

ALCF COMPUTATIONAL PERFORMANCE WORKSHOP

MAY 4-6, 2021



# CONTAINERS ON THETA AND THETA-GPU AN INTRODUCTION

**Romit Maulik, Taylor Childers, Corey Adams**

**Datascience team**

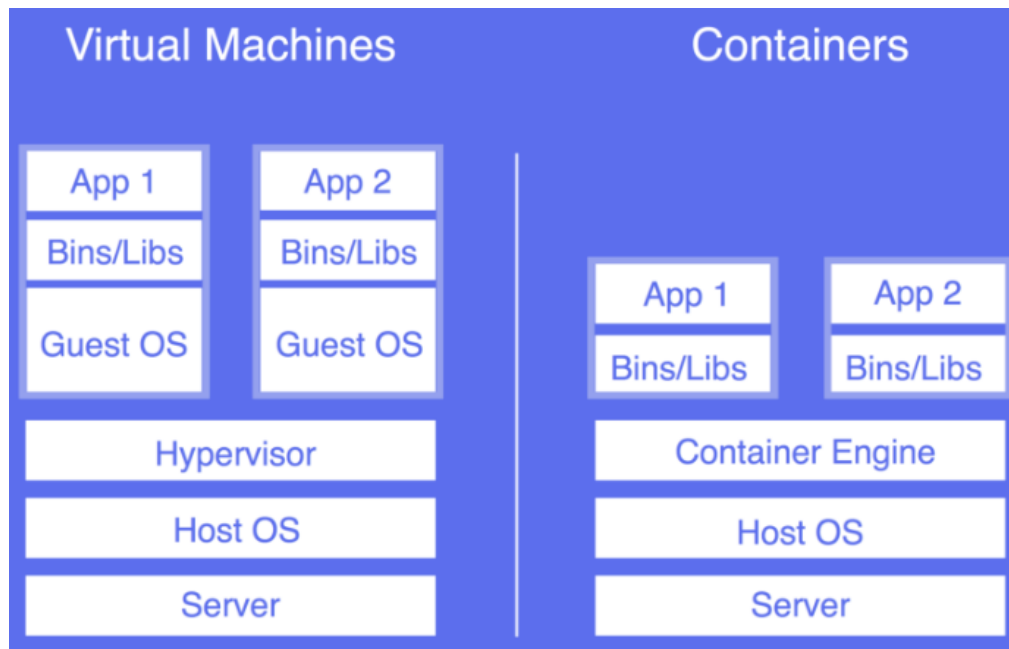
**Argonne Leadership Computing Facility**

# Outline

- Using singularity containers on Theta and Theta-GPU
- Using Docker-hub to build containers and build them using Singularity on Theta (for running MPI apps).
- Using pre-existing singularity images on Theta-GPU for deep learning workloads.
- When in doubt – message us on the CPW portal.

# Containers

- Lightweight alternatives to virtual machines and increasingly popular for cloud applications. Examples: Docker, Singularity, Rocket etc.



Singularity is our container of choice for security reasons.

However, Docker is the most widely used container platform.

# Dockerfiles and Docker-hub

- Option 1: Build a docker image on their local machine and upload to docker-hub (needs local installation of docker).
- Option 2: Link Github and docker-hub and auto-build using Dockerfile. (does not need local installation but useful for testing anyway).
- We'll show you how docker and docker-hub can be used to port your docker containers to Theta and build Singularity images.
- Due to time constraints – we cannot show you how to install docker on your local machine (but this is not very difficult):

Visit: <https://docs.docker.com/engine/install/> and choose your OS.

