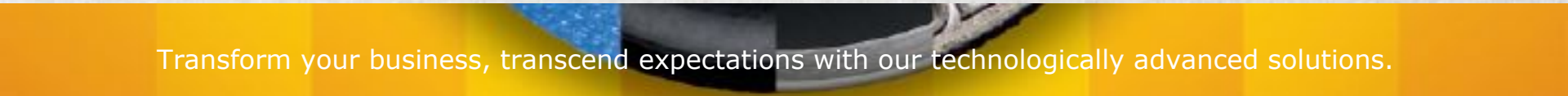
NTT Communications

Transform. Transcend.

# Container as a Service on GPU Cloud: Our Decision among K8s, Mesos, Docker Swarm and OpenStack Zun

Yoshifumi Sumida
Chihiro Yokoyama
Xiaojing Zhang



Transform your business, transcend expectations with our technologically advanced solutions.

# About us


Chihiro Yokoyama
c.yokoyama@ntt.com

Technology Development Div.
Software Engineer


R&D on IaaS, Container
Interested in DevOps, COE


Yoshifumi Sumida
y.sumida@ntt.com

Technology Development Div.
Software Engineer


R&D on IaaS, Container
Interested in FaaS, Networking

# Agenda

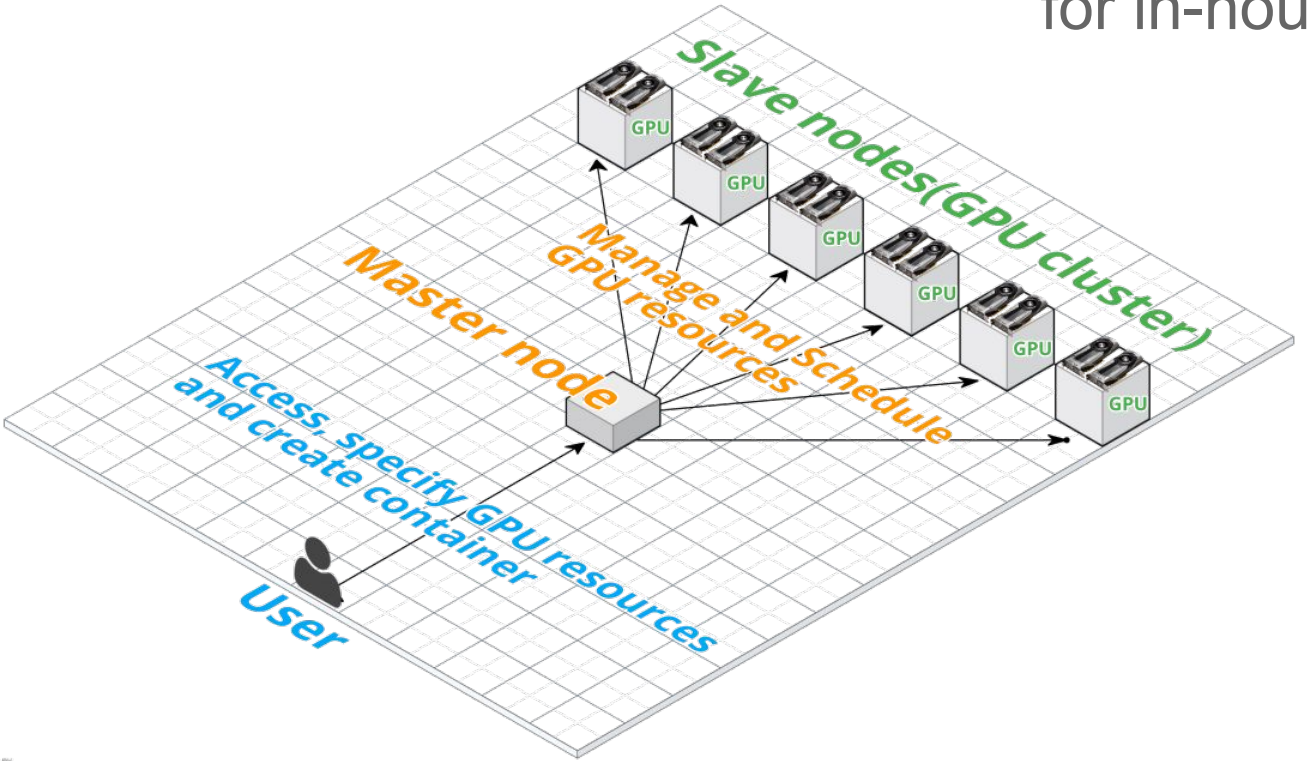
1. Goal, Motivation and Requirements
2. Why GPU?
   Why we provide GPU resources as containers?
3. Comparison of Container Related Tools
4. How we realized GPU Container as a Service?

# Agenda

1. **Goal, Motivation and Requirements**
2. Why GPU?
   Why we provide GPU resources as containers?
3. Comparison of Container Related Tools
4. How we realized GPU Container as a Service?

# Our goal

Provide simple Container as a Service on GPU cluster
for in-house users.



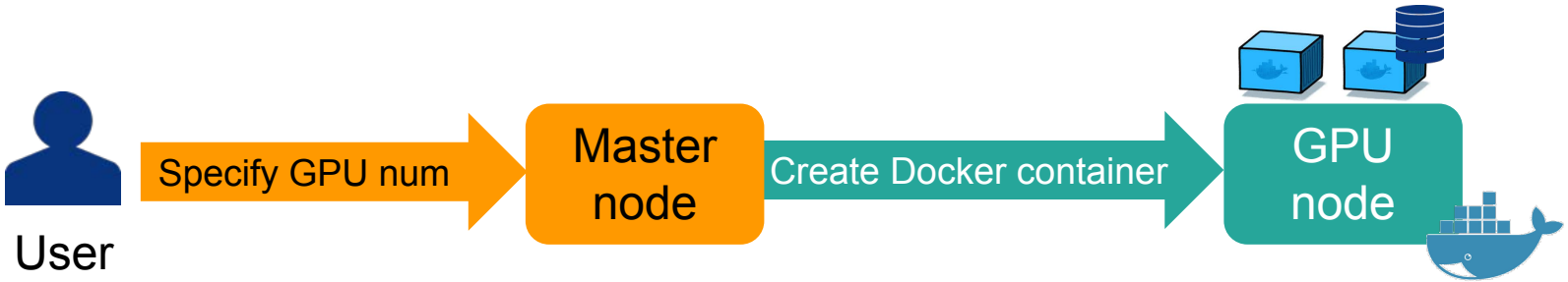
# Motivation

- Manage various GPU servers as unified resource cluster
  - Our GPU cluster should be able to:
    - Composed of different NVIDIA GPU series (e.g. K2, K10, P100…)
      - This could be a problem because of nvidia-driver version difference(Now, we have the solution)

- Provide our GPU cluster as Cloud Service
  - More and more in-house users would like to use GPU resources
  - They would like to focus on their tasks, not the environment provision

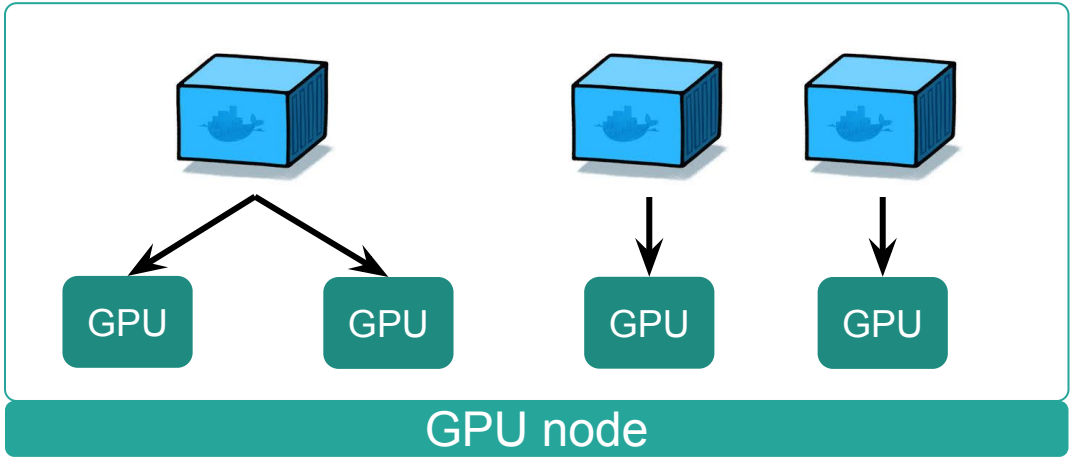
# Requirements 1/3

- User side
  - Deploy GPU container easily
    - e.g. only specify the number of GPUs
  - Use Docker container



