

GPU Container as a Service を 実現するための最新OSS徹底比較

張 曉晶・角田佳史
松本赳明・原田 和明

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

自己紹介



張 曉晶

Xiaojing Zhang
xiaojing.zhang@ntt.com

技術開発部
ソフトウェアエンジニア

興味: ソフトウェア工学、
ヘテロジニアスなクラウド



角田 佳史

Yoshifumi Sumida
y.sumida@ntt.com

技術開発部
ソフトウェアエンジニア

興味: IaaS、コンテナ

Agenda

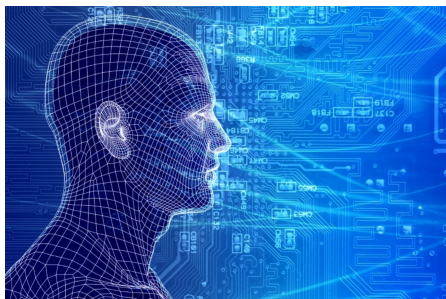
1. 背景 & 目的
2. GPU環境の望ましい要件
3. コンテナ技術関連の各種OSSツール比較
4. OpenStack連携によるマルチテナントの実現
5. GPU Container as a Service つくってみた

Agenda

1. 背景&目的
2. GPU環境の望ましい要件
3. コンテナ技術関連の各種OSSツール比較
4. OpenStack連携によるマルチテナントの実現
5. GPU Container as a Service つくってみた

背景

- AI、機械学習、ビッグデータ解析
 - 高速な計算資源としてGPUが幅広く活用されている



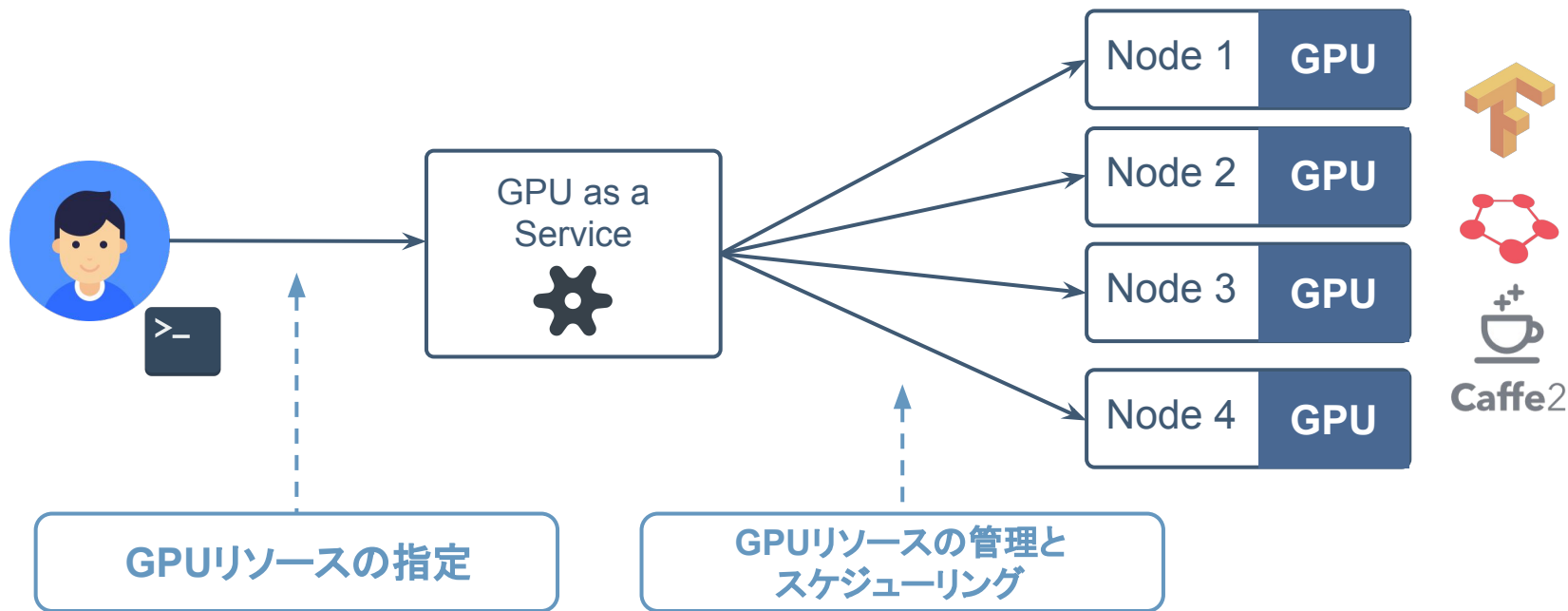
- NTTコミュニケーションズでも注目
 - AI エンジン“COTOHA”、三井化学様における品質予測事例、など
<http://www.ntt.com/business/services/application/ai/cotoha.html>
<http://www.ntt.com/about-us/press-releases/news/article/2016/20160915.html>
 - GPUを用いた学習・検証に対する社内ニーズの高まり

GPU利用のトレンド

- 高速な並列データ処理が可能なGPUの需要がある
 - 学習や解析などのワークロードは非常に高負荷
- 一方でGPUの調達における予算や時間がネックになることも...
 - クラウドサービスを利用する事により解決出来る
 - 主要クラウドプロバイダは既にGPUインスタンスの提供を開始
 - Azure、AWS、GCP、Bluemix (Bare)

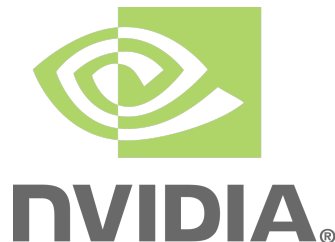
目的

社内向けに検証環境として GPU as a Service を提供する



社内プライベートクラウドでのGPU提供の現状

- クラウド基盤は **OpenStack** を利用
- VMインスタンスとしてGPUリソースを提供
 - VMIにGPUを認識させるため、**PCI Passthrough** を利用
 - GPUリソースとして、3種類の **NVIDIA製 GPU** を利用
- 主に機械学習やディープラーニングなどに利用されている



VMでのGPU提供の問題

1. 機械学習用の環境構築のユーザ負担が大きい
 - VM毎に適切なGPUデバイスドライバをインストールする必要がある
 - デバイスドライバ／CUDA／アプリ間でバージョン整合性を保つ必要がある
2. 特に処理が無い時でもGPUリソースが無駄に占有される
 - 社内向けの検証環境で提供可能なGPU数が少ないため
3. プロバイダ側でGPUデバイスを監視出来ない
 - GPUデバイスを VM へ Passthrough しているため

