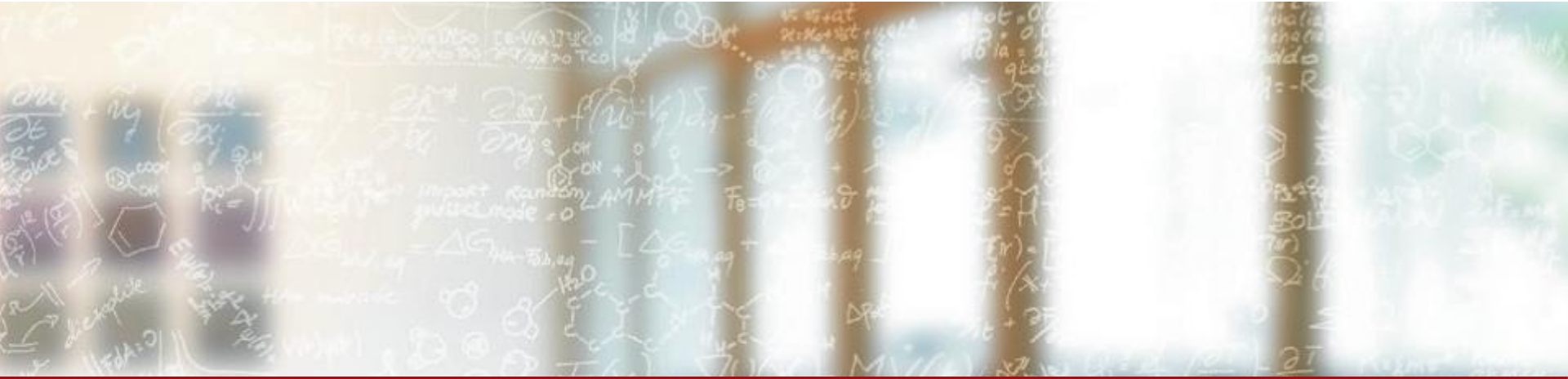




**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



# Scientific application deployment using CI/CD pipelines

SOS26

A. Fink, B. Cumming, T-I. Manitaras

March, 2024

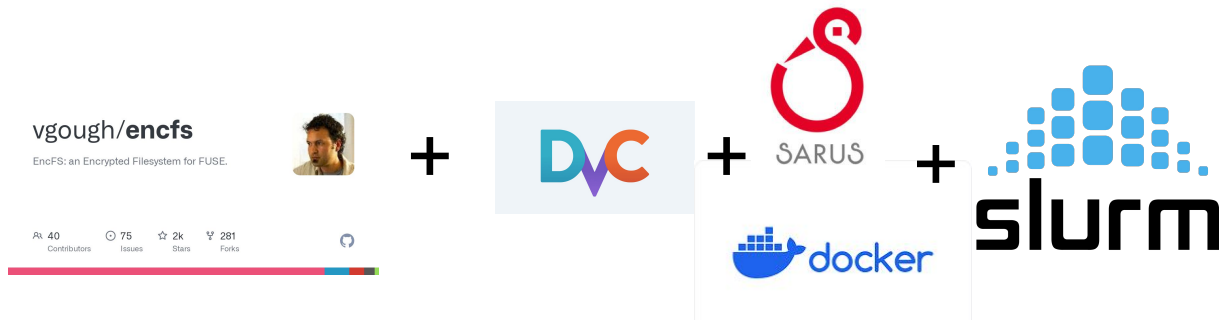
# async-encfs-dvc

Standalone pip-package extending DVC with

- support for asynchronous stages with SLURM
- transparent encryption with EncFS
- seamless Docker and Sarus integration
- YAML-based stage policies to guarantee consistency across devs and clean project structure...
- ...while preserving the full power of DVC

**Not part of this talk**, but fits the session.

Get in touch with Lukas Drescher if you are interested in more details.



Encourages a style, where application is agnostic to configuration

- simplifies portability between systems and reasoning
- enables inexperienced people to orchestrate workflows
- no assumptions on particular data layout

Performance benchmark results on Daint (TBD on Alps)

<https://github.com/eth-cscs/async-encfs-dvc>

# Application deployment using CI


On Alps we want to support different application lifecycles:

- Relevance: provide the latest releases of software for users that need them
- Stability: don't force users to update
- Support software for a full project lifecycle (3-4 years)

The Situation today on Piz Daint:

- Regular updates every 6 months – expect to break users' code
- Provided software is often at least 6-12 months out of date
- Supported application upgrades is a manual process

