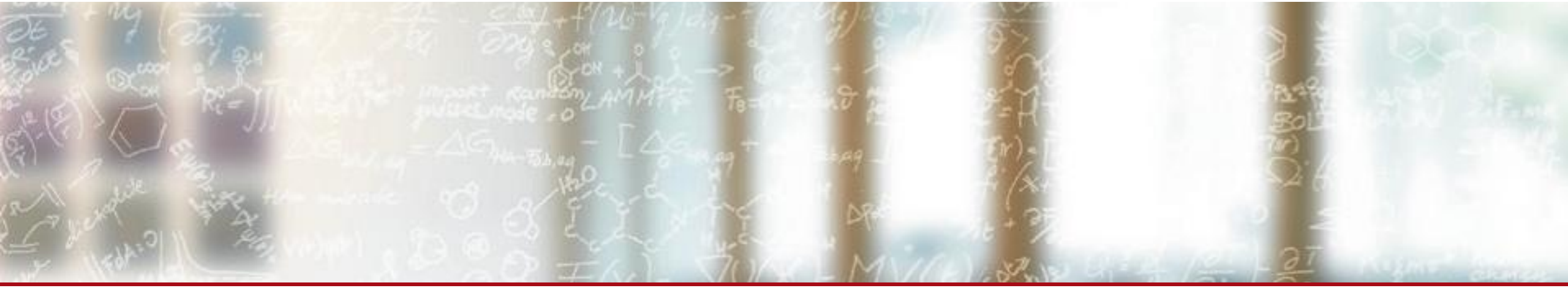
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



ML and PyTorch in Containers

September 2nd, 2024

Nick

CSCS Staff

CSCS Staff - Machine Learning Platform (MLP)

MLP Service Managers

Nick

Fawzi

Team ML

Lukas

Theofilos

Faveo

Nina

Auriane

Architecture Rep.

Stefano

Dep. Director

Joost

Henrique

Prashanth

Marcel

Alejandro

Fabian

Contents

Content

- Help & Support
- Preliminaries: SSH and IDE setup
- Containers
- Container Engine
- Building your first Container with Container Engine
- Using Containers for LLM Inference
- Using Containers for LLM Fine-Tuning

Sharing Issues and Discoveries

- Check the Knowledge Base first: https://confluence.cscs.ch/display/KB

- Slack:
  General discussions, CSCS staff **might** participate
  - cscs-user.slack.com
  - Discover which channels are relevant for you

- Tickets with support
  - support.cscs.ch

SSH Configuration

```
[nick@home ~]$ cat ~/.ssh/config

Host ela
    HostName ela.cscs.ch
    User cscs-username
    ForwardAgent yes
    IdentityFile ~/.ssh/cscs-key


Host alps-daint
    HostName daint-ln001.cscs.ch
    User cscs-username
    ProxyJump ela
    ForwardAgent yes
    IdentityFile ~/.ssh/cscs-key
```

`Host ela`
Entry for a specific host, can be used as an alias via `ssh ela`

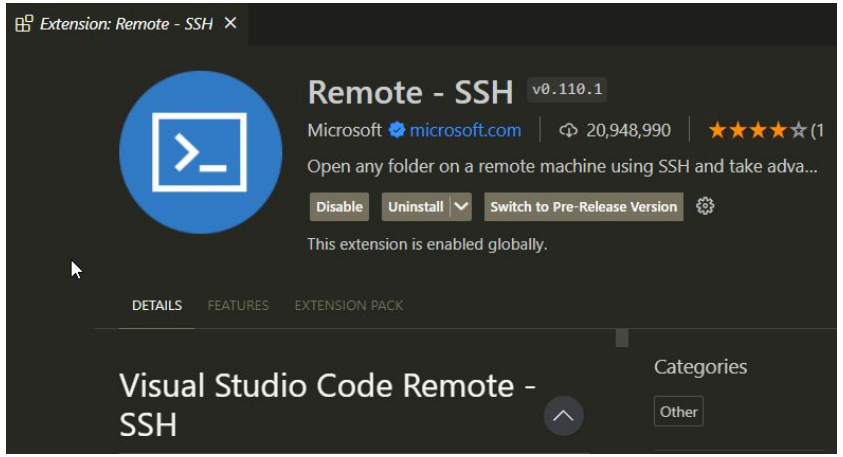
`HostName ela.cscs.ch`
Host address

`ForwardAgent yes`
Authentication agent on local machine used to authenticate connections made from remote machine

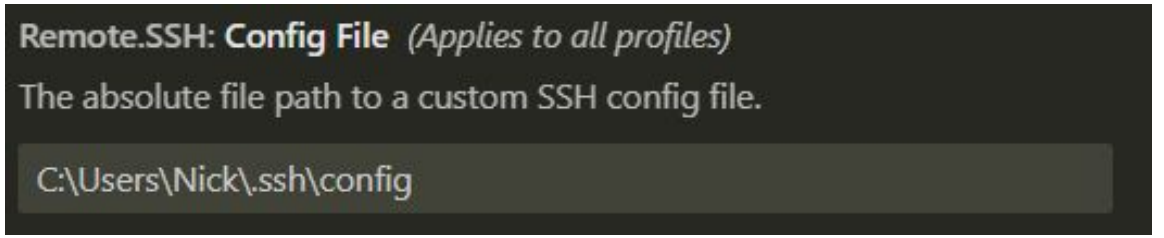
`ProxyJump ela`
Specify a jump host through which ssh agent should pass

`IdentityFile path`
Path to authorisation key

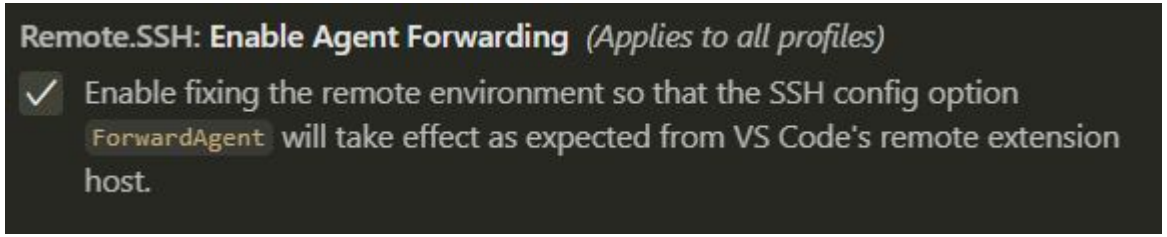
Remote IDE Setup - VS Code



Ensure path to `.ssh/config` is correct

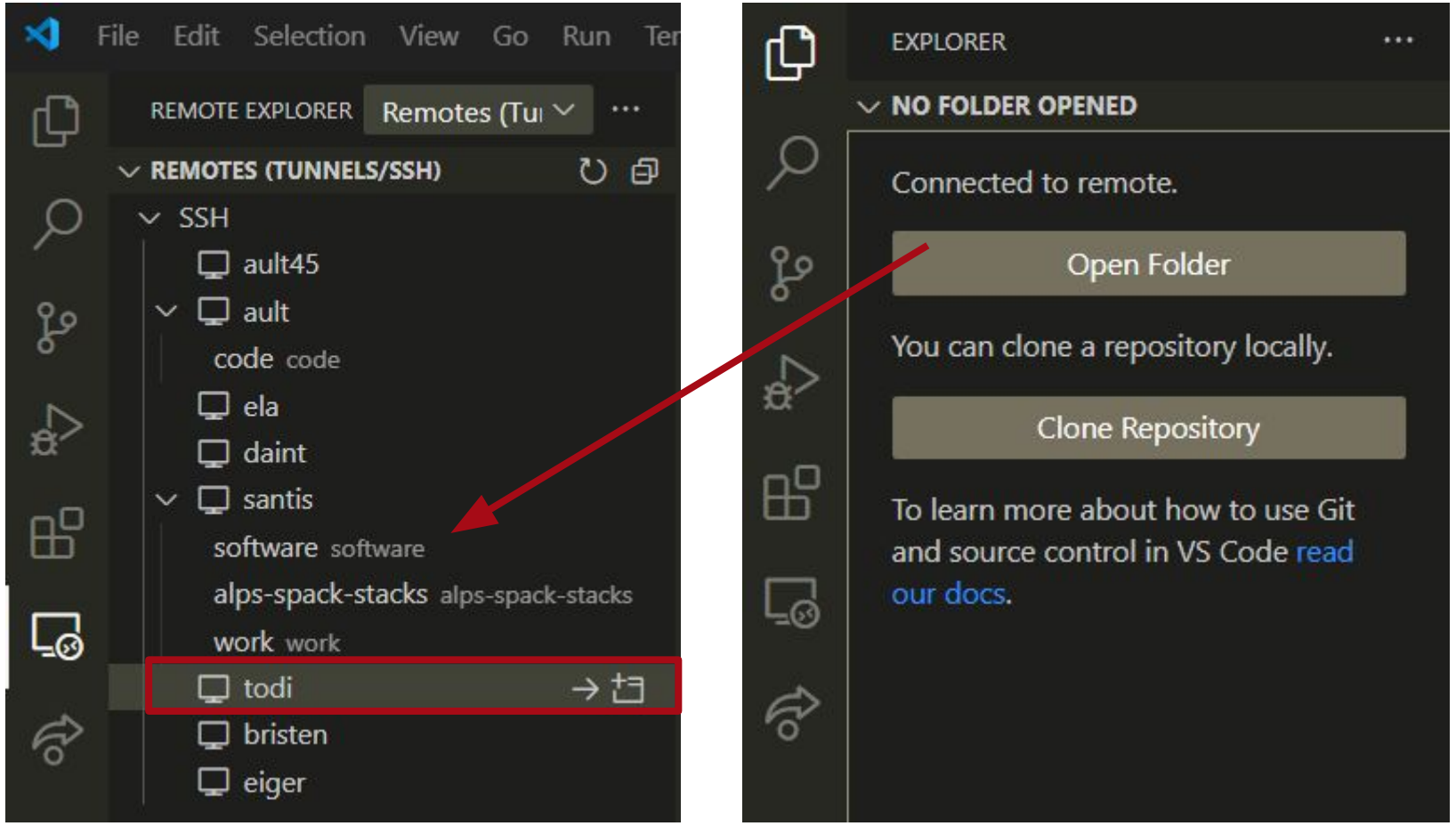


Enable Agent Forwarding



Remote IDE Setup - VS Code

1. Restart VS Code
2. Click the arrow icon next to the `todi` remote
3. After connection, go to the explorer panel (Ctrl + Shift + E)
4. Click Open Folder to open a directory on the remote host
5. Science!