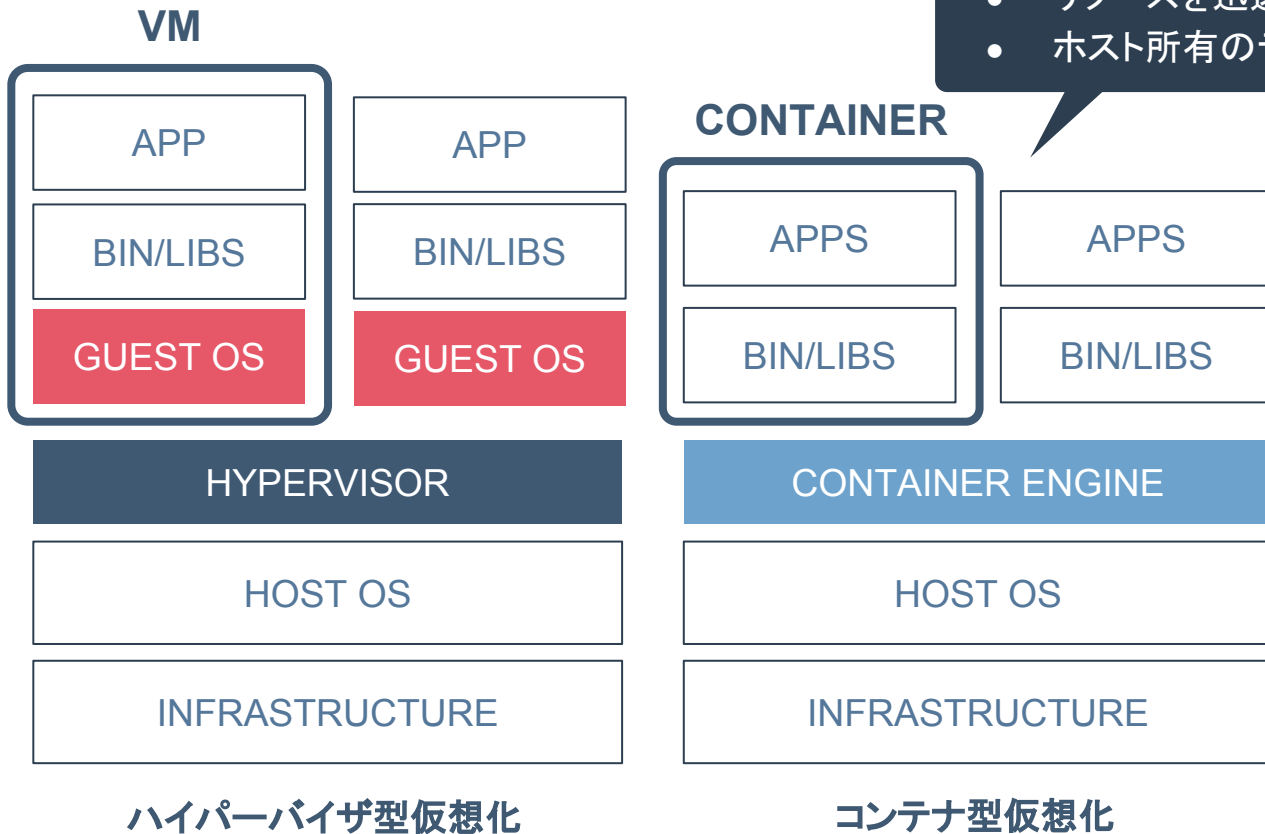
コンテナ技術による問題解決

1. ユーザの環境構築が容易
2. GPUリソースを使用後に迅速に解放出来る
3. GPUリソースの使用状況を監視出来る



GPU as a Service 実現のため **コンテナ技術** を活用

コンテナ技術とは

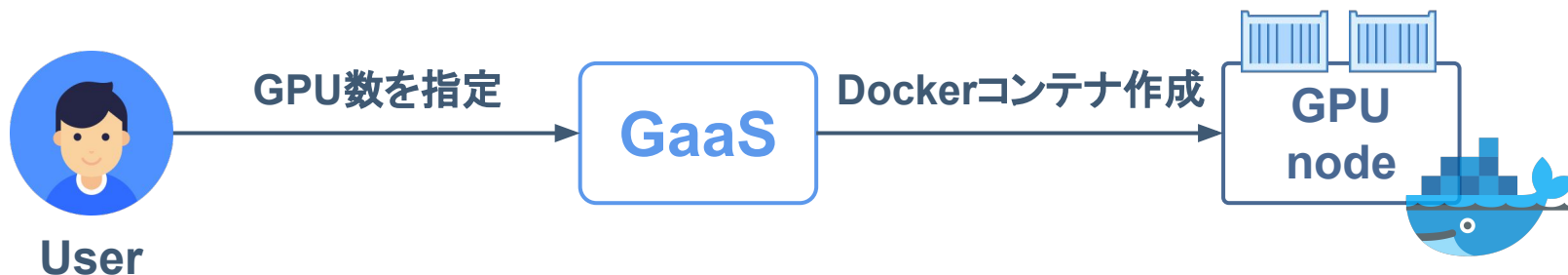


Agenda

1. 背景 & 目的
- 2. GPU環境の望ましい要件**
3. コンテナ技術関連の各種OSSツール比較
4. OpenStack連携によるマルチテナントの実現
5. GPU Container as a Service つくってみた

GPU as a Service の要件 ~ユーザー側~

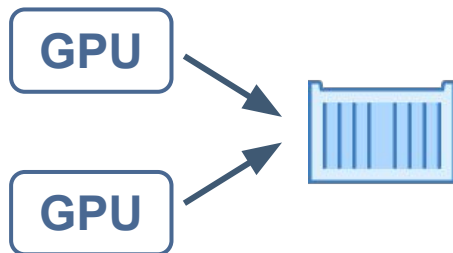
- As a Service として利用できる
 - 即時に利用可能・マルチテナント対応
- GPU環境のデプロイが容易である
 - GPU数など幾つかの事項を指定するのみで良い
 - コンテナ技術でデファクトスタンダードのDockerが利用できる



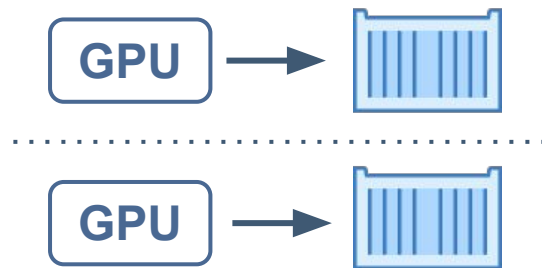
GPU as a Service の要件 ~プロバイダ側~ 1/2

- GPU搭載サーバをクラスタとして管理できる
 - PoC上のGPUクラスタは以下の NVIDIA製 GPU を利用している
 - Tesla K2, Tesla K10, Tesla P100
- コンテナ間で GPU の分離ができる

複数GPUの割り当て

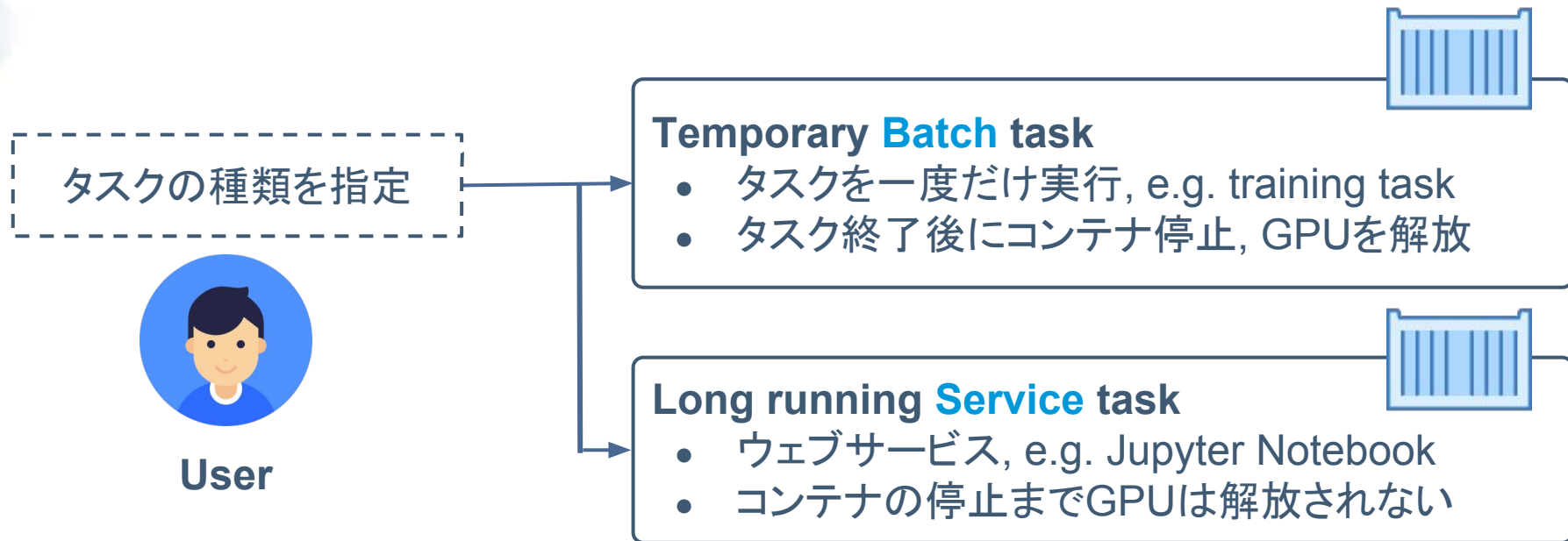


GPUの分離



GPU as a Service の要件 ~プロバイダ側~ 2/2

- 効率的にGPUリソースを扱う事が出来る
 - コンテナのライフサイクルをタスクの種別毎に分類

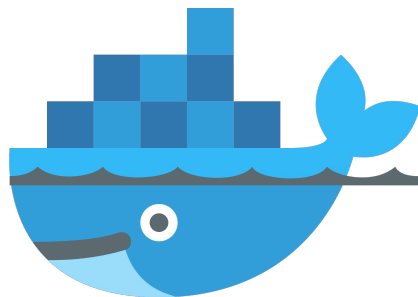


Agenda

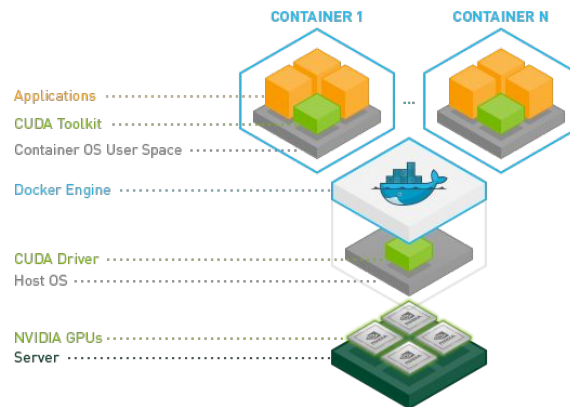
1. 背景 & 目的
2. GPU環境の望ましい要件
- 3. コンテナ技術関連の各種OSSツール比較**
4. OpenStack連携によるマルチテナントの実現
5. GPU Container as a Service つくってみた

