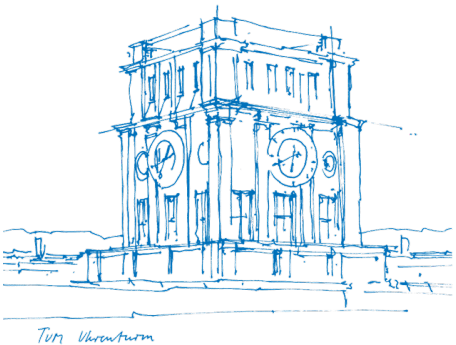
Continuous integration and documentation automation in preCICE

17th OpenFOAM Workshop
Cambridge, UK

Gerasimos Chourdakis
Technical University of Munich

July 11, 2022



A quality dashboard for your project

README.md

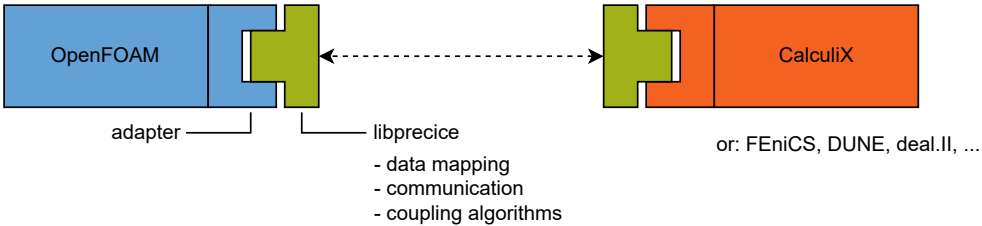
preCICE

Communication

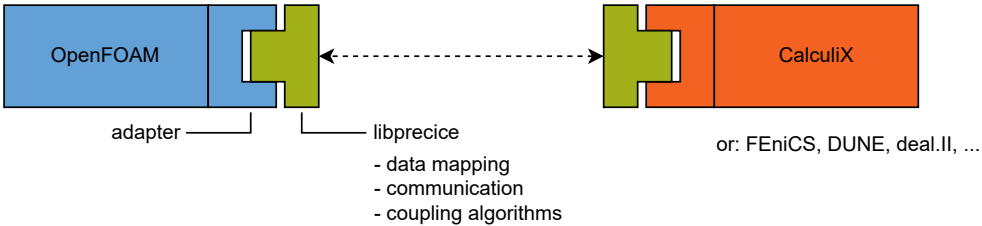
Project Status

Project Quality

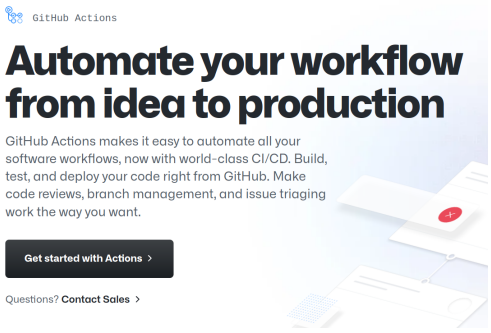
preCICE and its OpenFOAM adapter

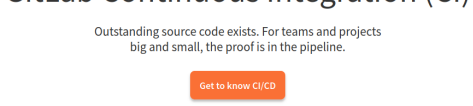


preCICE and its OpenFOAM adapter

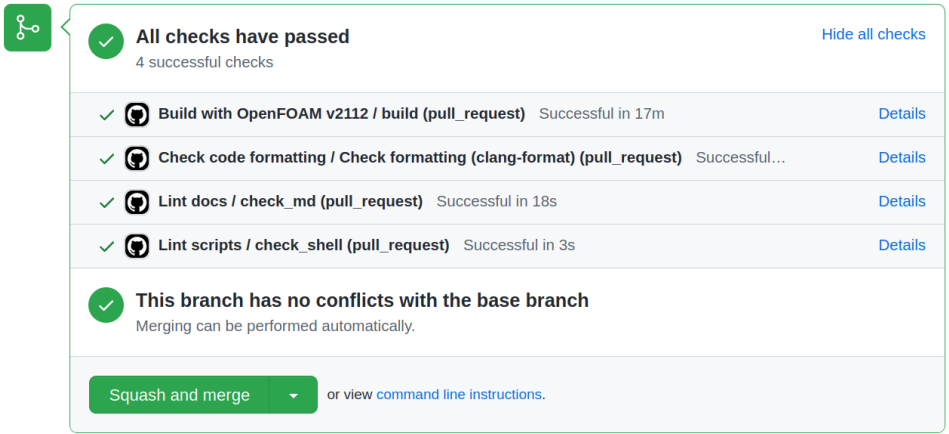


Continuous Integration tools

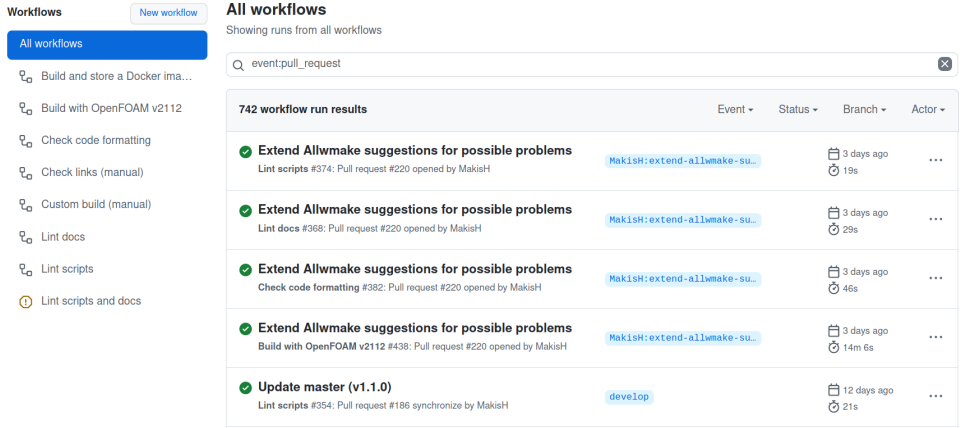




Safely contributing to the OpenFOAM adapter



GitHub Actions for the OpenFOAM adapter



Automatic format check with clang-format

13 lines (13 sloc) | 369 Bytes

```yaml
name: Check code formatting
on: [push, pull_request]
jobs:
  formatting-check:
    name: Check formatting (clang-format)
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Run clang-format style check for C/C++ programs.
      uses: jidicula/clang-format-action@main
      with:
        clang-format-version: '11'
        check-path: '.'
```

Automatic format check with clang-format

13 lines (13 sloc) | 369 Bytes

```yaml
name: Check code formatting
on: [push, pull_request]
jobs:
  formatting-check:
    name: Check formatting (clang-format)
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Run clang-format style check for C/C++ programs.
      uses: jidicula/clang-format-action@main
      with:
        clang-format-version: '11'
        check-path: '.'
```

By the way: we need an official style file

Development › openfoam › Issues › #1634

Open Created 2 years ago by Gerasimos Chourdakis Close issue

Style formatting config needed (e.g. clang-format)

Functionality to add/problem to solve

Add a clang-format or similar system config file in the code repository, so that contributions to the repository or derived projects can follow the Coding Style Guide.

Clang-format is an established formatting tool and integration is already provided in many editors and IDEs, including Emacs, Vim, and CLion.

Target audience

- Regular developers of OpenFOAM
- External contributors to OpenFOAM
- Contributors to third-party OpenFOAM projects (e.g. function objects, such as the preCICE OpenFOAM adapter)

Proposal

1. Create a `.clang-format` file, e.g. in the root directory of the repository, or any other directory preferred for developers' tools.
   - For example, with `clang-format -style=llvm -dump-config > .clang-format`
2. Define the preferred rules following the Clang-Format Style Options. Beware that the rules may depend on the Clang-Format version.

