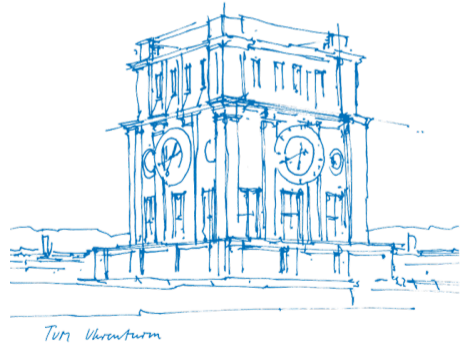


Continuous integration with GitHub Actions in preCICE


Munich RSE meetup

Gerasimos Chourdakis
Technical University of Munich





March 15, 2023




Safely contributing to your GitHub project



✓ All checks have passed [Hide all checks](#)
4 successful checks

✓  Build with OpenFOAM v2112 / build (pull_request) Successful in 17m Details
✓  Check code formatting / Check formatting (clang-format) (pull_request) Successful... Details
✓  Lint docs / check_md (pull_request) Successful in 18s Details
✓  Lint scripts / check_shell (pull_request) Successful in 3s Details

✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

[Squash and merge](#)  or [view command line instructions.](#)

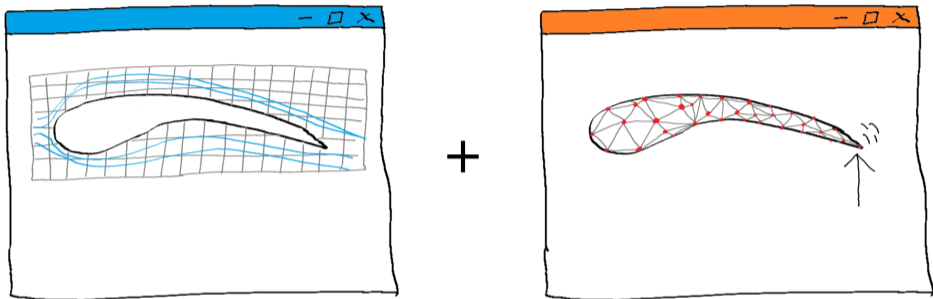


- Gerasimos → Gerasimakis → Makis
- Doctoral candidate at TUM CIT (2018)
- Growing a community for preCICE
- M.Sc. Computational Science & Engineering, TUM
- Dipl. Chemical Engineering, Athens

There will be code.
Slides & feedback: go.tum.de/389945

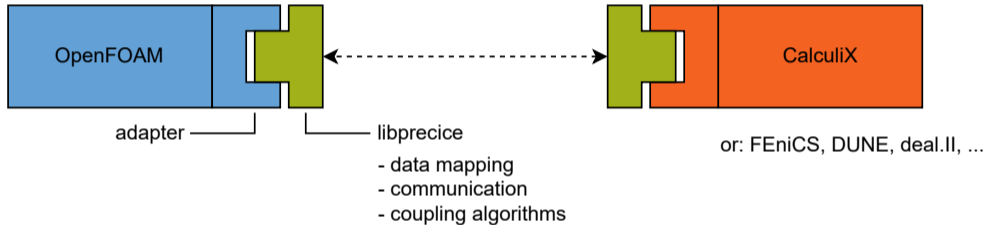


`gerasimos.chourdakis@tum.de` (@MakisH)

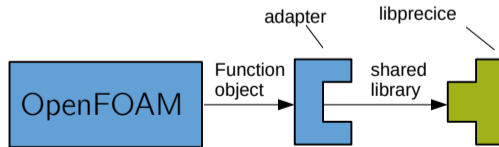
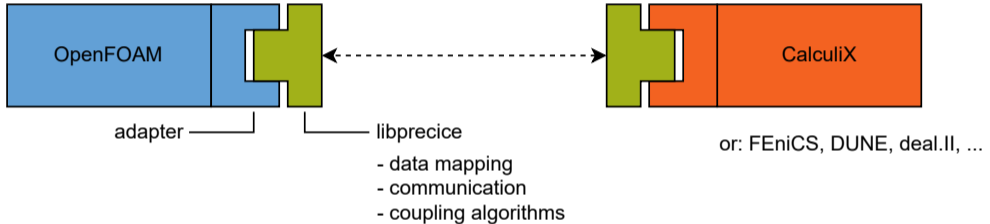


 preCICE

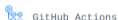
preCICE and its OpenFOAM adapter



preCICE and its OpenFOAM adapter



Continuous Integration tools

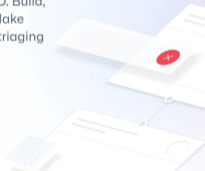


Automate your workflow from idea to production

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

[Get started with Actions >](#)

