



# NVIDIA GPU CLOUD GPU-OPTIMIZED CONTAINERS

Ashish Sardana | Deep Learning Solutions Architect

 @AshishSardana21

# PREREQUISITE & GOAL

- Familiar with GPU DL
- Some understanding of deep learning frameworks - TF and pyTorch mainly
- Have an understanding of common models like ResNet, AlexNet, VGG and Inception
- Can follow terminal exercises and commands
- When in doubt go to <https://ngc.nvidia.com/>
- Capture the value of GPU optimized containers
- Fast-track your GPU DL development using NGC
- Prevent you from becoming a system admin

# HOW DO YOU SETUP YOUR ENVS TODAY?

## python virtualenvs

```
$sudo apt-get install python-pip python-dev python-virtualenv
```

```
$virtualenv --system-site-packages tensorflow
```

```
$source ~/tensorflow/bin/activate bash
```

```
$source ~/tensorflow/bin/activate.csh
```

```
(tensorflow)$ pip install --upgrade tensorflow-gpu
```

```
(tensorflow)$ pip install myOtherLibsAndPkgs
```

# HOW DO YOU SETUP YOUR ENVS TODAY?

The conda way to getting TF-gpu running

```
$conda create -n tensorflow python=3.5
```

```
$source activate tensorflow
```

```
$conda install pandas matplotlib jupyter notebook scipy scikit-learn nb_conda  
nltk spyder
```

```
$conda install -c conda-forge tensorflow keras
```

```
$pip install gym
```

```
.
```

```
$manyMoreLibsAndPkgs
```

# \$conda list --envs

What gives?

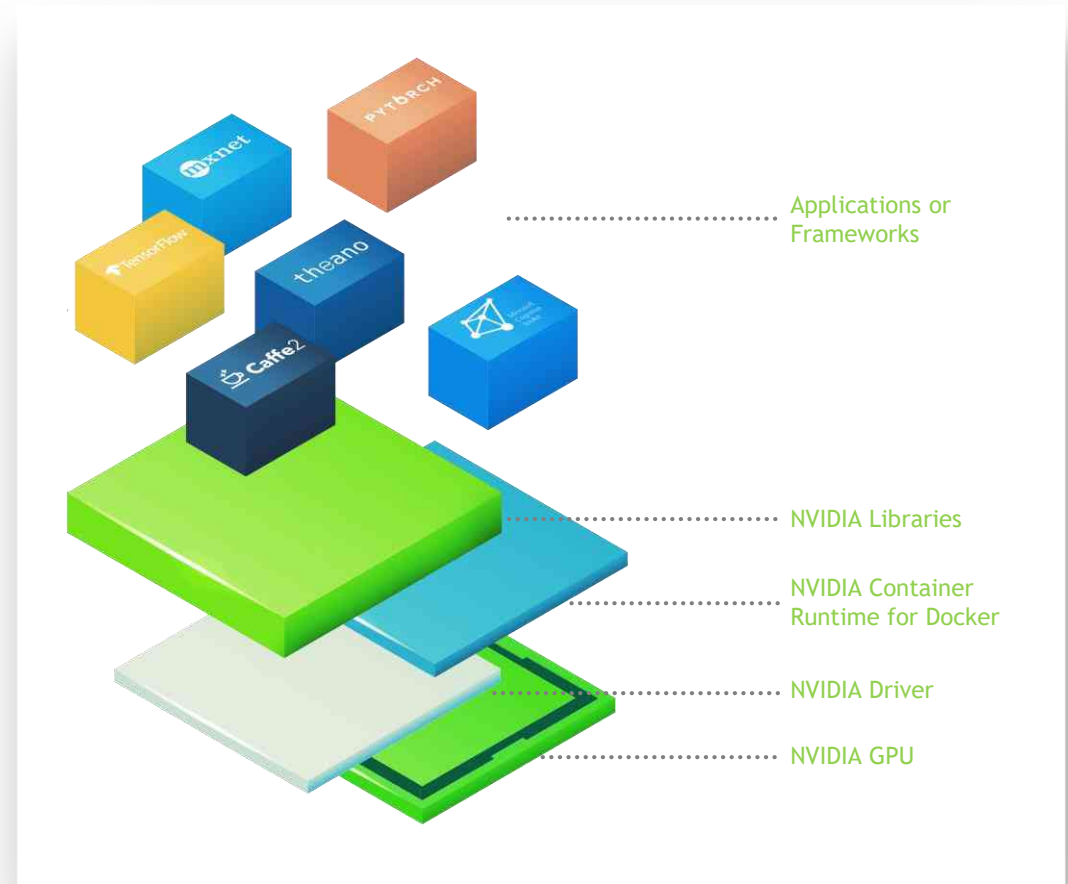
- Large collection of envs and highly fragmented by versions and features of frameworks
- Unable to cope with framework rel-cycles and newer pre-trained models/papers
- Exploding home directory and need to do have LDAP management
- The fear of `$sudo apt-get update`
- Data scientists becoming pro-bono FTE system admins!

