后厂理工学院
houchangtech.com

Lane Segmentation Week 7

HCT CV Course

学习目标

- 掌握权重初始化
- 掌握模型部署方法
- 项目问题讲解

TP/FP/FN/TN

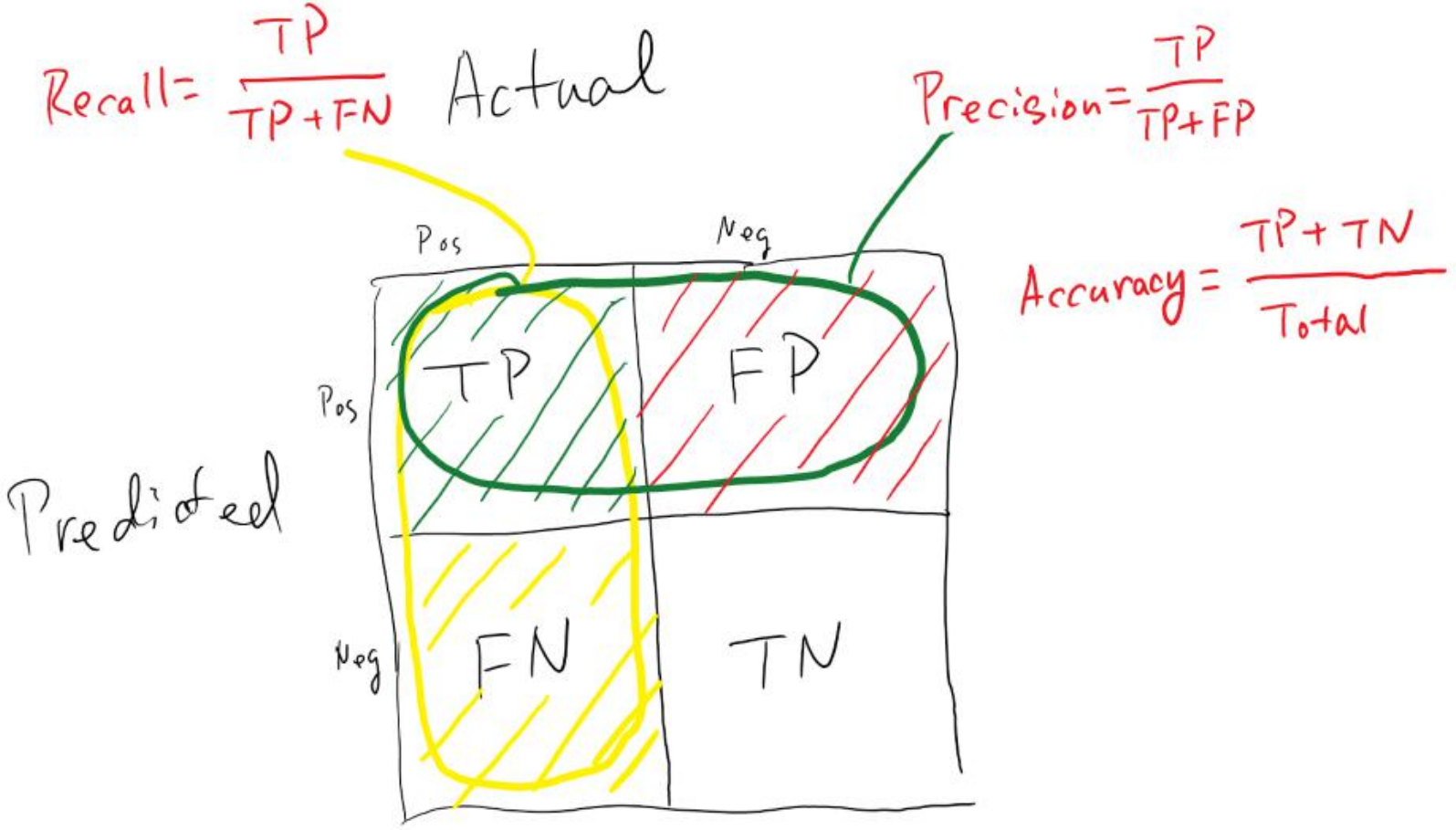
我们可以使用一个 2x2 混淆矩阵来总结我们的"狼预测"模型，该矩阵描述了所有可能出现的结果（共四种）：

