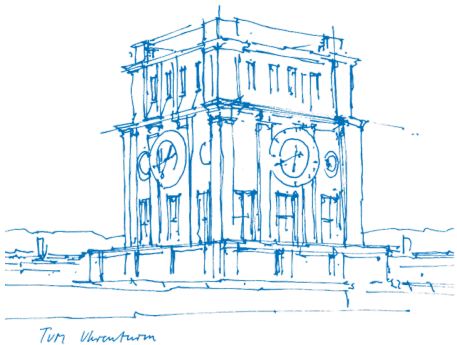
Testing the multi-component preCICE ecosystem

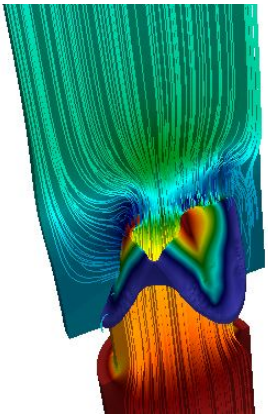
SIGDIUS seminar (online)

Gerasimos Chourdakis
Technical University of Munich

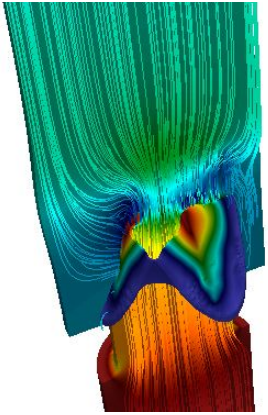
May 4, 2022

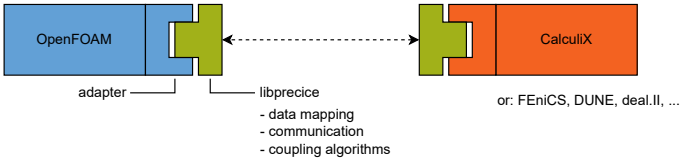


Partitioned multi-physics simulations



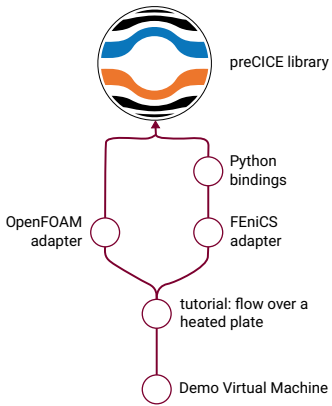
Partitioned multi-physics simulations

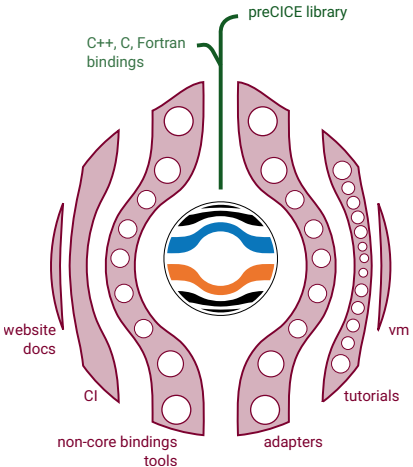




Left: “Evaluation of heart valve biomechanics”, Kyle Davis (Univ. Free State, South Africa) – preCICE Community Stories on `precice.org`

The multi-component preCICE ecosystem





Different levels of tests

Unit tests

```
enum struct Require { PETSc };

BOOST_TEST(ParallelPETScFeature) {
  // Test using 2 ranks and requiring PETSc to be setup.
  PRECICE_TEST(2_ranks, Require::PETSc);
  // test feature
}
```

Integration tests

```
PRECICE_TEST("Fluid"_on(2_ranks), "Solid"_on(1_rank));
             ^^^^^^^.............^^^^^^^..... Participant names
                   ^^^.................^^^..... String-literal
                      ^.......^...........^......^.. operator()
                       ^^^^^^^.............^^^^^^... Rank information
```

Different levels of tests

Different levels of tests

System tests

Testing the complete system for regressions:

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

(regression) testing $\neq$ validation!

But preCICE already has system tests! ...right?

