Caicloud TaaS Introduction

TensorFlow on Kubernetes

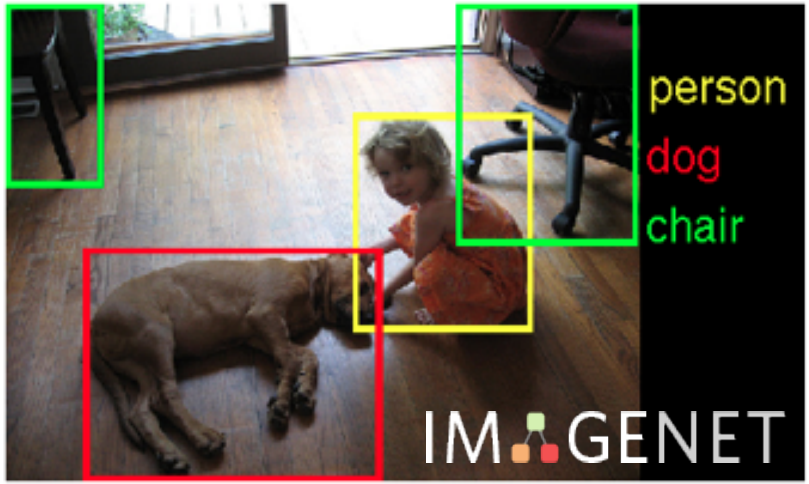
岑鹏浩 才云科技

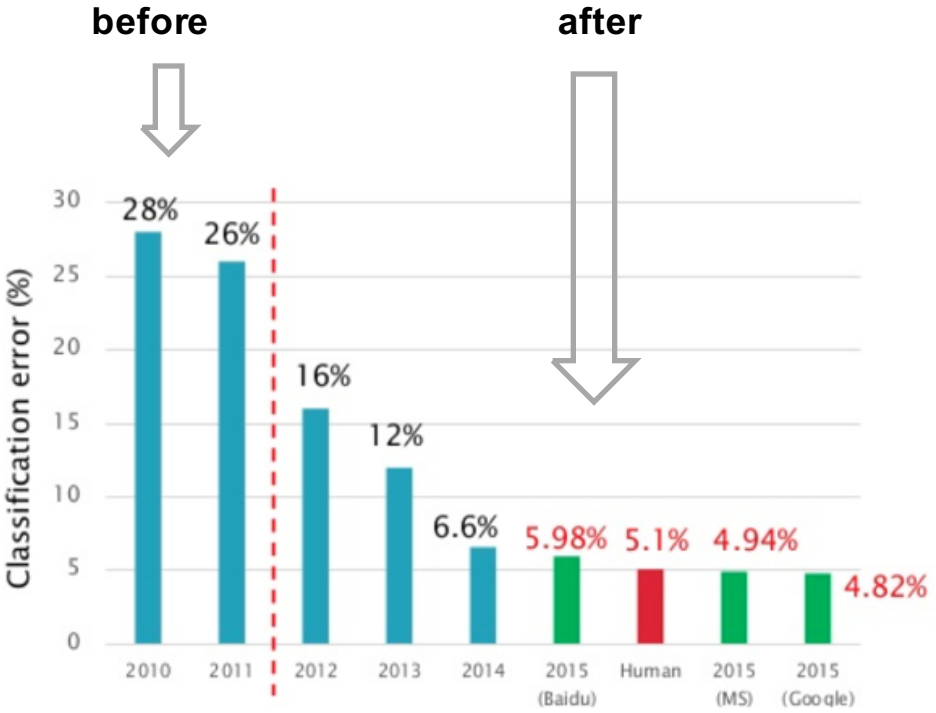
Agenda

- Deep Learning Introduction
- TensorFlow Introduction
- Distributed TensorFlow on Kubernetes
- TaaS Introduction
- TaaS Demo

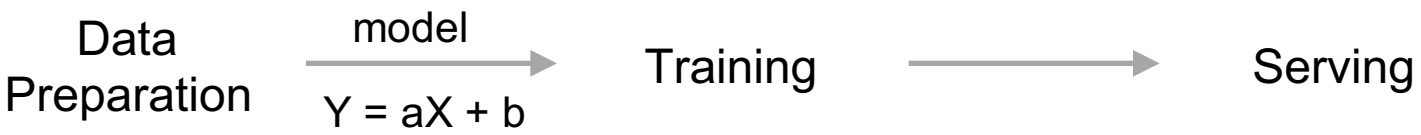

The Future of Go Summit, Match One: Ke Jie & AlphaGo
柯洁 KE JIE
00:15:19
ALPHAGO
01:45:34
浙江省体育局
4:33:05 / 6:18:48


Google
self-driving car


person
dog
chair
IMAGENET

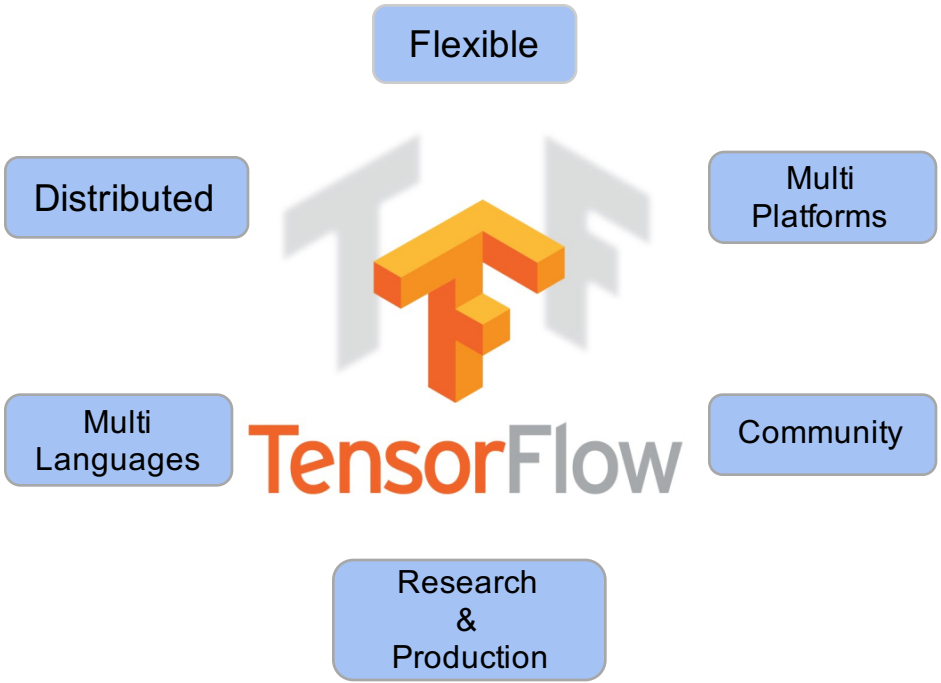

before
after
Classification error (%)
30
25
20
15
10
5
0
28%
26%
16%
12%
6.6%
5.98%
5.1%
4.94%
4.82%
2010
2011
2012
2013
2014
2015 (Baidu)
Human
2015 (MS)
2015 (Google)


