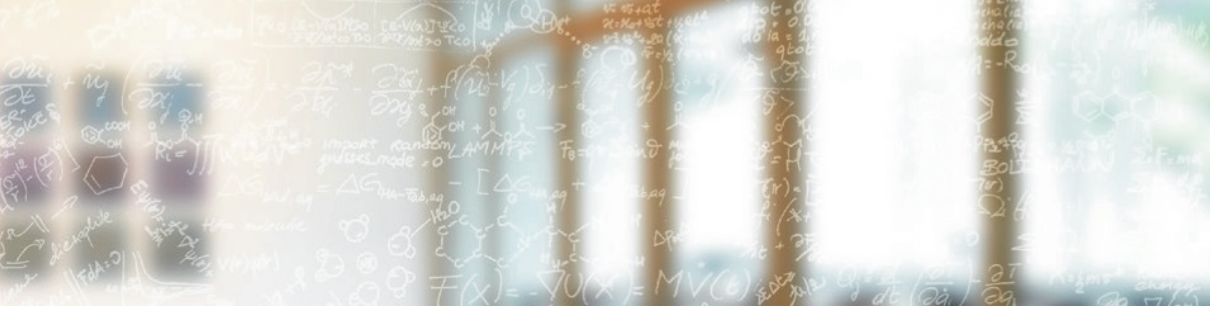


Advances in HPC container platforms at CSCS

Theofilos Manitaras and Rafael Sarmiento
ETHZürich / CSCS
CSCS User Lab Day - Luzern, September 9th, 2019

ETH zürich
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Outline

- Containers in a nutshell
- Docker & DockerHub
- Using Shifter at CSCS
- Using Singularity at CSCS
- Successfull use case by Parisa Khateri

Which problem do containers solve?

- Containers provide software portability between different computing environments.
- A container image consists of an entire runtime environment, i.e. an application, plus all its dependencies, libraries and configuration files.
- The container platforms supported by CSCS are Shifter & Singularity.
- Both container platforms allow running at scale using MPI and support GPU-enabled applications.

Introduction to 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 (1/2)

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
1b930d010525: Pull complete
Digest: sha256:451ce787d12369c5df2a32c85e5a03d52cbcef6eb3586dd03075f3034f10adcd
Status: Downloaded newer image for hello-world:latest

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

- **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 (2/2)

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
.
root@f82930c27cdf:/# cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME='Ubuntu 18.04.3 LTS'
```

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

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.
- 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.
- To push images to Docker Hub, an account is needed.

Container Advantages

- Deploy applications across operating systems without the need for reconfiguration/rebuilding.
- Containers interface directly with the Linux kernel of the host and thus deployment and execution is faster compared to virtual machines.
- Easy sharing via image registries, definition files and image archives.
- Compatibility with modern software development practices (CI/CD).

Containers in HPC

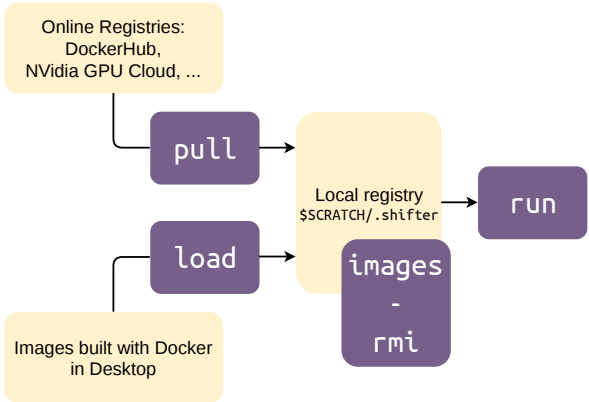
- Images are generally built for a single application, often with complicated dependencies.
- MPI support needs to be available.
- Container engines with no daemons are preferred.
- Containers are required to be run without sudo privileges.
- Examples of Container for HPC are Shifter and Singularity.

Shifter

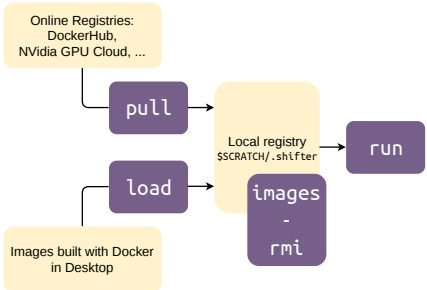
- Spawning of containers built by users to fit the deployment of specific applications
- Security oriented to HPC systems
- Native performance of custom HPC hardware
- Compatibility with Docker