真正例 (TP)： • 真实情况：受到狼的威胁。 • 牧童说："狼来了。" • 结果：牧童是个英雄。	**假正例 (FP)**： • 真实情况：没受到狼的威胁。 • 牧童说："狼来了。" • 结果：村民们因牧童吵醒他们而感到非常生气。
假负例 (FN)： • 真实情况：受到狼的威胁。 • 牧童说："没有狼"。 • 结果：狼吃掉了所有的羊。	**真负例 (TN)**： • 真实情况：没受到狼的威胁。 • 牧童说："没有狼"。 • 结果：大家都没事。

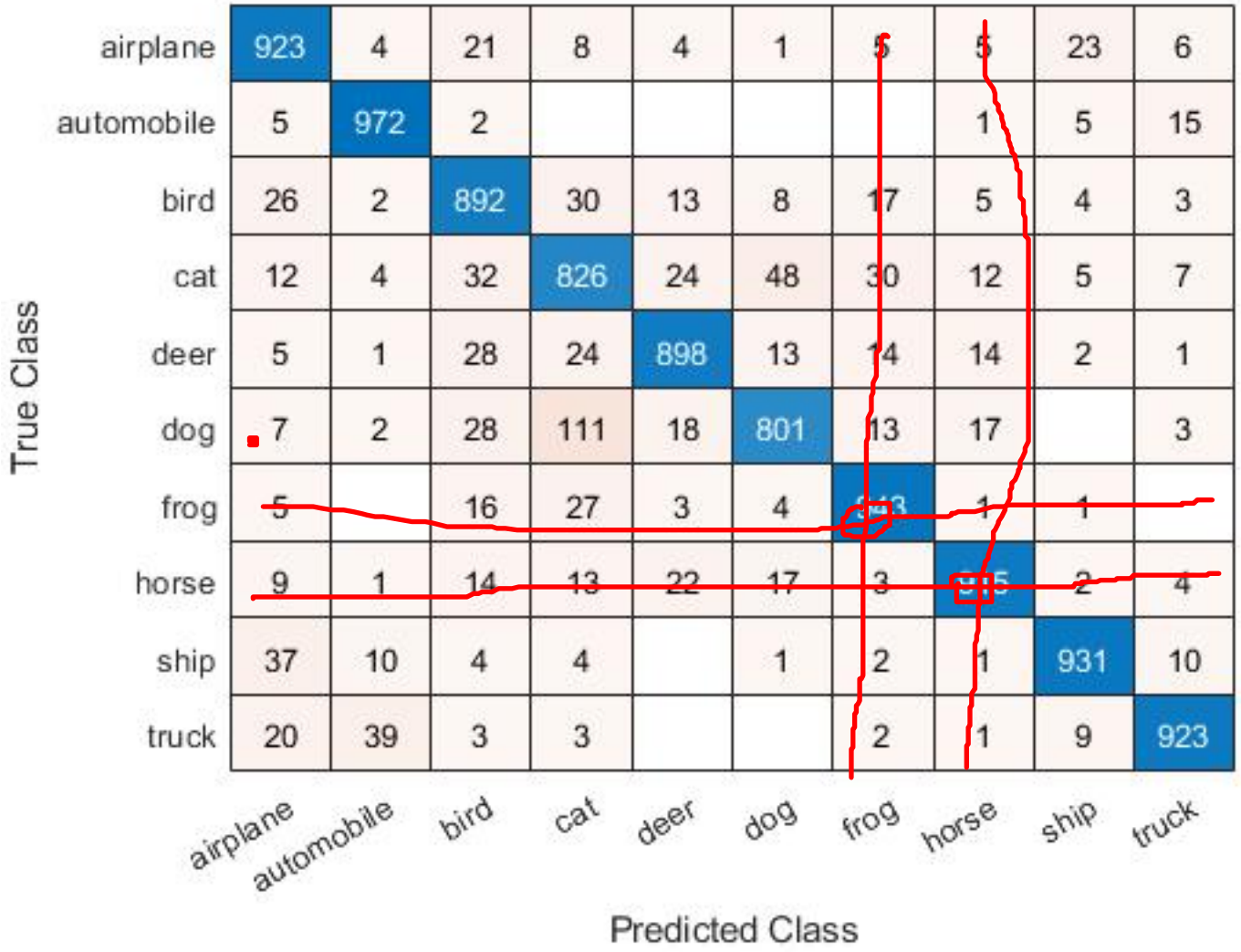
真正例是指模型将正类别样本正确地预测为正类别。同样，**真负例**是指模型将负类别样本正确地预测为负类别。

