Apache MESOS™

Supporting GPUs in Docker Containers on Apache Mesos

MesosCon Europe - 2016

Kevin Klues
Senior Software Engineer
Mesosphere

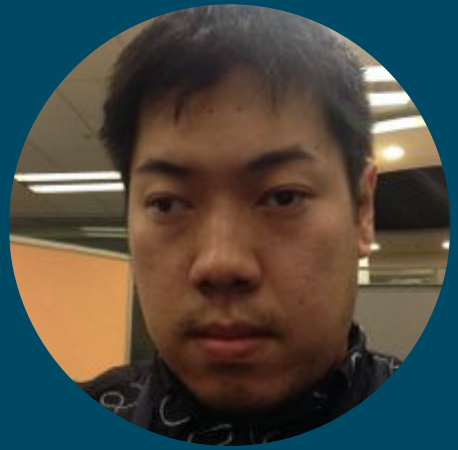
Yubo Li
Staff Researcher
IBM Research China

IBM

Kevin Klues

Kevin Klues is a Senior Software Engineer at Mesosphere working on the Mesos core team. Since joining Mesosphere, Kevin has been involved in the design and implementation of a number of Mesos's core subsystems, including GPU isolation and Pods. When not working, you can usually find Kevin on a snowboard or up in the mountains in some capacity or another.

Email: **klueska@mesosphere.com**
Slack: **@klueska**



Yubo Li

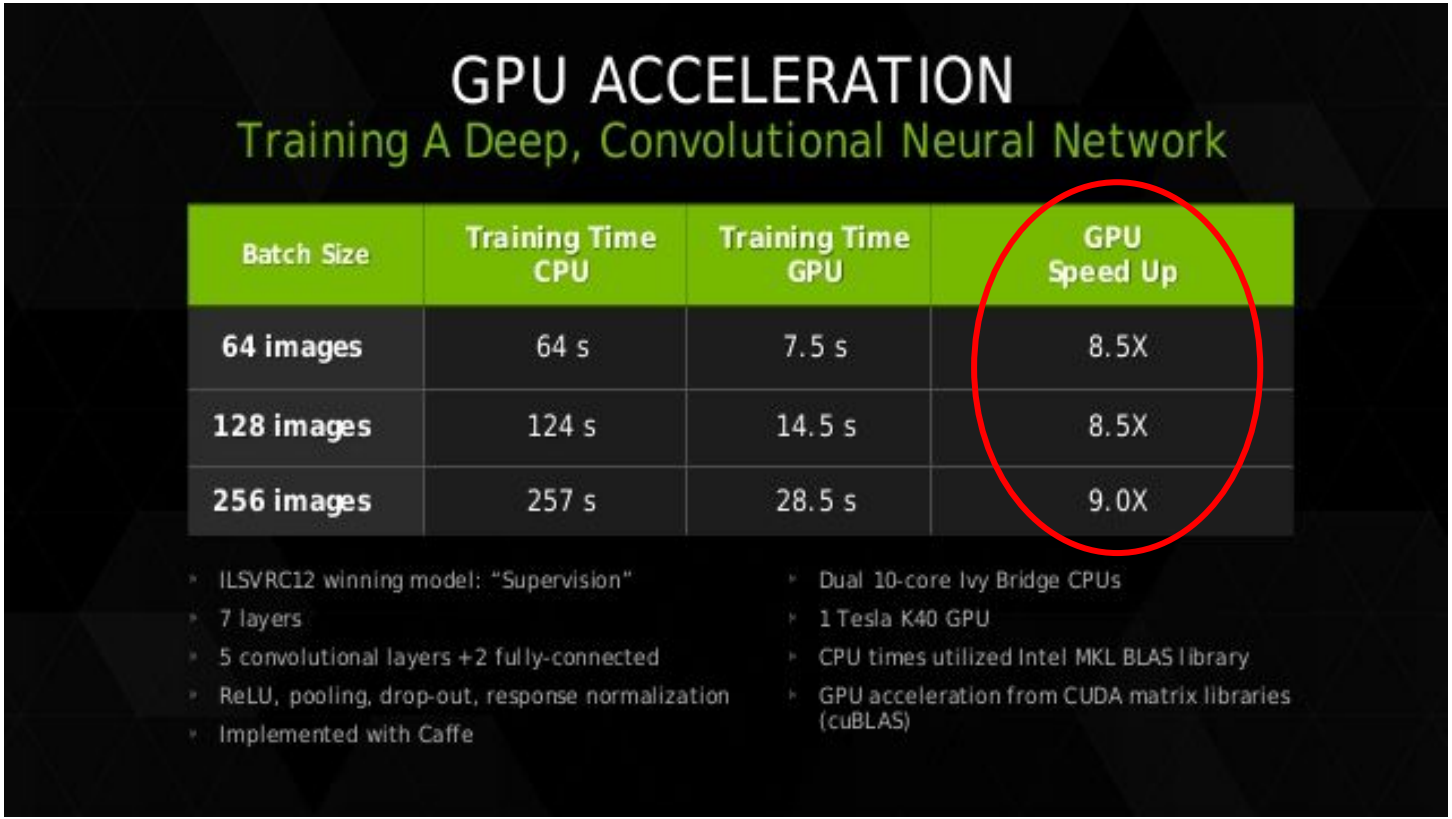
Dr. Yubo Li is a Staff Researcher at IBM Research, China. He is the architect of the GPU acceleration and deep-learning as a service (Dlaas) components of SuperVessel, an open-access cloud running OpenStack on OpenPOWER machines. He is currently working on GPU support for several cloud container technologies, including Mesos, Kubernetes, Marathon and Open Stack.

Email: **liyubobj@cn.ibm.com**
Slack: **@liyubobj**

Why GPUs?

- GPUs are the tool of choice for many big-data cloud applications
  - Deep Learning
  - Natural Language Processing
  - Genome Sequencing

Why GPUs?



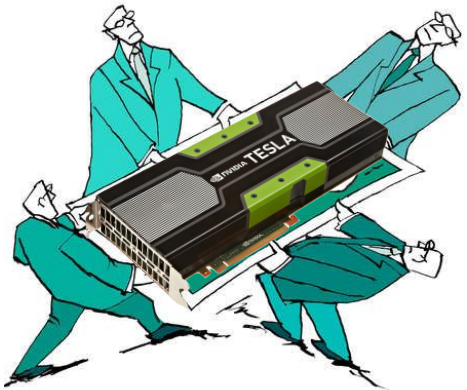
Credit: www.slideshare.net/DataScienceMD/deep-learning-with-gpus

Why GPUs?

- Mesos users have been asking for GPU support for years
  - First email asking for it can be found in the dev-list archives from 2011
  - The request rate has increased dramatically in the last 9-12 months

Why GPUs?

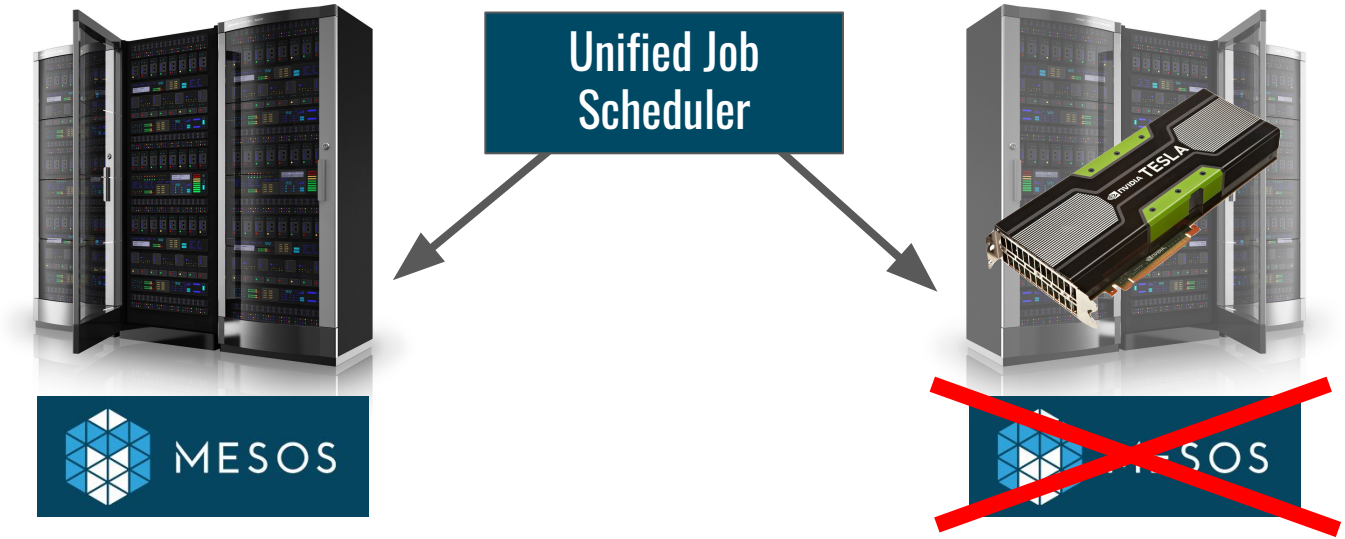
- Without direct support for GPUs, there is no isolation guarantee
  - No built-in coordination to restrict access to GPUs
  - Possible for multiple frameworks / tasks to access GPUs at the same time



