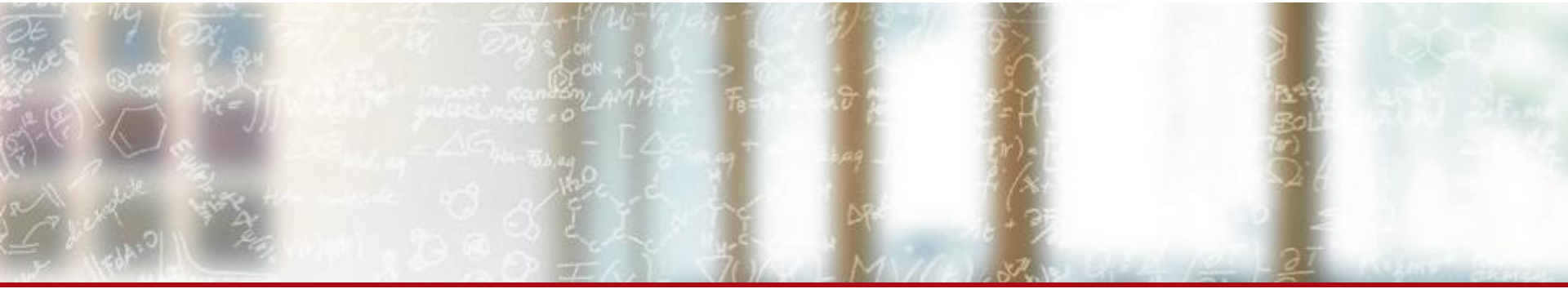
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Containers-First User Environments on HPE Cray EX

Felipe A. Cruz, Alberto Madonna
cruz@cscs.ch
May 2024

Overview

- Alps Research Infrastructure at CSCS
- On Managing User Environments in HPC systems
- Containers-First
  - The Approach
  - The Architecture
- Pathfinder Implementation
  - Environment Definition File
  - Container Ecosystem Integration for HPC
  - Running example
  - Enhancing functionality and usability
- Use-cases: AI

Alps Research Infrastructure

Image generated by ChatGPT

ALPS

- General-purpose compute and data Research Infrastructure
  - One Infrastructure
  - Many Uses
- Enabled by vCluster Tech
  - Versatile Clusters
  - Tailored solutions
  - Infrastructure as Code

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

On Managing User Environments for HPC vClusters

On Managing User Environments in HPC systems

Difficult and getting harder

- **Integration Complexity**
- Flexibility vs. Stability
- Efficiency in Deployment

Image generated by ChatGPT

On Managing User Environments in HPC systems

Difficult and getting harder

- Integration Complexity
- **Flexibility vs. Stability**
- Efficiency in Deployment

Image generated by ChatGPT

On Managing User Environments in HPC systems

Difficult and getting harder

- Integration Complexity
- Flexibility vs. Stability
- **Efficiency in Deployment**

Image generated by ChatGPT

Containers-First Approach

Containers-First Approach

Image generated by ChatGPT

- **Decoupling the User Environment**
- Enhancing Flexibility and Efficiency
- Increasing Consistency