Tests — 1 hr 31 min 36 sec

✓	# 2959.18	AMD64	Bionic	[18.04] OpenFOAM <-> OpenFOAM	5 min 16 sec
✓	# 2959.19	AMD64	Bionic	[18.04] Calculix <-> OpenFOAM	6 min 43 sec
✓	# 2959.20	AMD64	Bionic	[18.04] FEniCS <-> FEniCS	4 min 40 sec
✓	# 2959.21	AMD64	Bionic	[18.04] Bindings/Solverdummies	5 min 10 sec
✓	# 2959.22	AMD64	Bionic	[18.04] deal.ii <-> OpenFOAM [FSI][3D]	8 min 2 sec
✓	# 2959.23	AMD64	Bionic	[18.04] deal.ii <-> OpenFOAM [FSI][2D]	7 min 43 sec
✓	# 2959.24	AMD64	Bionic	[20.04 PETSc] deal.ii <-> OpenFOAM [FSI][2D]	6 min 59 sec
✓	# 2959.25	AMD64	Bionic	[18.04] OpenFOAM <-> OpenFOAM [nearest projection]	5 min 13 sec
✓	# 2959.26	AMD64	Bionic	[18.04] OpenFOAM <-> Code_Aster	5 min 21 sec
✓	# 2959.27	AMD64	Bionic	[18.04] Elastictube1D - Python	6 min 47 sec
✓	# 2959.28	AMD64	Bionic	[18.04] Elastictube1D - C++	4 min 9 sec
✗	# 2959.29	AMD64	Bionic	[18.04] SU2 <-> Calculix [unstable][failure allowed]	7 min 40 sec
✗	# 2959.30	AMD64	Bionic	[20.04 PETSc] SU2 <-> Calculix [unstable][failure allowed]	6 min 51 sec
✗	# 2959.31	AMD64	Bionic	[18.04] nutils <-> OpenFOAM [failure allowed]	7 min 58 sec
✓	# 2959.32	AMD64	Bionic	[18.04 PETSc] OpenFOAM <-> CalculiX [FSI][failure allowed]	4 min 58 sec

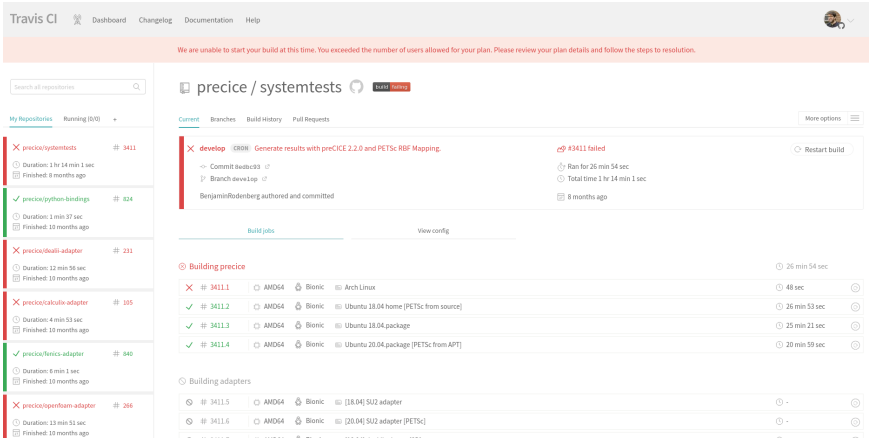
Contributors 9

Languages

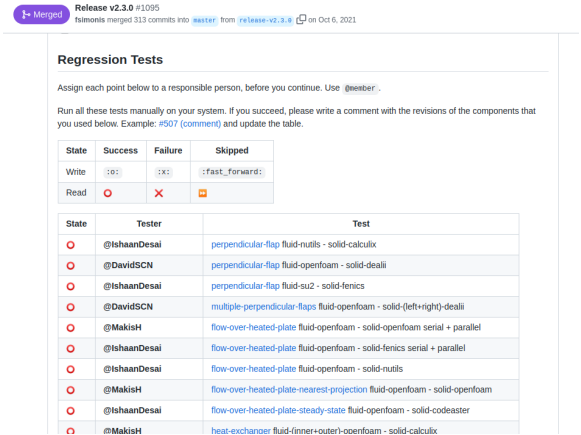
- **Python** 81.3%
- **Shell** 11.3%
- **Dockerfile** 2.8%
- **HTML** 2.6%
- **Roff** 2.0%

Since June 15th, 2021, the building on travis-ci.org is ceased. Please use travis-ci.com from now on.

But preCICE already has system tests! ...right?