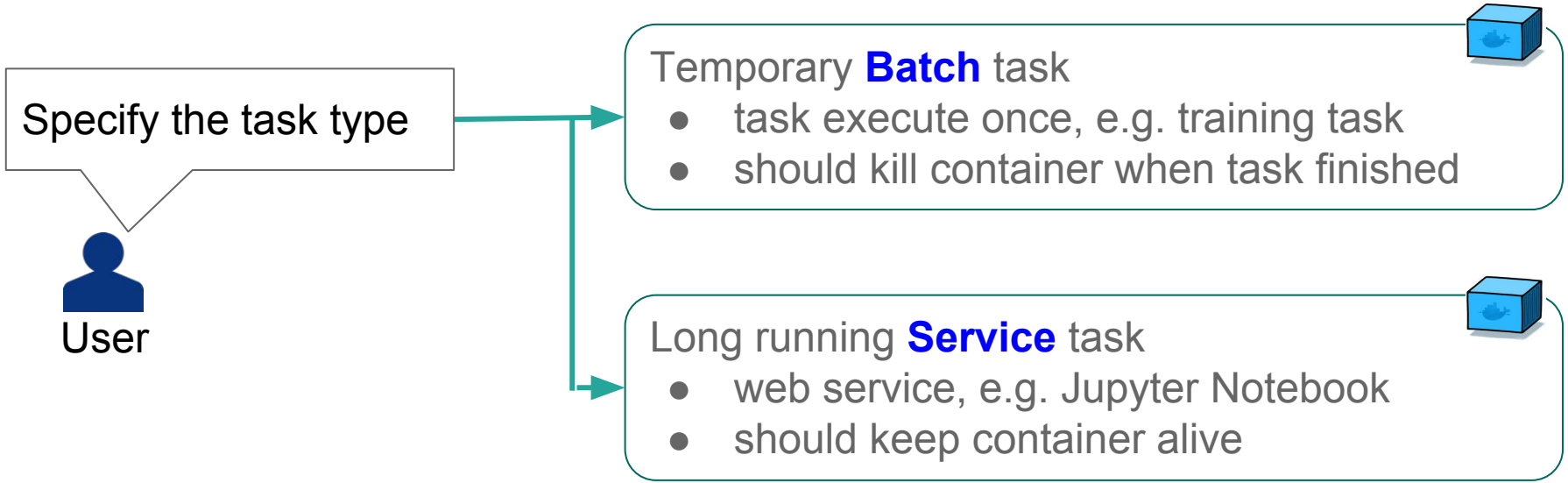
# Requirements 2/3

- Provider Side
  - Assure GPU isolation
    - Avoid attaching GPUs in use to new containers
    - Each container can see only its own GPUs



# Requirements 3/3

- Provider side
  - Distinguish container's lifecycle according to task types
  - Effectively utilize resources



# Agenda

1. Goal, Motivation and Requirements

2. **Why GPU?**

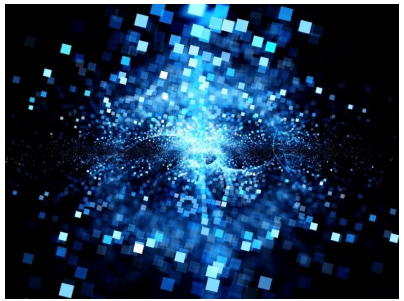
   **Why we provide GPU resources as containers?**

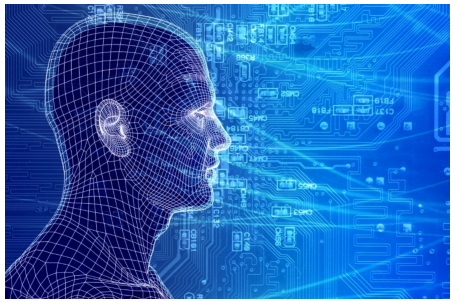
3. Comparison of Container Related Tools

4. How we realized GPU Container as a Service?

# Why GPU?

- GPU is in high demand for various fields or workloads
  - Machine learning
  - Big Data Analysis





- Cloud Providers as AWS, Azure, and GCP begin to offer GPU instances

# Why we provide GPU resources as containers?

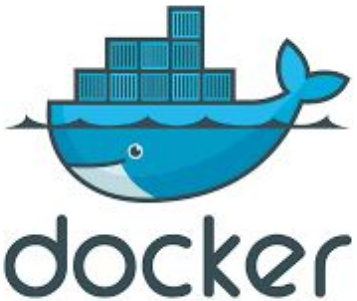
- At first, we provided GPU instances (VMs) on our private cloud
  - On OpenStack
  - Utilize PCI Passthrough to use the physical devices inside KVM
    - Select PCI Passthrough(not vGPU) because of simple provision
- However, VM based GPU cloud has the following problems

1. User needs to install appropriate nvidia-driver every time creating VM.
2. Cannot use NVIDIA Management Library (NVML) for GPU monitoring because of binding DUMMY driver to host's GPUs
3. Once users created specific environment, it's difficult to run various applications on it

# Why we provide GPU resources as containers?

Container Technology, e.g. Docker, can resolve these problems.

- Docker
  - Popular containerization tool
  - Users can make their own images
  - Isolation of individual devices



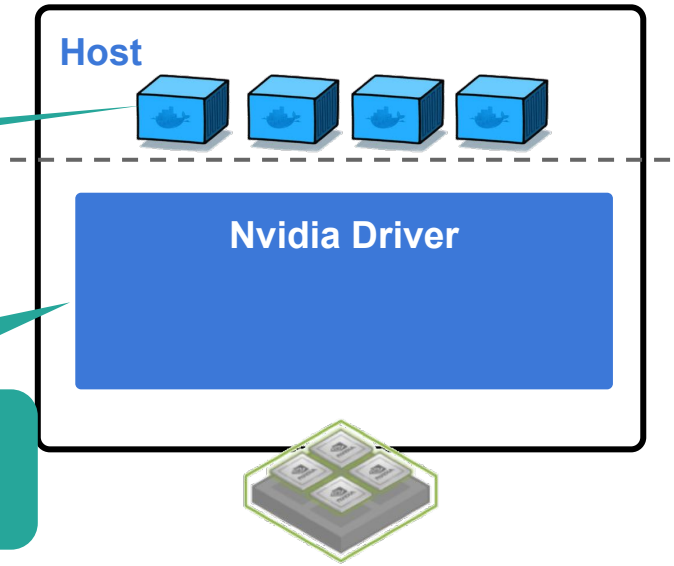
# Why we provide GPU resources as containers?

Utilize Docker...

- Once providers install nvidia-driver on host nodes, all users have to do is to **create and destroy containers**
- normal “nvidia-driver” for GPUs simply installed on host nodes

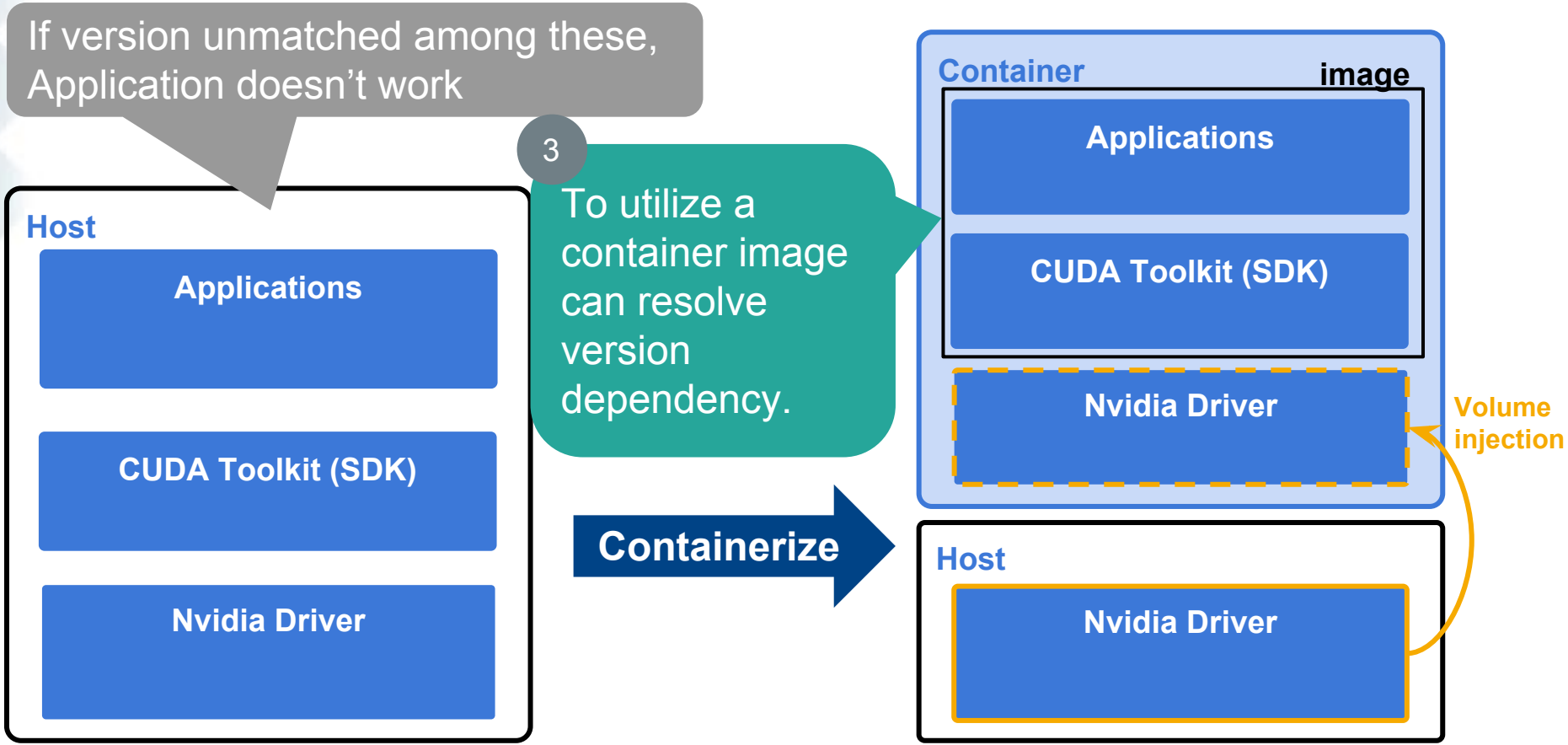
1. Users don’t need to consider which nvidia-driver version is required