Containers-First Approach

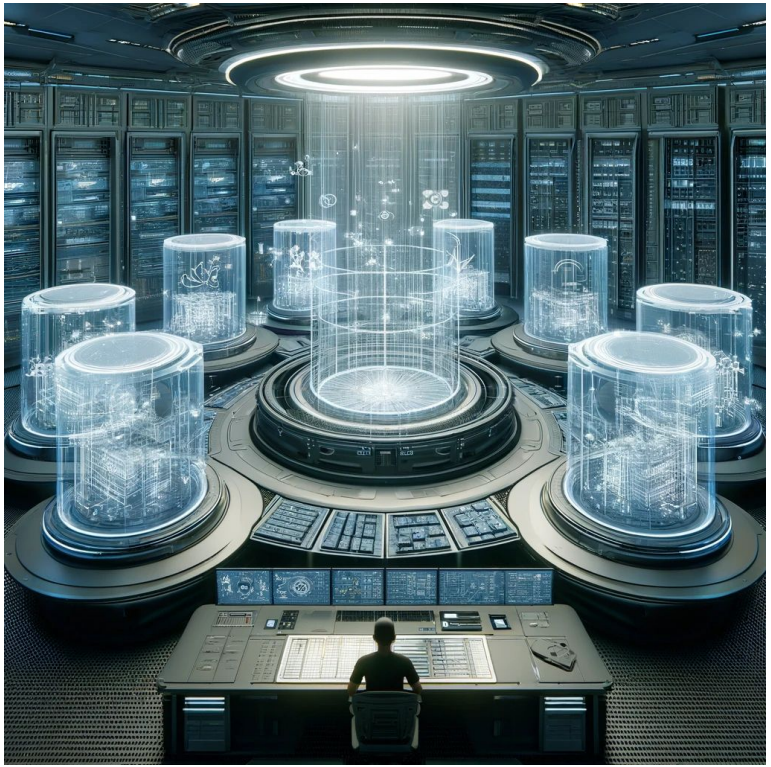


Image generated by ChatGPT

- Decoupling the User Environment
- **Enhancing Flexibility and Efficiency**
- Increasing Consistency

Containers-First Approach

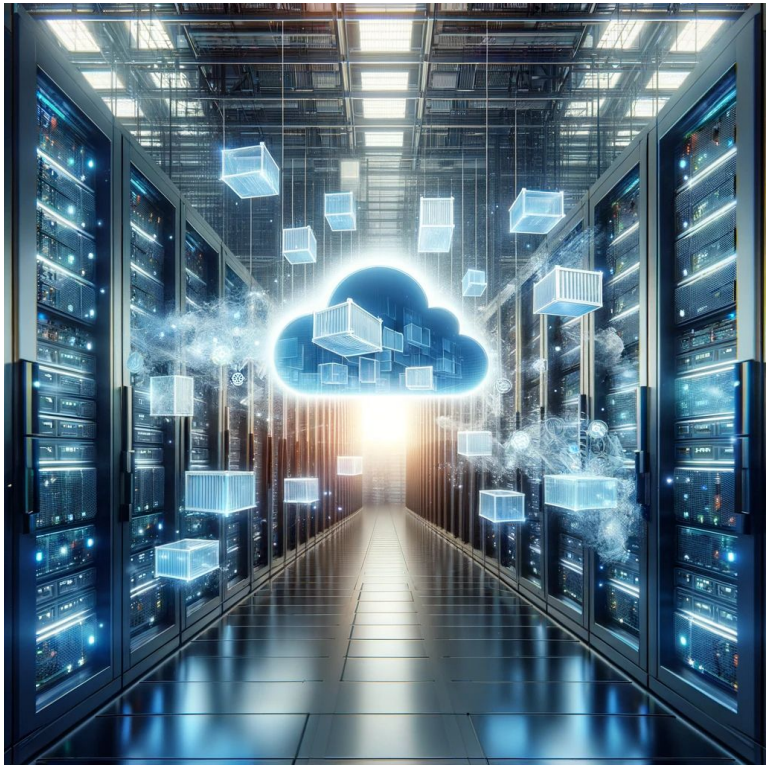


Image generated by ChatGPT

- Decoupling the User Environment
- Enhancing Flexibility and Efficiency
- **Increasing Consistency**

Containers-First Main Features

- **Leverage Containers**
  - Isolation of Environments
  - Reproducibility and Consistency
  - Efficient Deployability
  - Extended Compatibility
  - Flexibility for Experimentation
- Integration with HPC Systems
- Frictionless User Environments

Containers-First Main Features

- Leverage Containers
  - Isolation of Environments
  - Reproducibility and Consistency
  - Efficient Deployability
  - Extended Compatibility
  - Flexibility for Experimentation
- **Integration with HPC Systems**
- Frictionless User Environments