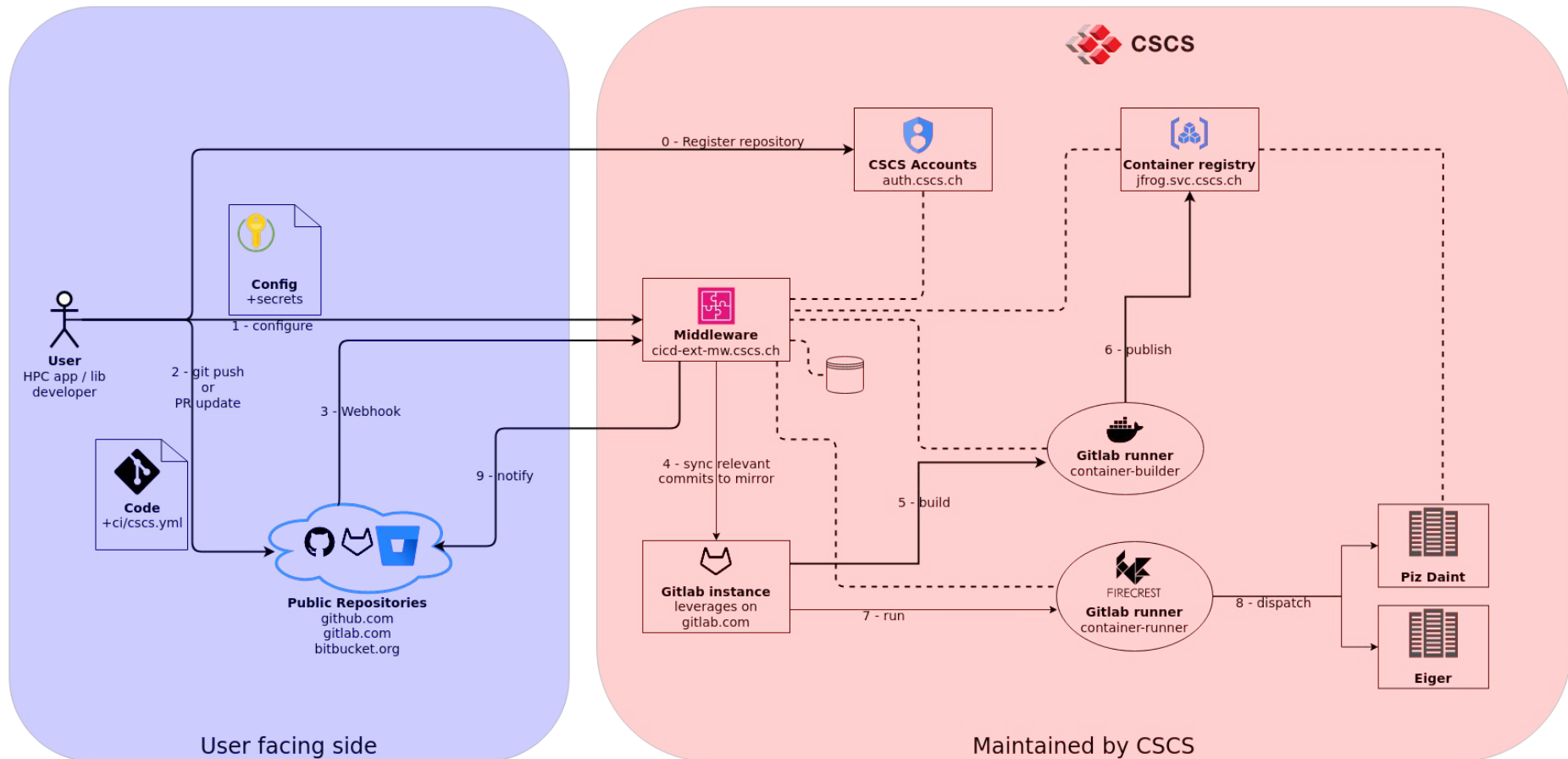
# Application deployment using CI - Objectives

- Fix all of the problems and have happy users 
- System updates should not break user's workflows
- Easy to maintain and easy upgrade/downgrade
- Easy artifact overview
- Easy (and maybe automated) artifact deprecation/deletion
- Extendible and customizable by users
- Not reinventing the wheel

# Application deployment using CI - Key components

- Three key components
  - CI targeting HPC infrastructure
  - User environments providing scientific applications to users
  - Regression testing framework