Discover issues in shell scripts with shellcheck

```
~$ shellcheck /usr/lib/openfoam/openfoam2206/bin/tools/RunFunctions

In /usr/lib/openfoam/openfoam2206/bin/tools/RunFunctions line 1:
#----------------------------------*- sh -*-------------------------------------
^-- SC2148 (error): Tips depend on target shell and yours is unknown. Add a shebang or a 'shell' directive.


In /usr/lib/openfoam/openfoam2206/bin/tools/RunFunctions line 119:
        value="$(foamDictionary -value $@ 2>/dev/null)" || return 2
                                ^-- SC2068 (error): Double quote array expansions to avoid re-splitting elements.


In /usr/lib/openfoam/openfoam2206/bin/tools/RunFunctions line 160:
    if [ -n "nFaces" ]
           ^----^ SC2157 (error): Argument to -n is always true due to literal strings.


In /usr/lib/openfoam/openfoam2206/bin/tools/RunFunctions line 184:
    set -- $(foamDictionary -entry numberOfSubdomains -value "$dict" 2>/dev/null)
         ^-- SC2046 (warning): Quote this to prevent word splitting.

                                        • • •

In /usr/lib/openfoam/openfoam2206/bin/tools/RunFunctions line 484:
           for file in $(grep -l "#include" * 2>/dev/null)
                       ^-- SC2013 (info): To read lines rather than words, pipe/redirect to a 'while read' loop.
                                     ^-- SC2035 (info): Use ./*glob* or -- *glob* so names with dashes won't become options.

For more information:
  https://www.shellcheck.net/wiki/SC2068 -- Double quote array expansions to ...
  https://www.shellcheck.net/wiki/SC2145 -- Argument mixes string and array. ...
  https://www.shellcheck.net/wiki/SC2148 -- Tips depend on target shell and y...
```

Run shellcheck on GitHub Actions

10 lines (10 sloc) | 265 Bytes

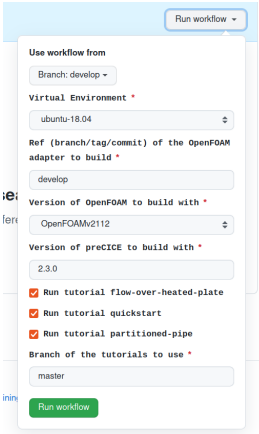
```yaml
name: Lint scripts
on: [push, pull_request]
jobs:
  check_shell:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repository
        uses: actions/checkout@v2
      - name: Lint shell scripts (shellcheck)
        uses: ludeeus/action-shellcheck@master
```

More Actions: check URLs

14 lines (14 sloc) | 445 Bytes

```yaml
name: Check links (manual)
on: workflow_dispatch
jobs:
  check_links:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repository
        uses: actions/checkout@v2
      - name: Check links in markdown files (markdown-link-check)
        uses: gaurav-nelson/github-action-markdown-link-check@v1
        with:
          use-quiet-mode: 'yes'
          use-verbose-mode: 'no'
          config-file: '.markdown-link-check-config.json'
```