```
module load daint-gpu # or daint-mc
module load shifter-ng
```

Shifter


Online Registries:
DockerHub,
NVidia GPU Cloud, ...
pull
load
Images built with Docker
in Desktop
Local registry
$SCRATCH/.shifter
images
-
rmi
run

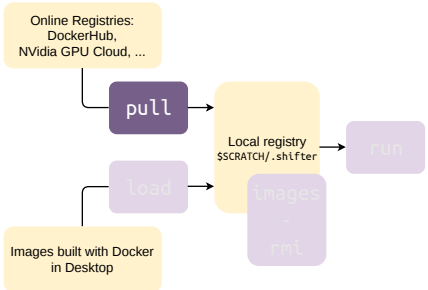
Shifter

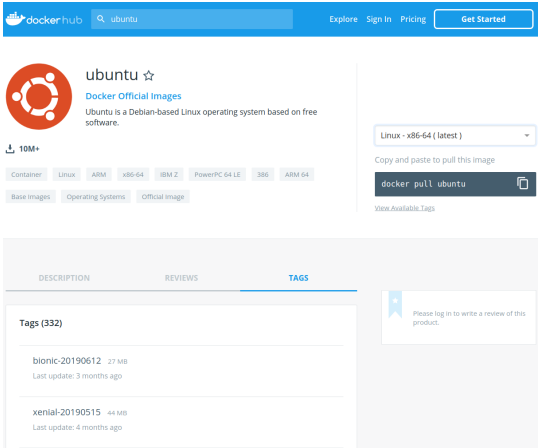



```
~> shifter --help
Usage: shifter COMMAND
Options:
 --help     Print help
 --version  Print version information and quit
 --debug    Enable debug mode (print log messages with DEBUG level or higher)
 --verbose  Enable verbose mode (print log messages with INFO level or higher)
Commands:
 help:      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 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.
```

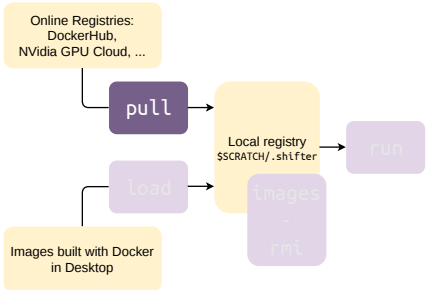
Shifter: Pulling Images from Online Registries





https://hub.docker.com

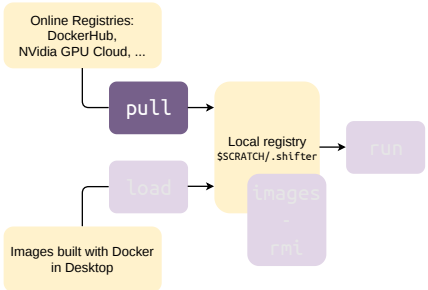
Shifter: Pulling Images from Online Registries

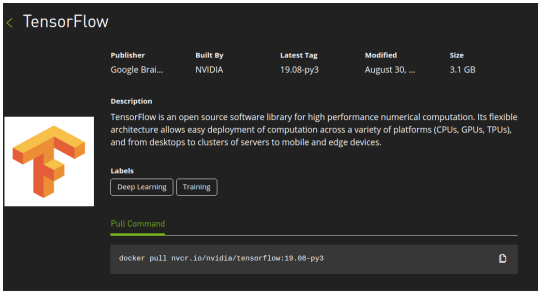