Containers-First Main Features

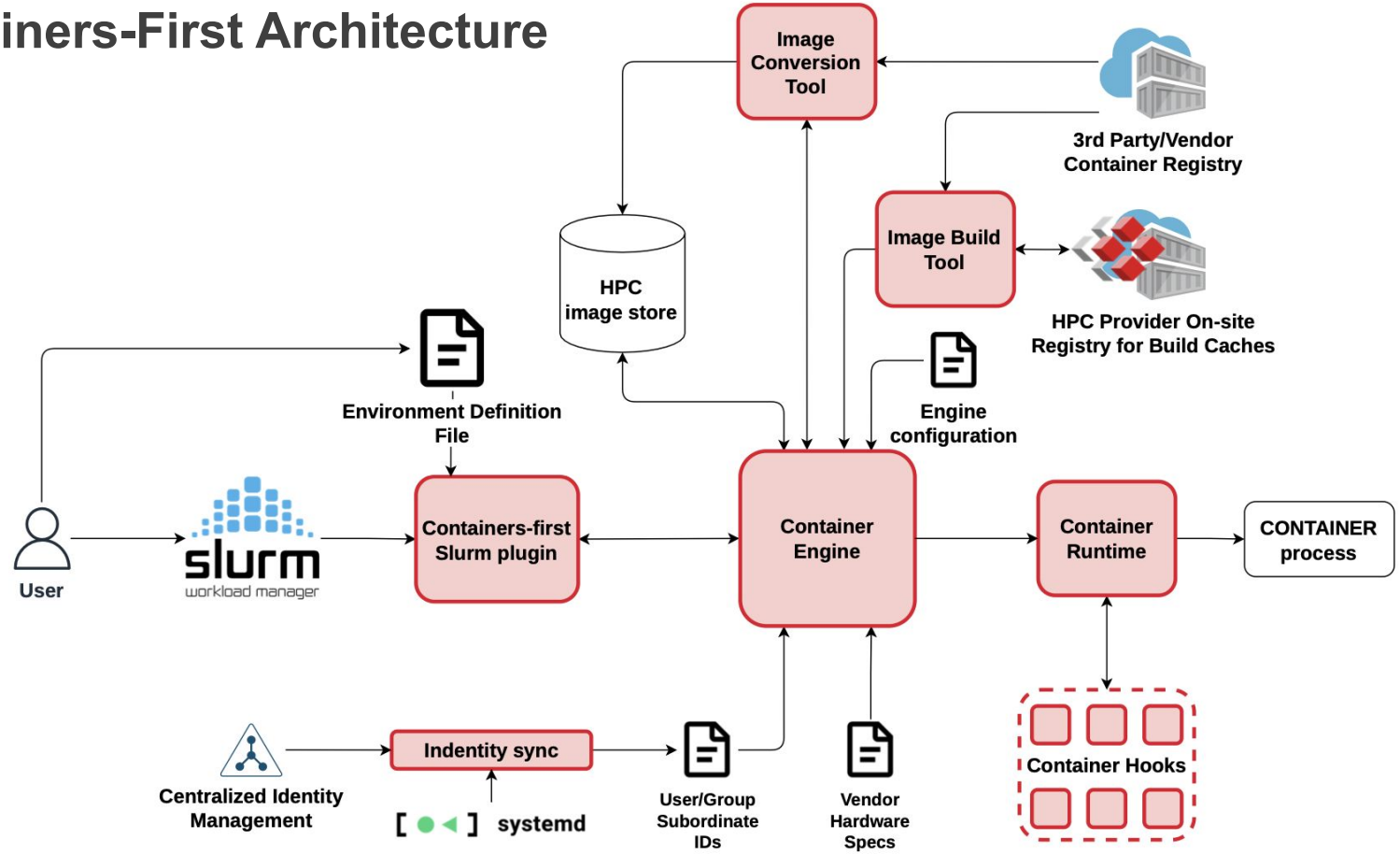
- Leverage Containers
  - Isolation of Environments
  - Reproducibility and Consistency
  - Efficient Deployability
  - Extended Compatibility
  - Flexibility for Experimentation
- Integration with HPC Systems
- **Frictionless User Environments**

Containers-First Architecture

Containers-First Architecture

- Container Ecosystem Tools
  - Runtime
  - Workload manager integration
  - Image Build
  - Image Conversion
- Environment Definition File
- Container Hooks

Containers-First Architecture



Pathfinder Implementation

Environment Definition File – Highlights

- Key Concept for Containers-First
- Standardization and Simplification
- Reproducible Environments
- Customization and Extensibility
- Integration with HPC Tools

Example EDF

```
base_environment = [
    "parent_env0",
    "research_group_env"
]
image = "nvcr.io/nvidia/nvhpc:23.11-devel-
    cuda_multi-ubuntu22.04"
containerfile = "./Containerfile.devtools"
workdir = "/workdir"
entrypoint = false
writable = true
mounts = [
    "/user/experiment:/workdir",
    "/configs:/workdir/configs",
    "${SCRATCH}/data:/workdir/data"
]

[env]
MELLANOX_VISIBLE_DEVICES = "none"
CONFDIR = "/workdir/configs"
UNSET_BEFORE_RUNNING = ""

[annotations]
com.hooks.cxi.enable = true
com.hooks.ofi_nccl.version = "cuda12"
```

