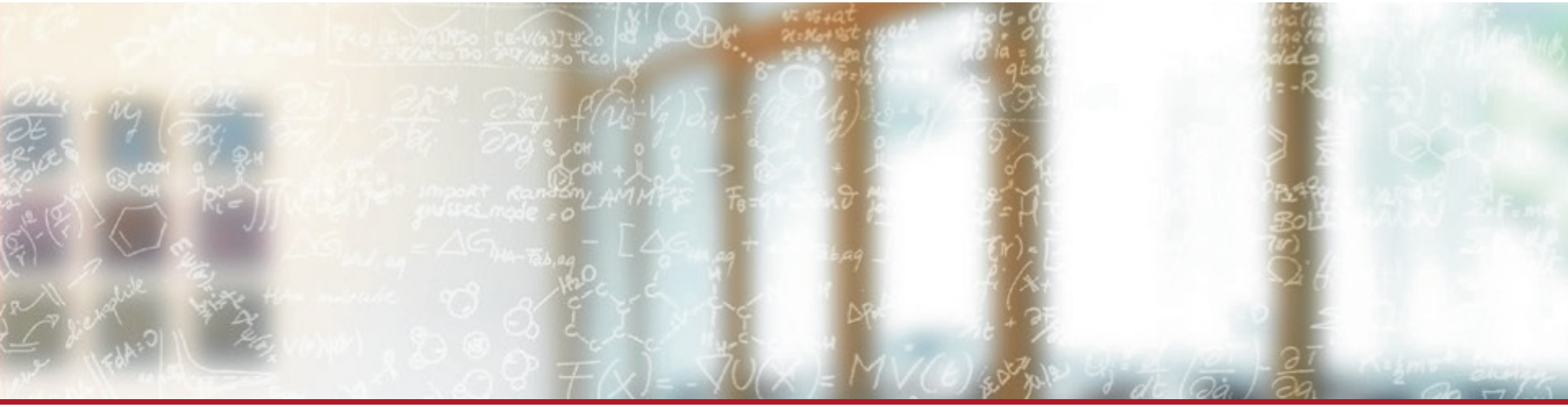
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



