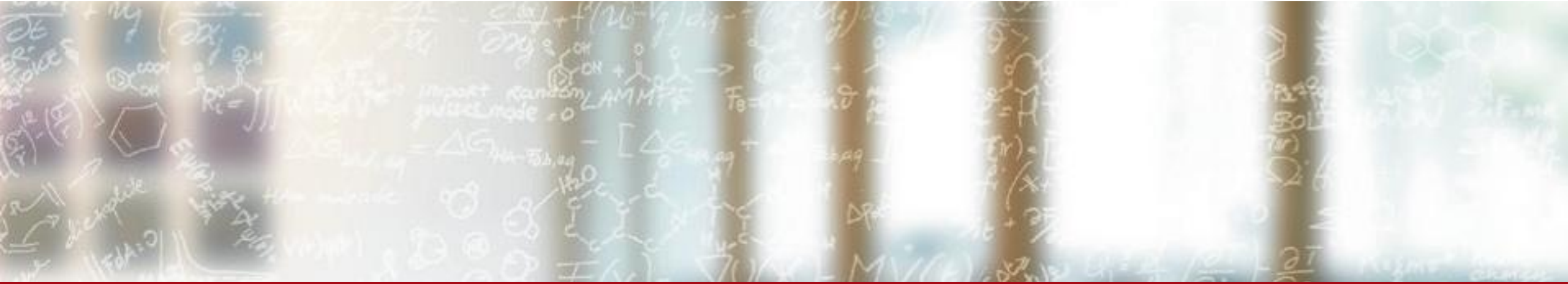




**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



# Mimic HPC Workloads in Kubernetes

Hussein Harake ([hussein@cscs.ch](mailto:hussein@cscs.ch))

Manuel Sopena Ballesteros ([manuel.sopena@cscs.ch](mailto:manuel.sopena@cscs.ch))

# Table of Contents

1. About CSCS
2. HPCAC
3. Motivation
4. Design/Challenges
5. Implementation
6. Feedback
7. Future work

# CSCS (Swiss National Supercomputing Centre)

- Founded in 1991
- Enables world-class research with a scientific user lab
- Available to domestic and international researchers through a transparent, peer-reviewed allocation process.
- Open to academia and are available as well to users from industry and the business sector.
- Part of ETH Zurich
- located in Lugano



# CSCS (Swiss National Supercomputing Centre)

- 2000 sq.m Machine Room
- 20 MW of power and Cooling capacity
- Lake Water cooling
- 700 Liters/s



# 13th Annual Swiss Conference & HPCXXL User Group



Developing cloud-native HPC clusters at CSCS - Developing cloud-native HPC clusters at CSCS

Container Technology for HPC - Alberto Garcia (Do IT Now)

HPC Container Conformance Project - Christian Kniep (QNIB Solutions)

## 13th Annual Swiss Conference & HPCXXL User Group



Mimic HPC Workloads in Kubernetes - Manuel Sopena (CSCS) and Hussein Harake (CSCS)

Kubernetes Clusters On-Demand: a New Era for HPC Centres - Elia Oggian (CSCS) and Riccardo Di Maria (CSCS)

Leveraging libfabric to compare containerized MPI applications performance over Slingshot 11 - Alberto Madonna (CSCS)

# Motivation

- R&D project not intended to have a fully operational production Slurm cluster
- Understand the challenges of deploying containerized infrastructure for HPC
- Spawn ephemeral Slurm clusters on demand?
- Reuse technologies familiar in the center
- Be able to run MPI
- Be able to SSH into the compute nodes (user access to login nodes)
- Security out of scope

# Design - Containers – Images

- Based on centos 7, Rocky 8, Redhat 8, CrayOS (Suse)
- Munge (Slurm authentication) - munged
- Slurmcn (Slurm compute and login pod) - slurmd & sshd
- Slurmctl (Slurm controller pod) – slurmctld & sshd
- Slurmdb (Slurm database pod for accounting) - slurmdbd
- Mariadb (database for slurmdb)

- **Challenges - System containers vs Application containers**
- Application/lightweight containers are meant to run only one application for the whole container lifespan
- System containers are meant to run one or more applications at any point of time
- Need a repository manager (yum, zypper, etc)
- Need a process manager inside a container