Questions? [Contact Sales >](#)



GitLab Continuous Integration (CI)

Outstanding source code exists. For teams and projects big and small, the proof is in the pipeline.

[Get to know CI/CD](#)

GitHub Actions for the OpenFOAM adapter

Workflows

[New workflow](#)

All workflows

Build and store a Docker ima...

Build with OpenFOAM v2112

Check code formatting

Check links (manual)

Custom build (manual)

Lint docs

Lint scripts

Lint scripts and docs


All workflows

Showing runs from all workflows

742 workflow run results		Event ▾	Status ▾	Branch ▾	Actor ▾
	Extend Allwmake suggestions for possible problems Lint scripts #374: Pull request #220 opened by MakisH	MakisH:extend-allwmake-su...	3 days ago	19s	...
	Extend Allwmake suggestions for possible problems Lint docs #368: Pull request #220 opened by MakisH	MakisH:extend-allwmake-su...	3 days ago	29s	...
	Extend Allwmake suggestions for possible problems Check code formatting #382: Pull request #220 opened by MakisH	MakisH:extend-allwmake-su...	3 days ago	46s	...
	Extend Allwmake suggestions for possible problems Build with OpenFOAM v2112 #438: Pull request #220 opened by MakisH	MakisH:extend-allwmake-su...	3 days ago	14m 6s	...
	Update master (v1.1.0) Lint scripts #354: Pull request #186 synchronize by MakisH	develop	12 days ago	21s	...

Automatic format check with clang-format

openfoam-adapter / .github / workflows /

 **MakishH** Update changelog for v1.2.2 ✓

Name

..

build-custom.yml

build.yml

check-format.yml

check-links.yml

check-markdown.yml

check-shell.yml

install-dependencies.sh


update-website.yml

13 lines (13 sloc) | 369 Bytes

```
1 name: Check code formatting
2 on: [push, pull_request]
3 jobs:
4   formatting-check:
5     name: Check formatting (clang-format)
6     runs-on: ubuntu-latest
7     steps:
8     - uses: actions/checkout@v2
9     - name: Run clang-format style check for C/C++ programs.
10       uses: jidicula/clang-format-action@main
11       with:
12         clang-format-version: '11'
13         check-path: '.'
```

Automatic format check with clang-format

openfoam-adapter / .github / workflows /

 **MakishH** Update changelog for v1.2.2 ✓

Name

..

build-custom.yml

build.yml

check-format.yml

check-links.yml

check-markdown.yml

check-shell.yml

install-dependencies.sh

update-website.yml

13 lines (13 sloc) | 369 Bytes

```
1 name: Check code formatting
2 on: [push, pull_request]
3 jobs:
4   formatting-check:
5     name: Check formatting (clang-format)
6     runs-on: ubuntu-latest
7     steps:
8     - uses: actions/checkout@v2
9     - name: Run clang-format style check for C/C++ programs.
10       uses: jidicula/clang-format-action@main
11       with:
12         clang-format-version: '11'
13         check-path: '.'
```

Discover issues in shell scripts with `shellcheck` `myscript.sh`

```

-$ 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

```
1 name: Lint scripts
2 on: [push, pull_request]
3 jobs:
4   check_shell:
5     runs-on: ubuntu-latest
6     steps:
7       - name: Check out repository
8         uses: actions/checkout@v2
9       - name: Lint shell scripts (shellcheck)
10        uses: ludeeus/action-shellcheck@master
```

GitHub Actions: Manual triggering

File .github/workflows/build-custom.yml

