Downstream tests

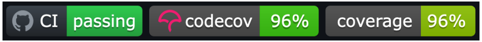
→ widely used best practice workflows



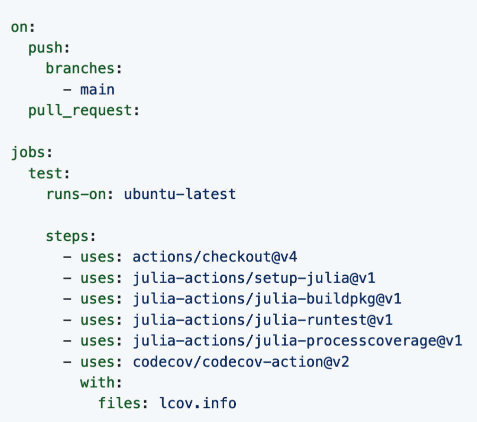
https://github.com/trixi-framework/Trixi.jl

Run tests and verify code coverage

- Julia has built-in testing module
- Run tests of Julia package manager (also offline)
- GitHub Actions scripts facilitate easy CI setup



Badges in `README.md`

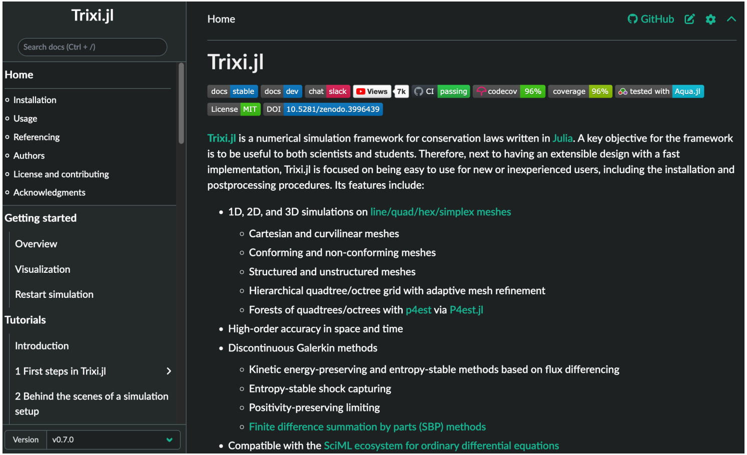


```yaml
on:
  push:
    branches:
      - main
  pull_request:

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4
      - uses: julia-actions/setup-julia@v1
      - uses: julia-actions/julia-buildpkg@v1
      - uses: julia-actions/julia-runtest@v1
      - uses: julia-actions/julia-processcoverage@v1
      - uses: codecov/codecov-action@v2
        with:
          files: lcov.info
```

Minimum workflow file for running Julia tests

Build documentation

- Standard in Julia: Documenter.jl
- Markdown-based, with in-source docs via docstrings
- Versioned docs for releases



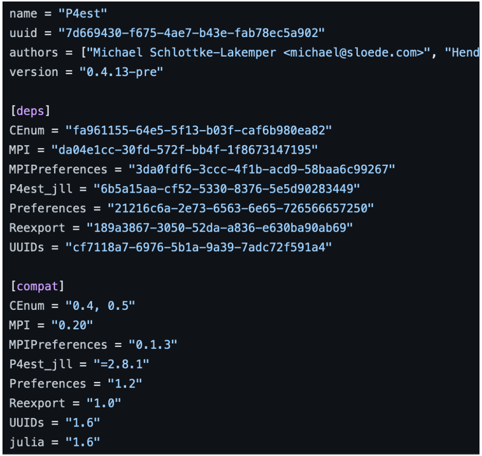
Trixi.jl

Trixi.jl is a numerical simulation framework for conservation laws written in Julia. A key objective for the framework is to be useful to both scientists and students. Therefore, next to having an extensible design with a fast implementation, Trixi.jl is focused on being easy to use for new or inexperienced users, including the installation and postprocessing procedures. Its features include:

- 1D, 2D, and 3D simulations on line/quad/hex/simplex meshes
  - Cartesian and curvilinear meshes
  - Conforming and non-conforming meshes
  - Structured and unstructured meshes
  - Hierarchical quadtree/octree grid with adaptive mesh refinement
  - Forests of quadtrees/octrees with p4est via P4est.jl
- High-order accuracy in space and time
- Discontinuous Galerkin methods
  - Kinetic energy-preserving and entropy-stable methods based on flux differencing
  - Entropy-stable shock capturing
  - Positivity-preserving limiting
  - Finite difference summation by parts (SBP) methods
- Compatible with the SciML ecosystem for ordinary differential equations

Update compatibility bounds

- Julia package manager uses semantic versioning (`v1.2.3`)
- `CompatHelper.yml`: increase upper bounds on new upstream releases
- `Downgrade.yml`: increase lower bounds to minimum supported version



```
name = "P4est"
uuid = "7d669430-f675-4ae7-b43e-fab78ec5a902"
authors = ["Michael Schlottke-Lakemper <michael@sloede.com>", "Hend
version = "0.4.13-pre"

[deps]
CEnum = "fa961155-64e5-5f13-b03f-caf6b980ea82"
MPI = "da04e1cc-30fd-572f-bb4f-1f8673147195"
MPIPreferences = "3da0fdf6-3ccc-4f1b-acd9-58baa6c99267"
P4est_jll = "6b5a15aa-cf52-5330-8376-5e5d90283449"
Preferences = "21216c6a-2e73-6563-6e65-726566657250"
Reexport = "189a3867-3050-52da-a836-e630ba90ab69"
UUIDs = "cf7118a7-6976-5b1a-9a39-7adc72f591a4"

[compat]
CEnum = "0.4, 0.5"
MPI = "0.20"
MPIPreferences = "0.1.3"
P4est_jll = "=2.8.1"
Preferences = "1.2"
Reexport = "1.0"
UUIDs = "1.6"
julia = "1.6"
```