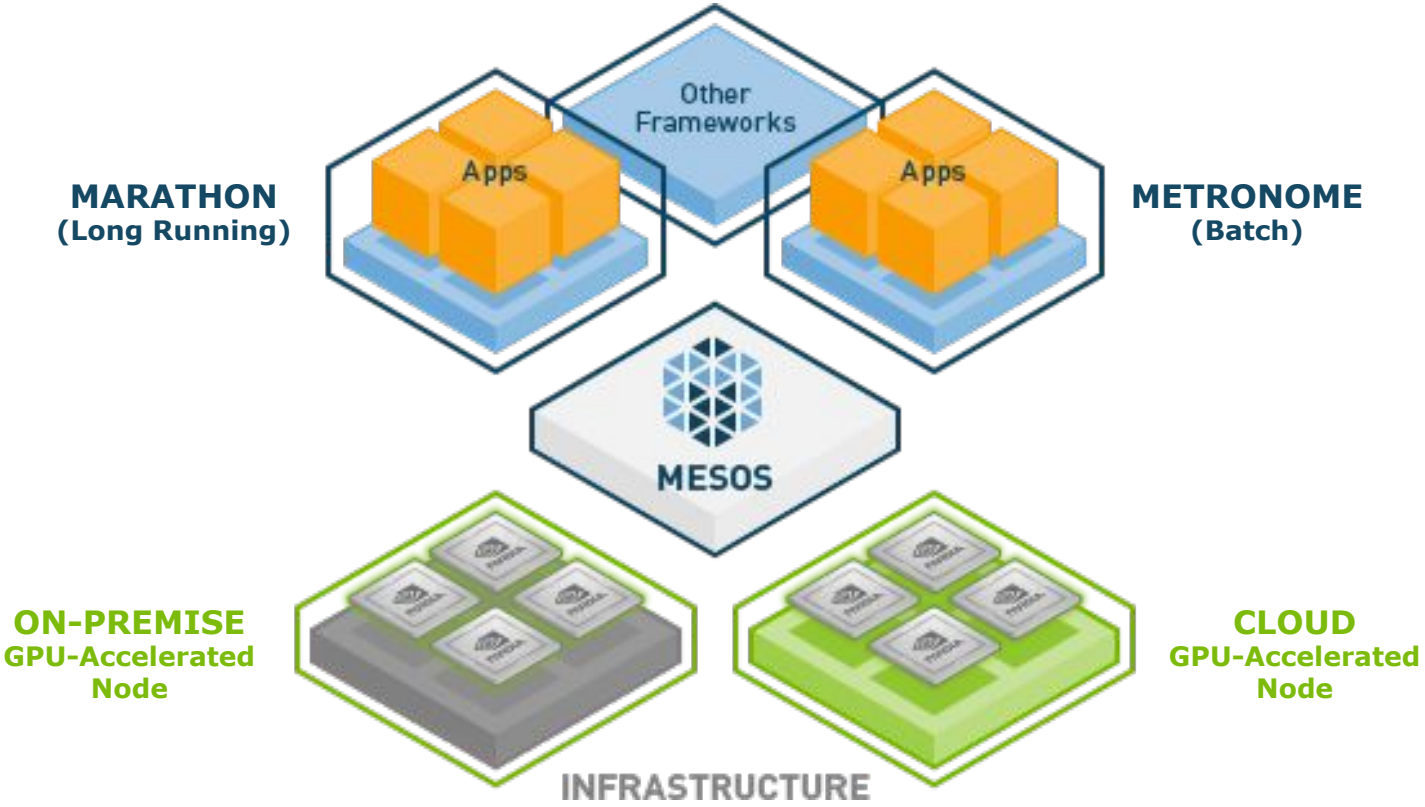
  - Ad-hoc solutions have emerged to give access to GPUs, but they all rely on co-operative scheduling rather than true isolation.

Why GPUs?

- Enterprise users currently partition clusters to make use of GPUs
  - They'd like to consolidate them
  - Not willing to do so without isolation guarantees



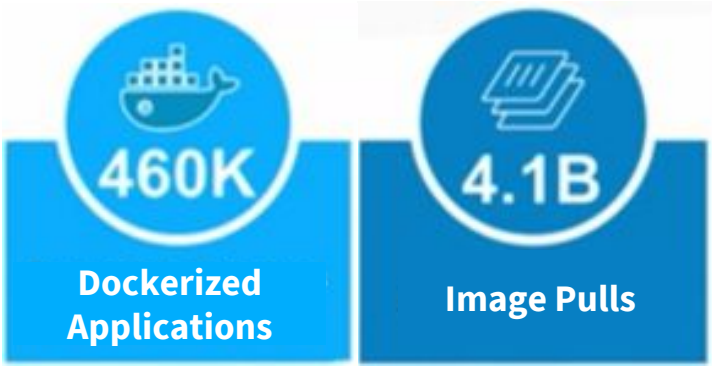
Why GPUs?



Source: www.nvidia.com/object/apache-mesos.html

Why Docker?

- Extremely popular image format for containers
  - Build once → run everywhere
  - Configure once → run anything

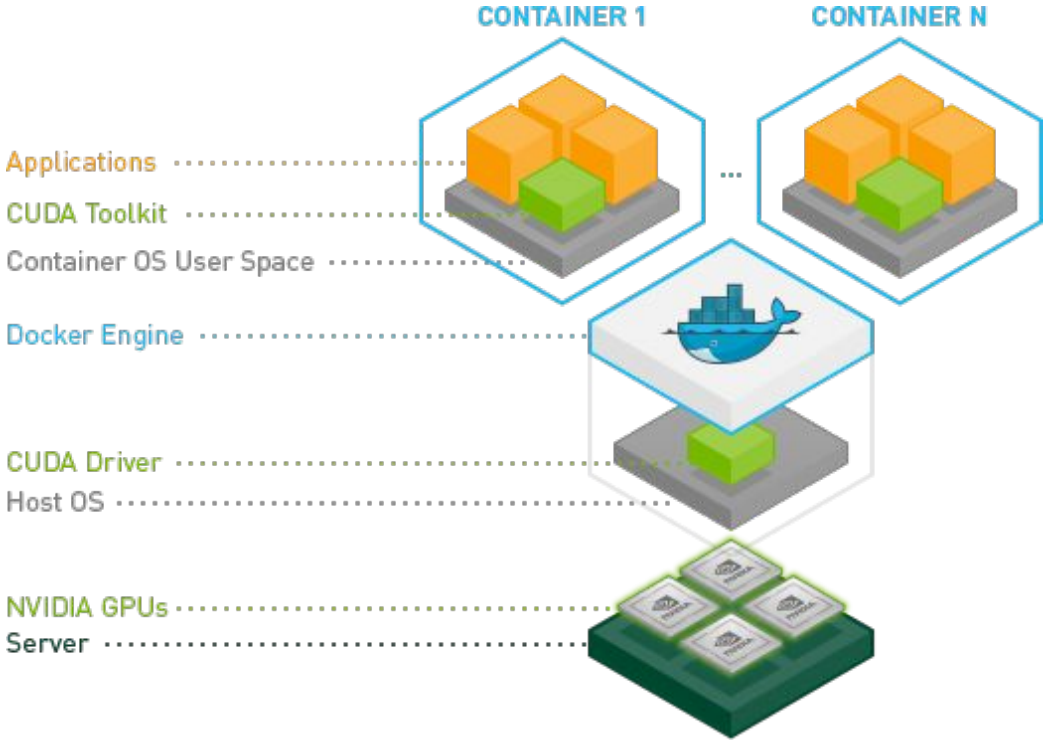


Source: DockerCon 2016 Keynote by Docker's CEO Ben Golub

Why Docker?

Nvidia-docker

Wrapper around docker to allow GPUs to be used inside docker containers

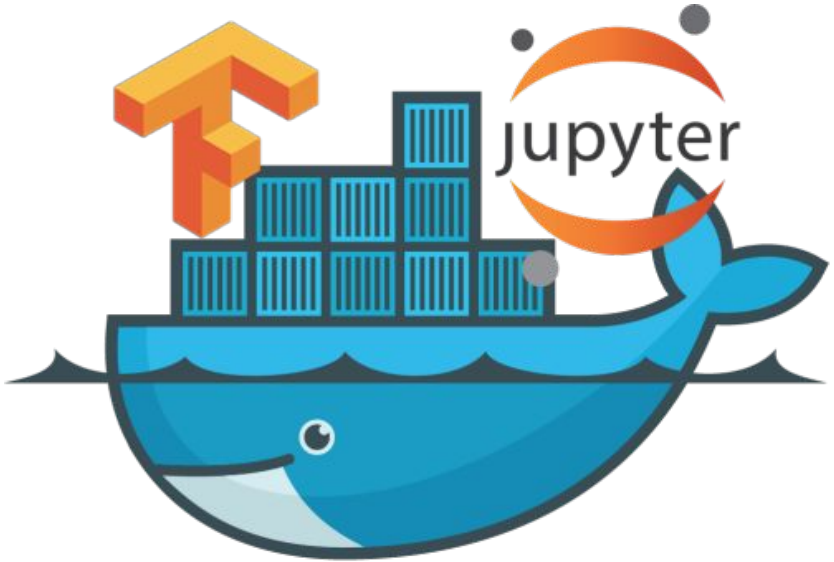


Source: https://devblogs.nvidia.com/parallelforall/nvidia-docker-gpu-server-application-deployment-made-easy/

Why Docker?

Machine Learning Frameworks

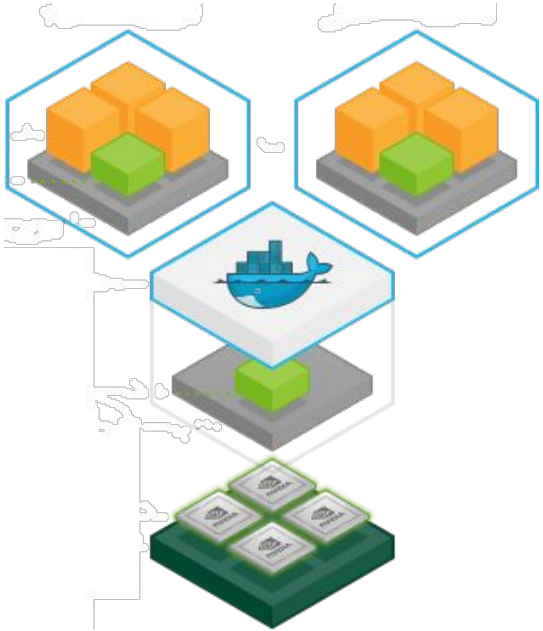
Support exists for many popular machine learning frameworks with nvidia-docker (including TensorFlow, Caffe, CNTK, etc.)