```
name: Custom build (manual)
```

```
on:
```

```
  workflow_dispatch:
```

```
    inputs:
```

```
      # ...
```

```
      versionOpenFOAM:
```

```
        type: choice
```

```
        options:
```

- OpenFOAMv2206
- OpenFOAM9

Run workflow ▾

Use workflow from

Branch: develop ▾

Virtual Environment *

ubuntu-18.04 ▾

Ref (branch/tag/commit) of the OpenFOAM adapter to build *

develop

Version of OpenFOAM to build with *

OpenFOAMv2112 ▾

Version of preCICE to build with *

2.3.0

Run tutorial flow-over-heated-plate

Run tutorial quickstart

Run tutorial partitioned-pipe

Branch of the tutorials to use *

master

Run workflow

Workflow logs

The screenshot shows the GitHub Actions interface for a workflow named 'Add a cleanup step Run tutorials with docker-compose #19'. The workflow is currently running, as indicated by a green checkmark. The main view displays the logs for the 'run-flow-over-heated-plate' job, which succeeded 12 days ago in 58 seconds. The logs are presented as a list of steps, each with a status icon, a description, and a duration.

preceice / tutorials Public

Unpin Unwatch 8 Fork 55 Starred 51

Code Issues 22 Pull requests 3 Discussions Actions Projects Security Insights Settings

✓ Add a cleanup step Run tutorials with docker-compose #19 Re-run all jobs

Summary

Jobs

✓ run-flow-over-heated-plate

run-flow-over-heated-plate succeeded 12 days ago in 58s

Search logs

> ✓ Set up job	1s
> ✓ Checkout repository	2s
> ✓ Run docker-compose	49s
> ✓ Archive logs	1s
> ✓ Archive case files	5s
> ✓ Post Checkout repository	0s
> ✓ Complete job	0s

Workflow artifacts

precice / tutorials Public

Unpin Unwatch 8 Fork 55 Starred 51

<> Code Issues 22 Pull requests 3 Discussions **Actions** Projects Security Insights Settings

✓ Add a cleanup step Run tutorials with docker-compose #19 Re-run all jobs

Summary

Jobs

- run-flow-over-heated-plate

Triggered via push 12 days ago


MakisH pushed → 8a77be3 `add-nightly-tests` **Success** **1m 10s** **2** Artifacts

run-tutorials-compose.yml
on: push

- run-flow-over-heated-plate 51s

Artifacts
Produced during runtime

Name	Size
case-files	6.63 MB
logs	155 KB



Documentation automation with Jekyll

adapter repository: docs/config.md

```

389 lines (292 sloc) | 15.2 KB
<>  [ ] Raw Blame [ ] [ ] [ ]
1 ---
2 title: Configure the OpenFOAM adapter
3 permalink: adapter-openfoam-config.html
4 keywords: adapter, openfoam, configuration, preciceDict, controlDict
5 summary: "Write a system/preciceDict, set compatible boundary conditions, and activate the adapter in your system/controlDict"
6 ---
7
8 In order to run a coupled simulation, you need to:
9
10 1. prepare a preCICE configuration file (described in the [preCICE configuration](https://www.precice.org/configuration))
11 2. prepare an adapter's configuration file,
12 3. set the coupling boundaries in the OpenFOAM case,
13 4. load the adapter, and
14 5. start all the solvers normally, from the same directory, e.g. in two different terminals.
15
16 If you prefer, you may find an already prepared case in our [tutorial for CHT: Flow over a heated plate](https://precice.org/tutorial-cht-flow-over-a-heated-plate)
17
18 You may skip the section "_Advanced configuration_" in the beginning, as it only concerns special cases. You may also
19
20 ## The adapter's configuration file
  
```

website rendering:

The screenshot shows the preCICE website interface. At the top, there's a navigation bar with links for 'Quickstart', 'Docs', 'Tutorials', 'Community', 'Blog', and 'About'. A search bar is also present. The main content area is titled 'Configure the OpenFOAM adapter'. On the left, a sidebar menu lists various documentation sections, with 'OpenFOAM' highlighted. The main content includes a 'Summary' section stating: 'Write a system/preciceDict, set compatible boundary conditions, and activate the adapter in your system/controlDict.' Below this is a 'Table of Contents' section listing sub-topics like 'Boundary conditions', 'CHT', 'FI', 'Load the adapter', 'Advanced configuration', 'Nearest-projection mapping', 'Adapter implementation', 'Additional properties for some solvers', 'Coupage heat transfer', 'Fluid structure interaction', 'Additional parameters in the adapter's configuration file', 'User-defined solver type', 'Parameters and fields with different names', 'Debugging', and 'Coupling OpenFOAM with 2D solvers'. An 'Edit me' button is located at the bottom right of the page.

Documentation automation: setup

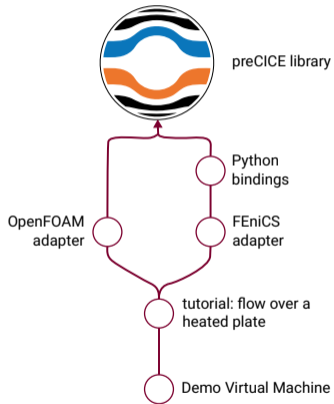
website repository: Git modules