GitHub Actions: Manual triggering

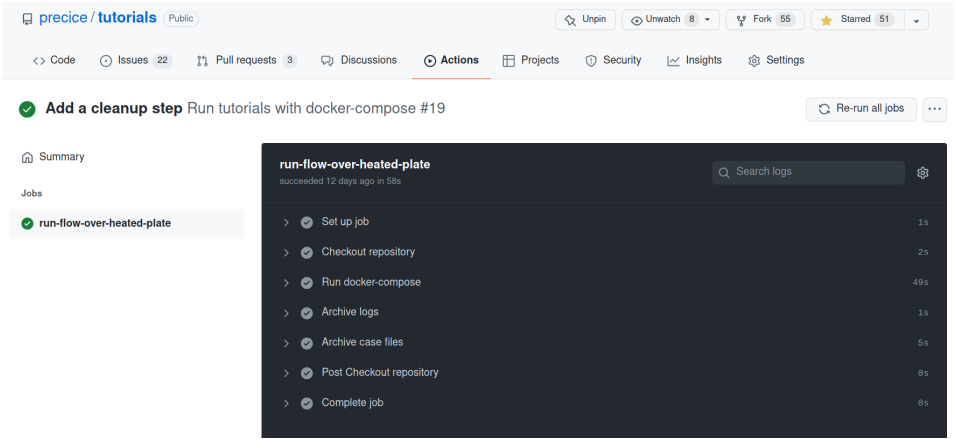


GitHub Actions: Manual triggering (config)

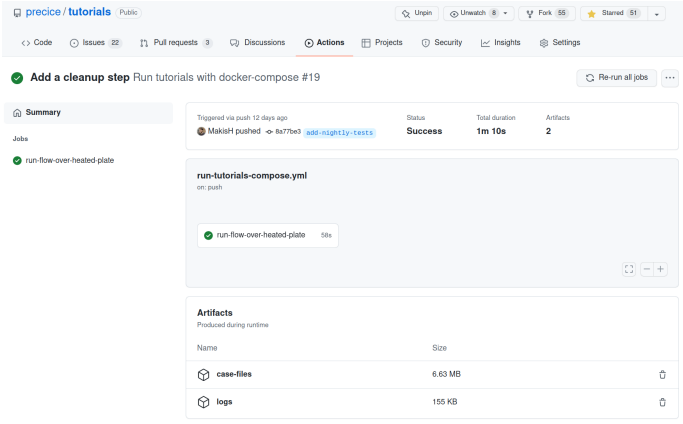
```yaml
# File .github/workflows/build-custom.yml
name: Custom build (manual)
on:
  workflow_dispatch:
    inputs:
      # ...
      versionOpenFOAM:
        type: choice
        options:
          - OpenFOAMv2206
          - OpenFOAM9

jobs:
  build:
    runs-on: ubuntu:latest
    steps:
    - name: Run something
      run: ./run.sh
```

Workflow logs

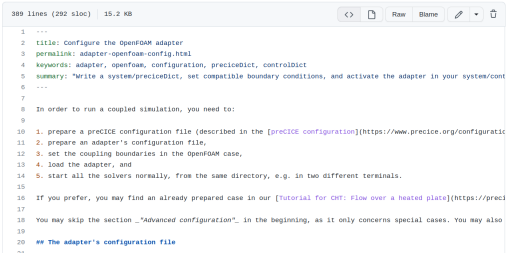


Workflow artifacts

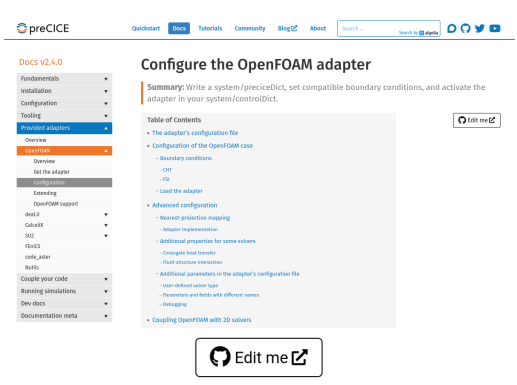


Documentation automation

adapter repository: `docs/config.md`



website rendering:



Documentation automation: setup

website repository: Git modules

8 lines (8 sloc) | 255 Bytes

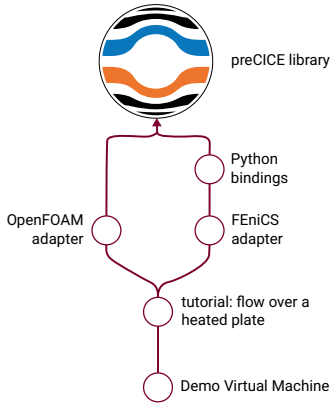
```
[submodule "tutorials"]
        path = imported/tutorials
        url = https://github.com/precice/tutorials.git
        branch = develop
[submodule "openfoam-adapter"]
        path = imported/openfoam-adapter
        url = https://github.com/precice/openfoam-adapter.git
        branch = develop
```

