

CONTAINERS

João Paulo Navarro (jpnavarro@nvidia.com)
Solutions Architect



Agenda

- Intro
- Containers Overview
- Managing Images, Containers and Running Containers
- Installing Docker and NVIDIA Docker

Introduction

DIFFERENT ROLES. SAME GOALS.

Driving Productivity and Faster Time-to-Solutions

Data Scientists and
Researchers



Eliminate mundane tasks, focus on
science and research

Developers



Speed up development with
existing building blocks

Sysadmins



Deploy to production immediately

CHALLENGES UTILIZING AI & HPC SOFTWARE

EXPERTISE



Building AI-centric solutions requires expertise

INSTALLATION



Complex, time consuming, and error-prone

OPTIMIZATION



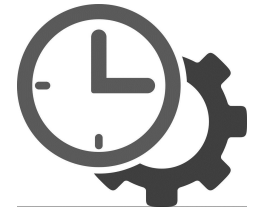
Requires expertise to optimize framework performance

PRODUCTIVITY



Users limited to older features and lower performance

MAINTAINENCE



IT can't keep up with frequent software upgrades

CONTAINERS - SIMPLIFYING AI & HPC WORKFLOWS

EMBEDDING EXPERTISE



Deliver greater value, faster

FASTER DEPLOYMENTS



Eliminates installations.
Simply Pull & Run the app

OPTIMIZED SOFTWARE



Key DL frameworks updated monthly for perf optimization

HIGHER PRODUCTIVITY



Better Insights and faster time-to-solution

ZERO MAINTENANCE



Empowers users to deploy the latest versions with IT support

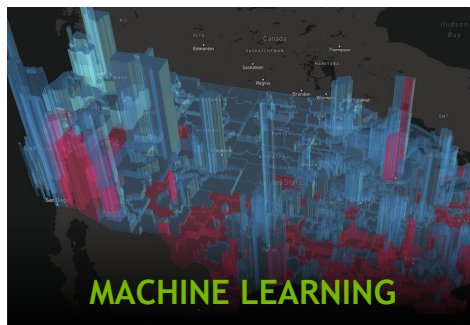
GPU-OPTIMIZED SOFTWARE CONTAINERS

Over 50 Containers on NGC - ngc.nvidia.com



DEEP LEARNING

TensorFlow | PyTorch | [more](#)



MACHINE LEARNING

RAPIDS | H2O | [more](#)



INFERENCE

TensorRT | DeepStream | [more](#)



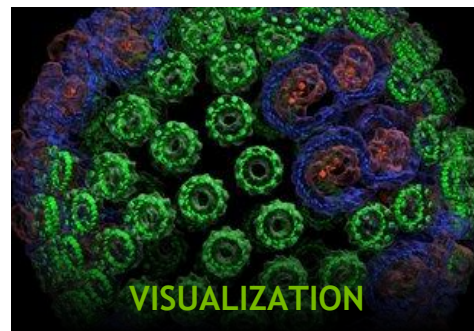
HPC

NAMD | GROMACS | [more](#)



GENOMICS

Parabricks
[s](#)



VISUALIZATION

ParaView | IndeX | [more](#)

NGC CONTAINERS: ACCELERATING WORKFLOWS

WHY CONTAINERS

Simplifies Deployments

- Eliminates complex, time-consuming builds and installs

Get started in minutes

- Simply Pull & Run the app

Portable

- Deploy across various environments, from test to production with minimal changes

WHY NGC CONTAINERS

Optimized for Performance

- Monthly DL container releases offer latest features and superior performance on NVIDIA GPUs

Scalable Performance

- Supports multi-GPU & multi-node systems for scale-up & scale-out environments

Designed for Enterprise & HPC environments

- Supports Docker & Singularity runtimes

Run Anywhere

- Pascal/Volta/Turing-powered NVIDIA DGX, PCs, workstations, and servers
- From Core to the Edge
- On-Prem to Hybrid to Cloud

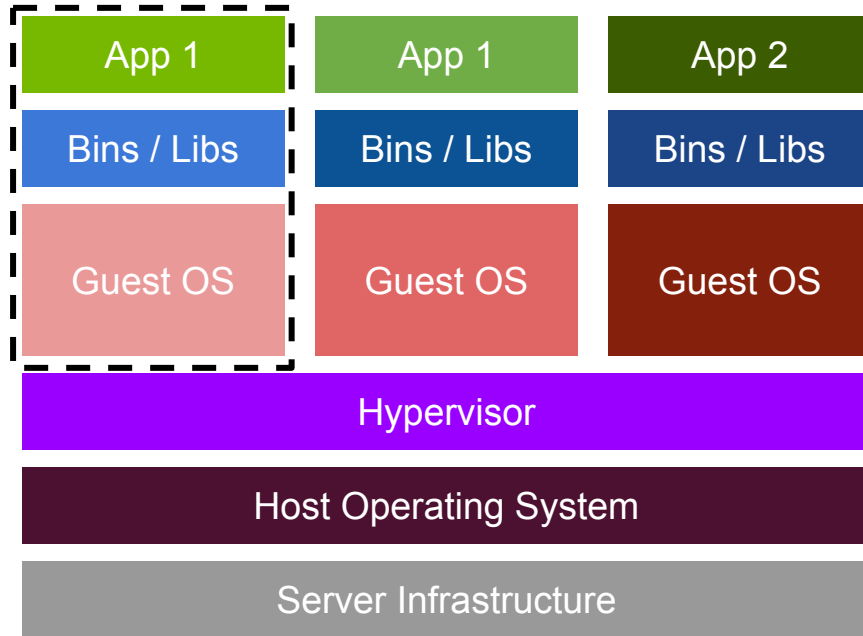
Container Overview

Containers

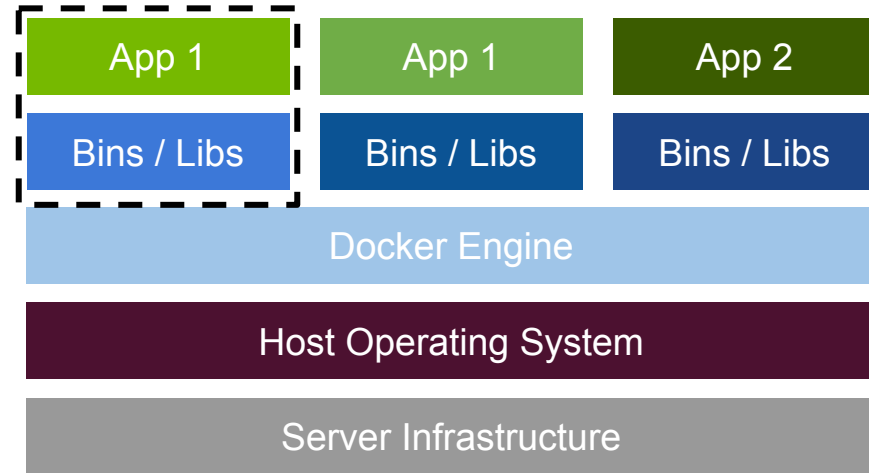
- Portable and reproducible builds
- Ease of deployment
- Run across heterogeneous CUDA toolkit environments (sharing the host driver)
- Bare Metal Performance
- Facilitate collaboration

Virtual Machine vs. Container

Not so similar

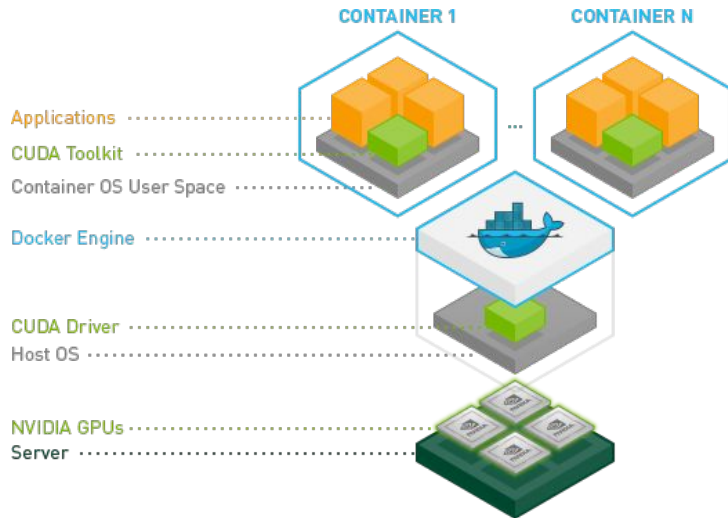


Virtual Machines



Containers

NVIDIA Container Runtime



<https://github.com/NVIDIA/nvidia-docker>

- Colloquially called “nvidia-docker”
- Docker containers are *hardware-agnostic* and *platform-agnostic*
- NVIDIA GPUs are specialized hardware that require the NVIDIA driver
- Docker does not natively support NVIDIA GPUs with containers
- NVIDIA Container Runtime makes the images agnostic of the NVIDIA driver
 - ◆ Required character devices and driver files are mounted when starting the container on the target machine
 - ◆ This makes Docker images portable while still leveraging NVIDIA GPUs

Docker

Definitions

Image

Docker images are the basis of containers. An Image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. An image typically contains a union of layered filesystems stacked on top of each other. An image does not have state and it never changes.