前提知識

GPUコンテナを実現するための要素技術



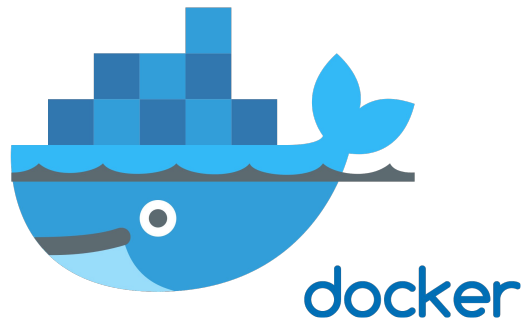
Docker



Nvidia Docker

Docker

- Docker 社による コンテナ管理プラットフォーム
 - 簡単なコマンドでアプリケーション環境をデプロイ出来る
- ```
$ docker run image_name -p 80:80
```
- コンテナ毎に個々のデバイスを分離出来る
  - ユーザが自身で用意したイメージを利用出来る
    - アプリの実行に必要なライブラリ等を纏めたもの



# Nvidia Docker

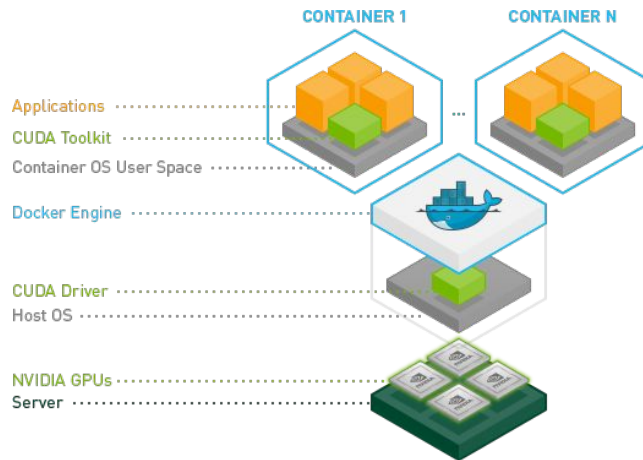
- DockerからGPUを扱う事を容易にする為のツール

- コンテナへGPUリソースを割り当てる事が出来る

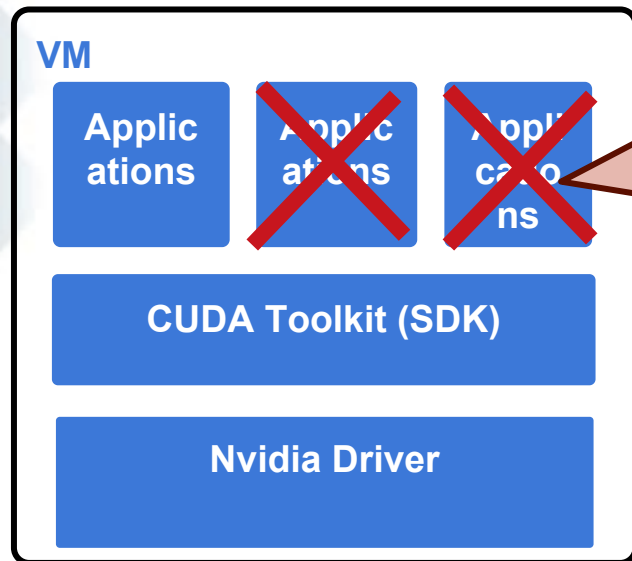
```
$ NV_GPU=0,1 nvidia-docker run nvidia/cuda nvidia-smi
```

- Deep Learning 用のイメージが公式で提供されている

- nvidia/cuda など
- CUDA Toolkit (SDK) を包含



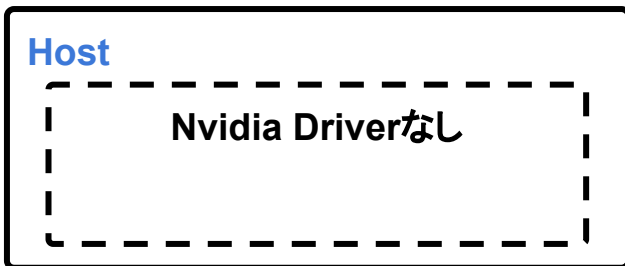
# Docker / Nvidia Docker の利点 1/2



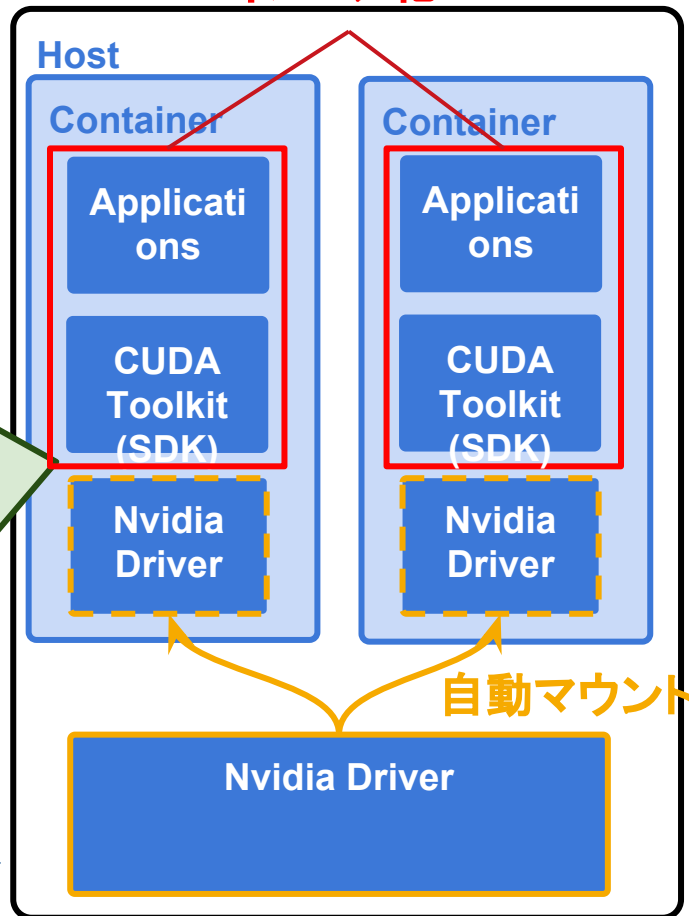
バージョンが整合してないとアプリが正常動作しない

Dockerイメージおよび  
NvidiaDocker  
の自動マウント  
機能により  
バージョン不整合を解決

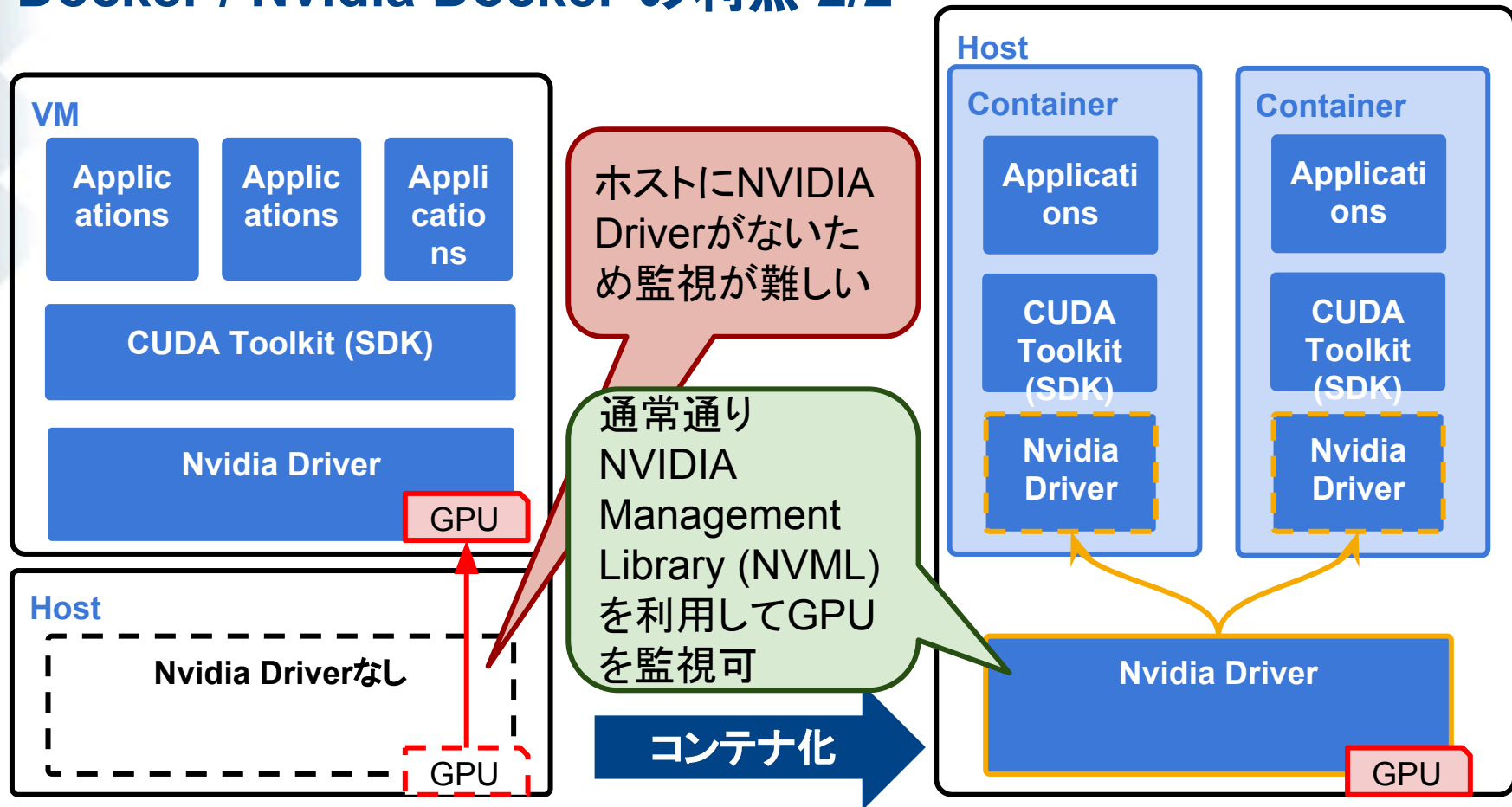
コンテナ化



イメージ化



## Docker / Nvidia Docker の利点 2/2



# コンテナ関連OSSツールの比較

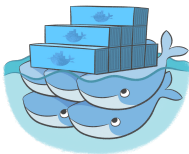
- サービス提供の為に COE (Container Orchestration Engine) が必要
  - GPU サーバ群をクラスタとして管理・提供する必要がある
  - 効率的にGPUリソースを提供する必要がある