2. Can monitor GPUs status using NVIDIA Management Library (NVML)



# Why we provide GPU resources as containers?

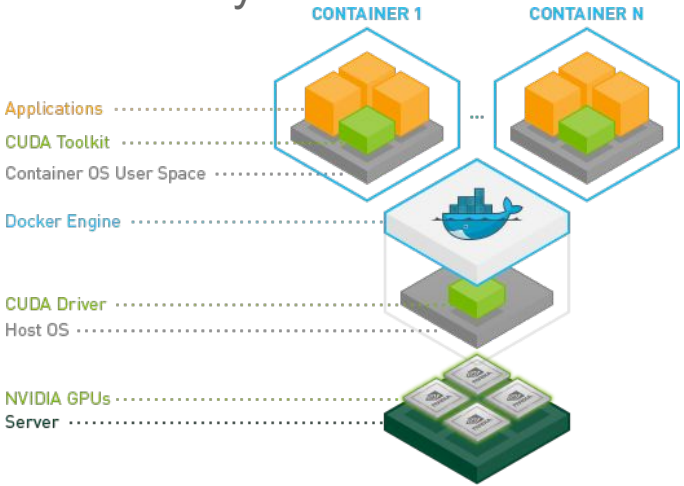
If version unmatched among these, Application doesn't work

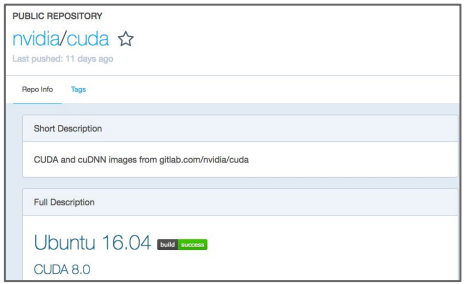


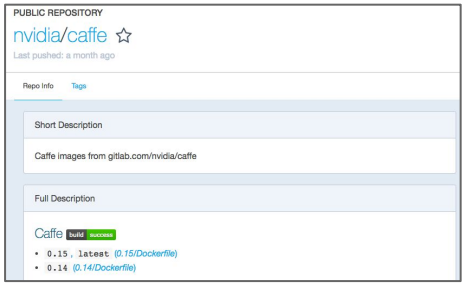
# Why we provide GPU resources as containers?

## Nvidia-docker can resolve previous problems efficiently.

- nvidia-docker
  - Docker's wrapper to use and isolate GPUs inside docker containers
  - Ready-to-use CUDA and various DL framework images



# nvidia-docker

- Wrap *docker run* and *docker create* commands
  - *e.g.* `$ nvidia-docker run --rm nvidia/cuda nvidia-smi`
  - Add docker cli options to mount nvidia driver files

**Docker cli option:**
--volume-driver=nvidia-docker
--volume=nvidia_driver_xxx.xx:/usr/local/nvidia:ro
--device=/dev/nvidiactl --device=/dev/nvidia-uvm
--device=/dev/nvidia-uvm-tools
--device=/dev/nvidia0

- *nvidia-docker plugin* can detect driver files for the options
  - Finds all nvidia libraries and binaries on the host

# nvidia-docker

- Check whether the image is compatible with the host driver version
  - In Dockerfile

```
LABEL com.nvidia.cuda.version="${CUDA_VERSION}"
```

If version unmatch occurs…

```
$ nvidia-docker run --rm nvidia/cuda
nvidia-docker | 2016/04/21 21:41:35 Error: unsupported CUDA version:
                                        driver 7.0 < image 8.0
```

# Agenda

1. Goal, Motivation and Requirements
2. Why GPU?
   Why we provide GPU resources as containers?
3. Comparison of Container Related Tools
4. How we realized GPU Container as a Service?

# Comparison of Container Related Tools

We surveyed and compared some tools

and determine how to provide our Container as a Service on GPU cluster.

- nvidia-docker
- Docker Swarm / swarm mode
- OpenStack Zun
- Mesos
- kubernetes

# Comparison of Container Related Tools

Comparison point of view

| | Manage GPU cluster | Specify the number of GPUs | GPU isolation | Docker Support | Exec Batch task |
|---|---|---|---|---|---|
| nvidia-docker | | | | | |
| DockerSwarm | | | | | |
| OpenStackZun | | | | | |
| mesos | | | | | |
| Kubernetes | | | | | |

# Comparison of Container Related Tools

Comparison point of view

| | Manage GPU cluster | Specify the number of GPUs | GPU isolation | Docker Support | Exec Batch task |
|---|---|---|---|---|---|
| nvidia-docker | | | | | |
| Docke | | | | | |
| Open | | | | | |
| meso | | | | | |
| Kubernetes | | | | | |

Users can specify the number of GPUs and allocate multiple GPUs(not only single) to containers.

# Comparison of Container Related Tools

Comparison point of view

| | Manage GPU cluster | Specify the number of GPUs | GPU isolation | Docker Support | Exec Batch task |
|---|---|---|---|---|---|
| nvidia-docker | | | | | |
| D | | | | | |
| O | | | | | |
| m | | | | | |
| Kubernetes | | | | | |

Each container process can see only its own GPUs.

The GPUs in use will not be attached to new containers.

# Comparison of Container Related Tools

Comparison point of view

| | Manage GPU cluster | Specify the number of GPUs | GPU isolation | Docker Support | Exec Batch task |
|---|---|---|---|---|---|
| nvidia-docker | | | | | |
| DockerSwarm | | | | | |
| | | | | | |

The container process can be killed automatically

when user task inside the container are successfully terminated.

# nvidia-docker

- Specify the number of GPUs | GPU isolation
  - Use NV_GPU option
    e.g. ***$ NV_GPU=0,1 nvidia-docker run --rm -it nvidia/cuda nvidia-smi***
  - Can isolate specified GPUs in the container

**Host**

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.57                 Driver Version: 367.
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage |
|===============================+======================+======================|
|   0  GRID K2             Off  | 0000:07:00.0     Off |
| N/A   39C    P8    18W / 117W |      0MiB /  4036MiB |
|-------------------------------+----------------------+----------------------|
|   1  GRID K2             Off  | 0000:08:00.0     Off |
| N/A   36C    P8    17W / 117W |      0MiB /  4036MiB |
|-------------------------------+----------------------+----------------------|
|   2  GRID K2             Off  | 0000:90:00.0     Off |
| N/A   38C    P8    17W / 117W |      0MiB /  4036MiB |
|-------------------------------+----------------------+----------------------|
|   3  GRID K2             Off  | 0000:91:00.0     Off |
| N/A   33C    P8    17W / 117W |      0MiB /  4036MiB |
```

**Container**

```
root@sv51u-maku:~# NV_GPU=0,1 nvidia-docker run --rm  -it
Thu Mar 30 08:38:42 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.57                 Driver Version: 367.5
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage |
|===============================+======================+======================|
|   0  GRID K2             Off  | 0000:07:00.0     Off |
| N/A   39C    P8    18W / 117W |      0MiB /  4036MiB |
+-------------------------------+----------------------+----------------------+
|   1  GRID K2             Off  | 0000:08:00.0     Off |
| N/A   36C    P8    17W / 117W |      0MiB /  4036MiB |
```

# nvidia-docker

- GPU isolation
  - **BUT,** different containers could get the same GPUs
  - User may create container attached with busy GPUs