```
0 lines (0 sloc) | 255 Bytes ...
1 [submodule "tutorials"]
2   path = imported/tutorials
3   url = https://github.com/precice/tutorials.git
4   branch = develop
5 [submodule "openfoam-adapter"]
6   path = imported/openfoam-adapter
7   url = https://github.com/precice/openfoam-adapter.git
8   branch = develop
```

adapter repository: Trigger update

```
22 lines (22 sloc) | 598 Bytes Raw Blame
1 name: Update website
2 on:
3   push:
4     branches:
5       - 'develop'
6     paths:
7       - 'docs/**'
8   jobs:
9     trigger:
10      runs-on: ubuntu-latest
11      env:
12        WORKFLOW_FILENAME: update-submodules.yml
13      steps:
14        - name: Trigger workflow
15          run: |
16            curl \
17              --request POST \
18              --url https://api.github.com/repos/precice/precice.github.io/actions/workflows/$WORKFLOW_FILENAME/dispatches \
19              --header "authorization: token ${ secrets.WORKFLOW_DISPATCH_TOKEN }" \
20              --header "Accept: application/vnd.github.v3+json" \
21              --data '{"ref":"master"}' \
22              --fail
```

The multi-component preCICE ecosystem



preCICE	Section 5.1	Section 5.4	Section 5.1	Section 5.1	Section 5.1	Section 5.4
	0110010 warning 01error	<ul style="list-style-type: none"> formatting clang-format tests coverage lcov, CodeCov code analysis codacy, codefactor coverity-scan, lgtm 	f()	API config features	C++ C Fortran	deb
	GCC, Clang		Boost.test	Boost.test + MPI	solver dummies	CPack, lintian (+ Spack)
	building	quality assurance	unit tests	integration tests	smoke tests	packaging

Python bindings	Section 5.2	Section 5.2	Section 5.2	Section 5.2	Section 5.2	Section 5.2
	unittest	fake preCICE lib	unittest	MagicMock	FEniCS FEniCS shell	<ul style="list-style-type: none"> formatting PEP8, markdownlint publishing Docker, twine, PyPI
	unit tests	integration tests	unit tests	integration tests	system tests	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	Section 5.4				
	OpenFOAM, FEniCS, Nutils flow over heated plate	OpenFOAM, deal.II, FEniCS, SU2, Nutils, CalculiX perpendicular flap	OpenFOAM, deal.II Turek-Hron FSI3	FEniCS, Nutils partitioned heat eq.	OpenFOAM partitioned pipe	<ul style="list-style-type: none"> linting shellcheck formatting PEP8, markdownlint links check markdown-link-check
	system / regression tests					more

vm	Section 5.4	Section 5.4	Section 5.4	Section 5.4
	lint scripts, build, package	ch-img Docker images for ubuntu, centos, arch	doxy publish Doxygen docs	.org format, build, deploy

A quality dashboard for your project

README.md 

preCICE

Communication

matrix [join chat](#) discourse QA [4.9k posts](#) mailing list [subscribe](#) twitter [@preCICE_org](#) website [up](#)

Project Status

release [v2.5.0](#) doi [10.18419/darus-2613](#)  Build and Test [passing](#)

Project Quality

xSDK [member](#) openssf best practices [passing](#) codefactor [A](#)  CodeQL [passing](#)  codecov [91%](#)

Key reference


 Search

Research and Innovation

Open Research Europe

SUBMIT YOUR RESEARCH

- Browse
- Gateways & Collections
- How to Publish
- About
- Resource Hub
- Blog

Sign in

95 Views | 49 Downloads | 4 Citations

- Cite
- Download
- Export
- Share
- Track

Home > Articles > preCICE v2: A sustainable and user-friendly coupling library

SOFTWARE TOOL ARTICLE

REVISED **preCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]**

Gerasimos Chourdakis, Kyle Davis, Benjamin Rodenberg, Miriam Schulte, Frédéric Simonis, Benjamin Uekermann, Georg Abrams, Hans-Joachim Bungartz, Lucia Cheung Yau, Ishaan Desai, Konrad Eder, Richard Hertrich, Florian Lindner, Alexander Rusch, Dmytro Sashko, David Schneider, Amin Toutouferoush, Dominik Volland, Peter Vollmer, Oguz Ziya Koseomur

This article is included in Horizon 2020 gateway



This article is included in Marie-Sklodowska-Curie Actions (MSCA) gateway



Open Peer Review

Approval Status ✓✓ ⓘ

	1	2
Version 2 (Revision) 30 Sep 22		
Version 1 29 Apr 22	✓ view	✓ view

- Axelle Viré, Delft University of Technology, Delft, The Netherlands
- Garth Wells, University of Cambridge, Cambridge, UK

Summary

- Check your PRs automatically
- Workflow: a YAML file under `.github/`
- Manual triggering with custom input with `on: workflow_dispatch`
- `shellcheck` is a cool tool
- preCICE: Many different repositories
- Source documentation from multiple repositories with Git submodule, GitHub Actions, and GitHub Pages

Read more: preCICE v2 paper, extra slides

Slides & feedback: go.tum.de/389945



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

Extras

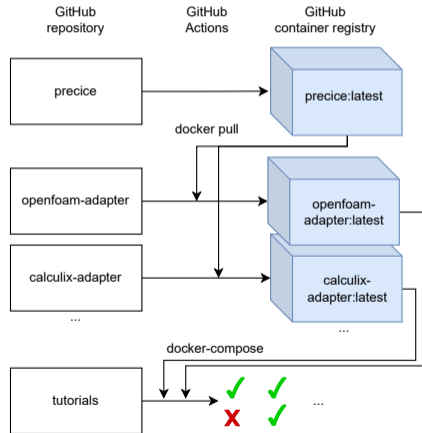
Prototype: System tests for the complete ecosystem

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

openfoam-adapter

 Install from the command line:

[Learn more](#)