# Challenges - Process orchestration (Compute pods)

- Supervisor vs Systemd
- Can install applications from distro repos already integrated into systemd
- Stability
- Systemd ([https://systemd.io/CONTAINER\\_INTERFACE/](https://systemd.io/CONTAINER_INTERFACE/))
- Share localhost paths /tmp, /run, /sys/fs/cgroup (ro)
- To start systemd in a container /usr/lib/systemd/systemd --system

- **Challenges – munge configuration**

- Init container
- Fix folder permissions and ownership on `/etc/munge` and `/run/munge`

# Design - Pods/Deployments

- Pod == Node
- Compute pod (slurmcn and munge images)
- Controller pod (slurmctl image)
- Accounting pod (slurmdb & mariadb images)

# Design - Services

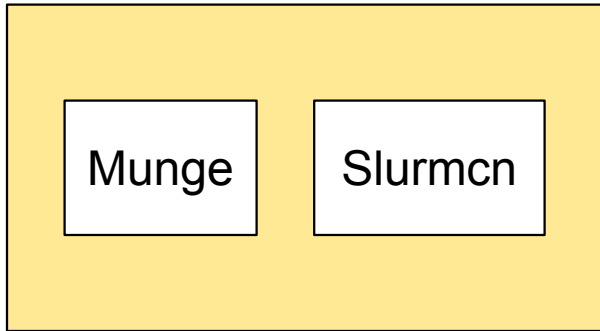
- Slurmd service (Cluster IP) – to expose slurmd port to other pods
- Surmctld service (Cluster IP) – to expose slurmctld port to other pods
- Public service (Node Port) – to expose login ssh port to outside k8s cluster

# Design – Volume

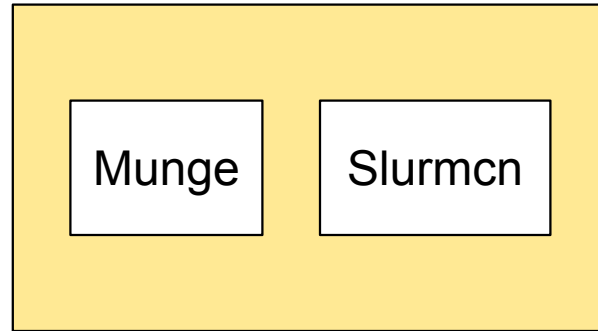
- `/etc/slurm/slurm.conf` (configmap for slurm configuration)
- `/etc/munge` - Munge key (ceph pvc)
- `/var/run/munge` - unix socket (empty dir)
- `/var/pool/slurmd` – slurm job data (ceph pvc)
- `/scratch` - user data shared across all clusters (ceph pvc)

# Design overview – Slurm cluster components in K8s - mycluster\_1

nid001001-mycluster\_1-xxxx

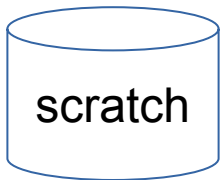
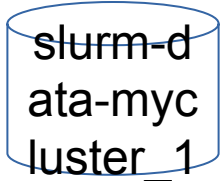
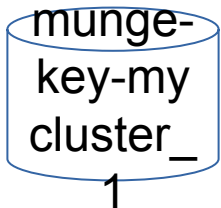
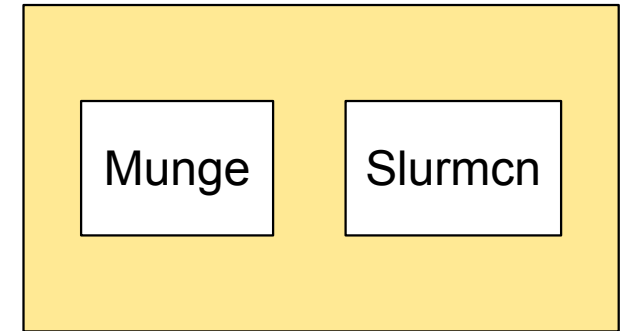


nid001002-mycluster\_1-xxxx

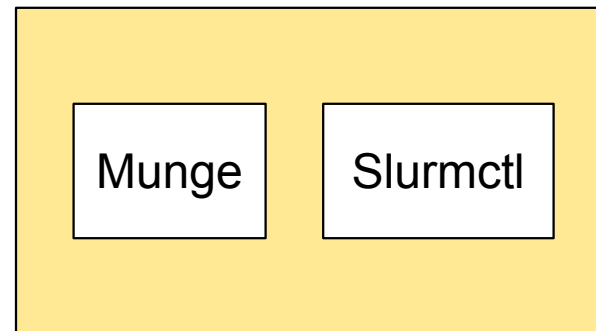


...