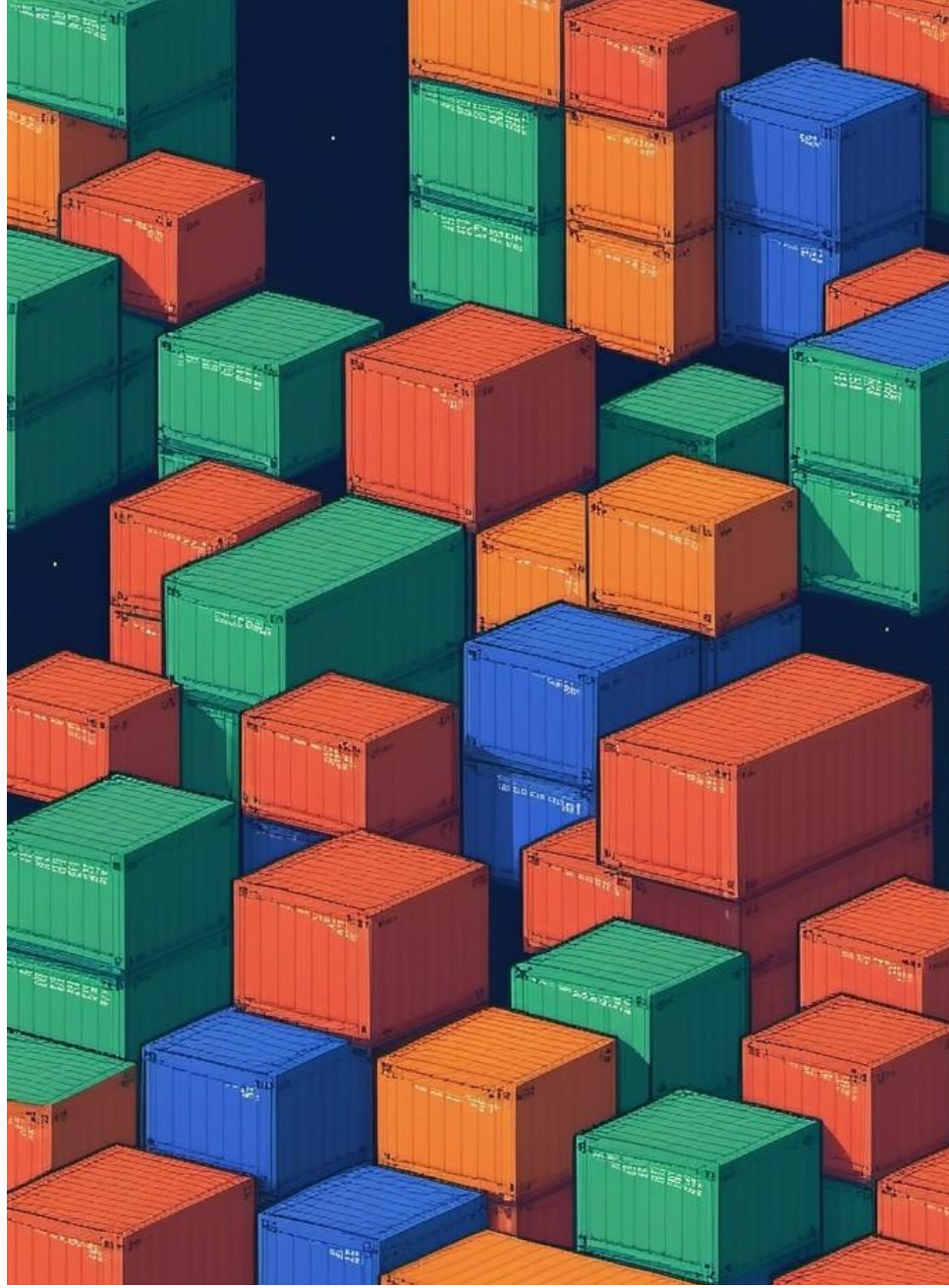


Containers

- Lightweight, standalone, (+executable) software stack that encapsulates an application in a single package, called an image
  - Code
  - Runtime + libraries
  - System tools
  - Settings

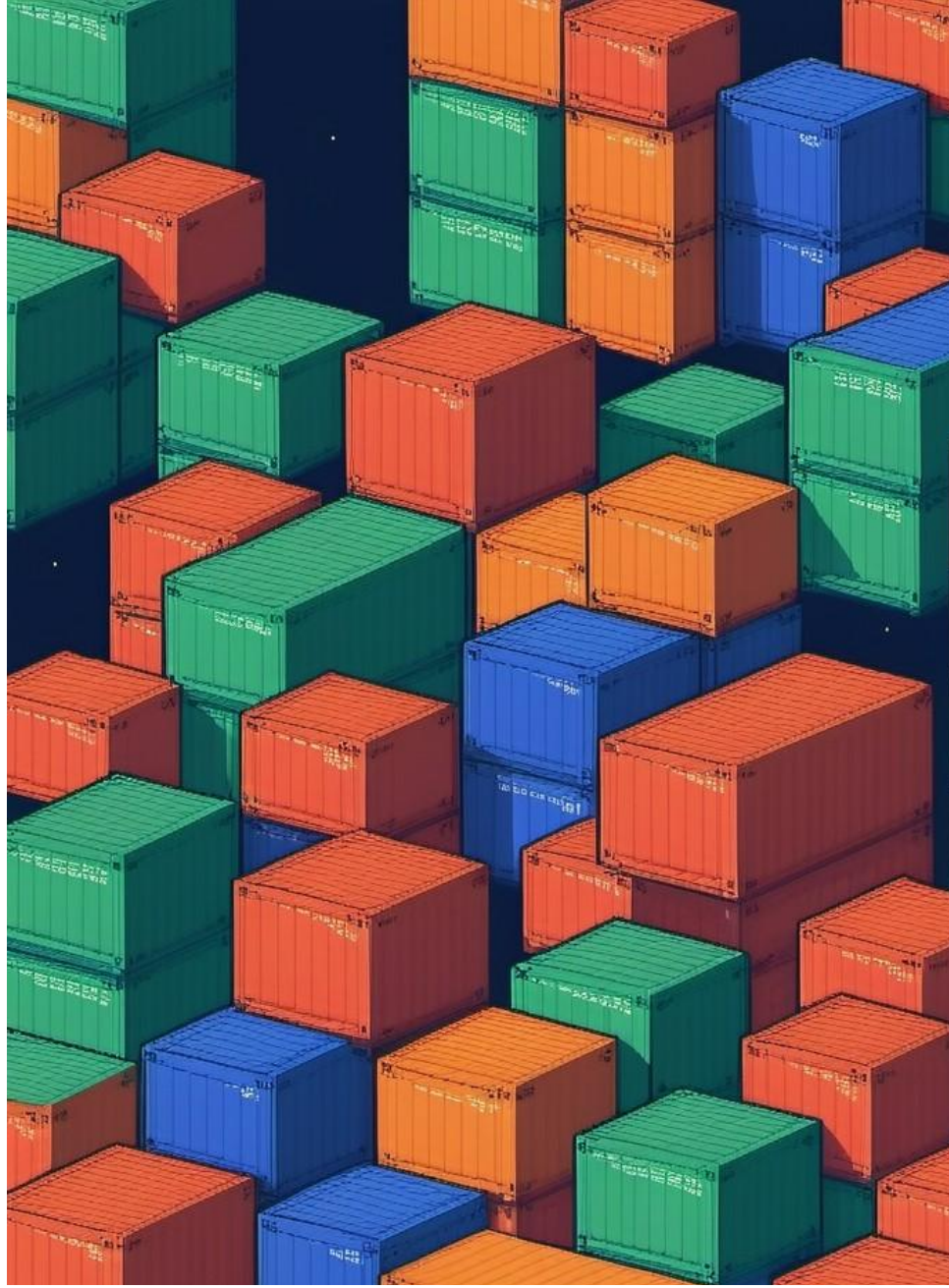
- Provide consistent environment for applications to run

- Easy to share across users/groups



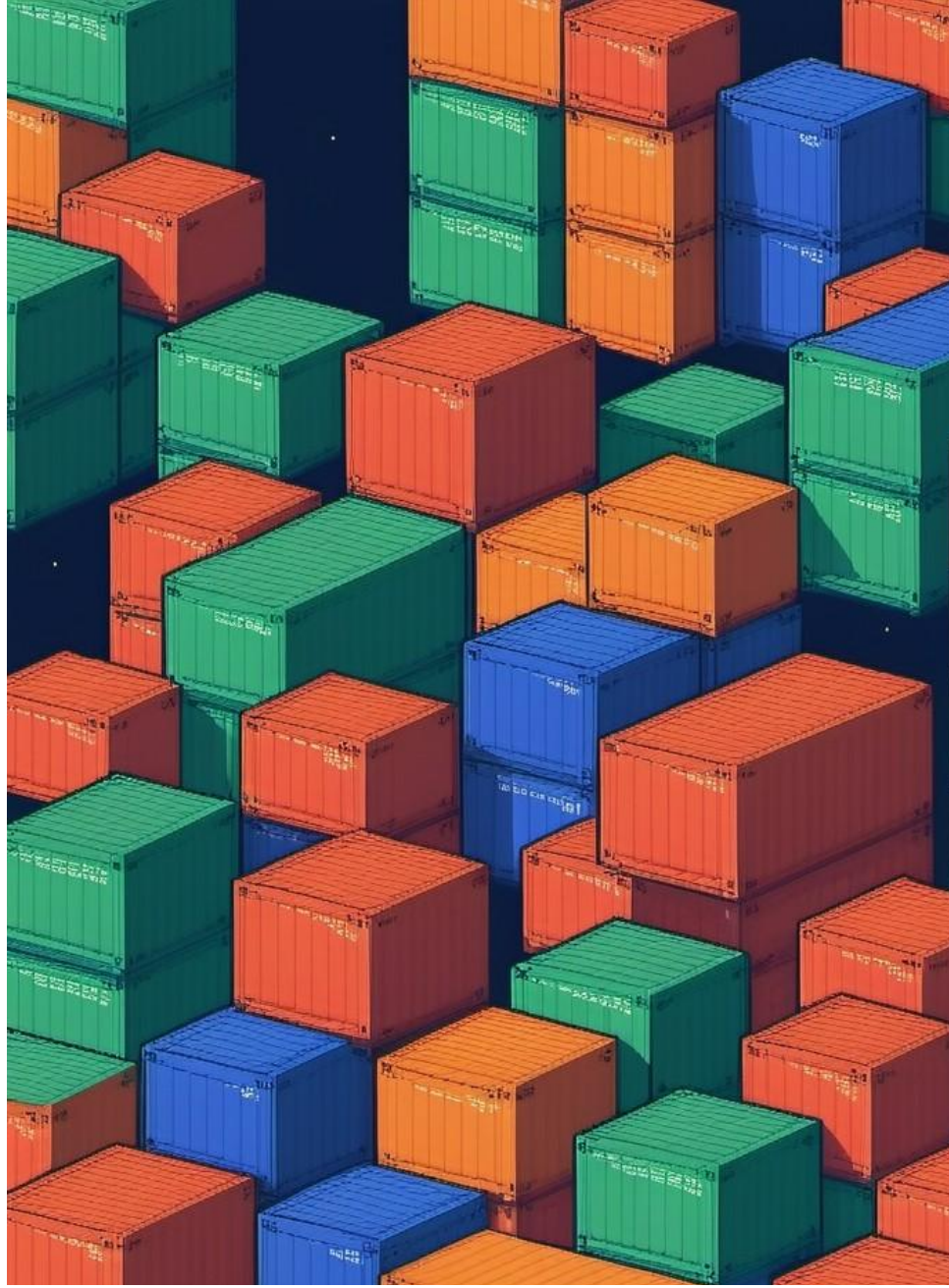
Containers

- Containers run in isolated user spaces, separate from the host system, though they share the same operating system kernel
- Large ecosystem of tools available
- Widely adopted
- In ML, vendors provide software stack optimised for their hardware as containers