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

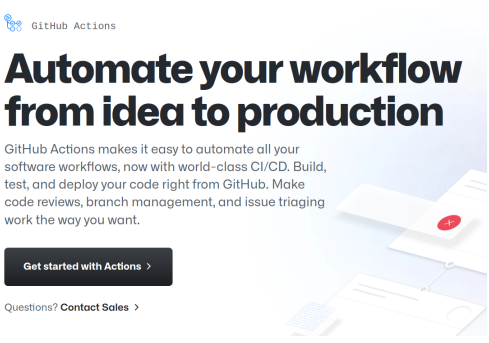
Assign each point below to a responsible person, before you continue. Use `@member`.

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.

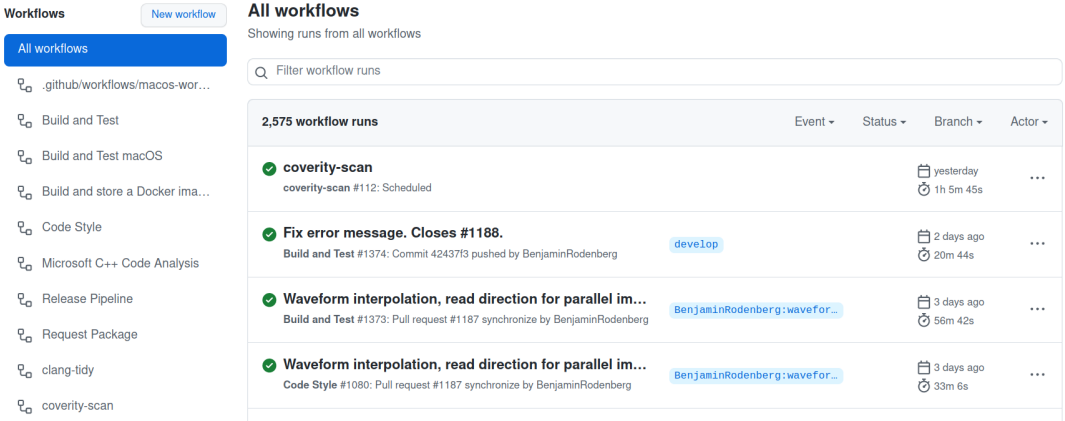
State	Success	Failure	Skipped
Write	`:o:`	`:x:`	`:fast_forward:`
Read	⭕	❌	⏭️