# CI/CD setup at CSCS



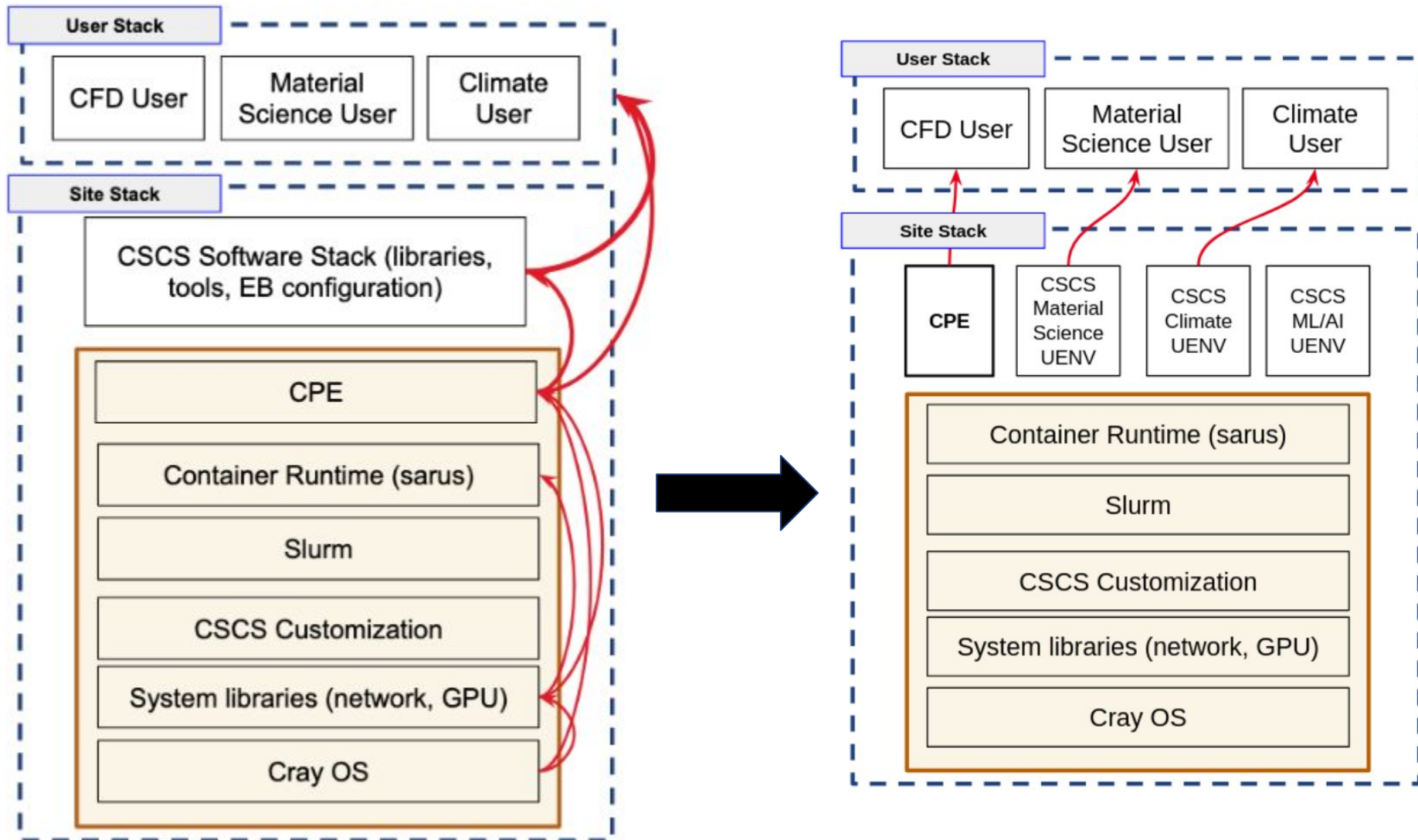
Middleware: <https://gitlab.com/cscs-ci/ci-testing/webhook-ci/middleware-go>

# CI/CD setup at CSCS

- Middleware is the orchestrator, gatekeeper and source of truth
- Gitlab runners (custom executors) communicate with middleware via REST API
- Middleware can be deployed on a server or kubernetes cluster
- At CSCS inside a kubernetes cluster (not Alps hardware)
  - Zero downtime upgrades possible
- Different deployment strategies of runners
  - Container-builder → kubernetes executor on Alps hardware
  - Container-runner (custom executor):
    - Login node of slurm cluster
    - Firecrest dispatcher
  - Baremetal-runner (custom executor):
    - Firecrest dispatcher