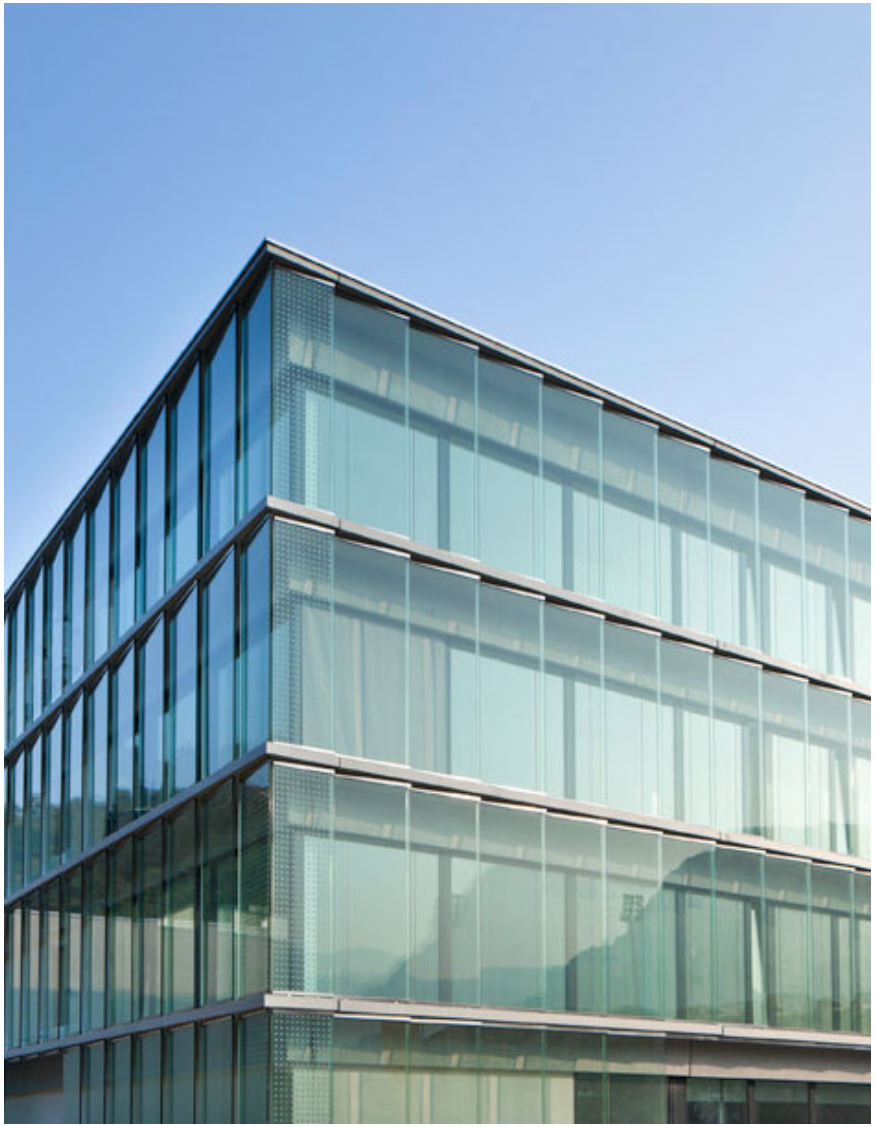
HPC Containers at CSCS

CSCS User Lab Day - Meet the Swiss National Supercomputing Centre

Theofilos Manitaras, Rafael Sarmiento - ETH Zurich/CSCS

September 11, 2018

HPC Containers at CSCS



- Introducing Containers
- Docker and DockerHub
- From Docker to Shifter
- Shifter basics
- Native MPI and GPU support
- NVidia GPU Cloud registry

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Introducing Containers

Which problem do containers solve?

- Containers solve the problem of making your software to run reliably when moved from one computing environment to another.
- A container consists of an entire runtime environment, i.e. an application, plus all its dependencies, libraries and other binaries and configuration files needed to run, bundled in one package.

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Docker and DockerHub

Introducing Docker

- Docker is a computer program that performs operating-system-level virtualization, also known as *containerization*.
- Docker consists of a command-line interface (cli), a background daemon, and a set of remote services.
- While there are other available container platforms, Docker is the most popular one today.

Using the Docker cli (1/3)

1. Run the hello-world Docker container

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
9db2ca6ccae0: Pull complete
Digest: sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac44aca35a85c90c5e3c7afacdc
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

- **Image:** An executable package that contains everything needed by an application to run; the code, a runtime, libraries, environment variables, and configuration files.
- **Container:** A runtime instance of an image, i.e. what the image becomes in memory when executed.

Using the Docker cli (2/3)

2. List the available images in your computer

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              2cb0d9787c4d        6 weeks ago         1.85kB
```

3. Run the ubuntu container interactively:

```
$ docker run -it --name my_ubuntu ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
124c757242f8: Pull complete
2ebc019eb4e2: Pull complete
dac0825f7ffb: Pull complete
82b0bb65d1bf: Pull complete
ef3b655c7f88: Pull complete
Digest: sha256:72f832c6184b55569be1cd9043e4a80055d55873417ea792d989441f207dd2c7
Status: Downloaded newer image for ubuntu:latest
root@73fb1d709762:/# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.1 LTS"
```

Using the docker cli (3/3)

- **Layer:** A layer is a collection of changes in files. An image consists of one or more read-only layers stacked one over the other.
- **Tag:** Tags are aliases to an image ID. They convey useful information about a specific image version/variant.

4. List all the containers

