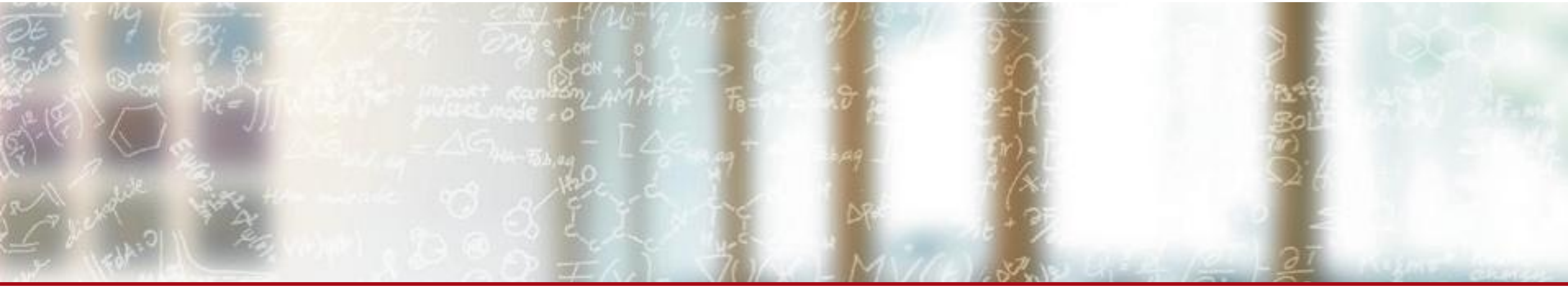
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



User Environments on Alps

Ben Cumming
September 2, 2022

Who am I?

Today I will be presenting new approaches for providing programming environments in development for User Lab on **Alps in 2023**

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Introduction

The Piz Daint user environment: one size fits all

Interactions with Daint use the well-established, tried-and-trusted HPC interface:

- Log in via ssh to a shell with a programming environment loaded
- Manipulate the environment using modules
  - Select Cray-provided compilers, libraries and tools
  - Select CSCS-provided libraries, tools and applications
- Use shared resources including storage and job scheduling.

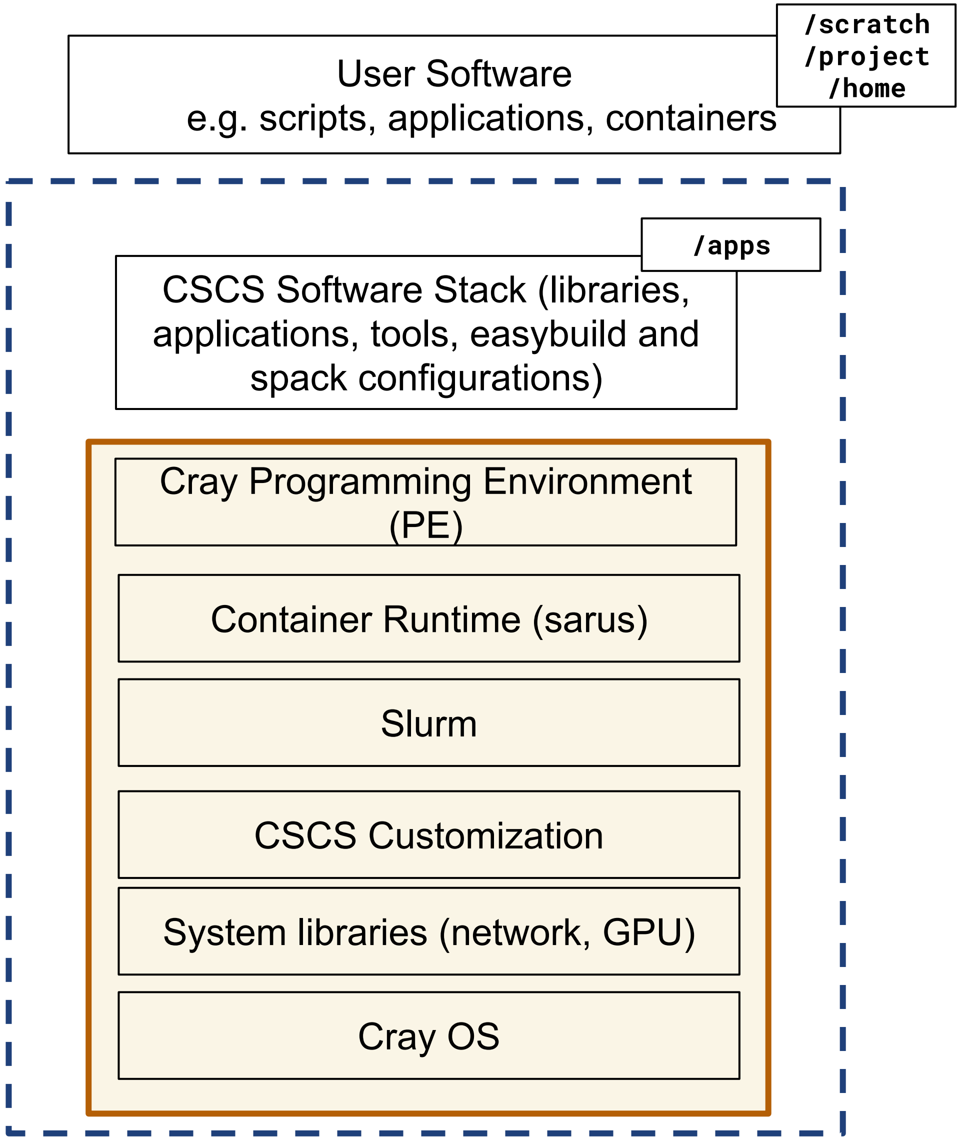
All workflows, such as JupyterLab, FirecREST present a layer over this