Container

A container is a runtime instance of a docker image.

A Docker container consists of

- A Docker image
- Execution environment
- A standard set of instructions

<https://docs.docker.com/engine/reference/glossary/>

Managing Images, Containers and Running Containers

Managing Images and Containers

Common Commands

List Images:

```
docker images
```

Remove an Image:

```
docker rmi imageID
```

```
docker rmi tensorRT
```

Remove all of your images:

The `-a` flag means "all" and the `-q` flag makes the output a list of imageID's.

```
docker rmi $(docker images -a -q)
```

List Containers:

```
docker ps -a
```

Stop a running Container:

```
docker stop containerID
```

Remove a Container:

```
docker rm containerID
```

Remove all containers:

Remove all running containers (`-f` will try to force a shutdown of the container if it is running.)

```
docker rm -f $(docker ps -a -q)
```

- Refer to images and containers by their ID hash
- The first few characters of the image/container hash will do

Docker images detail

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nvidia/cuda	8.0-devel	5094464ddfe8	2 weeks ago	1.62 GB
ubuntu	latest	f49eec89601e	2 weeks ago	129 MB
nvcr.io/nvidia/tensorflow	17.01	4352527009ae	2 weeks ago	2.77 GB

Image Name = Repository:Tag

ImageID = Unique Hash

Running Containers

docker run and option

docker run Options

- `--runtime=nvidia` enable GPU capabilities
- `--rm` remove the container after it exits
- `-i -t` or `-it` interactive, and connect a "tty"
- `-d --detach` run in the background
- `--name` give the container a name
- `-p 8080:80` port map from host to container
- `-v ~/data:/data` map storage volume from host to container (bind mount) i.e. bind the `~/data` directory in your home directory to `/data` in the container

Starts Tensorflow with ports, volumes, and console (All 1 line):

```
docker run
```

```
--runtime=nvidia
```

```
--rm -it
```

```
--name MyCoolContainer
```

```
-p 8888:80
```

```
-v ~/data:/data
```

```
nvcr.io/nvidia/tensorflow:18.01-py2
```

```
examples/nvcnn.py
```

Navigating the NGC WebUI

NVIDIA GPU Cloud

How do we actually use it?

Our challenge:

- Sign up for a *free* NGC account at www.nvidia.com/ngc/signup
- Login to the WebUI
- Generate an API key for Docker to use

What is an API Key?

(And why do you need one?)

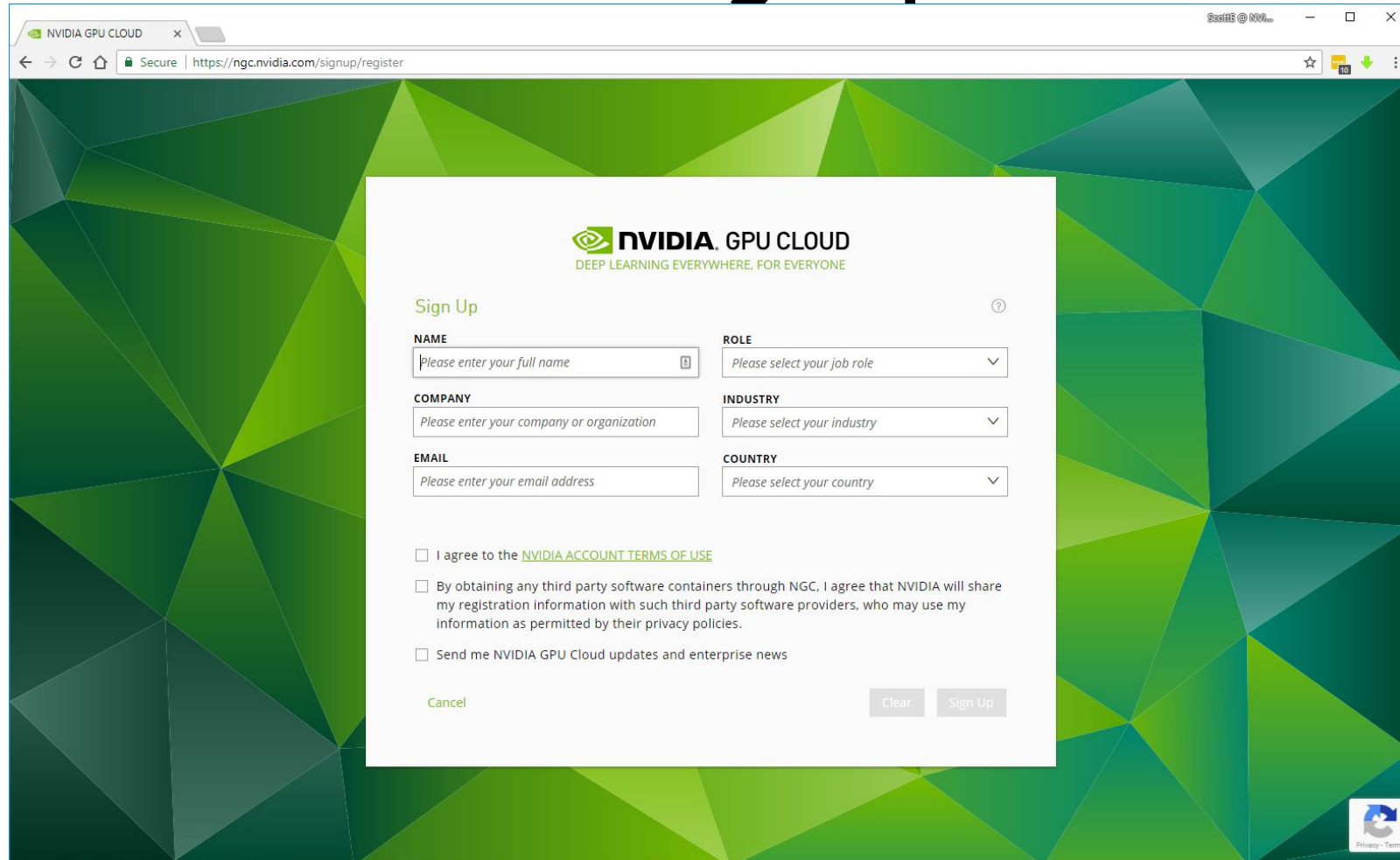
Your API key represents your credentials

- Used for programmatic interaction (e.g., docker, REST API, etc.)
- Uniquely identifies you (think “Username & Password”)
- There can be only one (regenerating your API key invalidates the old one)

WebUI at ngc.nvidia.com: Use Username & Password

Programmatic interface at nvcr.io: Use API Key

NGC Sign-up



NVIDIA GPU CLOUD
DEEP LEARNING EVERYWHERE, FOR EVERYONE

Sign Up

NAME
Please enter your full name

ROLE
Please select your job role

COMPANY
Please enter your company or organization

INDUSTRY
Please select your industry

EMAIL
Please enter your email address

COUNTRY
Please select your country

I agree to the [NVIDIA ACCOUNT TERMS OF USE](#)

By obtaining any third party software containers through NGC, I agree that NVIDIA will share my registration information with such third party software providers, who may use my information as permitted by their privacy policies.