GPU as a Service の適切な実現手段となる  
様々なコンテナ関連OSSツールの調査・検証を実施



OpenStack  
Zun



Docker Swarm /  
Swarm Mode



Apache Mesos



Kubernetes

# コンテナ関連OSSツールの検証項目

検証項目を以下のように設定

|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か |
|----------------------------|---------------|------------------|--------|----------------|-----------------|
| OpenStackZun               |               |                  |        |                |                 |
| DockerSwarm<br>/ SwarmMode |               |                  |        |                |                 |
| mesos                      |               |                  |        |                |                 |
| Kubernetes                 |               |                  |        |                |                 |

# コンテナ関連OSSツールの検証項目

検証項目を以下のように設定

|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か |
|----------------------------|---------------|------------------|--------|----------------|-----------------|
| OpenStackZun               |               |                  |        |                |                 |
| DockerSwarm<br>/ SwarmMode |               |                  |        |                |                 |
| mesos                      |               |                  |        |                |                 |
| Kubernetes                 |               |                  |        |                |                 |



- ・ユーザが自身でGPU数を指定可能
- ・複数GPUをコンテナへ割り当て可能

# コンテナ関連OSSツールの検証項目

検証項目を以下のように設定

|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か |
|----------------------------|---------------|------------------|--------|----------------|-----------------|
| OpenStackZun               |               |                  |        |                |                 |
| DockerSwarm<br>/ SwarmMode |               |                  |        |                |                 |
| mesos                      |               |                  |        |                |                 |
| Kubernetes                 |               |                  |        |                |                 |



- ・各コンテナへ異なるGPUを割り当て可能
- ・ビジーなGPUは他コンテナへ割り当て不可

# コンテナ関連OSSツールの検証項目

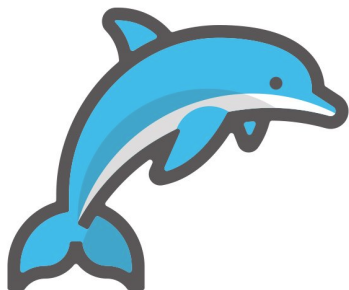
検証項目を以下のように設定

|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か                                                                     |
|----------------------------|---------------|------------------|--------|----------------|-------------------------------------------------------------------------------------|
| OpenStackZun               |               |                  |        |                |  |
| DockerSwarm<br>/ SwarmMode |               |                  |        |                |                                                                                     |
| mesos                      |               |                  |        |                |                                                                                     |
| Kubernetes                 |               |                  |        |                |                                                                                     |

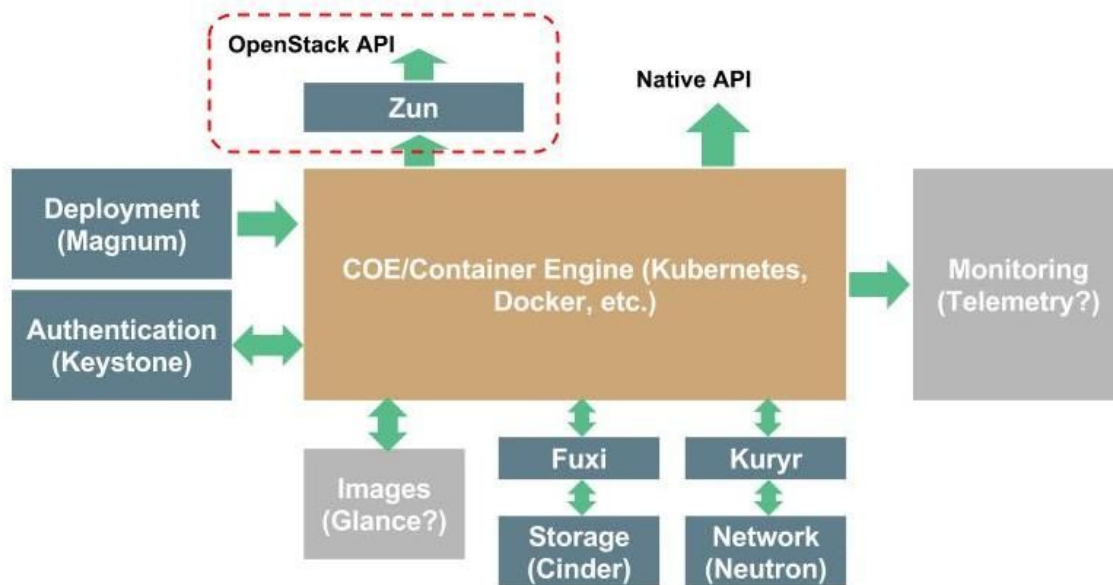
・コンテナ内部のプロセスが停止時に  
自動的にコンテナが停止する

# OpenStack Zun

- OpenStack 上でコンテナを管理するためのツール
- OpenStack内部で基本的なコンテナの制御 (i.e. CRUD) のみ提供



Zun の  
マスコットキャラクター



# OpenStack Zun

- GPUリソースがサポートされていない
  - Docker に “CpuShares” and “Memory” パラメータのみ渡す事が可能
  - GPU対応の話題がコミュニティ側で挙がっていない



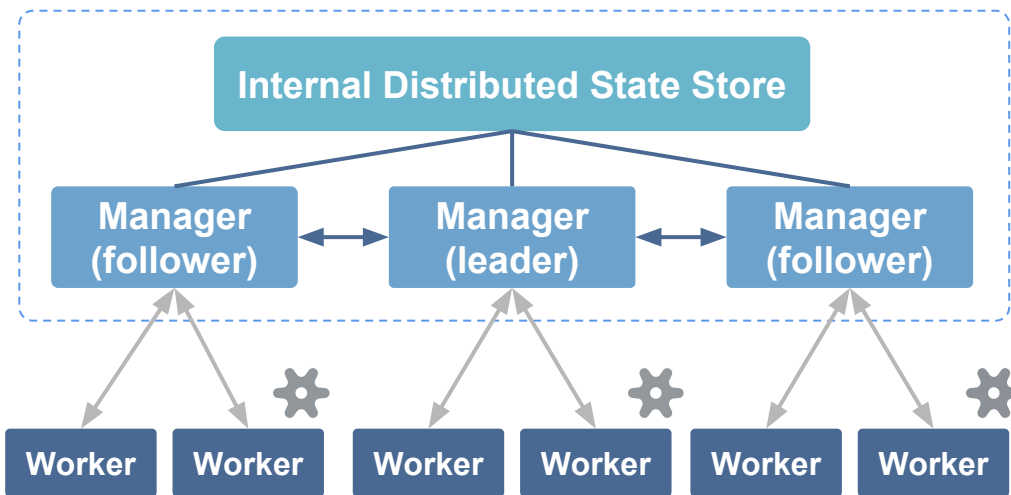
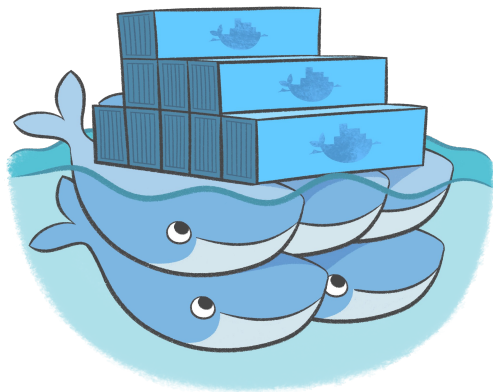
OpenStack Zun は Dockerサポート 以外の  
要件を満たしていない

# OpenStack Zun の検証結果

|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か |
|----------------------------|---------------|------------------|--------|----------------|-----------------|
| OpenStackZun               | ✗             | ✗                | ✗      | ✓              | ✗               |
| DockerSwarm<br>/ SwarmMode |               |                  |        |                |                 |
| mesos                      |               |                  |        |                |                 |
| Kubernetes                 |               |                  |        |                |                 |