```
$ docker pull ghcr.io/precice/openfoam-adapter:latest
```

Recent tagged image versions

latest

 8

Published 12 days ago · Digest 

[View and manage all versions](#)

Previous approach to system tests

Travis CI [Dashboard](#) [Changelog](#) [Documentation](#) [Help](#)

We are unable to start your build at this time. You exceeded the number of users allowed for your plan. Please review your plan details and follow the steps to resolution.

Search all repositories

My Repositories **Running (0/0)** +

- prece/systemtests** # 3411
 - Duration: 1 hr 14 min 1 sec
 - Finished: 8 months ago
- prece/python-bindings** # 824
 - Duration: 1 min 37 sec
 - Finished: 10 months ago
- prece/ocaml-adapter** # 231
 - Duration: 12 min 56 sec
 - Finished: 10 months ago
- prece/calculx-adapter** # 105
 - Duration: 4 min 53 sec
 - Finished: 10 months ago
- prece/ferics-adapter** # 840
 - Duration: 6 min 1 sec
 - Finished: 10 months ago
- prece/openfoam-adapter** # 266
 - Duration: 13 min 51 sec
 - Finished: 10 months ago

prece / systemtests **Build**

Current Branches Build History Pull Requests More options

develop **CRON** Generate results with preCICE 2.2.0 and PETSc RBF Mapping. **#3411 failed** Restart build

→ Commit 8edbc93
 ↳ Branch devvs1ep
 BenjaminRodenberg authored and committed

Build jobs View config

Building prece 26 min 54 sec

X # 3411.1	AMD64	Bionic	Arch Linux	48 sec
✓ # 3411.2	AMD64	Bionic	Ubuntu 18.04 home [PETSc from source]	26 min 53 sec
✓ # 3411.3	AMD64	Bionic	Ubuntu 18.04.package	25 min 21 sec
✓ # 3411.4	AMD64	Bionic	Ubuntu 20.04.package [PETSc from APT]	20 min 59 sec

Building adapters

# 3411.5	AMD64	Bionic	[18.04] SU2 adapter	-
# 3411.6	AMD64	Bionic	[20.04] SU2 adapter [PETSc]	-

Manual testing to the rescue

Merged **Release v2.3.0** #1095
 fsmionis 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	<code>:0:</code>	<code>:x:</code>	<code>:fast_forward:</code>
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

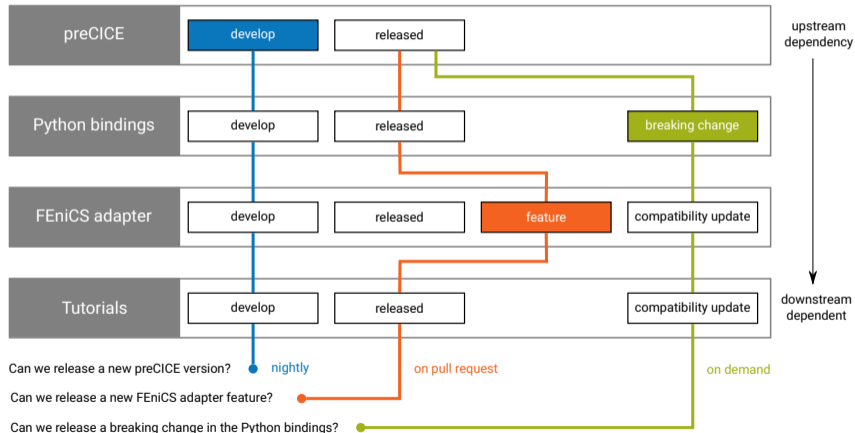
Docker Compose: Running