Middleware: <https://gitlab.com/cscs-ci/ci-testing/webhook-ci/middleware-go>

# User environments

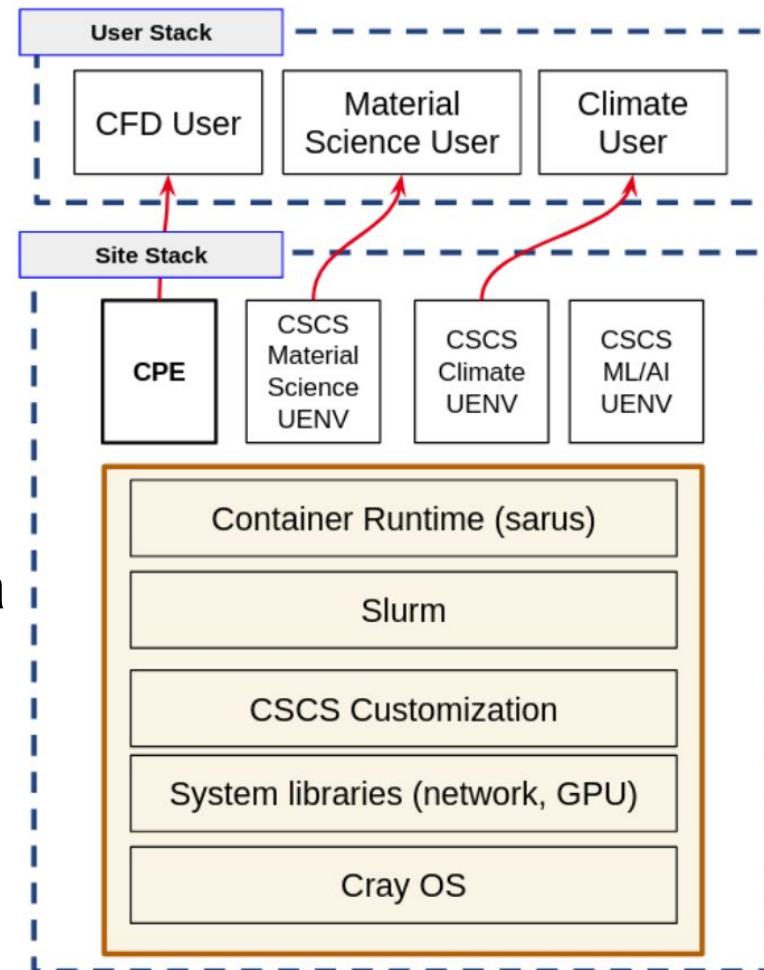




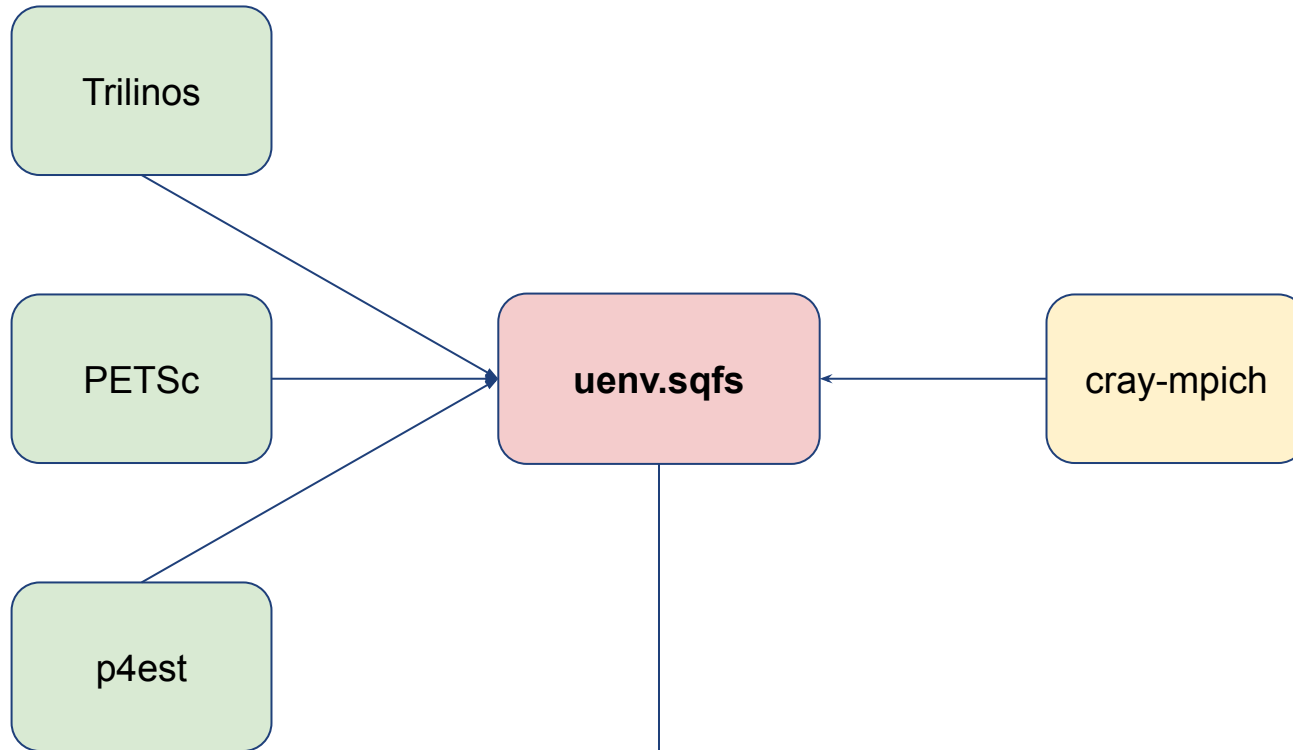
# User environments

- Provide workflow-specific software stacks
- Deployed independently
- Deployed by users and user-support
- Built using spack in memory
- One environment is packaged to a single squashfs-file
- A few yaml files define the recipe for a user environment
- Building a uenv is done by the opinionated tool stackinator:

<https://github.com/eth-cscs/stackinator>



# User environments



# Reframe

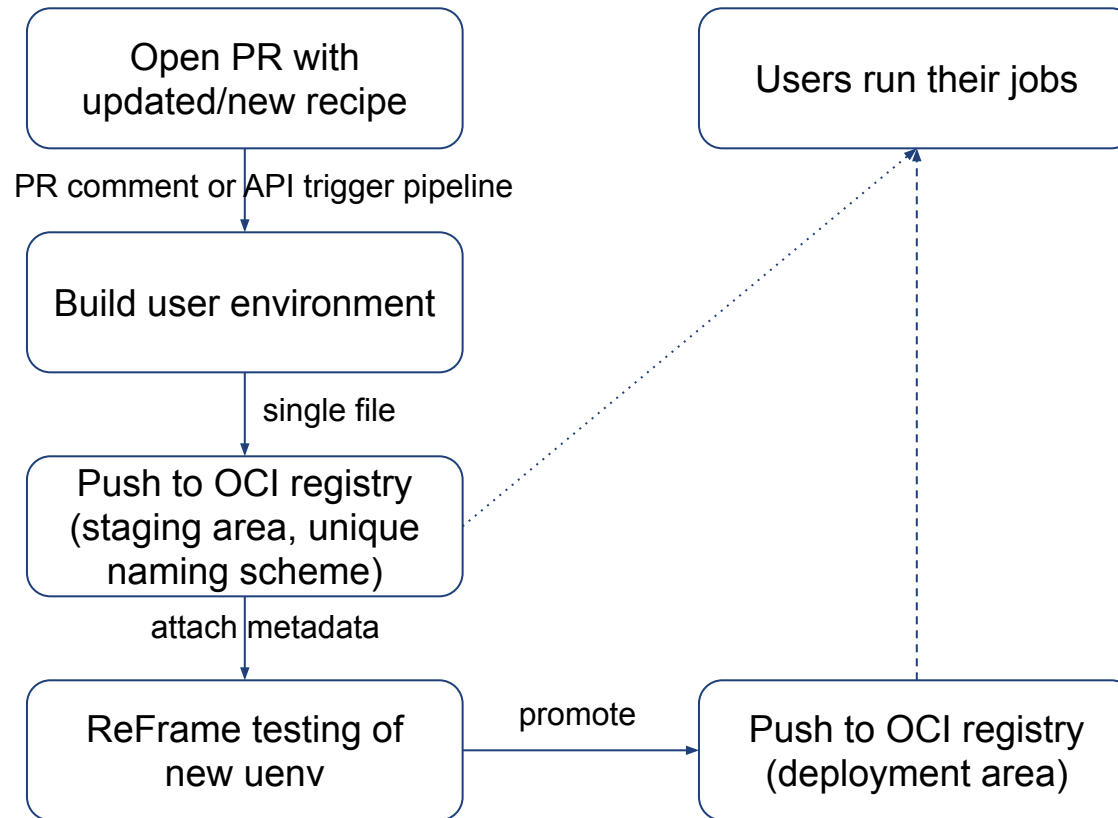
ReFrame is a powerful framework for writing system regression tests and benchmarks, specifically targeted to HPC systems

- Composable tests written in python
- Portable tests in a declarative way
- Multi-dimensional test parameterization
- Parallel test execution
- Support for multiple HPC schedulers, module systems and container runtimes
- Integration with Elastic and Graylog

Reframe: <https://github.com/reframe-hpc/reframe>

Tests: <https://github.com/eth-cscs/cscs-reframe-tests>

# Application deployment via CI/CD pipelines



User environments: <https://github.com/eth-cscs/alps-uenv>

# Application deployment via CI/CD pipelines

```
Code Blame 41 lines (41 loc) · 640 Bytes Raw Copy Download Edit View
```

```
1 nvhpc-env:
2   compiler:
3     - toolchain: llvm
4       spec: nvhpc@22.11
5     - toolchain: gcc
6       spec: gcc@11
7   mpi:
8     spec: cray-mpich
9   gpu: null
10  unify: true
11  specs:
12    - cmake%gcc
13    - cuda@11.8
14    - fftw
15    - openblas
16    - libxc@5.2.3%nvhpc-cuda
17    - quantum-espresso@7.1 %nvhpc +libxc +cuda ~scalapack ^openblas threads=openmp ^libxc
18    - patchelf%gcc
19  variants:
20    - +mpi
21    - +cuda
22    - cuda_arch=80
23  packages:
24    - gmake
25    - m4
26    - perl
27    - git
28    - pkgconf
29    - readline
30    - ncurses
31    - diffutils
32    - libiconv
33    - openssl
34    - sqlite
35    - tar
36    - libxml2
37    - gettext
38  views:
39    default:
40    develop:
41      exclude: ['quantum-espresso']
```

Build user environment

User environments: <https://github.com/eth-cscs/alps-uenv>

# Application deployment via CI/CD pipelines

Code Blame 41 lines (41 loc) · 640 Bytes

```
1  nvhpc-env:
2  compiler:
3  - toolchain: llvm
4  spec: nvhpc@22.11
5  - toolchain: gcc
6  spec: gcc@11
7  mpi:
8  spec: cray-mpich
9  gpu: null
10 unify: true
11 specs:
12 - cmake%gcc
13 - cuda@11.8
14 - fftw
15 - openblas
16 - libxc@5.2.3%nvhpc~cuda
17 - quantum-espresso@7.1 %nvhpc +libxc +cuda ~scalapack ^openblas threads=openmp ^libxc
18 - patchelf%gcc
```