# Docker Swarm / swarm mode

- Docker ネイティブなクラスタ管理ツール
- Dockerが展開されたマシン複数台から簡単にクラスタを構築出来る
- Docker v1.12 以上でDockerに標準で組み込まれている



## Docker Swarm / swarm mode

- Docker Swarm は GPU をサポートしていない
  - 現在もGPU対応が Docker Project において進められている
    - <https://github.com/docker/docker/issues/23917>
- Nvidia Docker も同様にサポートされていない
  - GPUコンテナに必要なライブラリ群をマウント出来ない



**Docker Swarm / swarm mode は  
“GPUクラスタ管理” を満たしていない**

# Docker Swarm / Swarm Mode の検証結果

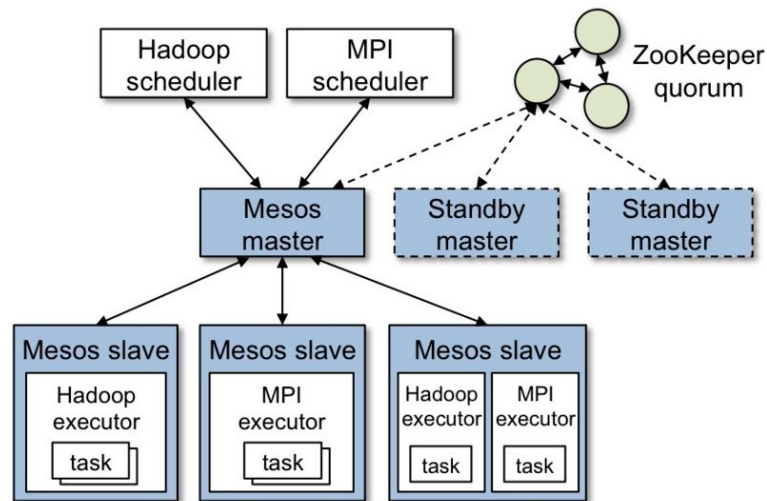
|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か |
|----------------------------|---------------|------------------|--------|----------------|-----------------|
| OpenStackZun               | ✗             | ✗                | ✗      | ✓              | ✗               |
| DockerSwarm<br>/ SwarmMode | ✗             | ✗                | ✗      | ✓              | ✗               |
| mesos                      |               |                  |        |                |                 |
| Kubernetes                 |               |                  |        |                |                 |

# Mesos

- Apacheソフトウェア財団によるクラスタ管理ツール
- 効率的かつ動的なリソースの分配や共有が可能
- 分散アプリケーションやフレームワークを Mesos Master が制御
  - e.g. Marathon, Chronos, Hadoop...



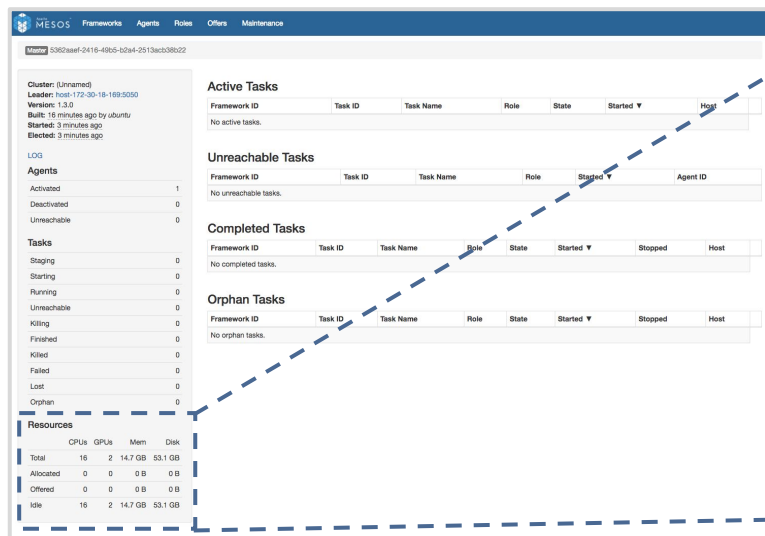
Apache  
**MESOS**™



# Mesos

- Mesos の GPU サポート状況

- Mesos v1.0.0 以上で Nvidia GPU に対応している
- CPU や Memoryと同様に GPU も管理する事が出来る



| Resources |     |     |         |         |
|-----------|-----|-----|---------|---------|
|           | CPU | GPU | Mem     | Disk    |
| Total     | 16  | 2   | 14.7 GB | 53.1 GB |
| Allocated | 0   | 0   | 0 B     | 0 B     |
| Offered   | 0   | 0   | 0 B     | 0 B     |
| Idle      | 16  | 2   | 14.7 GB | 53.1 GB |

# Mesos

Mesos フレームワークは二種類のコンテナを扱う事が可能

- Mesos コンテナ・Docker コンテナ

| Task Type | Frameworks | GPU + Mesos | GPU + Docker |
|-----------|------------|-------------|--------------|
| Batch     | Chronos    | ✗           | N/A          |
|           | Metronome  | ✗           | N/A          |
| Service   | Aurora     | ✓           | ✗            |
|           | Marathon   | ✓           | ✗            |

# Apache Mesos の検証結果

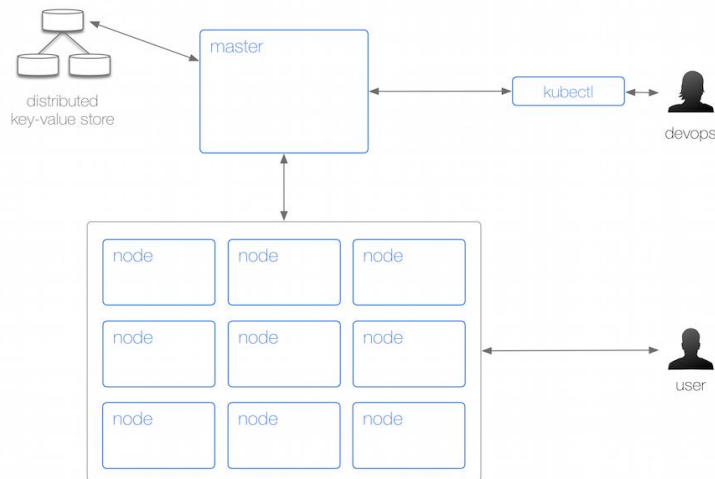
|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か |
|----------------------------|---------------|------------------|--------|----------------|-----------------|
| OpenStackZun               | ✗             | ✗                | ✗      | ✓              | ✗               |
| DockerSwarm<br>/ SwarmMode | ✗             | ✗                | ✗      | ✓              | ✗               |
| mesos                      | ✓             | ✓                | ✓      | ✗              | ✓               |
| Kubernetes                 |               |                  |        |                |                 |

# Kubernetes

- Google による Container Orchestration Engine (COE)
- コンテナのクラスタリング・オートスケーリング等の様々な機能を提供
- コミュニティも活発で頻繁に議論や機能の開発が行われている

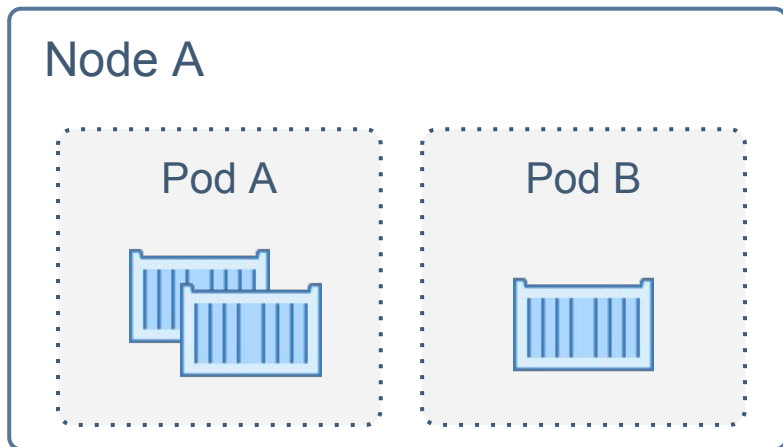


kubernetes



# Kubernetes

- コンテナは **Pod** と呼ばれる単位で管理される
  - Podは単一もしくは複数のコンテナで構成される
  - Pod単位で CRUD 等の様々な処理が行われる



## タスク種別毎の Pod 管理方法

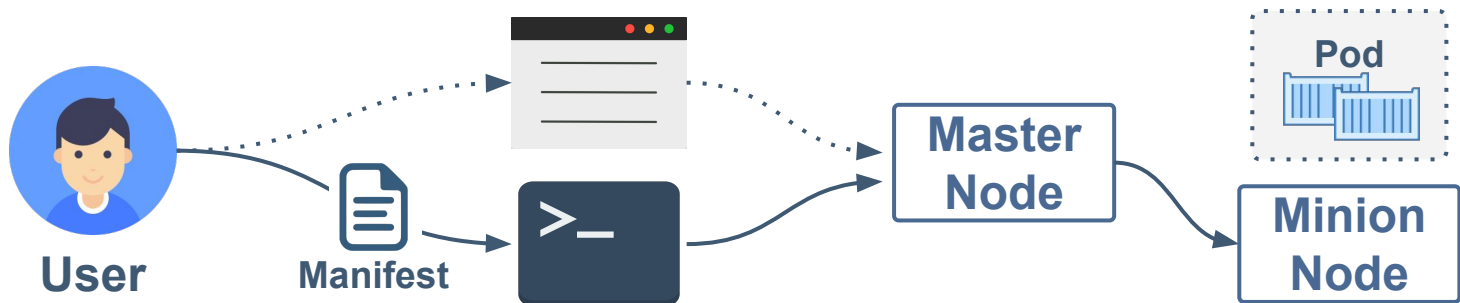
**Service Task** : Pod, ReplicaSet  
Deployment など