# 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 (NGC)

## Simple Access to GPU-Accelerated Software

### Discover 35 GPU-Accelerated Containers

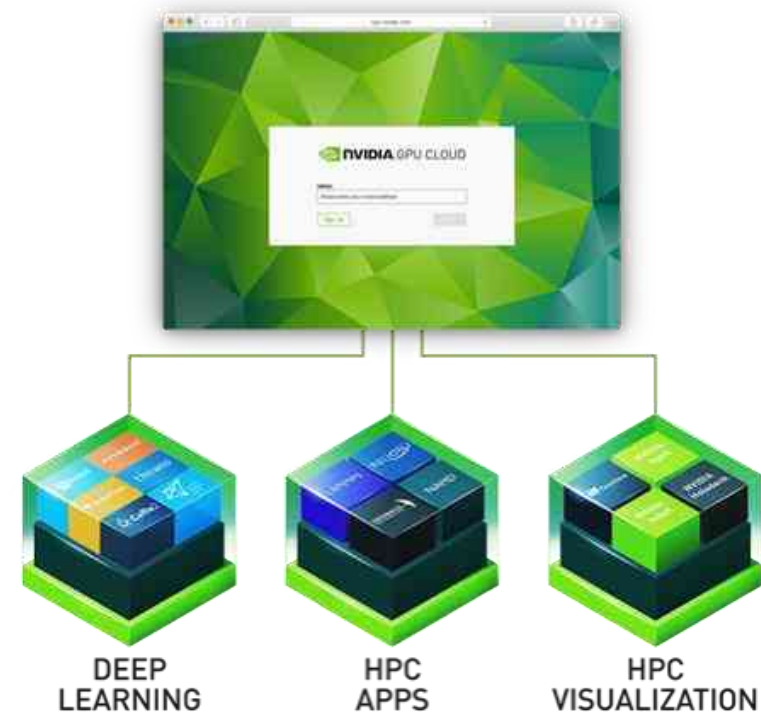
Deep learning, HPC applications, HPC visualization tools, and partner applications

### Innovate in Minutes, Not Weeks

Pre-configured, ready-to-run

### Run Anywhere

The top cloud providers, NVIDIA DGX Systems, and PCs and workstations with NVIDIA Volta or Pascal™ architecture GPUs



# GPU-ACCELERATED CONTAINERS

## Innovation for Every Industry

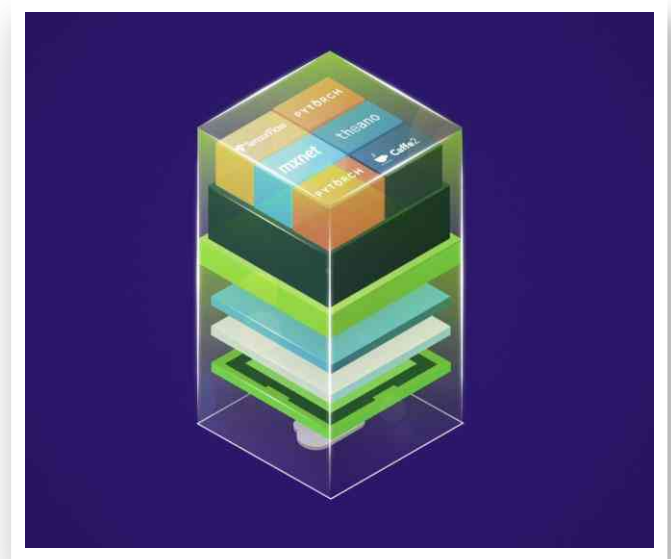
Quickly tap into the power of NVIDIA AI, from automotive, to healthcare, to fintech, and more