```
srun shifter pull ubuntu:bionic-20190612
```

- It's recommended to run `shifter pull` on the compute nodes through Slurm, so that Shifter can take advantage of their large RAM. This will reduce the pull process time and allow to pull larger images

Shifter: Pulling Images from Online Registries



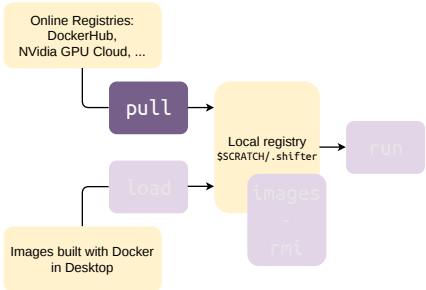


Overview Tags Layers

TAG	LAST UPDAT...	SIZE	PULL TAG
19.08-py3	August 30, 2019	3.1 GB	
19.08-py2	August 30, 2019	2.93 GB	
19.07-py3	July 31, 2019	2.82 GB	
19.07-py2	July 31, 2019	2.66 GB	

https://ngc.nvidia.com

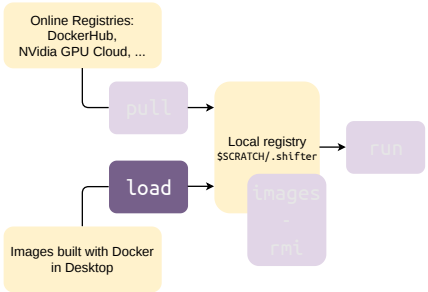
Shifter: Pulling Images from Online Registries

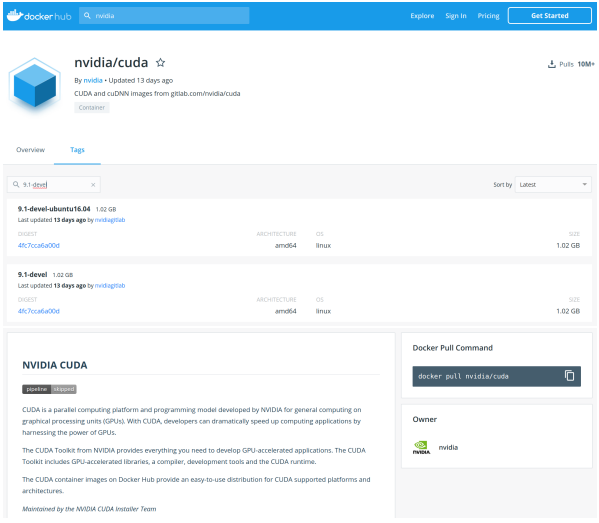



```
srun shifter pull --login nvcr.io/nvidia/tensorflow:19.08-py3
```

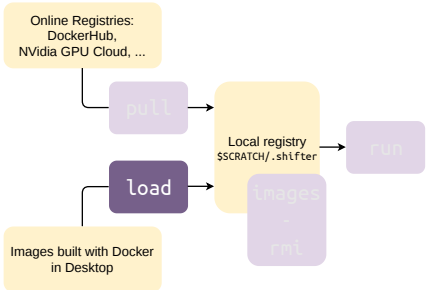
- Create and account on https://ngc.nvidia.com.
- Generate API Key.
- Use the API key as password and the generic username `$oauthtoken` to download images with Shifter.

From Dockerfiles to Shifter Images





From Dockerfiles to Shifter Images

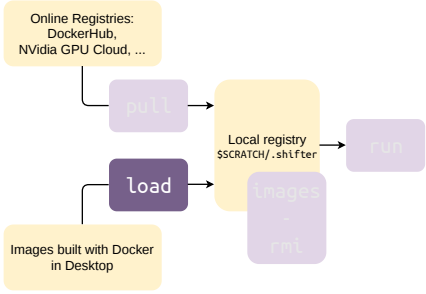