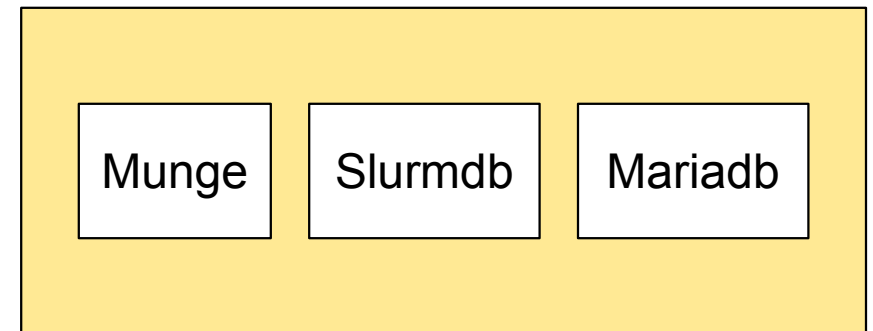
nid00100M-mycluster\_1-xxxx



slurmctl-mycluster\_1-xxxx

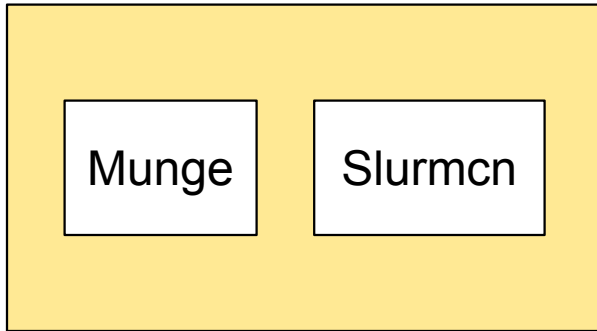


slurmdb-mycluster\_1-xxxx

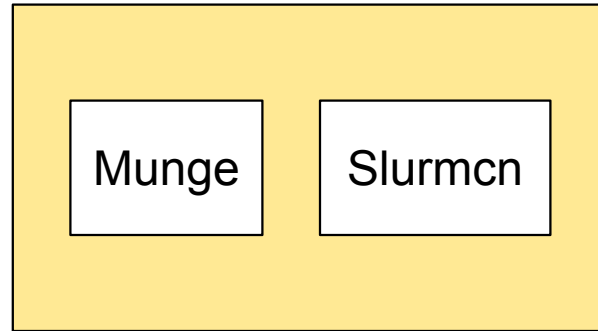


# Design overview - Slurm cluster components in K8s - mycluster\_N

nid001001-mycluster\_N-xxxx

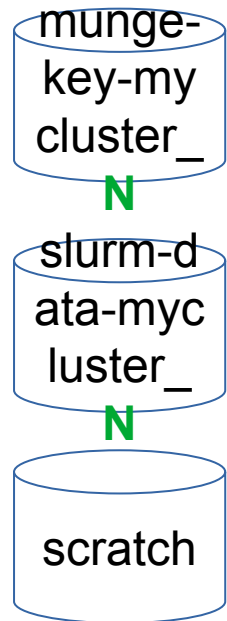
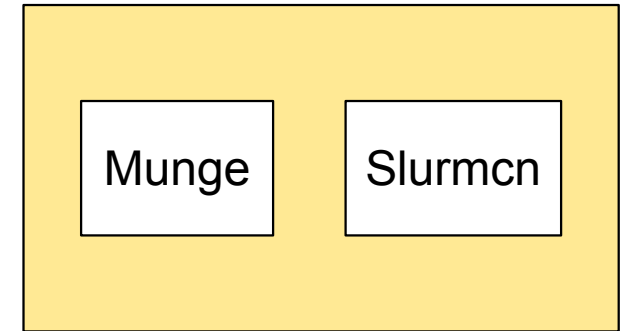


nid001002-mycluster\_N-xxxx

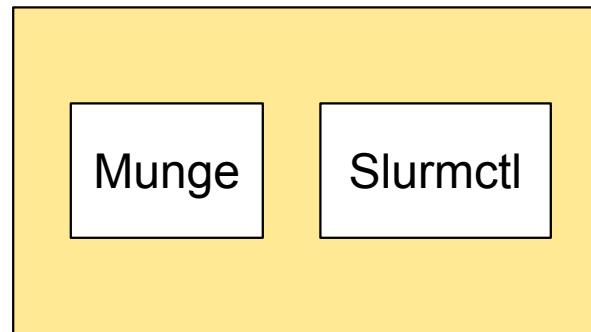


...