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

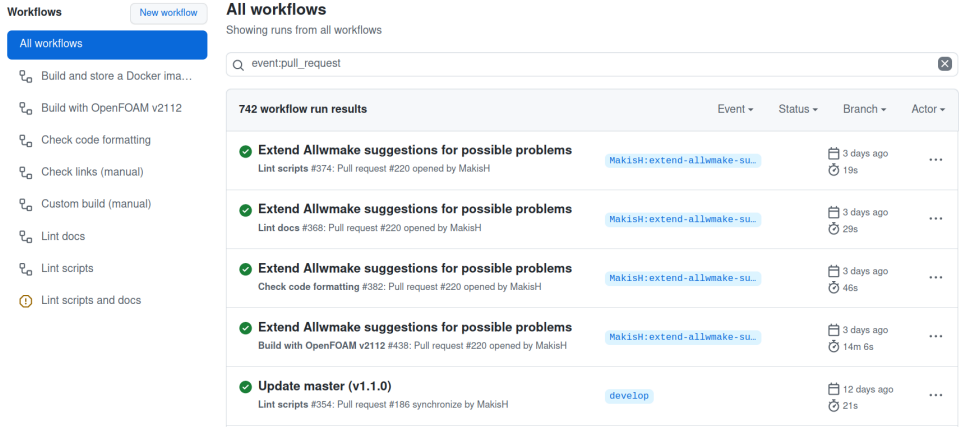
New tools at hand



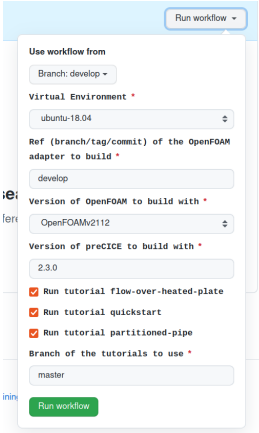
GitHub Actions: `precice/precice`



GitHub Actions: precice/openfoam-adapter

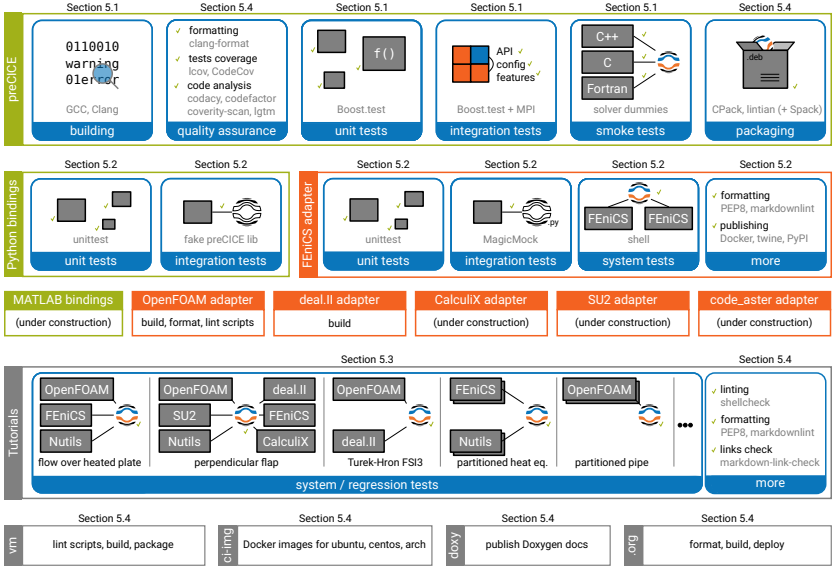


GitHub Actions: Manual triggering

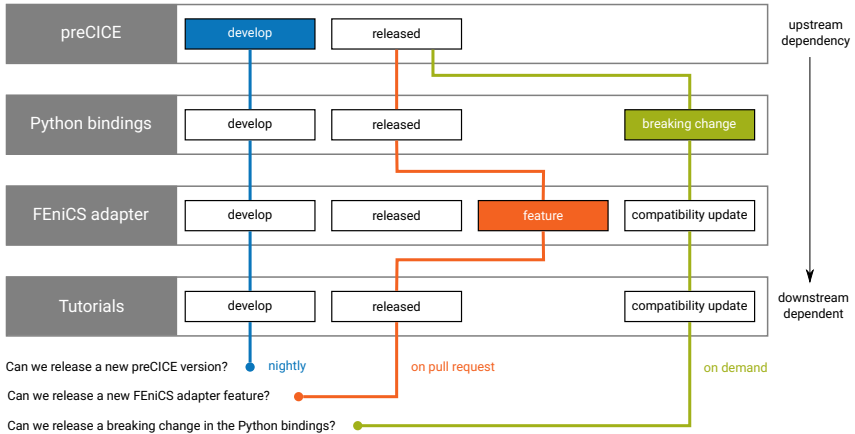


GitHub Actions: How does the config look like?

```yaml
# File .github/workflows/build-custom.yml
name: Custom build (manual)
on:
  workflow_dispatch:
    inputs:
      # ...
      versionOpenFOAM:
        type: choice
        options:
          - OpenFOAMv2112
          - OpenFOAM8

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


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

Different perspectives

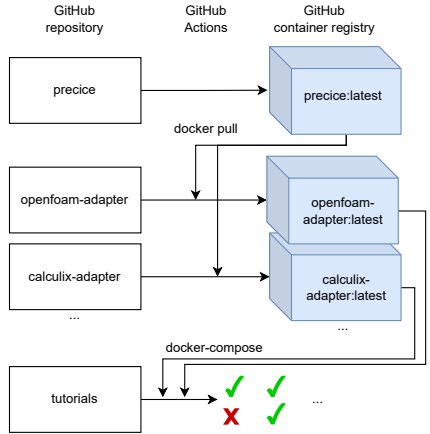


Current state

- Prototype
  - Running locally and on GitHub Actions
  - Only latest version of specific branches
  - Only preCICE, OpenFOAM adapter, and tutorials
  - Only flow-over-heated-plate OpenFOAM-OpenFOAM
  - No results comparison
- Results comparison
- Manual triggering for any branches (tags)
- Extension to all repositories and tutorials
- Automated triggering

Pull Requests precice/1090, openfoam-adapter/184, tutorials/220

Architecture



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

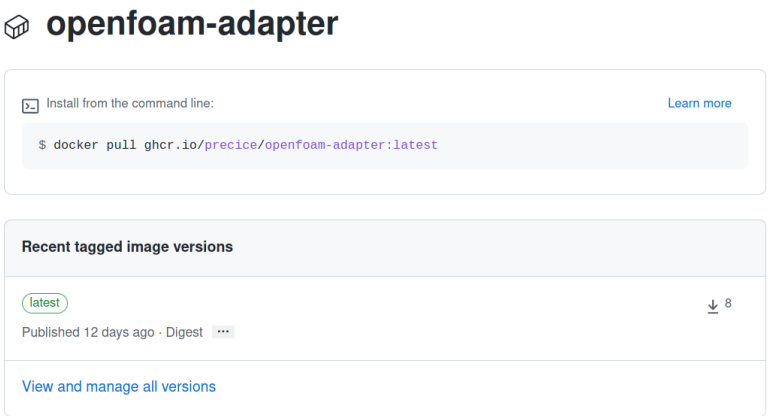
```yaml
# File: docker-compose.yml
services:
  # ...
  fluid-openfoam:
    image: "ghcr.io/precice/of-adapter:
            ${TAG_OPENFOAM_ADAPTER}"
    user: ${MY_UID}:${MY_GID}
    volumes:
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - ../..:/tests
    command: >
     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