Source: https://data-shaker.com/docker-tensorflow-with-jupyter-notebook-on-windows/

Overall Goal

Test locally with nvidia-docker



Deploy to production with Mesos

Overview of Talk

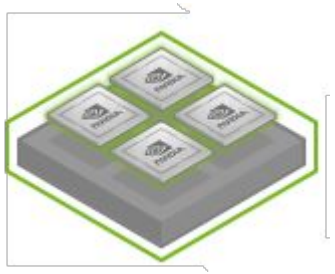
- Challenges of supporting Nvidia GPUs in docker containers
- How **nvidia-docker** addresses these challenges
- How **Apache Mesos** addresses these challenges
- Isolation Demo
- Demo from IBM

Challenges of Supporting Nvidia GPUs in Docker containers

- Before containers it was easy

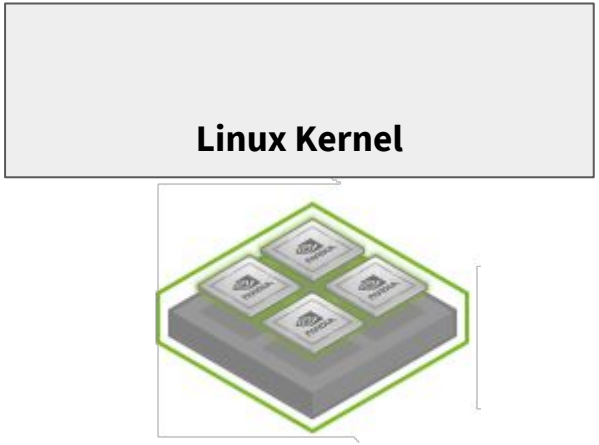
Challenges of Supporting Nvidia GPUs in Docker containers

- Before containers it was easy
  - **Buy some GPUs**



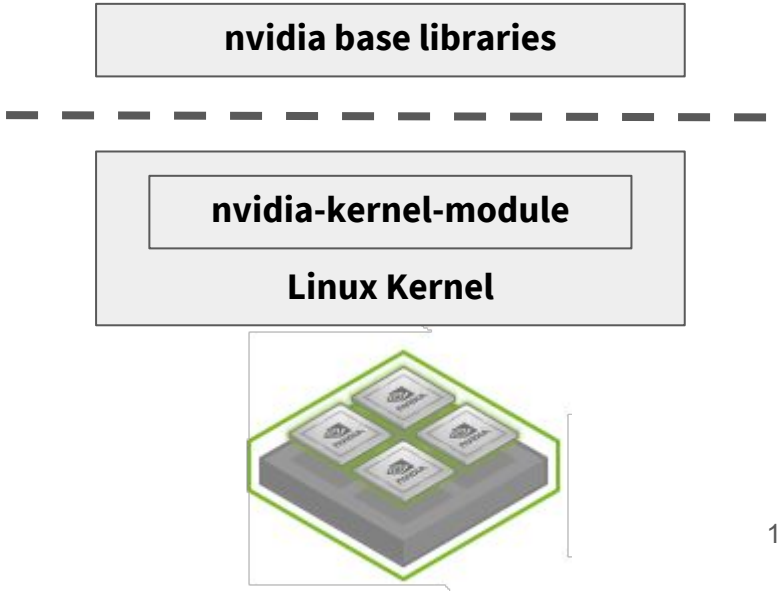
Challenges of Supporting Nvidia GPUs in Docker containers

- Before containers it was easy
  - Buy some GPUs
  - **Install them on your box**



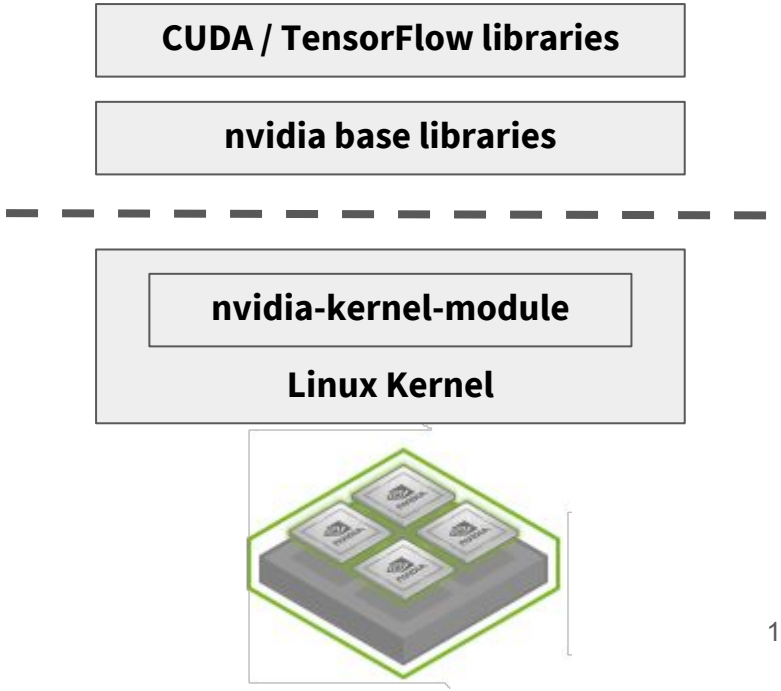
Challenges of Supporting Nvidia GPUs in Docker containers

- Before containers it was easy
  - Buy some GPUs
  - Install them on your box
  - **Install the base nvidia drivers**



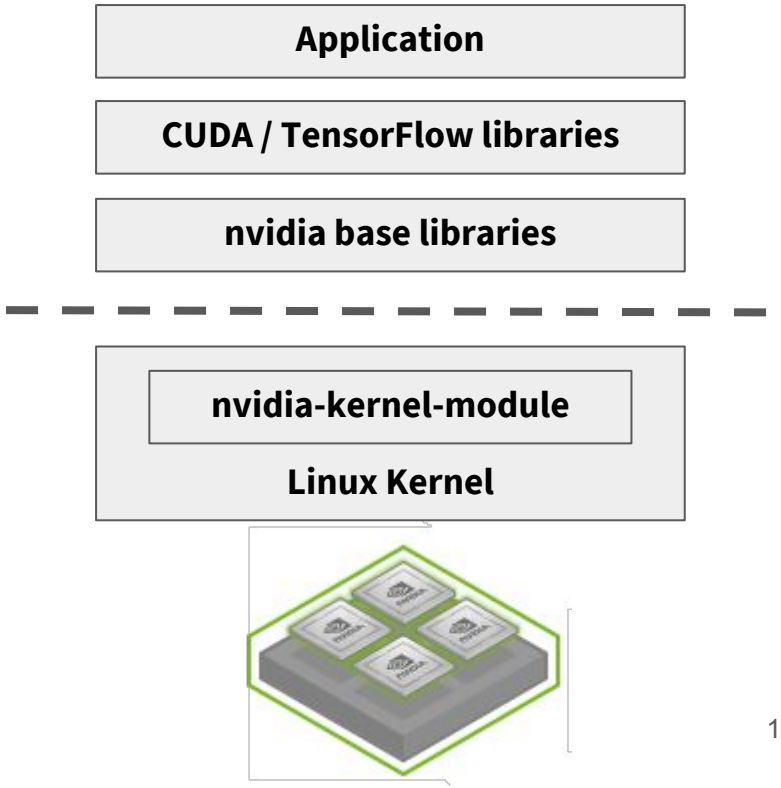
Challenges of Supporting Nvidia GPUs in Docker containers

- Before containers it was easy
  - Buy some GPUs
  - Install them on your box
  - Install the base nvidia drivers
  - **Install some advanced toolkit libraries**



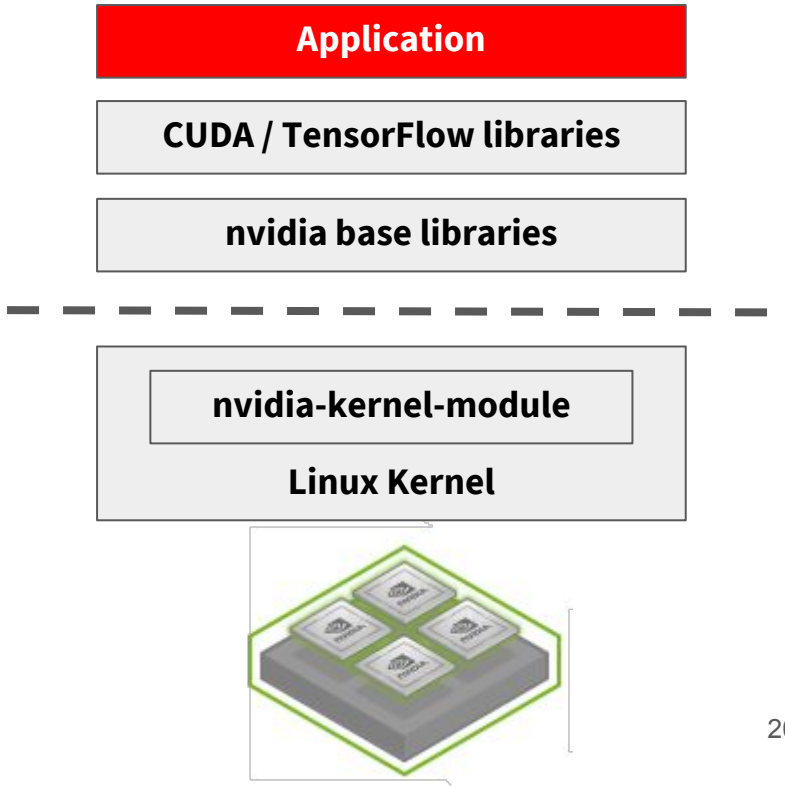
Challenges of Supporting Nvidia GPUs in Docker containers

- Before containers it was easy
  - Buy some GPUs
  - Install them on your box
  - Install the base nvidia drivers
  - Install some advanced toolkit libraries
  - **Link a GPU accelerated application against these libraries**