Our results/demos today from docker version 20.10.6



source



Dockerfile



submit.sh

# An example Dockerfile

```
FROM centos:latest
MAINTAINER Romit rmaulik@anl.gov

WORKDIR /mpich
COPY source/pi.c .

RUN yum update -y
RUN yum groupinstall -y "Development Tools"
RUN yum install -y gcc-c++ wget gcc-gfortran

ENV MPICH_VERSION 3.3
RUN wget http://www.mpich.org/static/downloads/$MPICH_VERSION/mpich-$MPICH_VERSION.tar.gz
RUN tar xf mpich-$MPICH_VERSION.tar.gz --strip-components=1

# disable the addition of the RPATH to compiled executables
# this allows us to override the MPI libraries to use those
# found via LD_LIBRARY_PATH
RUN ./configure --prefix=/mpich/install --disable-wrapper-rpath
RUN make -j 4 install

# add to local environment to build pi.c
ENV PATH $PATH:/mpich/install/bin
ENV LD_LIBRARY_PATH $LD_LIBRARY_PATH:/mpich/install/lib
RUN env | sort
RUN mpicc -o pi -fPIC pi.c

COPY submit.sh .

RUN chmod +x submit.sh

ENTRYPOINT ["/mpich/submit.sh"]
```

Container OS  
pulled from docker-hub  
repository

Working directory in  
container – copy source

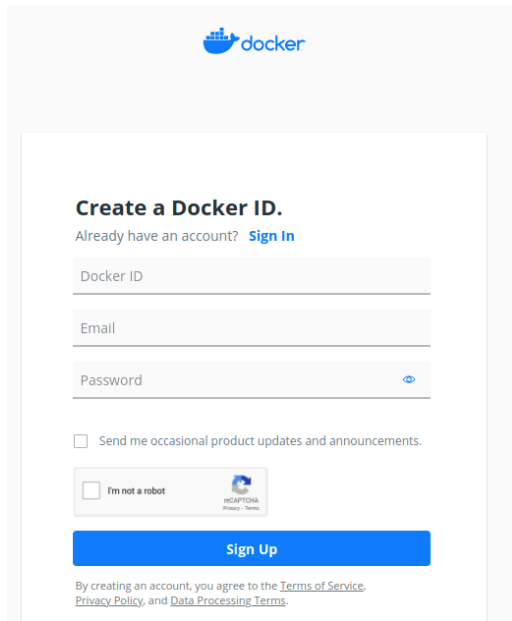
Install appropriate  
packages for building  
app

Build your app

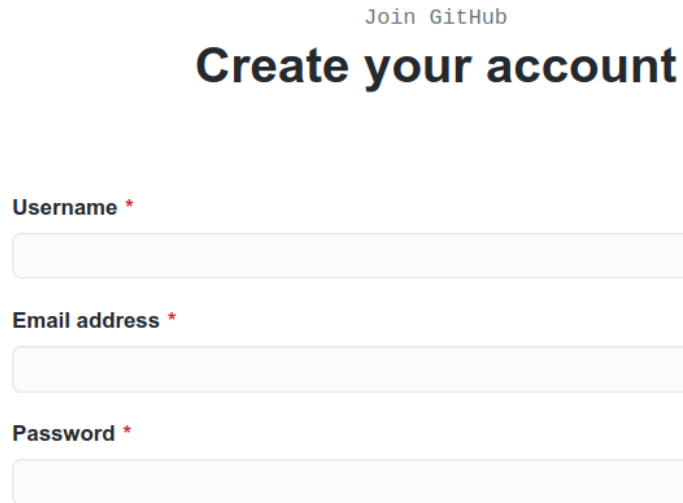
Specify script to run and  
copy from local

# Linking Github and Docker-hub

- Make sure you have both Github and Docker-hub accounts (sign up with email).



The screenshot shows the Docker website's sign-up page. At the top is the Docker logo. Below it, the heading 'Create a Docker ID.' is followed by the text 'Already have an account? [Sign In](#)'. There are three input fields: 'Docker ID', 'Email', and 'Password' (with an eye icon for toggling visibility). Below the fields is a checkbox labeled 'Send me occasional product updates and announcements.' and a reCAPTCHA widget with the text 'I'm not a robot'. A blue 'Sign Up' button is at the bottom. At the very bottom, a small line of text states: 'By creating an account, you agree to the [Terms of Service](#), [Privacy Policy](#), and [Data Processing Terms](#).'