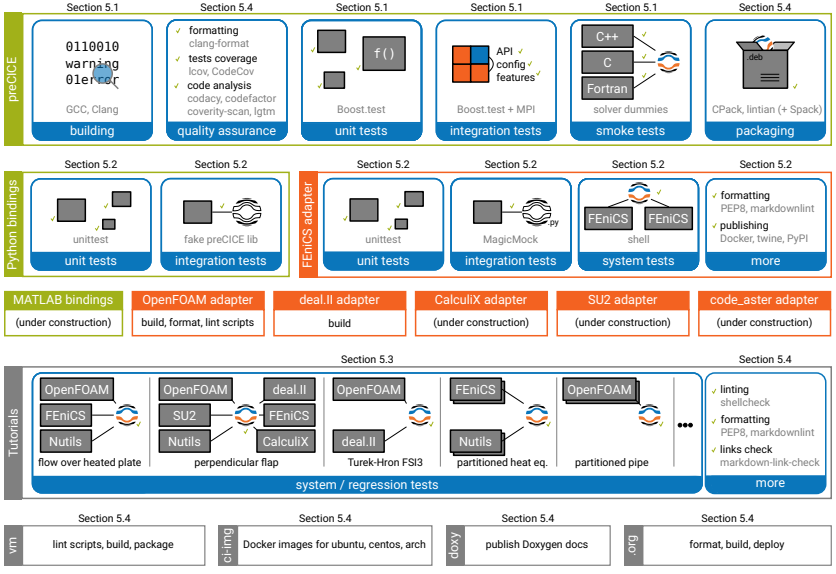
adapter repository: Trigger update

22 lines (22 sloc) | 598 Bytes — Raw | Blame

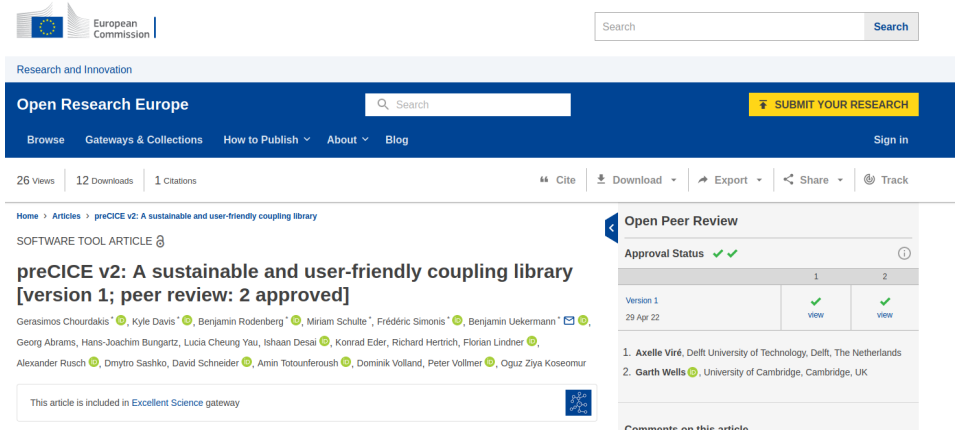
```yaml
name: Update website
on:
  push:
    branches:
      - 'develop'
    paths:
      - 'docs/**'
jobs:
  trigger:
    runs-on: ubuntu-latest
    env:
      WORKFLOW_FILENAME: update-submodules.yml
    steps:
    - name: Trigger workflow
      run: |
        curl \
        --request POST \
        --url https://api.github.com/repos/precice/precice.github.io/actions/workflows/$WORKFLOW_FILENAME/dispatches \
        --header "authorization: token ${{ secrets.WORKFLOW_DISPATCH_TOKEN }}" \
        --header "Accept: application/vnd.github.v3+json" \
        --data '{"ref":"master"}' \
        --fail
```

