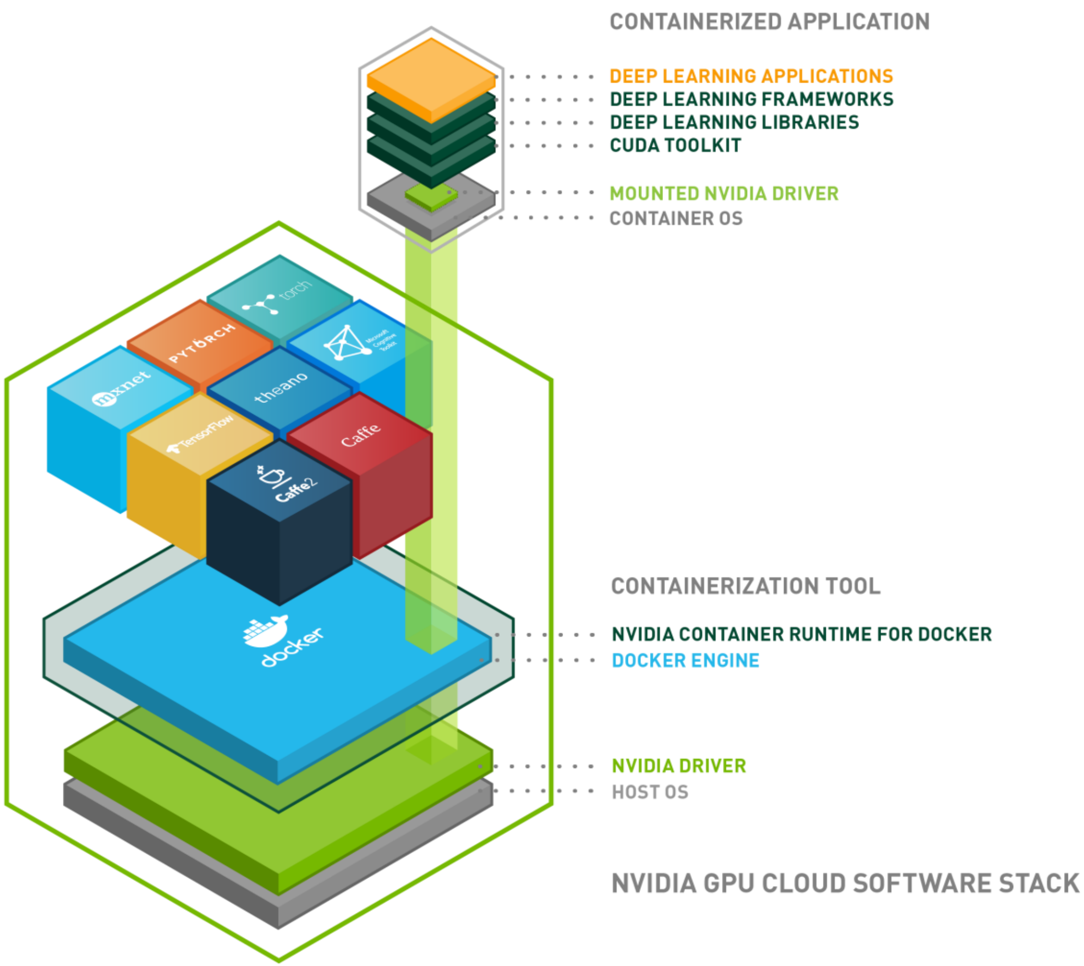
NVIDIA GPU CLOUD

Vishal Mehta



WHY CONTAINERS?

Benefits of Containers:

Simplify deployment of GPU-accelerated software, eliminating time-consuming software integration work

