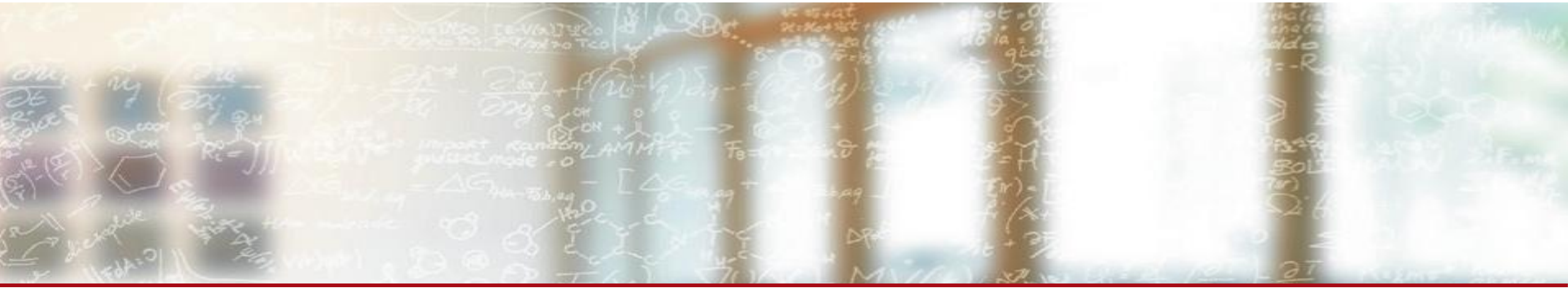
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Onboarding SDSC users at CSCS

Workshop

Prashanth Kanduri and Lukas Drescher
CSCS

19th October, 2023

Structure of the day

Morning

- Alps overview
- MFA access
- SSH configuration:
  - Daint login nodes and compute nodes
  - Login with VS code
- Running jobs using sbatch
- Conda environment
  - create custom jupyter-kernel
    - shared between Jupyter service, IDE and shell
    - finish by running python script through sbatch

Afternoon

- Running containers with Sarus on Piz Daint
  - …using NGC containers for single node and distributed deep learning
  - Large scale training on Piz Daint in MLPerf

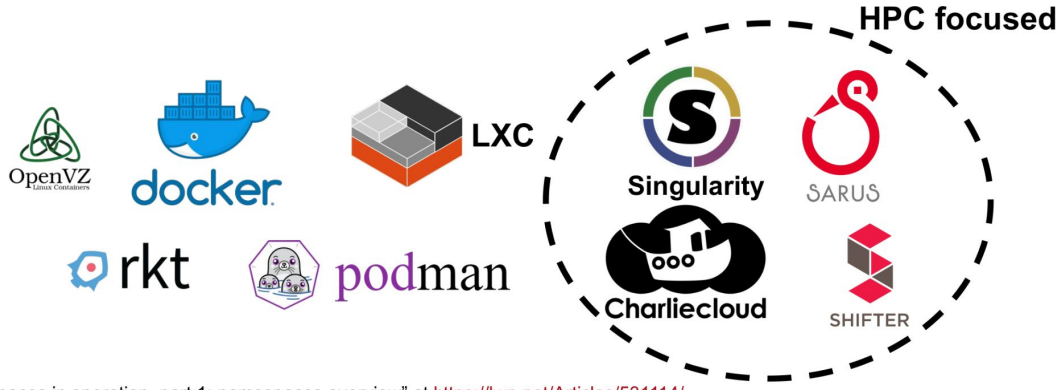
- Outlook on Alps
  - New container engine
  - FirecREST for automated workflows
  - High-performance data science with RAPIDS on Clariden

Containers - key concepts

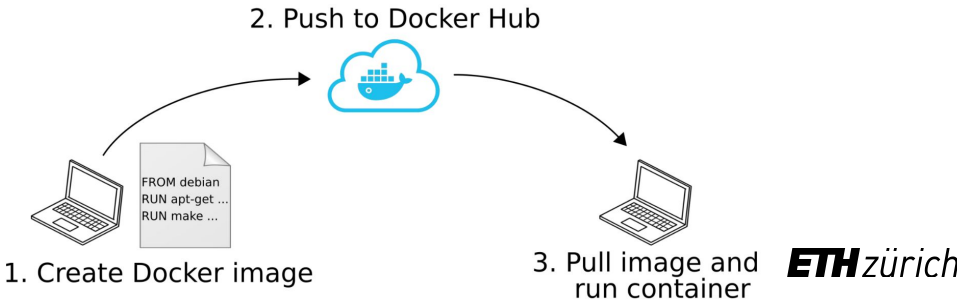
- **Isolated environments** to run applications/services
- **Images** include all software dependencies
  - standalone, executable package including code, runtime libraries, configuration files of an application
  - provides the filesystem and metadata (e.g. environment variables, initial working directory) for a container
- **Container** : a process isolated from the rest of the system through abstractions created by the OS.
  - The level of isolation can be controlled, allowing access to host resources.
  - Its filesystem content comes from an image.
  - “runtime instance” of an image: what the image becomes in memory when executed.
- **Reproducible** (explicit dependencies, consistent environment, immutable infrastructure, host-isol.), **portable, easy to build, quick to test & deploy**