- All access goes a standard user account

As the number and variety of use cases expand

is the “one size fits all model” a good fit?

The user environment on Daint: layer cake


/scratch
/project
/home
User Software
e.g. scripts, applications, containers
/apps
CSCS Software Stack (libraries, applications, tools, easybuild and spack configurations)
Cray Programming Environment (PE)
Container Runtime (sarus)
Slurm
CSCS Customization
System libraries (network, GPU)
Cray OS

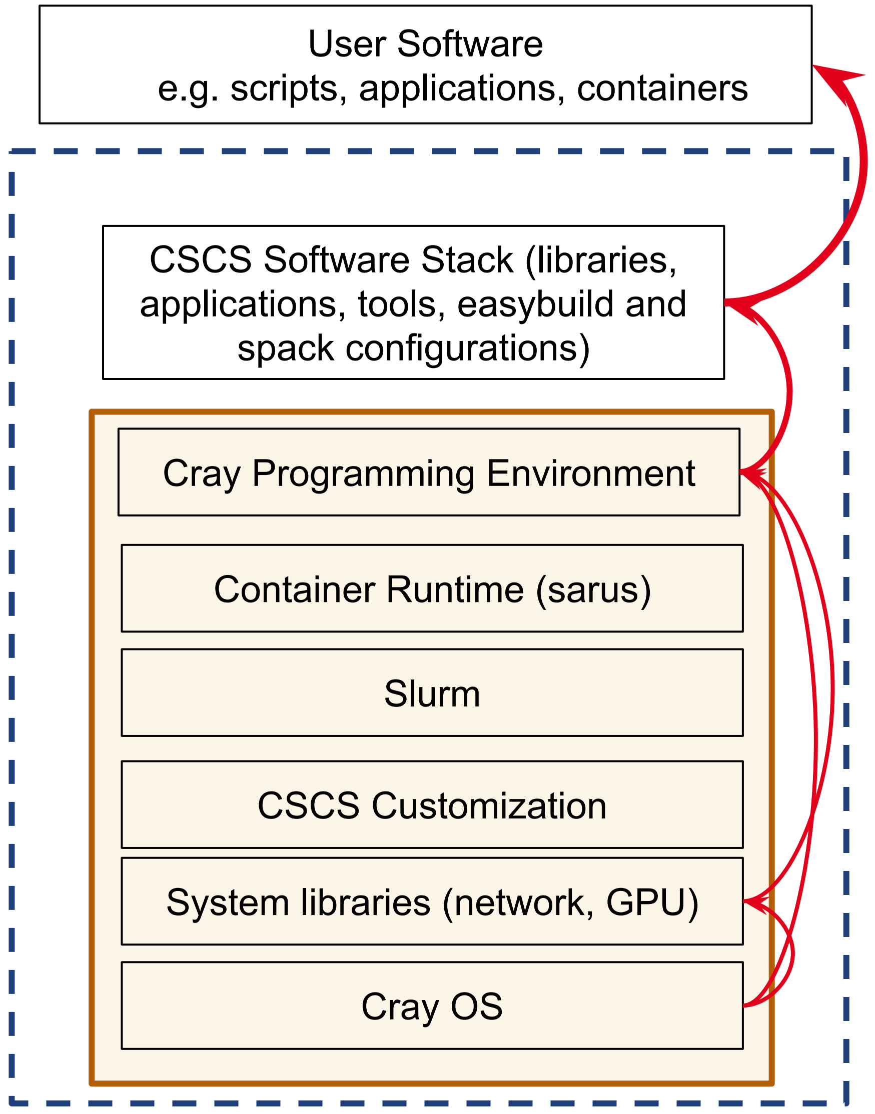
1. Essential services are installed on top of Cray OS
2. The Cray PE is installed as part of the underlying "node image"
3. CSCS provides a rich set of software products built with the Cray PE
4. Users build their software and workflows on top of that.