Data Preparation
Training
Serving


Data
Preparation
model
Y = aX + b
Training
Serving

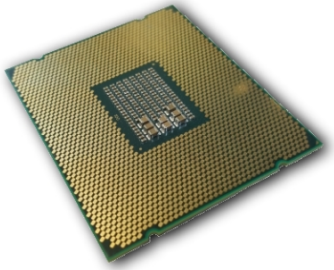

Data
Preparation
model
Y = aX + b
Training
Serving

TensorFlow: Machine Learning for Everyone

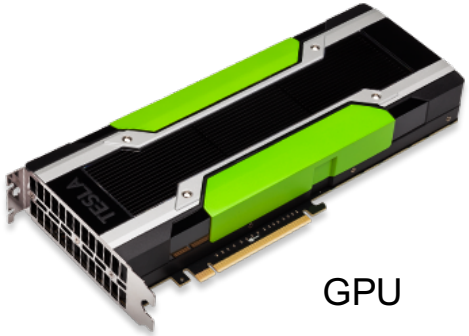


- Open Source
- Fast, Flexible, and Production-Ready
- Python and C++ API
- Distributed Processing
- Supports CPUs & GPUs
- Machine Learning & Deep Neural Network
- Based on data flow graphs

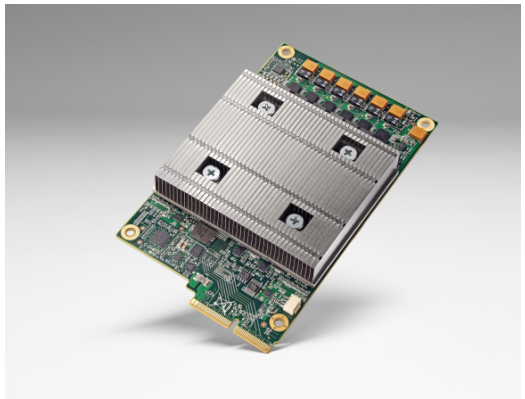
TensorFlow: Machine Learning for Everyone