Isolate individual deep learning frameworks and applications

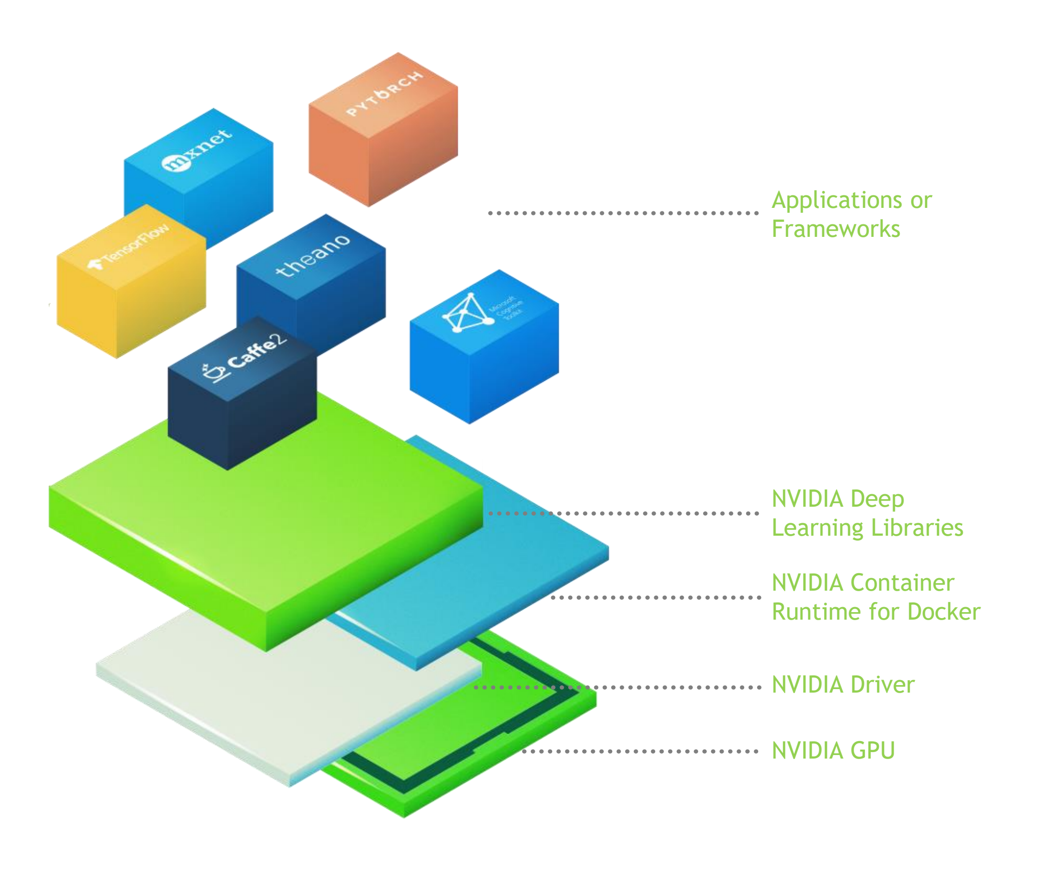
Share, collaborate, and test applications across different environments

CHALLENGES WITH COMPLEX SOFTWARE

Current DIY GPU-accelerated AI and HPC deployments are **complex** and **time consuming** to build, test and maintain

Development of software frameworks by the community is moving **very fast**

Requires high level of **expertise** to manage driver, library, framework dependencies



NVIDIA GPU CLOUD

Simple access to a comprehensive catalog of GPU-accelerated software

Discover 30 GPU-Accelerated Containers

Deep learning, third-party managed HPC applications, NVIDIA HPC visualization tools, and partner applications