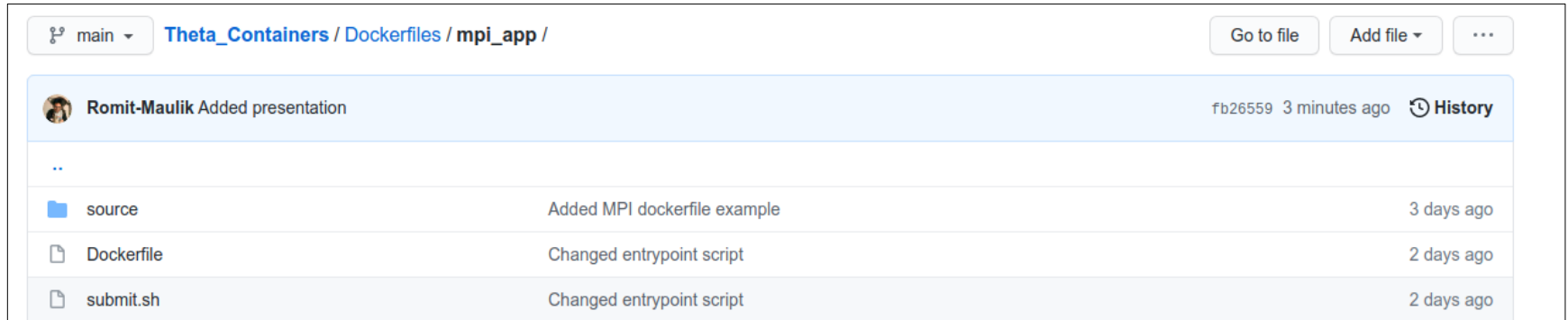


The screenshot shows the GitHub website's sign-up page. At the top right is the text 'Join GitHub'. The main heading is 'Create your account'. There are three input fields: 'Username \*', 'Email address \*', and 'Password \*'. The 'Password' field has a small icon indicating password strength requirements.

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

# Linking Github and Docker-hub

- Make sure you have both Github and Docker-hub accounts (sign up with email).
- Push your Dockerfile to a github repository



Example code at  
[https://github.com/Romit-Maulik/Theta\\_Containers/](https://github.com/Romit-Maulik/Theta_Containers/)

# Linking Github and Docker-hub

- Link that repository with a Docker-hub repository


Create Repository


romitmaulik1 | alctutorial1

Using docker/singularity to run containerized apps on ALCF systems

Visibility



Using 0 of 1 private repositories. [Get more](#)

☒ Public  Public repositories appear in Docker Hub search results

☐ Private  Only you can view private repositories

Build Settings (optional)



Autobuild triggers a new build with every git push to your source code repository. [Learn More](#)





 Connected  Disconnected

[Cancel](#) [Create](#) [Create & Build](#)

Build Settings (optional)

Autobuild triggers a new build with every git push to your source code repository. [Learn More](#)

 Connected  Disconnected

 Romit-Maulik   Theta\_Containers 

BUILD RULES +

The build rules below specify how to build your source into Docker images.

Source Type	Source	Docker Tag	Dockerfile Location	Build Caching
Branch	main	hello_world	Dockerfiles/hello_wor	<input checked="" type="checkbox"/>

View example build rules

[Cancel](#) [Create](#) [Create & Build](#)

Master branch is now "main"