Send me NVIDIA GPU Cloud updates and enterprise news

Cancel Clear Sign Up

Privacy - Terms

NGC Access

The screenshot shows a web browser window with the following details:

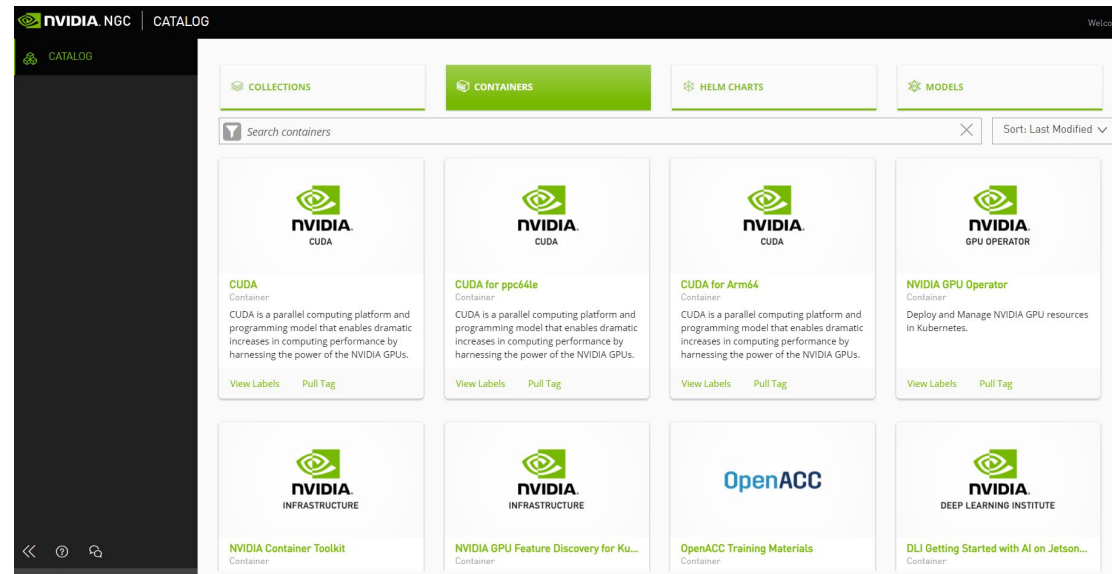
- Browser tab: NVIDIA GPU CLOUD
- Address bar: <https://ngc.nvidia.com/signin/email>
- Page Content:
 - Logo: NVIDIA GPU CLOUD
 - Section: Sign In
 - Label: EMAIL
 - Input field: Please enter your email address
 - Buttons: Sign Up, Next

NGC WebUI

Where it all begins

When you login...

- Collections
- Containers List
- Instructions and Info
- Image specifics
- Docker pull shortcut

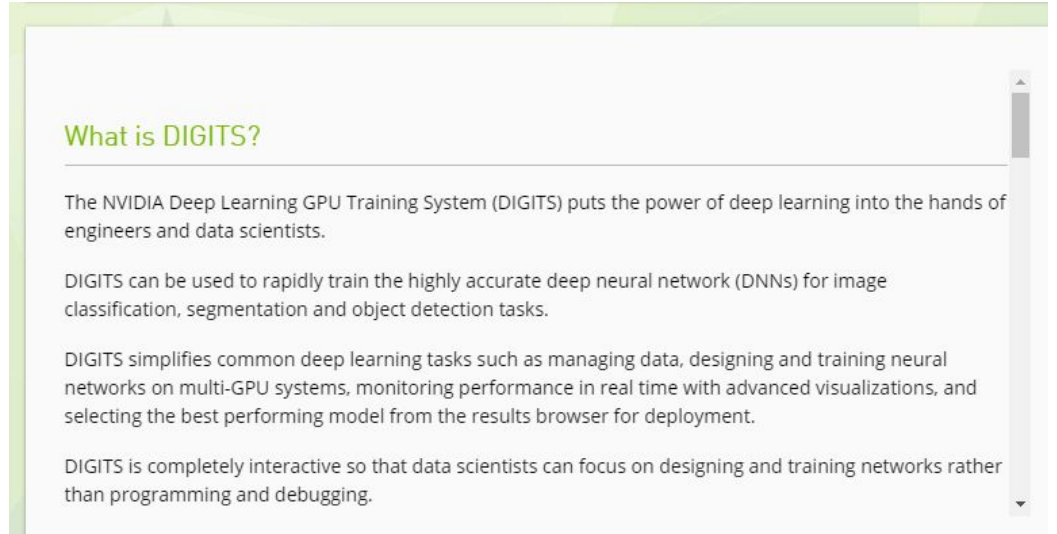


NGC WebUI

Instructions and Information

When you select a container

- Description on what the image contains
- Usually examples on running it
- Often has links for more information and tutorials



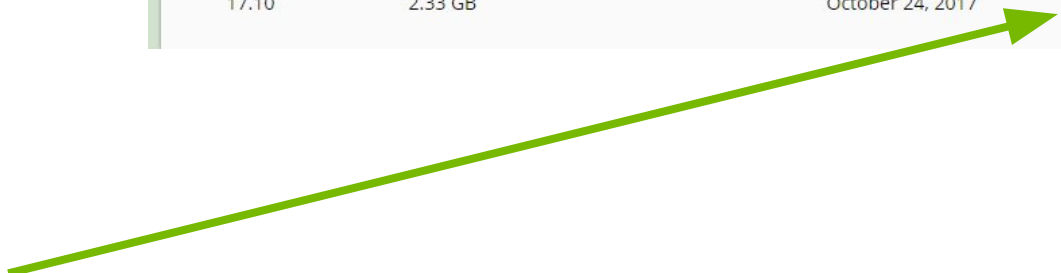
NGC WebUI

Image Specifics

List of images for that container

- **Tag** (`nvidia/caffe:18.01`)
 - Follows YY.MM format
- **Creation date**
 - Updated monthly
- **Shortcut to copy** `docker pull` command to clipboard

TAG	SIZE	USER	LAST MODIFIED	PULL
18.01	1.99 GB		January 23, 2018	↓
17.12	1.99 GB		December 3, 2017	↓
17.11	2.07 GB		November 16, 2017	↓
17.10	2.33 GB		October 24, 2017	↓

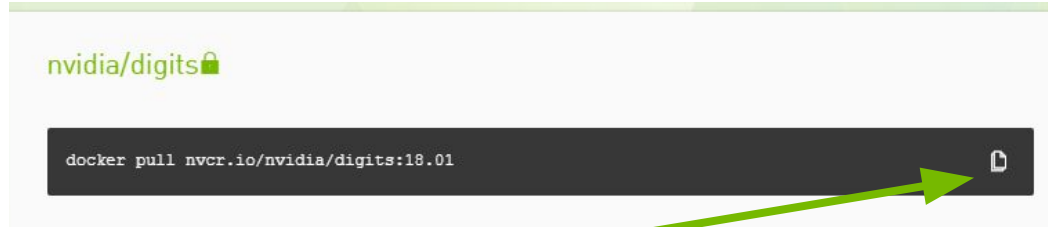


NGC WebUI

Docker pull shortcut

Shortcut to the latest at the top

- Shows full image name
(`nvcr.io/nvidia/digits:18.01`)
- Icon to copy to clipboard
 - Same as in image details

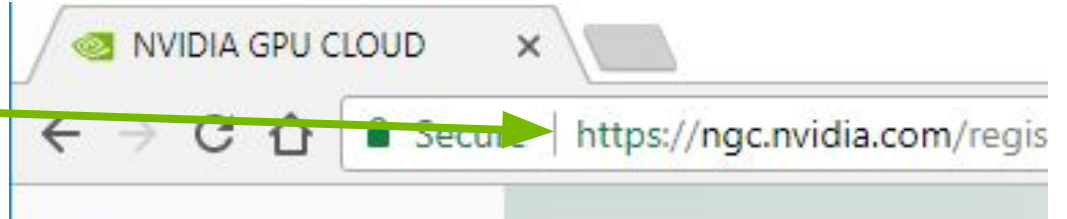


NGC vs. NVCR

Why are there two FQDNs?

ngc.nvidia.com

- NGC = NVIDIA GPU Cloud
- Used for administrative tasks
 - (a.k.a. The WebUI)