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Line 1 - 4: Defining a base environment from other Env.
- Inherit settings from parents
- Build upon existing environments

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Line 5: Specifies the container image to use.
- Example:
  - NVIDIA HPC container
  - from NVIDIA's container registry
  - development tools and CUDA support, based on Ubuntu 22.04

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Line 6: Points to a Containerfile that defines how to build the container image.
- Basically specify a Dockerfile
- Build dockerfile rather than pulled

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Line 7: Sets the working directory inside the container.
- When the container starts, this is the directory from which it will operate.

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Line 8: Disables the default entrypoint script provided by the container image.
- Override the start-up behavior of the container.

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Line 9: Allows the file system within the container to be writable.
- By default containers are immutable
- Enable modifications at runtime

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Lines 10-14: Specifies bind mounts, mapping directories from the host system into the container.

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Lines 16-19: Sets environment variables within the container.
- To configure how applications behave inside the container.

Example EDF (non-exhaustive)

```
1  base_environment = [
2      "parent_env0",
3      "research_group_env"
4  ]
5  image = "nvcr.io/nvidia/nvhpc:23.11-devel-
       cuda_multi-ubuntu22.04"
6  containerfile = "./Containerfile.devtools"
7  workdir = "/workdir"
8  entrypoint = false
9  writable = true
10 mounts = [
11     "/user/experiment:/workdir",
12     "/configs:/workdir/configs",
13     "${SCRATCH}/data:/workdir/data"
14 ]
15
16 [env]
17 MELLANOX_VISIBLE_DEVICES = "none"
18 CONFDIR = "/workdir/configs"
19 UNSET_BEFORE_RUNNING = ""
20
21 [annotations]
22 com.hooks.cxi.enable = true
23 com.hooks.ofi_nccl.version = "cuda12"
```

- Lines 21-23: Annotations that can be used to trigger hooks.
  - CXI
  - NCCL

Container Ecosystem Integration for HPC

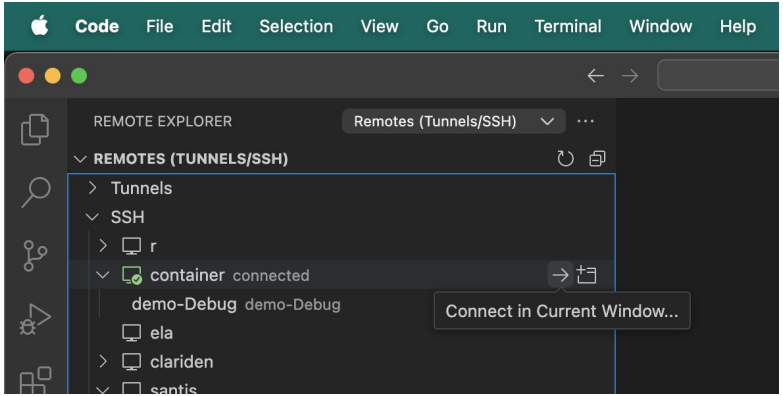
- WLM integration and runtime based on Pyxis and Enroot project
- WLM enhancements
  - TOML parsing for EDF support
  - `--environment` option
  - `EDF_PATH` variable
  - Advanced feature support
- Runtime enhancements
  - Direct image deployment
  - Layer caching
- User workflow simplification
  - Extended tooling integration
  - Extended container capabilities via hooks

