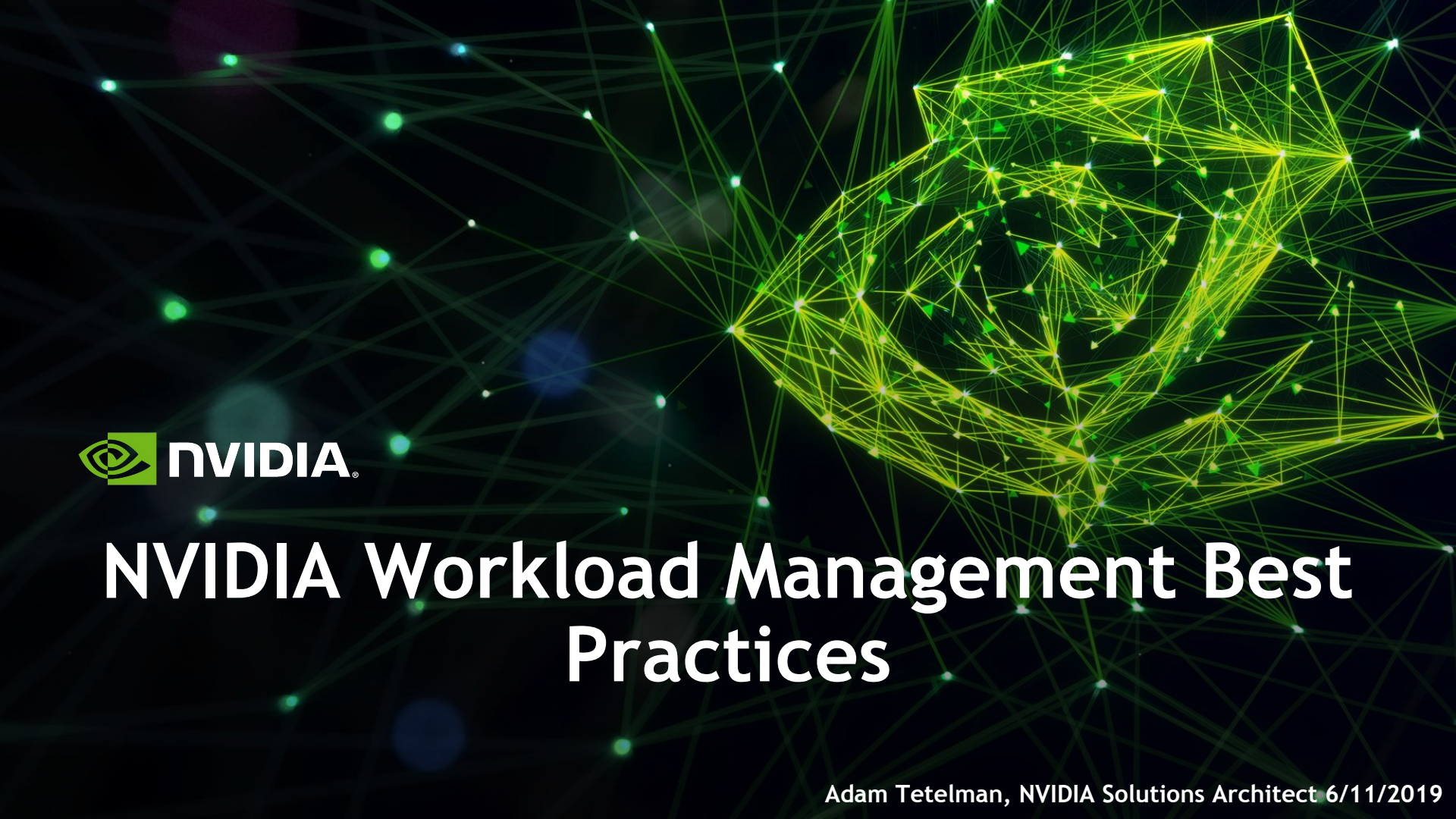


NVIDIA Workload Management Best Practices

Adam Tetelman, NVIDIA Solutions Architect 6/11/2019

Today's Discussion

- Docker for consistency and ease
- NGC for software and optimizations
- Kubernetes for management
- DeepOps for installation, configuration, and monitoring
- DevOps & development best practices

NVIDIA.

Docker

NVIDIA-Docker

NGC

- What is a container? What problems to they solve?
- What is NVIDIA-Docker?
- What are the problems with managing containers?
- How does NGC streamline this process?
- How does NGC guarantee the best performance?


Containers and
nvidia-docker

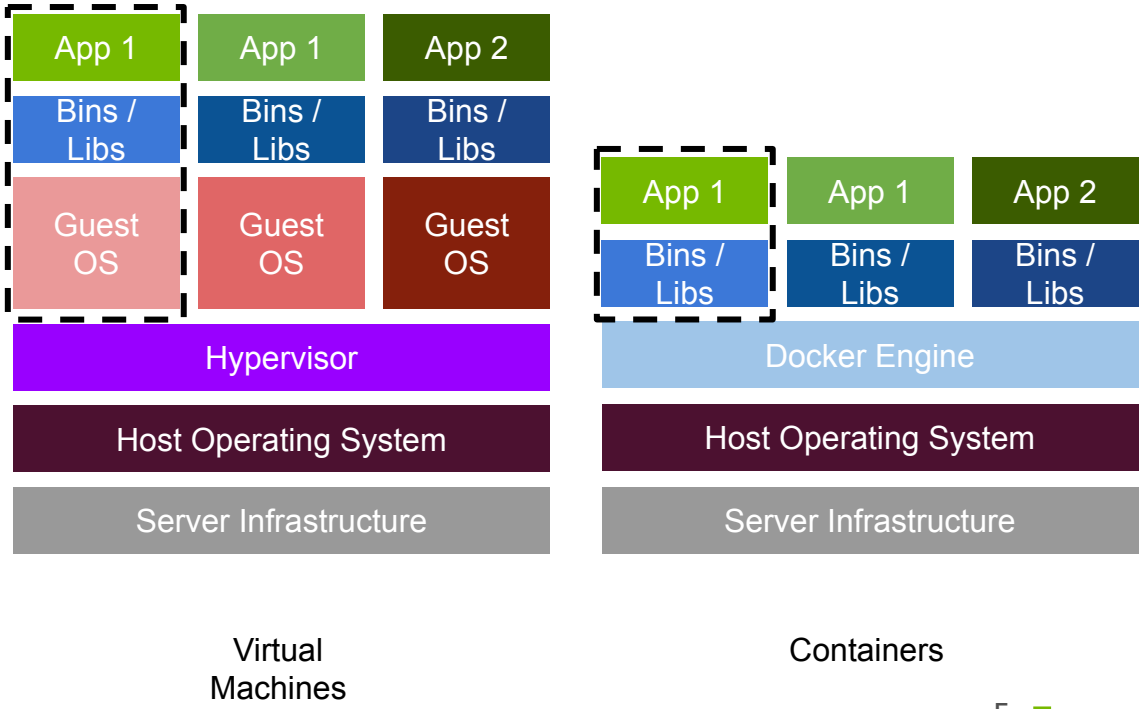
VIRTUAL MACHINES VS. CONTAINERS

Motivation

Packaging mechanism for applications

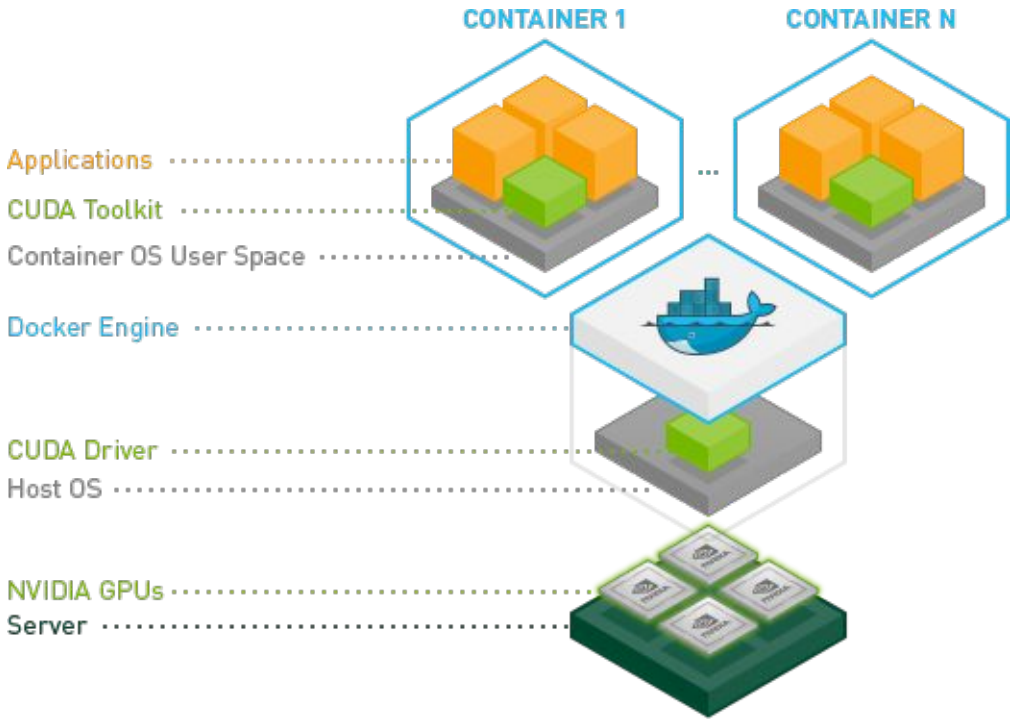
- Consistent and reproducible deployment
- Lightweight with faster startup than VMs
- Logical isolation from other applications (at the OS level)

No first-class support for GPUs was available in runtimes



NVIDIA Container Runtime (nvidia-docker)

Separation of NVIDIA drivers and CUDA runtime



Containers are hardware agnostic

nvidia-docker is required to use GPUs

NVIDIA drivers live on the host OS

CUDA toolkit live on the container

Old containers will work on new drivers

New containers will work on old drivers


What is NGC? Why NGC?

NVIDIA GPU CLOUD — ONE PLATFORM, RUN EVERYWHERE

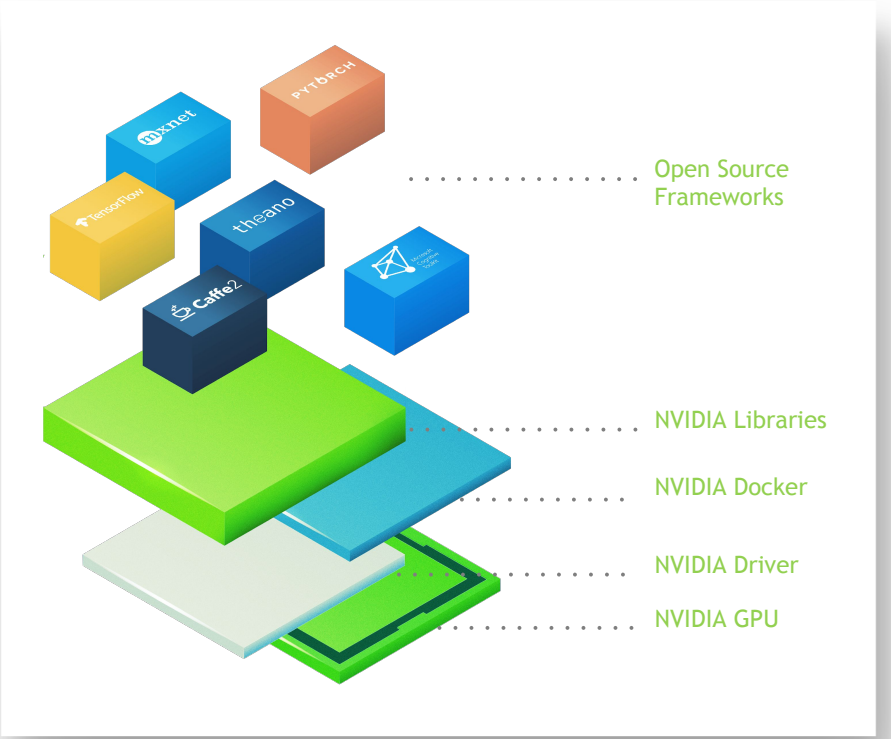
The NVIDIA GPU Cloud gives AI and HPC developers access to optimized software stacks wherever they want it — on PCs, in the data center, or via the cloud.
We offer containerized stacks for deep learning, HPC, and HPC visualization.
NVIDIA GPU Cloud makes it easier for developers to focus on their work.