## Say Goodbye to DIY

GPU-accelerated software is optimized and pre-configured in ready-to-run containers

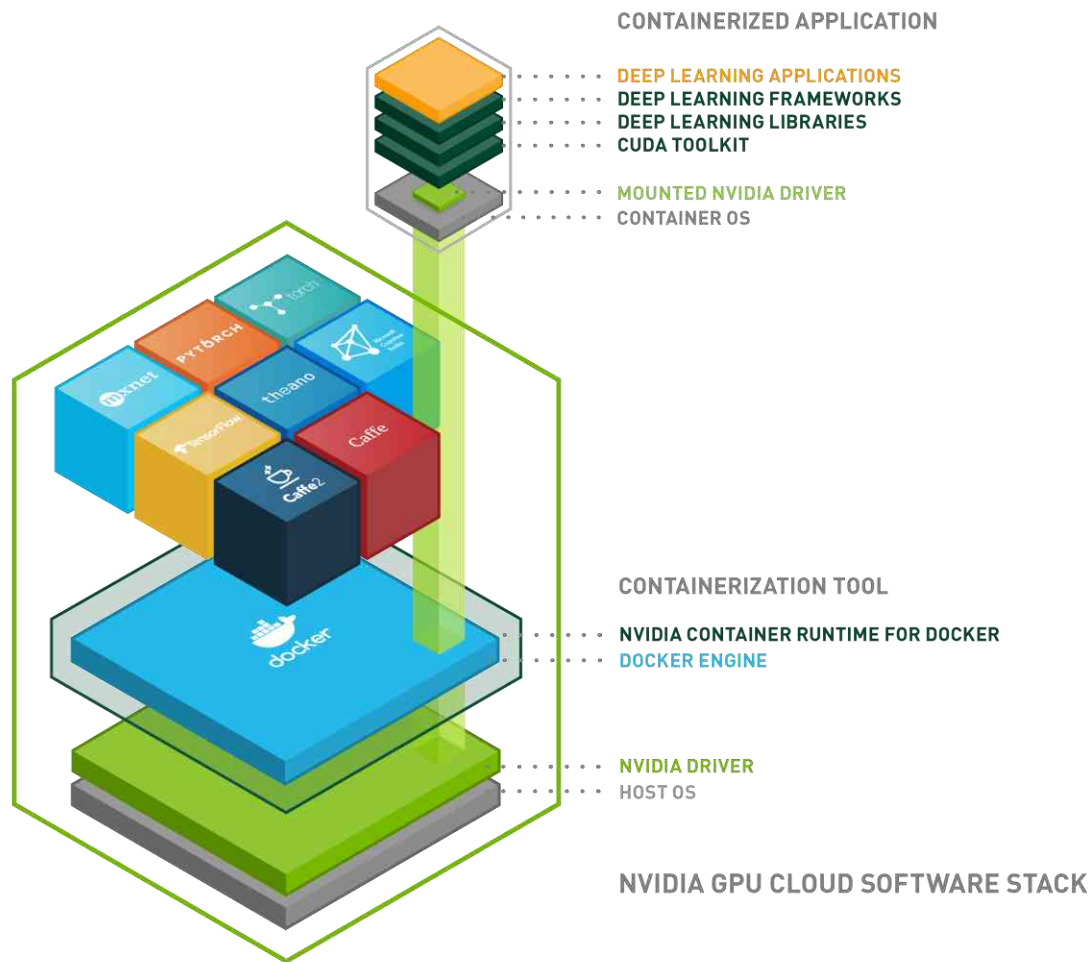
## Stay Up To Date

Monthly updates to supported deep learning containers



**NVIDIA GPU Cloud** integrates GPU-optimized software, runtimes, libraries, and OS into a ready-to-run container, available at no charge





# WHY CONTAINERS?

## Benefits of Containers:

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

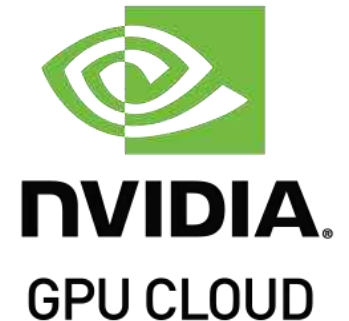
Isolate individual deep learning frameworks and applications