A change to one layer during an upgrade affects every layer above:

- A new Cray PE change often requires rebuilding the CSCS stack and user software

CSCS has developed tooling (ReFrame) and a very comprehensive test suite and CI/CD for maintaining the quality of this integration

Maintenance challenges


User Software
e.g. scripts, applications, containers
CSCS Software Stack (libraries, applications, tools, easybuild and spack configurations)
Cray Programming Environment
Container Runtime (sarus)
Slurm
CSCS Customization
System libraries (network, GPU)
Cray OS

The Cray Programming Environment is complex by necessity:

- Modules provide a combinatorial set of libraries and tools that serve as many use cases as possible on an increasing number of hardware types.
- Integration is provided by HPE: once an issue is identified HPE have to fix the issue in a future release
  - Long latency between issues reporting and the fix available on Daint.
- Each new release requires extensive testing to check that issues have been fixed
  - And to identify the inevitable new issues

The Cray PE is the best configuration from any vendor in my experience.

These challenges affect all HPC clusters.

Stability vs. Bug Fixes and New Features

Any feedback that in your opinion can help us improve the HPC environment?

Would it be possible to keep older versions?

I am very satisfied with the HPC environment on Piz Daint

Compilers that support the newest C++ standards as well as possible

I need a stabler environment: older versions of the software tools disappear too quickly, which means I have to rebuild my stack every few months.

Please regularly update C++ and CUDA compilers

By providing an environment on CPE it is very difficult to meet all requirements

- Regular updates are required to fix bugs, maintain security and provide updated versions of tools.
- The latest versions of compilers can't be installed before they are packaged by HPE and tested by CSCS.
- It is impractical to maintain:
  - Full stacks on top of more than one CPE
  - More than 2-3 CPE on a system

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Alps

Alps: Bespoke Research Infrastructure

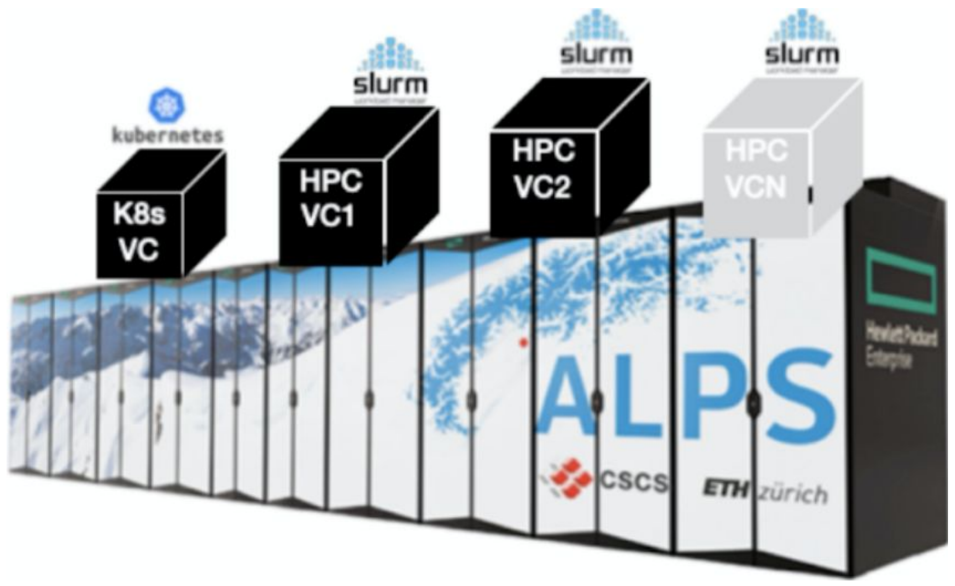
The new capabilities of Cray Shasta/CSM allow CSCS to expand and improve our offering on a single infrastructure

CSCS will provision separate virtualized clusters for different use cases

- Slurm and k8s with independent and customised configurations;
- Network and storage isolation;
- Different QoS for each system.