Challenges with Complex Software

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

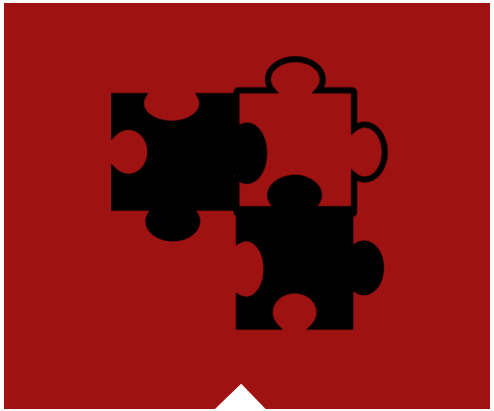
Development of software frameworks by the community is moving **very fast**

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



CHALLENGES WITH DEEP LEARNING

Performance



Version Compatibility

Best Practices

NVIDIA.

DEEP LEARNING FRAMEWORKS YEAR-OVER-YEAR HIGHLIGHTS

Best NVIDIA Performance

Deep Learning Frameworks optimizations for NVIDIA hardware

Volta Tensor Cores support for mixed-precision (FP16) across TensorFlow, MXNet, PyTorch, Caffe2, CNTK, NVCaffe, and Theano.

Latest NVIDIA Features

Latest NVIDIA Deep Learning libraries incorporated cuDNN, cuBLAS, and NCCL

cuDNN and cuBLAS have continuous CNN and RNN optimizations and fixes

NCCL has ongoing updates for multi-GPU and multi-node performance

NVIDIA Examples & QA Verified

Optimized popular network examples: ResNet-50,OpenNMT , AlexNet, ONNX, etc.

Improved documentation

Thorough monthly quality assurance testing

Incorporated more tools: Python 3, OpenMPI, and Horovod

NVIDIA.

Live NGC Demo!

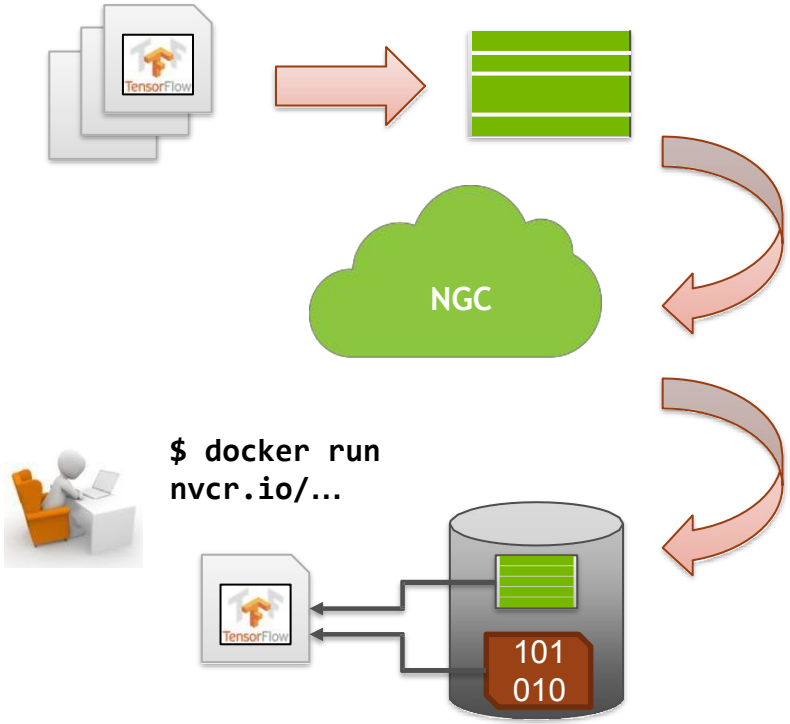
Run through an optional NGC DL training demo

- Visit NGC & NGC examples GitHub
  - https://ngc.nvidia.com
  - https://github.com/NVIDIA/DeepLearningExamples
- Review the available containers
- Run the following Docker command
  - ```
    docker run --runtime=nvidia -it --rm nvcr.io/nvidia/tensorflow:19.04-py3 \
        python /workspace/nvidia-examples/cnn/resnet.py --layers=50
    ```
  - For multi-gpu run

    ```
    docker run --runtime=nvidia -it --rm nvcr.io/nvidia/tensorflow:19.04-py3 \
        mpiexec --allow-run-as-root --bind-to-socket -np 4
            python /workspace/nvidia-examples/cnn/resnet.py --layers=50
    ```


NVIDIA.
    ```

EXAMPLE NGC CONTAINER WORKFLOW



NVIDIA builds application image composed of layers of files

Image(s) tested and released to NGC repository

User pulls image to a machine and runs it

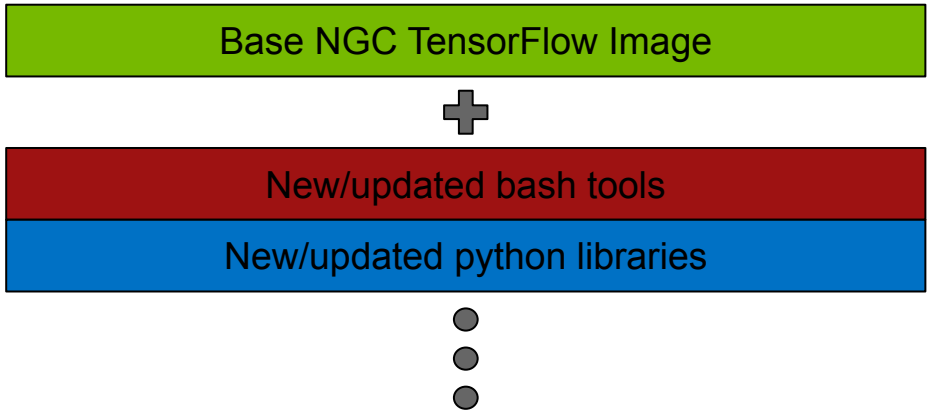
Image cached and OS isolated set of resources allocated (container) in which to execute

Data & results accessed as a filesystem volume

NVIDIA

Extending NGC Images

Tensorflow & Jupyter Exampe




```
# Use the NGC BaseImage

FROM nvcr.io/nvidia/tensorflow:19.02-py3

#Install Jupyter

RUN pip install jupyterlab

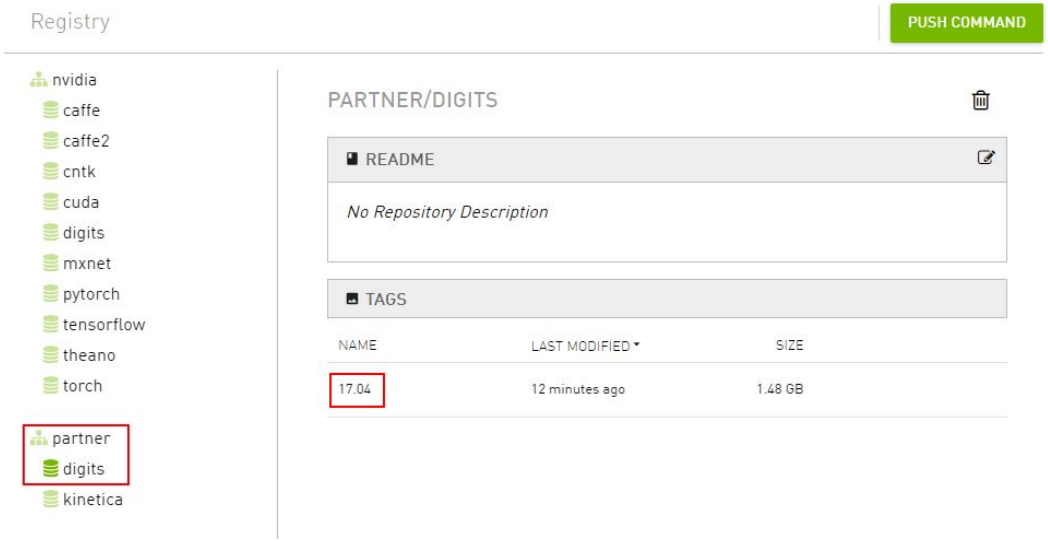
#Install Jupyter when the container is started
CMD ["jupyter", "lab", "--allow-root", "ip=0.0.0.0"]
```

- A Dockerfile is a configuration file that tell docker how to build a custom image - “layer by layer”
- Dockerfiles allow for building images based on the well tested & optimized NGC images

PUSHING AN IMAGE TO NGC

```
$ docker tag custom-image:latest nvcr.io/partner/digits:17.04

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



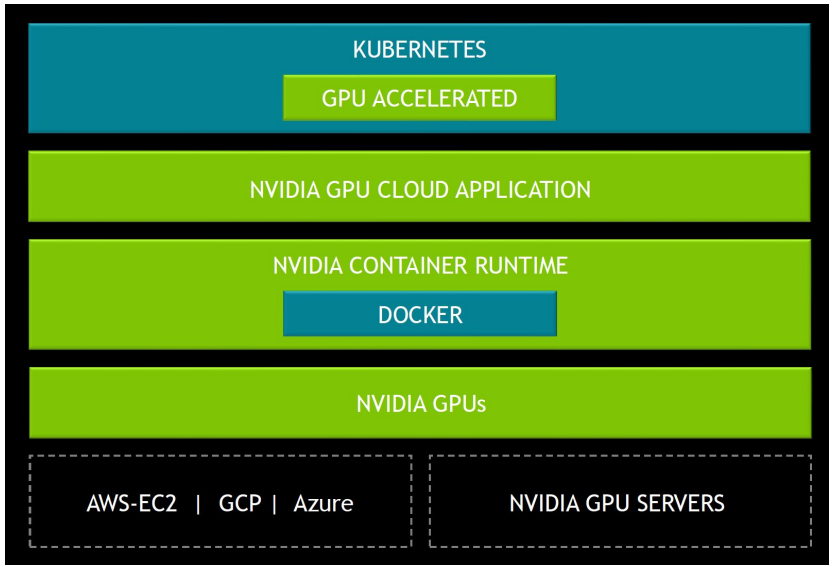
Kubernetes

- What is Kubernetes (k8s)?
- Why Kubernetes?
- Benefits of Kubernetes on NVIDIA GPUs
- How do I setup and use Kubernetes?


Kubernetes Overview

KUBERNETES ON NVIDIA GPUS

Preferred orchestration system for GPU containers

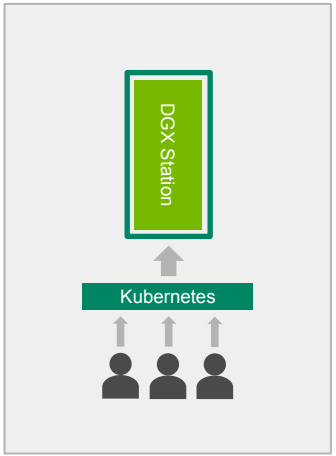


Kubernetes is an open source container orchestration platform by google

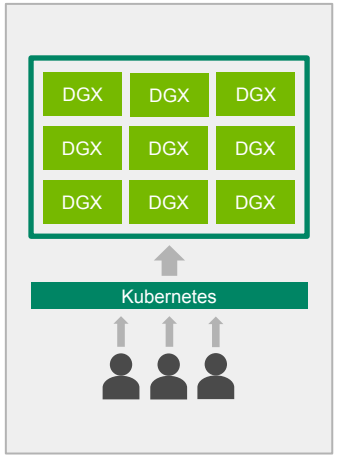
- Running containers across different machines
- Scale up/down pods to meet demands
- Keep storage consistent across pods
- Distribute load between containers
- Launching new containers on different machines if something fails

USE CASES

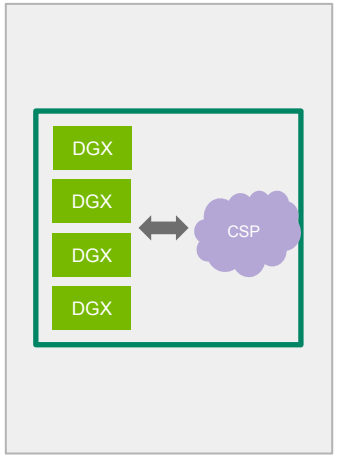
Where can it be leveraged?