Share, collaborate, and test applications across different environments

# GPU-ACCELERATED CONTAINERS

10 at Launch, 35 Today

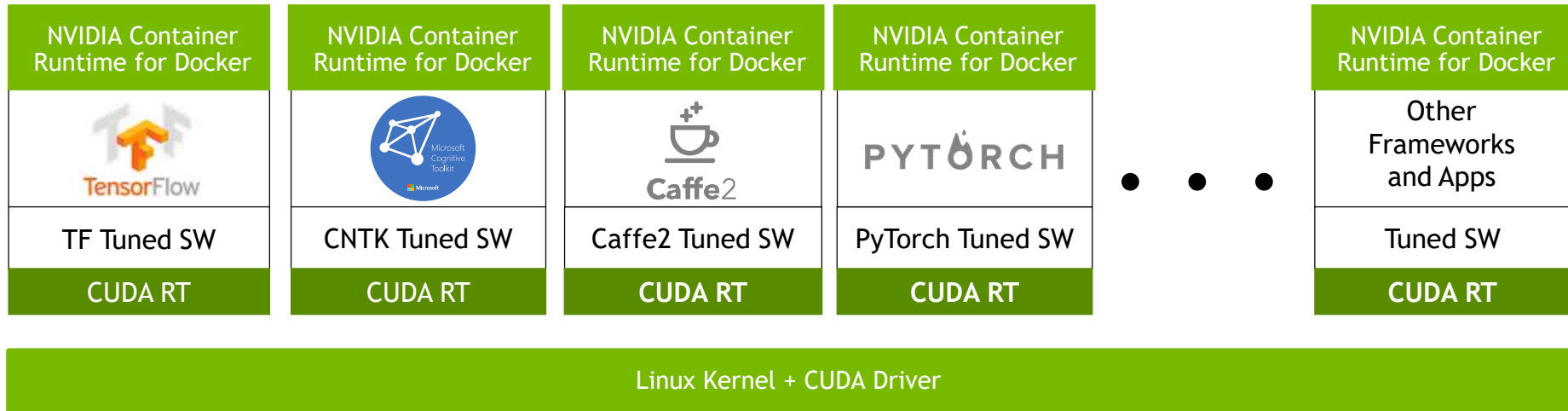
Deep Learning	HPC	HPC Visualization	NVIDIA/K8s	Partners
caffe	bigdft	index	Kubernetes on NVIDIA GPUs	chainer
caffe2	candle	paraview-holodeck		h20ai-driverless
cntk	chroma	paraview-index		kinetica
cuda	gamess	paraview-optix		mapd
digits	gromacs			Paddlepaddle
inferenceserver	lammps			MATLAB
mxnet	lattice-microbes			
pytorch	milc			
tensorflow	namd			
tensorrt	pgi			
theano	picongpu			
torch	relion			
	vmd			