Types of systems include:

- User Lab cluster;
- Clusters attached to experiments;
- Scientific software development clusters;
- Application-specific clusters (CI/CD, W&C);
- CI/CD for scientific applications.



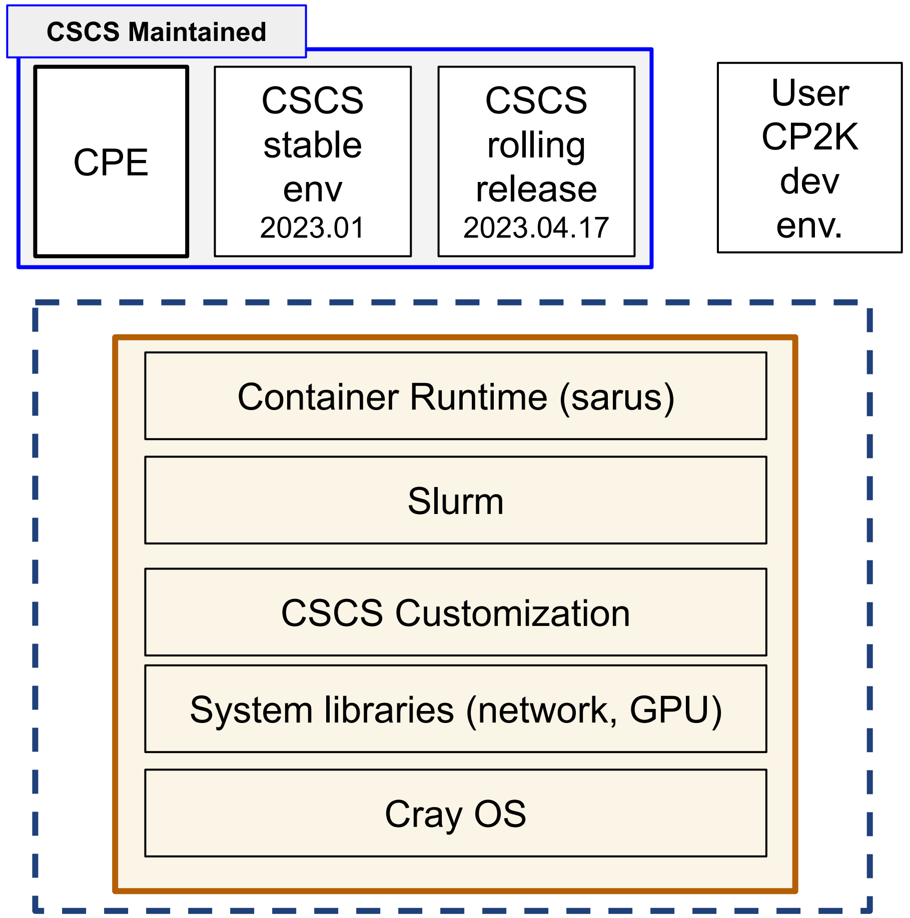
Opportunity and Necessity Call

HPE Cray EX systems provide a more flexible platform that uses lessons and technology from cloud providers

This presents an an opportunity to explore how to provide user environments

To meet the needs of more use cases and maintain quality of service this is a necessity

The User Environment Initiative



Provide multiple user environments

- Login to a clean environment
  - Cray OS with slurm + container runtime + drivers
- Multiple environments are available:
  - The familiar CPE with `env-load craype-22.08`
  - CSCS-provided user environments
  - User-built user environments
- Each environment is contained in a single file
  - Shared in an artifactory or stored on a filesystem.

The environments are **independent**

The environments are built on top of the base-image - not the Cray PE.

Boundary conditions

Complete stability can’t be guaranteed over a project lifetime:

- The underlying CrayOS + drivers + slurm + container runtime require patching for security, fixes and features
  - However these changes are fewer and less likely to impact
- Updates to CPE are more significant
  - Versions of software change
    - Old versions are deprecated
    - New versions are added
    - Default versions change
  - The behavior of CPE changes is subtle ways that have knock-on effects

The aim is to reduce the frequency and impact of changes at lower layers on user environments

- By building on top of solid foundations.