Container Images

- Container image: all binaries and libraries, replace the current OS filesystem
  - Host filesystem inaccessible unless explicitly mounted
- You can get them from registries: hub.docker.com, quay.io, catalog.ngc.nvidia.com (nvcr.io), AMD's Infinity Hub, …
- or build your own for HPC, cloud, laptop…



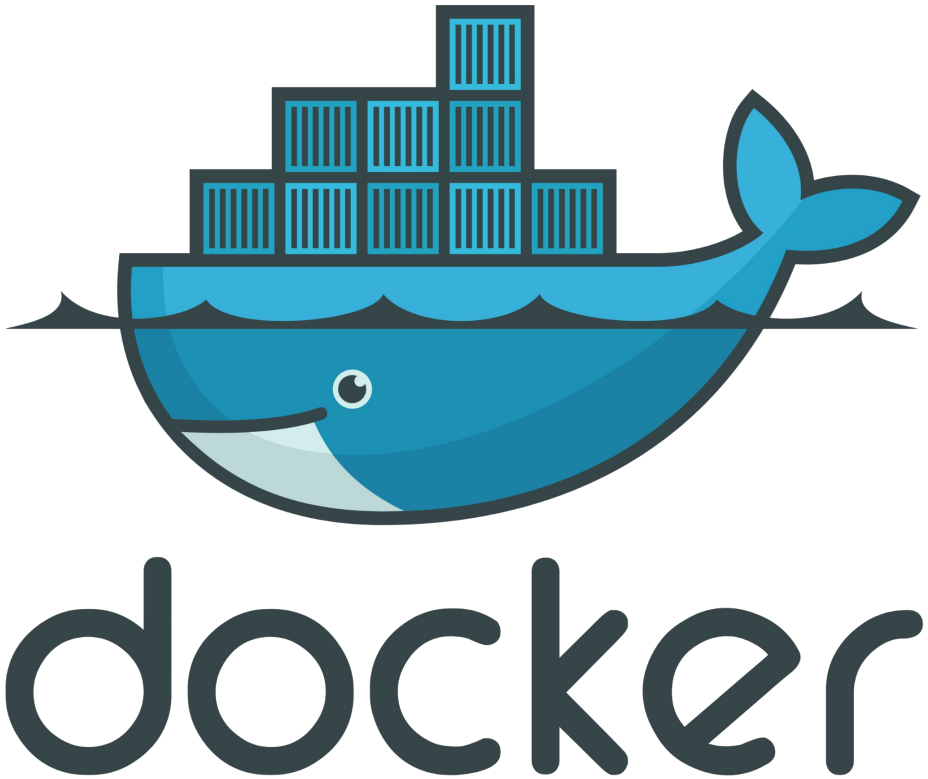
Docker

+ Widely adopted container engine

+ Thousands of images available @DockerHub

+ Robust ecosystem

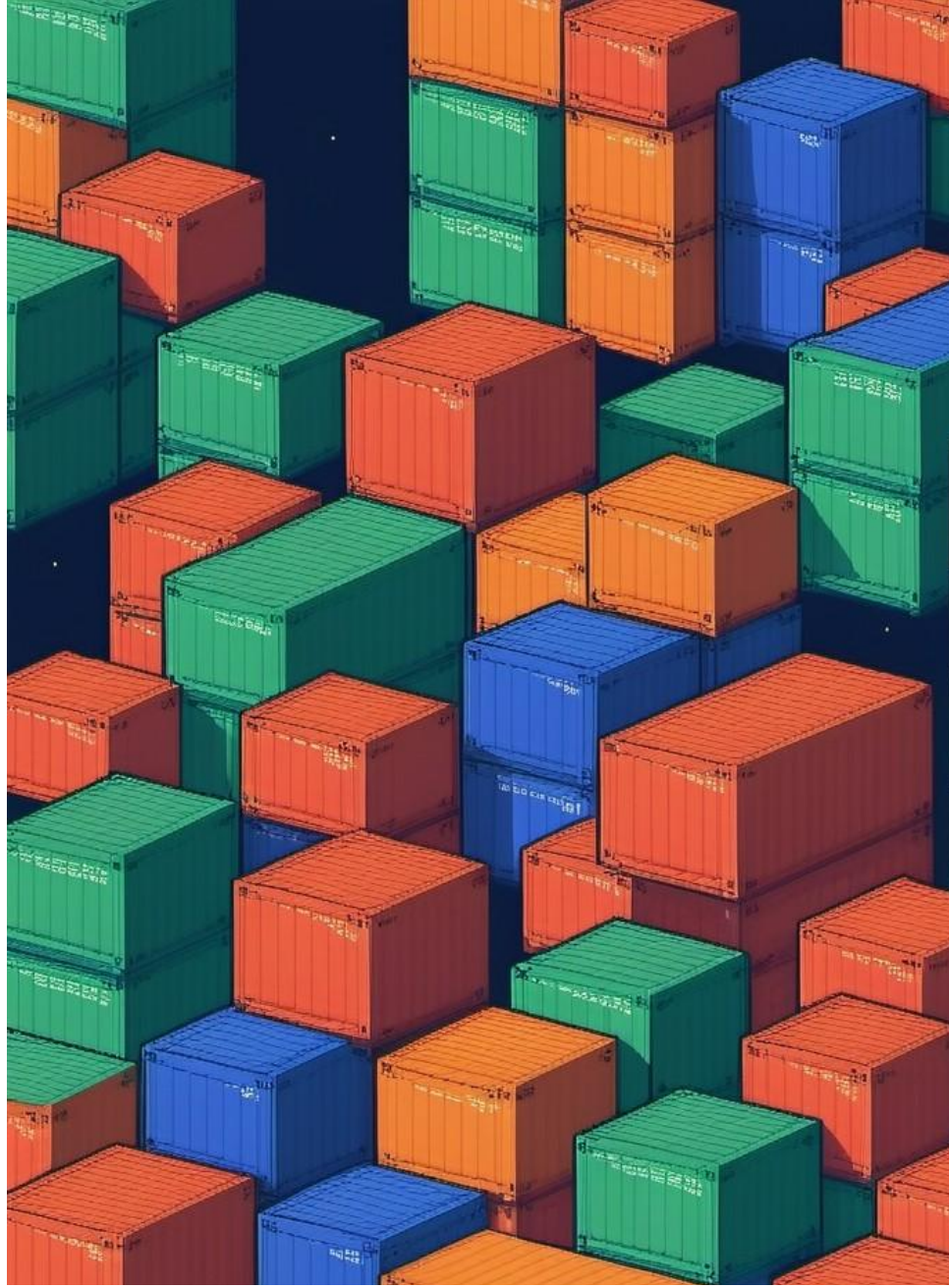
– Security concerns in HPC
  - Root daemon that runs your container with root privileges
– Difficulty in exposing specialized HPC hardware (e.g network interconnect)



Container Engine

Make usage of containers more transparent and seamlessly integrated into HPC systems

- Injection of HPC hardware accelerations (e.g Slingshot)
- Hooks
- Reproducible
- Integrated into existing container ecosystem
- Expose existing filesystem
- Secure
- Integrated into SLURM via plugins