```
$ docker container ls -a
```

5. Export the container as a tar file.

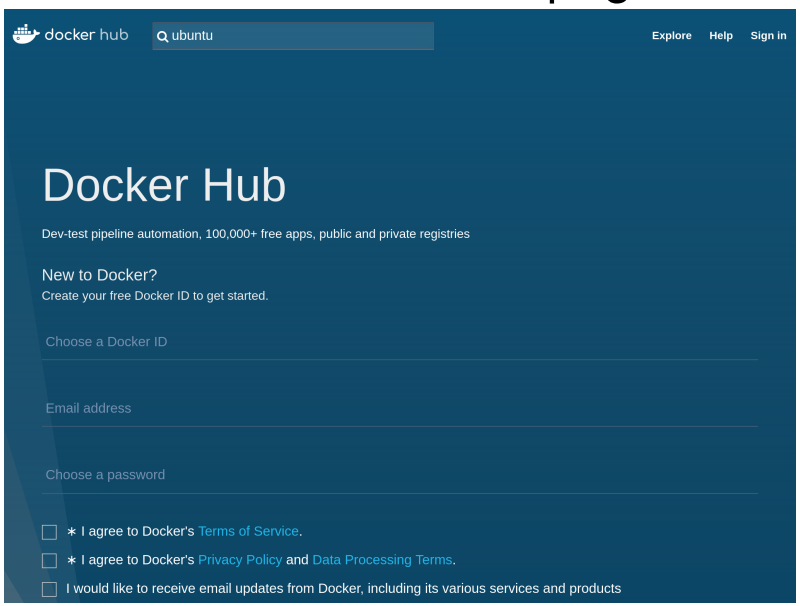
```
$ docker export my_ubuntu -o my_ubuntu_image.tar
```

Docker Hub(1/2)

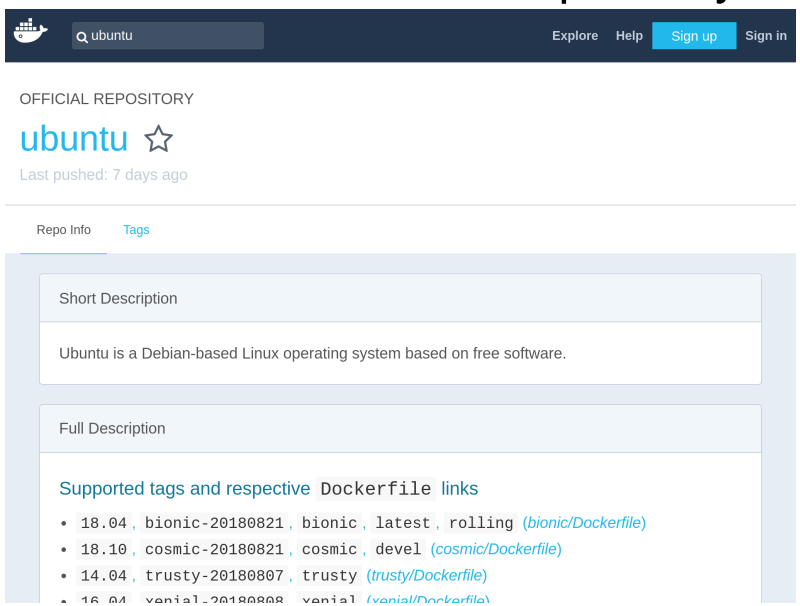
Docker Hub is the default cloud-based registry service for Docker images.

- **Registry:** A storage and content delivery system, holding named container images, available in different tagged versions.
- **Repository:** A named bucket of images.

Docker Hub main page



Docker Hub ubuntu repository



Docker Hub(2/2)

- The Docker Hub web interface can be used to get information on available images.
- The Docker cli can also be used to search for images in DockerHub. For example, to search for official **ubuntu** images and print their name/description, use:

```
$ docker search --filter "is-official=true" --format "{{.Name}}: {{.Description}}" ubuntu
ubuntu: Ubuntu is a Debian-based Linux operating sys
ubuntu-upstart: Upstart is an event-based replacement for th
neurodebian: NeuroDebian provides neuroscience research s
ubuntu-debootstrap: debootstrap --variant=minbase --components=m
```

- To push images to Docker Hub, an account is needed. The following commands are used to push the image:

```
$ docker login
```

Type the username and password for your Docker Hub account when prompted.

```
$ docker push <user name>/<repo name>:<image tag>
```

Dockerfiles

- Dockerfiles are text documents which contain the necessary instructions to construct an image.
- Special commands are used in order to build the image. For more information, visit the official Dockerfile reference.
- Simple Dockerfile:

```
# Simple Dockerfile of ubuntu:latest with Python 3

# Base image
FROM ubuntu:latest

# Install Python 3
RUN apt-get update && \
    apt-get install python3 -y

# Check the Python version
CMD ["python3", "--version"]
```

Advantages of containers

- Using containers allows to deploy applications across operating systems without having to build and configure separately.
- Programs running inside Docker containers interface directly with the host's Linux kernel. Thus, in contrast to virtual machines which virtualize the hardware and need a complete operating system, containers are faster to deploy and run.
- Effortless sharing via image registries and/or the recipes used to produce the image (Dockerfiles).
- Suitable with modern software developent practices (CI/CD).

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

From Docker to Shifter

Containers for HPC with Shifter

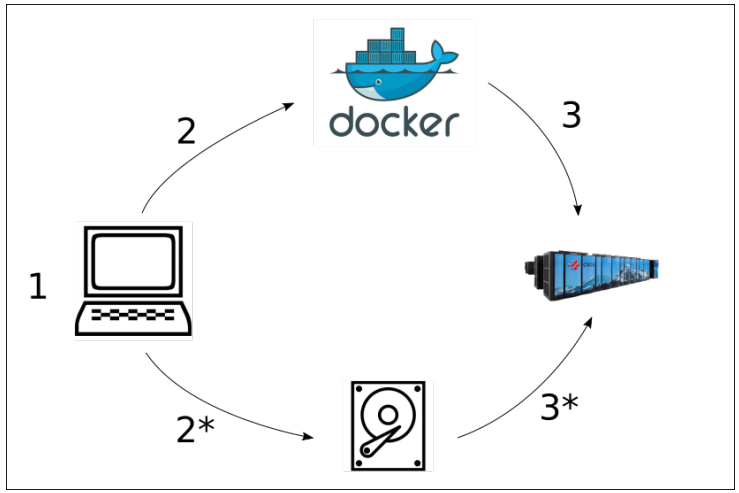
CSCS provides Shifter for running container workloads on HPC systems, addressing the unique needs of high-performance environments. It's key features are:

- Spawning of software environments (containers), built by users to fit the deployment of a specific application.
- Security oriented to HPC systems.
- Native performance of custom HPC hardware.
- Compatible with Docker, whose images can be pulled from cloud repositories, private repositories (password protected) or directly loaded from tar archives.

Transferring my images to Daint

Two paths can be followed to transfer the images to Daint:

1. $(1 \rightarrow 2 \rightarrow 3)$: Build the image, upload to Docker Hub, pull on Daint using shifter.
2. $(1 \rightarrow 2^* \rightarrow 3^*)$: Build the image, export image to tar file, copy and build on Daint.



Shifter basics

Shifter basics

```
# Load shifter-ng module (shifter-ng is a version of shifter developed at CSCS)
module load daint-gpu # or daint-mc
module load shifter-ng
```

```
# Pull image from DockerHub (Docker registry)
# shifter pull [shifter pull options] repo-name/image-name:tag
# index.docker.io/library is the default repo-name if none is specified
srun -C gpu shifter pull debian:jessie
```

* We strongly recommend to run the shifter pull command on the compute nodes through Slurm, so that Shifter can take advantage of their large RAM filesystem, which will greatly reduce the pull process time and will allow to pull larger images

```
# List pulled images
shifter images
# REPOSITORY     TAG         DIGEST        CREATED              SIZE        SERVER
# library/alpine latest      9797e5e798a0  2018-01-19T09:44:41 1.91MB      index.docker.io
# library/debian jessie      6bc9d2cd1831  2018-01-19T09:43:04 40.16MB     index.docker.io
```

```
# Remove images
# shifter rmi repo-name/image-name:tag
shifter rmi debian:jessie
# Removed index.docker.io/library/debian/jessie
```

```
# Run a command within a new container (run command and exit)
# shifter run [shifter run options] repo-name/image-name:tag command
srun -C gpu shifter run debian:jessie uname -s -n -r -m
# Linux nid0xxxx 4.4.103-6.38_4.0.134-cray_ari_c x86_64
```

```
# Run an interactive shell within the container
srun -C gpu --pty shifter run debian:jessie bash
# The option "--pty" is used to have a command prompt on the interactive session:
# user@nid0xxxx:/current/dir $> _
```

Shifter basics

```
# Help
$> shifter --help
Usage: shifter COMMAND
Options:
  --help              Print help
  --version           Print version information and quit
  --debug             Enable debug mode (print all log messages with DEBUG
                      level or higher)
  --verbose           Enable verbose mode (print all log messages with INFO
                      level or higher)
Commands:
   help: Print help message
   images: List images
   load: Load the contents of a tarball to create a filesystem image
   pull: Pull an image from a registry
   rmi: Remove an image
   run: Run a command in a new container
```

```
# shifter help command
$> shifter help run
Usage: shifter run [OPTIONS] [SERVER/]IMAGE[:TAG] [COMMAND] [ARG...]
Run a command in a new container
Options:
  --mount arg             Mount custom directories into the container
  -m [ --mpi ]            Enable MPI support
  --writable-volatile arg Make specified directory writable volatile. All
                          changes will be discarded after the container
                          exits.
  --centralized-repository Use centralized repository instead of the local one
```

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Docker and Shifter

Creating Images with Dockerfiles

* Green panels correspond to commands that are ran in a local computer