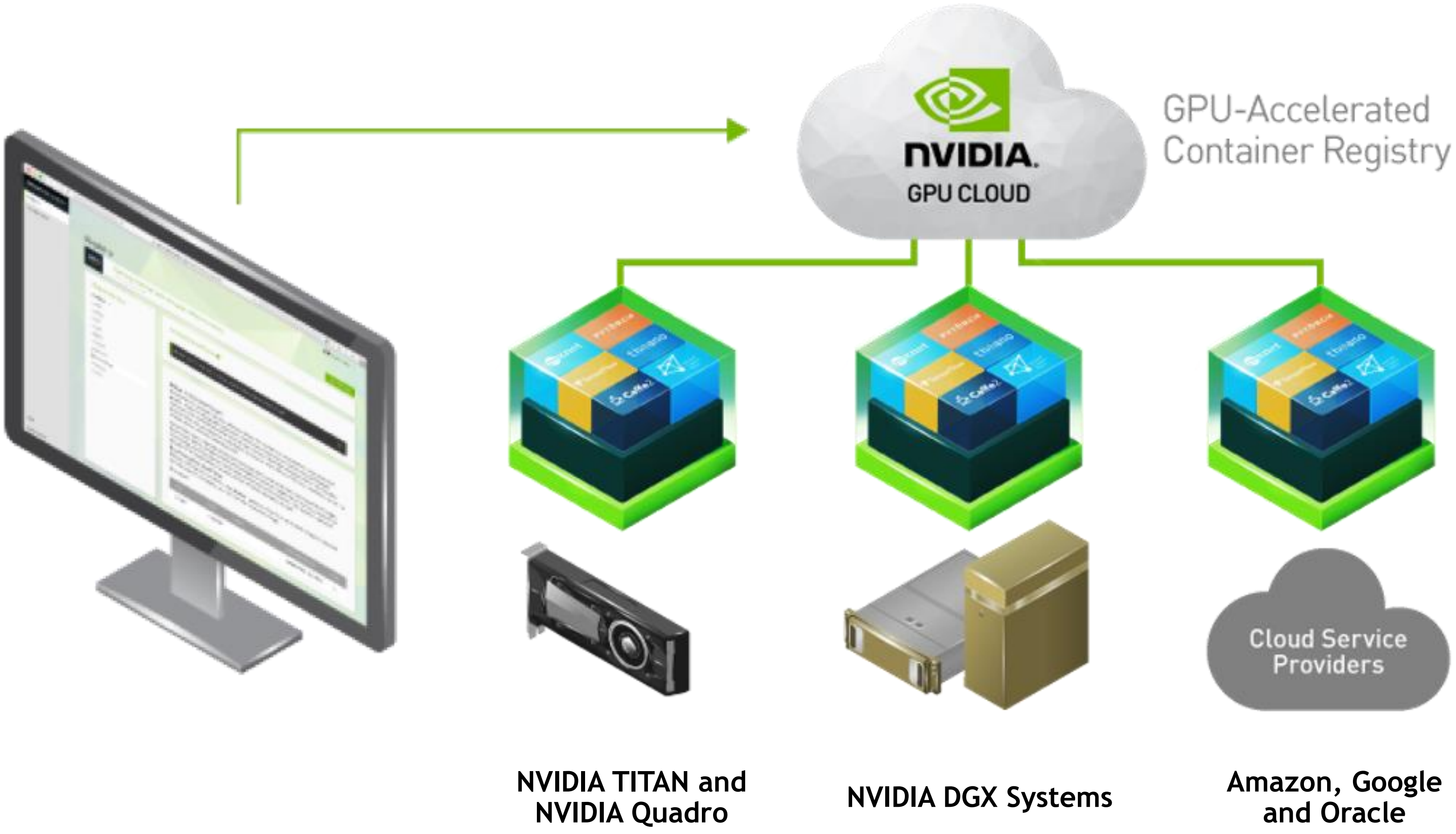
Innovate in Minutes, Not Weeks

Get up and running quickly and reduce complexity

Access from Anywhere

Use on PCs with NVIDIA Volta or Pascal™ architecture GPUs, NVIDIA DGX Systems, and supported cloud providers

FROM DESKTOP, TO DATA CENTER, TO CLOUD



NGC GPU-OPTIMIZED DEEP LEARNING CONTAINERS

A comprehensive catalog of deep learning software

- NVCaffe
- Caffe2
- Microsoft Cognitive Toolkit (CNTK)
- DIGITS
- MXNet
- PyTorch
- TensorFlow
- Theano
- Torch
- CUDA (base level container for developers)
- NVIDIA TensorRT inference accelerator with ONNX support

NVIDIA GPU CLOUD

ALWAYS UP-TO-DATE

Monthly updates from NVIDIA to deep learning containers

Containerized Applications

NVIDIA Container Runtime for Docker	NVIDIA Container Runtime for Docker	NVIDIA Container Runtime for Docker	NVIDIA Container Runtime for Docker		NVIDIA Container Runtime for Docker
TensorFlow	Microsoft Cognitive Toolkit	Caffe2	PYTORCH	• • •	Other Frameworks and Apps
TF Tuned SW	CNTK Tuned SW	Caffe2 Tuned SW	PyTorch Tuned SW		Tuned SW
CUDA RT	CUDA RT	CUDA RT	CUDA RT		CUDA RT