**Batch Task** : Job など

# Kubernetes

- コンテナの CRUD 管理 は マニフェストファイル で行う
  - Pod 及び Pod 内のコンテナの情報を定義する
    - Podの管理方法 (Kind)
    - コンテナ・コンテナイメージ名など
  - yaml・jsonの各種形式で定義可能
  - CLI (kubectl) や WebUI から利用可能

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



# Kubernetes

- KubernetesのGPUサポート状況
  - v1.3.x から試験的にGPUスケジューリングが導入された
    - GPU as a Service の要件を満たしていない
      - 複数GPUが割り当て出来ない
      - GPUの分離が出来ない
  - v1.6 以上で Alpha 版として GPUスケジューリングをサポート
    - GPUスケジューリングの機能が改善された
      - v1.3.x ~ v1.5.x までの問題が解決されている
    - 自動的にノード上のGPU数を検出できる

**v1.6系を利用**

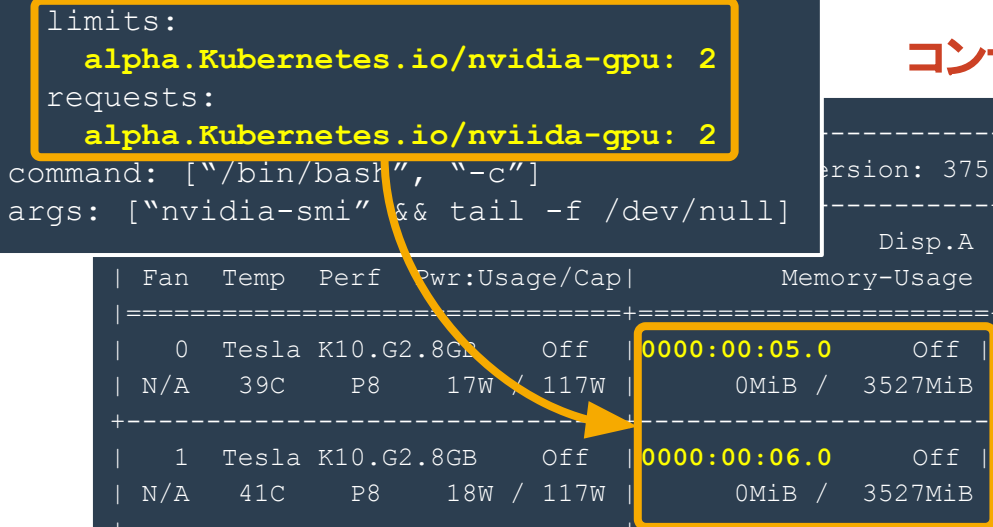
# Kubernetes

- 複数GPUを単一のコンテナに割り当て可能

マニフェスト内でGPU数を指定する

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

## コンテナ内の情報



| -----+-----     |       |            |            |                |     |              |          |            |  |
|-----------------|-------|------------|------------|----------------|-----|--------------|----------|------------|--|
| version: 375.39 |       |            |            |                |     |              |          |            |  |
| -----+-----     |       |            |            |                |     |              |          |            |  |
|                 | 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

- コンテナ間でGPUの分離が可能

```
$ 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-MI | Bus-Id        | Disp.A         |                      |  |
| Fan               | Temp    | Perf           | Pwr:Usage/Cap | Memory-Usage   |                      |  |
| 0                 | GRID K2 | Off            | 0000:07:00.0  | Off            |                      |  |
| N/A               | 38C     | P8             | 18W / 117W    | 0MiB / 4036MiB |                      |  |

\$ kubectl logs nvidia-smi-d88z9  
Mon Apr 10 15:55:22 2017

| NVIDIA-SMI 367.57 |         |                |               |                | Driver Version: 367. |  |
|-------------------|---------|----------------|---------------|----------------|----------------------|--|
| GPU               | Name    | Persistence-MI | Bus-Id        | Disp.A         |                      |  |
| Fan               | Temp    | Perf           | Pwr:Usage/Cap | Memory-Usage   |                      |  |
| 0                 | GRID K2 | Off            | 0000:90:00.0  | Off            |                      |  |
| N/A               | 38C     | P8             | 18W / 117W    | 0MiB / 4036MiB |                      |  |

# Kubernetes

- バッチタスクに対応

タスクの種類をマニフェスト内で定義

```
apiVersion: batch/v1
kind: Job
...
limits:
 alpha.kubernetes.io/nvidia-gpu: 2
requests:
 alpha.kubernetes.io/nvidia-gpu: 2
...
command: ["/bin/bash", "-c"]
args: ["nvidia-smi"]
...
```

## Podの情報 (起動時)

```
$ 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
```

## Podの情報 (終了時)