`Project.toml` for P4est.jl

Ensure code formatting and spelling

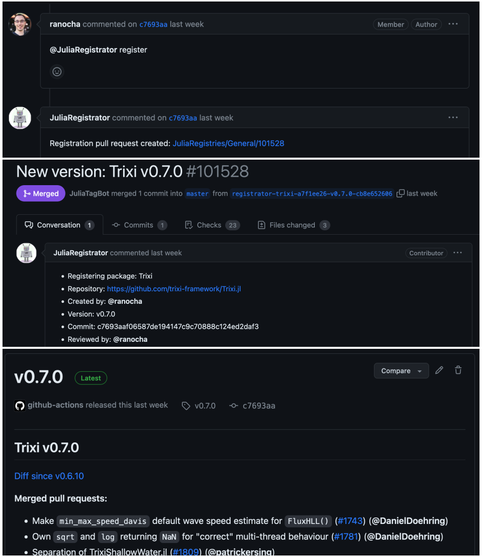
- Automatic code formatting via JuliaFormatter.jl (modelled on `clang-format`)
- Automatic spell checking using `crate-ci/typos` GitHub Action
- Reduces effort for developers and reviewers

Automated release process

- Automated package registration process

Release process

1. Update version in `Project.toml`
2. Trigger registration via comment
3. Auto-creation of PR to registry
4. Upon registry update: tag GitHub release with notes



ranocha commented on c7693aa last week

@JuliaRegistrator register

JuliaRegistrator commented on c7693aa last week

Registration pull request created: JuliaRegistries/General/101528

New version: Trixi v0.7.0 #101528

Merged **JuliaTagBot** merged 1 commit into `master` from `registrator-trixi-a7f1ee26-v0.7.0-cb8e652606` last week

JuliaRegistrator commented last week

- Registering package: Trixi
- Repository: https://github.com/trixi-framework/Trixi.jl
- Created by: **@ranocha**
- Version: v0.7.0
- Commit: c7693aaf06587de194147c9c70888c124ed2daf3
- Reviewed by: **@ranocha**

v0.7.0

github-actions released this last week v0.7.0 c7693aa

Trixi v0.7.0

Diff since v0.6.10

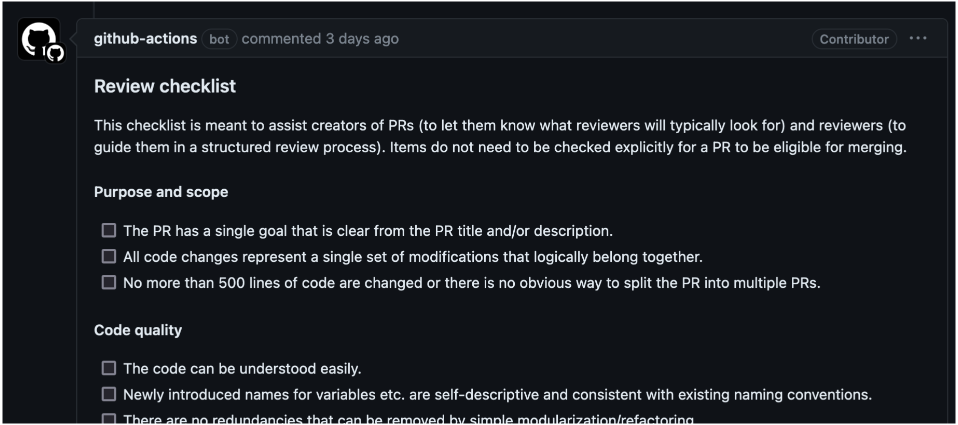
Merged pull requests:

- Make `min_max_speed_davis` default wave speed estimate for `FluxHLL()` (#1743) (**@DanielDoehring**)
- Own `sqrt` and `log` returning `NaN` for "correct" multi-thread behaviour (#1781) (**@DanielDoehring**)
- Separation of TrixiShallowWater.jl (#1809) (**@patrickersing**)

Registrator comment (top),
registry PR (center),
release notes (bottom)

Add review checklist

- Compiled by entire team (junior and senior members)
- Helps both reviewers and developers



Review checklist

This checklist is meant to assist creators of PRs (to let them know what reviewers will typically look for) and reviewers (to guide them in a structured review process). Items do not need to be checked explicitly for a PR to be eligible for merging.

Purpose and scope

- □ The PR has a single goal that is clear from the PR title and/or description.
- □ All code changes represent a single set of modifications that logically belong together.
- □ No more than 500 lines of code are changed or there is no obvious way to split the PR into multiple PRs.

Code quality

- □ The code can be understood easily.
- □ Newly introduced names for variables etc. are self-descriptive and consistent with existing naming conventions.
- □ There are no redundancies that can be removed by simple modularization/refactoring

Run downstream tests

- Run reduced testset for selected downstream packages
- Needs explicit support from downstream software
- Brings flexibility for code and repository management

Summary

- Julia programming language has elaborate CI ecosystem
- Great support for GitHub, limited support for GitLab
- Many CI workflows expected by users

Summary

- Julia programming language has elaborate CI ecosystem
- Great support for GitHub, limited support for GitLab
- Many CI workflows expected by users

Ease-of-use + wide-spread utilization of CI = spoiled users/developers