Container Engine

- Pyxis

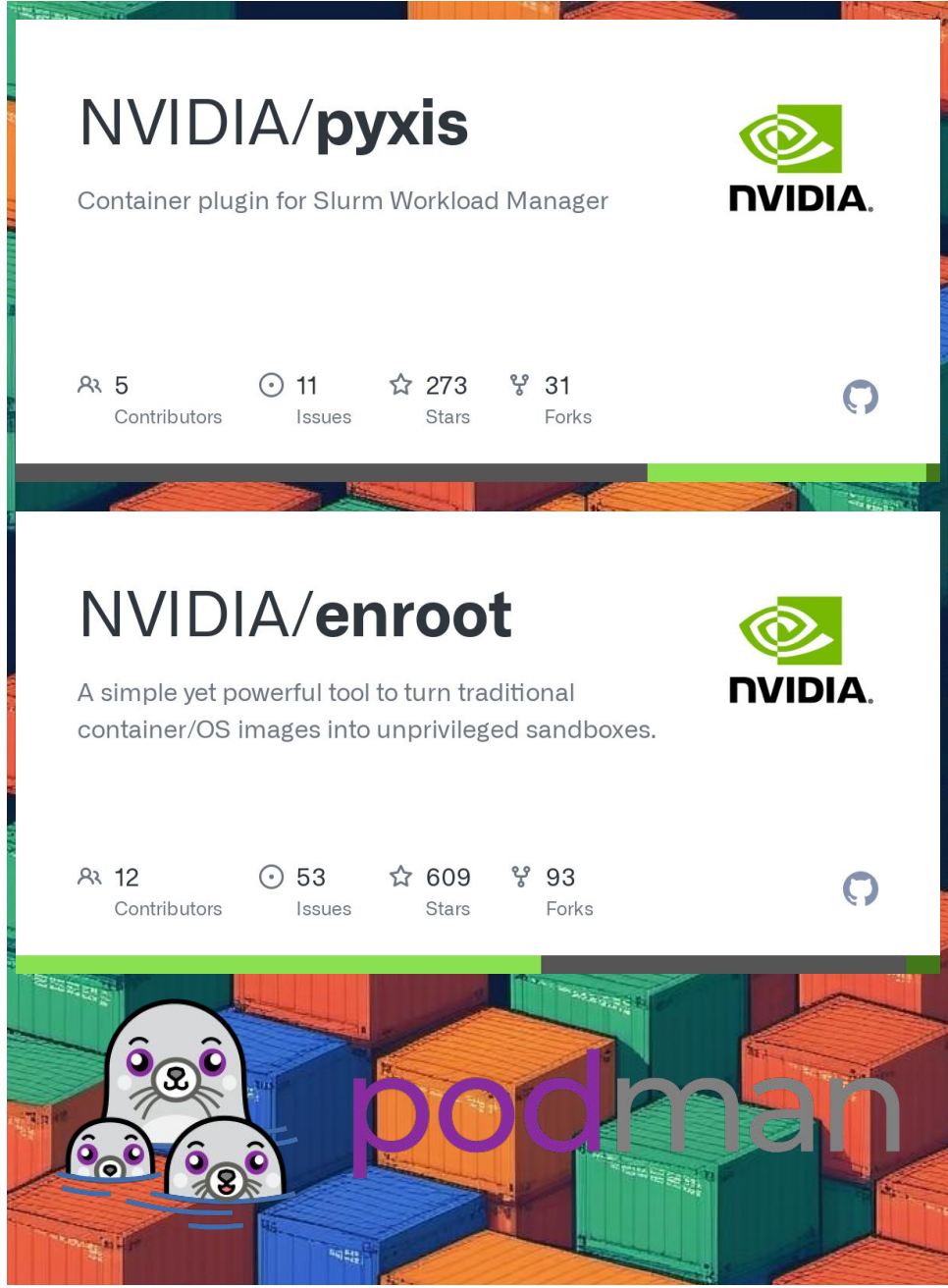
  Slurm plugin

- Enroot

  Container images in an unprivileged HPC container runtime

- Podman

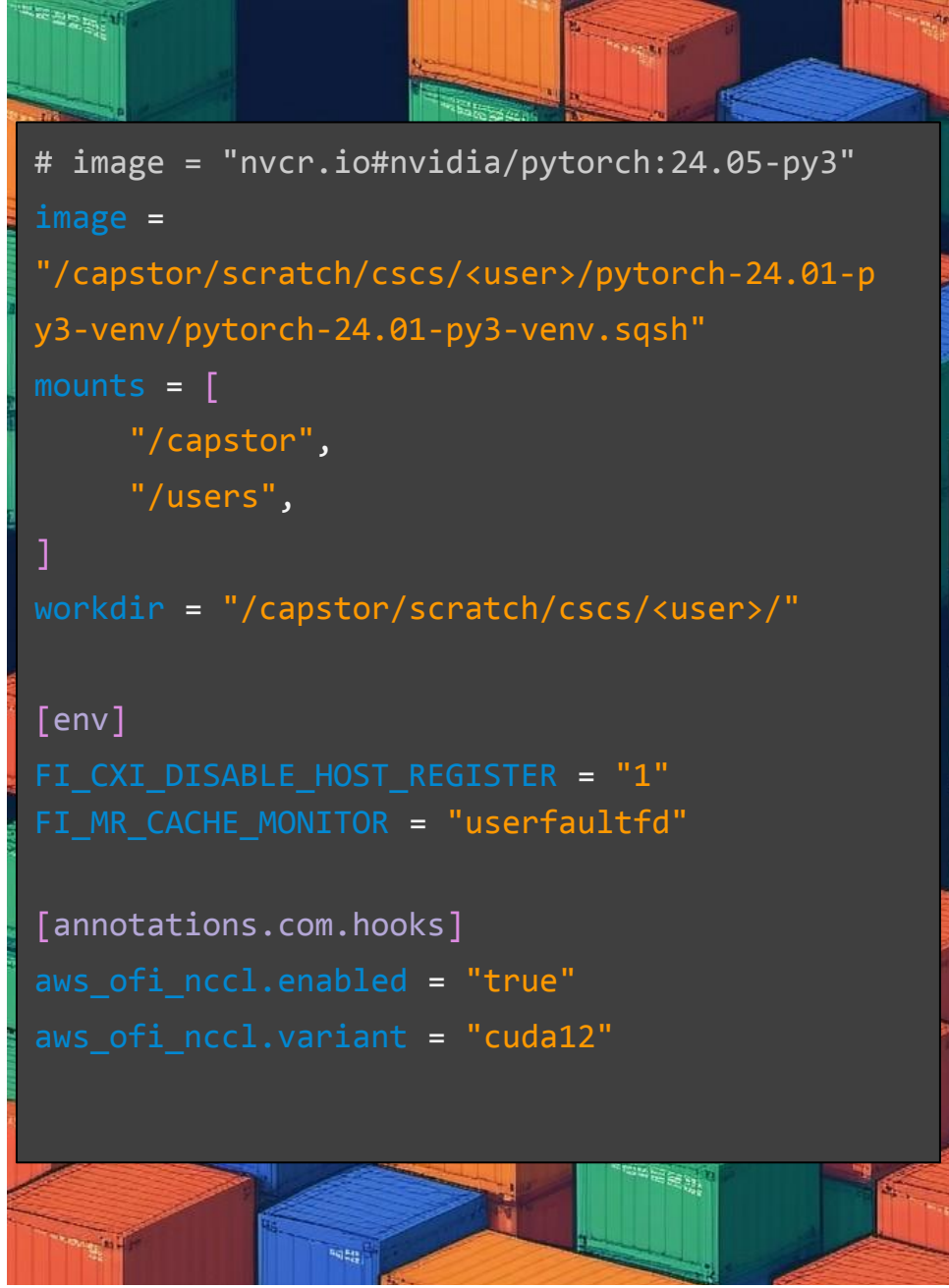
  Build container images



Container Engine - EDF: Environment Definition File

- a toml file that defines image, mount points, environment variables, annotations

- still in evolution
  - variable expansion
  - relative paths
  - Image building support



```
# image = "nvcr.io#nvidia/pytorch:24.05-py3"
image =
"/capstor/scratch/cscs/<user>/pytorch-24.01-p
y3-venv/pytorch-24.01-py3-venv.sqsh"
mounts = [
     "/capstor",
     "/users",
]
workdir = "/capstor/scratch/cscs/<user>/"

[env]
FI_CXI_DISABLE_HOST_REGISTER = "1"
FI_MR_CACHE_MONITOR = "userfaultfd"

[annotations.com.hooks]
aws_ofi_nccl.enabled = "true"
aws_ofi_nccl.variant = "cuda12"
```

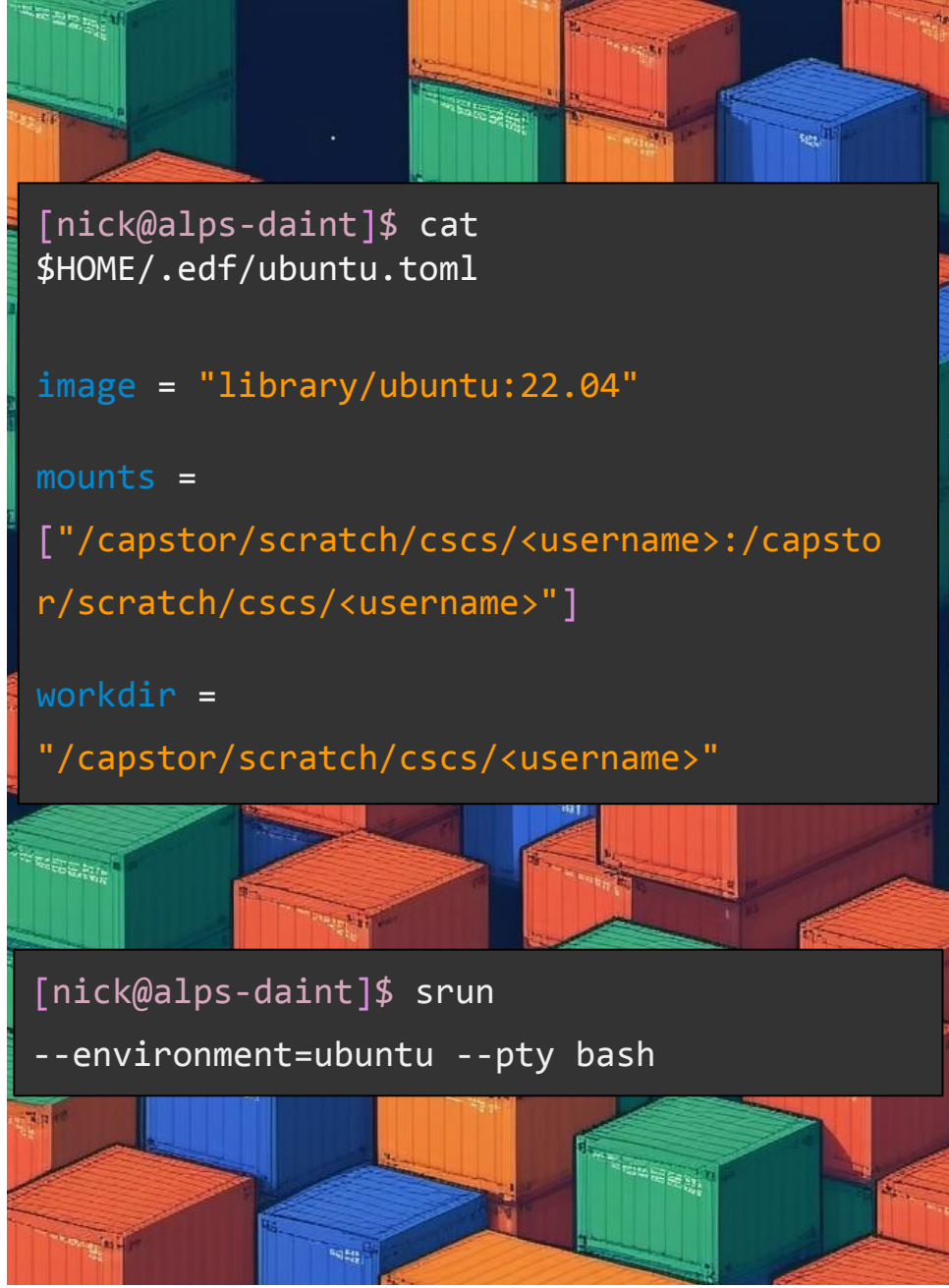
Container Engine - SLURM Integration

- **srun --help** [pyxis]
  - --environment=PATH
    the path to the Environment Definition File to use
  - --container-image=[USER@][REGISTRY#]IMAGE[:TAG]|PATH
  - --container-mounts=SRC:DST[:FLAGS][,SRC:DST...]
  - --container-env=NAME[,NAME...]