假正例是指模型将负类别样本错误地预测为正类别，而**假负例**是指模型将正类别样本错误地预测为负类别。

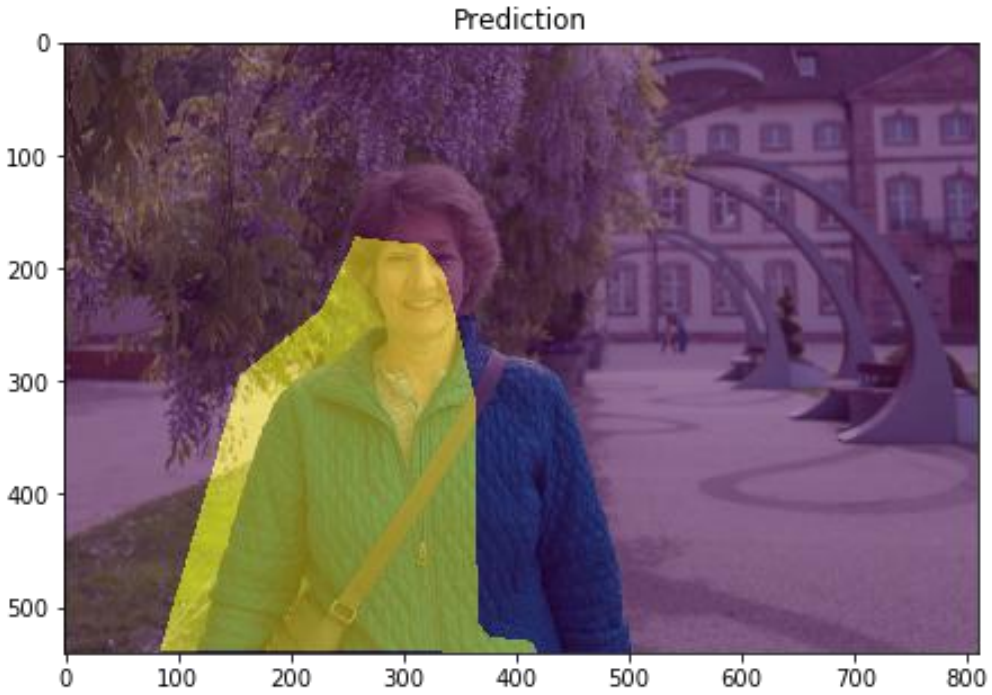
Precision/Recall



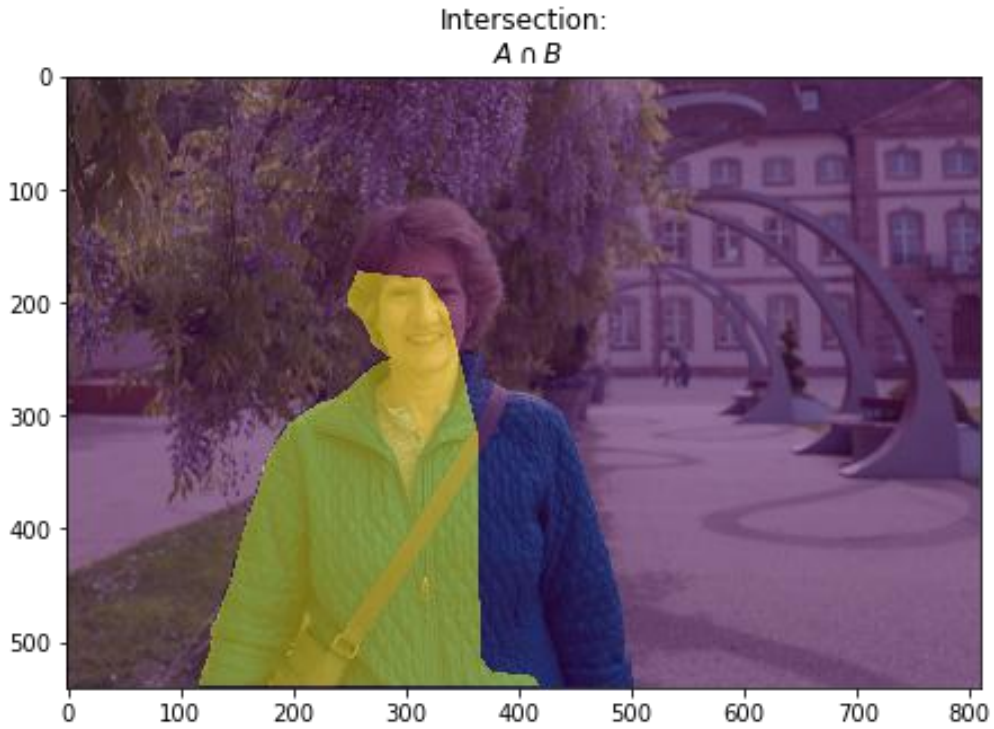
confusion matrix