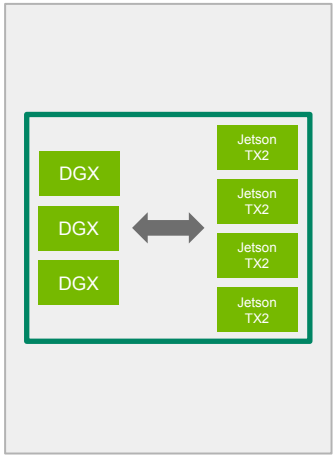
Many users, single node
On-prem



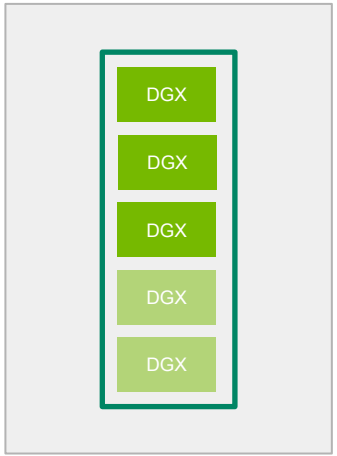
Many users, many nodes
On-prem



Cloud bursting
Hybrid



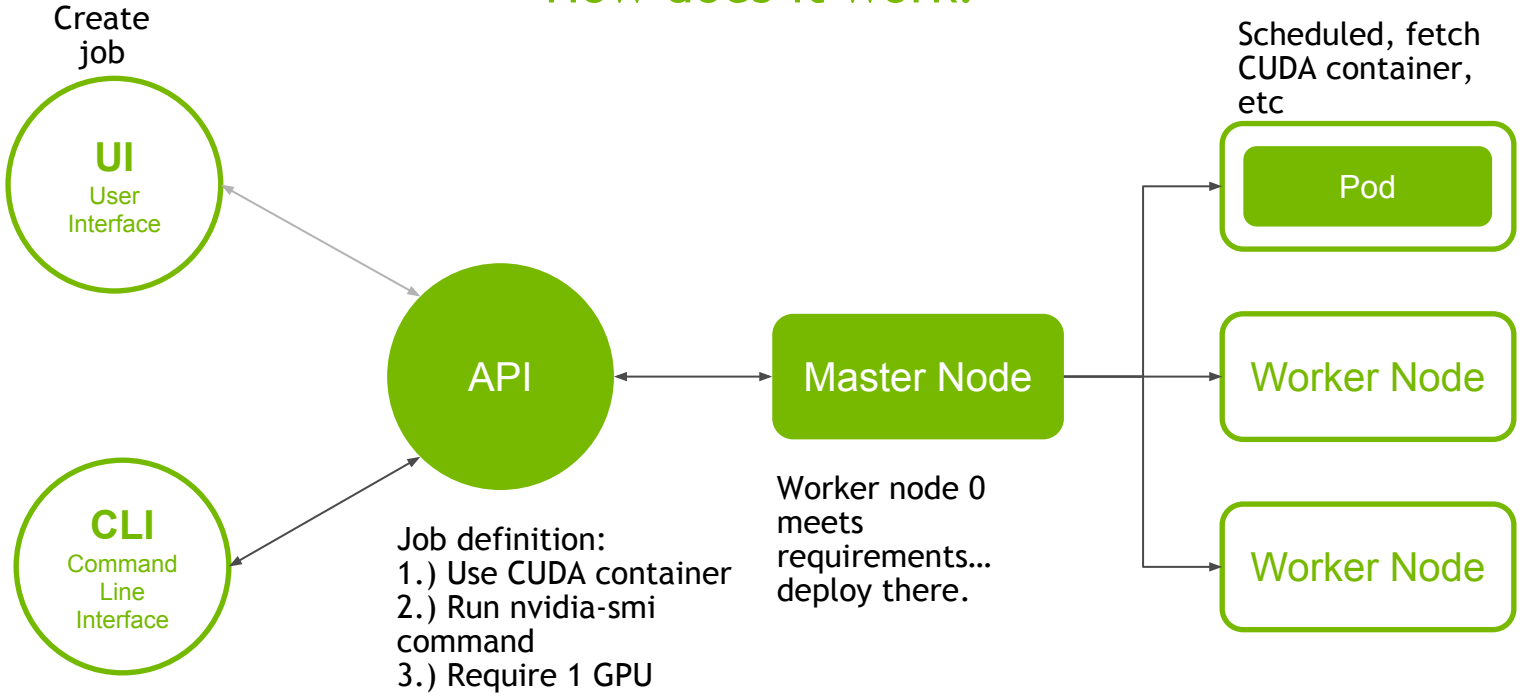
Edge/IoT
Multi-region



Production Inferencing
*

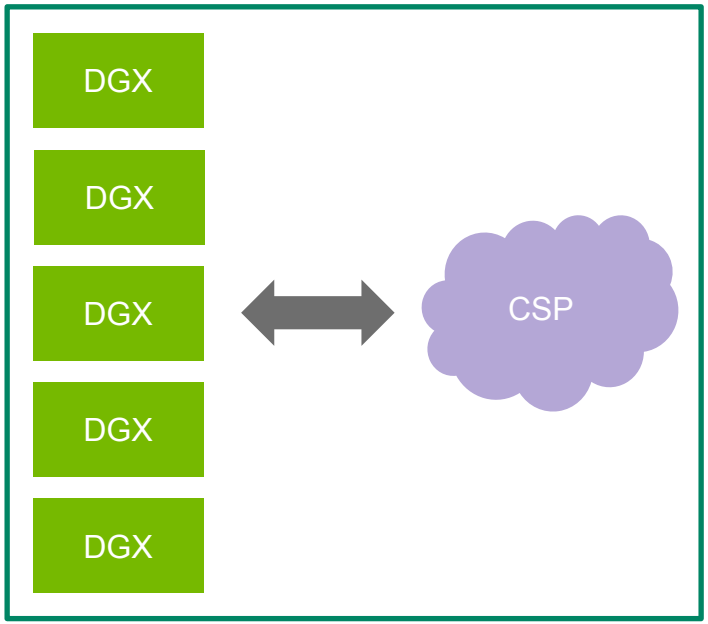
KUBERNETES

How does it work?



CLOUD BURSTING

Excess work pushed to cloud



Dozens of cluster users

Steady usage, with sudden bursts of unpredictable, but infrequent, demand

Access to multiple CSPs

Can offload extra jobs to CSPs when demand is high

Minimize costly CSP charges, arbitrage between CSPs

EDGE / IOT

Clusters can be spread across geography

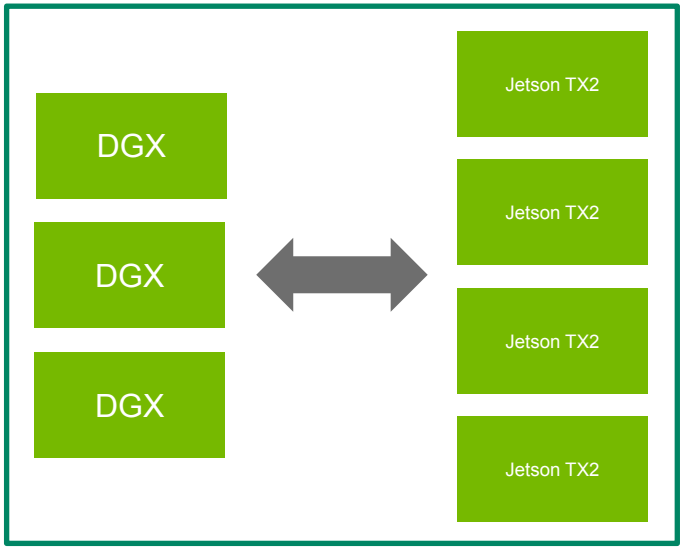
Edge devices collect data

Backend servers perform analysis on data and make suggestions

Suggestions pushed to edge devices

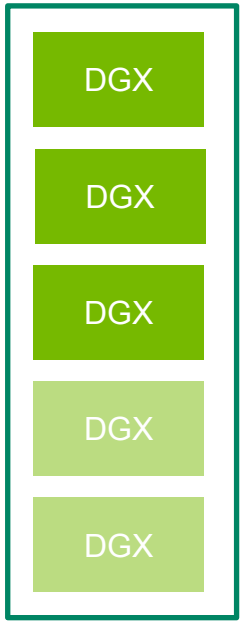
All systems contained in single cluster

Adding edge devices and backend servers to cluster is simplified



PRODUCTION INFERENCING

API exposing cluster of trained models



Very well-suited for Kubernetes

Cluster is self-healing

Scaling is trivial

On-prem, cloud, hybrid, multi-region

Upgrading services, balancing load is simplified

Ex: Flowers Demo at GTC 2017

Run GPU Tasks

Test GPU support has been properly set up by running a simple command

```
$ kubectl describe nodes | grep gpu
nvidia.com/gpu:  8
nvidia.com/gpu:  8
```

Start the CUDA sample workload

```
$ kubectl create -f /etc/kubeadm/examples/pod.yml
```