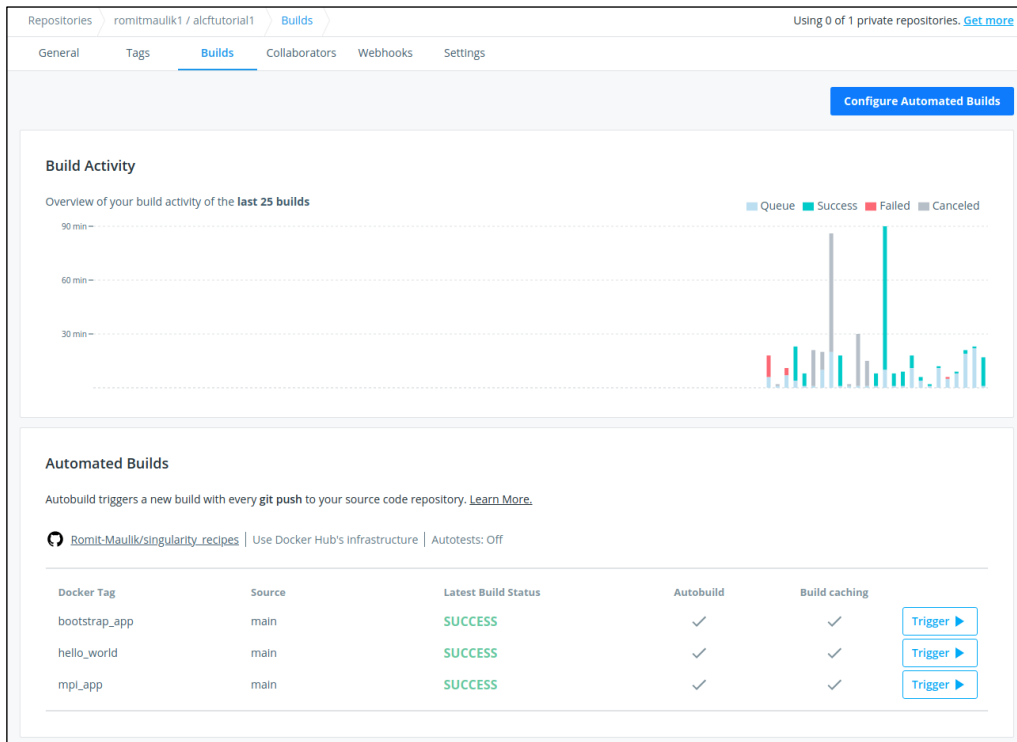
This will be disconnected the first time

Specify branch, image name (tag), path to docker file in repo  
For example:  
"Dockerfiles/hello\_world/Dockerfile"



# Linking Github and Docker-hub

- Building on docker-hub is slow (when compared to your local machine) but once complete you should see something like this



Using “Configure Automated Builds” to edit build names (tags), paths etc.

## Recent Builds



[Build in 'main:/Dockerfiles/bootstrap' \(e75e01da\)](#)

Clicking on a recent build will give you logs (useful for debugging fails) and the docker file itself.

# Linking Github and Docker-hub

- Building on docker-hub is slow (when compared to your local machine) but once complete you should see something like this

```
BUILD LOGS DOCKERFILE README

Cloning into '.'...
Warning: Permanently added the RSA host key for IP address '140.82.114.4' to the list of known hosts.
Reset branch 'main'
Your branch is up-to-date with 'origin/main'.
KernelVersion: 4.4.0-1060-aws
Components: [[{"Version": "u'19.03.8'", "Name": "u'Engine'", "Details": {"KernelVersion": "u'4.4.0-1060-aws'", "Os": "u'linux'", "BuildTime": "u'2020-03-11T01:24:30.000000000+00:00'", "Arch": "amd64", "BuildTime": "2020-03-11T01:24:30.000000000+00:00", "ApiVersion": "1.40", "Platform": {"Name": "u'Docker Engine - Community'"
Version: 19.03.8
MinAPIVersion: 1.12
GitCommit: afacbb7f0
Os: linux
GoVersion: go1.12.17
Starting build of index.docker.io/ronitmaulik1/alcftutorial1:bootstrap_app...
Step 1/11 : FROM ronitmaulik1/alcftutorial1:npi_app
----> 1d71b9dc4737
Step 2/11 : MAINTAINER Ronit rmaulik@anl.gov
----> Running in 5853cffe9863
Removing intermediate container 5853cffe9863
----> 2c33179ea695
Step 3/11 : WORKDIR /python_app
----> Running in 2c82a6974851
Removing intermediate container 2c82a6974851
----> b563dd74d077
```

Using “Configure Automated Builds” to edit build names (tags), paths etc.

## Recent Builds



[Build in 'main:/Dockerfiles/bootstrap' \(e75e01da\)](#)

Clicking on a recent build will give you logs (useful for debugging fails) and the docker file itself.