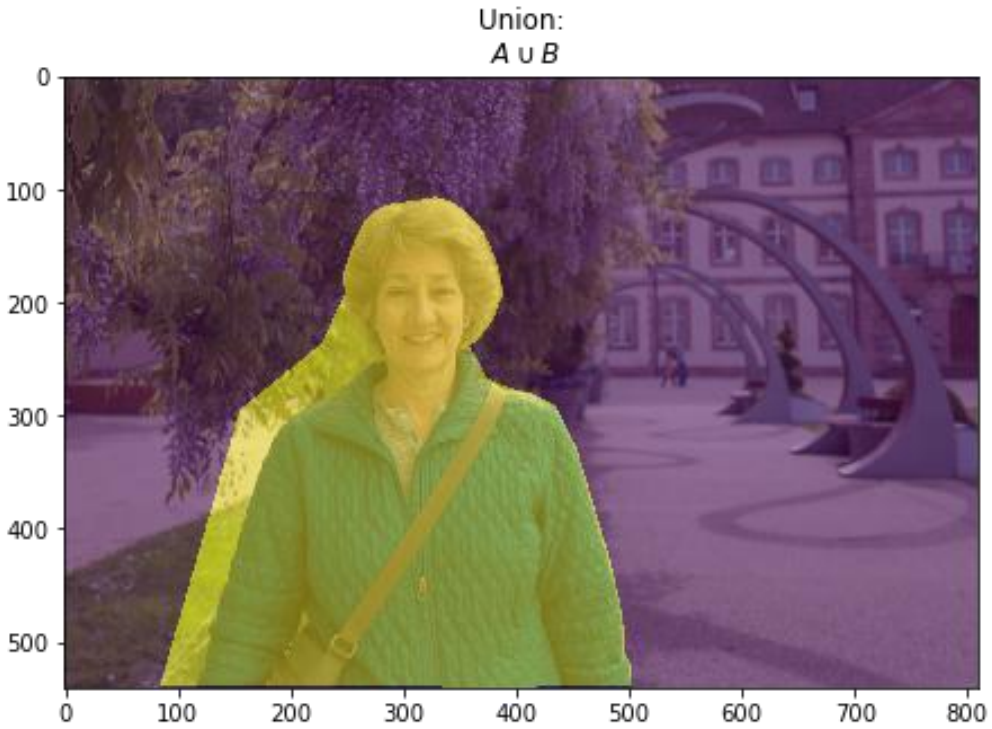


Metrics



Metrics





mIoU

$$MIoU = \frac{1}{k+1}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k}p_{ij} + \sum_{j=0}^{k}p_{ji} - p_{ii}}$$