When the pod is running, execute the nvidia-smi command inside the container:

```
$ kubectl exec -it gpu-pod nvidia-smi
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.125                Driver Version: 384.125                   |
|-------------------------------+----------------------+----------------------+
| 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 V100-SXM2...  On   | 00000000:00:1E.0 Off |                    0 |
| N/A   34C    P0    20W / 300W |     10MiB / 16152MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

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

Managing Team Resources

RBAC and Resource Quotas

- Create an isolated namespace for each team or project
- Set resource quotas for each namespace to restrict GPU/CPU availability
- Monitor utilization for entire cluster

NVIDIA.


DeepOps Services

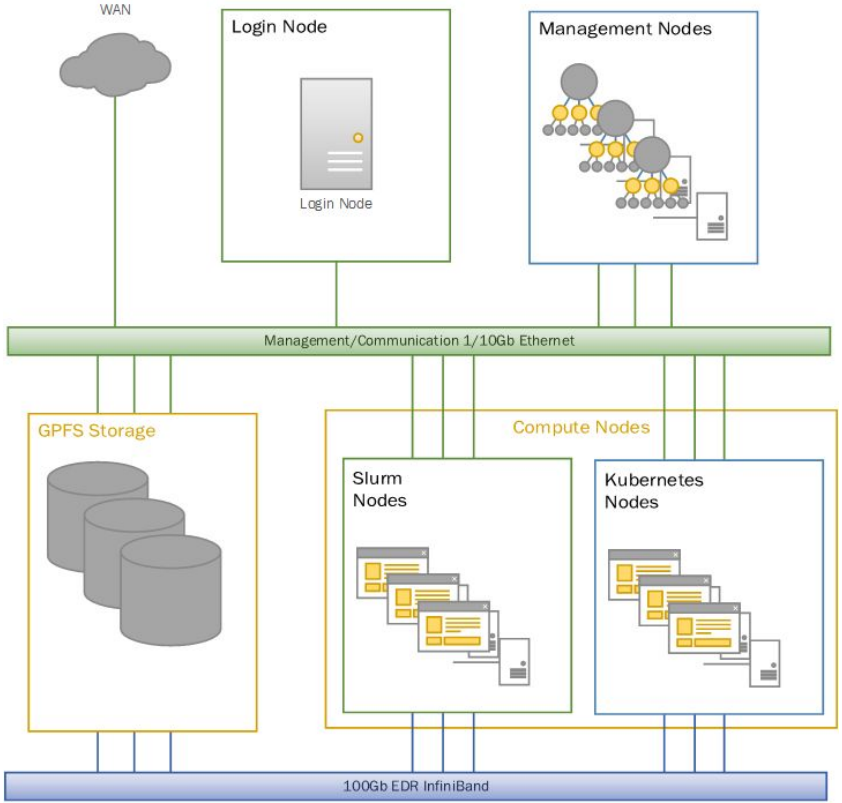
DeepOps Benefits

DeepOps streamlines the setup, monitoring, and management of your DGX

- Install and setup Kubernetes cluster (Ansible)
- Support hybrid Slurm/Kubernetes environments
- Deploy cluster-wide logging dashboard service (Grafana & Prometheus)
- Aggregated logging (ELK)
- Load Balancing (MetalLB)
- Machine diagnostics and configuration (DCGM)
- Air gapped systems
- Jupyter Notebooks (JupyterHub)
- DL/ML workflows (Kubeflow /Polyaxon / Dask)

NVIDIA.

Hybrid Kubernetes & Slurm Cluster



NVIDIA Data Center GPU Manager

Monitor the health of your hardware

- DCGM simplifies administration of Tesla GPUs
- Intelligent, lightweight userspace library/agent
- Functions:
  - GPU behavior monitoring
  - GPU configuration management
  - GPU policy oversight
  - GPU health and diagnostics
  - GPU accounting and process statistics
- Feeds the metrics dashboard via node exporter or collectd plugin

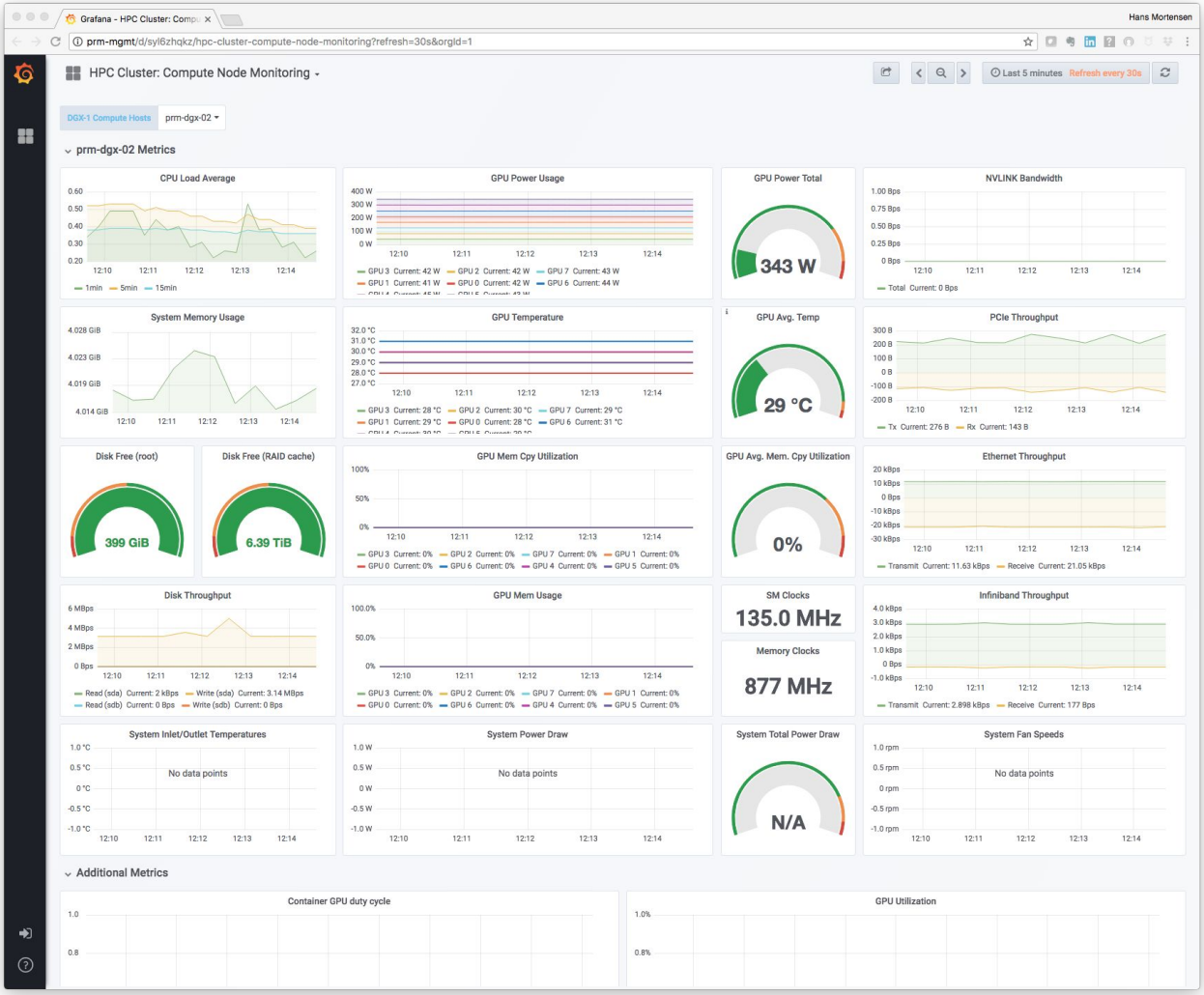
NVIDIA.

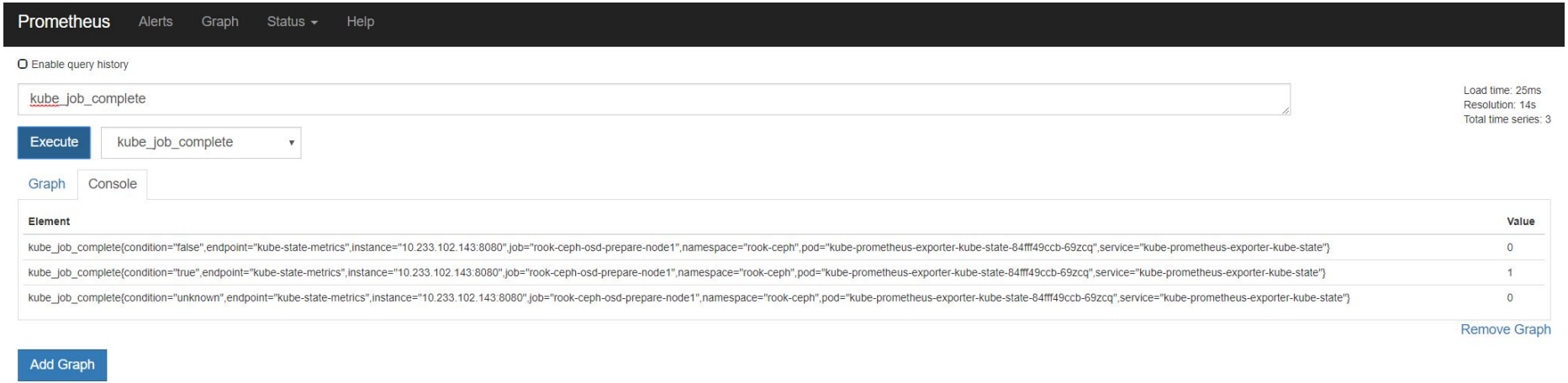
Grafana / Prometheus

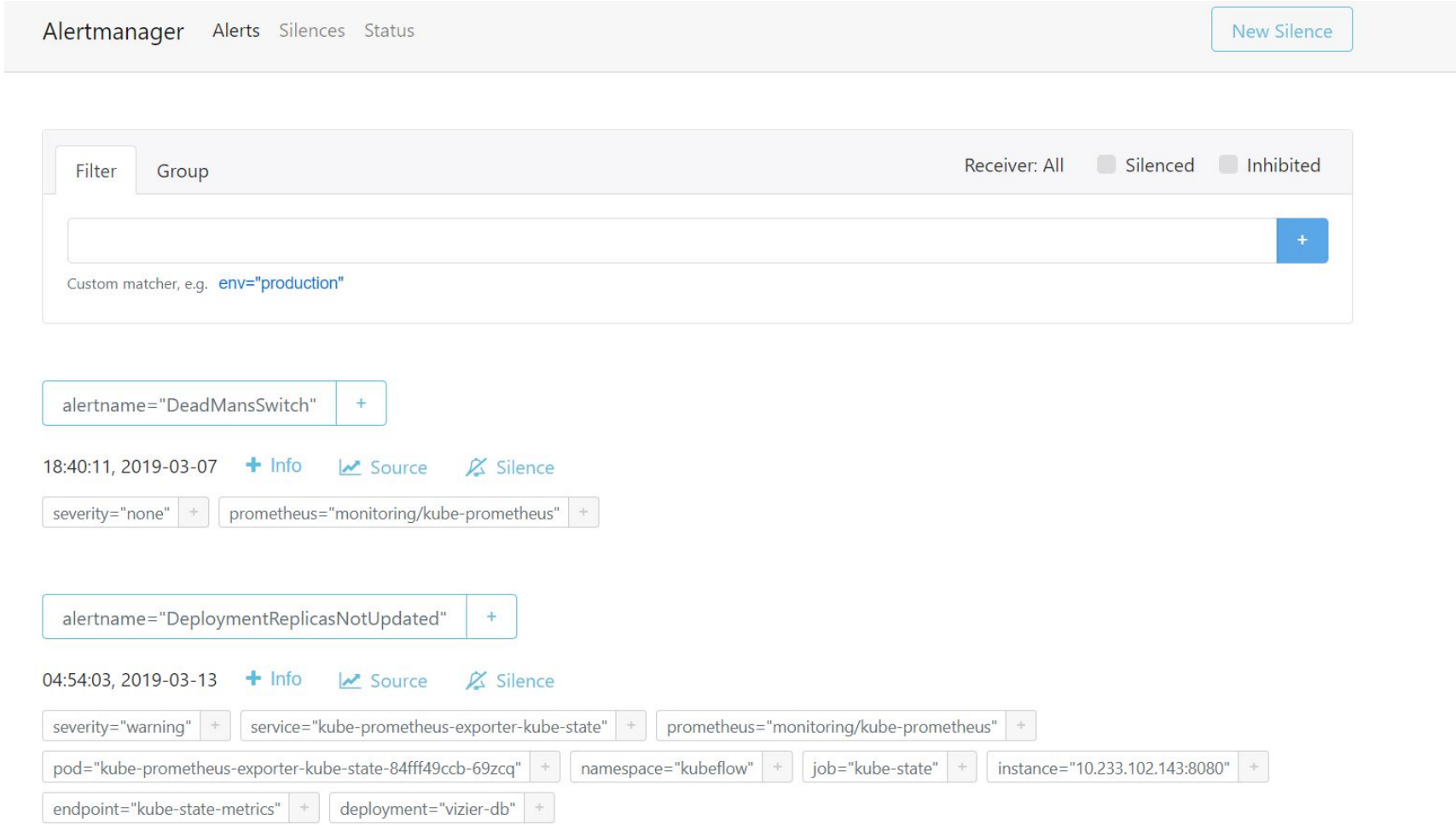
Monitor the status of your workloads

- Query
  - Collect metrics from virtually any data source
  - Mix disparate data sources in the same graph
- Visualize
  - Client-side graphs
  - Many different ways to view metrics & logs
- Alert on
  - Visually define alert rules
  - Email, Slack, PagerDuty, VictorOps, OpsGenie, or webhook
- Understand your metrics
  - Create dynamic and reusable dashboards
- Fosters a data driven culture
  - Annotate graphs with metadata/tags

NVIDIA.


Grafana - HPC Cluster: Compu
Hans Mortensen
prm-mgmt/d/syl6zhqkz/hpc-cluster-compute-node-monitoring?refresh=30s&orgId=1
HPC Cluster: Compute Node Monitoring
Last 5 minutes
Refresh every 30s
DGX-1 Compute Hosts
prm-dgx-02
prm-dgx-02 Metrics
CPU Load Average
GPU Power Usage
GPU Power Total
343 W
NVLINK Bandwidth
System Memory Usage
GPU Temperature
GPU Avg. Temp
29 °C
PCIe Throughput
Disk Free (root)
399 GiB
Disk Free (RAID cache)
6.39 TiB
GPU Mem Cpy Utilization
GPU Avg. Mem. Cpy Utilization
0%
Ethernet Throughput
Disk Throughput
GPU Mem Usage
SM Clocks
135.0 MHz
Memory Clocks
877 MHz
Infiniband Throughput
System Inlet/Outlet Temperatures
No data points
System Power Draw
No data points
System Total Power Draw
N/A
System Fan Speeds
No data points
Additional Metrics
Container GPU duty cycle
GPU Utilization


Prometheus
Alerts
Graph
Status
Help
Enable query history
kube_job_complete
Load time: 25ms
Resolution: 14s
Total time series: 3
Execute
kube_job_complete
Graph
Console
Element
Value
kube_job_complete{condition="false",endpoint="kube-state-metrics",instance="10.233.102.143:8080",job="rook-ceph-osd-prepare-node1",namespace="rook-ceph",pod="kube-prometheus-exporter-kube-state-84fff49ccb-69zcq",service="kube-prometheus-exporter-kube-state"} 0
kube_job_complete{condition="true",endpoint="kube-state-metrics",instance="10.233.102.143:8080",job="rook-ceph-osd-prepare-node1",namespace="rook-ceph",pod="kube-prometheus-exporter-kube-state-84fff49ccb-69zcq",service="kube-prometheus-exporter-kube-state"} 1
kube_job_complete{condition="unknown",endpoint="kube-state-metrics",instance="10.233.102.143:8080",job="rook-ceph-osd-prepare-node1",namespace="rook-ceph",pod="kube-prometheus-exporter-kube-state-84fff49ccb-69zcq",service="kube-prometheus-exporter-kube-state"} 0
Remove Graph
Add Graph


Alertmanager
Alerts
Silences
Status
New Silence
Filter
Group
Receiver: All
Silenced
Inhibited
Custom matcher, e.g. env="production"
alertname="DeadMansSwitch"
18:40:11, 2019-03-07
Info
Source
Silence
severity="none"
prometheus="monitoring/kube-prometheus"
alertname="DeploymentReplicasNotUpdated"
04:54:03, 2019-03-13
Info
Source
Silence
severity="warning"
service="kube-prometheus-exporter-kube-state"
prometheus="monitoring/kube-prometheus"
pod="kube-prometheus-exporter-kube-state-84fff49ccb-69zcq"
namespace="kubeflow"
job="kube-state"
instance="10.233.102.143:8080"
endpoint="kube-state-metrics"
deployment="vizier-db"

Logging

Aggregated logs for security and metrics

- “ELK” Stack
  - Elasticsearch database
  - Filebeat log collector
  - Kibana dashboard
- Visualize
  - Client-side graphs
  - Many different ways to view metrics & logs
- Consolidate
  - Centralized logs from the entire cluster
- Monitor
  - Audit commands like sudo and docker
  - Monitor SSH connections

NVIDIA.

kibana

Discover
Visualize
Dashboard
Timelion
Dev Tools
Management

Collapse

Dashboard / [Filebeat System] Sudo commands

Full screen Share Clone Edit Auto-refresh Last 30 days

*

Options

Add a filter +

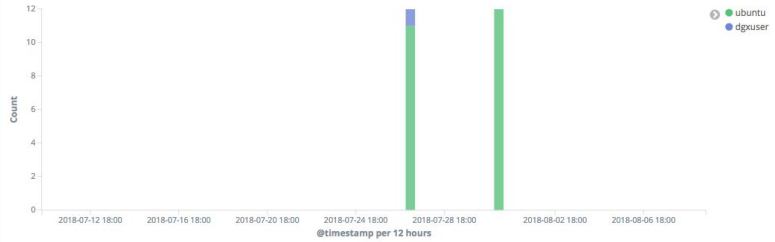
Dashboards [Filebeat System]

Syslog | Sudo commands | SSH logins | New users and groups

Top sudo commands [Filebeat System]

system.auth.sudo.command: Descending	system.auth.user: Descending	Count
/usr/bin/osqueryi .all deb_packages	ubuntu	3
/usr/bin/apt update	ubuntu	2
/usr/bin/apt upgrade	ubuntu	2
/usr/bin/apt-get dist-upgrade	ubuntu	2
/usr/bin/osqueryi	ubuntu	2

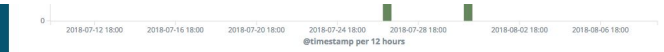
Sudo commands by user [Filebeat System]



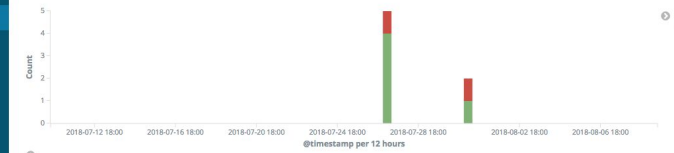
kibana

Discover
Visualize
Dashboard
Timelion
Dev Tools
Management

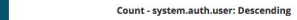
Collapse



Successful SSH logins [Filebeat System]



SSH users of failed login attempts [Filebeat System]



SSH failed login attempts source locations [Filebeat System]

SSH login attempts [Filebeat System]

1–7 of 7

Time	system.auth.ssh.event	system.auth.ssh.method	system.auth.user	system.auth.ssh.ip	system.auth.ssh.geoip.country_iso
July 31st 2018, 09:26:35.000	Accepted	password	ubuntu	192.168.1.1	-
July 31st 2018, 09:26:30.000	Accepted	publickey	ubuntu	10.31.241.164	-
July 27th 2018, 11:42:11.000	Accepted	password	dgxuser	192.168.1.1	-
July 27th 2018, 11:42:11.000	Accepted	publickey	ubuntu	10.31.241.164	-
July 27th 2018, 11:42:11.000	Accepted	publickey	ubuntu	10.31.241.164	-

NVIDIA DeepOps Project

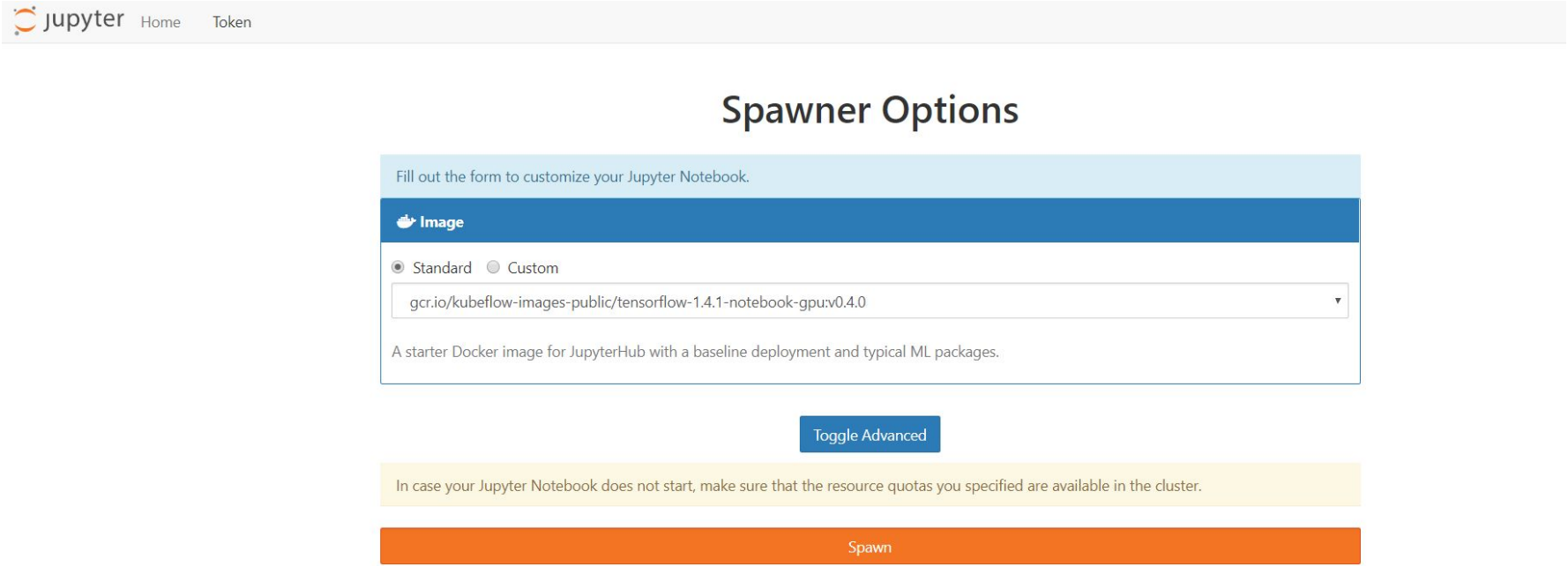
Load Balancing

- MetalLB is an on-prem Kubernetes load balancer
- Reserve and Allocate an IP range from your network
- MetalLB will distribute IPs to all “LoadBalancer” K8S services