# Linking Github and Docker-hub

- Building on docker-hub is slow (when compared to your local machine) but once complete you should see something like this

```

BUILD LOGS  DOCKERFILE  README

FROM centos:latest
MAINTAINER Romit rmaulik@anl.gov

WORKDIR /mpich
COPY source/pi.c .

RUN yum update -y
RUN yum groupinstall -y "Development Tools"
RUN yum install -y gcc-c++ wget gcc-gfortran

ENV MPICH_VERSION 3.3
RUN wget http://www.mpich.org/static/downloads/$MPICH_VERSION/mpich-$MPICH_VERSION.tar.gz
RUN tar xf mpich-$MPICH_VERSION.tar.gz --strip-components=1

# disable the addition of the RPATH to compiled executables
# this allows us to override the MPI libraries to use those
# found via LD_LIBRARY_PATH
RUN ./configure --prefix=/mpich/install --disable-wrapper-rpath
RUN make -j 4 install

# add to local environment to build pi.c
ENV PATH $PATH:/mpich/install/bin
ENV LD_LIBRARY_PATH $LD_LIBRARY_PATH:/mpich/install/lib
RUN env | sort
RUN mpicc -o pi -fPIC pi.c

COPY submit.sh .

RUN chmod +x submit.sh

ENTRYPOINT ["/mpich/submit.sh"]

```

Using “Configure Automated Builds” to edit build names (tags), paths etc.

## Recent Builds



[Build in 'main:/Dockerfiles/bootstrap' \(e75e01da\)](#)

Clicking on a recent build will give you logs (useful for debugging fails) and the docker file itself.

# Pulling image to Theta

- Once build is successful on Docker-hub – pull to Theta using singularity (version 3.6.4-1)

- Command:

```
singularity build mpi_app.img docker://romitmaulik1/alcftutorial1:mpi_app
```

- Explore the contents of your image with:

```
singularity shell mpi_app.img
```

- You can find your source code, downloaded libraries, submit scripts in

```
ls /mpich/
```

- Your entrypoint script will look like

```
1  #!/bin/bash
2  # submit.sh
3
4  echo "running MPI app"
5  cd /mpich/
6  ./pi
7  |
```

# Interface MPI apps with Theta

- We need to swap out binaries with Cray in the submission script

```

2  #!/bin/bash
3  #COBALT -t 30
4  #COBALT -q debug-cache-quad
5  #COBALT -n 2
6
7  RANKS_PER_NODE=4
8
9  # pass container as first argument to script
10 CONTAINER=$1
11
12 # Use Cray's Application Binary Independent MPI build
13 module swap cray-mpich cray-mpich-abi
14
15 # Output current modules being used (for debugging)
16 module list
17
18 # Only needed when interactive debugging
19 #module swap PrgEnv-intel PrgEnv-cray; module swap PrgEnv-cray PrgEnv-intel
20
21 export ADDITIONAL_PATHS="/opt/cray/diag/lib:/opt/cray/ugni/default/lib64:/opt/cray/udreg/default/lib64:/opt
22
23 # in order to pass environment variables to a Singularity container create the
24 # variable with the SINGULARITYENV prefix
25 export SINGULARITYENV_LD_LIBRARY_PATH="$CRAY_LD_LIBRARY_PATH:$LD_LIBRARY_PATH:$ADDITIONAL_PATHS"
26
27 # print to log file for debug
28 echo $SINGULARITYENV_LD_LIBRARY_PATH
29
30 TOTAL_RANKS=$(( $COBALT_JOBSIZE * $RANKS_PER_NODE ))
31
32 # this simply runs the command 'ldd /myapp/pi' inside the container and should
33 # show that the app is running againsts the host machines Cray libmpi.so not the
34 # one inside the container
35 BINDINGS="-B /opt -B /etc/alternatives"
36 aprun -n 1 -N 1 singularity exec $BINDINGS $CONTAINER bash -c "echo \$LD_LIBRARY_PATH"
37 aprun -n 1 -N 1 singularity exec $BINDINGS $CONTAINER bash -c "ldd /myapp/pi"
38
39 # run my container like an application, which will run '/myapp/pi'
40 aprun -n $TOTAL_RANKS -N $RANKS_PER_NODE singularity run $BINDINGS $CONTAINER

```

Use cray-mpich-abi for compatibility with your build (remember to check compatibility of versions of mpich and cray-mpich)

Pass paths to CRAY objects to your container using SINGULARITYENV\_LD\_LIBRARY\_PATH