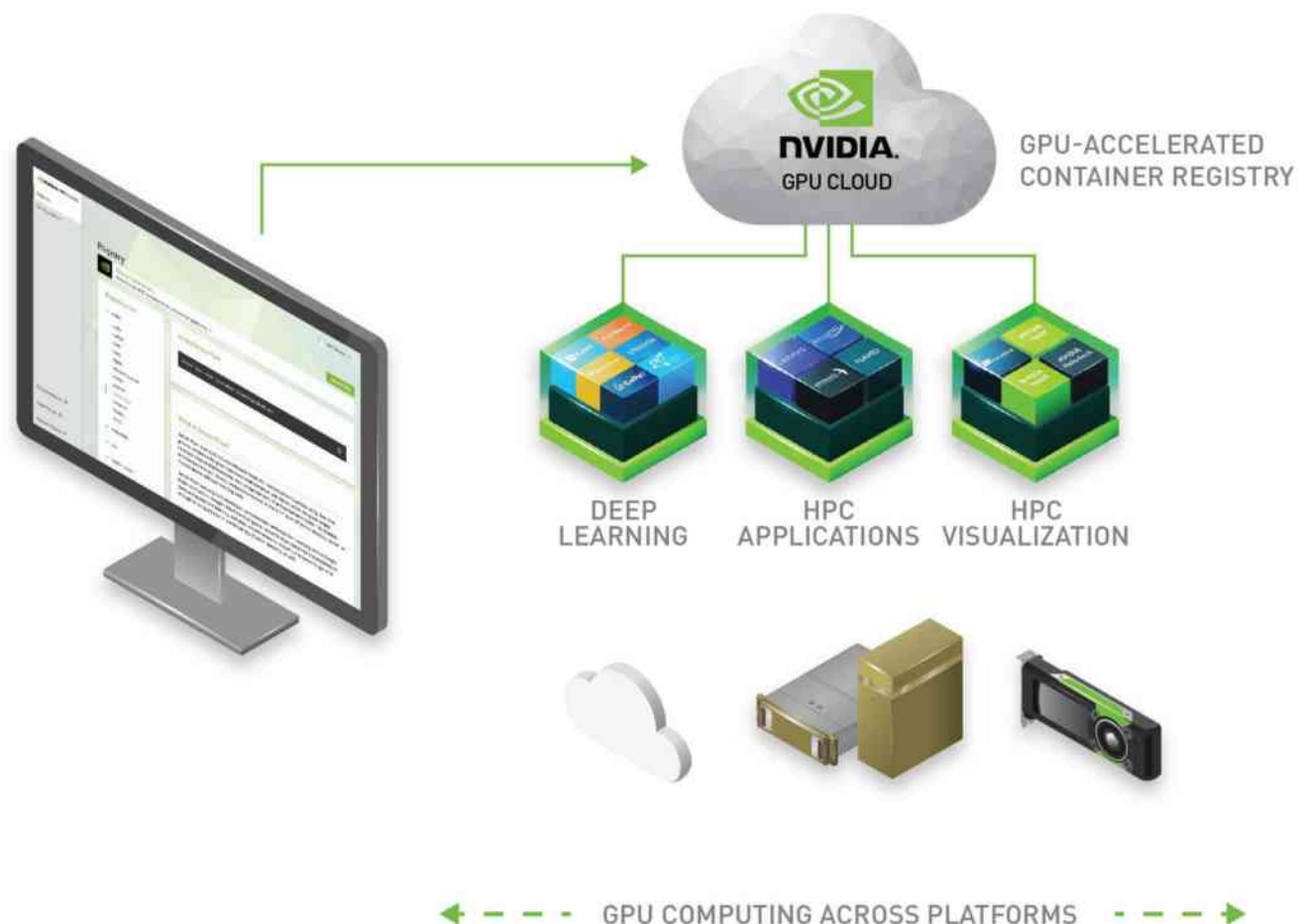
# ALWAYS OPTIMIZED AND UP-TO-DATE

NGC Deep Learning Containers are Tuned and Optimized Monthly to Deliver Maximum Performance on NVIDIA GPUs

## Containerized Applications

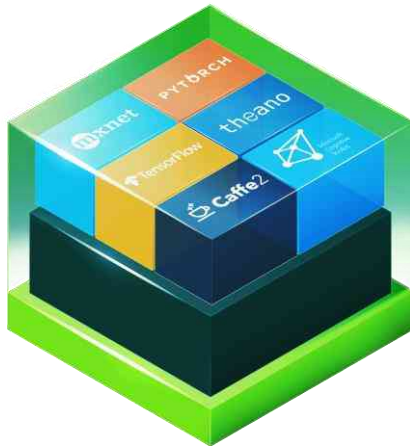


# A CONSISTENT, HYBRID CLOUD EXPERIENCE ACROSS COMPUTE PLATFORMS



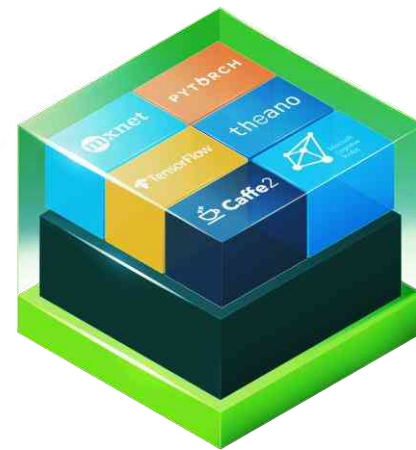
# NGC CONTAINERS ON NVIDIA GPU<sub>s</sub> IN THE CLOUD