```
NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
rapids-dask-jupyter   LoadBalancer   10.233.18.235   192.168.1.240   80:32001/TCP                 23h
rapids-dask-scheduler LoadBalancer   10.233.37.84    192.168.1.241   8786:32611/TCP,80:32558/TCP  23h
```

NVIDIA DeepOps Project

JupyterHub



NVIDIA DeepOps Project

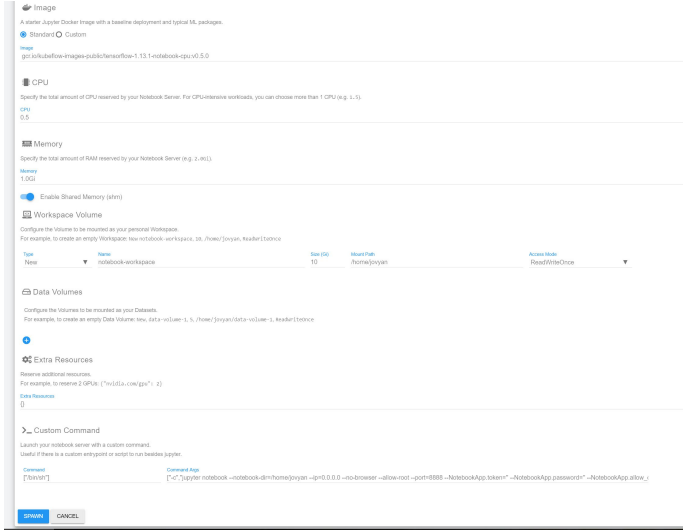
Polyaxon

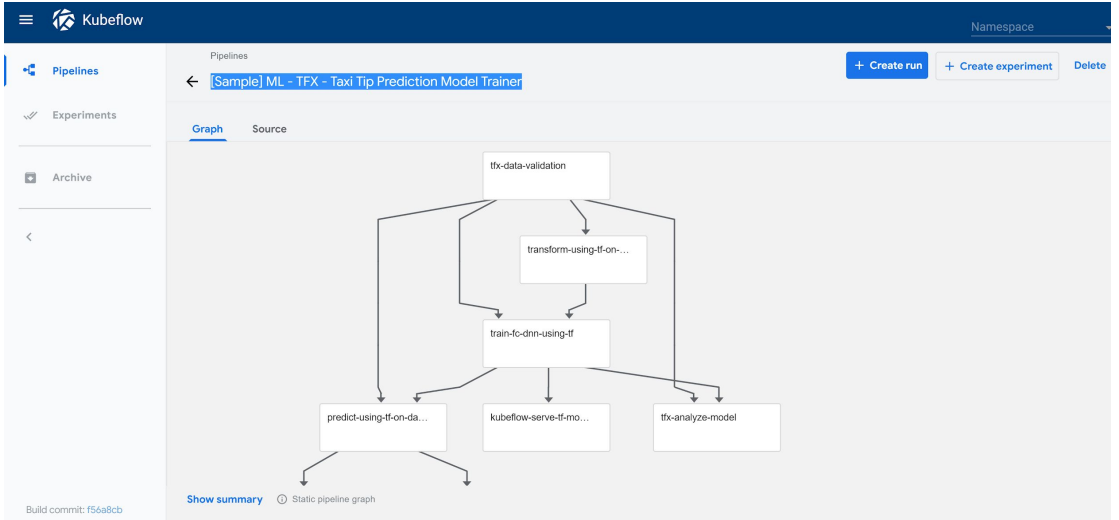
- TBD
- https://polyaxon.com/

NVIDIA DeepOps Project

KubeFlow

Define pipelines, deploy notebooks, and more: https://www.kubeflow.org/docs/about/kubeflow/

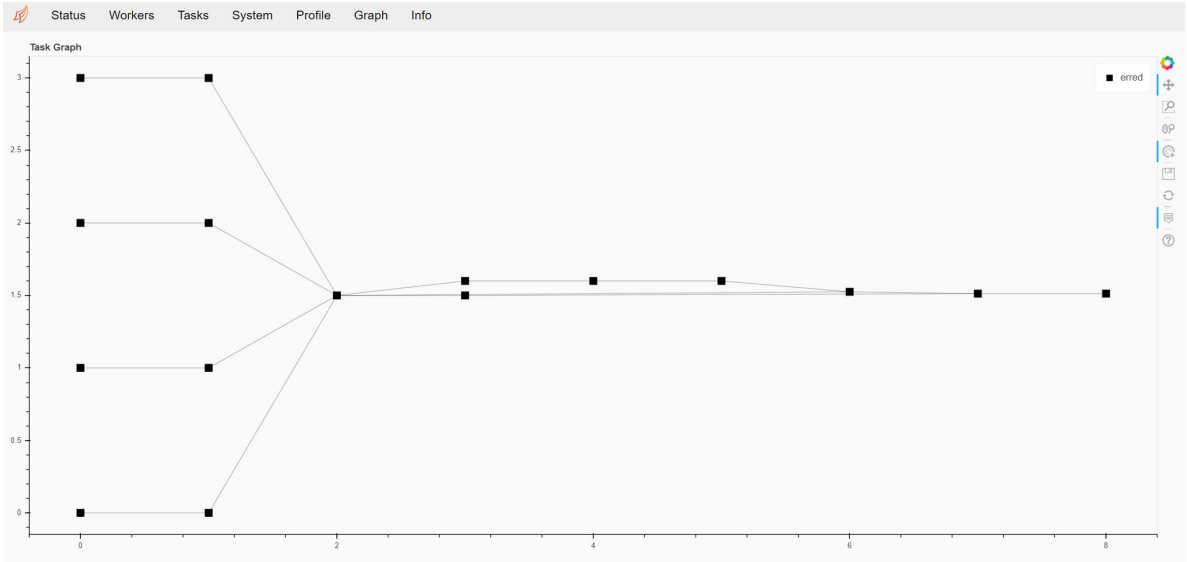




NVIDIA DeepOps Project

Dask - Distributed ML in Python

- http://docs.dask.org/en/latest/why.html



NVIDIA DeepOps Project

Horovod- Distributed DL in Python

- https://github.com/horovod/horovod
- https://github.com/helm/charts/tree/master/stable/horovod
- https://github.com/kubeflow/mpi-operator/
- https://github.com/NVIDIA/DeepLearningExamples/tree/master/TensorFlow/Classification/RN50v1.5#quick-start-guide

NVIDIA.

NVIDIA DeepOps Project

Supporting air-gapped clusters for high security deployments

- Clones DGX container repository (NGC Container Replicator)
- Packages OTA updates: OS, Docker, 3rd Party bits
- Hosts on the internal network
  - Docker registry
  - APT repository
  - DGX container website & repository
  - Automated provisioning services: PXE & Ansible
  - Job Scheduler/Orchestrator: Slurm & Kubernetes
  - Development sandbox: JupyterHub
  - Basic user & file services management

NVIDIA.


Using DeepOps

DeepOps Overview

A few simple steps

- DGX & management node OS install
  - DGX Official ISO and Ubuntu 18 recommended
- Install external storage
  - NFS recommended
- Install K8S cluster
- Install Slurm (optional)
- Install optional DeepOps services
  - K8S dashboard
  - Monitoring / logging
  - Load balancing
  - Storage orchestration
  - ML / DL workflow tooling
  - Container registry
  - …

NVIDIA.

DeepOps Installation

OS Bootstrapping

OS bootstrapping can be done in many ways. DeepOps provides directions to set up a PXE server, however for smaller (less than 5) clusters it is reasonable to install your OS using the remote DGX console.

DGX:

- Install the latest version of the DGX OS

Management Node:

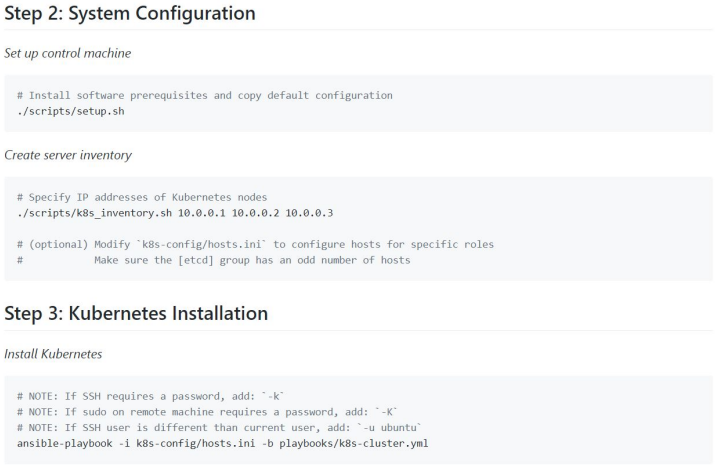
- Install a fresh version of Ubuntu 18

NVIDIA.

DeepOps Installation

Deploying K8S

Deploying Kubernetes with DeepOps involves two steps. Cluster configuration and cluster installation.



Step 2: System Configuration

Set up control machine