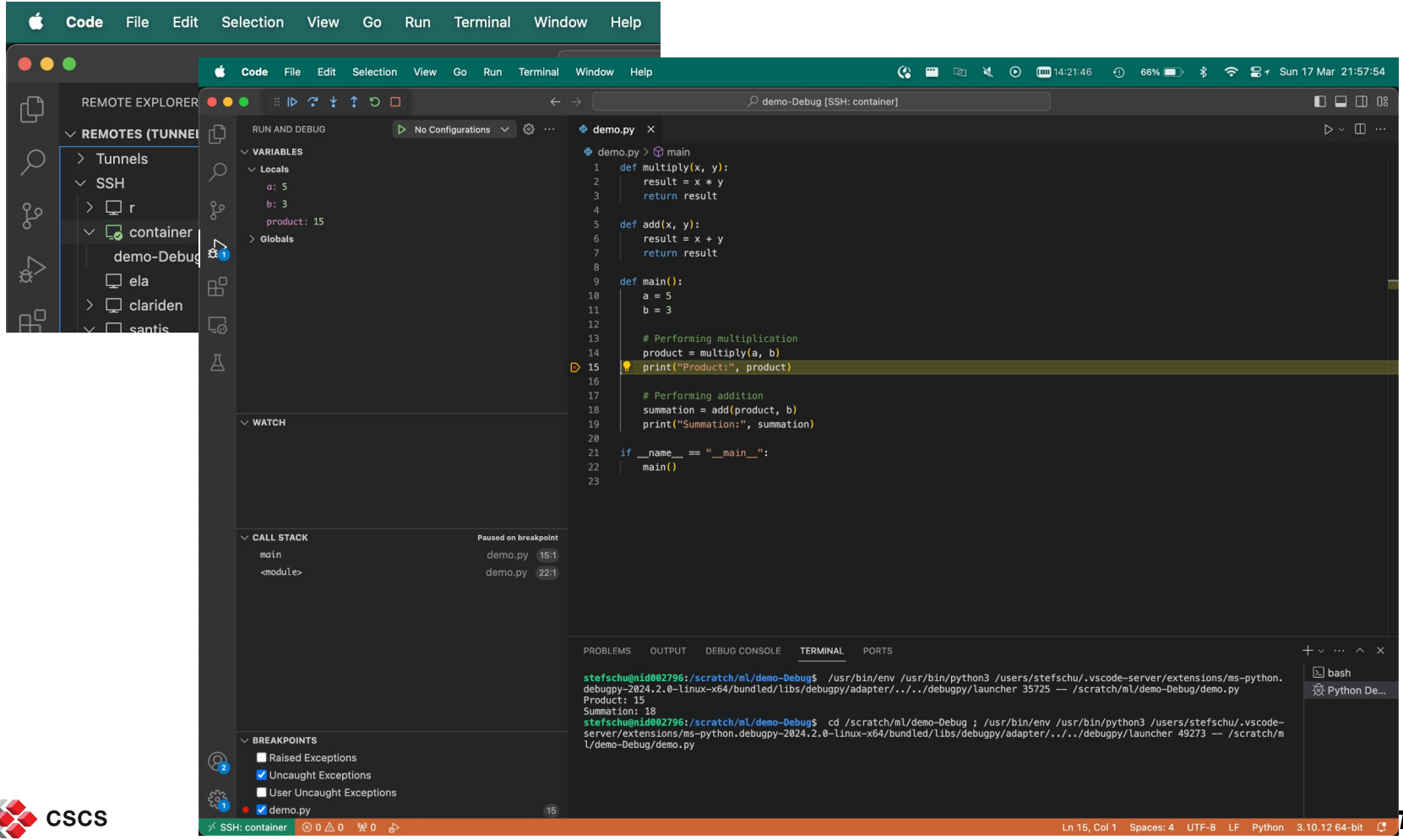
Running example

```
1 base_environment = "pytorch-24.01"
2
3 workdir = "/scratch/ml/demo-Debug"
4 writable = true
5
6 [annotations.com.hooks.ssh]
7 enabled = "true"
8 port = "51234"
9 authorize_ssh_key = "${HOME}/.ssh/cscs-key.
      pub"
```

```
$ srun --environment=pytorch.toml --pty bash
```

- On instantiation, statically linked SSH server starts as daemon in the container
- VS Code editor and debugger available
  - Edit, run, break and inspect the program state on CN


Code File Edit Selection View Go Run Terminal Window Help
REMOTE EXPLORER
Remotes (Tunnels/SSH)
REMOTES (TUNNELS/SSH)
Tunnels
SSH
r
container connected
demo-Debug demo-Debug
Connect in Current Window...
ela
clariden
santis