## NGC Containers on the Top Cloud Service Providers



# WORK AT SCALE ON AI SUPERCOMPUTERS

NGC Containers Run on NVIDIA DGX Systems

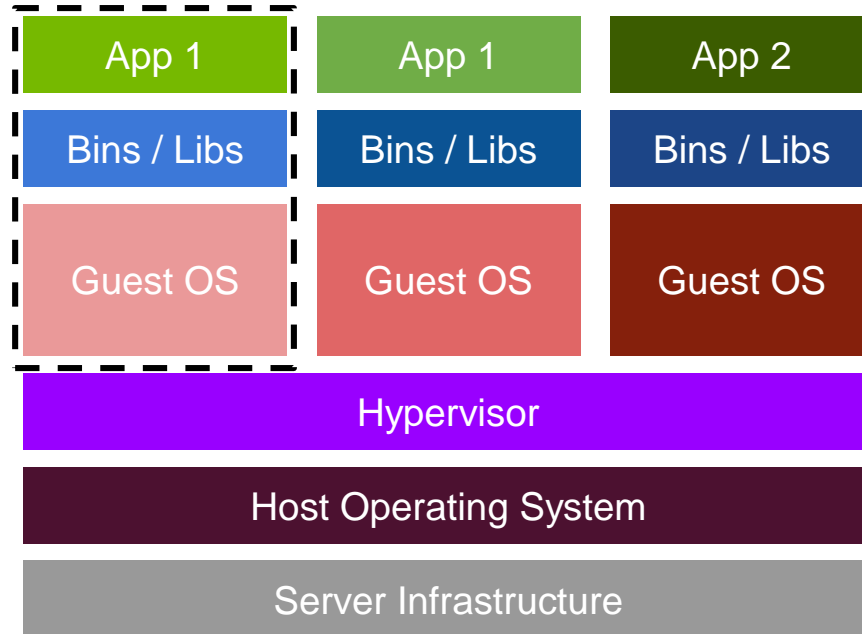




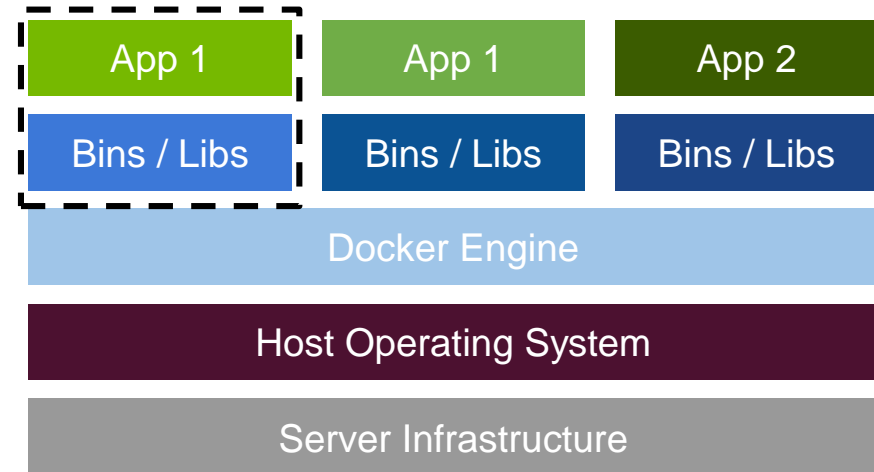
**What is this container thing you speak of?**