Build user environment

User environments: <https://github.com/eth-cscs/alps-uenv>

# Application deployment via CI/CD pipelines

```
19 variants:
20   - +mpi
21   - +cuda
22   - cuda_arch=80
23 packages:
24   - gmake
25   - m4
26   - perl
27   - git
28   - pkgconf
29   - readline
30   - ncurses
31   - diffutils
32   - libiconv
33   - openssl
34   - sqlite
35   - tar
36   - libxml2
37   - gettext
38 views:
39   default:
40   develop:
41     exclude: ['quantum-espresso']
```

Build user environment

User environments: <https://github.com/eth-cscs/alps-uenv>

# Image Management in Pipelines

The pipeline needs to store images and associated meta data

- Images can be in large (>1GB)
- Meta data is usually small, in the order of kB:
  - JSON descriptions of the image contents, recipe, when and where it was built, etc.
  - ReFrame test results.

We use ORAS to manage artifacts:

- ORAS provides a CLI to distribute artifacts across OCI-compliant registries.
- We leverage the artifact versioning and lifetime management features provided by robust OCI implementations without rolling our own.

We use a JFrog registry, however any DockerHub compliant provider would work:

- OCI-compliant == DockerHub API
- Workflows are not tied to a specific artifact storage service or provider,
- And can directly use services such as GitHub Container Registry (`ghcr.io`)



# Image Management in Pipelines

The CI pipeline stores the squashfs image in the **build** namespace of the registry

- and attaches the image meta data and ReFrame test results

```
# the full spec of the uenv: cluster/micro-architecture/uenv-name/version
$ uenv=eiger/zen2/prgenv-gnu/23.11

# push the image and meta data to OCI registry (JFrog)
$ oras push jfrog.svc.cscs.ch/uenv/build/$uenv:1153388082 \
  --artifact-type application/x-squashfs store.squashfs

# attach meta data to the image
$ oras attach jfrog.svc.cscs.ch/uenv/build/$uenv:1153388082 \
  --artifact-type uenv/meta ${rego}/${repo} ./meta

# attach reframe test results to the image
$ oras attach jfrog.svc.cscs.ch/uenv/build/$uenv:1153388082 \
  --artifact-type uenv/reframe ./reframe_results
```

# Image Management in Pipelines

Deployment by copying from built to the **deploy** namespace of the registry

- the attached meta data and test results

A simple CLI tool that wraps oras for end users, e.g:

- `uenv image pull prgenv-gnu/23.11:latest`

```
# the full spec of the uenv: cluster/micro-architecture/uenv-name/version
$ uenv=eiger/zen2/prgenv-gnu/23.11

# deploy by shallow-copying to the deploy namespace with tag :latest
$ oras copy --recursive jfrog.svc.cscs.ch/uenv/build/$uenv:1153388082 \
    jfrog.svc.cscs.ch/uenv/deploy/$uenv:latest

# the meta data and test results are still attached to the deployed artifact
$ oras discover jfrog.svc.cscs.ch/uenv/deploy/$uenv:latest
Discovered 1 artifact referencing latest
Digest: sha256:1e2d418fe383f793449e61e64c3700de4a07822ee16a89573d78f5e59a781519

Artifact Type Digest
uenv/meta sha256:ebaedae79ce2581c0213ec6a2126fb6d9c88c1f6cfcc0ba9200730fea891f55e
uenv/reframe sha256:f0b94dcaddec23bbd27da328dc23b48c48e8483bca8a0907fd7910aa0458a159

# copy the image to the local file system
$ oras pull -o ./store.squashfs jfrog.svc.cscs.ch/uenv/deploy/$uenv:latest
```

# Application deployment via CI/CD pipelines

- ReFrame inspects metadata
- Runs every test that matches metadata
- Tests are platform independent, the same test can be run in a container, baremetal or with a uenv

ReFrame testing of  
new uenv

Reframe: <https://github.com/reframe-hpc/reframe>

Tests: <https://github.com/eth-cscs/cscs-reframe-tests>

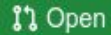
# Application deployment via CI/CD pipelines

- Pushing technically means to retag in the OCI registry
- No additional storage requirement, same artifact with a different name
- Allows all CSCS users to see the new artifact, when they list available user environments
- Upgrade / downgrade from a user perspective means to pull a newer / older version of the user environment
- Eager users can upgrade quickly, conservative stick with the old one for their project runtime

Push to OCI registry  
(deployment area)

User environments: <https://github.com/eth-cscs/alps-uenv>

# Application deployment via CI/CD pipelines



change prgenv-gnu view to only include roots #65

bcumming wants to merge 1 commit into `main` from `view/prgenv-gnu`

Add more commits by pushing to the `view/prgenv-gnu` branch on `eth-cscs/alps-uenv`.