```
root@sv51u-maku:~# docker ps
CONTAINER ID        IMAGE               COMMAND
3020131b1c95        nvidia/cuda         "/bin/bash"
608451a2b8ff        nvidia/cuda         "/bin/bash"
```

Container A

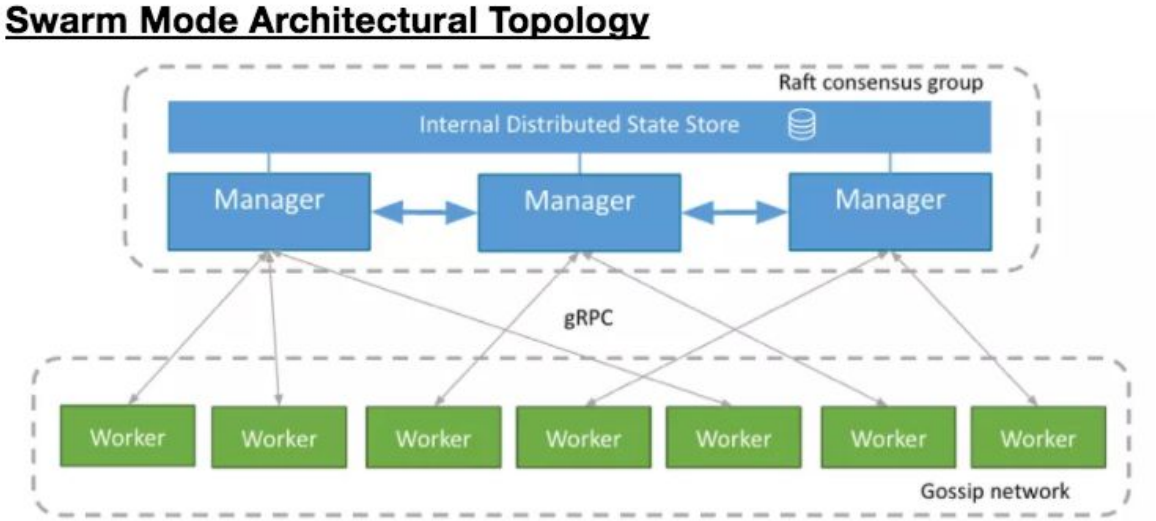
```
root@sv51u-maku:~# docker exec -it 3020131b1c95 nvidia-smi
Thu Mar 30 08:30:53 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.57                 Driver Version: 367.57                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GRID K2             Off  | 0000:07:00.0     Off |                  Off |
| N/A   39C    P8    18W / 117W |      0MiB /  4036MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GRID K2             Off  | 0000:08:00.0     Off |                  Off |
| N/A   36C    P8    17W / 117W |      0MiB /  4036MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

Container B

```
root@sv51u-maku:~# docker exec -it 608451a2b8ff  nvidia-smi
Thu Mar 30 08:30:55 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.57                 Driver Version: 367.57                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GRID K2             Off  | 0000:07:00.0     Off |                  Off |
| N/A   39C    P8    18W / 117W |      0MiB /  4036MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GRID K2             Off  | 0000:08:00.0     Off |                  Off |
| N/A   36C    P8    17W / 117W |      0MiB /  4036MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

# Docker Swarm / swarm mode

- Docker Swarm is a native clustering tool for Docker
- Docker 1.12 and later with build in “swarm mode”, orchestration feature


https://blog.docker.com/2016/07/docker-built-in-orchestration-ready-for-production-docker-1-12-goes-ga/

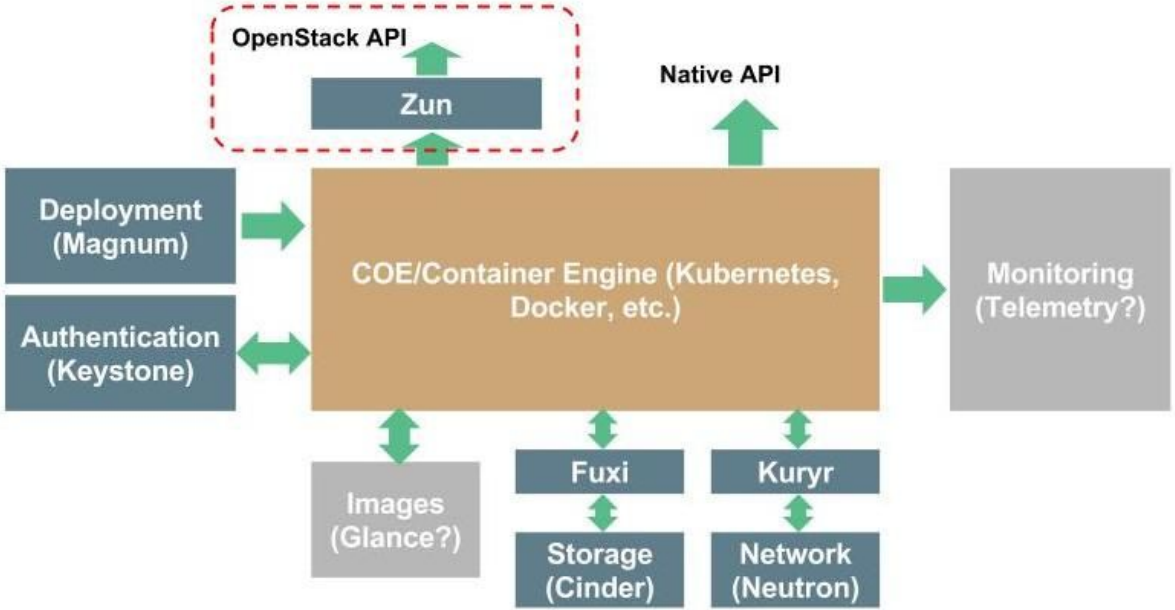
# Docker Swarm / swarm mode

- Docker itself cannot manage GPU resources like CPU
  - Current discussing here
    - https://github.com/docker/docker/issues/23917
- Has not been supported by nvidia-docker
  - Cannot inject necessary nvidia-driver files into containers in GPU cluster

**Docker Swarm / swarm mode doesn't satisfy "management of GPU cluster"**

# OpenStack Zun

Zun is a Container Management service for OpenStack.



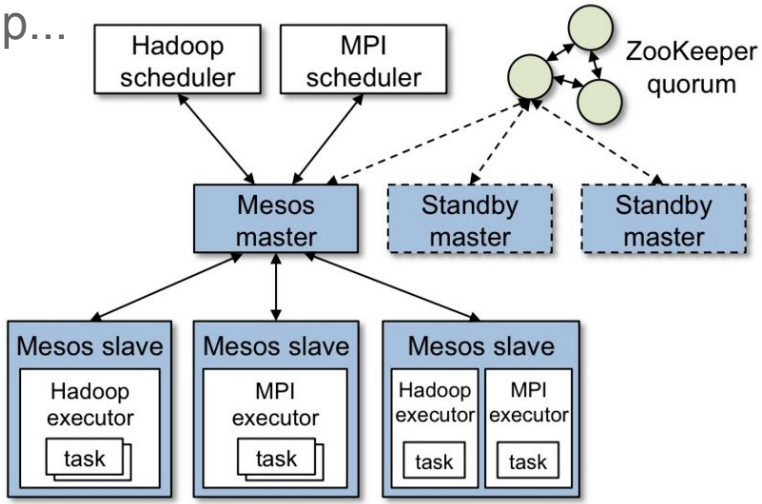

- Provide basic container operations (i.e. CRUD) within OpenStack

# OpenStack Zun

- However...
  - **GPU resource is not supported now**
    - Zun pass the “CpuShares” and “Memory” parameters to Docker
    - Use Docker API -> Not support GPU

# Mesos

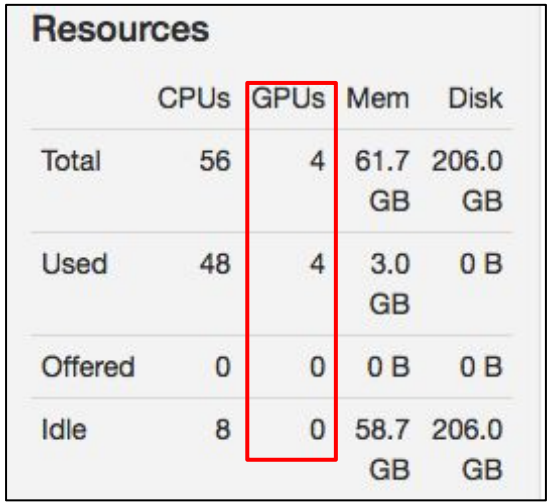
- What is Apache mesos?
  - Cluster manager
  - Provide efficient resource isolation and sharing
  - Control Mesos master across distributed applications or frameworks
    - e.g. Marathon, Chronos, Hadoop...

# Mesos

- Mesos GPU support status
  - Version 1.0.0 added Nvidia GPU support
  - **Manage GPU resource as same as CPU, memory, disk**
  - Now, Nvidia GPU support is only available for Mesos containerizer

**(not Docker Containerizer)**



Resources

| | CPUs | GPUs | Mem | Disk |
|---|---|---|---|---|
| Total | 56 | 4 | 61.7 GB | 206.0 GB |
| Used | 48 | 4 | 3.0 GB | 0 B |
| Offered | 0 | 0 | 0 B | 0 B |
| Idle | 8 | 0 | 58.7 GB | 206.0 GB |

# Mesos

## Major mesos framework's GPU and Docker support

| Task Type | Frameworks | GPU | Docker | GPU + Docker |
|---|---|---|---|---|
| Batch | Chronos | ✗ | ✔ | N/A |
| | Metronome | ✗ | ✔ | N/A |
| Service | Aurona | ✔ | ✔ | ✗ |
| | Marathon | ✔ | ✔ | ✗ |

# Mesos

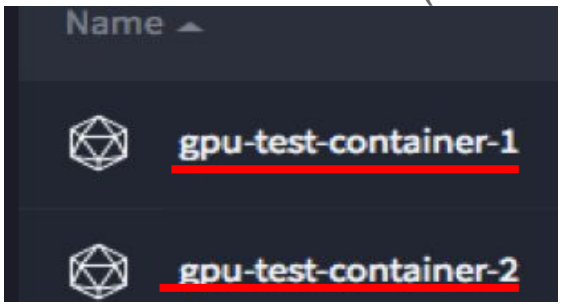
- Specify the number of GPUs (mesos v1.1.1, marathon v1.3.10)