```
# Dockerfile
FROM nvidia/cuda:9.1-devel # base image

RUN apt-get update && \
    apt-get install -y build-essential wget --no-install-recommends && \
    rm -rf /var/lib/apt/lists/*

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 -j4 \
    && make install \
    && ldconfig \
    && cd .. \
    && rm -rf mpich-3.1.4 && rm mpich-3.1.4.tar.gz
```

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

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

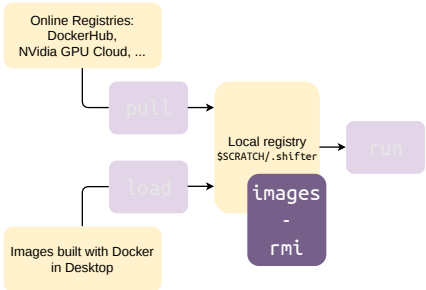
From Dockerfiles to Shifter Images




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

- Same as with `shifter pull`, it's recommended to run `shifter pull` on the compute nodes through Slurm.

Shifter: Listing Images in Local Registry



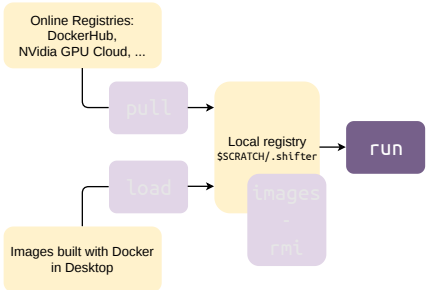

```
~> shifter images
```

```
# REPOSITORY        TAG              DIGEST CREATED    SIZE     SERVER
# library/ubuntu    bionic-20190612  979a0  2018-09-01 28.91MB  index.docker.io
# nvidia/tensorflow 19.03-py3        2bf8f  2019-04-23 2.70GB   nvcr.io
# library/deb-mpich cscs             1c960  2018-08-16 149.21MB load
```

```
~> shifter rmi nvcr.io/nvidia/tensorflow:19.03-py3
# removed image nvcr.io/nvidia/tensorflow/19.03-py3

# shifter rmi <SERVER>/<REPOSITORY>:<TAG>
```

```
# REPOSITORY        TAG              DIGEST CREATED    SIZE     SERVER
# library/ubuntu    bionic-20190612  979a0  2018-09-01 28.91MB  index.docker.io
# library/deb-mpich cscs             1c960  2018-08-16 149.21MB load
```

Shifter: Running Commands within Containers



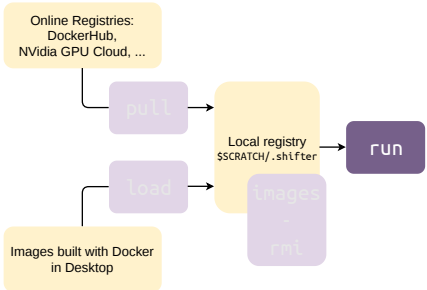
- Using the command `bash` allows to run the container interactively.

```
srun shifter run load/library/deb-mpich:cscs \
    bash -c 'export MY_VARIABLE=SOMETHING && my-app.x'
```

- To run a container with MPI support, `shifter run` needs the option `--mpi`

```
~> srun -N 2 -n 2 shifter run --mpi load/library/debian-mpich:cscs \
                              $SCRATCH/osu/latency.x
# Output
# # 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

# This example is the OSU benchmark to measure latency of
# the communication between two nodes
```

Shifter: Running Commands within Containers



- Using `bash` as command allows to run the container interactively.