**This branch has not been deployed**

No deployments



**Some checks were not successful**

3 failing and 1 successful checks

[Hide all checks](#)



**cscs/alps**

[Details](#)



**cscs/clariden-a100-prgenv-gnu:23.11**

[Details](#)



**cscs/santis-zen2-prgenv-gnu:23.11**

[Details](#)



**cscs/eiger-zen2-prgenv-gnu:23.11**

[Details](#)



**This branch has no conflicts with the base branch**

Merging can be performed automatically.

Update branch



Squash and merge



or view [command line instructions](#).

# Application deployment via CI/CD pipelines

CSCS-CI / ... / alps-uenv-551234120955960 / alps-uenv-alps / Pipelines / #1157608311

## change prgenv-gnu view to only include roots

✖ Failed CSCS Ciext created pipeline for commit `23c5146c` finished 1 month ago

For `__CSCSCI__pr65`

Child pipeline (parent) latest 2 jobs 9 minutes 14 seconds, queued for 2 seconds

**Pipeline** Needs Jobs 2 Failed Jobs 1 Tests 0

Group jobs by Stage Job dependencies Show dependencies

### Upstream

✖ alps-uenv-alps  
#1157607835  
[Parent](#)

✔ build-prgenv-gnu/23.11-clariden/a100  
build

✖ test-prgenv-gnu/23.11-clariden/a100  
test

# Application deployment via CI/CD pipelines

```
i/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/gpu_burn_build_743a0cc9'  
140 [ FAIL ] ( 2/28) cscs_gpu_burn_check /7f2ccd63 @clariden:nvgpu+store_default  
141 ==> test failed during 'startup': test staged in None  
142 [ OK ] ( 3/28) UENV_BuildDeviceCount ~clariden:nvgpu+store_default /94b16e56 @clariden:nvgpu+store_default  
143 [ RUN ] UENV_NvidiaDeviceCount /fb5e8cb7 @clariden:nvgpu+store_default  
144 [ OK ] ( 4/28) OSULatencyCuda /42896efa @clariden:nvgpu+store_default  
145 P: latency_256: 3.75 us (r:3.75, l:None, u:0.15)  
146 P: latency_4M: 184.92 us (r:180.0, l:None, u:0.15)  
147 [ OK ] ( 5/28) OSUBandwidthCuda /9b75552e @clariden:nvgpu+store_default  
148 P: bandwidth_256: 326.03 MB/s (r:200.0, l:-0.15, u:None)  
149 P: bandwidth_4M: 23482.23 MB/s (r:24000.0, l:-0.15, u:None)  
150 [ OK ] ( 6/28) OSUBandwidth /bd272703 @clariden:nvgpu+store_default  
151 P: bandwidth_256: 611.5 MB/s (r:600.0, l:-0.5, u:None)  
152 P: bandwidth_4M: 23888.14 MB/s (r:24000.0, l:-0.15, u:None)  
153 [ OK ] ( 7/28) OSULatency /1583f36f @clariden:nvgpu+store_default  
154 P: latency_256: 2.39 us (r:2.3, l:None, u:0.5)  
155 P: latency_4M: 180.94 us (r:180.0, l:None, u:0.15)  
156 [ FAIL ] ( 8/28) StreamTest /cdf4820d @clariden:nvgpu+store_default  
157 ==> test failed during 'sanity': test staged in '/iopsstor/scratch/cscs/jenkins/gilab-runner/builds/f_6tds4z/0/cscs-ci/ci-testing/webhook-ci/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/StreamTest'  
158 [ OK ] ( 9/28) UENV_NvidiaDeviceCount /fb5e8cb7 @clariden:nvgpu+store_default  
159 [ OK ] (10/28) UENV_CudaSamples %sample=conjugateGradientCudaGraphs /288cbcb8 @clariden:nvgpu+store_default  
160 [ OK ] (11/28) UENV_CudaSamples %sample=simpleCUBLAS /a095f959 @clariden:nvgpu+store_default  
161 [ OK ] (12/28) UENV_CudaSamples %sample=bandwidthTest /76d402e3 @clariden:nvgpu+store_default  
162 [ OK ] (13/28) UENV_CudaSamples %sample=deviceQuery /6fed6c95 @clariden:nvgpu+store_default  
163 [ FAIL ] (14/28) UENV_NVMLCheck /4dfb557a @clariden:nvgpu+store_default  
164 ==> test failed during 'sanity': test staged in '/iopsstor/scratch/cscs/jenkins/gilab-runner/builds/f_6tds4z/0/cscs-ci/ci-testing/webhook-ci/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/UENV_NVMLCheck'  
165 [ FAIL ] (15/28) MpiInitTest /13bc6cef @clariden:nvgpu+store_default  
166 ==> test failed during 'sanity': test staged in '/iopsstor/scratch/cscs/jenkins/gilab-runner/builds/f_6tds4z/0/cscs-ci/ci-testing/webhook-ci/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/MpiInitTest'  
167 [ FAIL ] (16/28) HelloWorldTestMPIOpenMP %linking=dynamic %lang=F90 /0903dadac @clariden:nvgpu+store_default  
168 ==> test failed during 'sanity': test staged in '/iopsstor/scratch/cscs/jenkins/gilab-runner/builds/f_6tds4z/0/cscs-ci/ci-testing/webhook-ci/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/HelloWorldTestMPIOpenMP_0903dadac'  
169 [ FAIL ] (17/28) HelloWorldTestMPIOpenMP %linking=dynamic %lang=cpp /4cf69907 @clariden:nvgpu+store_default  
170 ==> test failed during 'sanity': test staged in '/iopsstor/scratch/cscs/jenkins/gilab-runner/builds/f_6tds4z/0/cscs-ci/ci-testing/webhook-ci/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/HelloWorldTestMPIOpenMP_4cf69907'  
171 [ FAIL ] (18/28) HelloWorldTestMPIOpenMP %linking=dynamic %lang=c /486d764c @clariden:nvgpu+store_default  
172 ==> test failed during 'sanity': test staged in '/iopsstor/scratch/cscs/jenkins/gilab-runner/builds/f_6tds4z/0/cscs-ci/ci-testing/webhook-ci/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/HelloWorldTestMPIOpenMP_486d764c'  
173 [ FAIL ] (19/28) HelloWorldTestMPI %linking=dynamic %lang=F90 /17d00b3c @clariden:nvgpu+store_default  
174 ==> test failed during 'sanity': test staged in '/iopsstor/scratch/cscs/jenkins/gilab-runner/builds/f_6tds4z/0/cscs-ci/ci-testing/webhook-ci/mirrors/551234120955960/1440398897047560/6052925692/stage/clariden/nvgpu/store_default/HelloWorldTestMPI_17d00b3c'
```