```
{
  "id": "gpu-test-container",
  "cmd": "while [ true ]; do nvidia-smi;
  "cpus": 16,
  "mem": 1024,
  "disk": 0,
  "instances": 1,
  "gpus": 2,
  "container": {
    "type": "MESOS",
    "docker": {
      "image": "nvidia/cuda",
      "parameters": [],
      "network": "HOST",
      "forcePullImage": true
```

```
root@sv51u-maku:/var/lib/mesos/slaves/0d44e406-a3b4-47cc-90
utors/gpu-test-container1.ccb678fe-fef2-11e6-b86d-fa163e1ce
Thu Mar  2 04:46:39 2017
+-----------------------------------------------------------
| NVIDIA-SMI 367.57                 Driver Version: 367.57
|-------------------------------+----------------------+---
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Vo
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GP
|===============================+======================+===
|   0  GRID K2             Off  | 0000:07:00.0     Off |
| N/A   38C    P8    18W / 117W |      0MiB /  4036MiB |
+-------------------------------+----------------------+---
|   1  GRID K2             Off  | 0000:08:00.0     Off |
| N/A   34C    P8    17W / 117W |      0MiB /  4036MiB |
+-------------------------------+----------------------+---
```

# Mesos

- GPU isolation (mesos v1.1.1, marathon v1.3.10)




```
oot@sv51u-maku:/var/lib/mesos/slaves/0d44e406-a3b4-47c
tors# cat gpu-test-container-1.289da203-ff04-11e6-b86d

 No running processes found

hu Mar  2 04:55:50 2017

NVIDIA-SMI 367.57                 Driver Version: 367

GPU  Name        Persistence-M| Bus-Id        Disp.A
Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage

  0  GRID K2             Off  | 0000:07:00.0     Off
N/A   39C    P8    18W / 117W |      0MiB /  4036MiB
```

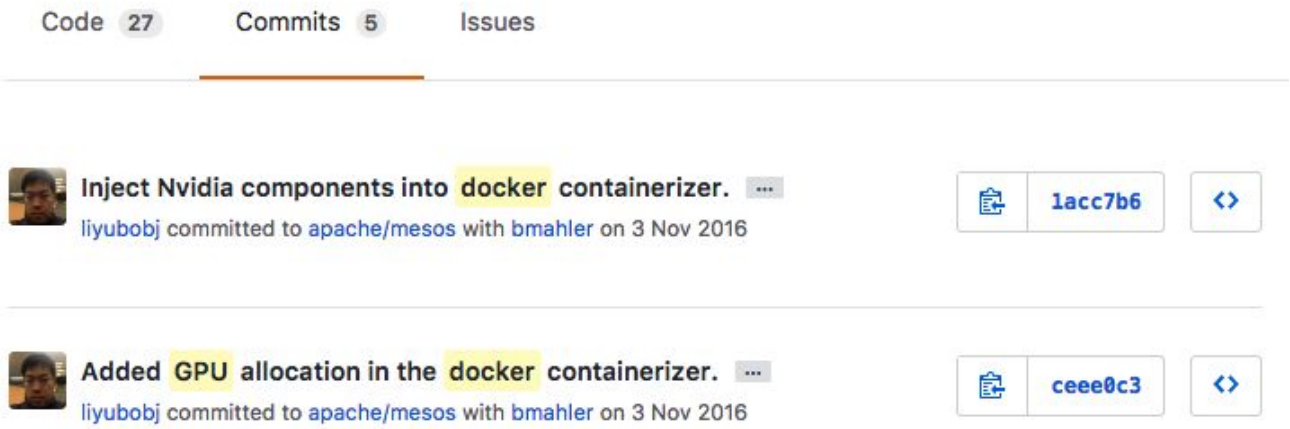
```
oot@sv51u-maku:/var/lib/mesos/slaves/0d44e406-a3b4-47c
tors# cat gpu-test-container-2.5079cba4-ff04-11e6-b86d

 No running processes found

hu Mar  2 04:55:57 2017

NVIDIA-SMI 367.57                 Driver Version: 367

GPU  Name        Persistence-M| Bus-Id        Disp.A
Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage

  0  GRID K2             Off  | 0000:08:00.0     Off
N/A   34C    P8    17W / 117W |      0MiB /  4036MiB
```

# Mesos

- Docker Support
  - Now, Nvidia GPU support is only available for **the Mesos containerizer**
    - **Not Docker containerizer**
  - Mesos containerizer supports Docker images
    - **However, cannot use Docker CLI**

# Mesos - next version

- Mesos may support Docker Containerizer with GPU in next v1.2

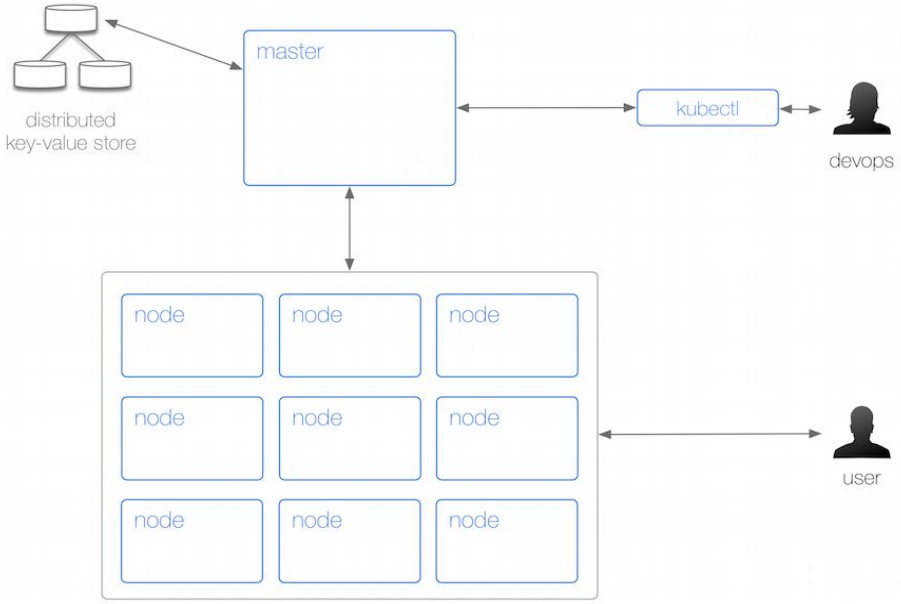



- However, Docker support with GPU in “mesos frameworks” like marathon, seems not progressing much...

# kubernetes

- What’s kubernetes(k8s) ?
  - Container orchestration engine from Google
  - Cluster-management, scaling, Storage orchestration and Batch execution...



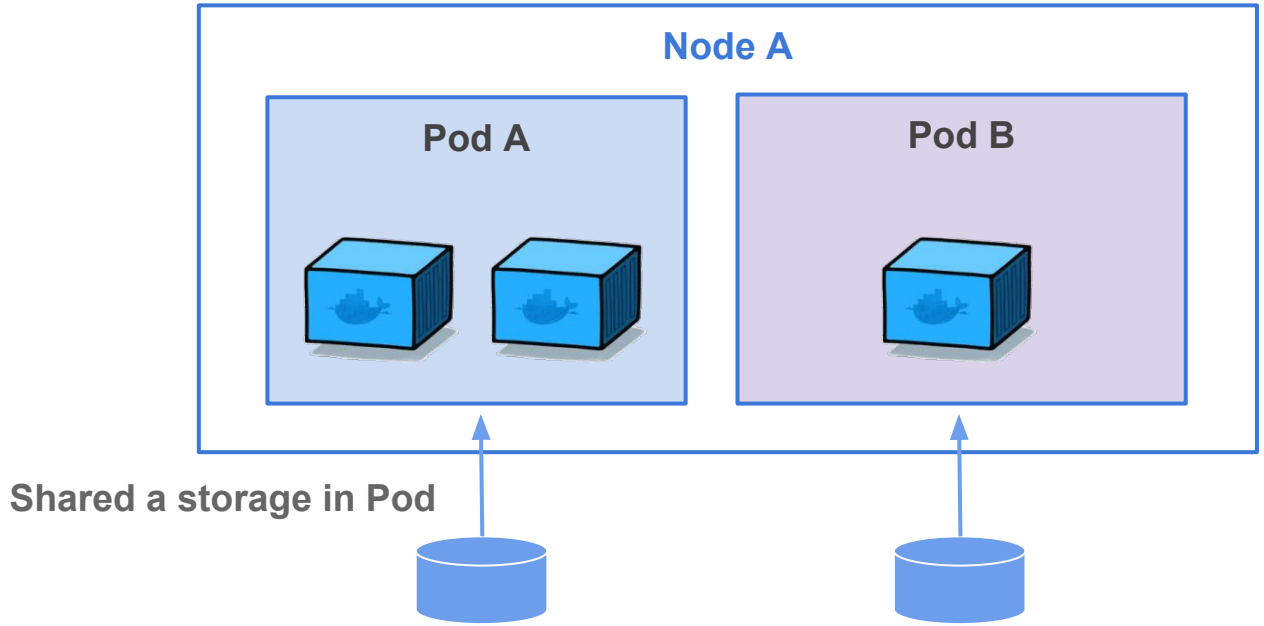



# kubernetes

- k8s's original concept of container management
  - Manage containers as a group of one or more containers (called "Pod")
    - Share a storage and options about how to run the containers in a Pod
  - How to deploy containers
    - **Pod**
    - **Controller**

# kubernetes

- **Pod**
  - Simply deploy one or more containers
  - Each volumes are shared in each Pods



# kubernetes

- **Controller**
  - Define how to create and manage pods with additional functions
  - Various kinds of controllers
    - e.g. Jobs, Replica Sets, Deployments, Stateful Sets...

# kubernetes

- How to manage container lifecycle
  - Generally, prepare ***Manifest file***(yaml or json)
    - Define the specification of container
      - e.g. kind, Container's name, image
  - Then, use this manifest file via **CLI** or **WebUI** to create / delete containers

Example: nginx.yaml

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

# kubernetes

- **Kubernetes GPU support status**
  - **v1.3.x~ added Nvidia GPU Scheduling support *experimentally***
    - Cannot assign multiple GPUs to one container
    - Each container cannot occupy its own GPUs(no GPU isolation)
  - **v1.6.x~ supports Nvidia GPU Scheduling *officially***
    - Improved GPU Scheduling
      - Solved the above problems on v1.3.x~v1.5.x
    - Can detect the number of GPUs in the node automatically

**Use this!**

# kubernetes

- **Specify the number of GPUs (k8s v1.6.1)**

Write the number of GPUs in manifest

```
resources:
  limits:
    alpha.kubernetes.io/nvidia-gpu: 2
  requests:
    alpha.kubernetes.io/nviida-gpu: 2
command: ["/bin/bash", "-c"]
args: ["nvidia-smi" && tail -f /dev/null]
```

**In Container**

```
+-----------------------------------------------------------------------------+
|                         Driver Version: 375.39                              |
|-------------------------------+----------------------+----------------------+
|ence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K10.G2.8GB    Off  | 0000:00:05.0     Off |                    0 |
| N/A   39C    P8    17W / 117W |      0MiB /  3527MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K10.G2.8GB    Off  | 0000:00:06.0     Off |                    0 |
| N/A   41C    P8    18W / 117W |      0MiB /  3527MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

# kubernetes

- GPU isolation (k8s v1.6.1)

```
$ kubectl get pods -a -o wide
NAME                READY     STATUS    RESTARTS   AGE       IP              NODE
nvidia-smi-47vpm    1/1       Running   0          1m        10.233.78.15    sv51u-maku
nvidia-smi-d88z9    1/1       Running   0          1m        10.233.78.16    sv51u-maku
```

```
$ kubectl logs nvidia-smi-47vpm
Mon Apr 10 15:55:20 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.57                 Driver Version: 367.
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GRID K2             Off  | 0000:07:00.0     Off |                  Off |
| N/A   38C    P8    18W / 117W |      0MiB /  4036MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

```
$ kubectl logs nvidia-smi-d88z9
Mon Apr 10 15:55:22 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.57                 Driver Version: 367.
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GRID K2             Off  | 0000:90:00.0     Off |                  Off |
| N/A   38C    P8    18W / 117W |      0MiB /  4036MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

Can attach different GPUs to different containers

# kubernetes

- **Run the batch task (k8s v1.6.1) - 1/2 -**

Define kind and the number of GPUs

```
apiVersion: batch/v1
kind: Job
  ...
  limits:
    alpha.kubernetes.io/nvidia-gpu: 2
  requests:
    alpha.kubernetes.io/nviida-gpu: 2
```

## Get Container Status

```
$ kubectl get pods
NAME           READY     STATUS    RESTARTS   AGE
nbody-swsdj    1/1       Running   0          5s
$ kubectl get jobs
NAME       DESIRED   SUCCESSFUL   AGE
nbody      1         0            7s
```

## In Container

```
root@nbody-swsdj:/# nvidia-smi

~ +-----------------------------------------------------+
~ |         Driver Version: 375.39                      |
~ |----+----------------------+----------------------+
~ | ce-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
~ | /Cap|         Memory-Usage | GPU-Util  Compute M. |
~ |====+======================+======================|
~ | ff  | 0000:00:05.0     Off |                    0 |
~ | 17W |     37MiB /  3527MiB |     100%    Default |
~ |----+----------------------+----------------------+
~ | ff  | 0000:00:06.0     Off |                    0 |
~ | 17W |     37MiB /  3527MiB |     100%    Default |
~ +----+----------------------+----------------------+
```

# Run the batch task (k8s v1.6.1) - 2/2 -

Pod status at the initial state(after just creating pod)

```
$ kubectl get pods
NAME           READY     STATUS    RESTARTS   AGE
nbody-swsdj    1/1       Running   0          5s
$ kubectl get jobs
NAME       DESIRED   SUCCESSFUL   AGE
nbody      1         0            7s
```

After completed processes in Pod, It's terminated automatically

```
$ kubectl get pods -a -o wide
NAME           READY     STATUS      RESTARTS   AGE
nbody-swsdj    0/1       Completed   0          2m
$ kubectl get jobs
NAME       DESIRED   SUCCESSFUL   AGE
nbody      1         1            2m
```

# Comparison Result

| | Manage GPU cluster | Specify the number of GPUs | GPU isolation | Docker Support | Exec Batch task |
|---|---|---|---|---|---|
| nvidia-docker | ✗ | ✔ | ✗ | ✔ | ✗ |
| DockerSwarm | ✗ | ✗ | ✗ | ✔ | ✗ |
| OpenStackZun | ✗ | ✗ | ✗ | ✔ | ✗ |
| mesos | ✔ | ✔ | ✔ | ✗ | ✔ |
| Kubernetes | ✔ | ✔ | ✔ | ✔ | ✔ |

# Agenda

1. Goal, Motivation and Requirements
2. Why GPU?
   Why we provide GPU resources as containers?
3. Comparison of Container Related Tools
4. **How we realized GPU Container as a Service?**

# How we realize GPU Container as a Service?

Based on results of our comparison...