Linux containers ecosystem

- Linux containers rely on abstraction features (namespaces[1]) provided by the kernel
- Different design decisions and use cases gave rise to several solutions:



[1] “Namespaces in operation, part 1: namespaces overview” at https://lwn.net/Articles/531114/



Using NVIDIA GPUs in Docker

GPU-accelerated application

- Included/built in the image, along with its runtime dependencies
- NVIDIA provides base images for CUDA, featuring compilers, runtime and accelerated libraries: https://hub.docker.com/r/nvidia/cuda
- Quickest way to get a Dockerfile going:
  `FROM nvcr.io/nvidia/cuda`

GPU driver

- It is tied to the hardware: cannot be part of a portable image!
- Has to be imported upon container creation
- NVIDIA Container Toolkit to the rescue!
  https://github.com/NVIDIA/nvidia-docker
- Docker >= 19.03 has native support:
  `docker run --gpus all nvcr.io/nvidia/cuda nvidia-smi`

Docker and HPC: not a good fit

- Security model assumes root privileges
- No integration with workload managers
- Missing support for diskless compute nodes
- Very limited support for kernel bypassing devices (e.g. accelerators and NICs)
- No adequate parallel storage driver

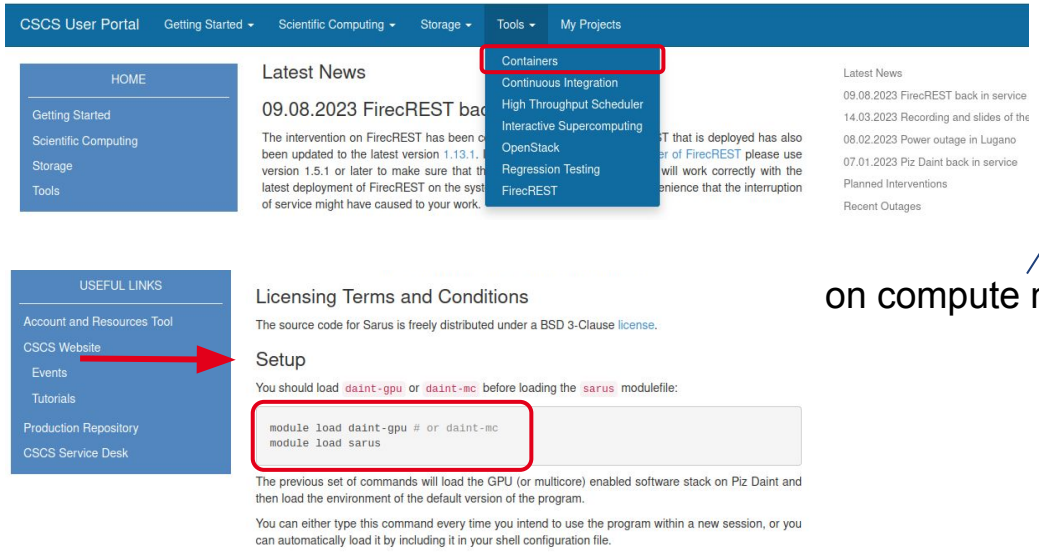
The Sarus container engine

- Security oriented to HPC systems
- SLURM-aware
- Integrates with HPC infrastructure and software
- OCI-compliant runtime that customizes containers through OCI hooks
  - HW can achieve native performance
- Pulls regular Docker images (e.g. from Docker Hub etc.) and provides Docker-like CLI

→ combines container portability with native HPC performance

Sarus resources

- user.cscs.ch→Containers→Sarus→Setup



- Comprehensive user guide
- Unofficial cheat sheet by Tomas Aliaga
- More materials at eth-cscs/containers-hands-on by Alberto Madonna

on compute node

Basic Commands	
help	List **commands** and *options*
version	Print sarus semantic version
--debug	Make commands output debug logs
--verbose	Make commands output info logs

Sarus is driven by its *sarus.json* configuration file, typically located at */opt/sarus/<version-number>/etc/sarus.json*.