Linux Kernel + CUDA Driver

END-TO-END PRODUCT FAMILY

TRAINING

DESKTOP

NVIDIA GPU CLOUD

GPU-Accelerated Container Registry

TITAN V

DGX Station

DATA CENTER

NVIDIA GPU CLOUD

GPU-Accelerated Container Registry

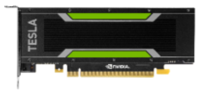
DGX-1 Server

TESLA V100

INFERENCE

DATA CENTER

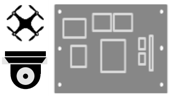
NVIDIA GPU CLOUD



TESLA P4

TESLA V100

EMBEDDED



JETPACK SDK

Jetson

AUTOMOTIVE

DriveWorks SDK

Drive PX

GET STARTED TODAY WITH NGC

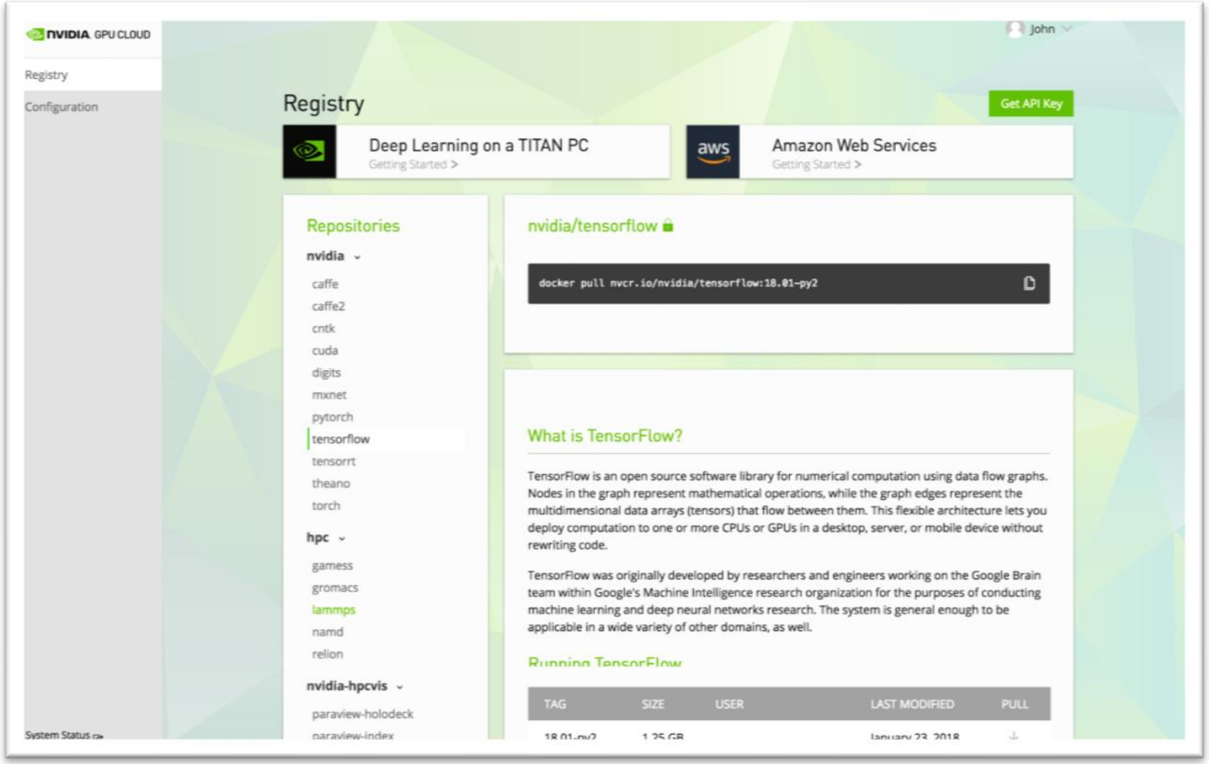
Sign up for no cost access

To learn more about all of the GPU-accelerated software on NVIDIA GPU Cloud, visit:

nvidia.com/cloud

To sign up, go to:

nvidia.com/ngcsignup



NGC CONTAINERS ON CSCS

In just 1,2,3……

CSCS uses shifter to manage containers. Shifters essentially does not allow root access and maintains user privileges.

Run your first NGC Tensorflow container in 3 simple steps

1) shifter pull --login nvcr.io/nvidia/tensorflow:18.03-py3

2) Username: $oauthtoken Password:<API_KEY>

3) srun -N1 -C gpu --pty shifter run --mount=type=bind,source=$HOME,destination=$HOME nvcr.io/nvidia/tensorflow:18.03-py3 bash