```
srun --pty shifter run load/library/deb-mpich:cscs bash
```

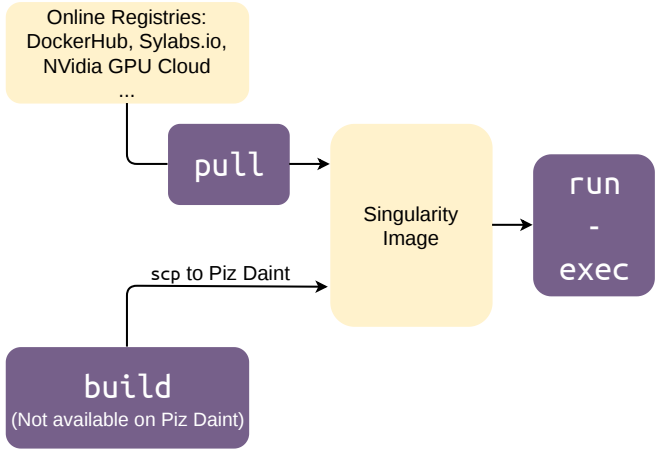
- The options `--pty` of `srun` is needed to see a command prompt when running interactively.

Singularity

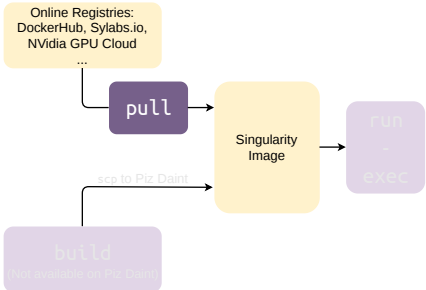
- Spawning of containers built by users to fit the deployment of specific applications
- Security oriented to HPC systems
- Native performance of custom HPC hardware
- Compatibility with Docker
- Image building (not available on Piz Daint)

```
module load daint-gpu # or daint-mc
module load singularity/3.2.1-daint
```

Singularity



Singularity: Pulling Images from Online Registries

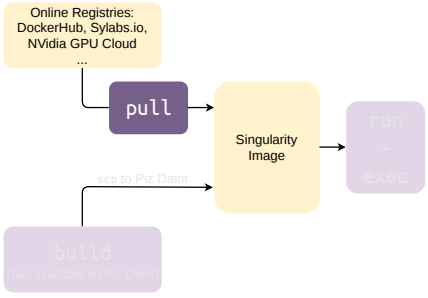



```
# From Dockerhub
srun singularity pull docker://ubuntu:bionic-20190612
```

```
# From Nvidia GPU Cloud
srun singularity pull docker://nvcr.io/nvidia/tensorflow:19.08-py3
```

- `docker://` specifies the type of image.
- This produces the singularity images `ubuntu_bionic-20190612.sif` and `tensorflow_19.08-py3.sif`.
- There is no registry. Images are saved by default on the directoy where the `singularity pull` was run.
- Same as with Shifter. It's recommended to run `shifter pull` on the compute nodes through Slurm.

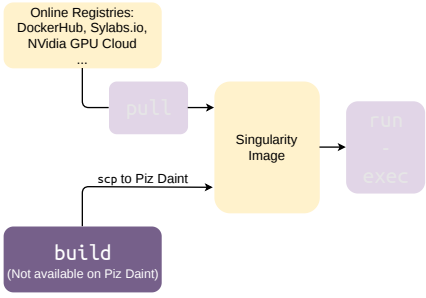
Singularity: Pulling Images from Online Registries



https://cloud.sylabs.io/library

```
srun singularity pull library://godlovedc/funny/lolcow
```

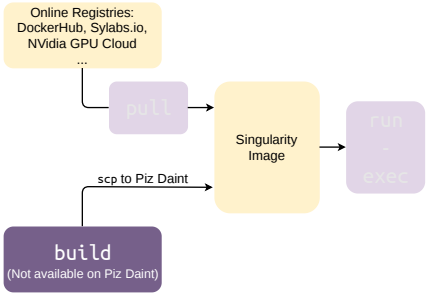
Singularity: Building Images




```
# Definition file `cuda_samples.def`
Bootstrap: docker
From: nvidia/cuda:9.2-devel

%post
    apt-get update
    apt-get install -y git
    git clone https://github.com/NVIDIA/cuda-samples.git /usr/local/cuda_samples
    cd /usr/local/cuda_samples
    git fetch origin --tags
    git checkout v9.2
    make