```
$ 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
```

# 検証結果

|                            | GPUクラスタ<br>管理 | GPUの指定<br>複数割り当て | GPUの分離 | Docker<br>サポート | バッチタスク<br>実行可能か |
|----------------------------|---------------|------------------|--------|----------------|-----------------|
| OpenStackZun               | ✗             | ✗                | ✗      | ✓              | ✗               |
| DockerSwarm<br>/ SwarmMode | ✗             | ✗                | ✗      | ✓              | ✗               |
| mesos                      | ✓             | ✓                | ✓      | ✗              | ✓               |
| Kubernetes                 | ✓             | ✓                | ✓      | ✓              | ✓               |

# Agenda

1. 背景 & 目的
2. GPU環境の望ましい要件
3. コンテナ技術関連の各種OSSツール比較
- 4. OpenStack連携によるマルチテナントの実現**
5. GPU Container as a Service つくってみた

# Kubernetes を用いたマルチテナントの実現

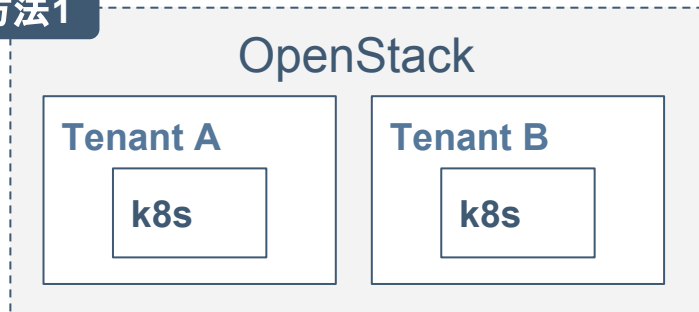
Kubernetes でのマルチテナント実現手法は **2種類** ある

方法1. OpenStack のテナントごとにk8sをデプロイする

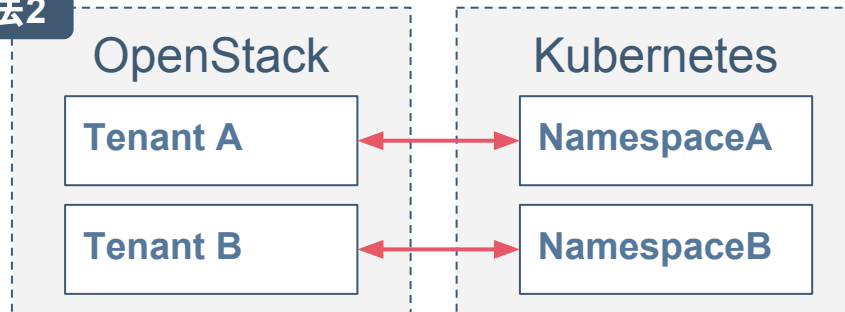
- 1-1 OpenStack Magnumでデプロイ
- 1-2 OpenStack Heatでデプロイ

方法2. Kubernetes の認証・認可に OpenStack Keystone を用いる

方法1

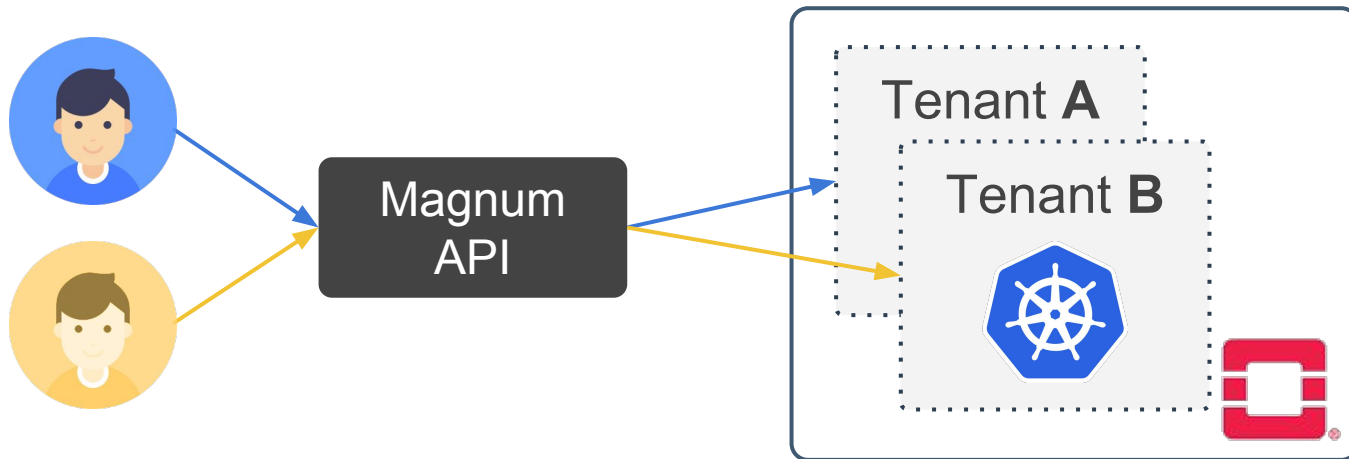


方法2



# 方法1-1 Magnum

- OpenStack 上で COE のクラスタ環境を簡単に構築してくれるコンポーネント
  - Kubernetes や Mesos などに対応している
- OpenStack 環境内に Kubernetes クラスタの構築が可能
  - OpenStack Cinder や Octavia (LBaaS) 等と連携出来る
  - 後から 容易に Minion Node 数を増減出来る



# 方法1-1 Magnum

- GPU 対応の k8s 1.6 はサポートされていない
  - 現在の Magnum の k8s 対応状況  
Minion Node 構築用のOSイメージにより異なる
    - [Fedora Atomic]: k8s 1.5.3 [CoreOS]: k8s 1.5.2
- 公式サポートのイメージに Nvidia Driver が含まれていない
  - GPU対応には 独自 OS イメージの作成が必要になる
- 独自設定で k8s 1.6 が利用出来るかを検証を行った
  - ユーザ側で書き換える事が出来ない内部の設定ファイルを変更
    - 現在の Magnum では構築出来ない
    - k8s 1.6 と Magnum が認証連携出来ない



# 方法1-1 Heat

- OpenStack 上で オーケストレーション を行うコンポーネント
  - Heat Template に基づいて VM や NWなどを構築出来る
  - k8s が公式メンテナンスしている Heat Template がある
    - k8s 1.6 向けのテンプレートが存在するが ...
- ユーザ自身が柔軟なクラスタ環境を構築出来る
  - ユーザ側で Heat Template を直接編集出来る
    - Heat Template を編集する事で Nvidia Driver のインストール自動化も可能
- k8s 1.6 を構築可能か検証を行った
  - 現在の OpenStack Heat では構築出来ない
    - 必要な各種ファイルの設定や配置が適切でない



## 方法2 Keystone 連携

- 認証処理について
  - Kubernetes のユーザ認証を Keystone へ委譲出来る
  - OpenStackの既存ユーザを用いた認証が可能
- 認可処理について
  - OpenStack のテナントと Namespace の対応付けの設定が必要
    - Kubernetes は Namespace と呼ばれるテナント分離の仕組みがある
    - Kubernetes の 認可の仕組み (RBACやABAC) を利用する
  - Keystone のみで認可を実現することは現状出来ない
    - <https://github.com/Kubernetes/Kubernetes/pull/25624>

# マルチテナント実現手法の比較

- k8s 1.6 対応 の Heat 及び Keystone の2手法を以下の観点で比較し、  
**検証環境における適切なマルチテナント実現方法を選択する**
  - GPUリソースの分離
  - プロバイダ視点でのGPU監視
  - コンテナ配備先の分離
  - ネットワークの分離
  - OpenStack 上のVMとの通信

# マルチテナント実現手法の比較

|                                               | GPUリソース<br>の分離 | プロバイダ視点<br>GPUの監視 | コンテナ<br>配備先の分離 | ネットワークの<br>分離                       | OpenStack<br>VMとの通信 |
|-----------------------------------------------|----------------|-------------------|----------------|-------------------------------------|---------------------|
| <b>方法1</b><br>OpenStackの<br>テナントごとに<br>k8sを展開 | テナント単位         | 困難                | テナント単位         | テナント単位<br>(VXLAN,<br>GRE...)        | Tenant<br>Network 内 |
| <b>方法2</b><br>k8s<br>認証/認可に<br>Keystone       | 全テナント<br>共有    | 可能                | 全テナント<br>共有    | 全テナント<br>共有<br>(Flat +<br>Iptables) | Floating IP<br>経由   |

## 今回の検証環境の要件

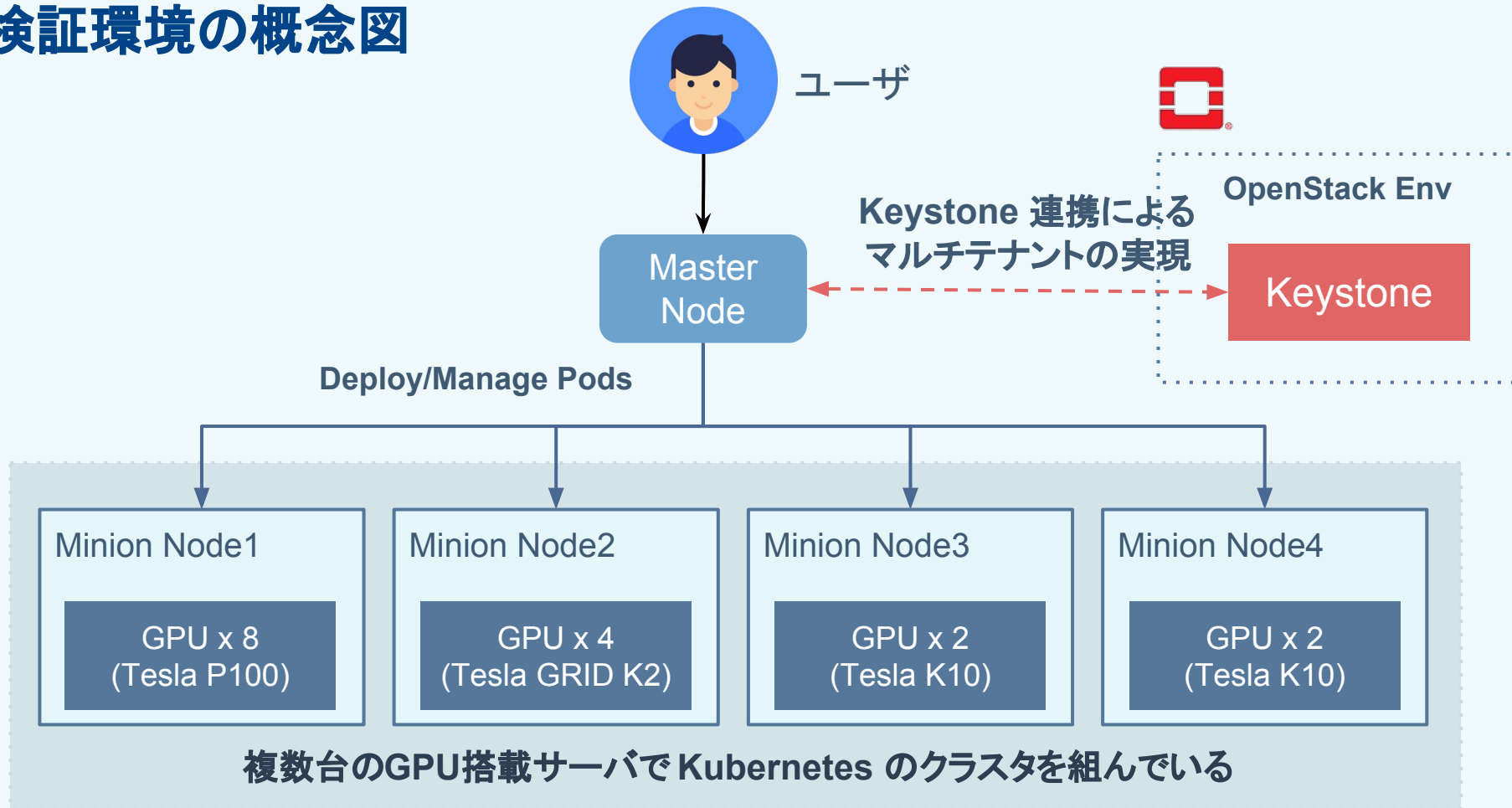
- GPUリソースをなるべく全体で共有したい
- GPUリソースをプロバイダ側で監視したい
- 厳密にNW等が分離されている必要がない

➡ **方法2を選択**

# Agenda

1. 背景 & 目的
2. GPU環境の望ましい要件
3. コンテナ技術関連の各種OSSツール比較
4. OpenStack連携によるマルチテナントの実現
5. **GPU Container as a Service つくってみた**

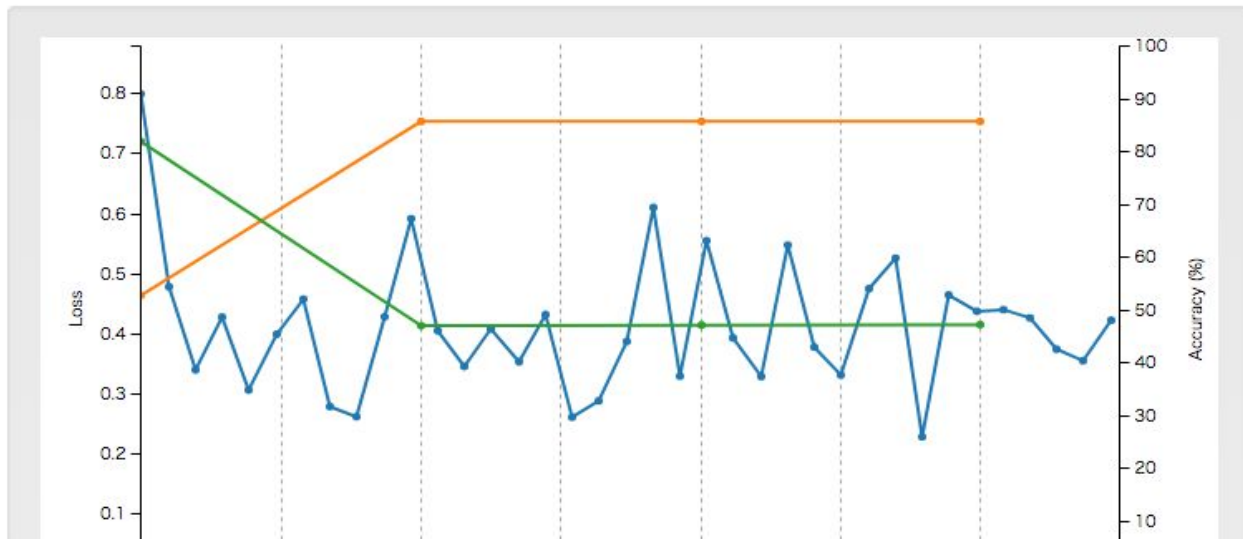
# 検証環境の概念図



# Demo

Kubernetes 上で Digits (GPU) を実行してみた

# NVIDIA. × digits



## Hardware

**Tesla K10.G2.8GB (#0)**

### Memory

181 MB / 3.44 GB (5.1%)

### GPU Utilization

92%

### Temperature

56 °C

## Process #246

### CPU Utilization

154.4%

### Memory

# GPUクラスタの構築と管理 ～構築～ 1/3

- 各ノード上でのGPUコンテナの有効化

1. Nvidia Driver のインストール

- <http://www.nvidia.co.jp/Download/index.aspx?lang=jp>

2. Nvidia docker のインストール

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

3. Kubernetes 環境の構築

- kubelet の起動時に以下のパラメータを付与する
- `--feature-gates=Accelerators=true`

## GPUクラスタの構築と管理 ～構築～ 2/3

- ユーザが指定のGPUを選択可能にする

- 該当ノードにラベル (GPU名) を設定する

- PoC環境では, GPUの種類に応じて`p100`, `k2`, `k10` を指定可能