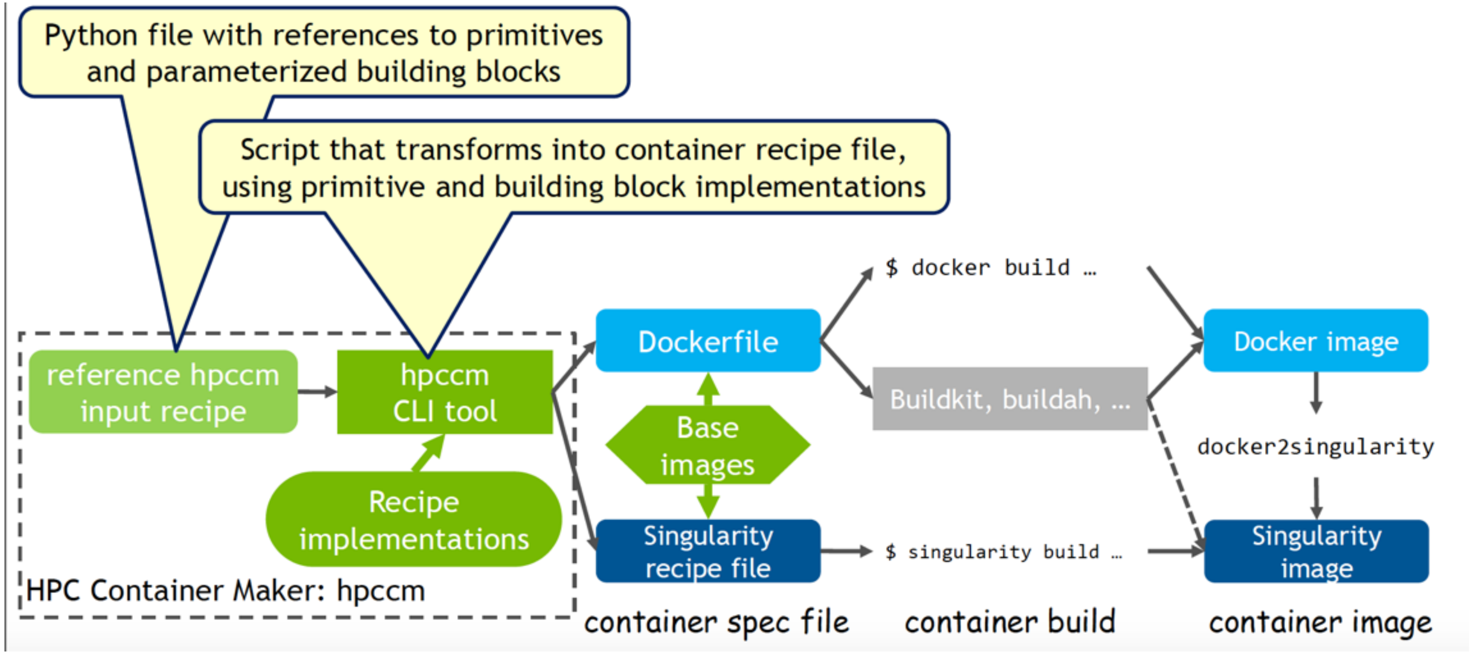
HPC CONTAINER MAKER

Vishal Mehta

HPC CONTAINER MAKER - HPCCM

"h-p-see-um"

- HPC Container Maker (HPCCM) generates container specification files (Dockerfiles or Singularity recipe) based on recipes
- A recipe specifies the series of steps to be performed when building a container



https://github.com/NVIDIA/hpc-container-maker

HPC CONTAINER MAKER - HPCCM

- Container implementation abstraction
  - The same recipe file generates specification files for Docker or Singularity
- Availability of full programming languages
  - A recipe is Python code. This means that you can use the full power of Python in a recipe for conditional branching, input validation, searching the web for the latest version of a component, etc.
- Higher level abstraction
  - Provides building blocks to simplify recipes and encapsulate best practices
- Container Maker generates human readable Dockerfiles and Singularity recipe files

FEATURES

BASE IMAGE & PACKAGES

- Ubuntu 16.04
- Cent OS

COMPILER

- PGI COMMUNITY EDITION
- GNU
- NVCC (CUDA TOOLKIT)

SOFTWARE - MORE ON REQUEST

- CMAKE
- HDF5
- NETCDF
- MLNX OFED
- OpenMPI
- MVAPICH2
- Python
- FFTW
- MKL

FULL SCALE APPLICATIONS

- MILC
- GROMACS

```
# Choose a base image

Stage0.baseimage('ubuntu:16.04')

# Install GNU Compilers

Stage0 += apt_get(ospackages=['gcc', 'g++', 'gfortran'])
```

```
hpccm.py --recipe recipes/basic.py --format docker
```

```
FROM ubuntu:16.04

RUN apt-get update -y && \
apt-get install -y --no-install-recommends \
gcc \
g++ \
gfortran && \
rm -rf /var/lib/apt/lists/*
```

```
hpccm.py --recipe recipes/basic.py --format singularity
```

```
BootStrap: docker

From: ubuntu:16.04
%post
apt-get update -y
apt-get install -y --no-install-recommends \
gcc \
g++ \
gfortran
rm -rf /var/lib/apt/lists/*
```

AVAILABILITY OF A FULL PROGRAMMING LANGUAGE

- Full power of Python in a recipe for conditional branching, input validation, searching the web for the latest version of a component, etc.