CPU



GPU

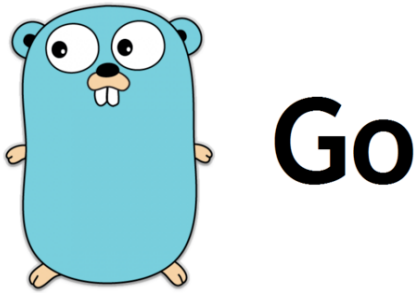


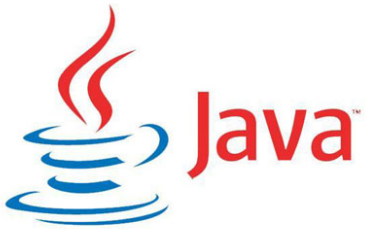
TPU

iOS

Android

TensorFlow: Machine Learning for Everyone

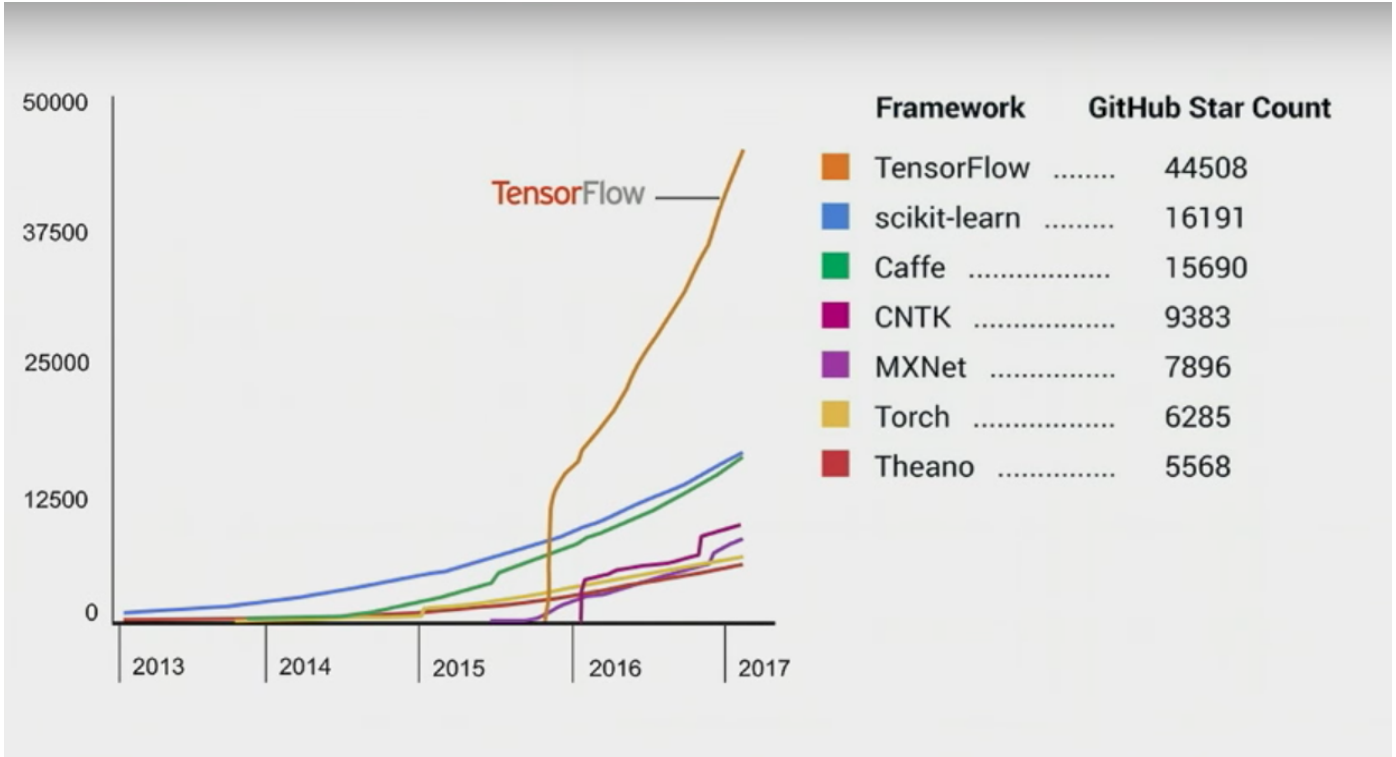




TensorFlow: Machine Learning for Everyone



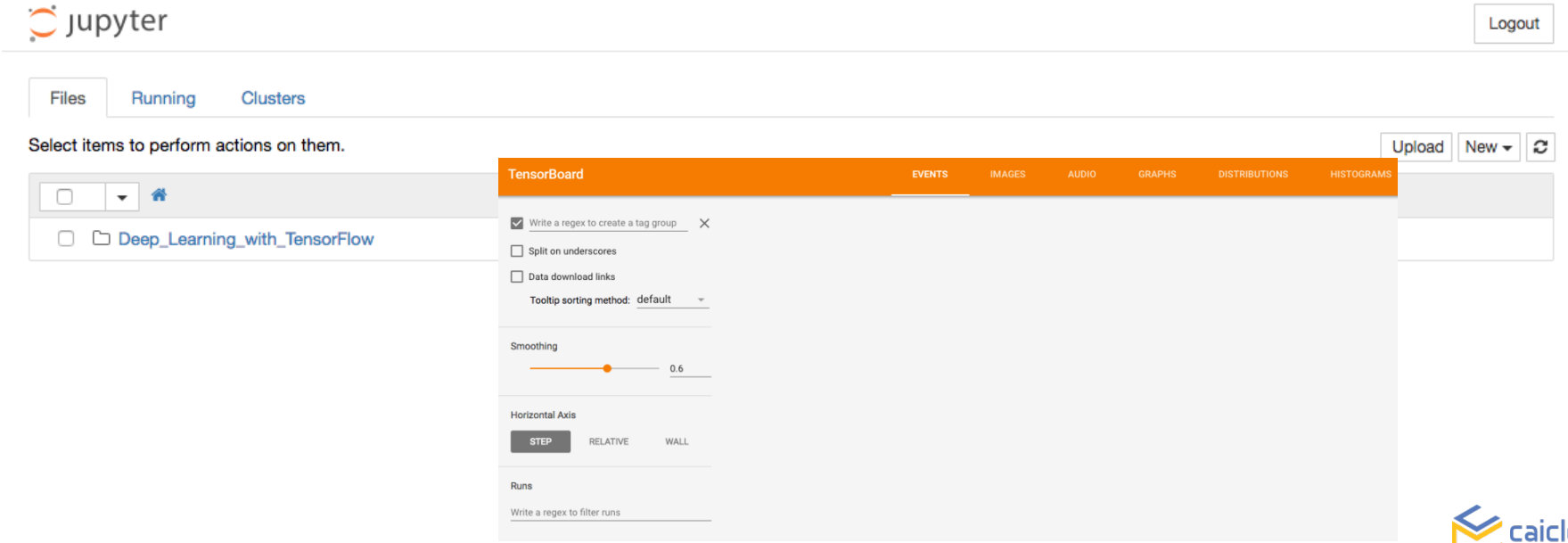
TensorFlow: Machine Learning for Everyone



- docker run -p 8888:8888 -p 6006:6006 cargo.caicloud.io/tensorflow/tensorflow:1.0.0

Jupyter Editor Port

TensorBoard Port



TensorFlow Introduction -- Hello World

```python
#coding=utf-8
#
#Copyright 2017 caicloud authors. All rights reserved.
#

import tensorflow as tf

# tensorflow version 1.1.0
print "tensorflow version: " + tf.__version__

# tensorflow 通过 session 维护上下文，所有执行都需要通过 session
session = tf.InteractiveSession()

with tf.name_scope('input'):
    # 数据都存储在 "tensor" 中
    input1 = tf.constant([1.0, 2.0, 3.0], name = "input1")
    # 变量都存储在 "variable" 中
    input2 = tf.Variable(tf.random_uniform([3]), name = "input2")

# 在运行前，所有变量都需要初始化
tf.global_variables_initializer().run()

with tf.name_scope('add'):
    output = tf.add(input1, input2, name = "add")

# 把日志文件写入本地
writer = tf.summary.FileWriter("./log", session.graph)

# 所有结果都需要先运行才能获取
print output.eval()
```