```
$ kubectl label nodes <nodeName> gputype=p100
```

- ユーザはマニフェストで需要に見合ったGPUラベル名を指定

```
nodeSelector.gputype: p100
```



# GPUクラスタの構築と管理 ～構築～ 3/3

## ● Kubernetes と Keystone の連携

### ○ Kubernetes 側での設定

- kube-apiserver へのパラメータ付与

`--experimental-keystone-url`

`--experimental-keystone-cafile`

### ○ OpenStack 側での設定

- Keystone Endpoint の HTTPS 化

- リバースプロキシ等を活用する

### ○ 上記ののちに認可設定を Kubernetes で行う

- Keystone に存在する既存ユーザへ Role の適用

# GPUクラスタの構築と運用 ～運用～ 1/3

- GPUリソースの監視

- Kubernetes における GPU監視方法
  - Kubernetes に標準で実装されているGPUリソース監視機能
  - NVIDIAによって提供されている NVdia Management Library (NVML)

## GPUクラスタの構築と運用 ～運用～ 2/3

- k8sではGPU数を監視できる機能が標準で備わっている
- 以下のコマンドで使用状況を見ることが可能
  - `$ kubectl describe node`

```
Capacity:
 alpha.kubernetes.io/nvidia-gpu: 2
 cpu: 16
 memory: 16430856Ki
 pods: 110
Allocatable:
 alpha.kubernetes.io/nvidia-gpu: 2
 cpu: 15900m
 memory: 15828456Ki
 pods: 110
```

- ・利用可能なGPU数を正確に取得する事ができない
- ・どのコンテナがGPUを利用中かは分からない

# GPUクラスタの構築と運用 ～運用～ 3/3

- NVIDIAは NVdia Management Library (NVML) を提供している
  - 様々なGPUメトリクスを取得する事が可能

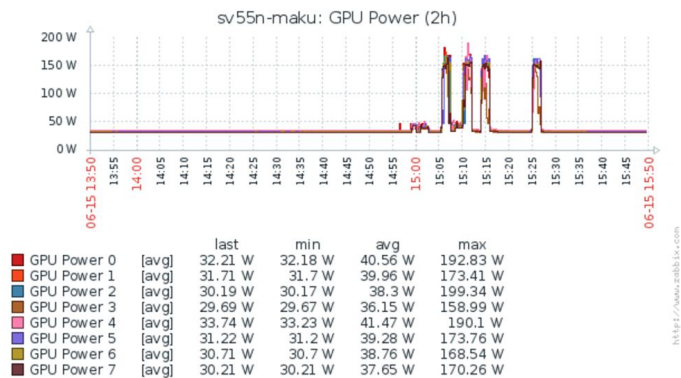
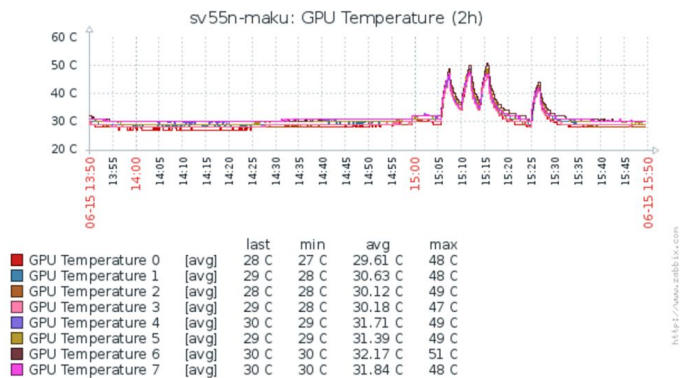
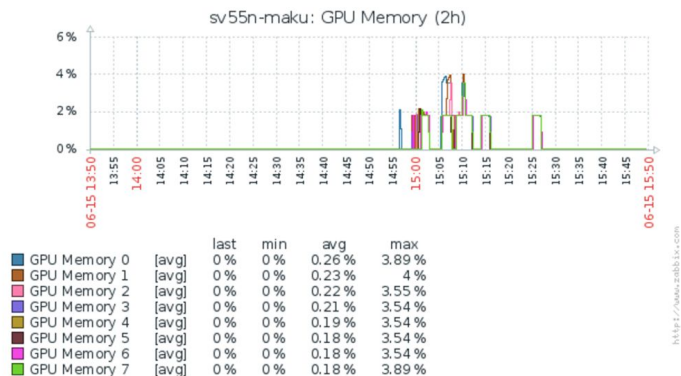
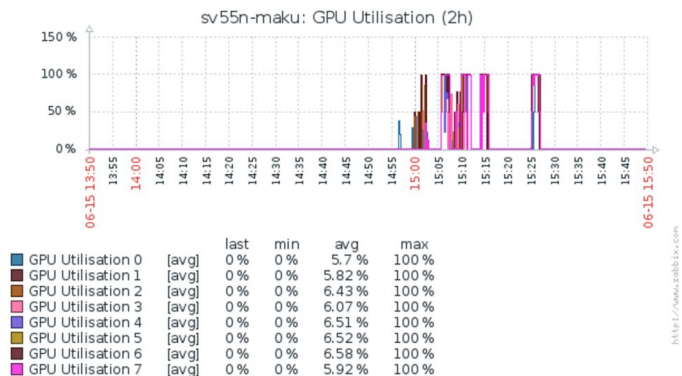
e.g.

**nvml.util.gpu** : GPU 利用率 (%)

**nvml.mem.used** : GPUの使用量メモリ量

**nvml.temp** : GPUの温度

# GPUクラスタの構築と運用 ~運用~ 3/3



# まとめ

1. 背景 & 目的
2. GPU環境の望ましい要件
3. コンテナ技術関連の各種OSSツール比較
4. OpenStack連携によるマルチテナントの実現
5. GPU Container as a Service つくってみた

## Special Thanks to...

- 横山智大
- 松下正樹
- 小倉真人
- Ankit Purohit
- 奥村昌和
- 逸見彰一郎



ご清聴ありがとうございました