For example, the LAMMPS application may be built in single, double, or mixed precision mode. (hpccm.py --userarg LAMMPS_PRECISION=...)

```
# get and validate precision
VALID_PRECISION = ['single', 'double', 'mixed']
precision = USERARG.get('LAMMPS_PRECISION', 'single')
if precision not in VALID_PRECISION:
    raise ValueError('Invalid precision')
...

Stage0 += shell(commands=['make -f Makefile.linux.{}'.format(precision), ...])

...
```

HIGHER LEVEL ABSTRACTION

Building blocks to simplify recipes and encapsulate best practices

```
Stage0 += openmpi(cuda=True, infiniband=True,
                  prefix='/usr/local/openmpi', version='3.0.0')
```

```
# OpenMPI version 3.0.0 RUN
apt-get update -y && \
apt-get install -y --no-install-recommends \
   file \
   hwloc \
   openssh-client \
   wget && \
rm -rf /var/lib/apt/lists/*

RUN mkdir -p /tmp && wget -q --no-check-certificate -P /tmp https://www.open-
mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.bz2 && \
  tar -x -f /tmp/openmpi-3.0.0.tar.bz2 -C /tmp -j && \
  cd /tmp/openmpi-3.0.0 && ./configure --prefix=/usr/local/openmpi --disable-
getpwuid --enable-orterun-prefix-by-default --with-cuda --with-verbs && \
   make -j4 && \
   make -j4 install && \
   rm -rf /tmp/openmpi-3.0.0.tar.bz2 /tmp/openmpi-3.0.0

ENV PATH=/usr/local/openmpi/bin:$PATH \
LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PATH
```

```
# OpenMPI version 3.0.0
%post
   apt-get update -y
   apt-get install -y --no-install-recommends \
      file \
      hwloc \
      openssh-client \
      wget
   rm -rf /var/lib/apt/lists/*

%post
   mkdir -p /tmp && wget -q --no-check-certificate -P /tmp https://www.open-
mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.bz2
  tar -x -f /tmp/openmpi-3.0.0.tar.bz2 -C /tmp -j
  cd /tmp/openmpi-3.0.0 && ./configure --prefix=/usr/local/openmpi --disable-getpwuid
--enable-orterun-prefix-by-default --with-cuda --with-verbs
  make -j4
  make -j4 install
  rm -rf /tmp/openmpi-3.0.0.tar.bz2 /tmp/openmpi-3.0.0

%environment
  export PATH=/usr/local/openmpi/bin:$PATH
  export LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PATH
```

TOOLCHAIN SUPPORT

Using toolchain for building blocks

```
GNU = gnu()
Stage0 += GNU
TC     = GNU.toolchain
TC.CUDA_HOME = /usr/local/cuda

OMPI = openmpi(version='3.0.0',toolchain=TC)
Stage0 += OMPI

FFTW = fftw(version='3.3.7',toolchain=TC)
Stage0 += FFTW

HDF5 = hdf5(version='1.10.1',toolchain=TC)
Stage0 += HDF5

NETCDF =netcdf(version='4.6.1',toolchain=TC)
Stage0 += NETCDF
```

USING HPCCM FOR CSCS

ON LOCAL LINUX BOX:

1) Git clone https://github.com/NVIDIA/hpc-container-maker
2) ./hpccm.py --format docker --recipe hpcbase-gnu-mvapich2.py > Dockerfile
3) Docker build –t container - < Dockerfile
4) Docker save container –o container.tar.gz

ON CSCS: (https://user.cscs.ch/user_services/containers/advanced_shifter/)

1) Srun –N 1 –C gpu shifter load container.tar.gz container
2) srun -N1 -C gpu --pty shifter run --mount=type=bind,source=$HOME,destination=$HOME load/library/container:latest bash

NVIDIA