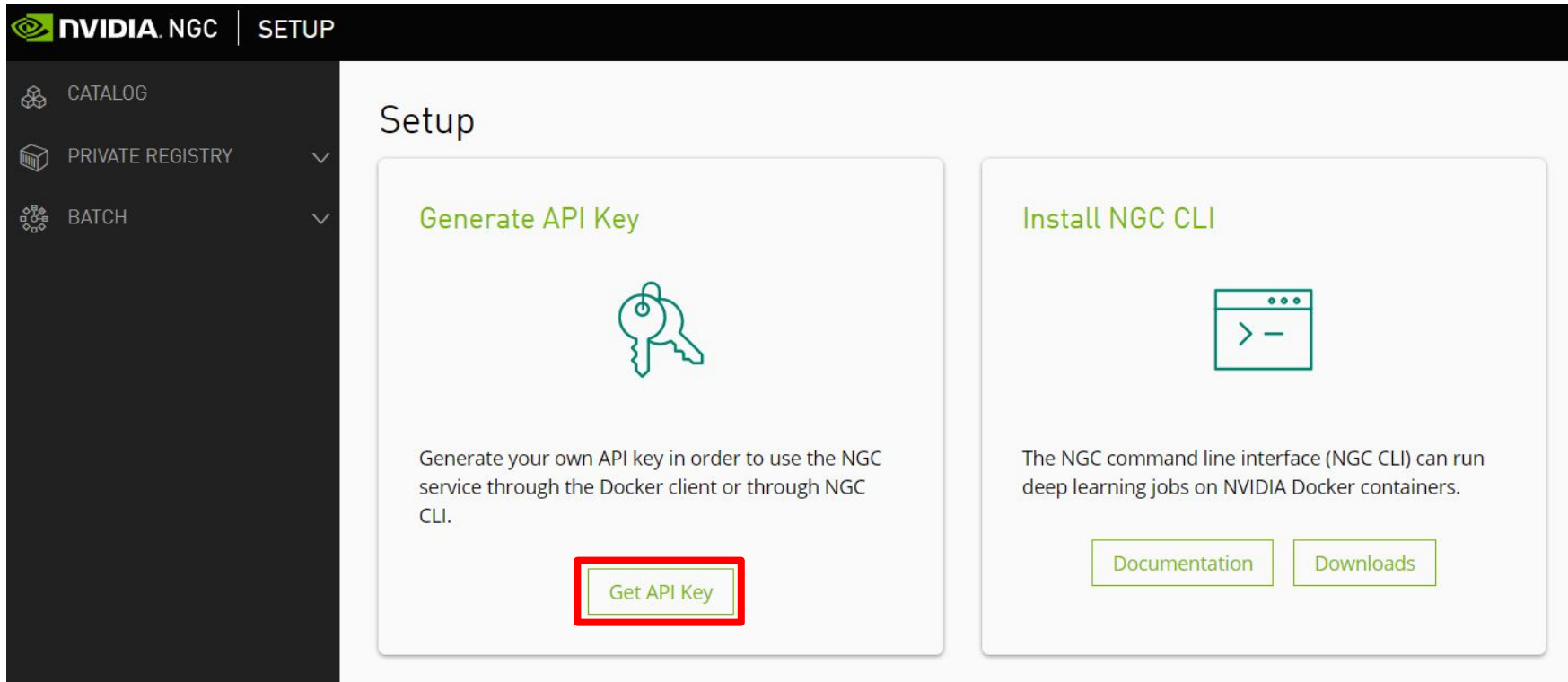


nvcr.io

- NVIDIA Container Repository
- Used for Docker tasks



NGC API Key



The screenshot shows the NVIDIA NGC Setup page. The left sidebar contains navigation links: CATALOG, PRIVATE REGISTRY, and BATCH. The main content area is titled 'Setup' and contains two cards. The first card, 'Generate API Key', features a key icon and a red-bordered button labeled 'Get API Key'. The second card, 'Install NGC CLI', features a terminal icon and two buttons: 'Documentation' and 'Downloads'.

Generate API Key

Generate your own API key in order to use the NGC service through the Docker client or through NGC CLI.

[Get API Key](#)

Install NGC CLI

The NGC command line interface (NGC CLI) can run deep learning jobs on NVIDIA Docker containers.

[Documentation](#) [Downloads](#)

NGC API Key

The screenshot shows the NVIDIA NGC Setup interface. The top navigation bar includes the NVIDIA NGC logo, a 'SETUP' tab, and user information for 'sae' and 'João Paulo de Oliveira'. A left sidebar contains navigation options: 'CATALOG', 'PRIVATE REGISTRY', and 'BATCH'. The main content area is titled 'Setup > API Key' and features a prominent green 'Generate API Key' button in the top right corner, which is highlighted with a red border. Below the title, there is a section for 'API Information' explaining the key's purpose and a 'Generate API Key' button. A 'Usage' section provides instructions for logging in to the NGC registry using the CLI and Docker. The CLI section shows the command '\$ ngc config set' in a dark terminal box. The Docker section shows the command '\$ docker login nvr.io' and the prompts for 'Username: \$oauthtoken' and 'Password: <Your Key>' in another dark terminal box.

Generate API Key

API

API Information

Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage

Use your API key to log in to the NGC registry by entering the following command and following the prompts:

NGC CLI

```
$ ngc config set
```

Docker™

For the username, enter '\$oauthtoken' exactly as shown. It is a special authentication token for all users.

```
$ docker login nvr.io

Username: $oauthtoken
Password: <Your Key>
```


NGC API Key Generate

The screenshot shows a web browser window at <https://ngc.nvidia.com/configuration/api-key>. The page title is "Configuration > API Key" and there is a "Generate API Key" button in the top right. The main content area is titled "API" and contains the following text:

API Information
Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage
Use your API key

Docker™
For the username

```
$ docker login  
Username: goah  
Password: <Your Key>
```

A modal dialog box titled "Generate a New API Key" is overlaid on the page. It contains the text: "Selecting 'Confirm' will generate a new API Key, and your old API Key will become invalid." The dialog has a "Cancel" button and a "Confirm" button, which is highlighted with a red square.

NGC API Key Save

The screenshot shows the NVIDIA GPU Cloud (NGC) configuration page for generating an API key. The browser address bar shows the URL `https://ngc.nvidia.com/configuration/api-key`. The page title is "Configuration > API Key". A green button labeled "Generate API Key" is visible in the top right. The main content area contains the following text:

API

API Information
Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage

Use your API key to log in to the NGC registry as follows.

Docker™

For the username, enter 'soauthtoken' exactly as shown. It is a special authentication token for all users.

```
$ docker login nvcr.io

Username: soauthtoken
Password: a2VmcWkxZG11NHE2amMyNj1zZGY2MkUwOVM6Zjg4ZGQxMGMtNGV1Yy00NTAyLThhY2EtMmQ3OD1jMWFhMmQ4
```

API Key generated successfully. This is the only time your API Key will be displayed. Keep your API Key secret. Do not share it or store it in a place where others can see or copy it.

API Key: `a2VmcWkxZG11NHE2amMyNj1zZGY2MkUwOVM6Zjg4ZGQxMGMtNGV1Yy00NTAyLThhY2EtMmQ3OD1jMWFhMmQ4`

EXERCISE

- Use the API key to login into `nvcr.io`
- Pull the TensorFlow container `nvcr.io/nvidia/tensorflow:18.09-py3`

Run a Container

Container Execution

Quick TensorFlow Run

Our challenge:

- Download a TensorFlow container from NGC to local machine
- Run `nvcnn.py` sample with Resnet-50 with synthetic ImageNet data for 200 epochs

Container Execution

The screenshot shows the NVIDIA GPU Cloud Registry interface. The browser address bar indicates the URL `https://ngc.nvidia.com/registry/nvidia-tensorflow`. The page title is "Registry" and the user is logged in as "Scott Ellis".

On the left sidebar, the "Registry" menu item is highlighted with a red box. Below it, the "Repositories" list includes "nvidia" (expanded) and "tensorflow" (highlighted with a red box). Other repositories listed include "caffe", "caffe2", "cntk", "cuda", "digits", "mxnet", "tensorflow", "theano", "torch", "hpc", "nvidia-hpcvis", and "partners".

The main content area displays the "nvidia/tensorflow" repository details. A "Get API Key" button is visible in the top right. Below the repository name, a code block contains the Docker pull command: `docker pull nvcr.io/nvidia/tensorflow:18.02-py3`. The "D" icon next to the command is highlighted with a red box.

Below the code block, there is a section titled "What is TensorFlow?" followed by a description of TensorFlow as an open source software library for numerical computation using data flow graphs. Below this is a section titled "Running TensorFlow" which contains a table of tags.