```
# Install software prerequisites and copy default configuration
./scripts/setup.sh
```

Create server inventory

```
# Specify IP addresses of Kubernetes nodes
./scripts/k8s_inventory.sh 10.0.0.1 10.0.0.2 10.0.0.3

# (optional) Modify `k8s-config/hosts.ini` to configure hosts for specific roles
#            Make sure the [etcd] group has an odd number of hosts
```

Step 3: Kubernetes Installation

Install Kubernetes

```
# NOTE: If SSH requires a password, add: `-k`
# NOTE: If sudo on remote machine requires a password, add: `-K`
# NOTE: If SSH user is different than current user, add: `-u ubuntu`
ansible-playbook -i k8s-config/hosts.ini -b playbooks/k8s-cluster.yml
```

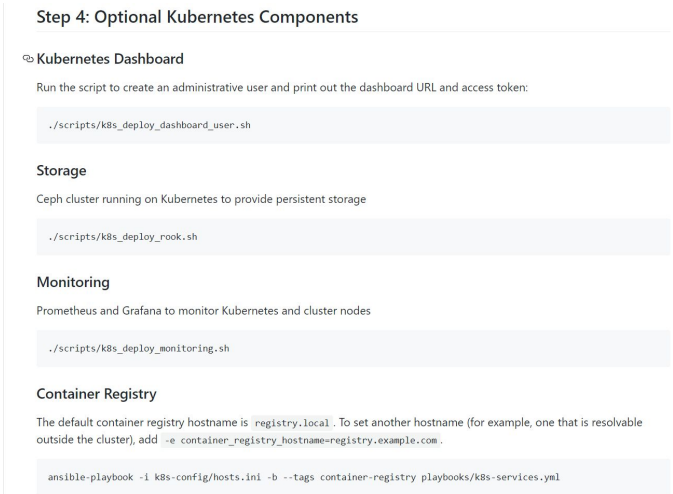
https://github.com/NVIDIA/deepops/blob/master/docs/kubernetes-cluster.md

NVIDIA.

DeepOps Installation

Installing additional services

Installing additional services requires running the corresponding script. Many of these scripts make use of helm to install stable packages according to the custom configuration files included in DeepOps.



Step 4: Optional Kubernetes Components

Kubernetes Dashboard

Run the script to create an administrative user and print out the dashboard URL and access token:

```
./scripts/k8s_deploy_dashboard_user.sh
```

Storage

Ceph cluster running on Kubernetes to provide persistent storage

```
./scripts/k8s_deploy_rook.sh
```

Monitoring

Prometheus and Grafana to monitor Kubernetes and cluster nodes

```
./scripts/k8s_deploy_monitoring.sh
```

Container Registry

The default container registry hostname is `registry.local`. To set another hostname (for example, one that is resolvable outside the cluster), add `-e container_registry_hostname=registry.example.com`.

```
ansible-playbook -i k8s-config/hosts.ini -b --tags container-registry playbooks/k8s-services.yml
```

https://github.com/NVIDIA/deepops/blob/master/docs/kubernetes-cluster.md

NVIDIA.

DeepOps Installation

Deploying Kubeflow

Kubeflow is a tool for defining Deep Learning workflows, managing user/data volumes, and quickly getting engineers and scientists access to containerized development environments.

Installing Kubeflow involves running a single script that creates a K8S namespace and deploys various services.

https://github.com/NVIDIA/deepops/blob/master/docs/kubernetes-cluster.md#kubeflow

NVIDIA.

DeepOps Installation

Deploying SLURM for HPC

SLURM is a batch job scheduler often used for HPC applications.

Installing SLURM involves setting up an optional user login node, installing the slurmctld scheduler, and installing/building any required HPC software/environment modules.

SLURM can run alongside K8S.

https://github.com/NVIDIA/deepops/blob/master/docs/slurm-cluster.md

NVIDIA.


Deploying a job

Create a pod YAML file

YAML for all K8S objects

Create a YAML file to use a NGC container and run a training command

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pytorch-example
spec:
  backoffLimit: 5
  template:
    spec:
      imagePullSecrets:
        - name: nvcr.dgxkey
      containers:
        - name: pytorch-container
          image: nvcr.io/nvidia/pytorch:18.08-py3
          command: ["/bin/sh"]
          args: ["-c", "python
/workspace/examples/upstream/mnist/main.py"]
          extendedResourceRequests: ["nvidia-gpu"]
      restartPolicy: Never
      extendedResources:
        - name: "nvidia-gpu"
          resources:
            limits:
              nvidia.com/gpu: 4
            requests:
              nvidia.com/gpu: 4
```


NVIDIA.

Attach storage

Saving output and model files

DeepOps sets up persistent storage using the DGX built in drives. To use additional storage you can create persistent volumes and attach them to the pod YAML

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: nfs-pv
spec:
 capacity:
 storage: 10Gi
 accessModes:
 - ReadWriteMany
 nfs:
Kubernetes on NVIDIA DGX Servers DU-09176-001| 20
 server: <nfs_server>
 path: <nfs_mount_path>
```

```
volumes:
- name: nfs-vol
persistentVolumeClaim:
claimName: nfs-pvc
```


NVIDIA.

Execute a job

Simple create the pod you just defined to execute the job.

Optionally, Kubernetes can also be installed with a GUI to define and execute all these files

```
kubectl create -f ./pytorch-job.yml
```


NVIDIA.


http://github.com/nvidia/deepops


DevOps & Development
Best Practices

Versioning & Storage Best Practices

Infrastructure as Code (IaS) - Configuration as Code (CaC)

Code:

- Source controlled in Git
- Checked in and pushed on a regular basis
- Small incremental commits, no large binaries/data-sets

Infrastructure:

- Docker execution/creation/run commands saved off into version controlled scripts
- Dockerfiles, build scripts, etc. checked in “as code”
- Docker images saved off to external persistent repository

Data & Models:

- Data/models on DGX considered “cached storage” and treated as ephemeral
- Data saved off in persistent storage
- Models saved off in persistent storage in time-stamped directories

NVIDIA.

Make Provisioning Easy and Consistent

MAAS as example tool

A simple platform that allows management and provisioning of bare-metal hardware. Provisioning is repeatable, consistent, and can be tracked in SCM.

Provisioning

- Server inventory tracking
- PXE booting
- OS configurations
- Networking Configurations
- Application Deployment
- Monitoring

Similar tools include Foreman, Stacki, etc.

NVIDIA.

Make Configuration Repeatable and Testable

Ansible as example tool

Simple, agent-less and powerful open source IT automation tool

Configuration

- Configuration Management
- Application Deployment
- Continuous Delivery
- Security & Compliance
- Orchestration

Extensible Tool

- Ansible Galaxy (open repos)
- Customizable roles and filters

NVIDIA.


Resources & Links

Resources

- Docker
- NVIDIA Docker Runtime
- Kubernetes
- NVIDIA GPU Device plugin
- Internal Release README
- Getting Started Repo

- DeepOps
- NGC
- Ansible
- Prometheus
- Grafana
- AlertManager

- KubeFlow
- Polyaxon
- JupyterHub
- Helm
- Dask
- Horovod

Additional Presentations & Documents

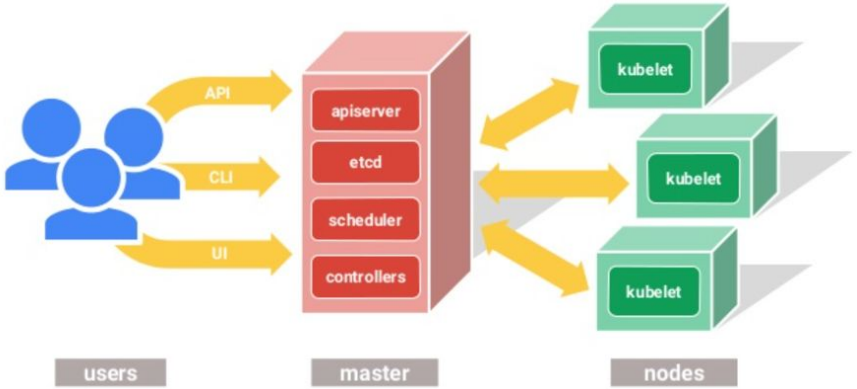
- GaaS Technologies Summary
- K8S Getting Started
- DeepOps Examples
- DeepOps Docs


Kubernetes Extra Slides

Kubernetes

What is it?

- Cloud-scale platform for deploying and managing containerized applications
- Orchestrates deployment through the cluster
- Has the makings of a great sysadmin tool
- High-availability and fault-tolerant by design
- If you can run it in a container, you can likely run it on kubernetes at scale

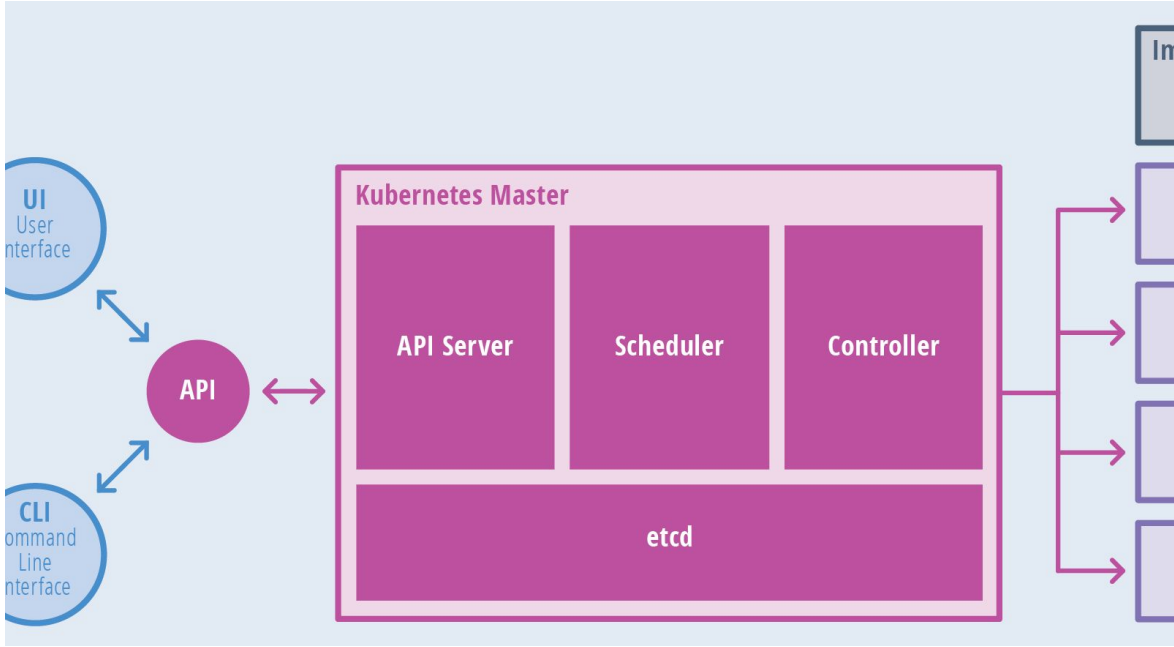


NVIDIA.

Kubernetes Master

The master is responsible for exposing the application program interface (API), scheduling the deployments and managing the overall cluster

This is typically a CPU server collocated with a DGX Pod

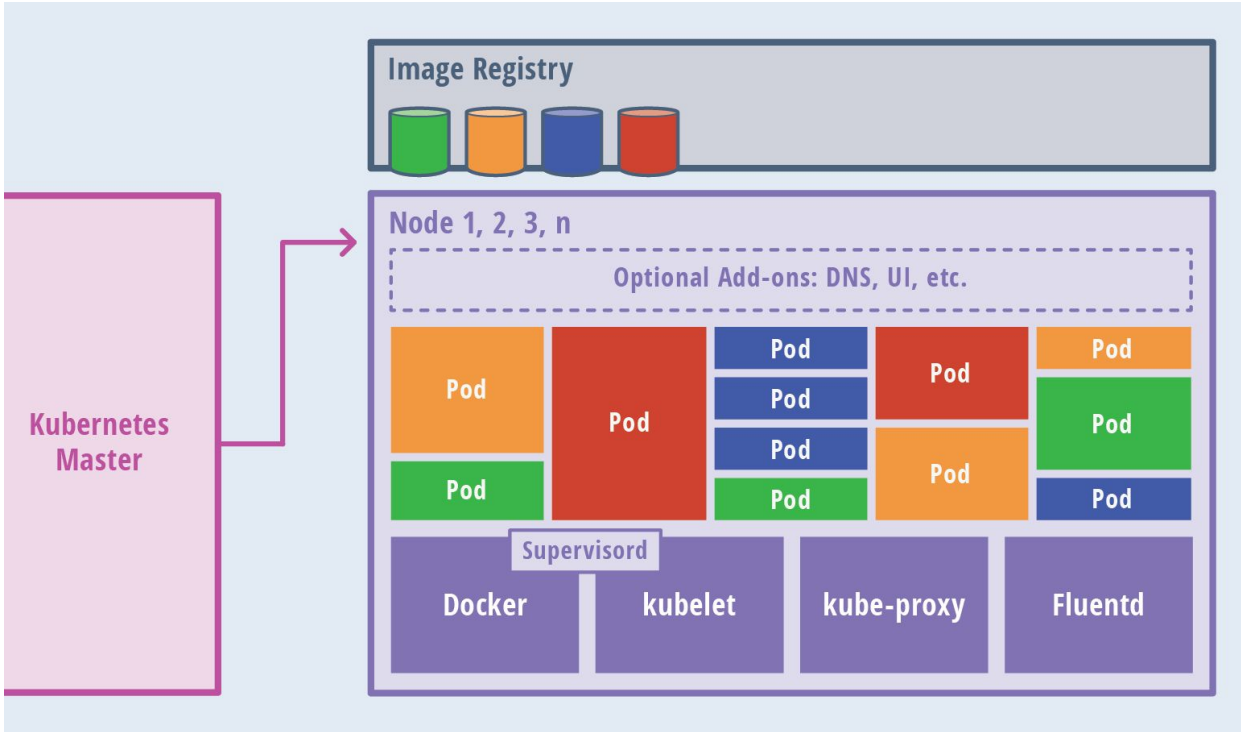


Kubernetes Node

Applications run on nodes

Each node runs a container runtime and an agent that communicates with the master

Additional components for logging, monitoring, service discovery and optional add-ons.



Deploying NVIDIA GPU device plugin

The NVIDIA device plugin for Kubernetes is a Daemonset that allows you to automatically:

- Expose the number of GPUs on each nodes of your cluster
- Keep track of the health of your GPUs
- Run GPU enabled containers in your Kubernetes cluster

NVIDIA GPU device plugin has the following requirements:

- Kubernetes nodes have to be pre-installed with NVIDIA drivers
- Kubernetes nodes have to be pre-installed with nvidia-docker 2.0
- nvidia-container-runtime configured as the default runtime for docker instead of runc
- NVIDIA drivers >= 361.93

To deploy the NVIDIA device plugin once your cluster is running:

```
# For Kubernetes v1.10
kubectl create -f
https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v1.10/nvidia-device-plugin.yml
```

https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/

DeepOps Extra Slides

Why Ansible?

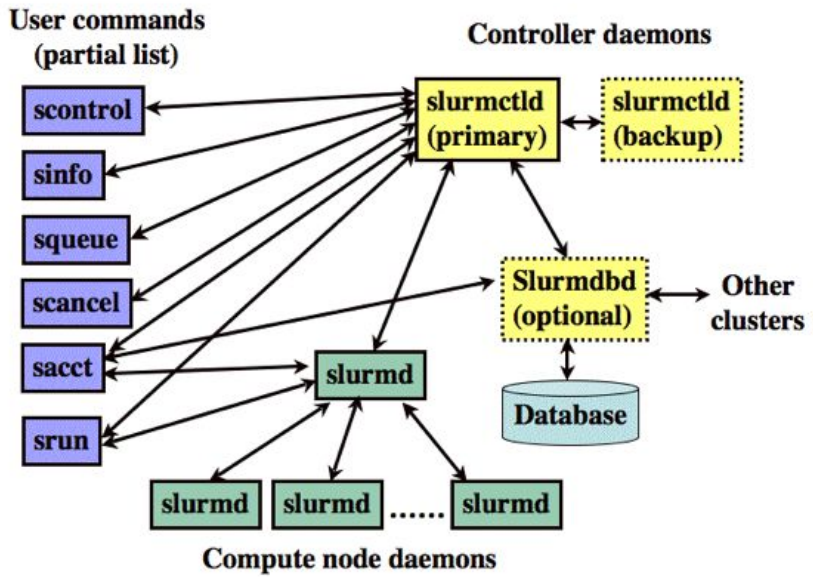
- Agent-less – No need for agent installation and management
- Secure – SSH based connection or direct connect
- Python and YAML based
- Highly flexible and configuration management of systems
- Large number of ready to use modules available for system management
- Custom modules can be added if needed
- Human readable and self-documenting
- Idempotent

NVIDIA.

SLURM

What is it?

- Simple Linux Utility for Resource Management
- Free and open source software
- HPC resource management and job scheduling system
- Simple to use and administer
- Used on 60% of the TOP500 supercomputers
- Highly scalable



Kubernetes vs. Slurm

Cloud vs. HPC

- Service oriented
- Containerized
- Meant to scale
- Distributes load
- Keeps applications running

- Batch job oriented
- Bare metal
- Efficiently allocates resources
- Fine grained user control
- Maximizes system utilization

NVIDIA.

Hybrid Strategy

Have your cake and eat it too.

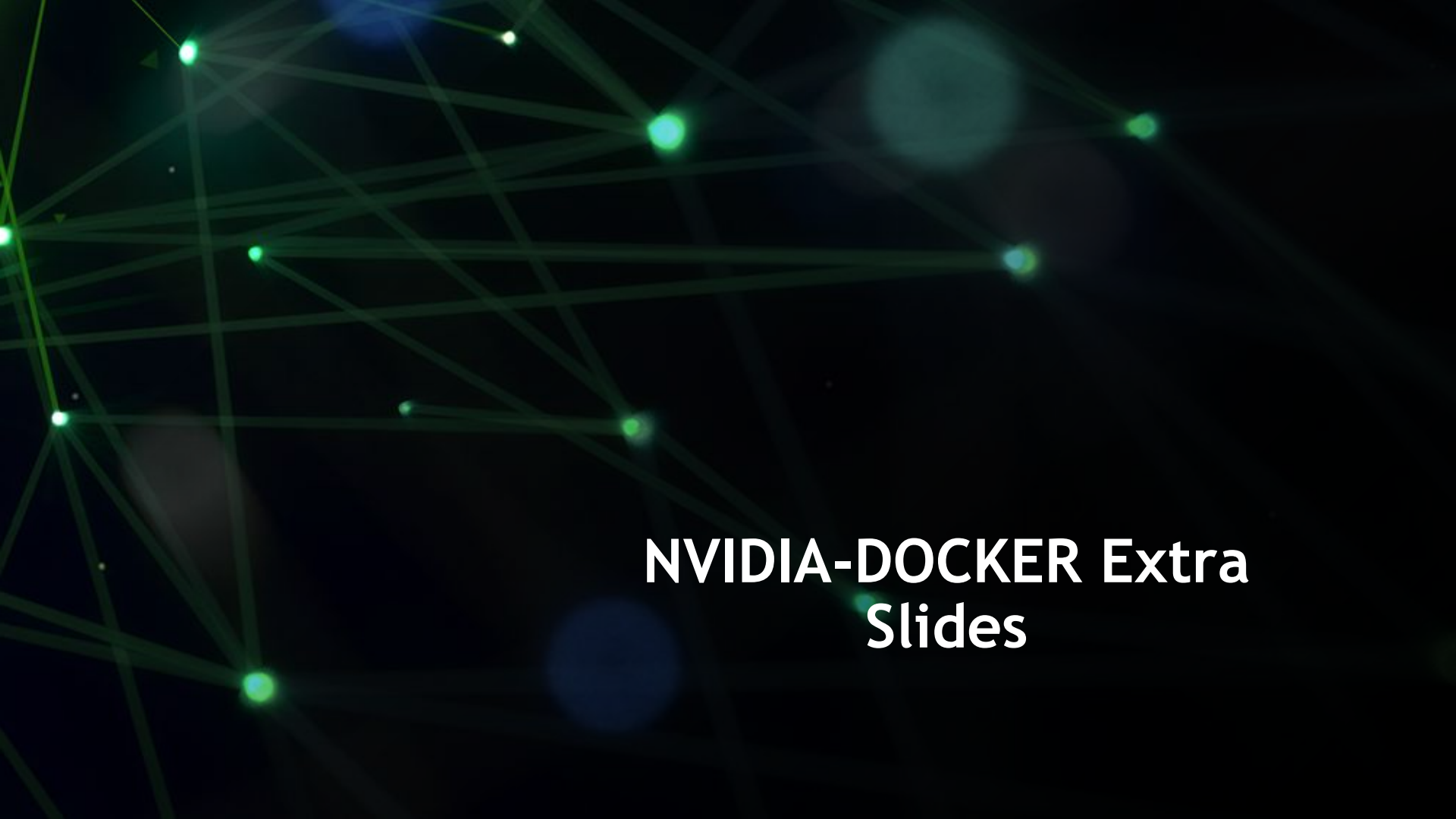
- Augment legacy HPC schedulers with new features
  - Cluster management services
  - Jupyter notebooks
  - Deep learning inference deployments (TensorRT)
- Keep data in the same place, no need to have separate clusters
- Free up deep learning researchers to do DL, not become devsecops/sysadmin
- Why not Slurm on Kubernetes?
  - Same reason HPC jobs may not be a good fit for Kubernetes & Containerization
    - Latency sensitive/multi-node comms/direct access to hardware
  - Slurm in containers on Kubernetes adds unneeded complexity

NVIDIA.

Hybrid Strategy - cont.

- Kubernetes manages cluster infrastructure
  - DHCP, PXE
  - Container registry/repository
  - APT repository
  - Metrics/monitoring/logging collection daemons on every node
  - Graphical interfaces for metrics
  - Alerting
  - Logging
- Kubernetes runs some workloads
  - DL inference tasks
  - Jupyter notebooks
  - Multi-GPU/Multi-node DL training with add-ons like tensorboard
- Slurm runs HPC workloads
  - Traditional non-containerized workloads
  - Containerized workloads via Singularity

NVIDIA.


NVIDIA-DOCKER Extra
Slides

Installing NVIDIA Docker 2.0

- NVIDIA driver and a supported version of Docker for your distribution
- Ubuntu 14.04/16.04/18.04, Debian Jessie/Stretch
- If you have nvidia-docker 1.0 installed, remove it and all existing GPU containers

```
docker volume ls -q -f driver=nvidia-docker | xargs -r -I{} -n1 docker ps -q -a -f volume={}
| \ xargs -r docker rm -f
sudo apt-get purge -y nvidia-docker
```

- Add the package repositories

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
  sudo apt-key add - \
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | \
  sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
```

- Install nvidia-docker2 and reload the Docker daemon configuration

```
sudo apt-get install -y nvidia-docker2
sudo pkill -SIGHUP dockerd
```

- Test nvidia-smi with the latest official CUDA image

```
docker run --runtime=nvidia --rm nvidia/cuda nvidia-smi
```

Installing NVIDIA Docker 2.0

- Enable the nvidia runtime as your default runtime on your node.