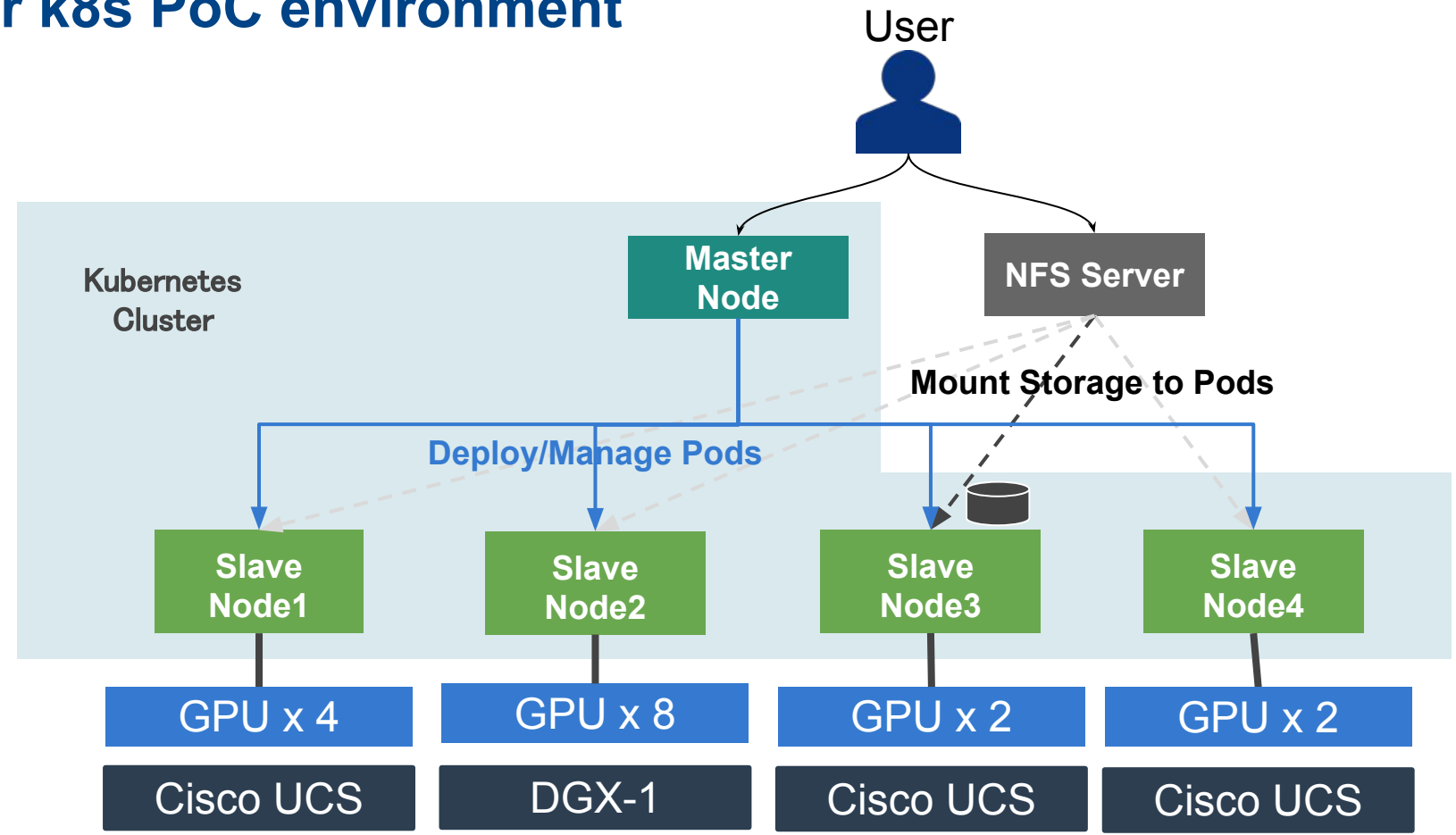
we choose **kubernetes** to provide GPU Container as a Service

- Until recently, Mesos has been superior to other tools to satisfy our requirements
  - GPU isolation is better than other tools ✔
  - No Docker support(this is a fatal point) ✗

- However,

  kubernetes have already had **Docker support** and gained **better GPU isolation at v1.6.x**

# Our k8s PoC environment



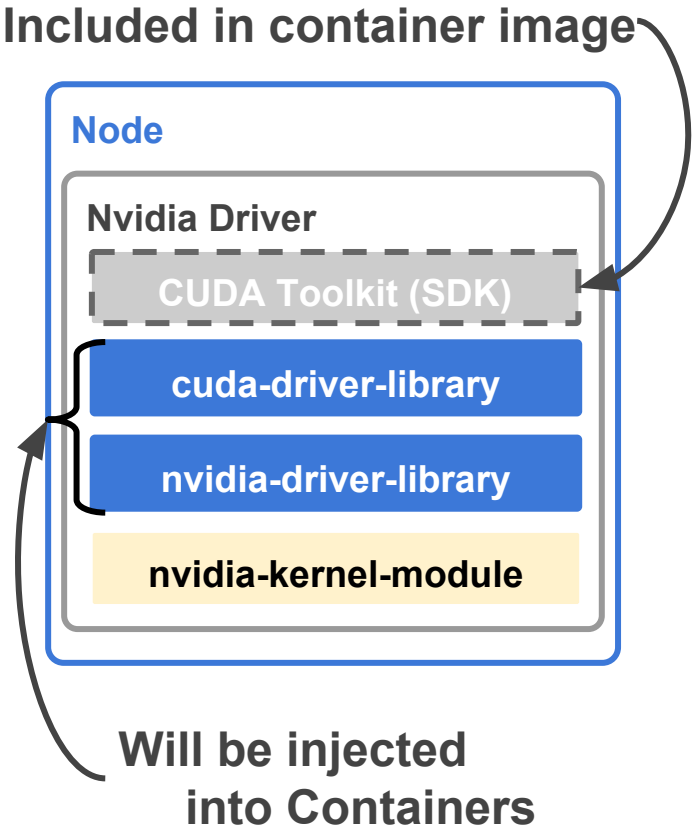
# Tips for Provider - GPU Cluster deployment -

- **Enable GPU container on node**
  1. Install Nvidia Driver
  2. Install nvidia-docker
  3. Add a parameter to kubelet
     - Only add *--feature-gates=Accelerators=true*

# Tips for Provider - GPU Cluster deployment -

## 1. Install Nvidia Driver

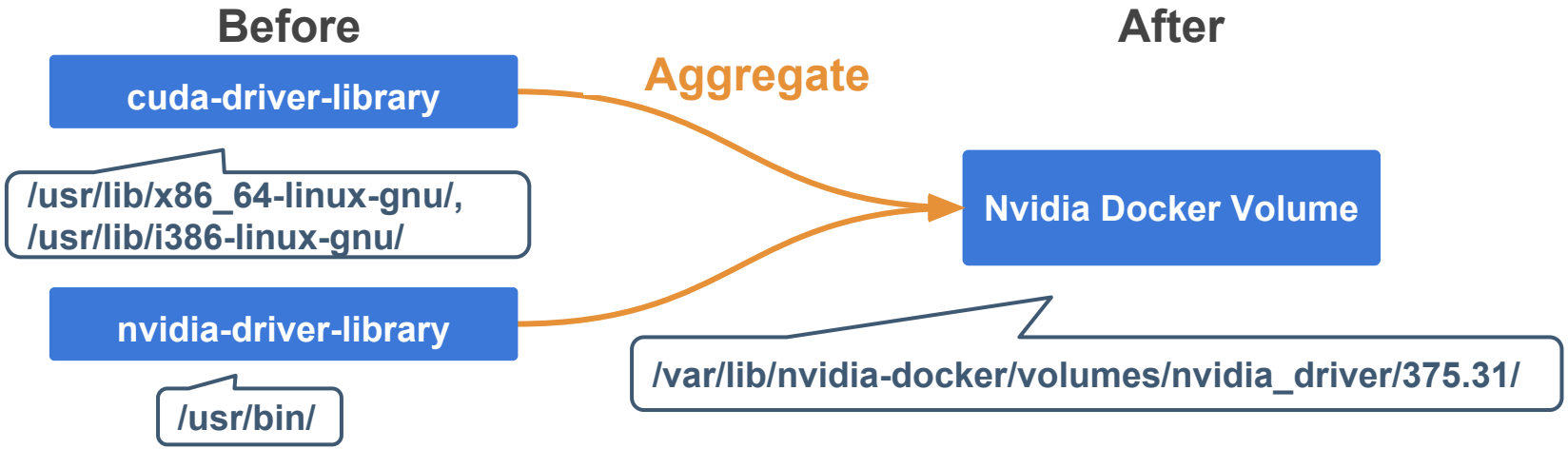
- Install necessary files to use NVIDIA GPU
  - nvidia-driver-library
  - cuda-driver-library
  - nvidia-kernel-module
- DO NOT install CUDA toolkit (SDK) on Slave Node
  - Add option *--no-install-recommends*
  - use CUDA inside container to avoid version dependency problems