TAG	SIZE	USER	LAST MODIFIED	PULL
18.02-py3	1.28 GB		March 2, 2018	↓
18.02-py2	1.28 GB		March 2, 2018	↓
18.01-py2	1.25 GB		January 23, 2018	↓

Container Execution

```
ubuntu@ip-172-31-7-211: ~$ docker pull nvr.io/nvidia/tensorflow:18.02-py3
18.02-py3: Pulling from nvidia/tensorflow
f2233041f557: Pulling fs layer
f321bcc6a76c: Pulling fs layer
2f25d8d1d058: Pulling fs layer
87bfe0d2f0e8: Waiting
145c1bf7947a: Waiting
15202f146e8c: Pulling fs layer
fc343880341c: Waiting
c76797225986: Pulling fs layer
fc343880341c: Extracting 41.78MB/441.4MB
237b60dff0dc: Download complete
1351c57ada10: Download complete
664148fee1cd: Download complete
2b5f7180c2ba: Download complete
f73c577f846c: Download complete
c2805e961180: Download complete
a11994d7f7e6: Download complete
e4ec81895411: Download complete
c7109af566c9: Download complete
d4937e88f3c4: Download complete
41069482ab6f: Download complete
cbe901b383c0: Download complete
60d515739c44: Download complete
6cca44f2668a: Download complete
ec064d02537e: Downloading 124.7MB/169.6MB
d31c94251367: Download complete
85757e374f6b: Download complete
```


Container Execution

```
ubuntu@ip-172-31-7-211: ~  
2b5f7180c2ba: Pull complete  
664148fee1cd: Extracting      472B/472B  
664148fee1cd: Download complete  
2b5f7180c2ba: Download complete  
f73c577f846c: Pull complete  
c2805e961180: Pull complete  
a11994d7f7e6: Pull complete  
e4ec81895411: Pull complete  
c7109af566c9: Pull complete  
d4937e88f3c4: Pull complete  
41069482ab6f: Pull complete  
cbe901b383c0: Pull complete  
60d515739c44: Pull complete  
6cca44f2668a: Pull complete  
ec064d02537e: Pull complete  
d31c94251367: Pull complete  
85757e374f6b: Pull complete  
a685c53320ed: Pull complete  
f7e832cb61d2: Pull complete  
f743b7cb9be2: Pull complete  
0c395732af81: Pull complete  
7ee97eeb04b4: Pull complete  
e8c1d8550a0d: Pull complete  
65154325fda5: Pull complete  
fb91e851e672: Pull complete  
Digest: sha256:899f5407ac404eb94c8277d8ff845e2946e1e5e24639aa3b6e75f15de12a7120  
Status: Downloaded newer image for nvcr.io/nvidia/tensorflow:18.02-py3  
ubuntu@ip-172-31-7-211:~$
```

Container Execution

```
ubuntu@ip-172-31-7-211: ~$ docker run --runtime=nvidia --rm -ti nvcr.io/nvidia/tensorflow:18.02-py3 nvidia-examples/cnn/mcnn.py --img_resnet50 --num_batches=200 --fp16

=====
== TensorFlow ==
=====

NVIDIA Release 18.02 (build 337102)

Container image Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.
Copyright 2017 The TensorFlow Authors. All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or file.
```

Container Execution

```
ubuntu@ip-172-31-7-211: ~  
178 1 641.4 11.027 0.10000  
179 1 640.2 11.050 0.10000  
180 1 639.4 11.035 0.10000  
181 1 639.4 11.027 0.10000  
182 1 639.8 11.014 0.10000  
183 1 637.2 11.001 0.10000  
184 1 640.5 10.994 0.10000  
185 1 641.1 10.986 0.10000  
186 1 640.4 10.979 0.10000  
187 1 641.6 10.972 0.10000  
188 1 642.4 10.965 0.10000  
189 1 642.7 10.961 0.10000  
190 1 642.5 10.980 0.10000  
191 1 640.0 10.992 0.10000  
192 1 642.3 10.986 0.10000  
193 1 641.6 10.977 0.10000  
194 1 641.3 10.976 0.10000  
195 1 641.5 10.990 0.10000  
196 1 641.7 10.981 0.10000  
197 1 642.4 10.975 0.10000  
198 1 640.9 10.967 0.10000  
199 1 641.0 10.962 0.10000  
200 1 641.6 10.957 0.10000  
-----  
Images/sec: 640.9 +/- 0.1 (jitter = 1.2)  
-----  
ubuntu@ip-172-31-7-211:~$
```

Container Execution

Quick TensorFlow Run

Our challenge:

- Download a TensorFlow container from NGC to local machine
- Run nvcnn.py sample with Resnet-50 with synthetic ImageNet data for 200 epochs
- Hint:

```
# docker run --runtime=nvidia --rm -ti nvcr.io/nvidia/tensorflow:18.04-py3  
./nvidia-examples/cnn/nvcnn.py -m resnet50 --num_batches=200 --fp16
```

Wait. What Just Happened?

That was too fast.

1. Logged into NGC and created an API key
2. Downloaded the TensorFlow container from NGC
3. Ran nvcnn.py sample with Resnet-50 with synthetic ImageNet data for 200 epochs
4. Profit!

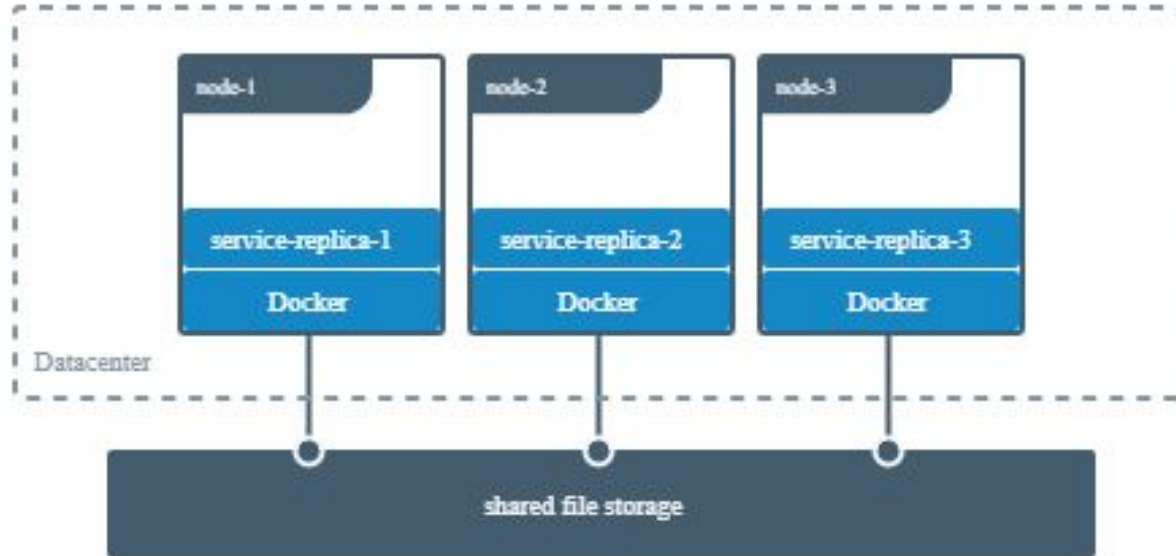
Keeping your Data Persistent: Docker Volumes

HPC Cluster Architecture



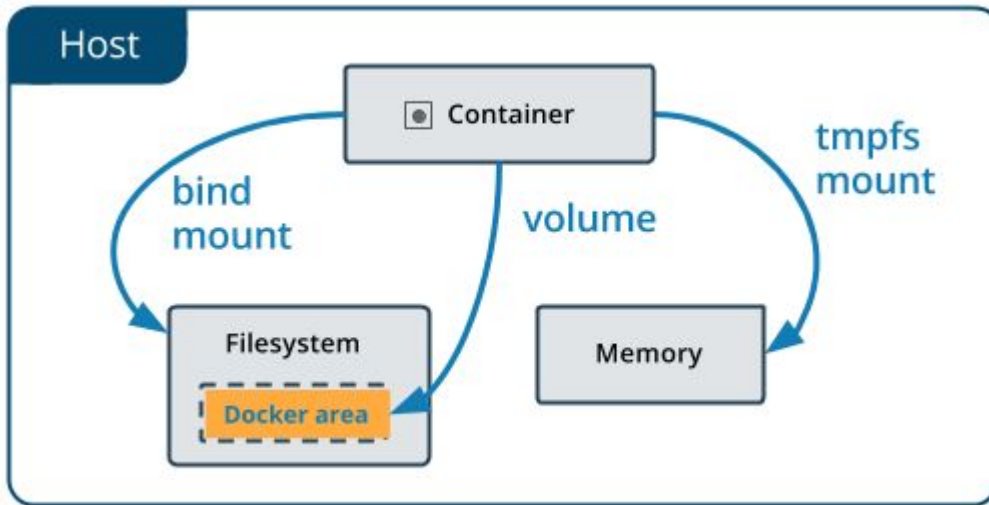
Docker Volume

Persistent Data Across Multiple Containers



Docker Volume

Persistent Data Across Containers Executions



```
docker run \  
  --runtime=nvidia \  
  --rm -it \  
  --name MyCoolContainer \  
  -p 8888:80 \  
  -v ~/data:/data \  
  nvcr.io/nvidia/tensorflow:18.04-py3 \  
  bash
```

<https://docs.docker.com/storage/volumes/>

Docker Volume Demo

Exposing Ports

Info: Accessing Container Services

What about things in the container?