- Allows the same configuration in a toml format

Container Engine - Quickstart

- image is a string representing an image reference on a registry, or a path to a local image file
- mounts is TOML array of strings, representing bind mounts
  - format “source-on-host:destination-path-container”
- workdir is a basic string representing initial working directory inside container



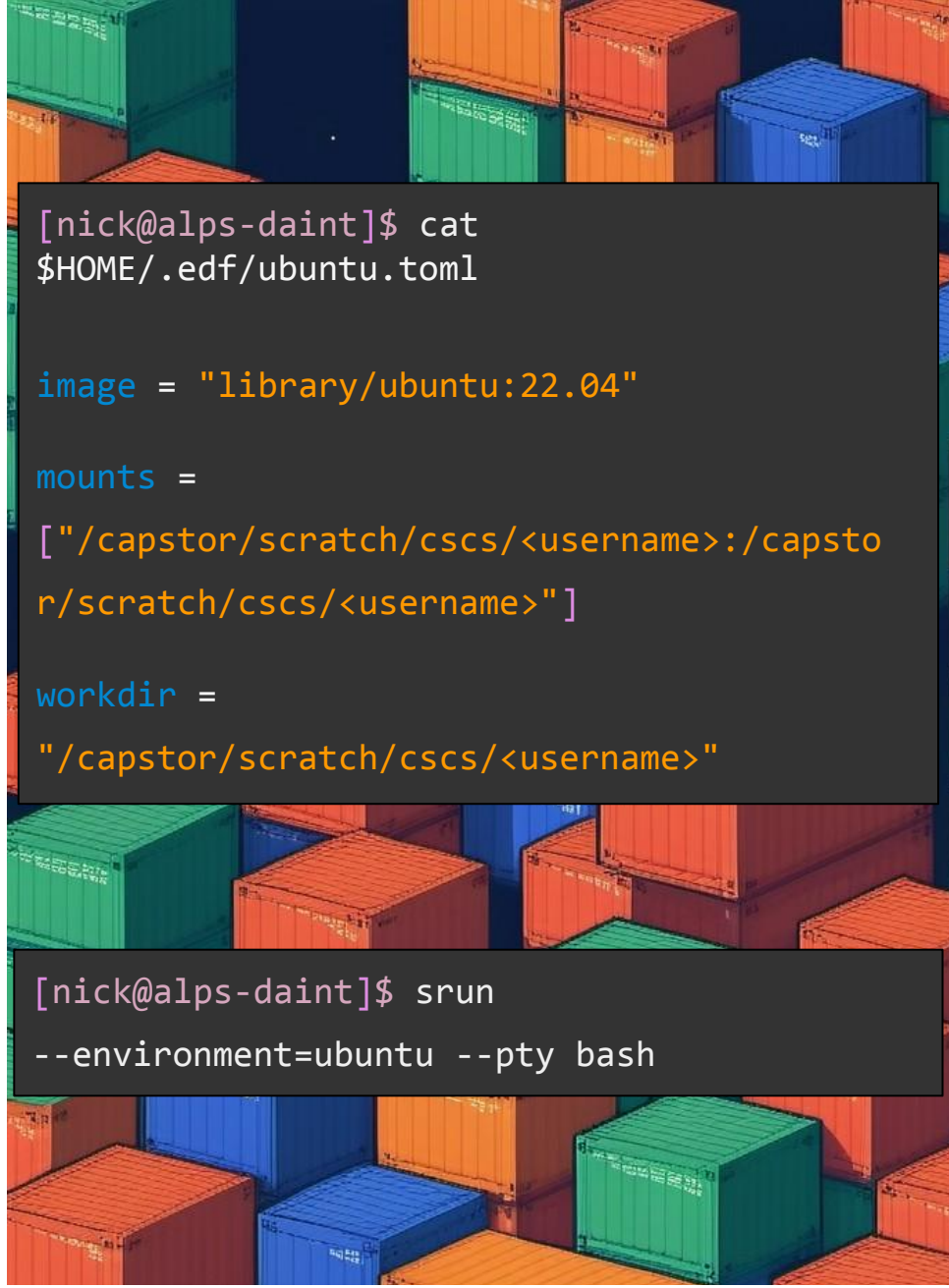
```
[nick@alps-daint]$ cat
$HOME/.edf/ubuntu.toml

image = "library/ubuntu:22.04"

mounts =
["/capstor/scratch/cscs/<username>:/capsto
r/scratch/cscs/<username>"]

workdir =
"/capstor/scratch/cscs/<username>"
```

```
[nick@alps-daint]$ srun
--environment=ubuntu --pty bash
```

Container Engine - Quickstart

- $HOME/.edf/ directory is default search path for .toml files
  - Search path controlled through $EDF_PATH environment variable
  - Colon separated, absolute paths to directories where CE looks for TOML files

- You don't need to specify full path or filename
  - "ubuntu" refers to $HOME/.edf/ubuntu.toml

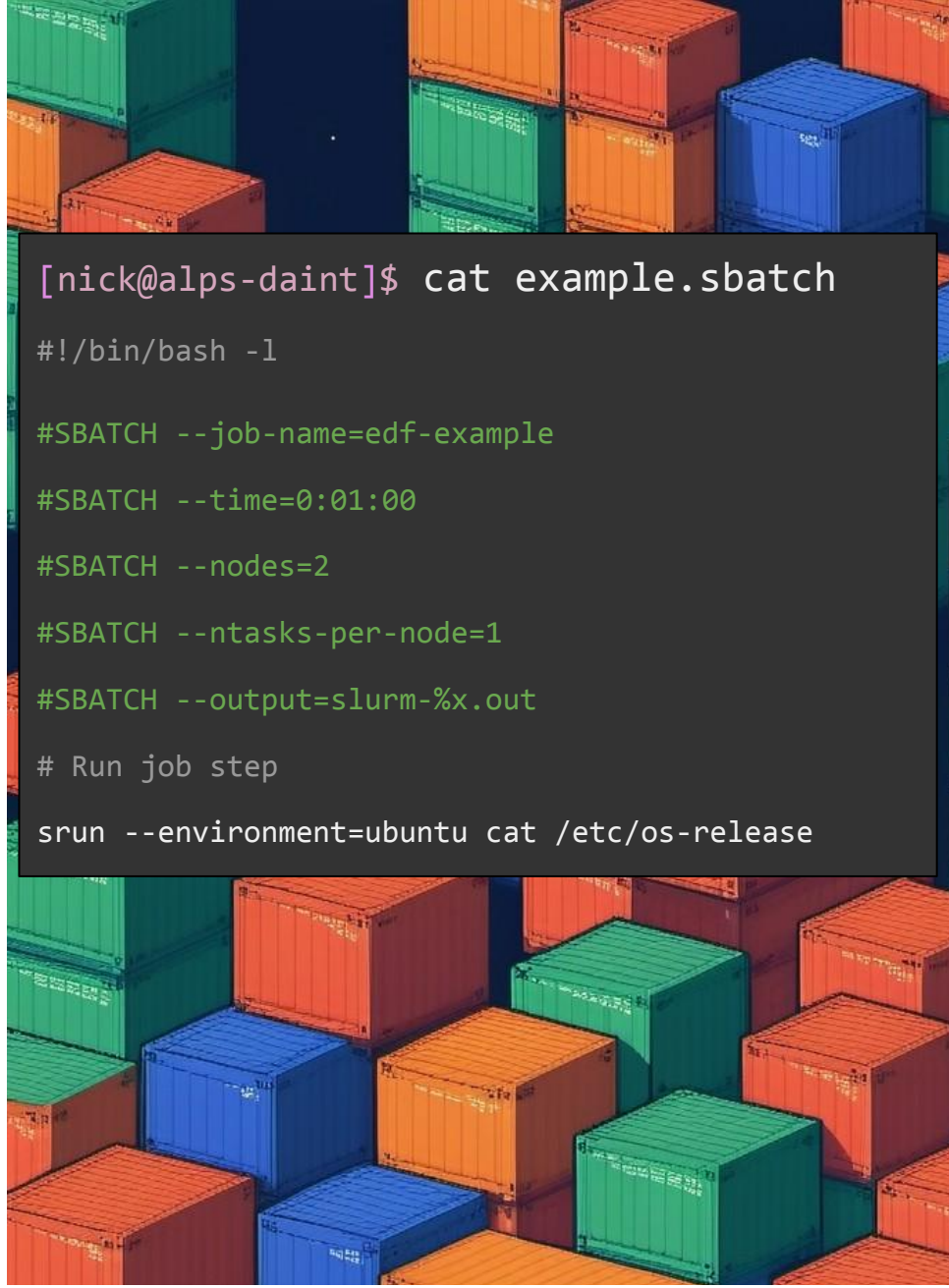


```
[nick@alps-daint]$ cat
$HOME/.edf/ubuntu.toml

image = "library/ubuntu:22.04"

mounts =
["/capstor/scratch/cscs/<username>:/capsto
r/scratch/cscs/<username>"]

workdir =
"/capstor/scratch/cscs/<username>"
```

```
[nick@alps-daint]$ srun
--environment=ubuntu --pty bash
```

Container Engine - Quickstart

- --environment option can be used as an #SBATCH option, but…
  - the CE toolset does not currently support accessing the Slurm workload manager
    - Cannot use srun or scontrol
- Best to use --environment as part of the Slurm commands



```
[nick@alps-daint]$ cat example.sbatch
#!/bin/bash -l

#SBATCH --job-name=edf-example
#SBATCH --time=0:01:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --output=slurm-%x.out

# Run job step
srun --environment=ubuntu cat /etc/os-release
```

Container Engine - Pulling Images

- Registry reference strings (library/ubuntu:22.04) results in each job discarding image upon termination
  - Large images can have noticeable import times
- Can pull images explicitly beforehand using enroot
- Formats
  - enroot import docker://[REGISTRY#]IMAGE[:TAG]
  - enroot import docker://IMAGE[:TAG]
- DockerHub is default registry

Container Engine - Third Party or Private Repositories

- URL has to be prepended to image reference using (#) as separator
- For private repositories, access credentials should be configured in $HOME/.config/enroot/.credentials
- For NGC containers (nvcr.io), you'll need to register and create an API key here: catalog.ngc.nvidia.com

```
[nick@alps-daint]$ enroot import
docker://nvcr.io#nvidia/nvhpc:23.7-runtime-cuda11.8-ubuntu22.04
```