Run using aprun

# What is finally observed

```

linux-vdso.so.1 (0x00002aaaaad7000)
libmpi.so.12 => /opt/cray/pe/mpt/7.7.14/gni/mpich-intel-abi/16.0/lib/libmpi.so.12 (0x00002aaaaacd6000)
libc.so.6 => /lib64/libc.so.6 (0x00002aaaaab299000)
libxpmem.so.0 => /opt/cray/xpmem/default/lib64/libxpmem.so.0 (0x00002aaaab65c000)
librt.so.1 => /lib64/librt.so.1 (0x00002aaaab85f000)
libugni.so.0 => /opt/cray/ugni/default/lib64/libugni.so.0 (0x00002aaaaba67000)
libudreg.so.0 => /opt/cray/udreg/default/lib64/libudreg.so.0 (0x00002aaaabceb000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00002aaaabef5000)
libpmi.so.0 => /opt/cray/pe/pmi/5.0.16/lib64/libpmi.so.0 (0x00002aaaac115000)
libifport.so.5 => /opt/intel/compilers_and_libraries_2020.0.166/linux/compiler/lib/intel64/libifport.so.5 (0x00002aaaac35e000)
libifcore.so.5 => /opt/intel/compilers_and_libraries_2020.0.166/linux/compiler/lib/intel64/libifcore.so.5 (0x00002aaaac58c000)
libimf.so => /opt/intel/compilers_and_libraries_2020.0.166/linux/compiler/lib/intel64/libimf.so (0x00002aaaac8f0000)
libsvml.so => /opt/intel/compilers_and_libraries_2020.0.166/linux/compiler/lib/intel64/libsvml.so (0x00002aaaace8e000)
libm.so.6 => /lib64/libm.so.6 (0x00002aaaae815000)
libintlc.so.5 => /opt/intel/compilers_and_libraries_2020.0.166/linux/compiler/lib/intel64/libintlc.so.5 (0x00002aaaaeb97000)
/lib64/ld-linux-x86-64.so.2 (0x00002aaaaaab0000)
libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00002aaaaee0e000)
libdl.so.2 => /lib64/libdl.so.2 (0x00002aaaaf026000)
Application 22973002 resources: utime ~1s, stime ~2s, Rss ~37072, inblocks ~18068, outblocks ~0
running MPI app
running MPI app
running MPI app
running MPI app
running MPI app
running MPI app
running MPI app
running MPI app
worker 3 of 8
worker 2 of 8
worker 1 of 8
worker 0 of 8
worker 6 of 8
worker 7 of 8
worker 4 of 8
worker 5 of 8
pi is approximately 3.1417259869152532, Error is 0.0001333333254601
Application 22973003 resources: utime ~10s, stime ~8s, Rss ~36056, inblocks ~41708, outblocks ~0

```

# Bootstrapping

- When possible reuse previous builds in opensource repositories

```
1 # Bootstrap our previous image
2 FROM romitmaulik1/alcftutorial1:mpi_app
3 MAINTAINER Romit rmaulik@anl.gov
4
5 WORKDIR /python_app
6 COPY source/hello_world.py .|
7
8 RUN yum update -y && yum -y install epel-release
9 RUN yum -y install https://repo.ius.io/ius-release-el7.rpm https://dl.fedoraproject.org/pub/epel/
   epel-release-latest-7.noarch.rpm
10
11 RUN yum -y makecache && yum -y install python3 python3-pip python3-devel && yum clean all
12
13 RUN pip3 install numpy tensorflow matplotlib
14
15 COPY submit.sh .
16
17 RUN chmod +x submit.sh
18
19 ENTRYPOINT ["/python_app/submit.sh"]
20
```