```
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

```
# 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

```
# 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"

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


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)

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



The screenshot shows the top navigation bar of the BSSW.io website, which is dark blue with white text. The logo 'better scientific software' is on the left, and navigation links for 'Resources', 'Blog', 'Events', and 'About' are on the right. Below the navigation bar, the breadcrumb trail reads 'HOME > BLOG > Overcoming Complexity in Testing Multiphysics Coupling...'. The main heading of the article is 'Overcoming Complexity in Testing Multiphysics Coupling Software'. Below the heading are social media share icons for LinkedIn, Facebook, Twitter, and a general share icon. The introductory paragraph states: 'Testing complex software can easily get out of hand, especially when your product is a multiphysics coupling library. Fortunately, we've found some strategies that have helped tame the nightmare.' At the bottom, there are metadata sections: 'PUBLISHED FEB 07, 2022', 'AUTHORS FRÉDÉRIC SIMONIS, GERASIMOS CHOURDAKIS, AND BENJAMIN UEKERMANN', and 'TOPICS BETTER PLANNING, BETTER RELIABILITY, BETTER DEVELOPMENT, SOFTWARE INTEROPERABILITY, TESTING, DEVELOPMENT TOOLS'.

better scientific software

Resources ▾ Blog Events About ▾ 🔍

HOME > BLOG > Overcoming Complexity in Testing Multiphysics Coupling...

Overcoming Complexity in Testing Multiphysics Coupling Software

SHARE in f t ↻

Testing complex software can easily get out of hand, especially when your product is a multiphysics coupling library. Fortunately, we've found some strategies that have helped tame the nightmare.

PUBLISHED FEB 07, 2022

AUTHORS **FRÉDÉRIC SIMONIS, GERASIMOS CHOURDAKIS, AND BENJAMIN UEKERMANN**

TOPICS **BETTER PLANNING** **BETTER RELIABILITY** **BETTER DEVELOPMENT** **SOFTWARE INTEROPERABILITY** **TESTING** **DEVELOPMENT TOOLS**