# Virtual Machine vs. Container

Not so similar



Virtual Machines

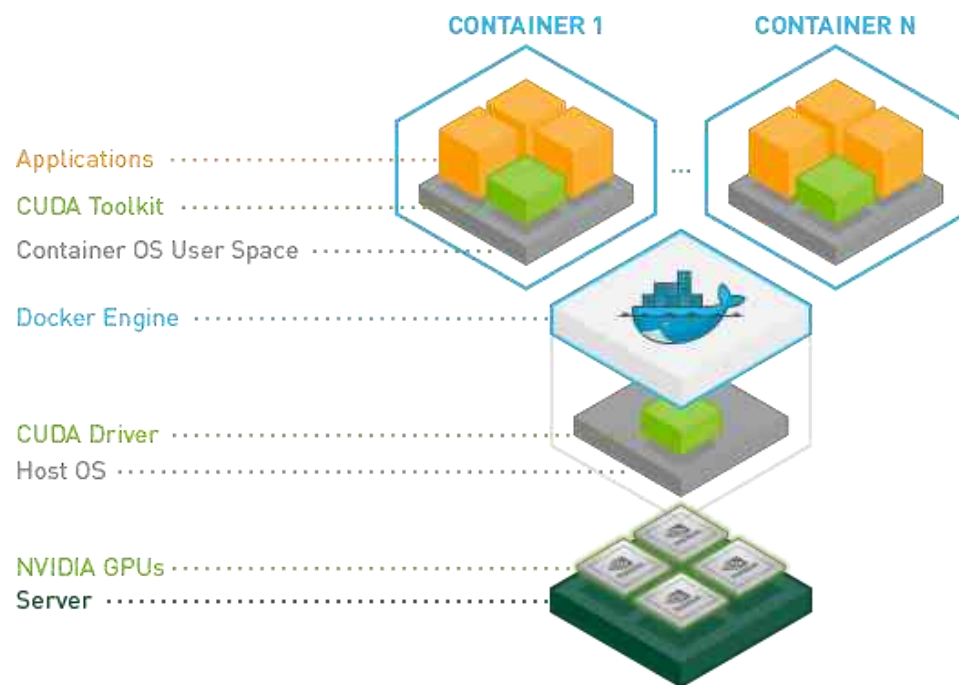


Containers

# NVIDIA container runtime

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

- Colloqually 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



# Docker Terms

## 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 and Containers

## Common Commands

### List Images:

```
docker images
```

### Remove an Image:

```
docker rmi imageID
```

```
docker rmi tensorRT
```

### List Containers:

```
docker ps -a
```

### Stop a running Container:

```
docker stop containerID
```

### Remove a Container:

```
docker rm containerID
```

Note: imageID and containerID can be either a hash or a name

# 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
```



The background is a solid green color with a subtle, abstract pattern of overlapping geometric shapes, primarily triangles and polygons, in a lighter shade of green. The pattern is more prominent on the right side of the image.

**Terminal Time!**

# TERMINAL TIME

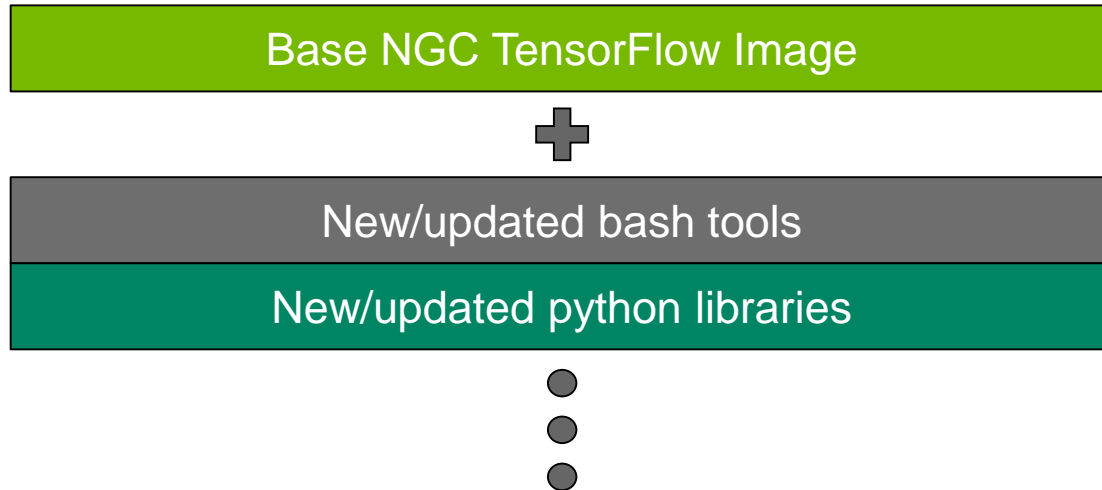
So this is the plan...

- Login in to NGC
- Understand the logistics of NGC Web UI
- Spin-up and instance on Cloud and On-prem (DGX)
- Go from spin-up to running models under 5 mins!
- Some interesting next steps

# Extending NGC Images

# Extending NGC Images

Layers layers layers



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

# 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

# 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:18.02-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 (387kB)
Collecting jupyter-console (from jupyter)
  Downloading jupyter_console-5.2.0-py2.py3-none-any.whl
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



# 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

The background is a solid green color with a subtle, abstract pattern of overlapping geometric shapes, primarily triangles and polygons, in a lighter shade of green. The pattern is more prominent on the right side of the image.

Where do we go from here?

# GET STARTED TODAY WITH NGC

Sign Up and pull containers

To learn more about all of the GPU-accelerated software from NGC, visit:

[nvidia.com/ngc](https://nvidia.com/ngc)

To sign up or explore NGC, visit:

[ngc.nvidia.com](https://ngc.nvidia.com)