session

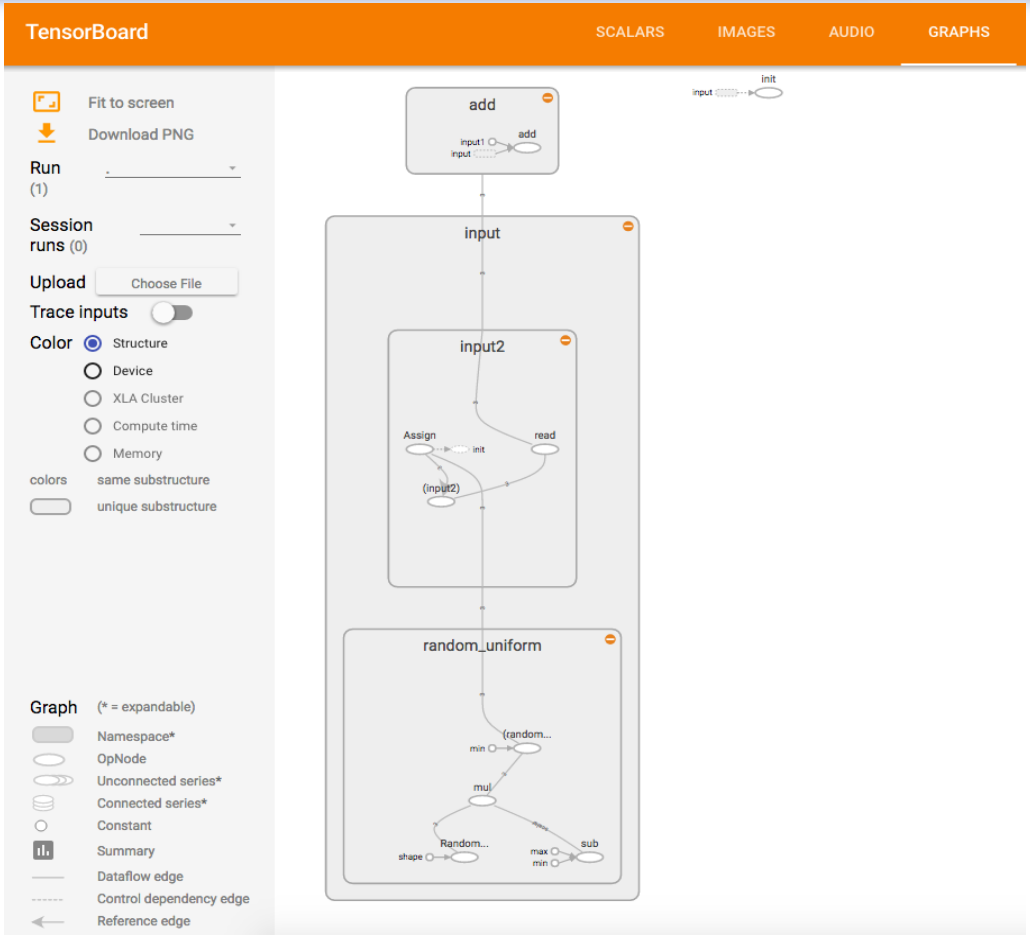
name_scope

constant

variable

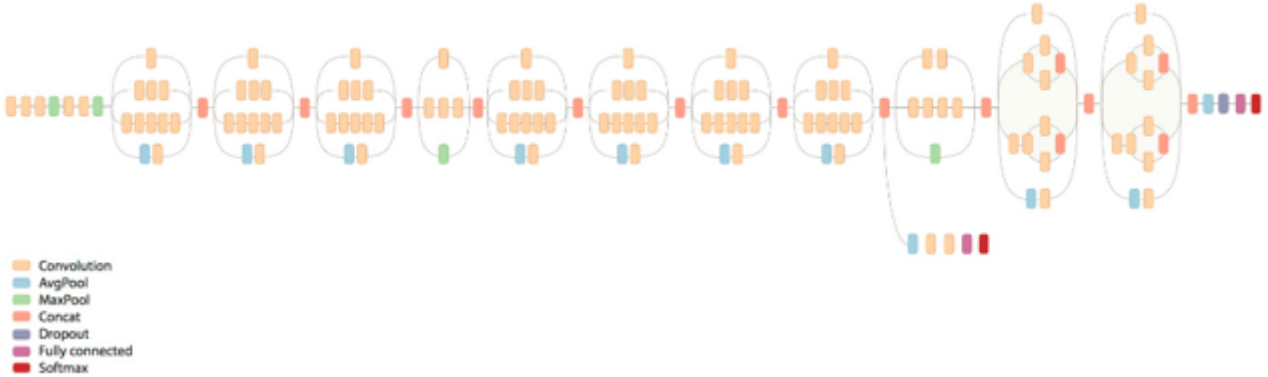
$ tensorboard --logdir ./log/

Starting TensorBoard at http://0.0.0.0:6006

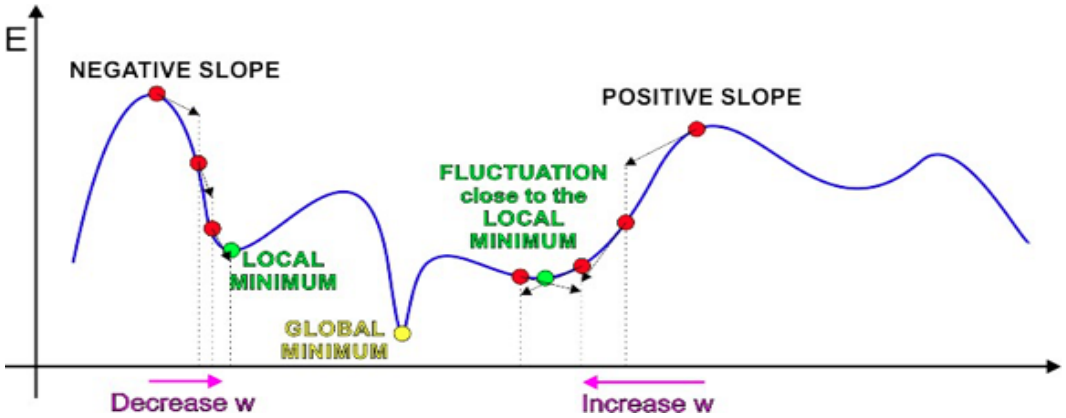


- Computational resource requirement

  Inception-v3 model for ImageNet
  - 25 million parameters
  - 5 billion multiplication/addition operations each inference/forward-prorogate



- Computational resource requirement
  - Structure of neural network is complex, hard to optimize
  - Iterative optimization algorithm -- gradient descent method
  - Need massive data and massive computation
  - It takes **six months** to reach 78% accuracy using single GPU




+
Cluster on
TensorFlow
kubernetes

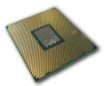
- on local machine

```
with tf.device("/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
with tf.device("/gpu:0"):  // calculation
    output = tf.matmul(input + var1) + var2
    loss = loss_function(output)
```

client

/job:local/task:0

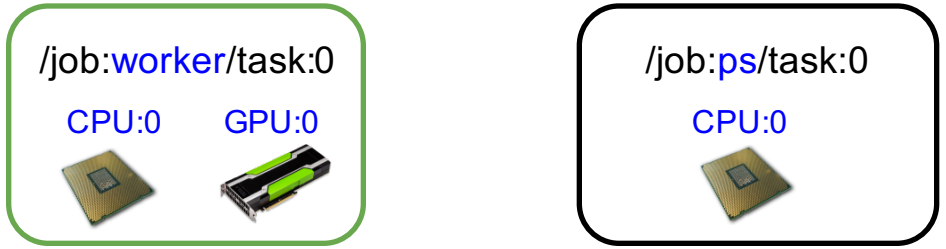
CPU:0 GPU:0



- on cluster (1 Worker + 1 PS)

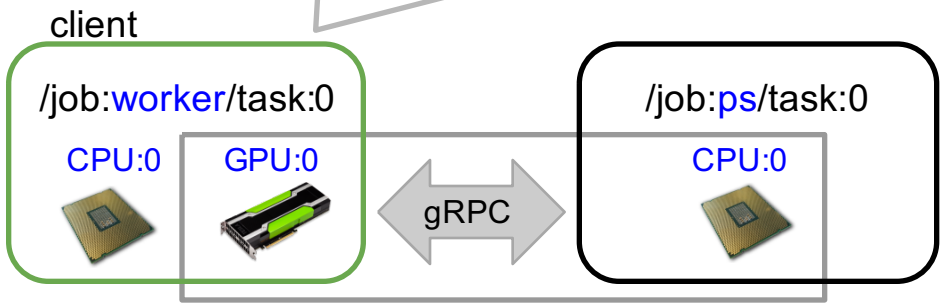
```
with tf.device("/job:ps/task:0/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
with tf.device("/job:worker/task:0/gpu:0"):  // calculation
    output = tf.matmul(input + var1) + var2
    loss = loss_function(output)
```

client



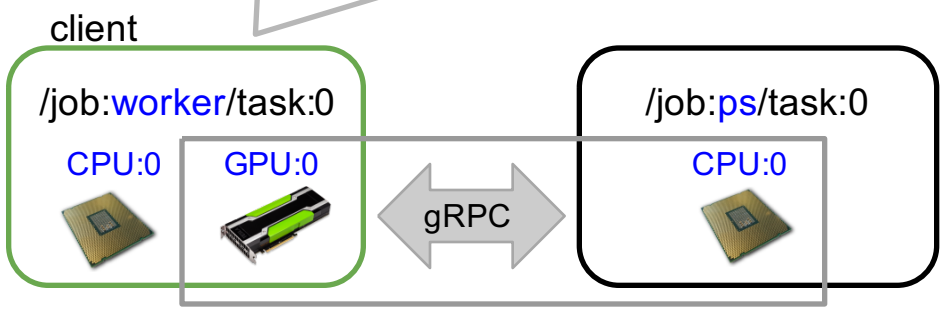
- on cluster (1 Worker + 1 PS)

```
with tf.device("/job:ps/task:0/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
with tf.device("/job:worker/task:0/gpu:0"):  // calculation
    output = tf.matmul(input + var1) + var2
    loss = loss_function(output)
```



- on cluster (1 Worker + 1 PS)

```
with tf.device("/job:ps/task:0/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
with tf.device("/job:worker/task:0/gpu:0"):  // calculation
    output = tf.matmul(input + var1) + var2
    loss = loss_function(output)
```

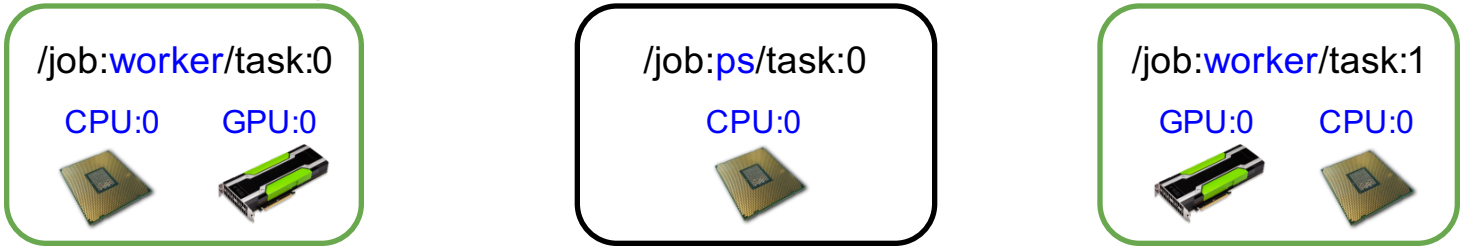


- PS task
  - Variables
  - Update parameters
- Worker task
  - Pre-processing
  - Loss calculation
  - Back-propagation