We add a python application into this pre-existing image and build a new one

Install some common datascience packages

# Bootstrapping

- When possible reuse previous builds in opensource repositories

```
rmaulik@thetalogin5:~/singularity_tutorial/bootstrap> singularity run bootstrap_app.img
running bootstrap app
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing
, it will be understood as (type, (1,)) / '(1,)type'.
  np_qint8 = np.dtype [("qint8", np.int8, 1)]
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing
, it will be understood as (type, (1,)) / '(1,)type'.
  np_quint8 = np.dtype [("quint8", np.uint8, 1)]
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing
, it will be understood as (type, (1,)) / '(1,)type'.
  np_qint16 = np.dtype [("qint16", np.int16, 1)]
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:519: FutureWarning: Passing
, it will be understood as (type, (1,)) / '(1,)type'.
  np_quint16 = np.dtype [("quint16", np.uint16, 1)]
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:520: FutureWarning: Passing
, it will be understood as (type, (1,)) / '(1,)type'.
  np_qint32 = np.dtype [("qint32", np.int32, 1)]
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing
, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype [("resource", np.ubyte, 1)]
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning:
f numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_qint8 = np.dtype [("qint8", np.int8, 1)]
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning:
f numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_quint8 = np.dtype [("quint8", np.uint8, 1)]
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning:
f numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_qint16 = np.dtype [("qint16", np.int16, 1)]
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning:
f numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_quint16 = np.dtype [("quint16", np.uint16, 1)]
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning:
f numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_qint32 = np.dtype [("qint32", np.int32, 1)]
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning:
f numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype [("resource", np.ubyte, 1)]
Hello world
Numpy version: 1.19.5
tensorflow version: 1.19.5
```

```
1 import numpy as np
2 import tensorflow as tf
3
4 print('Hello world')
5 print('Numpy version:', np.__version__)
6 print('tensorflow version:', np.__version__)
```



# ThetaGPU

- There are several containers on ThetaGPU that will help you get started with deep learning experiments that can efficiently use the A100 GPUs.
- Documentation: <https://argonne-lcf.github.io/ThetaGPU-Docs/>
- The different optimized containers for DL are available at:  
</lus/theta-fs0/projects/datascience/thetaGPU/containers/>
- To leverage these containers – you must either perform the following in a shell script that acquires a compute node or do this interactively.
- Further details may be found in:  
[https://github.com/argonne-lcf/sdl\\_ai\\_workshop/tree/master/05\\_Simulation\\_ML/ThetaGPU](https://github.com/argonne-lcf/sdl_ai_workshop/tree/master/05_Simulation_ML/ThetaGPU)

# ThetaGPU

18

- For interactive/debug DL, grab an interactive node with:

```
qsub -n 1 -q training -A project_name -l -t 1:00:00
```

- Activate the singularity container:

```
singularity exec -B /lus:/lus --nv /lus/theta-fs0/software/thetagpu/nvidia-containers/tensorflow2/tf2_20.08-py3.simg bash
```

- If your application needs new packages, setup access to the internet:

```
export http_proxy=http://theta-proxy.tmi.alcf.anl.gov:3128
```

```
export https_proxy=https://theta-proxy.tmi.alcf.anl.gov:3128
```

- Create a virtual environment and install your packages for example:

```
python -m pip install --user virtualenv
```

```
export VENV_LOCATION=/home/$USER/THETAGPU_TF_ENV # Add your path here
```

```
python -m virtualenv --system-site-packages $VENV_LOCATION
```

```
source $VENV_LOCATION/bin/activate
```

```
pip install sklearn
```

# ThetaGPU

- Now that packages are installed – run your DL script like you would on your local machine or construct a script for submission to the queue:

```
#!/bin/bash
```

```
#COBALT -n 1
```

```
#COBALT -t 00:10:00
```

```
#COBALT -q training
```

```
#COBALT -A project_name
```

```
CONTAINER=/lus/theta-fs0/software/thetagpu/nvidia-containers/tensorflow2/tf2_20.08-py3.simg
```

```
SCRIPT=/path/to/project_location/queue_submission.sh
```

```
echo "Running Cobalt Job $COBALT_JOBID."
```

```
mpirun -n 1 -npnode 1 -hostfile $COBALT_NODEFILE singularity run --nv -B /lus:/lus  
$CONTAINER $SCRIPT
```

- Where `queue_submission.sh` is:

```
#!/bin/bash
```

```
cd /path/to/project_location
```

```
export VENV_LOCATION=/home/rmaulik/THETAGPU_TF_ENV
```

```
source $VENV_LOCATION/bin/activate
```

```
python myscript.py
```

- You may also use Docker-hub to build images on the fly on ThetaGPU (after exporting proxies) – although we recommend using what we just discussed to add your libraries to a virtual environment.

# Fin!

21

- I encourage you to try these tutorials and get back to us with any questions.

# Fin!

22

- I encourage you to try these tutorials and get back to us with any questions.