%runscript
    /usr/local/cuda_samples/Samples/deviceQuery/deviceQuery
```

```
# Build the image
sudo singularity build cuda_samples.sif cuda_samples.def
```

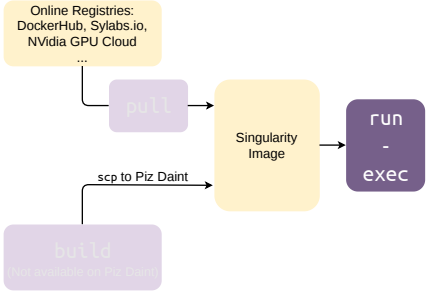
Singularity: Building Images




```
# Definition file `mpi_osu.def`
Bootstrap: docker
From: ubuntu:bionic-20190612

%post
    apt-get update
    apt-get install -y file g++ gcc gfortran make gdb strace realpath \
            wget --no-install-recommends

    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

    # ... download and compile the OSU micro benchmarks ...
    # see the complete file at
    # https://user.cscs.ch/tools/containers/#running-an-mpi-enabled-container-

%runscript
    /usr/local/libexec/osu-micro-benchmarks/mpi/pt2pt/osu_bw
```

```
# Build the image
sudo singularity build mpi_osu.sif mpi_osu.def
```

Singularity: Running Containers

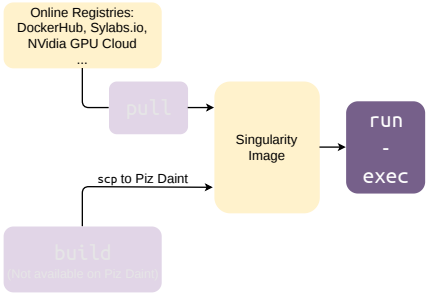



```
# run command
srun -C gpu singularity exec --nv cuda_samples.sif /usr/local/<...>/deviceQuery
```

```
# run command
srun -C gpu singularity exec mpi_osu.sif /usr/local/<...>/osu_bw
```

- `--nv` needs to be passed to enable Nvidia support.
- Loading the module `singularity/3.2.1-daint` sets up everything needed to run GPU and MPI-enabled containers.

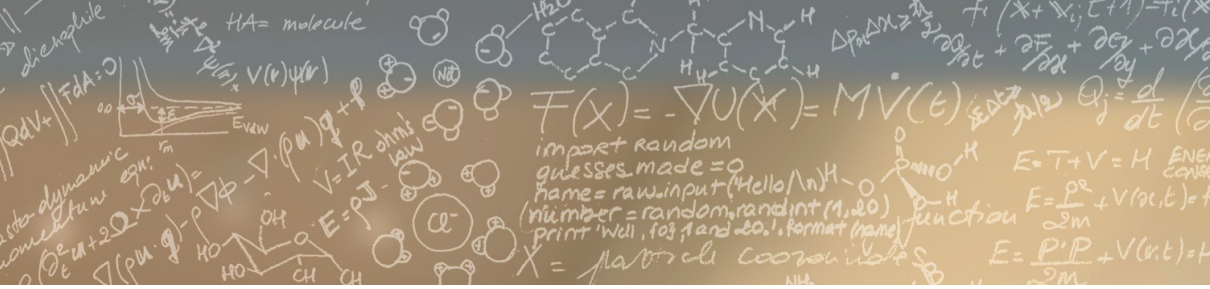
Singularity: Running Containers




```
# execute image's runscript
srun -C gpu singularity run --nv cuda_samples.sif
```

```
# execute image's runscript
srun -C gpu singularity exec mpi_osu.sif
```

```
# execute image's runscript
~> srun singularity run lolcow_latest.sif
#  ______________________________________
# / You have been selected for a secret \
# \ mission!                            /
#  --------------------------------------
#         \   ^__^
#          \  (oo)\_______
#             (__)\       )\/\
#                 | |----w |
#                 | |    | |
```



Thank you for your attention!