Applications in a container are on their own network ('docker0' bridge)

```
ubuntu@ip-172-31-7-161: ~  
ubuntu@ip-172-31-7-161:~$ ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000  
    link/ether 0a:8f:c6:32:37:24 brd ff:ff:ff:ff:ff:ff  
    inet 172.31.7.161/20 brd 172.31.15.255 scope global ens3  
        valid_lft forever preferred_lft forever  
    inet6 fe80::88f:c6ff:fe32:3724/64 scope link  
        valid_lft forever preferred_lft forever  
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default  
    link/ether 02:42:75:48:55:43 brd ff:ff:ff:ff:ff:ff  
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0  
        valid_lft forever preferred_lft forever  
ubuntu@ip-172-31-7-161:~$
```

Tell Docker you want to use them at runtime (remember `-p` ?)

Challenge: Accessing Container Services

Passing through a TCP port

Our challenge:

- Launch the TensorFlow container
 - Run interactively (-ti)
 - Expose Tensorboard port 6006 (-p)
- Repeat our prior training run
 - Save log data to /tmp
- Run Tensorboard
- Visualize our training run

Hint: Remember if you only have 1 GPU in your AMI, you won't be able to flag for 8.

```
# docker run --runtime=nvidia --rm -ti -p 6006:6006 nvcr.io/nvidia/tensorflow:18.09-py3
/workspace# mpiexec --allow-run-as-root --bind-to socket -np 1 python nvidia-examples/cnn/resnet.py
--layers=18 --precision=fp16 --num_iter=200 --log_dir=/tmp --batch_size=64
/workspace# tensorboard --logdir=/tmp
```

Solution: Accessing Container Services

```
2. root@b2615a14c51c: /workspace (ssh)
ubuntu@ip-172-31-8-19:~$ docker run --runtime=nvidia --rm -ti -p 6006:6006 nvcr.io/nvidia/tensorflow:18.09-py3

=====
== TensorFlow ==
=====

NVIDIA Release 18.09 (build 687558)

Container image Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.
Copyright 2017 The TensorFlow Authors. All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or file
.

root@b2615a14c51c:/workspace#
```

Solution: Accessing Container Services

```
2. root@c6a430040b11: /workspace (ssh)

=====
== TensorFlow ==
=====

NVIDIA Release 18.09 (build 687558)

Container image Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.
Copyright 2017 The TensorFlow Authors. All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or file
.

root@c6a430040b11:/workspace# mpiexec --allow-run-as-root --bind-to socket -np 1 python nvidia-exa
mples/cnn/resnet.py --layers=50 --precision=fp16 --num_iter=200 --log_dir=/tmp
-----
WARNING: Open MPI tried to bind a process but failed. This is a
warning only; your job will continue, though performance may
be degraded.

Local host:      c6a430040b11
```


Solution: Accessing Container Services

```
2. root@b2615a14c51c: /workspace (ssh)
10 10.0 125.5 4.497 5.469 1.84320
20 20.0 856.1 0.094 1.071 1.65620
30 30.0 865.6 0.432 1.415 1.47920
40 40.0 864.7 0.183 1.169 1.31220
50 50.0 863.5 0.062 1.052 1.15520
60 60.0 865.0 0.381 1.374 1.00820
70 70.0 865.1 0.054 1.050 0.87120
80 80.0 865.5 0.171 1.170 0.74420
90 90.0 865.3 0.255 1.256 0.62720
100 100.0 865.5 0.134 1.137 0.52020
110 110.0 865.6 0.073 1.078 0.42320
120 120.0 865.7 0.049 1.057 0.33620
130 130.0 864.6 0.035 1.044 0.25920
140 140.0 865.7 0.085 1.095 0.19220
150 150.0 865.4 0.004 1.014 0.13520
160 160.0 865.5 0.008 1.018 0.08820
170 170.0 865.8 0.001 1.010 0.05120
180 180.0 866.9 0.000 1.009 0.02420
190 190.0 866.8 0.000 1.009 0.00720
200 200.0 865.9 0.000 1.009 0.00020
root@b2615a14c51c:/workspace#
```


Solution: Accessing Container Services

```
2. root@c6a430040b11: /workspace (ssh)

120 120.0 867.4 0.057 1.050 0.33620
130 130.0 866.6 0.122 1.117 0.25920
140 140.0 866.9 0.128 1.123 0.19220
150 150.0 867.8 0.135 1.131 0.13520
160 160.0 867.4 0.135 1.131 0.08820
170 170.0 867.0 0.002 0.999 0.05120
180 180.0 868.1 0.001 0.997 0.02420
190 190.0 868.6 0.000 0.996 0.00720
200 200.0 868.1 0.000 0.996 0.00020

root@c6a430040b11:/workspace# tensorboard --logdir=/tmp
W1030 17:49:45.120258 Reloader tf_logging.py:120] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph events. Overwriting the graph with the newest event.
W1030 17:49:45.120258 139844522014464 tf_logging.py:120] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph events. Overwriting the graph with the newest event.
W1030 17:49:45.124734 Reloader tf_logging.py:120] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
W1030 17:49:45.124733 139844522014464 tf_logging.py:120] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
TensorBoard 1.10.0 at http://c6a430040b11:6006 (Press CTRL+C to quit)
```

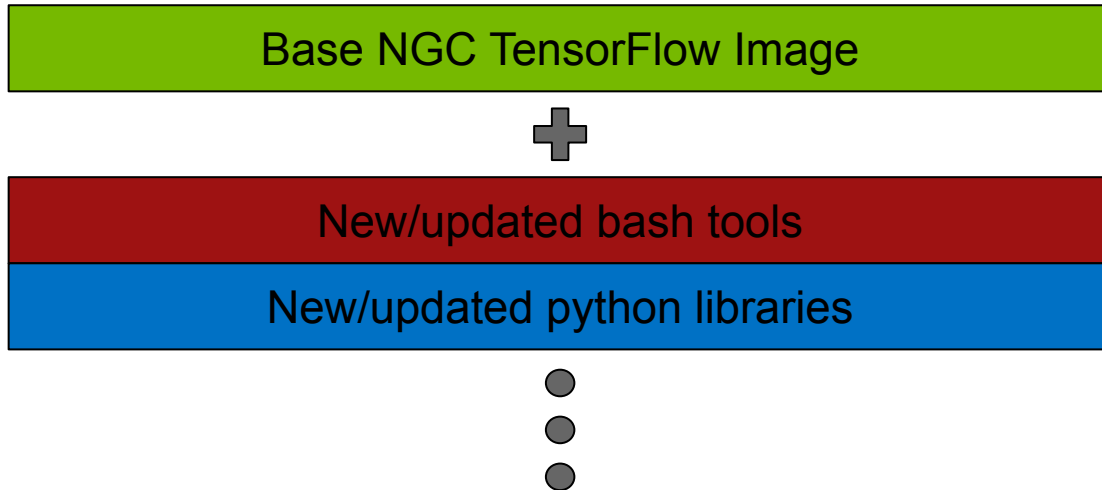
Solution: Accessing Container Services

The screenshot displays the TensorBoard web interface. The browser's address bar shows the URL `ec2-54-213-139-71.us-west-2.compute.amazonaws.com:6006/#projector`, which is highlighted with a red rectangular box. A large green arrow originates from this box and points towards a green callout box on the right side of the interface. The callout box contains the text "Works because of Security Group settings!". The TensorBoard interface itself shows a navigation bar with tabs for SCALARS, GRAPHS, DISTRIBUTIONS, HISTOGRAMS, PROJECTOR, and INACTIVE. The main content area displays a PCA plot of data points, with a left sidebar for data management and a right sidebar for search and selection options.