```
FROM debian:jessie # base image

# Run a command to modify something inside the image
RUN apt-get update && apt-get install -y build-essential wget --no-install-recommends \
    && rm -rf /var/lib/apt/lists/*

# Installing mpich-3.1.4
RUN wget -q http://www.mpich.org/static/downloads/3.1.4/mpich-3.1.4.tar.gz \
    && tar xf mpich-3.1.4.tar.gz \
    && cd mpich-3.1.4 \
    && ./configure --disable-fortran --enable-fast=all,O3 --prefix=/usr \
    && make -j$(nproc) \
    && make install \
    && ldconfig \
    && cd .. \
    && rm -rf mpich-3.1.4 && rm mpich-3.1.4.tar.gz
```

- Other frequently used instructions are COPY and ENV. For more info see the Dockerfile reference.

- Reducing the size of the container image, besides saving disk space, also speeds up the process of importing it into Shifter later on. The easiest ways to limit image size are cleaning the package manager cache after installations and deleting source codes and other intermediate build artifacts when building software manually. For practical examples and general good advice, please refer to the official Best practices for writing Dockerfiles.

```
# docker build -t "new-image-name:tag" -f Dockerfile
docker build -t "debian-mpich:cscs" -f Dockerfile
```

```
$> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
debian-mpich        cscs                39f70b8f8951        3 hours ago         326MB
```

Move Image to Piz Daint

```
# Save image as a tar file:
docker save --output debian-mpich-cscs.tar debian-mpich:cscs

# then move it to Piz Daint (scp debian-mpich-cscs.tar daint:some/dir)
```

```
# Load the contents of the tarball to create a filesystem image
srun -C gpu shifter load debian-mpich-cscs.tar debian-mpich:cscs
```

```
$> shifter images
REPOSITORY           TAG         DIGEST       CREATED              SIZE        SERVER
library/debian-mpich cscs        1c960e73e579 2018-08-16T10:33:43 149.21MB    load
```

```
# Run a command
srun -C gpu shifter run load/library/debian-build:tutorial uname -s -n -r -m
# Linux nid0xxxx 4.4.103-6.38_4.0.134-cray_ari_c x86_64
```

Native MPI Support

Shifter provides the container with access to the compute node's MPI implementation in order to use the Aries Interconnect of Piz Daint. To take advantage of this feature, the MPI installed in the container (and dynamically linked to your application) needs to be ABI-compatible with the compute node's MPI on Piz Daint. To best meet the required ABI-compatibility, it is recommend that the container application uses one of the following MPI implementations:

- MPICH v3.1.4 (Feburary 2015)
- MVAPICH2 2.2 (September 2016)
- Intel MPI Library Library 2017 Update 1

Native MPI Support

```
# Run the container in interactive mode to compile the mpi code
srun -C gpu --pty shifter run load/library/debian-mpich:cscs bash
```

```
# The option "--mpi" to "shifter run" is used to run a container on Piz Daint with MPI support
# example: OSU benchamark to measure latency of the MPI communication between two nodes
srun -C gpu -N 2 -n 2 shifter run --mpi load/library/debian-mpich:cscs $SCRATCH/osu/latency.x
#
# Output (not on Piz Daint):
# # OSU MPI Latency Test
# # Size          Latency (us)
# 0                     1.09
# 1                     1.10
# 2                     1.10
# 4                     1.10
# 8                     1.11
# 16                    1.12
# 32                    1.12
# 64                    1.12
# 128                   1.13
# 256                   1.14
# 512                   1.17
# 1024                  1.40
```

Native GPU Support

Shifter allows any container to access the GPUs on the compute nodes. To take advantage of this feature on Piz Daint, the container needs to include the CUDA 8.0 Runtime. One way to achieve this is by basing your container on the official Docker image provided by NVIDIA:

```
# Dokerfile
FROM nvidia/cuda:8.0
```

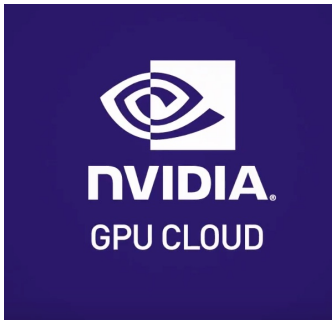
* On your workstation, we recommend using nvidia-docker to run and test the container using an NVIDIA GPU.

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

NVidia GPU Cloud

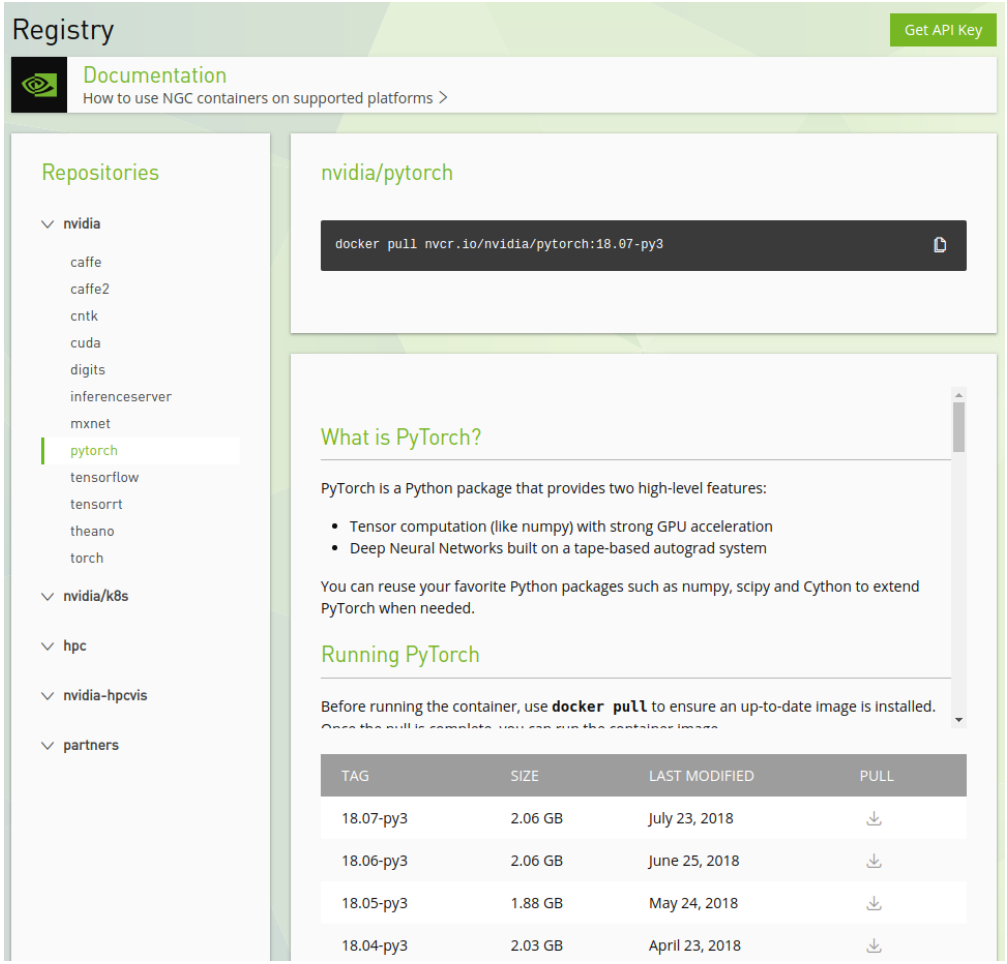
NVidia GPU Cloud: Access and Setup



The NVIDIA GPU Cloud (NGC) registry provides docker images for running many common deep learning frameworks within containers.
https://ngc.nvidia.com

- Create and account on https://ngc.nvidia.com.
- Generate API Key.
- Use the API key as password and the generic username "$oauthtoken" to download images with Shifter.

NVidia GPU Cloud: Registry



NVidia GPU Cloud: Run Containers with Shifter

```
module load daint-gpu
module load shifter-ng
```

```
# Pull image with shifter
# docker pull nvcr.io/nvidia/pytorch:18.07-py3
srun -C gpu shifter pull --login nvcr.io/nvidia/pytorch:18.07-py3
```

```
# List the images downloaded (images are stored in $SCRATCH/.shifter)
$> shifter images
REPOSITORY     TAG         DIGEST        CREATED              SIZE        SERVER
nvidia/pytorch 18.07-py3   b2998beea62f  2018-08-14T10:14:45 2.00GB      nvcr.io
```