- on cluster (2 Workers + 1 PS)

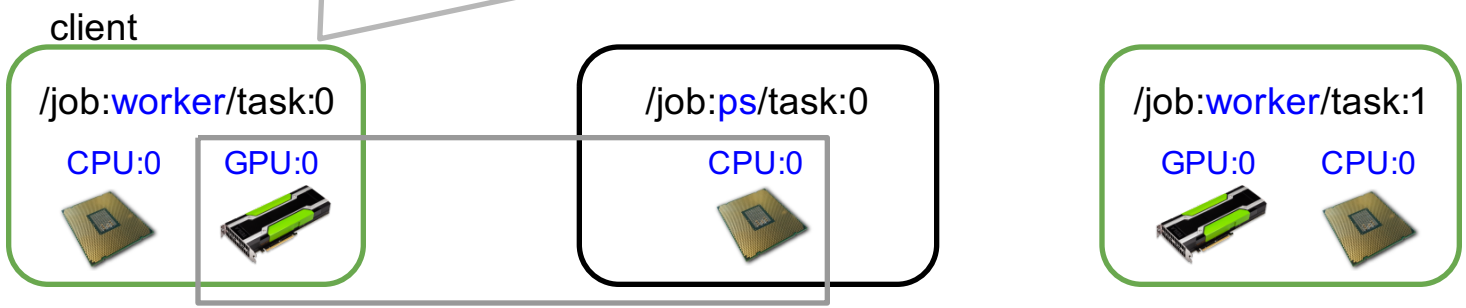
```
with tf.device("/job:ps/task:0/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
inputs = tf.split(0, num_workers, input)
outputs = []
for i in range(num_workers)
    with tf.device("/job:worker/task:%d/gpu:0" % i):  // calculation
        outputs.append(tf.matmul(input + var1) + var2)
loss = loss_function(output)
```

client



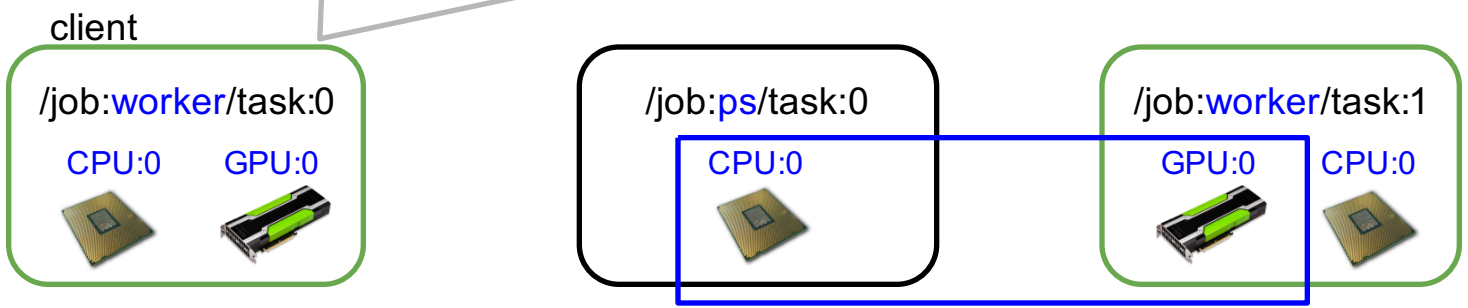
- on cluster (2 Workers + 1 PS)

```
with tf.device("/job:ps/task:0/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
inputs = tf.split(0, num_workers, input)
outputs = []
for i in range(num_workers)
    with tf.device("/job:worker/task:%d/gpu:0" % i):  // calculation
        outputs.append(tf.matmul(input + var1) + var2)
loss = loss_function(output)
```



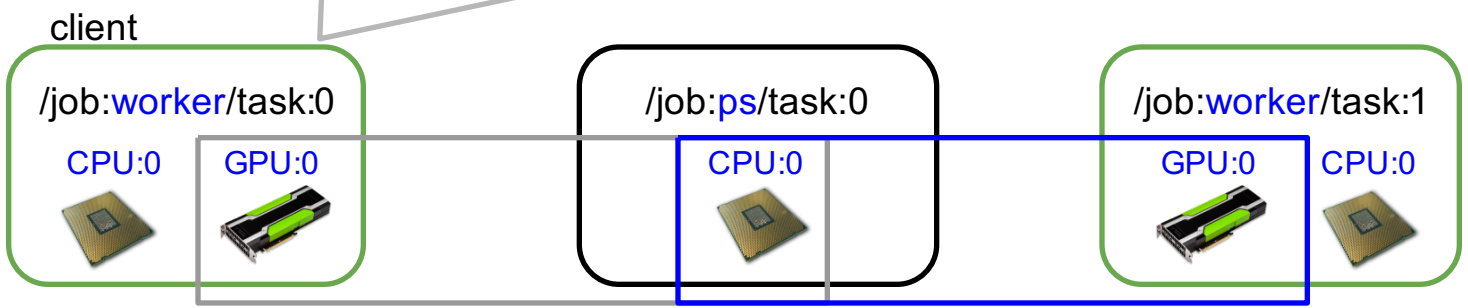
- on cluster (2 Workers + 1 PS)

```
with tf.device("/job:ps/task:0/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
inputs = tf.split(0, num_workers, input)
outputs = []
for i in range(num_workers)
    with tf.device("/job:worker/task:%d/gpu:0" % i):  // calculation
        outputs.append(tf.matmul(input + var1) + var2)
loss = loss_function(output)
```



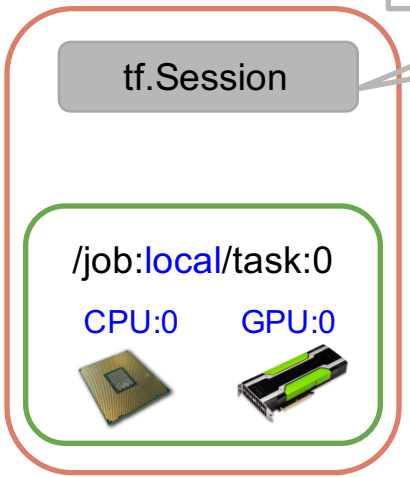
- on cluster (2 Workers + 1 PS)

```
with tf.device("/job:ps/task:0/cpu:0"):  // parameters
    var1 = tf.Variable(...)
    var2 = tf.Variable(...)
inputs = tf.split(0, num_workers, input)
outputs = []
for i in range(num_workers)
    with tf.device("/job:worker/task:%d/gpu:0" % i):  // calculation
        outputs.append(tf.matmul(input + var1) + var2)
loss = loss_function(output)
```