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

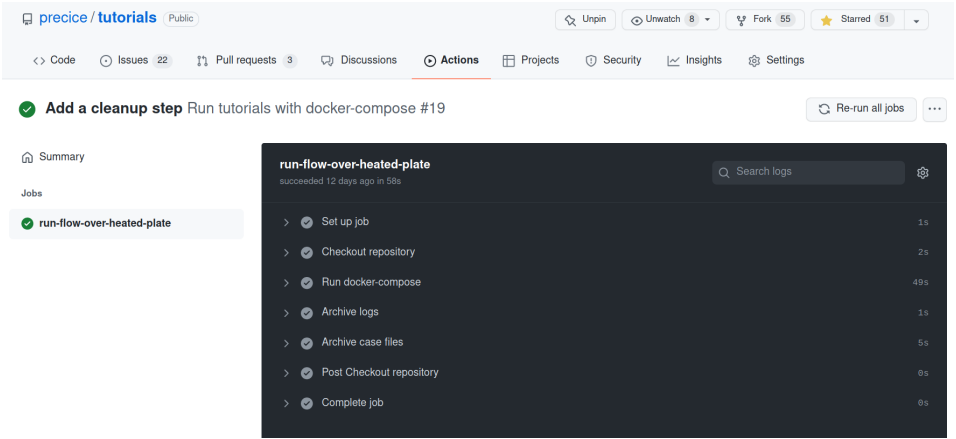
Docker Compose: Running

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

export TAG_OPENFOAM_ADAPTER=latest

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

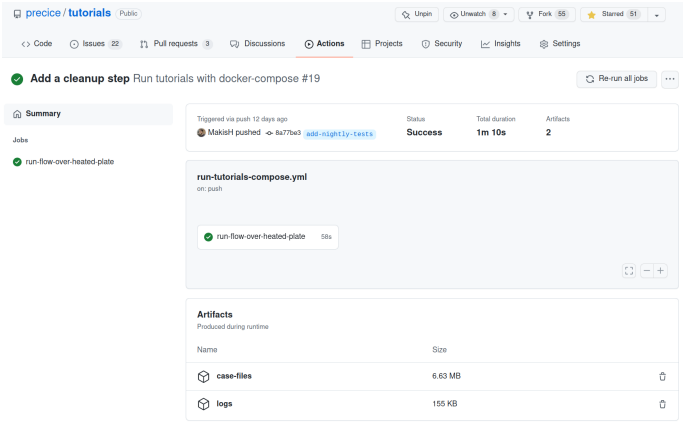
GitHub Container Registry



Running the tutorials: logs



Running the tutorials: artifacts



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
```

So, how is the new system better?

	Old	New
Tools	Travis CI + local	GitHub Actions + local
Local runners	–	Possible (proprietary solvers)
Container storage	DockerHub	GHCR
Artifacts storage	Dedicated repo	Next to the job
What to test	All possible combination	Selected by default, any on demand
Goal	Many features, coverage	Minimal code, sustainability

Key reference (fresh!)

Summary

- the library is tested widely,
  the ecosystem follows (complex)
- system tests grew, but hit by bus and
  vendor lock-in
- new, simpler implementation based on new
  tools and previous lessons
- work in progress

Read more in preCICE v2 paper (in review)
and BSSW blog.

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

Give me feedback: `go.tum.de/820189`