```
# Use Shifter to run a command within a container
# srun [slurm options] shifter run [shifter run options] nvcr.io/nvidia/app-name:tag command
#
# Example: run the command <python -c "import torch; print(torch.cuda.is_available())"> within
# the container
srun -C gpu shifter run nvcr.io/nvidia/pytorch:18.07-py3 python -c "import torch;
print(torch.cuda.is_available())"
# True
```

```
# Delete image
shifter rmi nvcr.io/nvidia/pytorch:18.07-py3
```

Example: MNIST dataset with PyTorch

```
cd $SCRATCH
mkdir mnist_example
cd mnist_example

# Download the PyTorch example for the MNIST dataset from branch "0.4" of
# the pytorch/examples repository (https://github.com/pytorch/examples)
wget https://raw.githubusercontent.com/pytorch/examples/0.4/mnist/main.py
# and change the download location of the dataset inside main.py ("../data" -> "./data").
```

```
#!/bin/bash -l

#SBATCH --job-name=mnist_pytorch
#SBATCH --time=00:05:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-core=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=12
#SBATCH --partition=normal
#SBATCH --constraint=gpu

module load daint-gpu
module load shifter-ng

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun shifter run nvcr.io/nvidia/pytorch:18.07-py3 python main.py
```

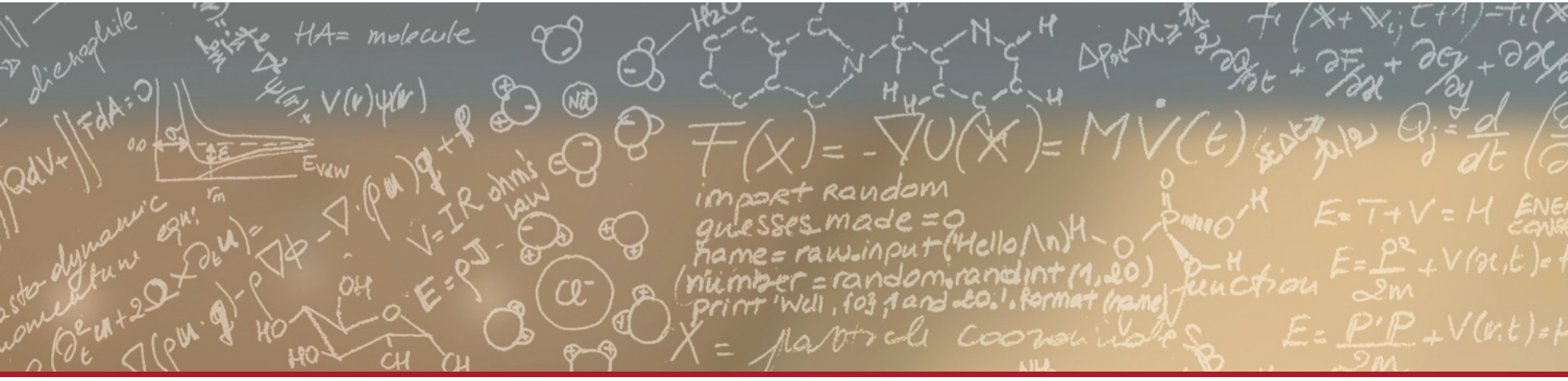
Example: MNIST dataset with PyTorch

```
# After a couple of messages from shifter the output looks like this:
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Processing...
Done!
Train Epoch: 1 [0/60000 (0%)] Loss: 2.373651
Train Epoch: 1 [640/60000 (1%)] Loss: 2.310517
Train Epoch: 1 [1280/60000 (2%)]    Loss: 2.281828
Train Epoch: 1 [1920/60000 (3%)]    Loss: 2.315809
...
...
...
Train Epoch: 10 [57600/60000 (96%)] Loss: 0.375870
Train Epoch: 10 [58240/60000 (97%)] Loss: 0.172069
Train Epoch: 10 [58880/60000 (98%)] Loss: 0.379510
Train Epoch: 10 [59520/60000 (99%)] Loss: 0.294709

Test set: Average loss: 0.0490, Accuracy: 9828/10000 (98%)
```

Documentation at CSCS User Portal

- How to run containers on Piz Daint
- Advanced Shifter documentation



Thank you for your attention.