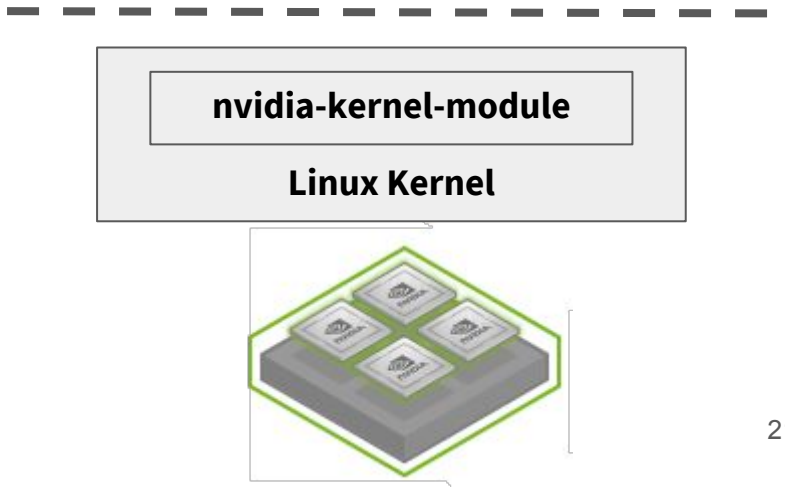
Challenges of Supporting Nvidia GPUs in Docker containers

- Before containers it was easy
  - Buy some GPUs
  - Install them on your box
  - Install the base nvidia drivers
  - Install some advanced toolkit libraries
  - Link a GPU accelerated application against these libraries
  - **Run your application**



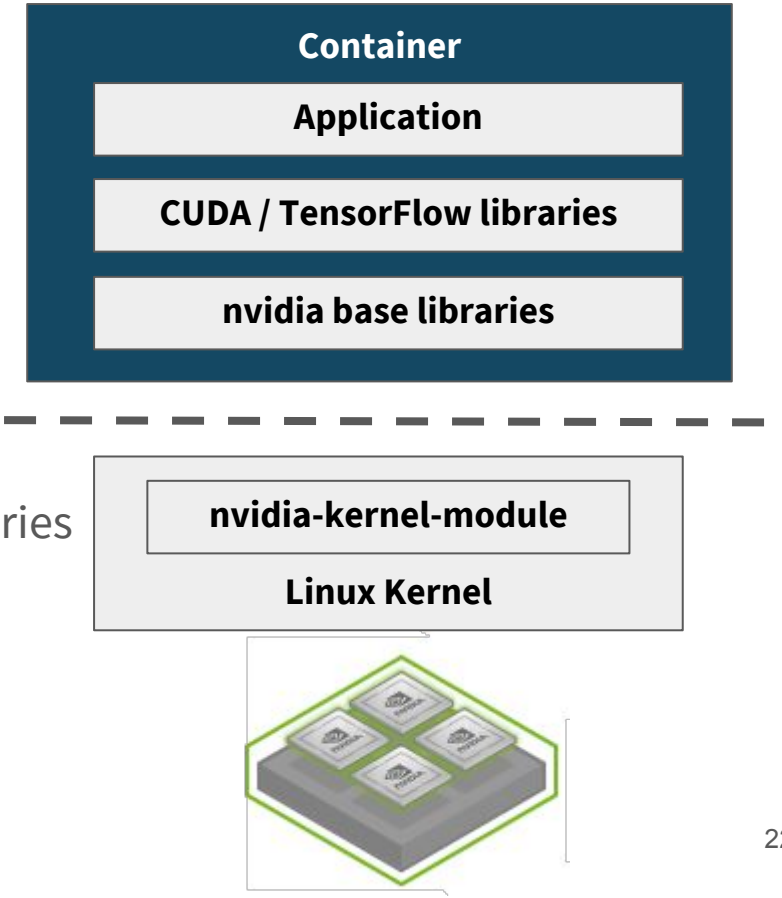
Challenges of Supporting Nvidia GPUs in Docker containers

- So what about containers?
  - Buy some GPUs
  - Install them on your box
  - Install the nvidia-kernel-modules



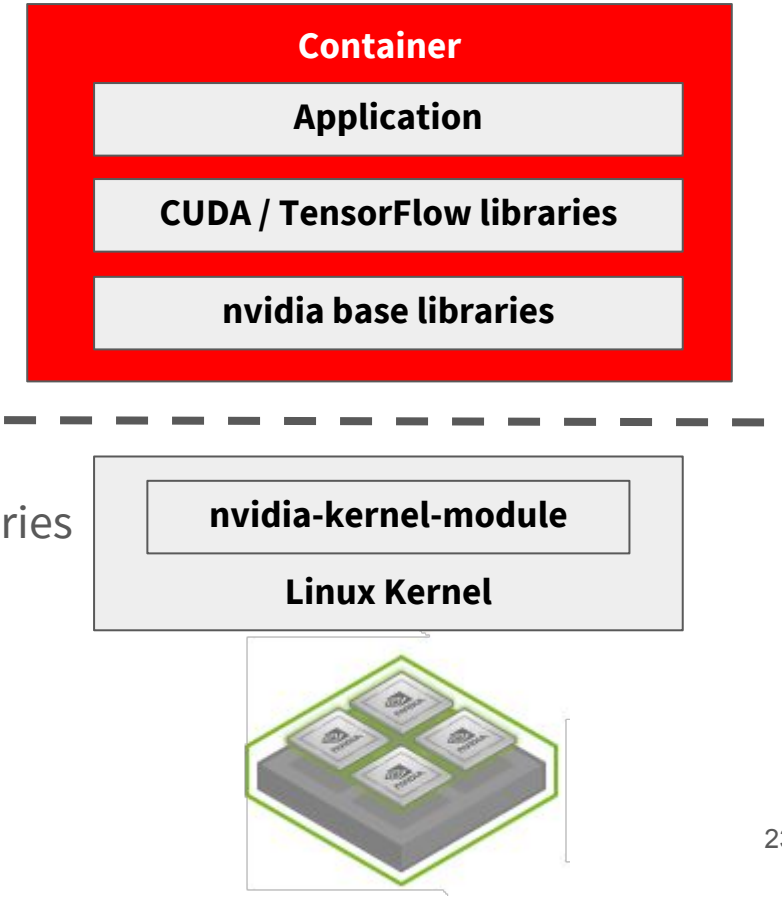
Challenges of Supporting Nvidia GPUs in Docker containers

- So what about containers?
  - Buy some GPUs
  - Install them on your box
  - Install the nvidia-kernel-module
  - **Build a docker image**
    - Bundle the base nvidia libraries
    - Bundle some advanced toolkit libraries
    - Bundle a GPU accelerated application to use these libraries



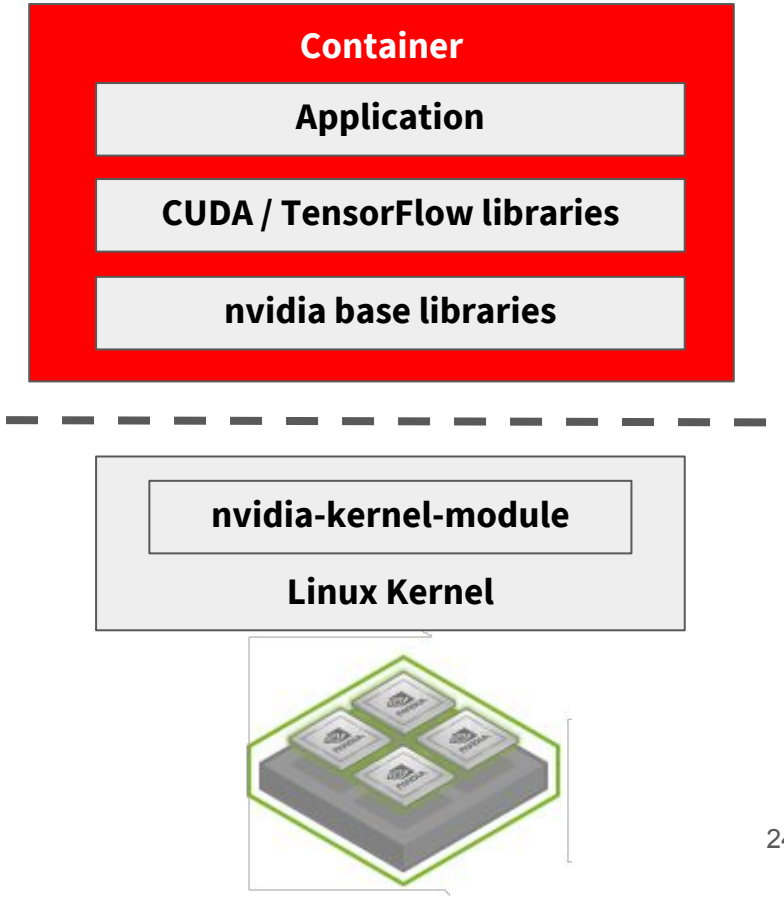
Challenges of Supporting Nvidia GPUs in Docker containers

- So what about containers?
  - Buy some GPUs
  - Install them on your box
  - Install the nvidia-kernel-module
  - Build a docker image
    - Bundle the base nvidia libraries
    - Bundle some advanced toolkit libraries
    - Bundle a GPU accelerated application to use these libraries
  - **Run your docker container**