Extending NGC Images

Info: Extending NGC Images

Layers layers layers



- Often the out-of-the-box image is not enough
- Need extra tools/applications
 - Additional layers on top of base image

- Dockerfile allows for building custom images
 - `docker build` command creates new image from set of instructions

Info: Extending NGC Images

A Dockerfile is a script that contains instructions to custom configure a container from a base image

Here are some common commands:

- **FROM** is Mandatory as the first instruction. It denotes the base image to be built from. Use a tag to specify the image.
- **RUN** = Creates a new layer with the output of the specified commands.
- **WORKDIR** = Directory the command will start it
- **CMD** = Default command executed when Docker container is started. Use only one CMD instruction in a Dockerfile.

```
1 FROM nvcr.io/nvidia/tensorflow:18.02-py3
2
3 RUN pip install jupyter
4
5 WORKDIR /notebooks
6
7 CMD jupyter notebook --allow-root --ip=0.0.0.0
```

Best practices for writing Dockerfiles

https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/

Challenge: Extending NGC Images

Add Jupyter

Our challenge:

- Add Jupyter to the NVIDIA TensorFlow Image
- Launch jupyter notebook automatically when the container starts
 - By default jupyter listens on port 8888
- Verify it worked!

```
# mkdir MyImage
# vi MyImage/Dockerfile

FROM nvcr.io/nvidia/tensorflow:18.09-py3
RUN pip install jupyter
WORKDIR /notebooks
CMD jupyter notebook --allow-root --ip=0.0.0.0

# docker build -t myimage:latest MyImage
# docker images
# docker run --runtime=nvidia --rm -ti -p 8888:8888 myimage:latest
```

Solution: Extending NGC Images

```
ubuntu@ip-172-31-7-211: ~$ mkdir MyImage
ubuntu@ip-172-31-7-211: ~$ vi MyImage/Dockerfile
ubuntu@ip-172-31-7-211: ~$ docker build -t myimage:latest MyImage
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM nvcr.io/nvidia/tensorflow:1.15.0-py3
--> 57ae51ee8b74
Step 2/4 : RUN pip install jupyter
--> Running in b9c14a05670f
Collecting jupyter
  Downloading jupyter-1.0.0-py2.py3-none-any.whl
Collecting nbconvert (from jupyter)
  Downloading nbconvert-5.3.1-py2.py3-none-any.whl (386kB)
Collecting jupyter-console (from jupyter)
  Downloading jupyter_console-5.2.0-py2.py3-none-any.whl (121kB)
Collecting ipywidgets (from jupyter)
  Downloading ipywidgets-7.1.2-py2.py3-none-any.whl (68kB)
Collecting ipykernel (from jupyter)
  Downloading ipykernel-4.8.2-py3-none-any.whl (108kB)
Collecting notebook (from jupyter)
```

Use the example from the prior slide as content

Solution: Extending NGC Images

```
ubuntu@ip-172-31-7-211: ~  
ipykernel-4.8.2 ipython-6.2.1 ipython-genutils-0.2.0 ipywidgets-7.1.2 jedi-0.11.1 jinja  
2-2.10 jsonschema-2.6.0 jupyter-1.0.0 jupyter-client-5.2.3 jupyter-console-5.2.0 jupyter  
-core-4.4.0 mistune-0.8.3 nbconvert-5.3.1 nbformat-4.4.0 notebook-5.4.1 pandocfilters-1.  
4.2 parso-0.1.1 pickleshare-0.7.4 prompt-toolkit-1.0.15 pygments-2.2.0 python-dateutil-2  
.7.0 pyzmq-17.0.0 qtconsole-4.3.1 simplegeneric-0.8.1 terminado-0.8.1 testpath-0.3.1 tor  
nado-5.0.1 traitlets-4.3.2 wcwidth-0.1.7 widgetsnbextension-3.1.4  
You are using pip version 9.0.1, however version 9.0.3 is available.  
You should consider upgrading via the 'pip install --upgrade pip' command.  
Removing intermediate container b9c14a05670f  
---> b70170557b8e  
Step 3/4 : WORKDIR /notebooks  
Removing intermediate container 44a11df4fe6e  
---> 79586757db8b  
Step 4/4 : CMD jupyter notebook --allow-root --ip=0.0.0.0  
---> Running in b91c0a2f389a  
Removing intermediate container b91c0a2f389a  
---> befe343b36d3  
Successfully built befe343b36d3  
Successfully tagged myimage:latest  
ubuntu@ip-172-31-7-211:~$ docker images  
REPOSITORY          TAG                IMAGE ID           CREATED  
SIZE  
myimage             latest            befe343b36d3     20 seconds ago  
3GB  
nvcr.io/nvidia/tensorflow  18.02-py3        57ae51ee8b74     5 weeks ago  
2.91GB  
ubuntu@ip-172-31-7-211:~$
```

Our new image is here!
myimage:latest

Solution: Extending NGC Images

```
ubuntu@ip-172-31-7-211: ~  
ubuntu@ip-172-31-7-211:~$ docker run --runtime=nvidia -p 8888:8888 --rm -ti myimage:latest  
  
=====  
== TensorFlow ==  
=====  
  
NVIDIA Release 18.02 (build 337102)  
  
Container image Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.  
Copyright 2017 The TensorFlow Authors. All rights reserved.  
  
Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.  
NVIDIA modifications are covered by the license terms that apply to the underlying project or file.  
  
[I 21:24:49.047 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret  
[I 21:24:49.348 NotebookApp] Serving notebooks from local directory: /notebooks  
[I 21:24:49.348 NotebookApp] 0 active kernels  
[I 21:24:49.348 NotebookApp] The Jupyter Notebook is running at:  
[I 21:24:49.348 NotebookApp] http://0.0.0.0:8888/?token=46edb99a6d0fdddd72a0cb463ab5f4bcba1334fd100e0fd5  
[I 21:24:49.348 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).  
[W 21:24:49.349 NotebookApp] No web browser found: could not locate runnable browser.  
[C 21:24:49.349 NotebookApp]
```

Solution: Extending NGC Images

```
ubuntu@ip-172-31-7-211: ~  
=====
```

NVIDIA Release 18.02 (build 337102)

Container image Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.
Copyright 2017 The TensorFlow Authors. All rights reserved.

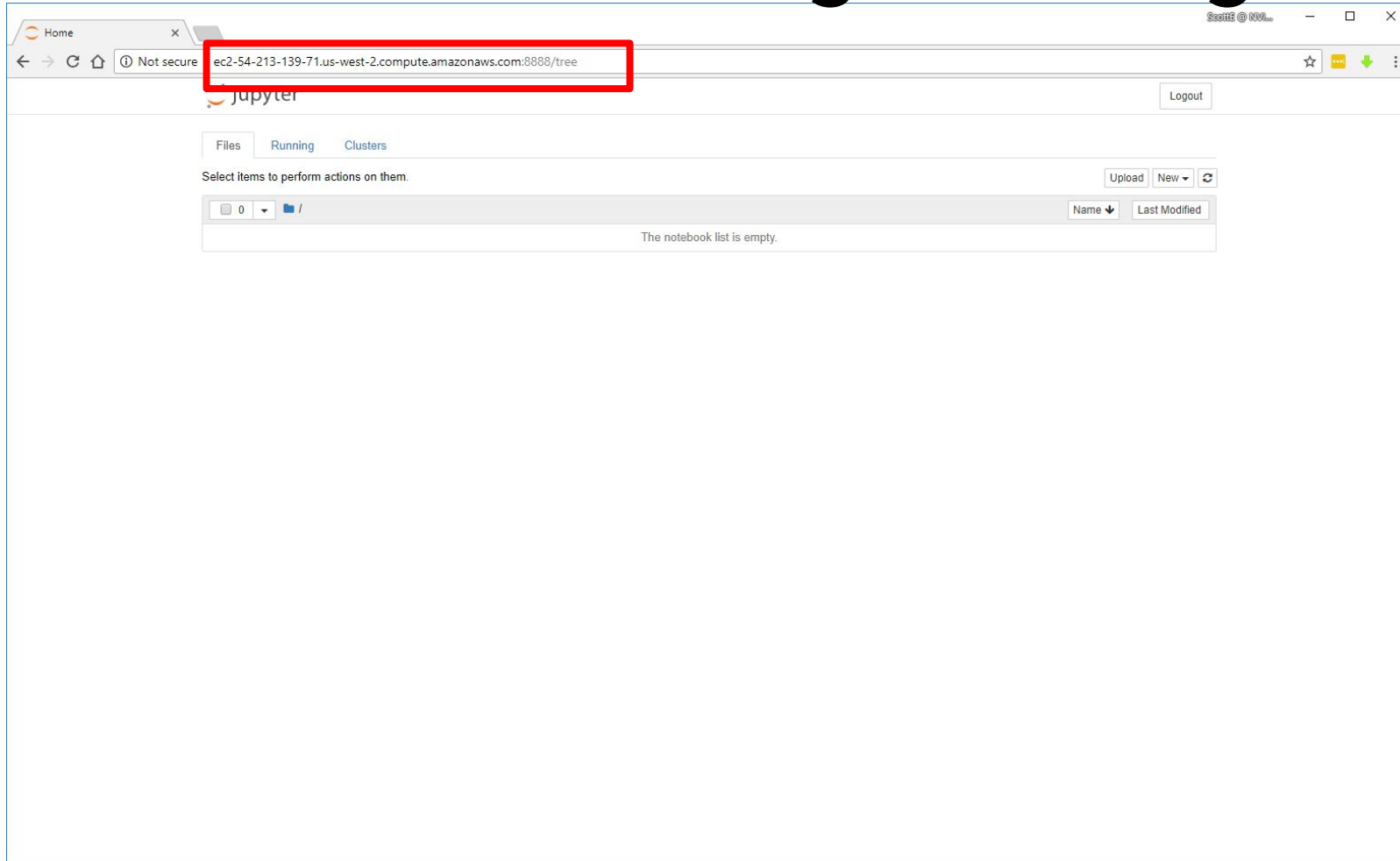
Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or file.

```
[I 21:24:49.047 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret  
[I 21:24:49.348 NotebookApp] Serving notebooks from local directory: /notebooks  
[I 21:24:49.348 NotebookApp] 0 active kernels  
[I 21:24:49.348 NotebookApp] The Jupyter Notebook is running at:  
[I 21:24:49.348 NotebookApp] http://0.0.0.0:8888/?token=46edb99a6d0fddddd72a0cb463ab5f4bcba1334fd100e0fd5  
[I 21:24:49.348 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).  
[W 21:24:49.349 NotebookApp] No web browser found: could not locate runnable browser.  
[C 21:24:49.349 NotebookApp]
```