Images		
images	`sarus images`	List local images
pull	`sarus pull python:alpine`	Pull image from DockerHub
load	`sarus load ./debian.tar my_debian`	Loads tarball as image
rmi	`sarus rmi centos:7`	Deletes image

Runtime	
run	`sarus run alpine cat /etc/os-release`
--tty	Allocate a pseudo-TTY in the container
--entrypoint echo hi	Overwrite the default ENTRYPOINT of the image
--mount=type=bind, src=HD,dst=CD	Mount custom host directories HD into the container at CD
--mpi	Enable MPI support
--ssh	Enable SSH in the container
--centralized-repository	Use centralized repository instead of the local one

You need to **pull** the image before you can **run** the container.

Useful Links	
documentation	https://sarus.readthedocs.io
</> code	https://github.com/eth-cscs/sarus/
download	https://github.com/eth-cscs/sarus/releases

on compute node (prev. salloc)

Examples on Localhost

```
GPU    sarus run ethcscs/cudasamples:9.2
nbody  /usr/local/cuda/samples/bin/x86_64/linux/release/nbody
       -benchmark -fp64 -numbodies=2000
```

on login node, through SLURM

Examples on HPC

```
GPU    srun -C gpu -N1 -t1 sarus run
nbody  ethcscs/cudasamples:9.2
       /usr/local/cuda/samples/bin/x86_64/linux/release/nbody
       -benchmark -fp64 -numbodies=200000
MC     srun -C gpu -N2 -t2 sarus run --mpi
all2all ethcscs/osu-mb:5.3.2-mpich3.1.4 ./osu_latency
```

Find more examples in the Sarus Cookbook.

Practical part - developing with Sarus on Piz Daint

Go to https://github.com/lukasgd/SDSC-user-onboarding

Distributed Training

Instructions for Horovod (alternative to DDP):

- user.cscs.ch→ Data Science → PyTorch

Large-scale distributed training comes in different flavors

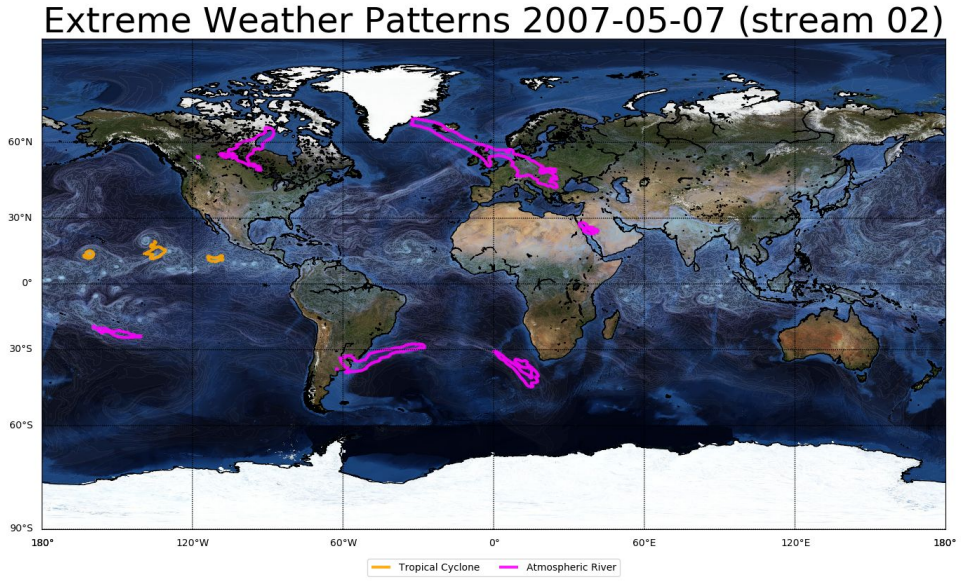
- data-parallel is most accessible
- pipeline-, model-parallel more advanced
- Experience from MLPerf HPC benchmark
  - https://mlcommons.org/en/training-hpc-07/, hpc-10/
  - significant speedups possible, e.g. DeepCAM with 9 TB requiring ~300 GPU hours converges in ~20 mins on 1000 GPUs!
  - more details presented at MLHPC SC'21

Considerations for scaling SGD with data-parallelism

- Optimize on a single GPU first
  - data loader - check GPU utilization, compare to loading synthetic data, be aware of kernel launch overhead
- Then scale horizontally
  - aim for ideal weak scaling, tune communication
- Be aware of convergence (#epochs) as a function of batch size