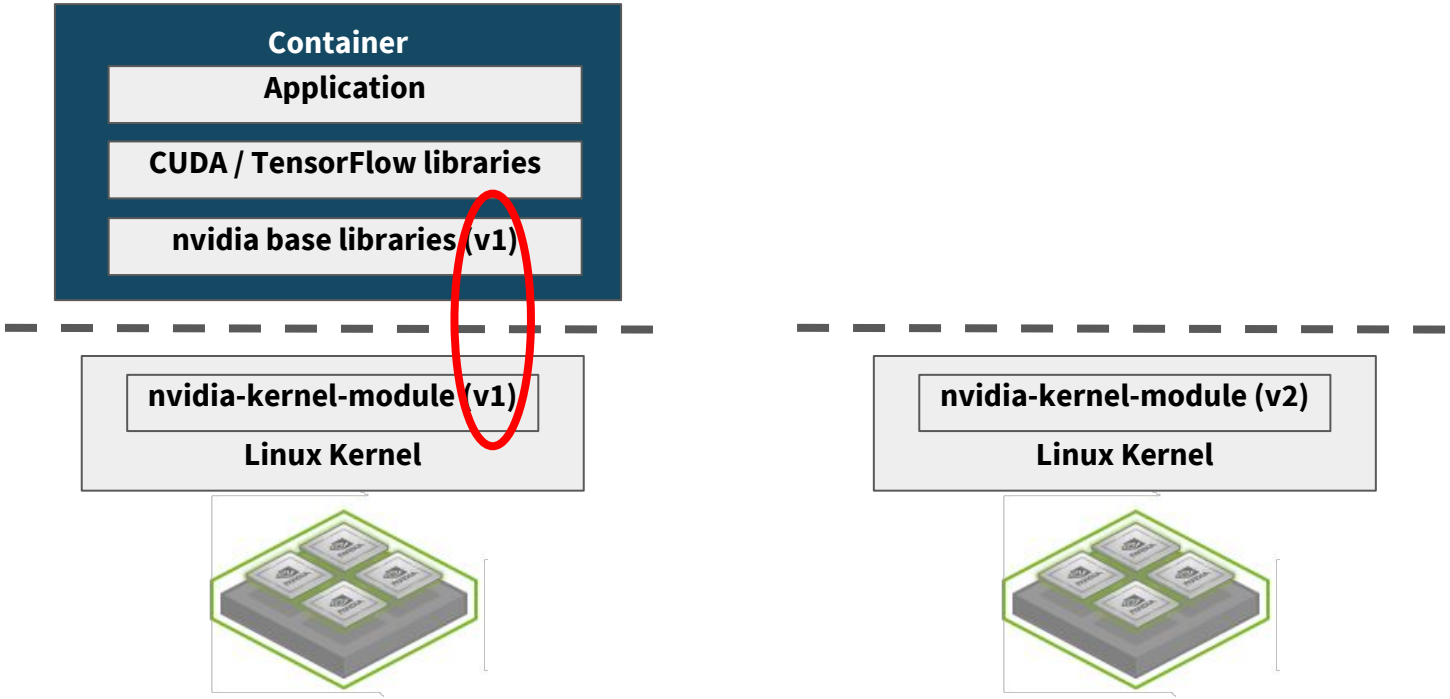
Challenges of Supporting Nvidia GPUs in Docker containers

- Straightforward, right?



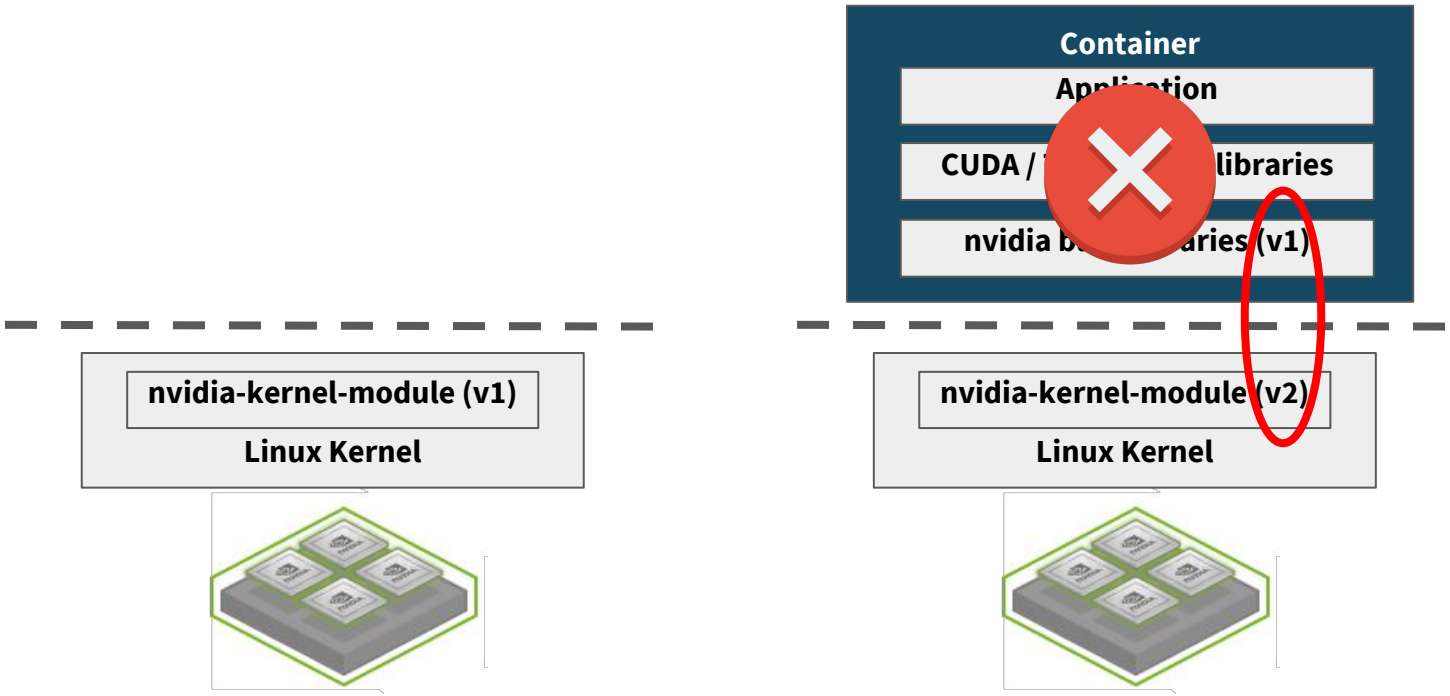
Challenges of Supporting Nvidia GPUs in Docker containers

- Will only work if the kernel / user driver versions match



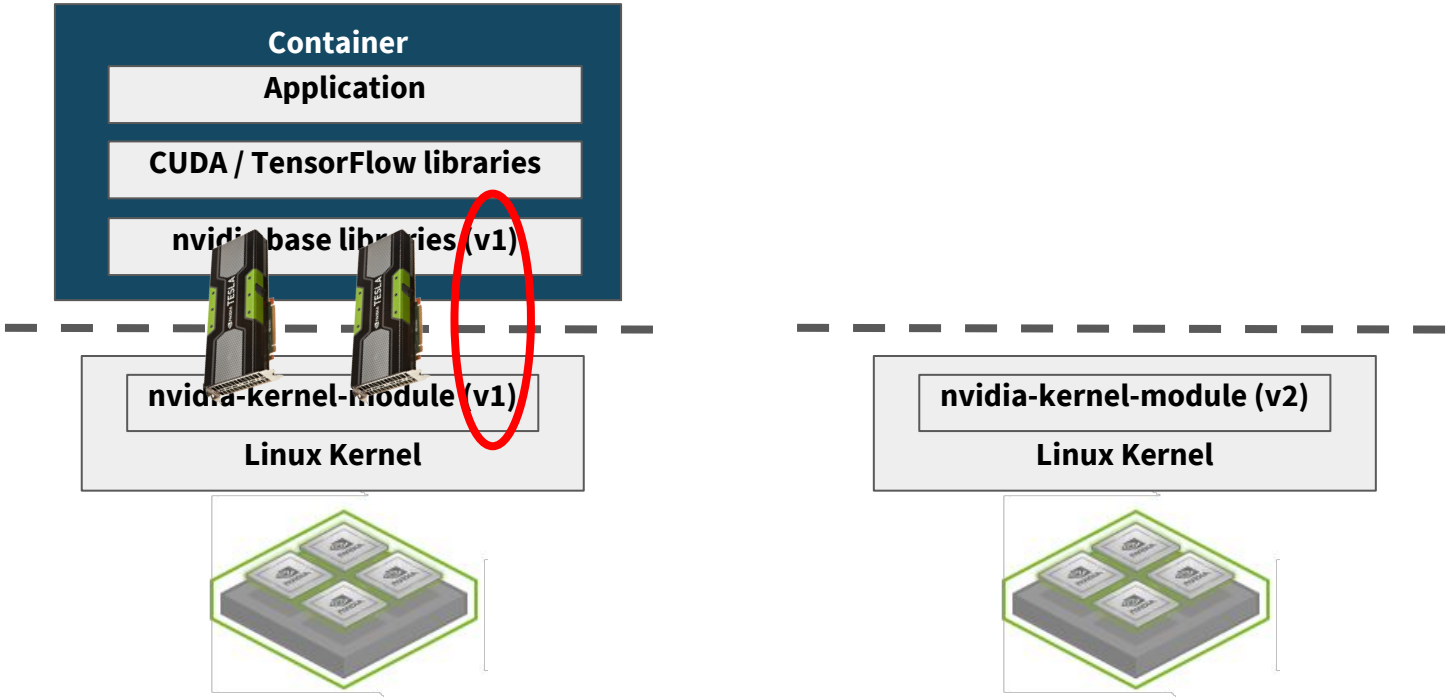
Challenges of Supporting Nvidia GPUs in Docker containers

- Won’t work if they don’t



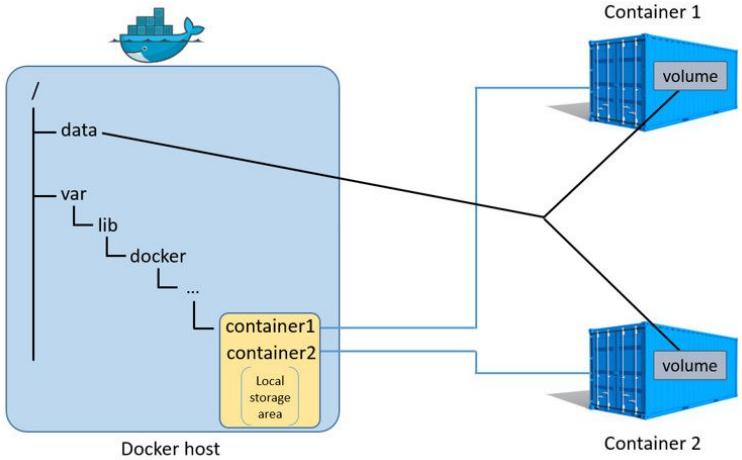
Challenges of Supporting Nvidia GPUs in Docker containers