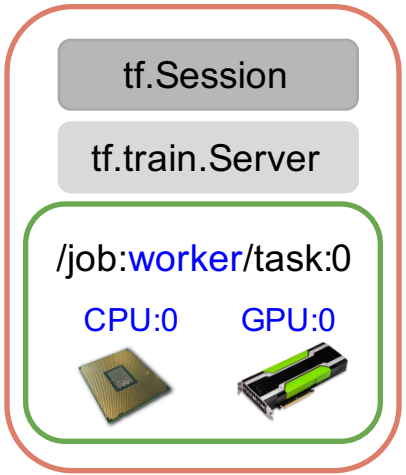
- runs on local machine

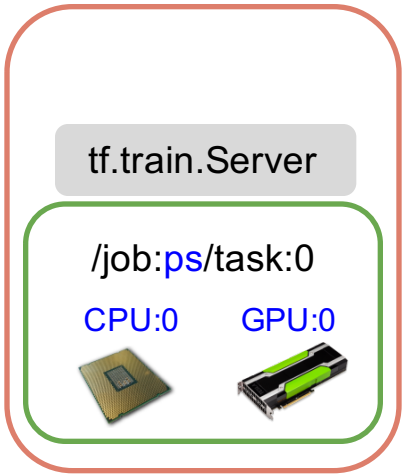
```
with tf.Session() as sess  // declare session
    sess.run(init_operation)  // init
    for _ in range(NUM_STEPS)  // training iteration
        sess.run(train_operation)  // training step
```



- runs on cluster

- runs on cluster





- runs on cluster

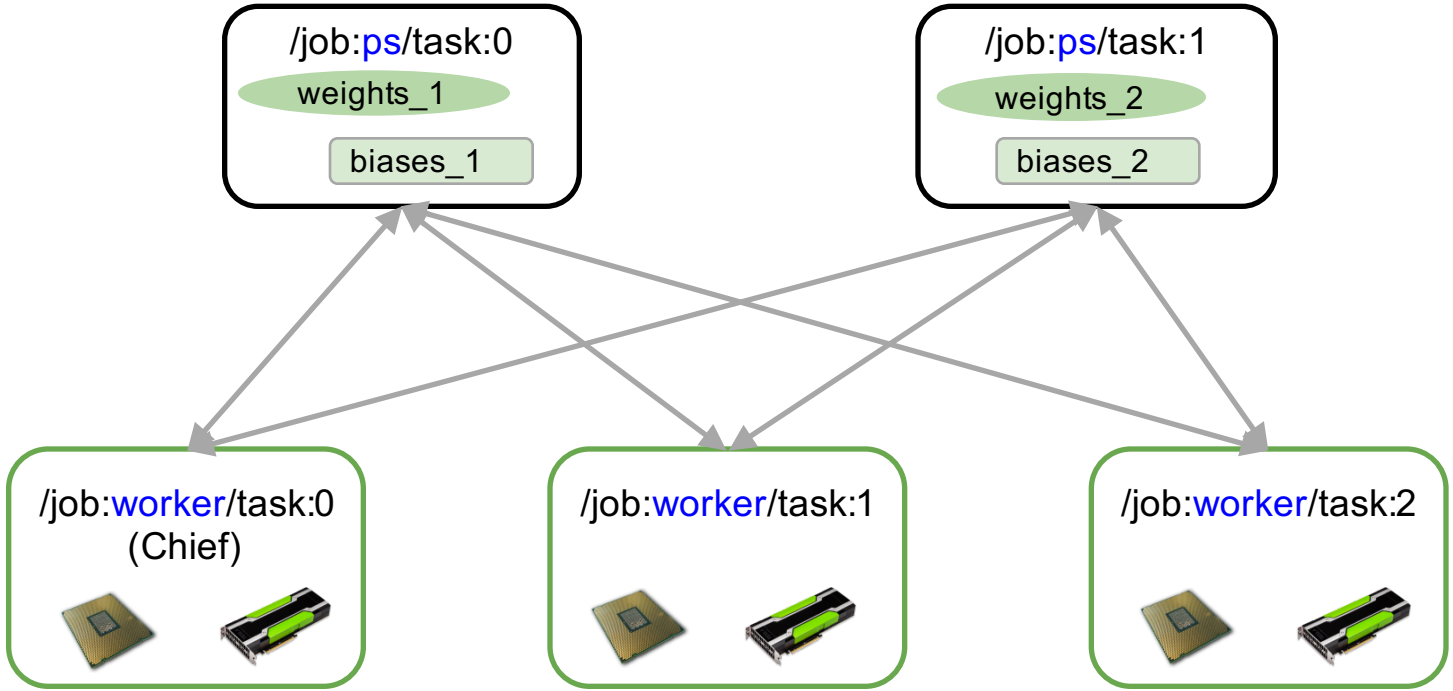
- runs on cluster

```
# in worker 0
clusterSpec = tf.train.ClusterSpec({        // declare cluster spec
    "worker": ["192.168.1.100:2222", ...],
    "ps": ["192.168.1.101:2222", ...]})
server = tf.train.Server(clusterSpec, job_name = "worker", task_index = 0)
with tf.Session(server.target) as sess  // create session using server.target
    # do something ...
```

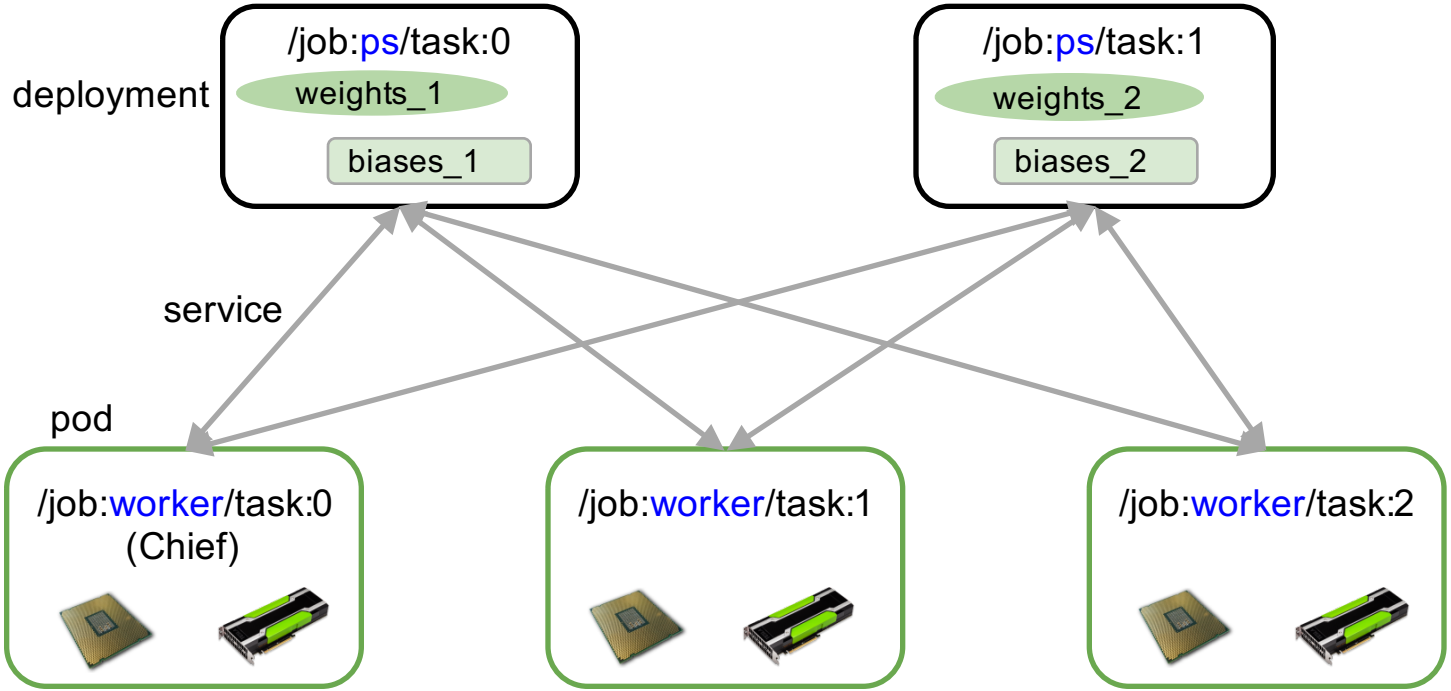
- runs on cluster

```
# in ps 0
clusterSpec = tf.train.ClusterSpec({        // declare cluster spec
    "worker": ["192.168.1.100:2222", ...],
    "ps": ["192.168.1.101:2222", ...]})
server = tf.train.Server(clusterSpec, job_name = "ps", task_index = 0)
 # wait for incoming connections …
server.join()      // waiting for connection...
```