Duration: 3 minutes 33 seconds  
Finished: 1 month ago  
Queued: 1 second  
Timeout: 1h (from project) [?](#)  
Runner: #22823746 (f\_Gtds4z-) ci-ext  
gitlab-runner on Clariden

Tags: [clariden-login-baremetal](#)

Commit [23c5146c](#) [🔗](#)  
change prgenv-gnu view to only  
include roots

Pipeline #1157608311 ✖ Failed for  
[\\_\\_CSCSCI\\_\\_pr65](#) [🔗](#)

test [▼](#)

Related jobs

→ ✖ test-prgenv-gnu/23.11-clar...

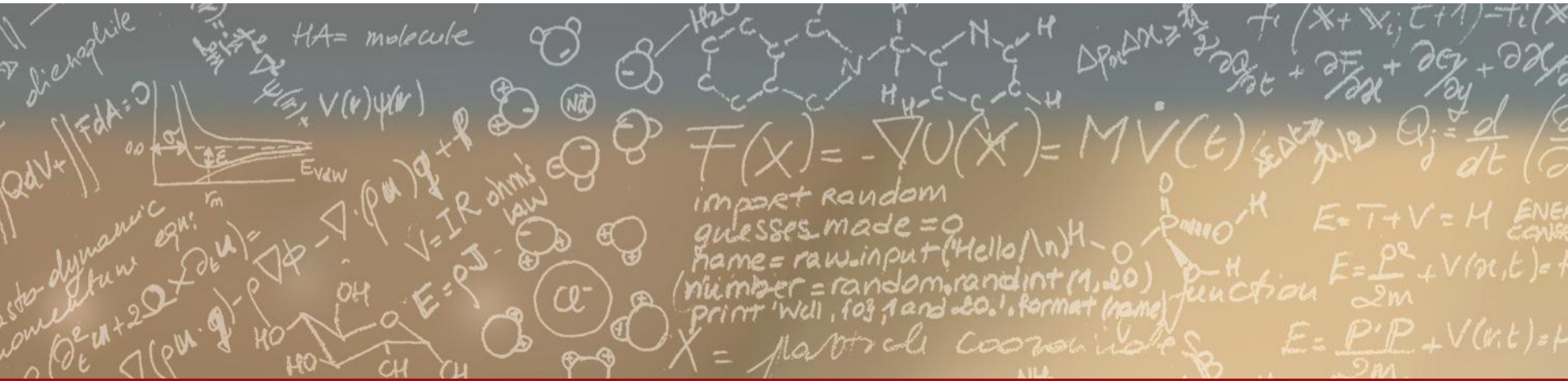




CSCS

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

ETH zürich



**Thank you for your attention.**  
**Slides inspired by presentations by Ben Cumming**  
**and Theofilos Manitaras**



# Application deployment via CI/CD pipelines

- Leverage *oras* (OCI registry as storage), to push squashfs image to JFrog
- Metadata is attached to the same OCI object
- Metadata describes software that is supposed to be inside uenv
- Naming scheme in staging area is unique, i.e. every built is kept, and can be reproducibly tested again
- Allows early-access users to test potential bugfixes without the need of deploying it

The logo for ORAS (OCI Registry As a Service) is displayed in a stylized, colorful font. The letters are outlined in blue and filled with various colors: 'O' is red, 'R' is green, 'A' is white, and 'S' is yellow.

Push to OCI registry  
(staging area, unique  
naming scheme)

User environments: <https://github.com/eth-cscs/alps-uenv>