- Either way, you have to map in the GPU devices somehow



nvidia-docker and GPUs

- Components of **`nvidia-docker`**
  - Set of docker images that set custom labels / environment variables
  - **`nvidia-docker-plugin`** (standard docker volume plugin)
  - **`nvidia-docker`** (wrapper script around **`docker`** itself)




```
docker run ...
```

```
nvidia-docker run ...
```

nvidia-docker and GPUs

- **`nvidia-docker-plugin`**

  Finds all standard nvidia libraries / binaries on the host and consolidates them into a single place as a docker volume

```
/var/lib/docker/volumes
└── nvidia_XXX.XX (version number)
    ├── bin
    ├── lib
    └── lib64
```

nvidia-docker and GPUs

- **`nvidia-docker`** wrapper script

  Looks for the label:

  `com.nvidia.volumes.needed = nvidia_driver`

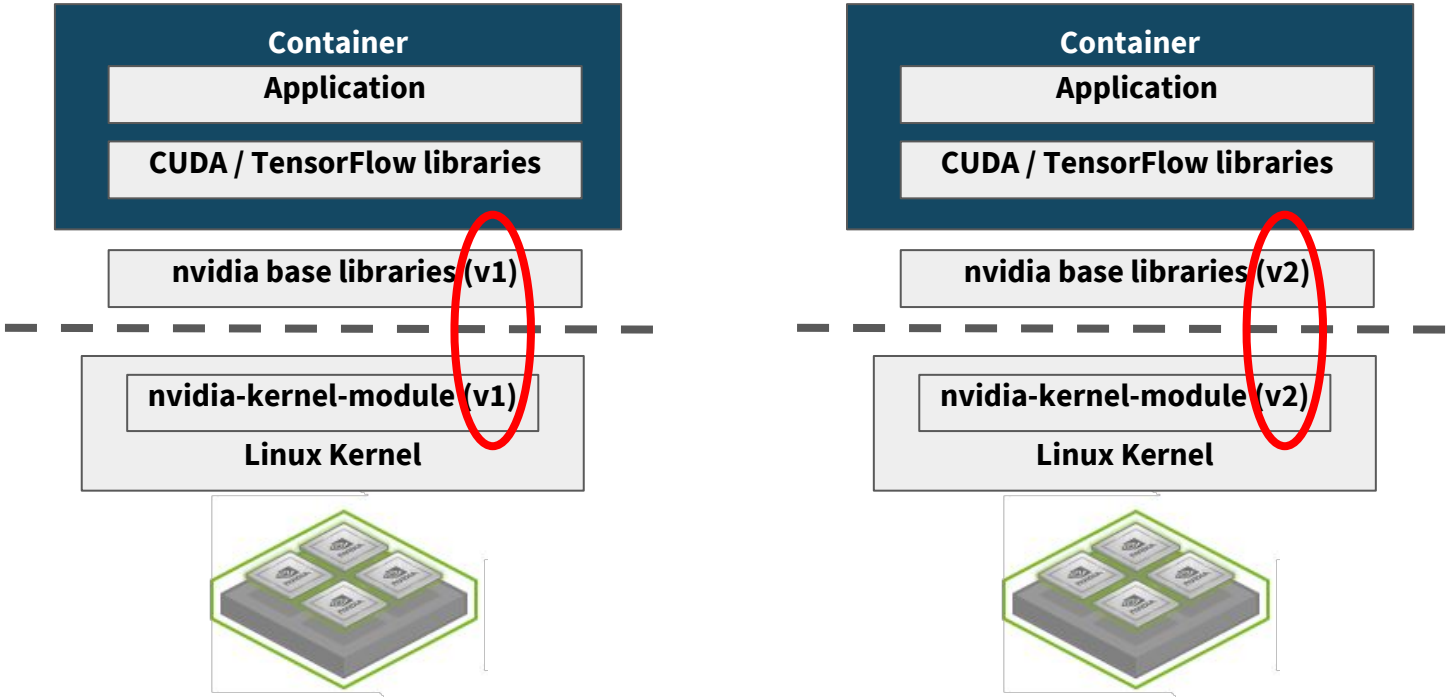
  When found, it maps the **`nvidia_XXX.XX`** volume into the container at:

  `/usr/local/nvidia`

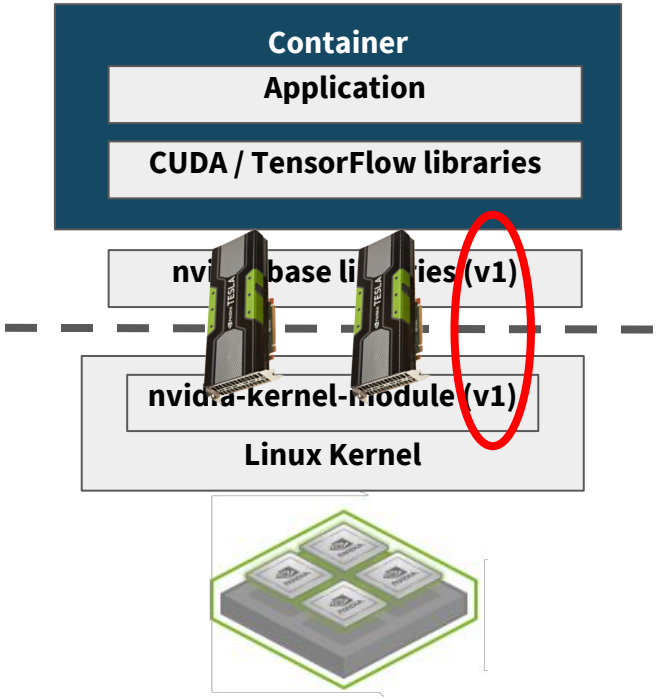
  Enumerates all GPUs on the machine and maps them into the container as available devices

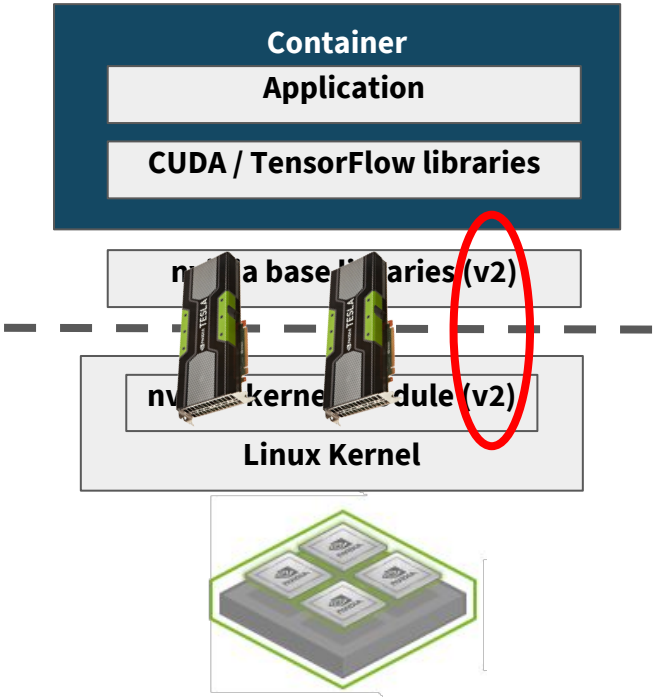
  Passes all other docker options straight through to **`docker`**

nvidia-docker and GPUs



nvidia-docker and GPUs





Apache Mesos and GPUs

- Mimics functionality of **`nvidia-docker`**
  - Supports nvidia docker images with custom labels
  - Maps consolidated volume of binaries / libraries into **`/usr/local/nvidia`**
  - Enumerates GPUs and injects them into containers

- Additionally isolates access to GPUs between tasks
  - Not possible with **`nvidia-docker`** alone

Apache Mesos and GPUs

- Multiple Containerizer support
  - Mesos (aka unified) containerizer (fully supported)
  - Docker containerizer (currently being worked on)

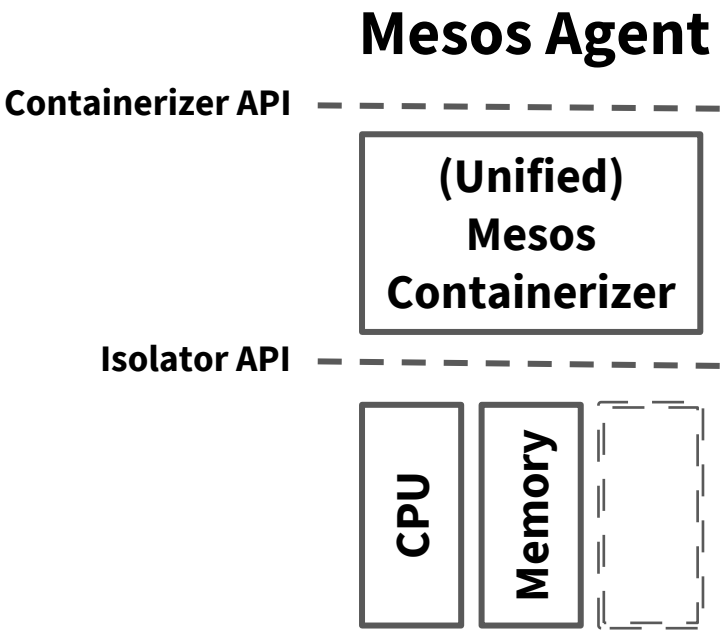
- Why support both?
  - Many people are asking for Docker containerizer support to bridge the feature gap
  - People are already familiar with existing docker tools
  - Unified containerizer needs time to mature