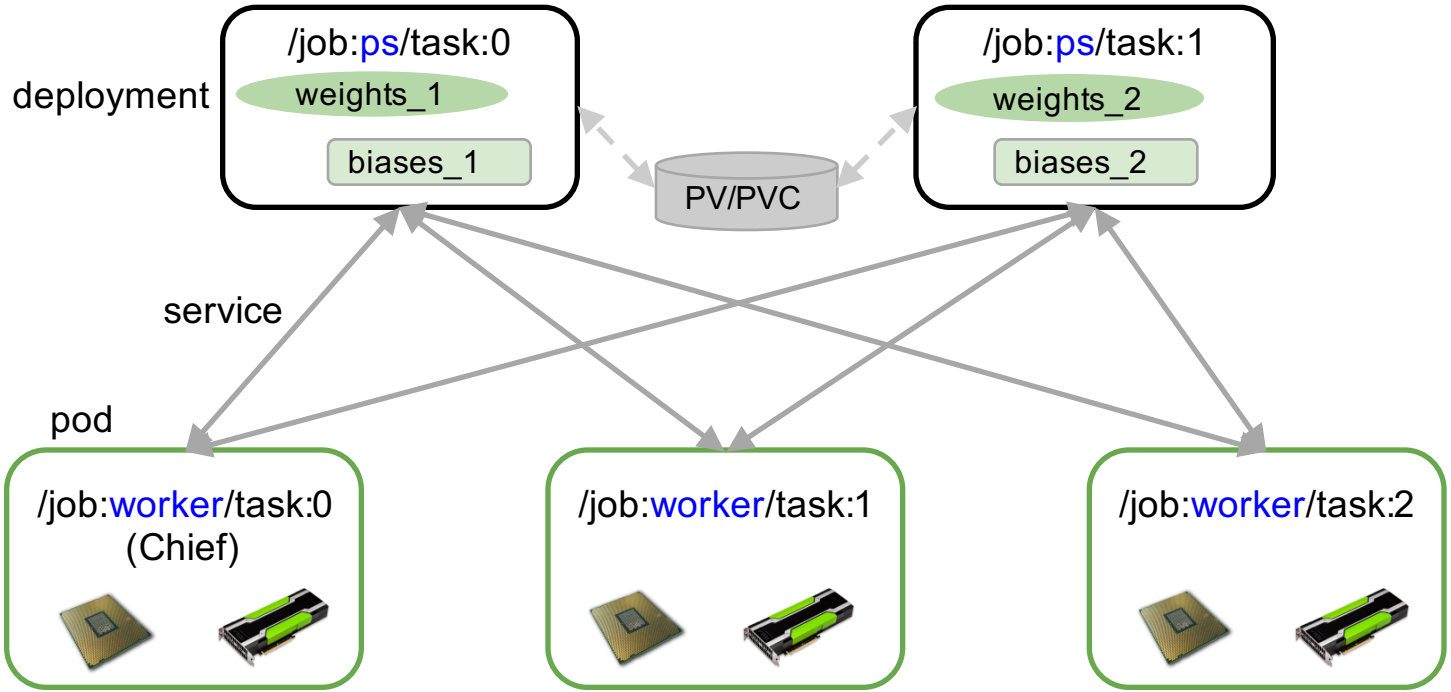
- on kubernetes



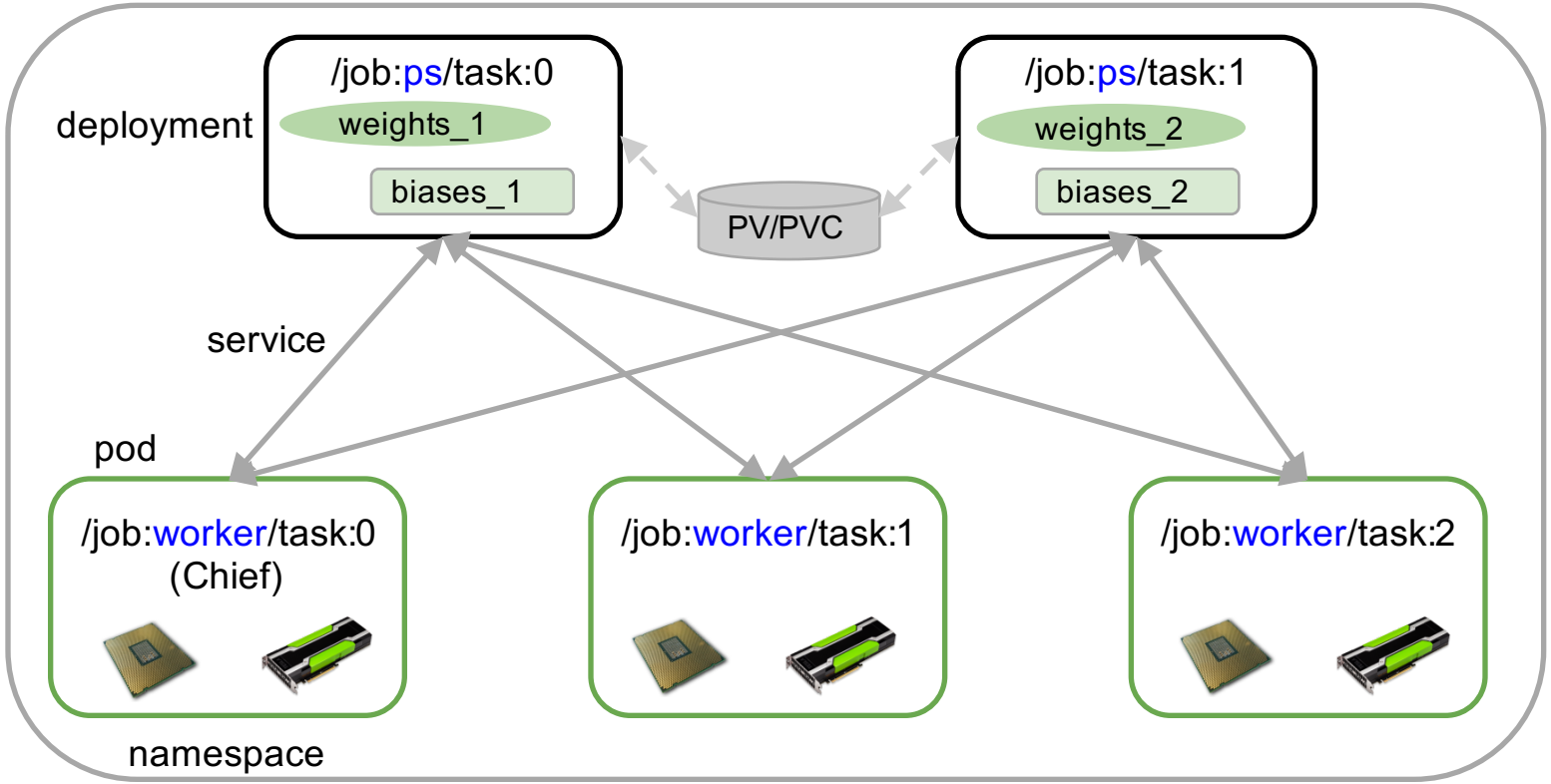
- on kubernetes



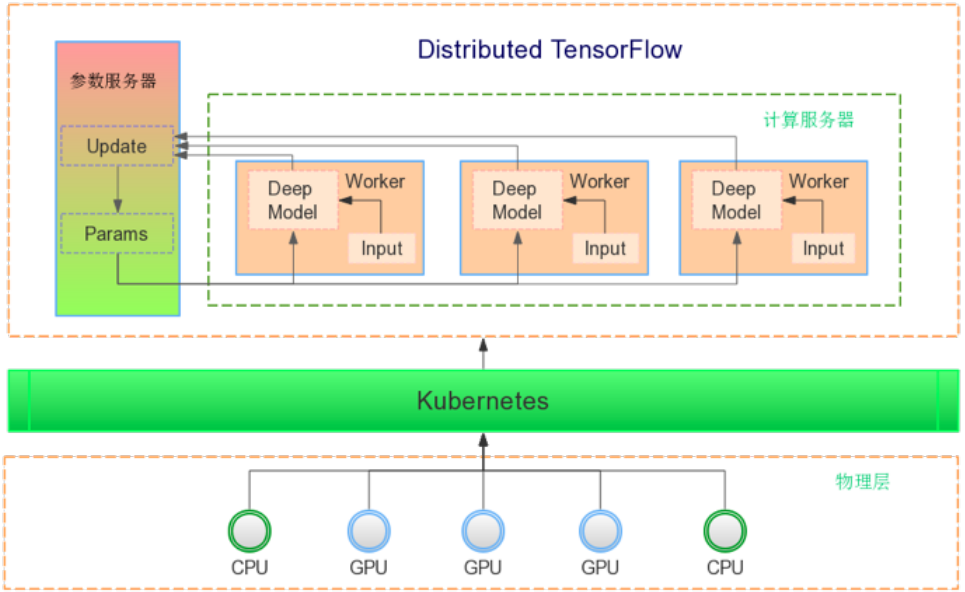
- on kubernetes



- on kubernetes



- on kubernetes



Cluster Management

- **Deployment/pod** for lifecycle management
- **Namespace** for resource isolation