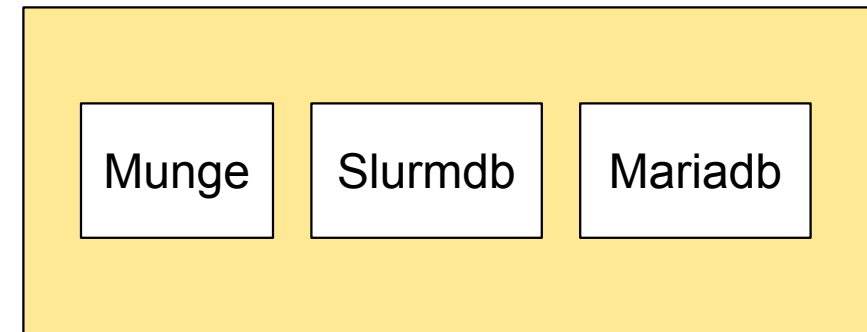
nid00100M-mycluster\_N-xxxx



slurmctl-mycluster\_N-xxxx



slurmdb-mycluster\_N-xxxx



# Implementation overview

- Components
  - Vanilla K8s
  - Vanilla Slurm
- Containerised slurm components
- Pods for login, cn, ctl and accounting
- Deployment orchestrated by ansible (ansible talks directly to kube api through kubectl)

```
msopena@support-ThinkPad-T490s:~/ownCloud/Documents/ALPSTRANS/slurm-k8s-deployment/slurm-k8s-systemd/utis$ kubectl -n psi-dev get pods
```

NAME	READY	STATUS	RESTARTS	AGE
accounting-cluster-1-dc6b9649d-4pmlr	2/2	Running	0	5h18m
compute001-cluster-1-7fcdff6b6d-55w2m	1/1	Running	0	5h18m
compute002-cluster-1-77848fd5b-qz75m	1/1	Running	0	5h18m
compute003-cluster-1-64b4f59b88-wcfjq	1/1	Running	0	5h18m
login-cluster-1-655d6f6bb8-tcgtd	1/1	Running	0	5h18m
slurmctl-cluster-1-5765f7bddf-nf2q6	1/1	Running	0	5h18m

## • Provisioning (ansible playbook)

- 1 Setup slurm.conf template
- 2 Setup k8s template files:
  - Deployment
  - PVCs
  - Services
- 3 Deploy cluster
- 4 Configure ssh
  - External ssh connection (using user public ssh key)
  - Internal ssh connection (for simple mpi tests)