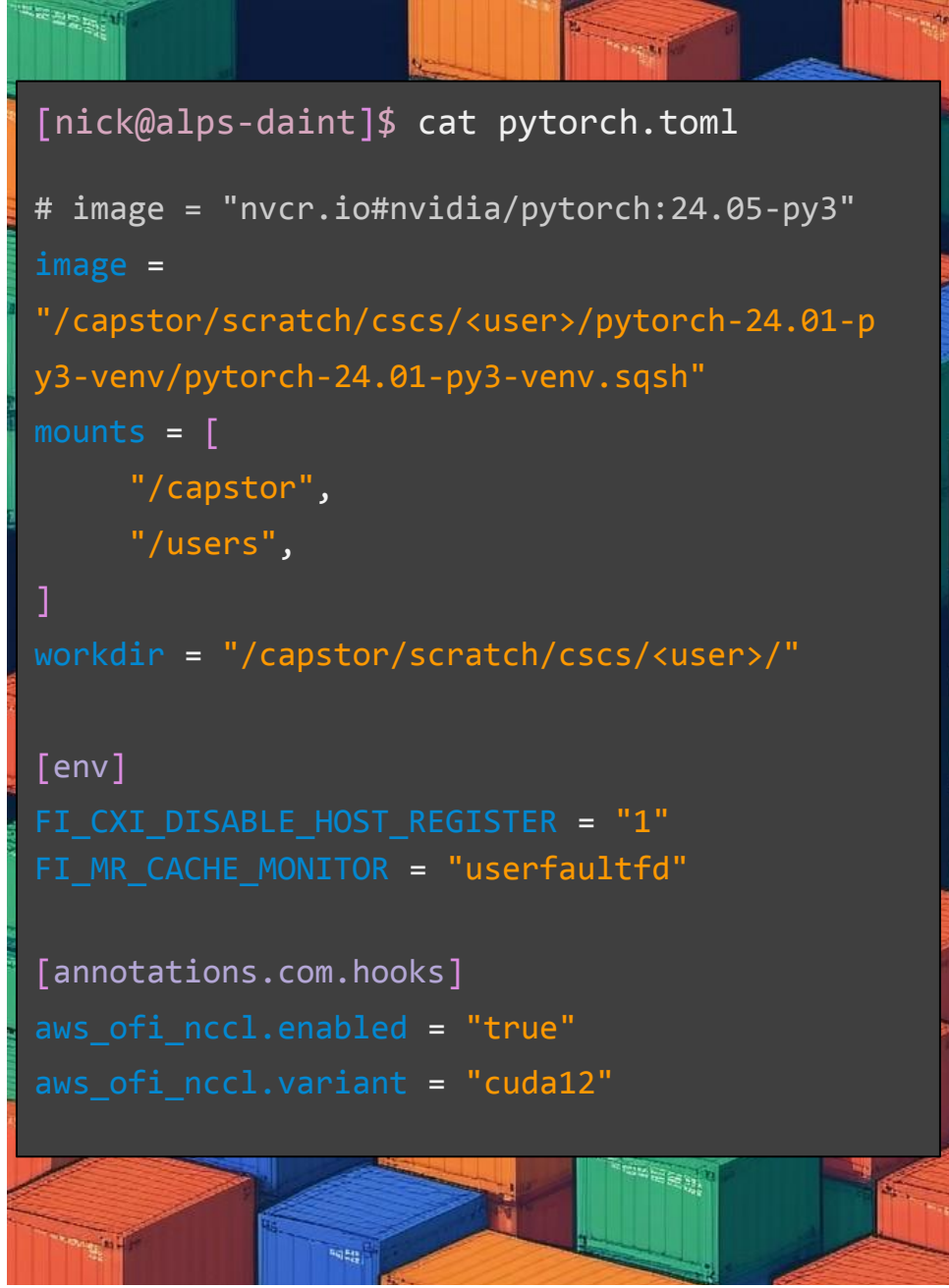
```
[nick@alps-daint]$ cat
$HOME/.config/enroot/.credentials

# NVIDIA NGC catalog (both endpoints are required)
machine nvcr.io login $oauthtoken password
<token>

machine authn.nvidia.com login $oauthtoken
password <token>

# DockerHub
machine auth.docker.io login <login> password
<password>

# Github.com Container Registry (GITHUB_TOKEN
needs read:packages scope)
machine ghcr.io login <username> password
<GITHUB_TOKEN>
```

Container Engine - Hooks

- Containers do not natively support hardware-specific optimisations

- Hooks
  - inject host libraries and generally help making optimised libraries available
  - Provide things like ssh to the container
  - Normally enabled via annotations
  - See the CE documentation

- Automatically enabled hooks: libcxi, slurm



```
[nick@alps-daint]$ cat pytorch.toml

# image = "nvcr.io#nvidia/pytorch:24.05-py3"
image =
"/capstor/scratch/cscs/<user>/pytorch-24.01-p
y3-venv/pytorch-24.01-py3-venv.sqsh"
mounts = [
     "/capstor",
     "/users",
]
workdir = "/capstor/scratch/cscs/<user>/"

[env]
FI_CXI_DISABLE_HOST_REGISTER = "1"
FI_MR_CACHE_MONITOR = "userfaultfd"

[annotations.com.hooks]
aws_ofi_nccl.enabled = "true"
aws_ofi_nccl.variant = "cuda12"
```

Container Engine - Building Containers

1. Create credentials file so we can pull NGC containers
2. Set up a dockerfile
   a. Let's call it pytorch:24.01-py3-venv
3. Configure storage locations for podman
   a. $HOME/.config/containers/storage.conf

```
[nick@alps-daint]$ cat
~/.config/enroot/.credentials

machine nvcr.io login $oauthtoken password
<api-token>
```

```
[nick@alps-daint]$ cat
pytorch:24.01-py3-venv

FROM nvcr.io/nvidia/pytorch:24.01-py3

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y
python3.10-venv && apt-get clean && rm -rf
/var/lib/apt/lists/*
```

```
[nick@alps-daint]$ cat storage.conf

[storage]
  driver = "overlay"
  runroot = "/dev/shm/$USER/runroot"
  graphroot = "/dev/shm/$USER/root"
 [storage.options.overlay]
  mount_program =
"/usr/bin/fuse-overlayfs-1.13"
```

Container Engine - Building Containers

4. Build container and create compressed container image
   a. podman build -t pytorch:24.01-py3-venv
   b. enroot import -x mount -o pytorch-24.01-py3-venv.sqsh podman://pytorch:24.01-py3-venv

5. Set up an EDF file

6. Now we can launch the container

```
[nick@alps-daint]$ mkdir $SCRATCH/gemma-inference
[nick@alps-daint]$ cd $SCRATCH/gemma-inference
[nick@alps-daint gemma-inference]$ srun
--environment=gemma-pytorch
--container-workdir=$PWD --pty bash
```

```
[nick@alps-daint]$ srun --pty bash
[nick@nid001234]$ podman build -t
pytorch:24.01-py3-venv .
[nick@nid001234]$ enroot import -x mount -o
pytorch-24.01-py3-venv.sqsh
podman://pytorch:24.01-py3-venv
```

```
[nick@alps-daint]$ cat gemma-pytorch.toml

image =
"/capstor/scratch/cscs/<user>/pytorch-24.01-
py3-venv/pytorch-24.01-py3-venv.sqsh"

mounts = ["/capstor", "/users"]

writable = true

[annotations]
com.hooks.aws_ofi_nccl.enabled = "true"
com.hooks.aws_ofi_nccl.variant = "cuda12"

[env]
FI_CXI_DISABLE_HOST_REGISTER = "1"
FI_MR_CACHE_MONITOR = "userfaultfd"
NCCL_DEBUG = "INFO"
```

Container Engine - Gemma Inference

- Set up python virtual environment on top of container
  - Use --system-site-packages so we don't install over container-provided software
- Install additional dependencies
  - Accelerate (multinode, multigpu support)
  - Transformers
  - huggingface_hub[cli] so we can get access to huggingface models - you need to create an API key (https://huggingface.co/)

```
[nick@nid001234 gemma-inference]$ python -m venv
--system-site-packages ./gemma-venv
[nick@nid001234 gemma-inference]$ source
./gemma-venv/bin/activate
[nick@nid001234 gemma-inference]$ pip install
accelerate==0.30.1 transformers==4.38.1
huggingface_hub[cli]
[nick@nid001234 gemma-inference]$
HF_HOME=$SCRATCH/huggingface huggingface-cli login
```

```
[nick@nid001234 gemma-inference]$ cat gemma-inference.py

from transformers import AutoTokenizer,
AutoModelForCausalLM
import torch
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('--prompt', type=str, default='Write
me a poem about the Swiss Alps.')
parser.add_argument('--model', type=str,
default='google/gemma-7b-it')
args = parser.parse_args()

tokenizer =
AutoTokenizer.from_pretrained('google/gemma-7b-it')
model = AutoModelForCausalLM.from_pretrained(args.model,
device_map="auto")
input_ids = tokenizer(args.prompt,
return_tensors="pt").to("cuda")
outputs = model.generate(**input_ids,
max_new_tokens=1024)
print(tokenizer.decode(outputs[0]))
```

Container Engine - Gemma Inference

- Now we can prompt the LLM!