  Edit **`/etc/docker/daemon.json`**

```
{
    "default-runtime": "nvidia",
    "runtimes": {
        "nvidia": {
            "path": "/usr/bin/nvidia-container-runtime",
            "runtimeArgs": []
        }
    }
}
```

NVIDIA Docker Debugging

- TBD


NGC Extra Slides

LATEST NVIDIA FEATURES

Frameworks

- Monthly updates with latest Deep Learning Libraries
- Top network optimizations
- Volta Tensor Core updates

cuBLAS

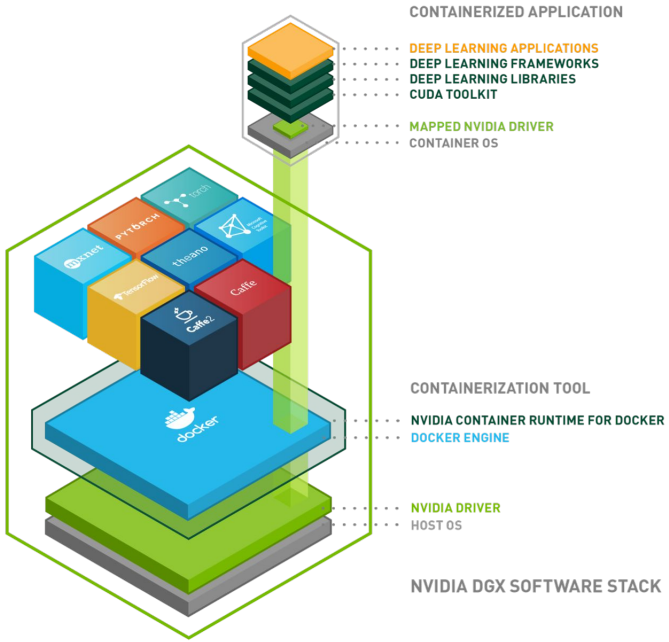
- Volta optimized GEMM and heuristics
- Improved RNN model specific GEMM performance for key sizes
- Monthly Releases

cuDNN

- Better performance for CNNs & RNNs
- Improved documentation (install guides & release notes)
- Better error logging
- New RNN/CNN functionalities
- Monthly releases

NCCL

- Added multi-node communication
- Low latency communication
- ROCE v1 and v2 support
- More collectives: allgatherv and reducescatterv



NVIDIA DEEP LEARNING FRAMEWORKS TEAM

OVERVIEW OF COMMUNITY INTERACTIONS

Open-source Framework

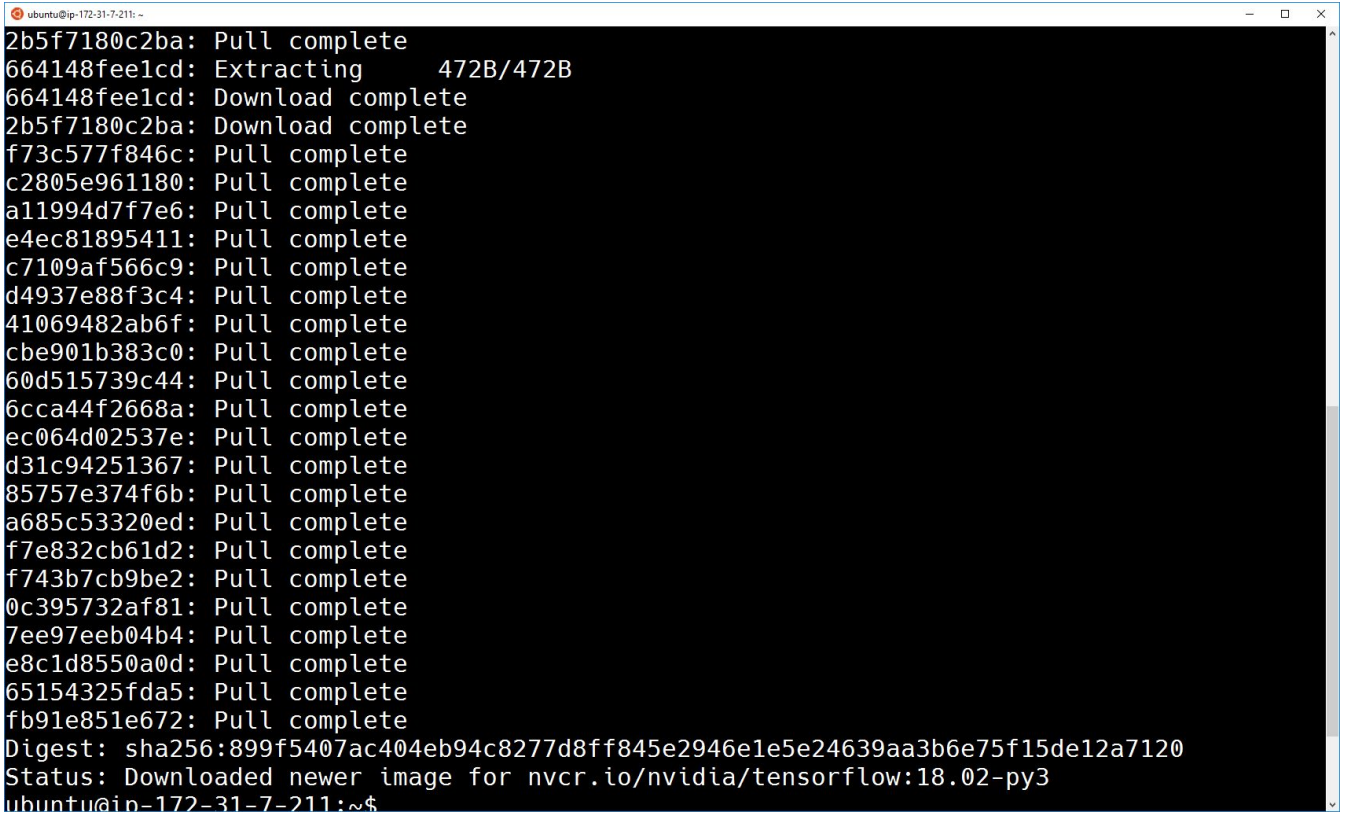
DGX and NGC Customers

Upstream work

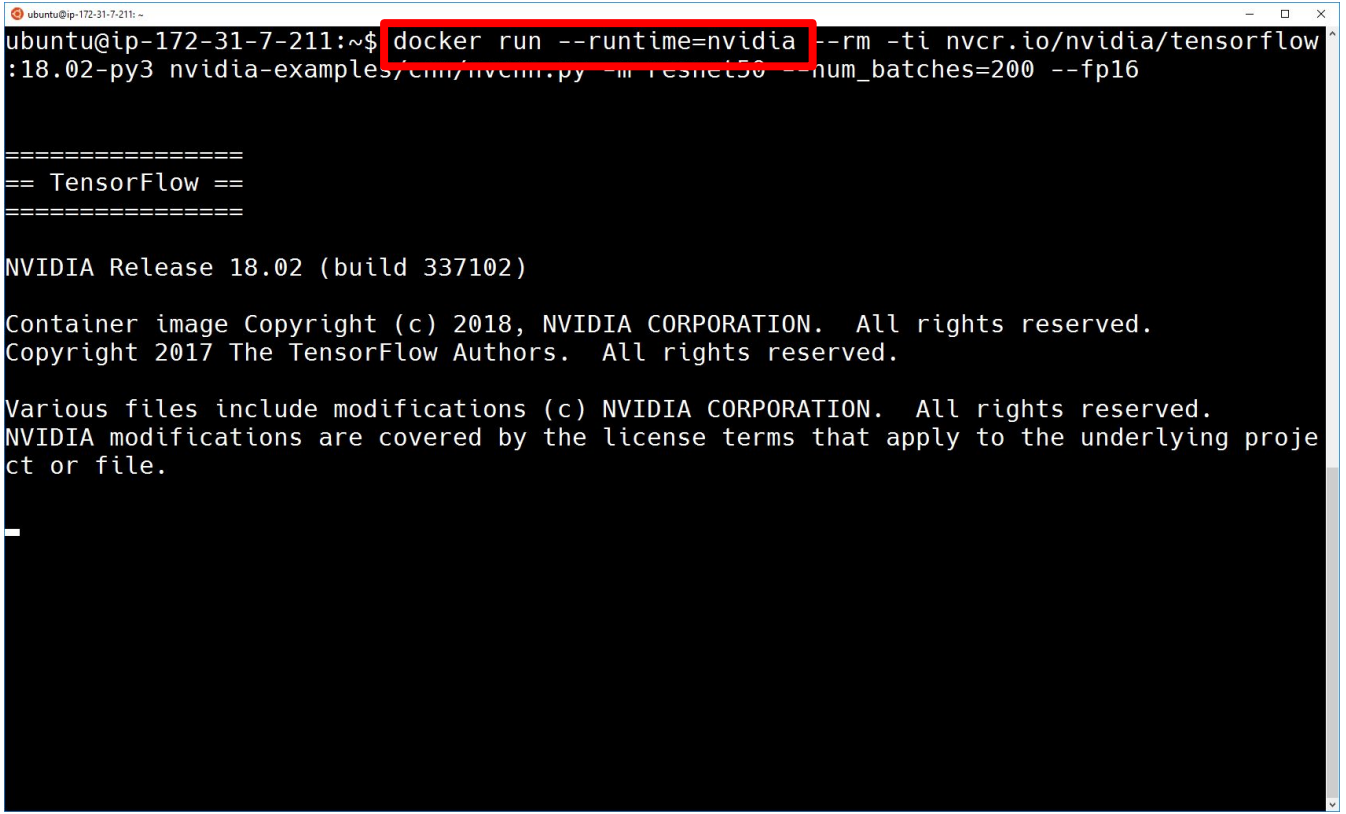
Containerize work

NVIDIA Deep Learning Frameworks

NGC Container Execution



NGC Container Execution



GPU ISOLATION

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

Use NVIDIA_VISIBLE_DEVICES to assign specific GPUs to the running container

Examples of GPU isolation:
Running nvidia-docker isolating specific GPUs by index

```
docker run --runtime=nvidia -e NVIDIA_VISIBLE_DEVICES=0,1 --rm
nvidia/cuda nvidia-smi
```

Running nvidia-docker isolating specific GPUs by UUID

```
docker run --runtime=nvidia -e NVIDIA_VISIBLE_DEVICES=GPU-fef8089b
nvcr.io/nvidia/cuda:9.0-cudnn7.1-devel-ubuntu16.04 nvidia-smi
```


Q&A

Misc. Questions/Answers

Can a Kubernetes Pod use resources from multiple nodes (DGX GPUs)?:

No, a single Pod will be limited to the available GPUs on each node, not the total in the system. K8S should be smart enough to minimize this issue, but it might still happen from time to time.

What Alerting is available with K8s:

If you install K8S using DeepOps it will come with Prometheus and Grafana as we demoed. In addition to this we install AlertManager which can be used to alert based on utilization or K8S job completion status.

Can I use K8S to split a GPU?:

No. The number of GPUs assigned to a pod must be an integer. These GPUs will not be usable by any other pod. Regardless, splitting GPUs for DL or ML workloads is not recommended by NVIDIA.

Can I use a VM as a worker node?:

Yes. A VM running on a hypervisor can be used as a K8S worker node. This has been less tested and will have some performance impacts. Depending on the setup, it may also restrict the availability of multi-GPU jobs.

Misc. Questions/Answers 2

Can I have a K8S pod that uses GPUs from multiple nodes?:

No. A K8S pod is restricted to the number of GPUs available on a single node. If, for example, a job requires 128 GPU it must be setup as a muli-node job using Horovod, Dask, etc. to run.

If there is a security issue in a NGC container will there be a patch?:

No. If a security issue is found in a NGC container an alert will be sent out and a new image recommended.

How do I handle storage that only certain users have rights to?

TBD.

What performance benchmarks do we do for NGC image regression?

TBD

Misc. Questions/Answers 3

How do we use and expand GPU monitoring?:

TBD

What are best practices around tracking GPU utilization?:

TBD

How do I share resources in a team or across teams?:

TBD

How do I submit batch jobs?:

TBD

How do I use Slurm alongside K8S?:

TBD