# • Implementation - Ansible input params

- `commit_id`: cluster id
- `k8s_api_context_name`: kubectl config file
- `k8s_storageclass_name`: k8s storage class name to create pvcs
- `k8s_namespace`: namespace to deploy the slurm cluster
- `num_nodes`: number of compute nodes to deploy
- `node_name`: name to identify compute nodes
- `login_node_name`: name to identify login nodes
- `slurm_cluster_name`: name to identify slurm cluster
- `slurmdb_node_name`: name to identify slurmdbd node
- `slurmdb_deploy`: boolean to (yes|no) to see if want to deploy a new accounting server or reuse the one with `slurmdb_node_name`
- `public_key`: user's public key to ssh into slurm compute pod exposed to outside k8s
- `slurmcn_image`: container image name for slurm compute pod
- `slurmctl_image`: container image name for slurm controller pod
- `slurmdb_image`: container image name for slurm db pod

```
msopena@support-ThinkPad-T490s:~/ownCloud/Documents/ALPSTRANS/slurm-k8s-deployment$ kubectl -n psi-dev exec -it compute001-cluster-1-7fcdff6b6d-55w2m -- bash
Defaulted container "slurmcn" out of: slurmcn, rectify-mount-permissions (init)
[root@compute001-cluster-1 tmp]# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  43252  3492 ?        Ss   10:30   0:07 /usr/lib/systemd/systemd --system
root        24  0.0  0.0  47252 13500 ?        Ss   10:30   0:00 /usr/lib/systemd/systemd-journald
root        30  0.0  0.0 112984  4312 ?        Ss   10:30   0:00 /usr/sbin/sshd -D
dbus        31  0.0  0.0  58088  2116 ?        Ss   10:30   0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
munge       35  0.0  0.0 221664  1488 ?        Sl   10:30   0:00 /usr/sbin/munged
root        40  0.0  0.0 149872  3680 ?        Ss   10:30   0:00 /usr/sbin/slurmd -D
root       11502  0.6  0.0  11828  1904 pts/0    Ss   13:34   0:00 bash
root       11527  0.0  0.0  51736  1732 pts/0    R+   13:34   0:00 ps aux
[root@compute001-cluster-1 tmp]#
```

```
msopena@support-ThinkPad-T490s:~/ownCloud/Documents/ALPSTRANS/slurm-k8s-deployment/slurm-k8s-systemd$ kubectl -n psi-dev exec -it compute001-cluster-1-757749c485-nl8xl -- srun -N3 /scratch/mpi-helloworld
Defaulted container "slurmcn" out of: slurmcn, rectify-mount-permissions (init)
Hello world from processor compute002-cluster-1, rank 1 out of 3 processors
Hello world from processor compute001-cluster-1, rank 0 out of 3 processors
Hello world from processor compute003-cluster-1, rank 2 out of 3 processors
```

```
[nid003207:~ # kubectl get pod -n slurm -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
accounting-cluster-1-79c7f7ff46-skrv4	2/2	Running	2 (33h ago)	34h	10.42.151.23	nid003207
compute001-cluster-1-74f468d8c9-5ndkx	1/1	Running	1 (33h ago)	34h	10.42.151.24	nid003207
compute002-cluster-1-78cdc9d69-n2cf5	1/1	Running	1 (33h ago)	34h	10.42.224.73	nid003206
compute003-cluster-1-76947898b7-g49kb	1/1	Running	1 (33h ago)	34h	10.42.151.34	nid003207
login-cluster-1-747b8586d7-11162	1/1	Running	1 (33h ago)	34h	10.42.62.136	nid003205
slurmctl-cluster-1-856f76d74f-fv4jm	1/1	Running	1 (33h ago)	34h	10.42.224.75	nid003206

```
nid003207:~ # █
```

```
docker run --detach --privileged --volume=/sys/fs/cgroup:/sys/fs/cgroup:rw --volume=/dev:/dev
--volume=/sys/class/cxi_user:/sys/class/cxi_user --volume=/sys/class/net:/sys/class/net
--volume=/sys/devices/virtual/net:/sys/devices/virtual/net
--volume=/sys/bus/pci/drivers/cxi_core:/sys/bus/pci/drivers/cxi_core husseinharake/rocky9.1:07032023
```