Code File Edit Selection View Go Run Terminal Window Help
REMOTE EXPLORER
REMOTES (TUNNEL
Tunnels
SSH
r
container
demo-Debug
ela
clariden
santis
Code File Edit Selection View Go Run Terminal Window Help
14:21:46
66%
Sun 17 Mar 21:57:54
demo-Debug [SSH: container]
RUN AND DEBUG
No Configurations
VARIABLES
Locals
a: 5
b: 3
product: 15
Globals
WATCH
CALL STACK
Paused on breakpoint
main demo.py 15:1
<module> demo.py 22:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
demo.py 15
demo.py
demo.py > main
1 def multiply(x, y):
2 result = x * y
3 return result
4
5 def add(x, y):
6 result = x + y
7 return result
8
9 def main():
10 a = 5
11 b = 3
12
13 # Performing multiplication
14 product = multiply(a, b)
15 print("Product:", product)
16
17 # Performing addition
18 summation = add(product, b)
19 print("Summation:", summation)
20
21 if __name__ == "__main__":
22 main()
23
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
Python De...
stefschu@nid002796:/scratch/ml/demo-Debug$ /usr/bin/env /usr/bin/python3 /users/stefschu/.vscode-server/extensions/ms-python.debugpy-2024.2.0-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 35725 -- /scratch/ml/demo-Debug/demo.py
Product: 15
Summation: 18
stefschu@nid002796:/scratch/ml/demo-Debug$ cd /scratch/ml/demo-Debug ; /usr/bin/env /usr/bin/python3 /users/stefschu/.vscode-server/extensions/ms-python.debugpy-2024.2.0-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 49273 -- /scratch/ml/demo-Debug/demo.py
SSH: container
0 0
0
Ln 15, Col 1 Spaces: 4 UTF-8 LF Python 3.10.12 64-bit

Enhancing functionality and usability

Type of Hook	Description
Slurm	Integrates Slurm workload management capabilities within containers, allowing for the use of Slurm commands directly in containerized jobs.
Libfabric for Slingshot 11	Ensures compatibility with the HPE Slingshot 11 high-speed network, enabling high-performance networking capabilities.
NVIDIA GPU	Provides access to NVIDIA GPUs, crucial for simulations and deep learning models.
OFI NCCL Plugin	Integrates the NVIDIA Collective Communications Library (NCCL) with libfabric, for inter-GPU communications for distributed computing tasks.
SSH	Enables SSH access within containers, facilitating remote debugging and management of containerized applications.
MPI Replacement	Replaces the container's MPI installation with an ABI-compatible version from the host.