write me a haiku about switzerland

…

<bos>write me a haiku about switzerland

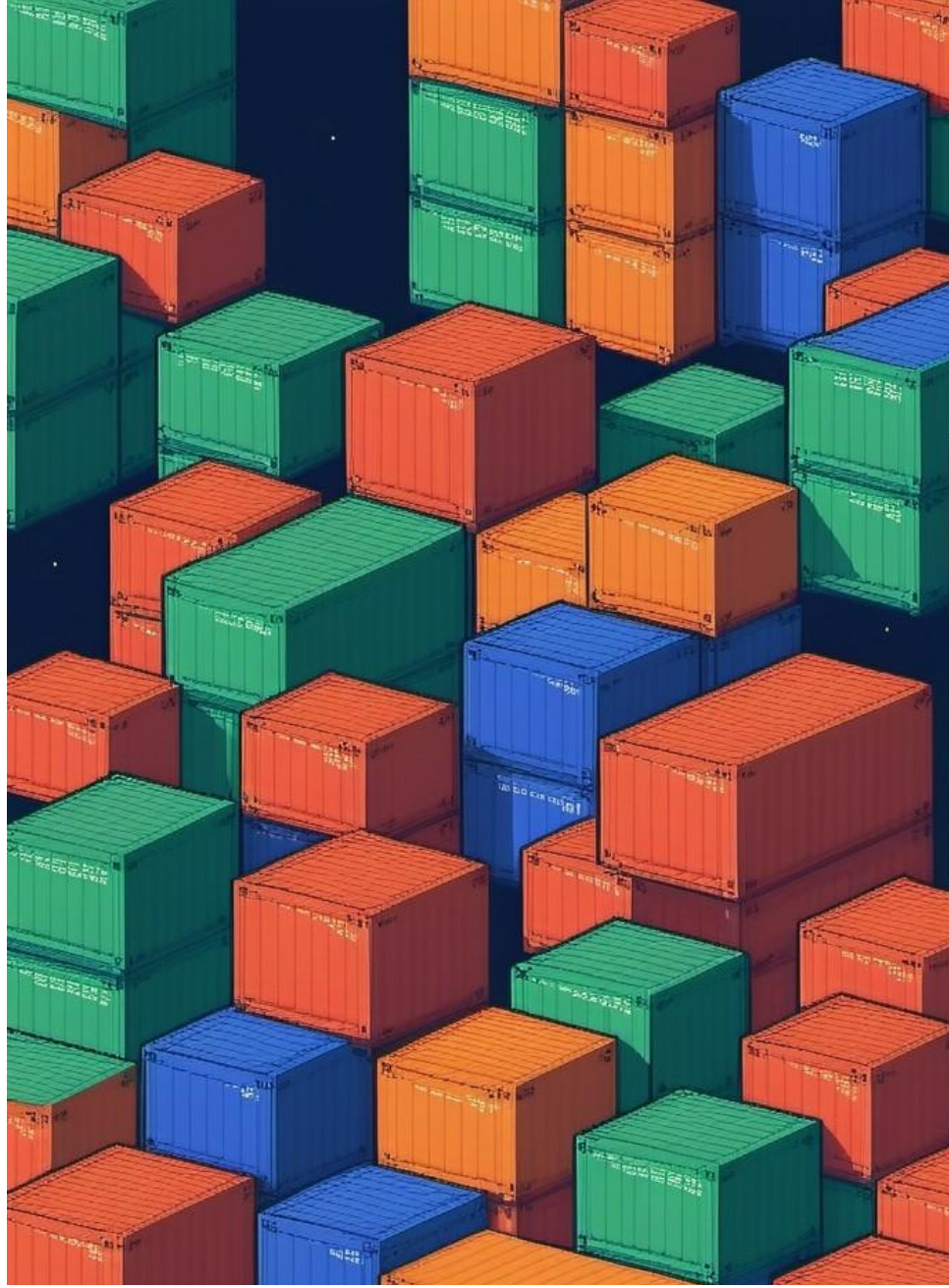
Snow-capped Alps so tall,
Chocolate rivers flow below,
Winter's wonderland.<eos>

```
[nick@nid001234 gemma-inference]$ python
./gemma-inference.py --prompt “write me a
haiku about switzerland”
```

```
[nick@nid001234 gemma-inference]$ cat
gemma-inference.sbatch

#!/bin/bash

#SBATCH --job-name=gemma-inference
#SBATCH --time=00:15:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=288
#SBATCH --environment=gemma-pytorch
#SBATCH --account=<project>

export HF_HOME=$SCRATCH/huggingface
export TRANSFORMERS_VERBOSITY=info

cd $SCRATCH/gemma-inference/
source ./gemma-venv/bin/activate

set -x

python ./gemma-inference.py --prompt “write
me a haiku about switzerland”
```

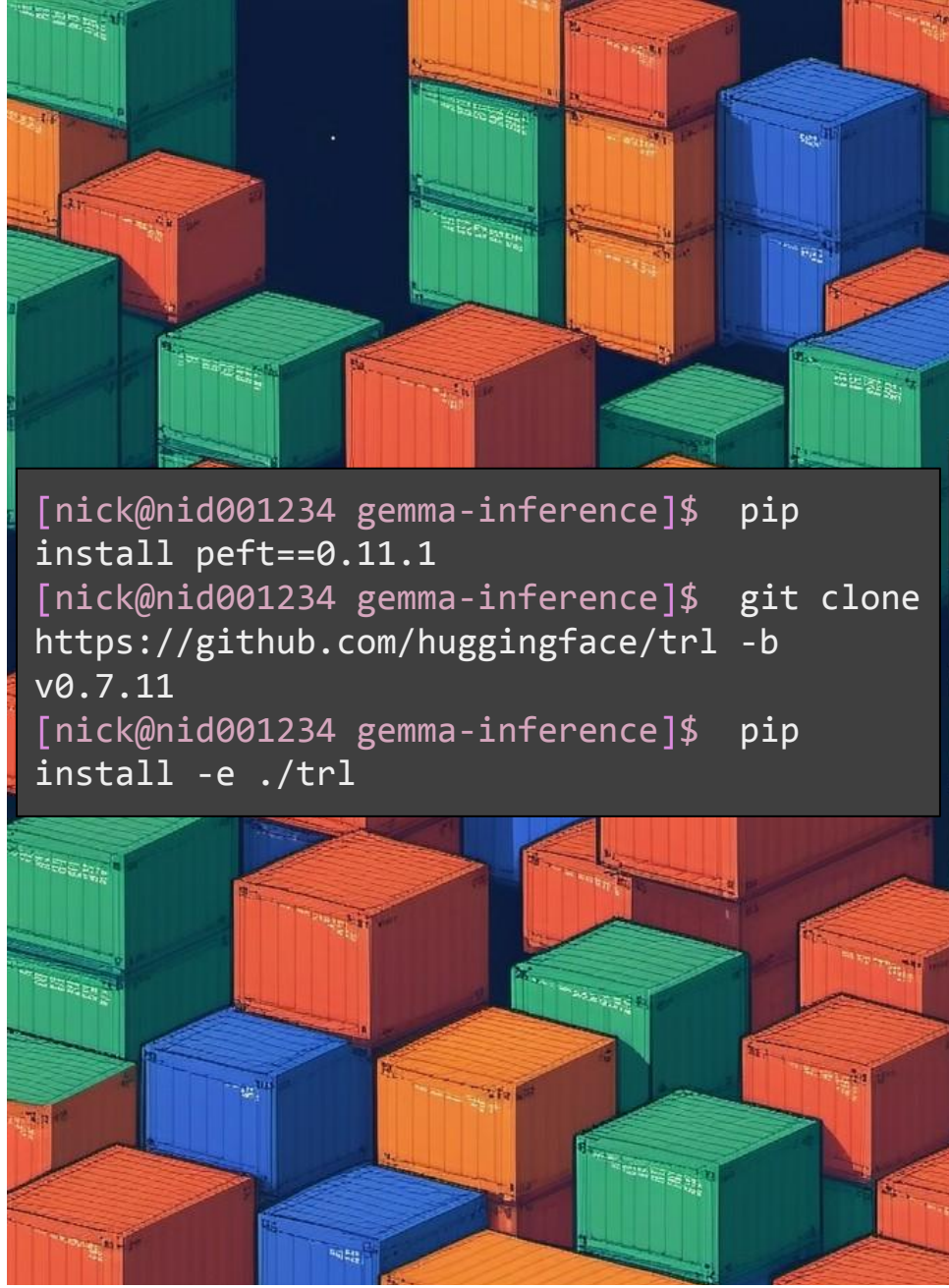
Container Engine - Gemma Fine-Tuning

- A simple example to showcase multinode/multigpu training
- We will fine-tune Gemma-7B on the OpenAssistant dataset
  - https://huggingface.co/datasets/OpenAssistant/oasst_top1_2023-08-25



Container Engine - Gemma Fine-Tuning

- We use the accelerate package from huggingface
  - https://huggingface.co/docs/accelerate
- We use the Parameter Efficient Fine-Tuning (PEFT) and Transformer Reinforcement Learning (TRL) packages to finetune the model
  - https://github.com/huggingface/trl
  - https://github.com/huggingface/peft



```
[nick@nid001234 gemma-inference]$  pip
install peft==0.11.1
[nick@nid001234 gemma-inference]$  git clone
https://github.com/huggingface/trl -b
v0.7.11
[nick@nid001234 gemma-inference]$  pip
install -e ./trl
```