Apache Mesos and GPUs

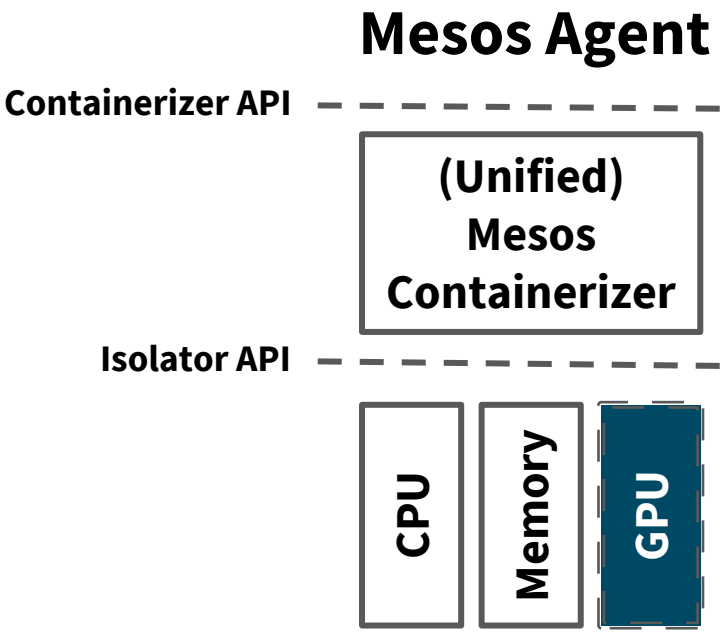
- **`GPU_RESOURCES`** framework capability
  - Frameworks must opt-in to receive offers with GPU resources
  - Prevents legacy frameworks from consuming non-GPU resources and starving out GPU jobs

- Use agent attributes to select specific type of GPU resources
  - Agents advertise the type of GPUs they have installed via attributes
  - Only accept an offer if the attributes match the GPU type you want