DeepCAM

- Gordon-Bell prize paper at SC18 (Kurth et al)
- convolutional encoder-decoder segmentation model
- identifies extreme weather phenomena: atmospheric rivers and tropical cyclones (3 per-pixel classes)
- 8.8 TB climate simulation data (HDF5), 0.25° spatial, 3h temporal resolution, shape (768, 1152, 16)
- Reference implementation in PyTorch



Outlook on Alps

Flash-based scratch FS

New container engine on Alps
- based on enroot/pyxis
- more user-friendly
  - TOML format
  - Infrastructure-as-Code
  - closer integration with SLURM

```
$ cat env.toml
image = "/iopsstor/scratch/.../.enroot/my_images/NGC-pytorch-23.09-py3.sqsh"
mounts = ["/iopsstor/scratch/cscs/lukasd:/scratch", "/users/lukasd"]
workdir = "/iopsstor/scratch/cscs/lukasd/path/to/repo"
writable = true

$ srun -ul --environment=$(pwd)/env.toml --gpus-per-task=1 bash -c '...'
```

Clariden
- Current testbed for the ML platform

Node configuration
- 128-core CPU
- > 500 GB RAM
- 4 A100 GPUs per node

Enables high-performance data science with RAPIDS:
https://github.com/lukasgd/SDSC-user-onboarding/tree/main/clariden

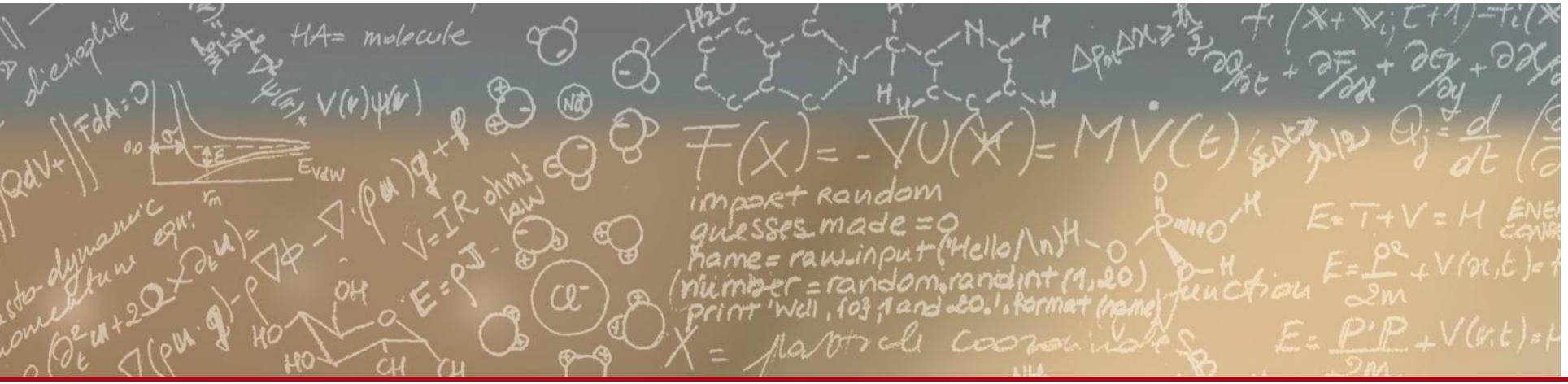
FIRECREST

REST API for HPC systems
https://firecrest-api.cscs.ch/

Take Home Messages

- It is very easy to set up persistent Conda environments on CSCS machines
  - They are reusable in the shell, IDE and the Jupyterlab service
  - Use miniconda to create custom development environments
  - It is also possible to attach IDEs (like VS Code) to compute nodes
  - Writing multi-gpu ML programs is easy, with DDP/Horovod/etc, then launch with `SBATCH` scripts
- **Sarus**, our high performance container engine allows using Docker images pushed to Dockerhub on compute nodes at CSCS
  - Start with **Nvidia NGC containers** for key packages like PyTorch/TensorFlow/etc as a basis
  - Containers are used for **CI on CSCS hardware** for development projects hosted on Git repos
- The **future ALPS platform** will be a **containers-first ecosystem** where replicating environments is designed to be seamless and user-friendly
  - **Full flash storage** for home and scratch for fast I/O
  - **Powerful CPU+GPU** hardware with **large memory** and **high bandwidths**
  - Dedicated cluster and service offering for the ML/AI user community
  - Customized workflows and services can be built with **FirecREST** (products.cscs.ch)

… also, we are just one floor away!



Thank you for your attention.