```
docker run -it --rm --privileged \
  --device=/dev/xpmem:rw \
  --device=/dev/cxi0:rw \
  --volume=/opt/cray/libfabric/1.15.2.0/lib64/libfabric.so.1:/usr/lib/libfabric.so.1 \
  --volume=/usr/lib64/libcxi.so.1:/usr/lib64/libcxi.so.1 \
  --volume=/usr/lib64/libcurl.so.4:/usr/lib64/libcurl.so.4 \
  --volume=/usr/lib64/libjson-c.so.3:/usr/lib64/libjson-c.so.3 \
  --volume=/usr/lib64/libpals.so.0:/usr/lib64/libpals.so.0 \
  --volume=/usr/lib64/libnghttp2.so.14:/usr/lib64/libnghttp2.so.14 \
  --volume=/usr/lib64/libidn2.so.0:/usr/lib64/libidn2.so.0 \
  --volume=/usr/lib64/libssh.so.4:/usr/lib64/libssh.so.4 \
  --volume=/usr/lib64/libpsl.so.5:/usr/lib64/libpsl.so.5 \
  --volume=/usr/lib64/libssl.so.1.1:/usr/lib64/libssl.so.1.1 \
  --volume=/usr/lib64/libcrypto.so.1.1:/usr/lib64/libcrypto.so.1.1 \
  --volume=/usr/lib64/libgssapi_krb5.so.2:/usr/lib64/libgssapi_krb5.so.2 \
  --volume=/usr/lib64/libldap_r-2.4.so.2:/usr/lib64/libldap_r-2.4.so.2 \
  --volume=/usr/lib64/liblber-2.4.so.2:/usr/lib64/liblber-2.4.so.2 \
  --volume=/lib64/libz.so.1:/lib64/libz.so.1 \
  --volume=/usr/lib64/libunistring.so.2:/usr/lib64/libunistring.so.2 \
  --volume=/usr/lib64/libkrb5.so.3:/usr/lib64/libkrb5.so.3 \
  --volume=/usr/lib64/libk5crypto.so.3:/usr/lib64/libk5crypto.so.3 \
  --volume=/lib64/libcom_err.so.2:/lib64/libcom_err.so.2 \
  --volume=/usr/lib64/libkrb5support.so.0:/usr/lib64/libkrb5support.so.0 \
  --volume=/usr/lib64/libsas12.so.3:/usr/lib64/libsas12.so.3 \
  --volume=/usr/lib64/libkeyutils.so.1:/usr/lib64/libkeyutils.so.1 \
  --volume=/usr/lib64/libpcre.so.1:/usr/lib64/libpcre.so.1 \
  --volume=/var/spool/slurmd:/var/spool/slurmd \
  --volume=/var/lib/lugetlbfs:/var/lib/lugetlbfs \
  --env LD_PRELOAD="$preload_libsdd" \
  --tmpfs /tmp \
  --tmpfs /run \
  --tmpfs /run/lock \
  --volume /sys/fs/cgroup:/sys/fs/cgroup:ro \
  --entrypoint /sbin/init \
  husseinharake/rocky9.1:07032023
```

```
docker run --detach --privileged \  
  --volume=/sys/fs/cgroup:/sys/fs/cgroup:rw  
  --volume=/dev:/dev  
  --volume=/sys/class/cxi_user:/sys/class/cxi_user  
  --volume=/sys/class/net:/sys/class/net  
  --volume=/sys/devices/virtual/net:/sys/devices/virtual/net  
  --volume=/sys/bus/pci/drivers/cxi_core:/sys/bus/pci/drivers/cxi_core  
  --husseinharake/rocky9.1:07032023
```

```
[root@login-cluster-1 /]# module avail
```

```
----- /usr/share/Modules/modulefiles -----  
dot module-git module-info modules null use.own
```

Key:

**modulepath**

```
[root@login-cluster-1 /]# █
```

- **Future work**
- Image generation pipeline (Same pipeline building physical machines should create container images)
- Low footprint images? Is your PE small?
- Mount points? How to mount Luster or GPFS?
- Parametrise slurm.conf
- Security
- Affinity and anti-affinity
- Better ways to expose services (MetalLB)
- Investigate helm charts
- Scalability and performance (from user job perspective)

# Questions

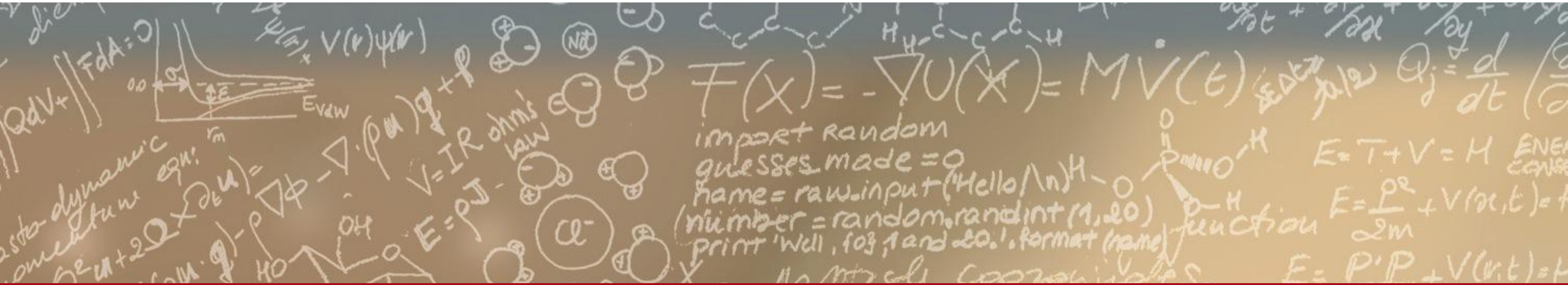
---



CSCS

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

ETH zürich



Thank you for your attention.