# Tips for Provider - GPU Cluster deployment -

## 2. Install Nvidia Docker

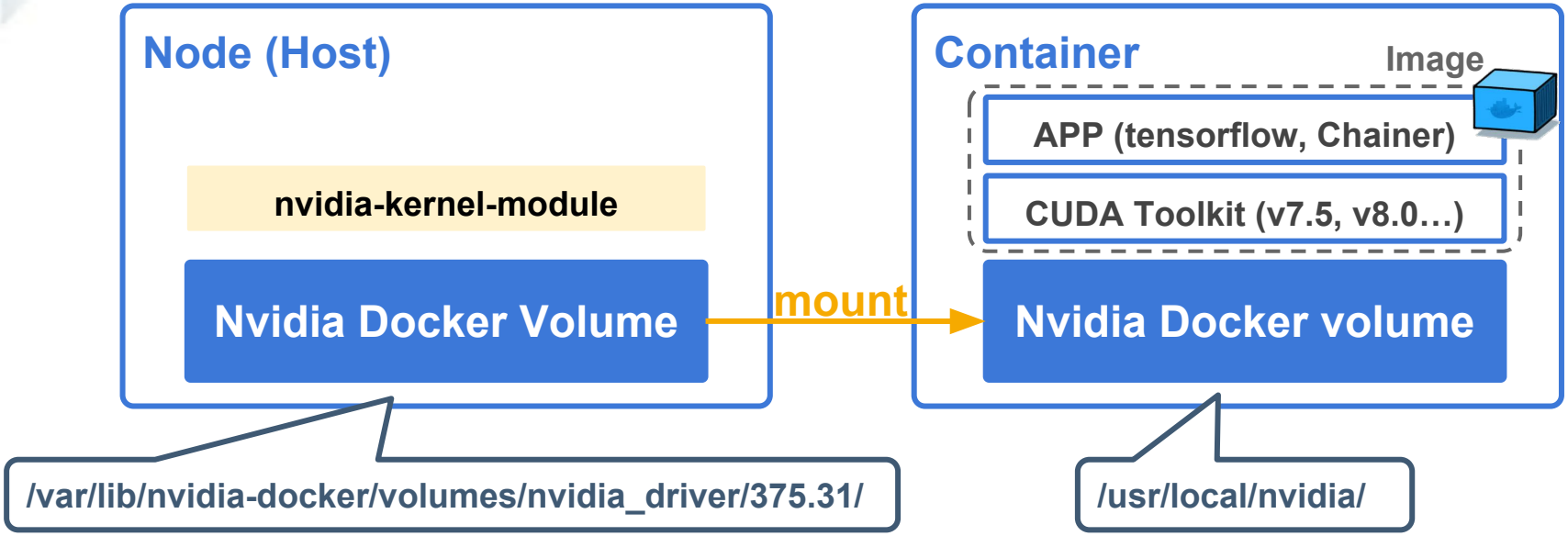
- Pick up the necessary files to inject into containers
  - Nvidia-docker plugin detects **cuda / nvidia-driver-library files**
    - e.g. nvidiactl, nvidia-uvm, libcuda.so.1, libnvidia-ml.so.1
  - It aggregates these files into one specific directory



# Tips for Provider - GPU Cluster deployment -

## 2. Install Nvidia Docker

- Pick up the necessary files to inject into containers
  - **then the directory can be mounted into containers**



# Tips for Provider - GPU Cluster deployment -

## 2. Install Nvidia Docker

- **But, version unmatch problem remains**
  - PATH of “Nvidia Docker Volume” includes **Nvidia Driver version number**
  - With multiple kinds of GPU, this will be a problem


- Make Nvidia Docker Volume PATH unified
  - Create **symbolic link** on each node

```
$ ln -s .../volumes/nvidia_driver/375.31 /usr/local/lib/nvidia
```

In manifest, provider can define the single PATH for Nvidia Docker Volume

# Tips for Provider - Attach label to GPU node -

- **Enable users to select GPUs**
  - In some case, users want to select a specific GPU.
    - e.g. P100, high performance GPU
- Provider attaches label to node according to GPU series
  - In our PoC, `*p100*` label to DGX-1 `*k2*` or `*k10*` to Cisco UCS Servers.

```
$ kubectl label nodes <DGX-1> gputype=p100
```

  - Then user can enable the label in manifest file

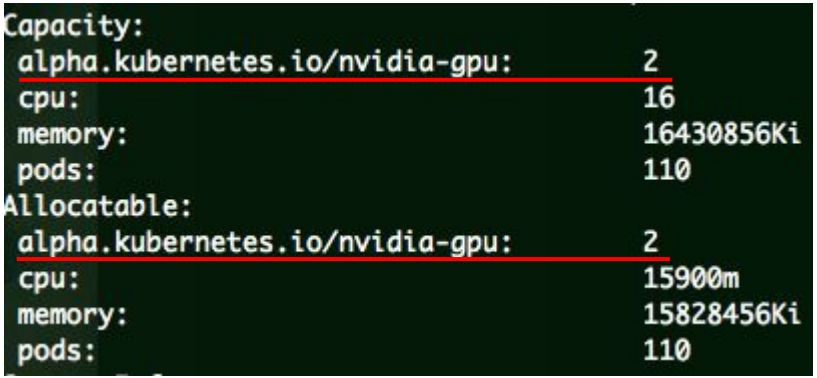
    *nodeSelector: <LABEL NAME>*

```
nodeSelector:
  gputype: p100
```

# Tips for Provider - GPU Cluster monitoring 1/2-

- **Enable monitoring of GPU**
  - If GPUs are few remaining, provider needs to add resources
  - Advise users to release unnecessary GPU resources

- Monitor available GPU numbers on each node by **k8s**
  - *$ kubectl describe node*




However, this nvidia-gpu counter cannot reflect an actual status currently.

# Tips for Provider - GPU Cluster monitoring 2/2-

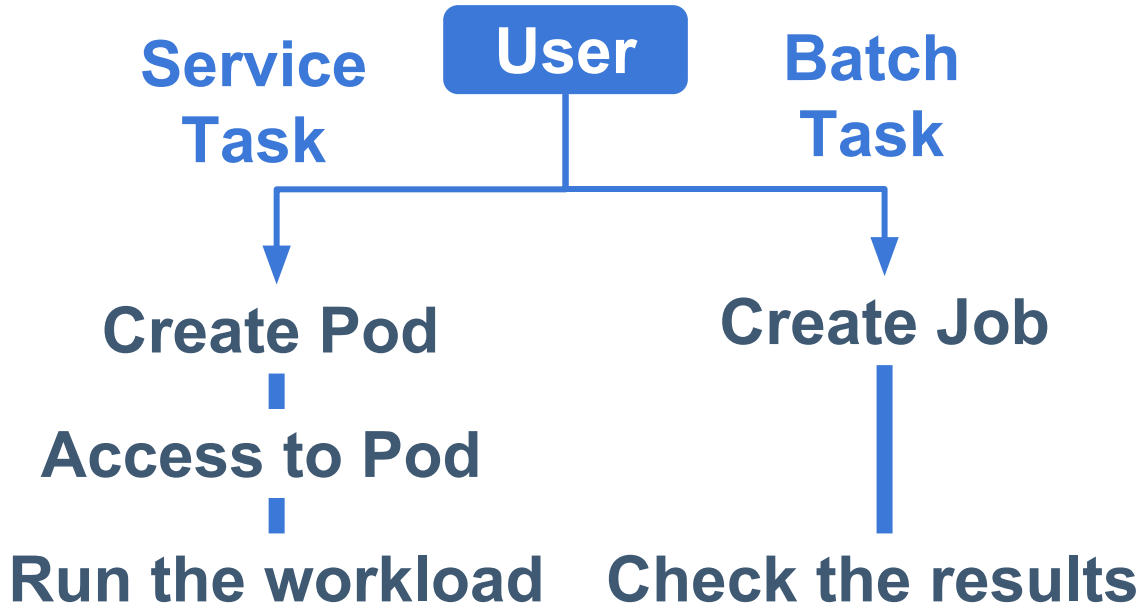
- Monitor GPU use rate on each node by **NVML**
  - NVIDIA Management Library (NVML) get some metrics about GPU

e.g.
**nvml.util.gpu** : GPU utilization rate
**nvml.mem.used** : used GPU memory rate
**nvml.temp** : GPU temperature

Currently, our choice is limited to utilize **NVML** to monitor GPU utilization rate.

# Use Cases and Demo

- In our PoC, users have two ways to execute their task



# Demo

1. About k8s cluster in Our PoC
2. User run tensorflow as Job [Batch Task]
3. User run Digits as Pod [Service Task]

# Future Work

- Multi-tenancy