The multi-component preCICE ecosystem




preCICE
Section 5.1
0110010
warning
01error
GCC, Clang
building
Section 5.4
formatting
clang-format
tests coverage
lcov, CodeCov
code analysis
codacy, codefactor
coverity-scan, lgtm
quality assurance
Section 5.1
f()
Boost.test
unit tests
Section 5.1
API
config
features
Boost.test + MPI
integration tests
Section 5.1
C++
C
Fortran
solver dummies
smoke tests
Section 5.4
.deb
CPack, lintian (+ Spack)
packaging
Python bindings
Section 5.2
unittest
unit tests
Section 5.2
fake preCICE lib
integration tests
FEniCS adapter
Section 5.2
unittest
unit tests
Section 5.2
MagicMock
integration tests
Section 5.2
FEniCS
FEniCS
shell
system tests
Section 5.2
formatting
PEP8, markdownlint
publishing
Docker, twine, PyPI
more
MATLAB bindings
(under construction)
OpenFOAM adapter
build, format, lint scripts
deal.II adapter
build
CalculiX adapter
(under construction)
SU2 adapter
(under construction)
code_aster adapter
(under construction)
Tutorials
Section 5.3
OpenFOAM
FEniCS
Nutils
flow over heated plate
OpenFOAM
deal.II
SU2
FEniCS
Nutils
CalculiX
perpendicular flap
OpenFOAM
deal.II
Turek-Hron FSI3
FEniCS
Nutils
partitioned heat eq.
OpenFOAM
partitioned pipe
system / regression tests
Section 5.4
linting
shellcheck
formatting
PEP8, markdownlint
links check
markdown-link-check
more
vm
Section 5.4
lint scripts, build, package
ci-fenics
Section 5.4
Docker images for ubuntu, centos, arch
doxy
Section 5.4
publish Doxygen docs
org
Section 5.4
format, build, deploy

Key reference (fresh!)



Summary

- CI: execute workflows automatically
- GitHub Actions: easy and powerful
- Workflow: a YAML file under `.github/`
- preCICE: Many workflows in different repositories

Read more in the preCICE v2 paper (new)

Slides & feedback: go.tum.de/613308

`gerasimos.chourdakis@tum.de` (`@MakisH`)

(Note: looking for a research stay abroad in 2023)

Extras

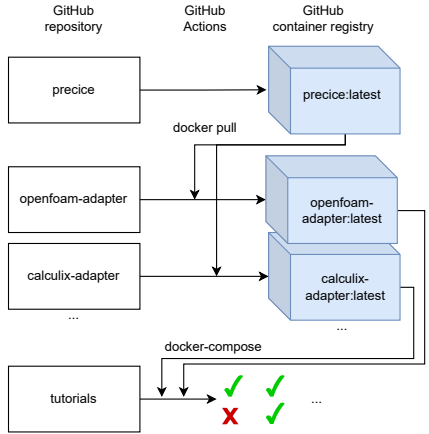
Prototype: System tests for the complete ecosystem

Testing the complete system for regressions:

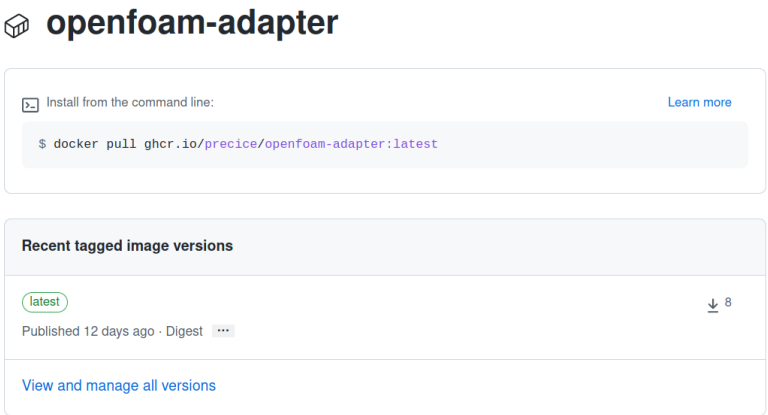
- Using multiple layers together
- Running complete examples
- Comparing results