Copy/paste this URL into your browser when you connect for the first time, to login with a token:

```
http://0.0.0.0:8888/?token=46edb99a6d0fddddd72a0cb463ab5f4bcba1334fd100e0fd5
```

Solution: Extending NGC Images



Push an image to a Repository

Summary

- Push an image or a repository to a registry
- If you are pushing to the NVIDIA DGX container registry an Internet connection is required
- Allows images to be shared between systems















```
$ docker tag nvcr.io/nvidia/digits:17.04  
nvcr.io/partner/digits:17.04  
  
$ docker push nvcr.io/partner/digits:17.04
```

Info: Pushing an Image to a Repository

```
o/nvidia/cuda      8.0-cudnn6-devel-ubuntu16.04  9a68a1b7ebbc    6 weeks ago    1.616 GB
world              latest                          48b5124b2768    4 months ago   1.84 kB
@dgx-1:~$
@dgx-1:~$ docker tag nvcr.io/nvidia/digits:17.04 nvcr.io/partner/digits:17.04
@dgx-1:~$
@dgx-1:~$ docker images
REPOSITORY          TAG                IMAGE ID          CREATED          SIZE
o/nvidia/digits    17.04             3736f3fe071f     5 weeks ago     4.171 GB
o/partner/digits   17.04             3736f3fe071f     5 weeks ago     4.171 GB
o/nvidia/cuda      8.0-cudnn6-devel-ubuntu16.04  9a68a1b7ebbc    6 weeks ago    1.616 GB
world              latest            48b5124b2768    4 months ago   1.84 kB
@dgx-1:~$
@dgx-1:~$ docker push nvcr.io/partner/digits:17.04
The push refers to a repository [nvcr.io/partner/digits]
975e2d: Pushed
06b8a8: Pushed
3a11af: Pushed
783618: Pushed
372284: Pushed
1ab8fe: Pushed
40342a: Pushed
e31131: Pushed
3a58f3: Pushed
379d53: Pushed
557c24: Pushed
e22efa: Pushed
753c6b: Pushed
5d4f80: Pushed
56827159aa8b: Pushed
440e02c3dcde: Pushed
29660d0e5bb2: Pushed
85782553e37a: Pushed
745f5be9952c: Pushed
17.04: digest: sha256:b5da6bc51bf0da3f414db691cbee8030dfcf1659d9d95308b210a77e95b9df08 size: 15193
dgxuser@dgx-1:~$
```

Info: Pushing an Image to a Repository

Registry PUSH COMMAND

-  nvidia
-  caffe
-  caffe2
-  cntk
-  cuda
-  digits
-  mxnet
-  pytorch
-  tensorflow
-  theano
-  torch
-  partner
-  digits
-  kinetica

PARTNER/DIGITS 🗑️

📄 README✎

No Repository Description

📁 TAGS

NAME	LAST MODIFIED ▾	SIZE
17.04	12 minutes ago	1.48 GB

GPU Isolation

GPU Isolation

By default, NVIDIA-DOCKER will grant access to all GPUs in the system

Use `NV_GPU` to assign specific GPUs to the running container

Examples of GPU isolation:

Running `nvidia-docker` isolating specific GPUs by index

```
NV_GPU=0,1,5 nvidia-docker run --rm -it  
nvcr.io/nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04
```

Running `nvidia-docker` isolating specific GPUs by UUID

```
NV_GPU='GPU-836c0c09,GPU-b78a60a' nvidia-docker <docker-options>  
<docker-command> <docker-args>
```


Challenge: GPU Isolation

Individual Challenge

- For the two GPU Isolation challenges below run `nvidia-smi` against the image you pulled so the container is removed automatically when the command finishes
- Use `nvidia-docker` with GPU isolation
 - ◆ GPU Isolation - Run `nvidia-smi` with only two (2) GPU's
 - ◆ GPU Isolation - Run `nvidia-smi` interactively with three (3) GPUs

hint: <https://github.com/NVIDIA/nvidia-docker/wiki/GPU-isolation>

Solution: GPU Isolation

Summary

GPU Isolation - Run nvidia-smi with two (2) GPU's

```
$ NV_GPU=0,5 nvidia-docker run --rm  
nvcr.io/nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04 nvidia-smi
```

GPU Isolation - Run nvidia-smi with three (3) GPU's

```
$ NV_GPU=0,1,4 nvidia-docker run --rm -it  
nvcr.io/nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04  
root@<container ID>:/# nvidia-smi  
root@<container ID>:/# exit
```

Solution: GPU Isolation

GPU Isolation with two (2) GPUs

```
dgxuser@dvt8:~$ NV_GPU=0,5 nvidia-docker run --rm nvcr.io/nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04 nvidia-smi
Fri Jun 23 21:31:06 2017

+-----+
| NVIDIA-SMI 375.66                Driver Version: 375.66                |
+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+
|    0   Tesla P100-SXM2...    Off | 0000:06:00.0    Off |             Off |
| N/A   31C    P0     32W / 300W | 0MiB / 16276MiB |    0%      Default |
+-----+-----+-----+
|    1   Tesla P100-SXM2...    Off | 0000:86:00.0    Off |             Off |
| N/A   32C    P0     31W / 300W | 0MiB / 16276MiB |    0%      Default |
+-----+-----+-----+

+-----+
| Processes:                         GPU Memory |
| GPU       PID  Type  Process name                        Usage |
+-----+-----+
| No running processes found |
+-----+

dgxuser@dvt8:~$
```

Solution: GPU Isolation

GPU Isolation interactively with three (3) GPUs

```
dgxuser@dgx-1:~$ NV_GPU=0,1,4 nvidia-docker run --rm -it nvcr.io/nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04
root@22e93990a493:/# nvidia-smi
Mon Apr 17 04:19:15 2017

+-----+
| NVIDIA-SMI 375.20                Driver Version: 375.20          |
+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0   Tesla P100-SXM2...    Off          | 0000:06:00.0  Off  | 0          0          |
| N/A   32C   P0      30W / 300W |  8MiB / 16308MiB |    0%    Default  |
+-----+-----+
|  1   Tesla P100-SXM2...    Off          | 0000:07:00.0  Off  | 0          0          |
| N/A   29C   P0      33W / 300W |  8MiB / 16308MiB |    0%    Default  |
+-----+-----+
|  2   Tesla P100-SXM2...    Off          | 0000:85:00.0  Off  | 0          0          |
| N/A   31C   P0      34W / 300W |  8MiB / 16308MiB |    0%    Default  |
+-----+-----+

+-----+
| Processes:                        GPU Memory Usage |
| GPU       PID  Type  Process name                               | Memory Used |
+-----+-----+
|           |           |      |                   |             |
| No running processes found |
+-----+

root@22e93990a493:/# exit
exit
dgxuser@dgx-1:~$
```

CONTAINERS

João Paulo Navarro (jpnavarro@nvidia.com)
Solutions Architect