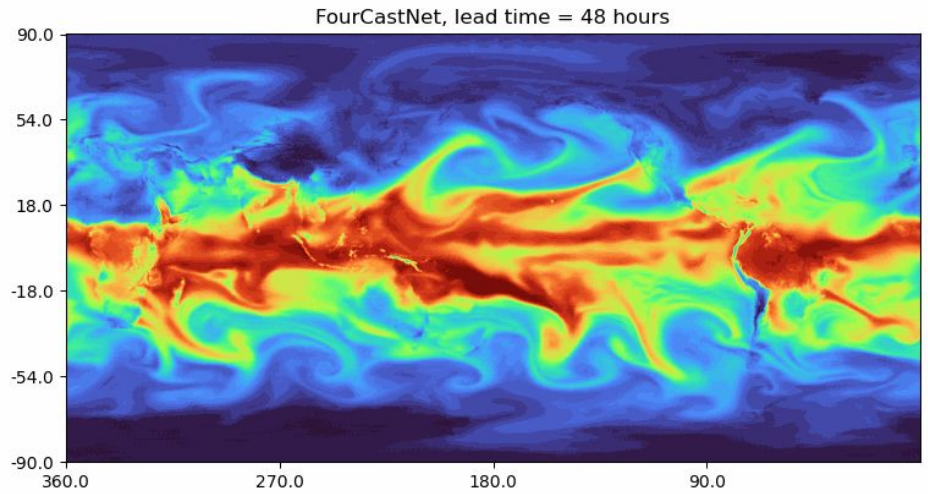
Use-cases

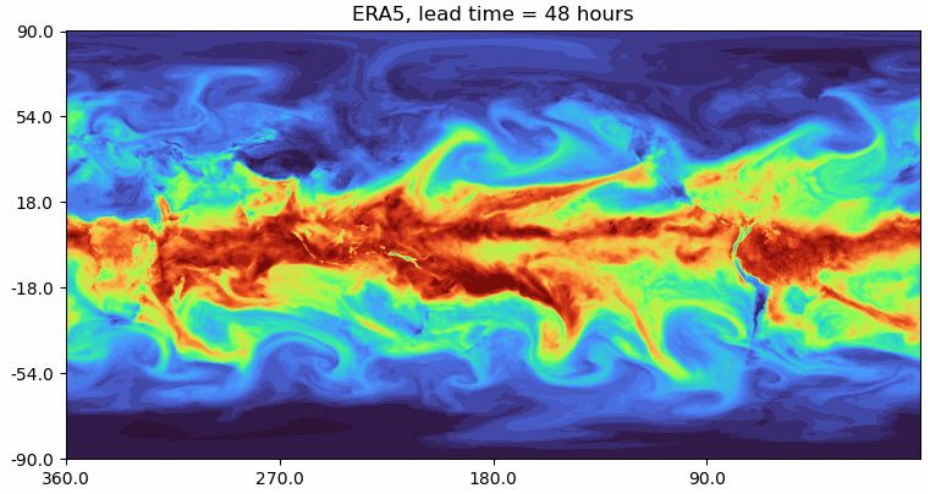
Use-cases

Weather and Climate

```
image = "${SCRATCH}/fcn-collaboration/ml+
    fourcastnet+22.12.sqsh"

mounts = [
    "${SCRATCH}/fcn-collaboration:/workdir",
    "${SCRATCH}/fcn-results/:/output",
    "${SCRATCH}/init_dir_fcn:/init_dir",
    "${SCRATCH}/fcn-collaboration/
    profiler_output:/profiler_output",
    "${SCRATCH}/m01/p_levels_raw:/input"
]

workdir = "/workdir"

[annotations]
com.hooks.cxi.enable = true
com.hooks.ofi_nccl.version = "cuda11"
```

Listing 3: Example of an EDF used to train FourCastNet on a CSCS cluster.





Use-cases

Machine Learning

```
mounts = [
    "${HOME}/.bash_history:/.bash_history",
    "${SCRATCH}:${SCRATCH}",
    "/tmp:/tmp"
]

[env]
HISTFILE = "/.bash_history"

[annotations]
com.hooks.cxi.enable = true
com.hooks.ofi_nccl.version = "cuda12"
```

Listing 6: Example of an EDF providing system settings to be reused as defaults.

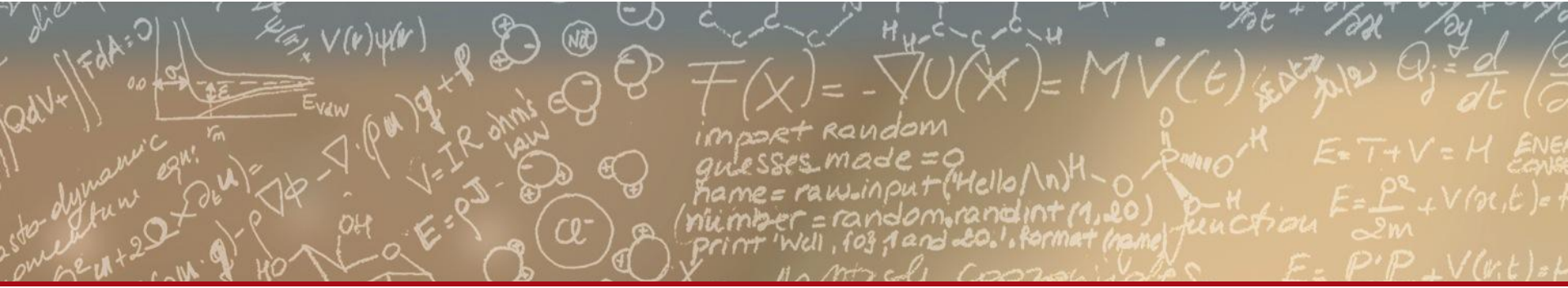
```
base_environment = "cscs-mlp"
image = "nvcr.io/nvidia/pytorch:24.01-py3"
```

Listing 7: Example of an use-case specific EDF using a base environment to inherit system defaults on a CSCS cluster for Machine Learning.

Conclusion

Conclusion

- Addresses the current needs of User Environments
- Isolates user environments from the system software stack
- Reproducibility and streamlining deployment processes
- Current pathfinder implementation
  - Accessible,
  - Modular,
  - Immediate,
  - and extensible
- For details, examples, and discussion see the paper!



Thank you!

Felipe A. Cruz (cruz@cscs.ch)