Container Engine - Gemma Fine-Tuning

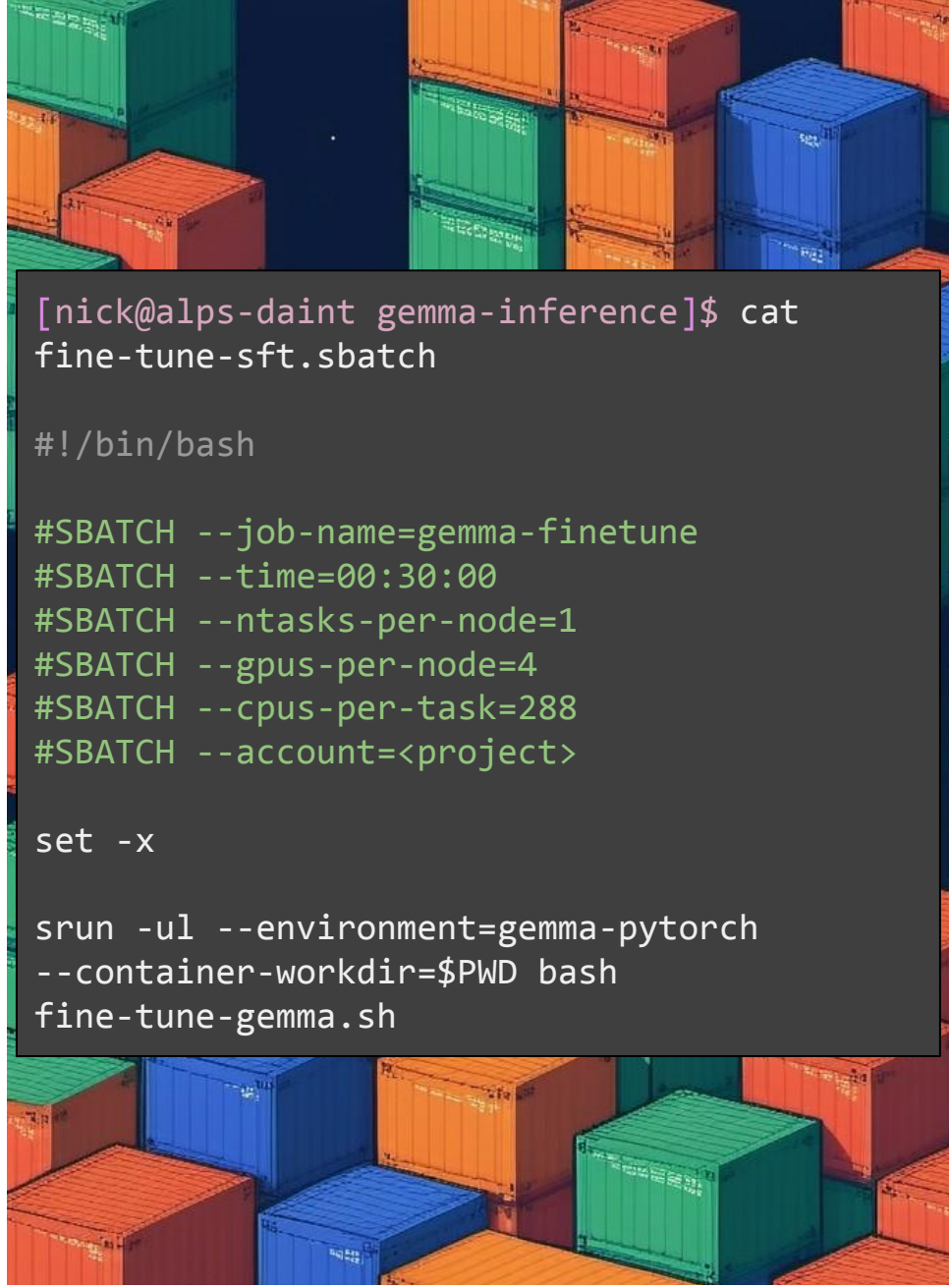
- We need to create a fine-tune-gemma.sh script that configures accelerate and launches the training job

```
[nick@nid001234 gemma-inference]$ cat fine-tune-sft.sbatch

#!/bin/bash

source ./gemma-venv/bin/activate

set -x

export HF_HOME=$SCRATCH/huggingface
export TRANSFORMERS_VERBOSITY=info

ACCEL_PROCS=$(( $SLURM_NNODES * $SLURM_GPUS_PER_NODE ))

MAIN_ADDR=$(echo "${SLURM_NODELIST}" | sed 's/[],].*//g;
s/\[//g')
MAIN_PORT=12802

accelerate launch --config_file
trl/examples/accelerate_configs/multi_gpu.yaml \
          --num_machines=$SLURM_NNODES
--num_processes=$ACCEL_PROCS \
          --machine_rank $SLURM_PROCID \
          --main_process_ip $MAIN_ADDR --main_process_port
$MAIN_PORT \
          trl/examples/scripts/sft.py \
          --model_name google/gemma-7b \
          --dataset_name OpenAssistant/oasst_top1_2023-08-25
\
          --per_device_train_batch_size 2 \
          --gradient_accumulation_steps 1 \
          --learning_rate 2e-4 \
          --save_steps 200 \
          --max_steps 400 \
            --use_peft \
          --lora_r 16 --lora_alpha 32 \
          --lora_target_modules q_proj k_proj v_proj o_proj
\
          --output_dir gemma-finetuned-openassistant
```

Container Engine - Gemma Fine-Tuning

- We need to create a fine-tune-gemma.sh script that configures accelerate and launches the training job

- We also need a short Slurm batch script to launch our job



```
[nick@alps-daint gemma-inference]$ cat
fine-tune-sft.sbatch

#!/bin/bash

#SBATCH --job-name=gemma-finetune
#SBATCH --time=00:30:00
#SBATCH --ntasks-per-node=1
#SBATCH --gpus-per-node=4
#SBATCH --cpus-per-task=288
#SBATCH --account=<project>

set -x

srun -ul --environment=gemma-pytorch
--container-workdir=$PWD bash
fine-tune-gemma.sh
```

Container Engine - Gemma Fine-Tuning

- We need to create a fine-tune-gemma.sh script that configures accelerate and launches the training job
- We also need a short Slurm batch script to launch our job
- Let's launch it with 2 nodes

```
[nick@alps-daint gemma-inference]$ cat
fine-tune-sft.sbatch

#!/bin/bash

#SBATCH --job-name=gemma-finetune
#SBATCH --time=00:30:00
#SBATCH --ntasks-per-node=1
#SBATCH --gpus-per-node=4
#SBATCH --cpus-per-task=288
#SBATCH --account=<project>

set -x

srun -ul --environment=gemma-pytorch
--container-workdir=$PWD bash
fine-tune-gemma.sh
```

```
[nick@alps-daint gemma-inference]$ sbatch
--nodes=2 fine-tune-sft.sbatch
```

Container Engine - Gemma Fine-Tuning

- Now we can instruct our code to use the fine-tuned model

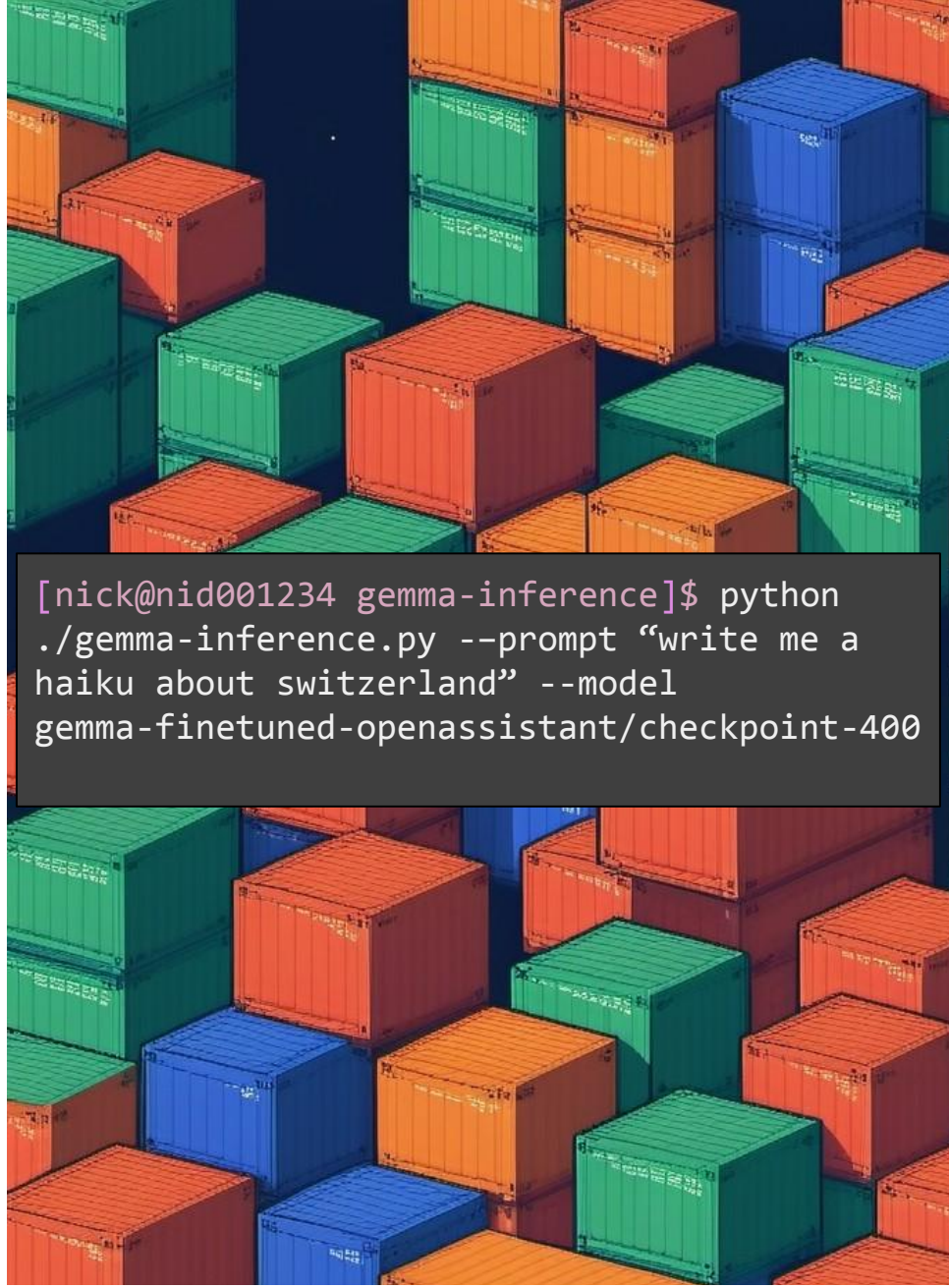
write me a haiku about switzerland

…

<bos>write me a haiku about switzerland

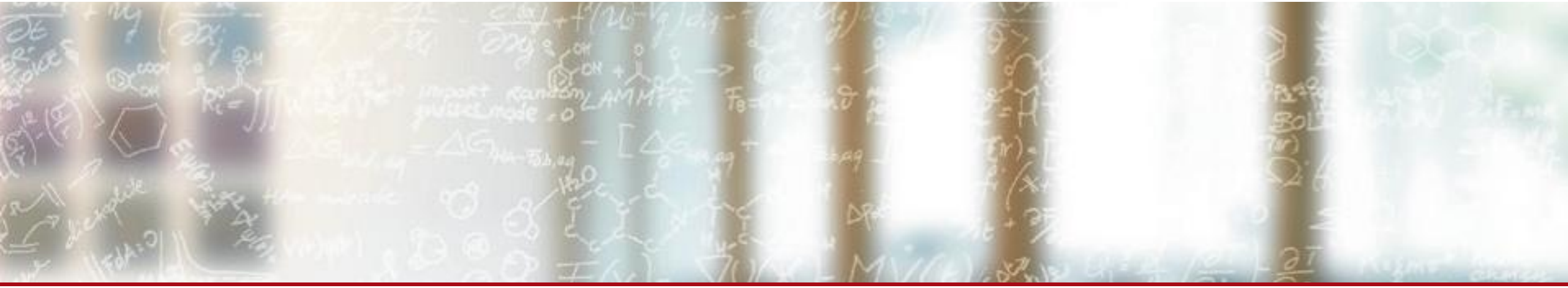
A haiku about Switzerland:

The Alps, so high and steep,
The cows, so cute and sweet,
The cheese, so rich and creamy,
Switzerland, so beautiful and serene.



Useful Links

- Alps (daint) Early Access
  - https://confluence.cscs.ch/display/KB/Alps+%28Daint%29+early+access
- Container Engine
  - https://confluence.cscs.ch/display/KB/Container+Engine
- ML Tutorials on Alps (todi)
  - https://confluence.cscs.ch/display/KB/LLM+Inference
  - https://confluence.cscs.ch/display/KB/LLM+Finetuning
  - https://confluence.cscs.ch/display/KB/Nanotron+Training
- NGC Container Catalogue
  - https://catalog.ngc.nvidia.com/



Thanks for your attention!