Status of work

User Environments in Practice

There are 3 components:

1. A standard approach for packaging environments
   - Not tied to any method for building or describing environments (e.g. Modules, Spack, Easybuild)
   - Simple to understand.
   - Simple to copy, store and version.
2. Tooling to describe and build environments
   - Provide a flat description of an environment as code
   - Efficiently build a packaged environment from the description
   - **Reproducible** for a (description, base image) pair
3. Tooling to load a user environment
   - Simple: one command to load an environment
   - Can be integrated with SLURM
   - Easy use of debuggers and profilers

Packaging and Loading an Environment

Environments are stored in a single compressed file that contains a directory structure.

Starting an environment: `squashfs-run $SCRATCH/gromacs-cuda.squashfs bash`

1. Mount: the directory tree in a fixed `/user-environment` path
2. Execute: a built-in prologue script that configures the environment
3. Launch: a shell in the new environment

Some key properties

- Environments are simply a compressed directory tree
  - Can be created with Spack, Easybuild, Conda or by hand.
- Per-process: other users on the system can mount their own environments without affecting one-another.
- Much lower latency for configure-then-make development workflows than HPC file systems.

Creating an environment with Spack

CSCS is developing tooling to build bespoke programming environments with compilers, MPI, and libraries defined in a git repository

- A matrix of compiler versions and libraries (e.g. MPI, netcdf, fftw) are described in YAML files.

Fast builds: a full stack with multiple gcc and clang versions, NVIDIA HPC SDK, libraries, MPI, and tools in less than an hour.

Environments can be versioned:

- rebuilt when the underlying node image is updated.

CSCS and users can configure their own environments

- CSCS can offer stable and cutting edge environments in our service catalogue

github.com/eth-cscs/spack-stack

Ongoing work

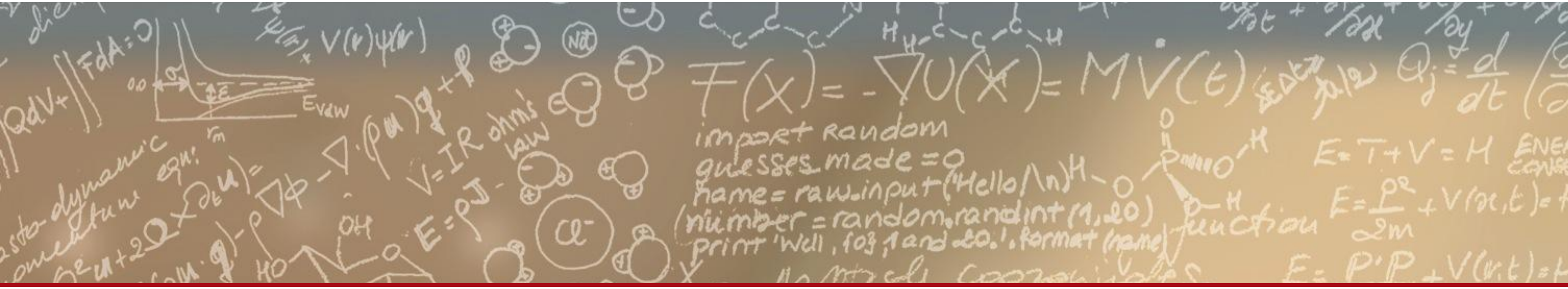
CSCS is busy working on making user environments easy to use and intuitive user experience.

Planned features include:

- Slurm integration
- CI/CD for automatic build and push to the CSCS artifactory when:
  - The target node description is updated
  - The software description is updated
- Optimised communication libraries for SlingShot 11
  - GPU-aware MPI
  - NCCL, RCCL, NVSHMEM, etc.
- Environments for Python and Julia.
- Simplifying the Spack description of programming environments
- Test drive with PASC software development teams at CSCS

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Thank you!

Any questions?

What about Containers?

CSCS is investing heavily in container runtimes and container-first workflows

- Containers provide an excellent mechanism for encapsulating a set of dependencies and their configuration independent of the host OS.

HPC Containers are not completely independent of the underlying system

- They need a way to access system-dependent libraries (MPI) and drivers for bare-metal performance.
- CSCS are collaborating with vendors and other HPC centers to improve portability, e.g. for containers with different flavours of MPI.