- Monitoring、alerting & logging

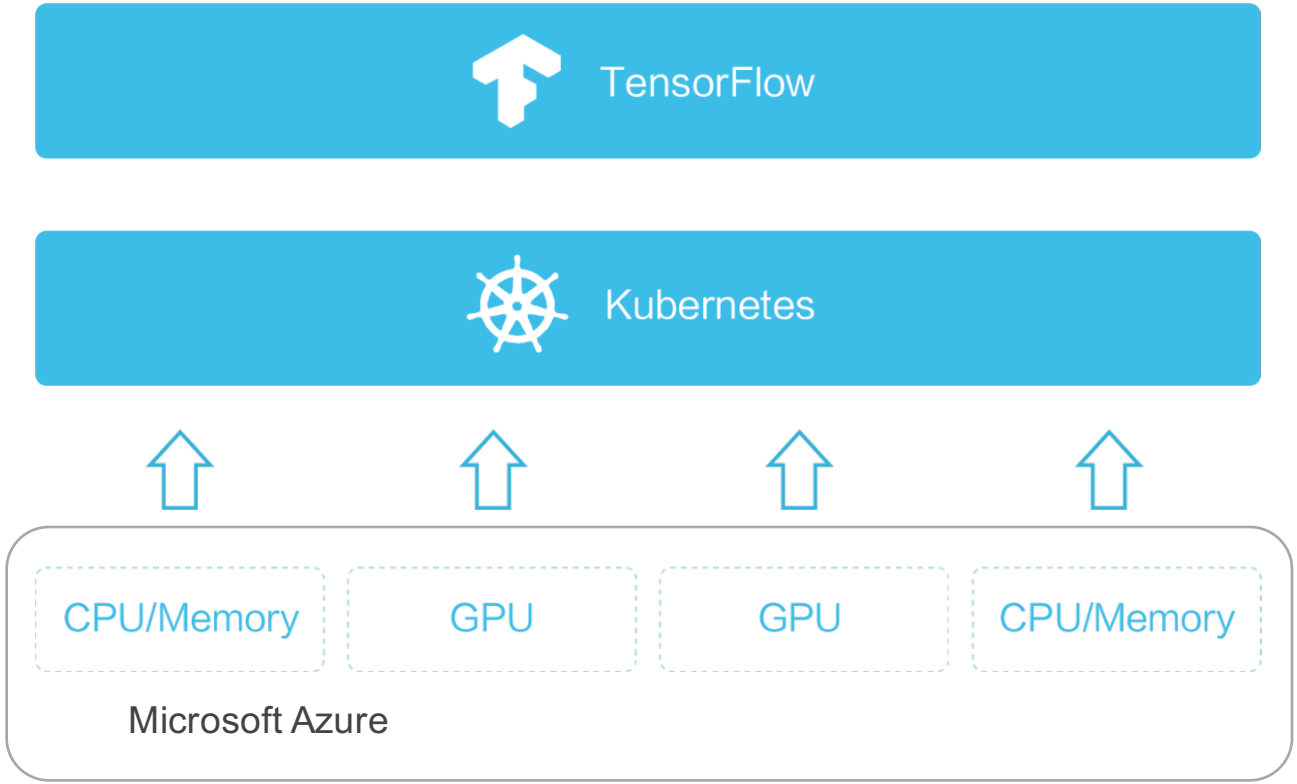
Network

- **Service** for service discovery

Storage

- **PV/PVC** for persistent storage
- GlusterFS、ceph for distributed storage

TaaS Introduction -- TensorFlow as a Service



TaaS Demo

https://taas.caicloud.io

Q&A

caicloud
才云

Thanks！