(regression) testing $\neq$ validation!

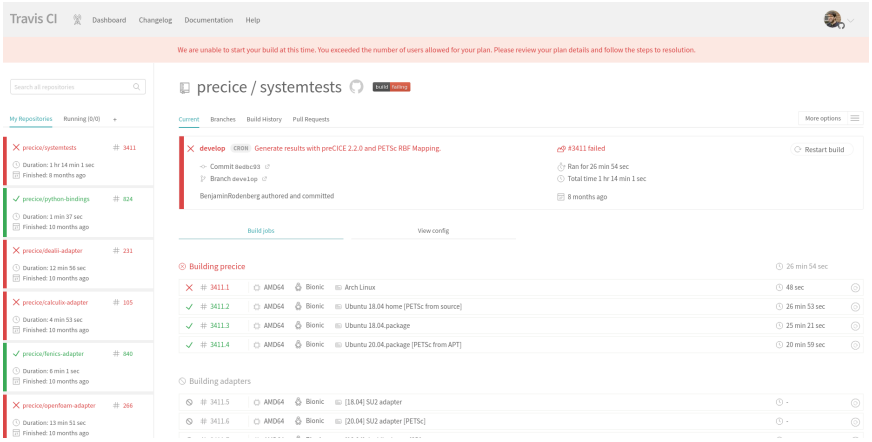
System tests architecture (prototype)



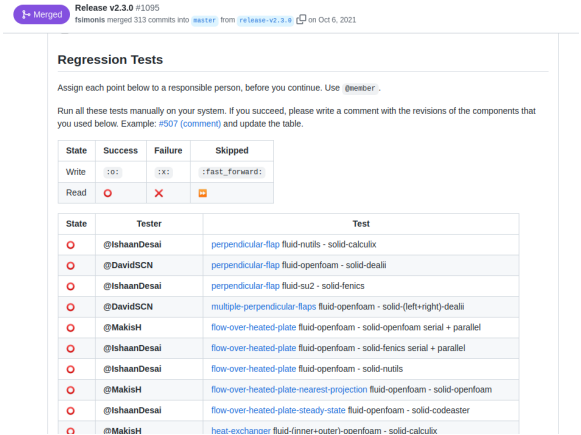
GitHub Container Registry



Previous approach to system tests



Manual testing to the rescue



Merged **Release v2.3.0** #1095

fsimonis merged 313 commits into `master` from `release-v2.3.0` on Oct 6, 2021

Regression Tests

Assign each point below to a responsible person, before you continue. Use `@member`.

Run all these tests manually on your system. If you succeed, please write a comment with the revisions of the components that you used below. Example: #507 (comment) and update the table.

State	Success	Failure	Skipped
Write	`:o:`	`:x:`	`:fast_forward:`
Read	⭕	❌	⏭️

State	Tester	Test
⭕	**@IshaanDesai**	perpendicular-flap fluid-nutils - solid-calculix
⭕	**@DavidSCN**	perpendicular-flap fluid-openfoam - solid-dealii
⭕	**@IshaanDesai**	perpendicular-flap fluid-su2 - solid-fenics
⭕	**@DavidSCN**	multiple-perpendicular-flaps fluid-openfoam - solid-(left+right)-dealii
⭕	**@MakisH**	flow-over-heated-plate fluid-openfoam - solid-openfoam serial + parallel
⭕	**@IshaanDesai**	flow-over-heated-plate fluid-openfoam - solid-fenics serial + parallel
⭕	**@IshaanDesai**	flow-over-heated-plate fluid-openfoam - solid-nutils
⭕	**@MakisH**	flow-over-heated-plate-nearest-projection fluid-openfoam - solid-openfoam
⭕	**@IshaanDesai**	flow-over-heated-plate-steady-state fluid-openfoam - solid-codeaster
⭕	**@MakisH**	heat-exchanger fluid-(inner+outer)-openfoam - solid-calculix