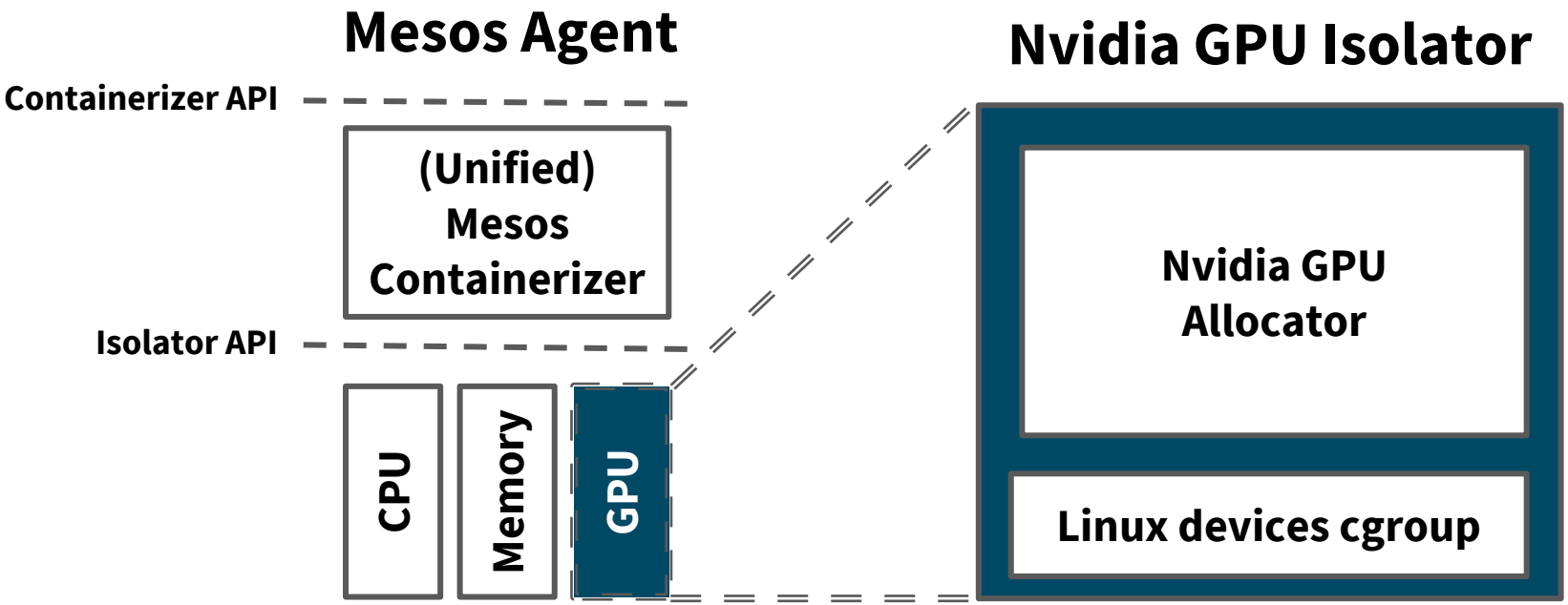
Apache Mesos and GPUs



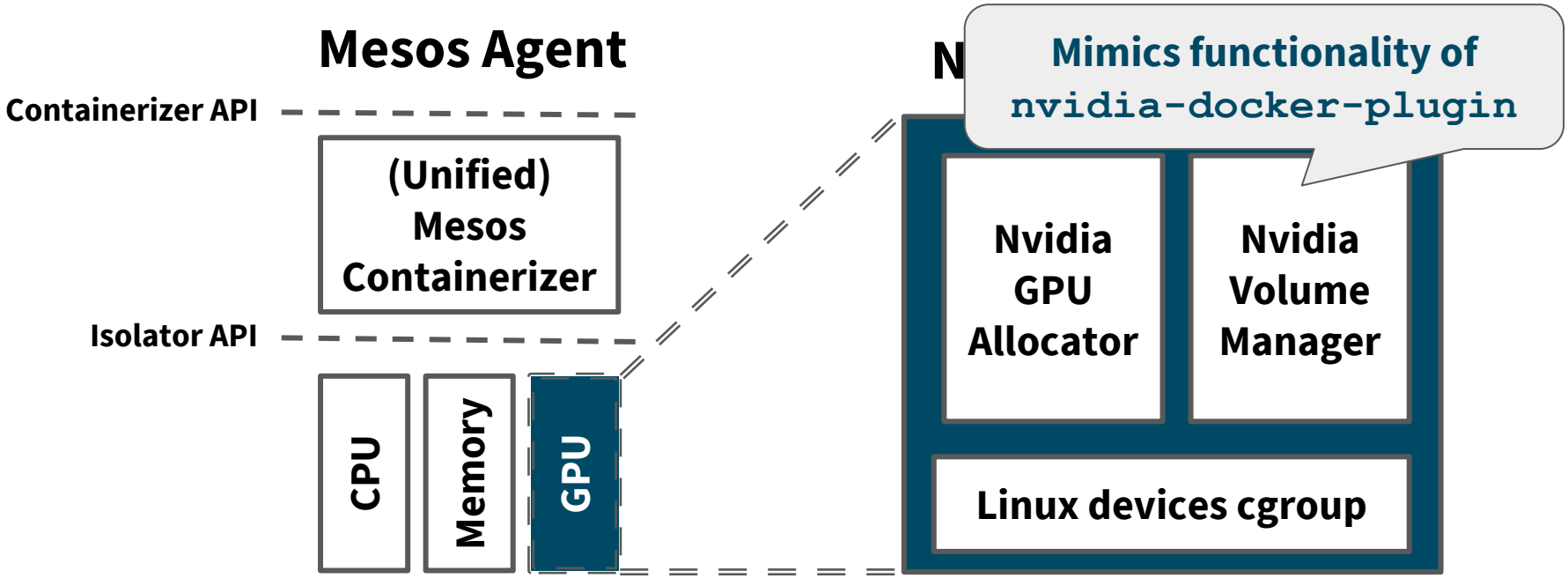
Apache Mesos and GPUs



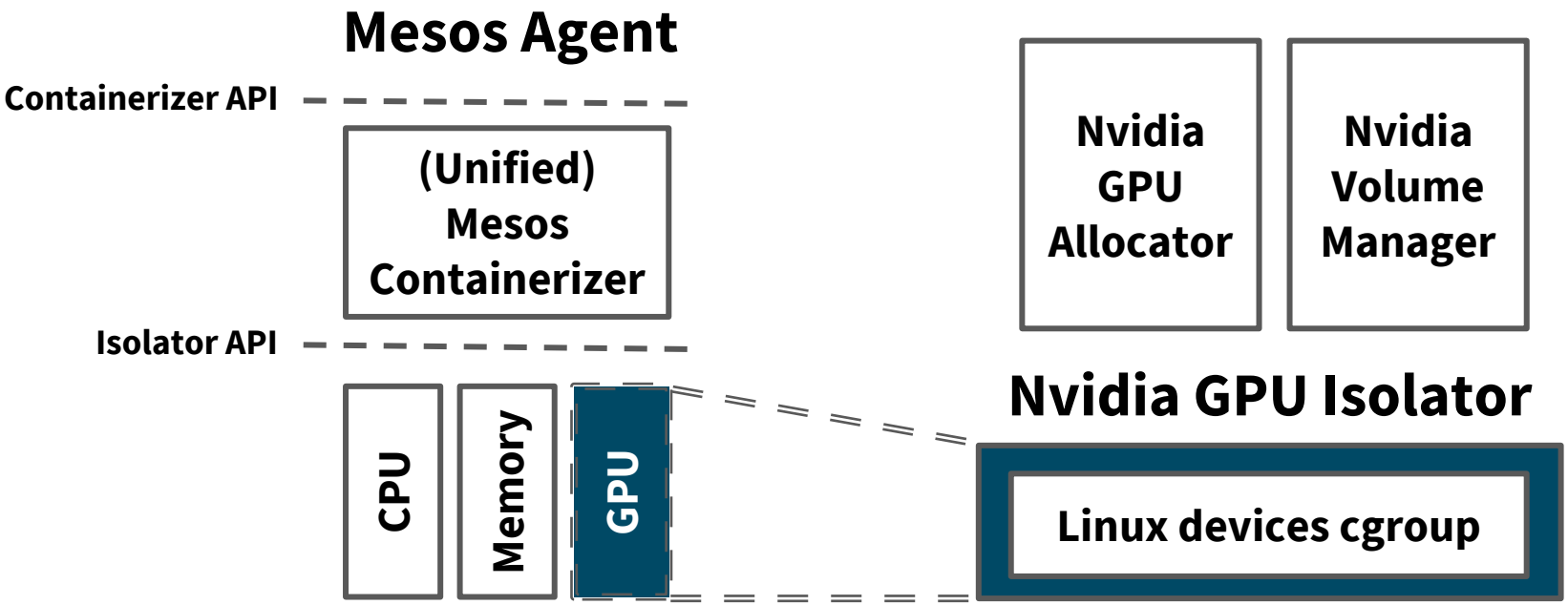
Apache Mesos and GPUs



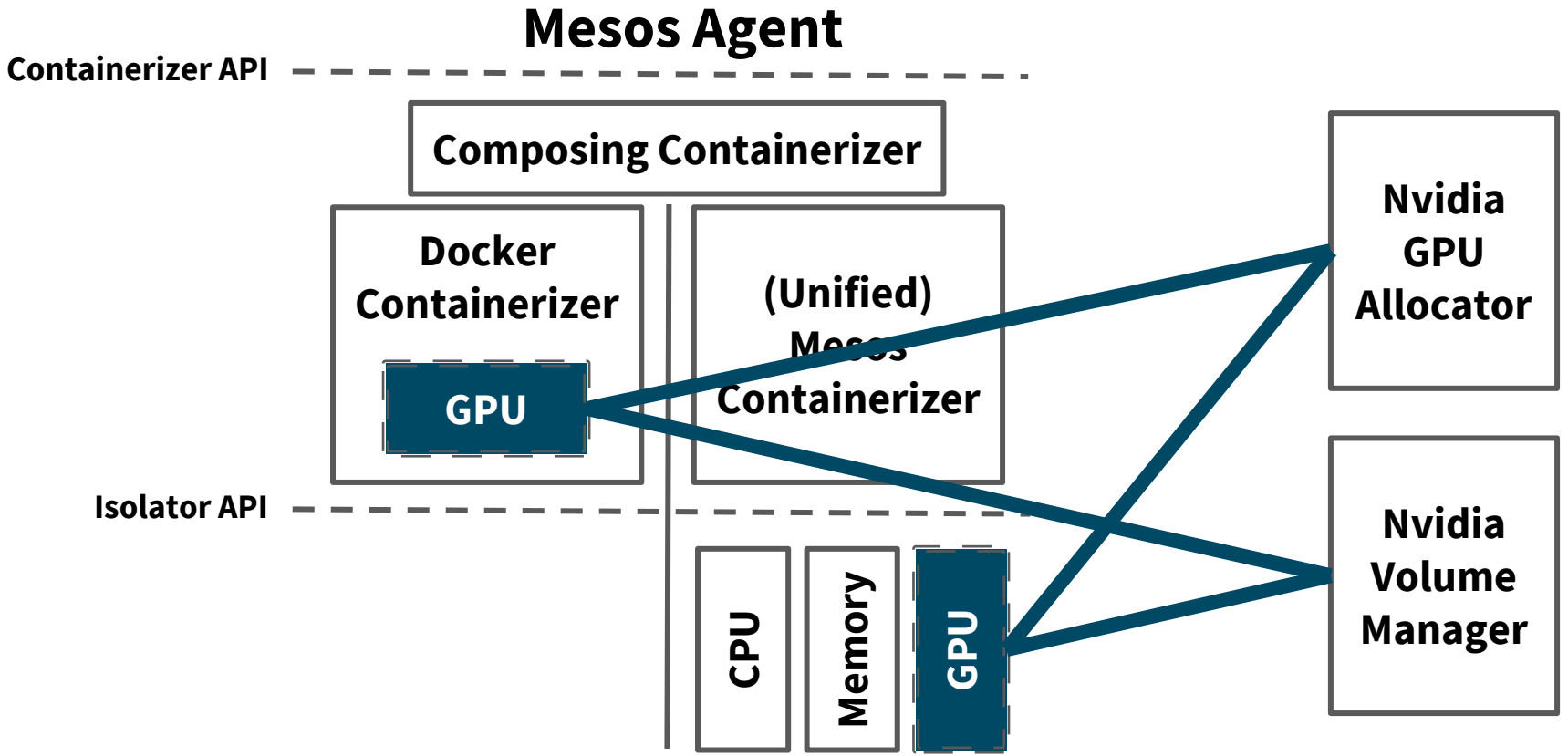
Apache Mesos and GPUs



Apache Mesos and GPUs



Apache Mesos and GPUs



Implementation Timeline

- Support for the Unified containerizer
  - Supports both image-less and docker-image based containers (fully compatible with nvidia-docker)
  - Supported in the recent Mesos 1.0.0 release (with patch fixes in 1.0.1)
  - Supported in upcoming Marathon 1.3 release (coming in next few weeks)
  - Targeted for DC/OS 1.9 release (mid October)

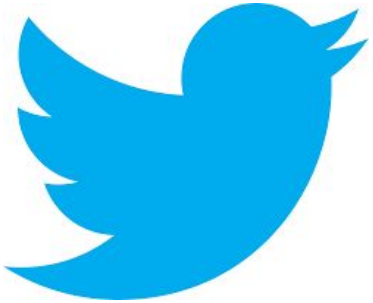
- Support for the Docker Containerizer
  - Designed and partially implemented
  - Target Mesos release: 1.1.0
  - Target Marathon release: 1.4
  - Target DC/OS release: 1.10

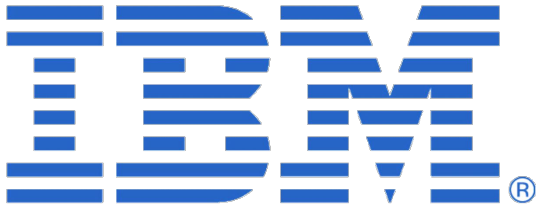
Looking forward

- Expose Underlying GPU Topology
  - Include in offers sent to schedulers
  - Help make more informed decision about which GPUs to choose

- Add support for other GPU / accelerator types
  - AMD, Intel, Custom FPGAs, etc.

Special Thanks to All Collaborators



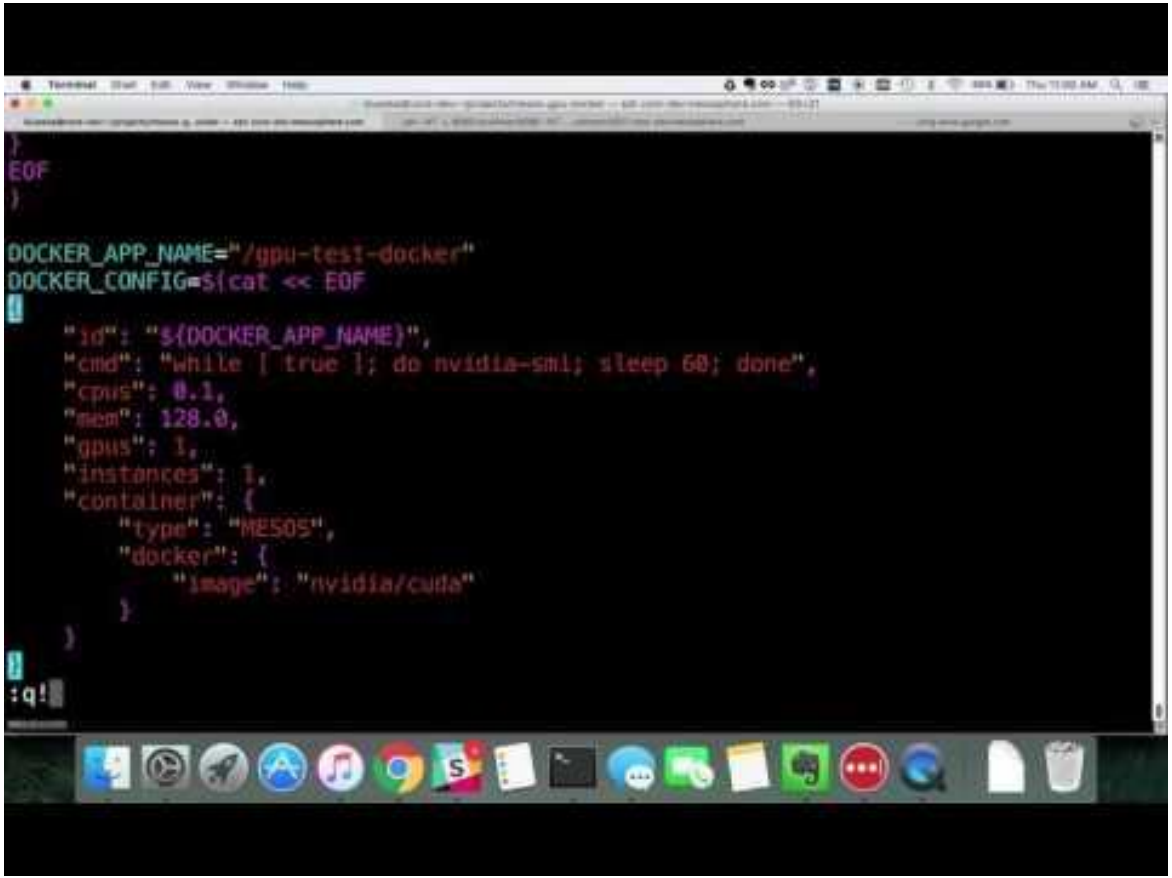


Vikram Ditya	Seetharami Seelam
Andrew Iles	Yong Feng
Jonathan Calmels	Guangya Liu
Felix Abecassis	Ian Downes
Rob Todd	Niklas Nielson
Rajat Phull	Connor Doyle
Shivi Fotedar	Benjamin Mahler
	Tim Chen

ISOLATION

DEMO

https://github.com/klueska-mesosphere/mesos-gpu-docker


}
EOF
)
DOCKER_APP_NAME="/gpu-test-docker"
DOCKER_CONFIG=$(cat << EOF
{
"id": "${DOCKER_APP_NAME}",
"cmd": "while [true]; do nvidia-smi; sleep 60; done",
"cpus": 0.1,
"mem": 128.0,
"gpus": 1,
"instances": 1,
"container": {
"type": "MESOS",
"docker": {
"image": "nvidia/cuda"
}
}
}
:q!

DEMO

FROM IBM