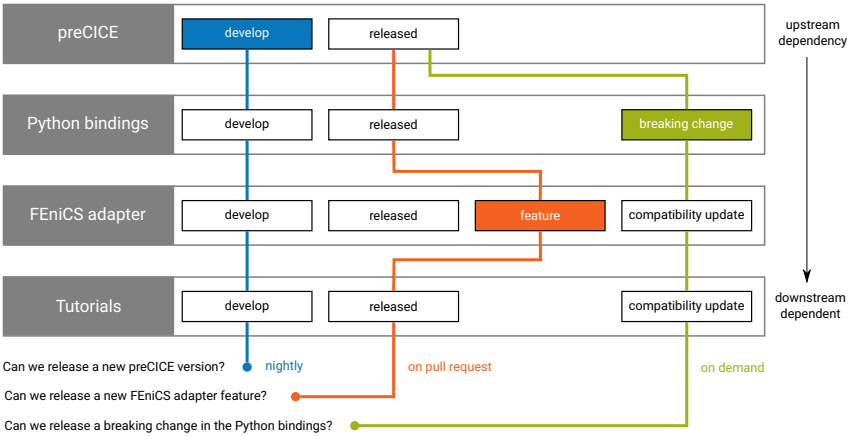
Docker Compose: Running

```
cd tutorials/flow-over-heated-plate/tests

export TAG_OPENFOAM_ADAPTER=latest

MY_UID="$(id -u)" MY_GID="$(id -g)" docker-compose up
```

Different perspectives



Docker Compose

```yaml
# File: docker-compose.yml
services:
  # ...
  fluid-openfoam:
    image: "ghcr.io/precice/of-adapter:
            ${TAG_OPENFOAM_ADAPTER}"
    user: ${MY_UID}:${MY_GID}
    volumes:
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - ../..:/tests
    command: >
      /bin/bash -c "openfoam2112 ./run.sh"
```

Docker Compose

```yaml
# File: docker-compose.yml
services:
  # ...
  fluid-openfoam:
    image: "ghcr.io/precice/of-adapter:
            ${TAG_OPENFOAM_ADAPTER}"
    user: ${MY_UID}:${MY_GID}
    volumes:
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - ../..:/tests
    command: >
      /bin/bash -c "openfoam2112 ./run.sh"
```

```yaml
solid-openfoam:
  image: # same as fluid-openfoam
  user: # same
  volumes:
    - # same
  command: # same, different directory
```

What to compare to?

Reference data:

- Solver logs $\rightarrow$ often not identical
- Solver results $\rightarrow$ various formats, too much
- preCICE exports $\rightarrow$ same format,
  no time-related noise, enough

  (demonstrated in TUM FSI Seminar paper 2020 by Mohamad Kanj)

What to compare to?

Reference data:

- Solver logs → often not identical
- Solver results → various formats, too much
- preCICE exports → same format, no time-related noise, enough

(demonstrated in TUM FSI Seminar paper 2020 by Mohamad Kanj)

Tutorials structure extension:

```
- flow-over-heated-plate/
  - fluid-openfoam/
  - solid-openfoam/
  - precice-config.xml
  - reference-data/
    - fluid-openfoam_solid-openfoam/
  - tests/
    - docker-compose.yml
- tools/
  - run-tests.sh
```

BSSW.io article: Unit and integration tests in preCICE