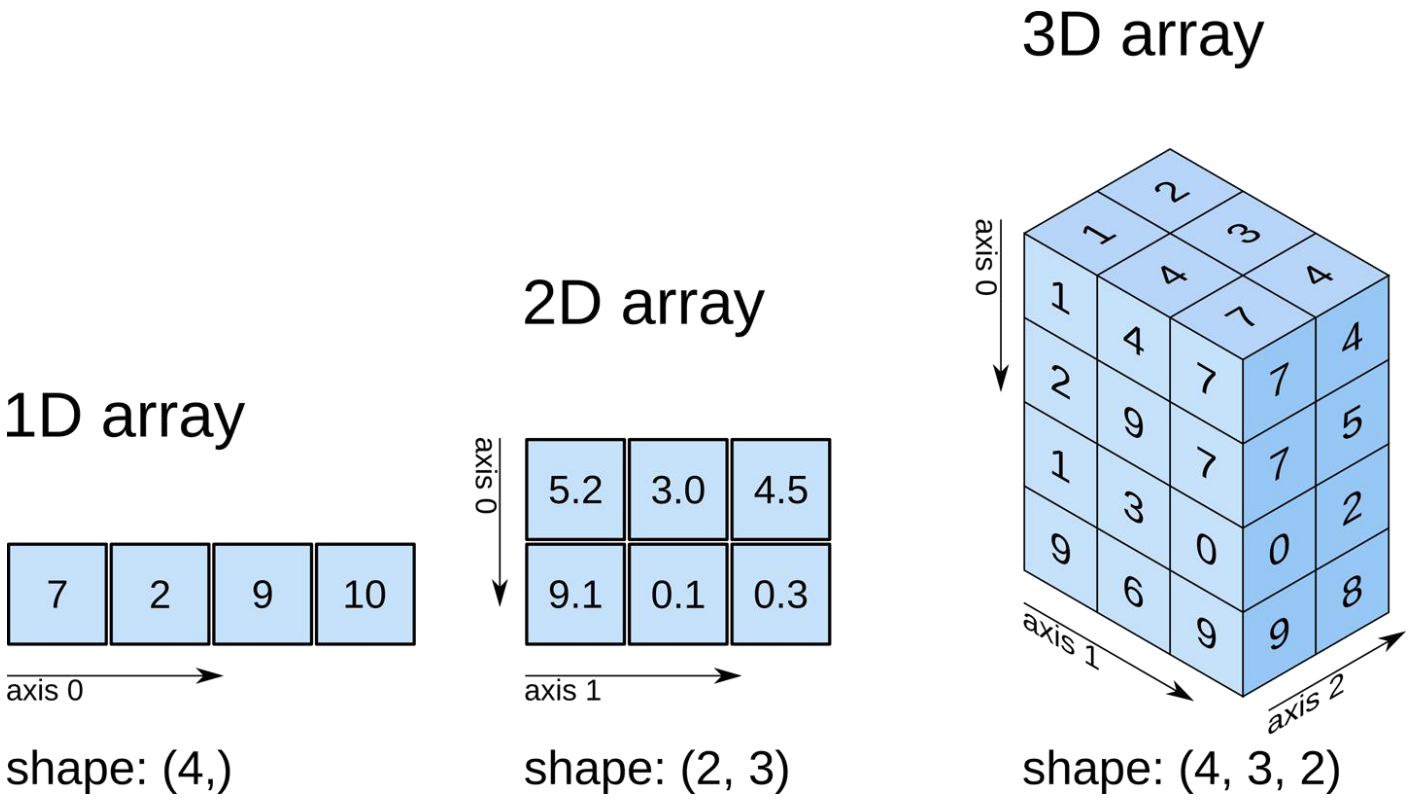
p_{ij} 表示真实值为 i，被预测为 j 的数量。

Cross-Entropy Loss

- In a sense, IoU is to segmentation what an F1 score is to classification. Both are non-differentiable, and not normally optimized directly. Optimizing cross entropy loss is a common proxy for these scores, that usually leads to decent performance, provided everything else has been setup correctly, e.g regularization, stopping training at an appropriate time.

Numpy

- numpy.argmax(a, axis=None, out=None)



Note

Tensorflow 2.0

- https://www.tensorflow.org/api_docs/python/tf/keras/metrics/MeanIoU?version=stable

Weight Initialization

- Initialization of neural networks isn’t something we think a lot about nowadays. It’s all hidden behind the different Deep Learning frameworks we use, like TensorFlow or PyTorch. However, it’s at the heart of why and how we can make neural networks as deep as they are today, and it was a significant bottleneck just a few years ago.

Weight Initialization

- Why is initialization essential to deep networks? It turns out that if you do it wrong, it can lead to exploding or vanishing weights and gradients. That means that either the weights of the model explode to infinity, or they vanish to 0 (literally, because computers can’t represent infinitely accurate floating point numbers), which make training deep neural networks very challenging. And the deeper the network, the harder it becomes to keep the weights at reasonable values.

Weight Initialization

- The aim of weight initialization is to prevent layer activation outputs from exploding or vanishing during the course of a forward pass through a deep neural network. If either occurs, loss gradients will either be too large or too small to flow backwards beneficially, and the network will take longer to converge

Initializing all the weights with zeros

- Initializing all the weights with zeros leads the neurons to learn the same features during training.
- In fact, any constant initialization scheme will perform very poorly.

Break Symmetry

- If two hidden units have the same inputs and same activation function, then they must have different initial parameters
- It’s desirable to initialize each unit to compute a different function

Random initialization

- Despite breaking the symmetry, initializing the weights with values (i) too small or (ii) too large leads respectively to (i) slow learning or (ii) divergence.

Random initialization

- Hopefully intializing with small random values does better. The important question is: how small should be these random values be?

Random initialization

- keeping the standard deviation of layers' activations around 1 will allow us to stack several more layers in a deep neural network without gradients exploding or vanishing

AlexNet

- initialization with Gaussian (normal) noise with mean equal to zero and standard deviation set to 0.01 with bias equal to one for some layers
- However, this normal random initialization approach does not work for training very deep networks, especially those that use the ReLU (rectified linear unit) activation function, because of the vanishing and exploding gradient problem

Weight Initialization

- the variance is the square of the standard deviation.
- the variance of a sum of independent random variables is the sum of the variances of the individual random variables

Xavier Initialization

- Understanding the difficulty of training deep feedforward neural networks
- http://proceedings.mlr.press/v9/glorot10a.html

Xavier Initialization

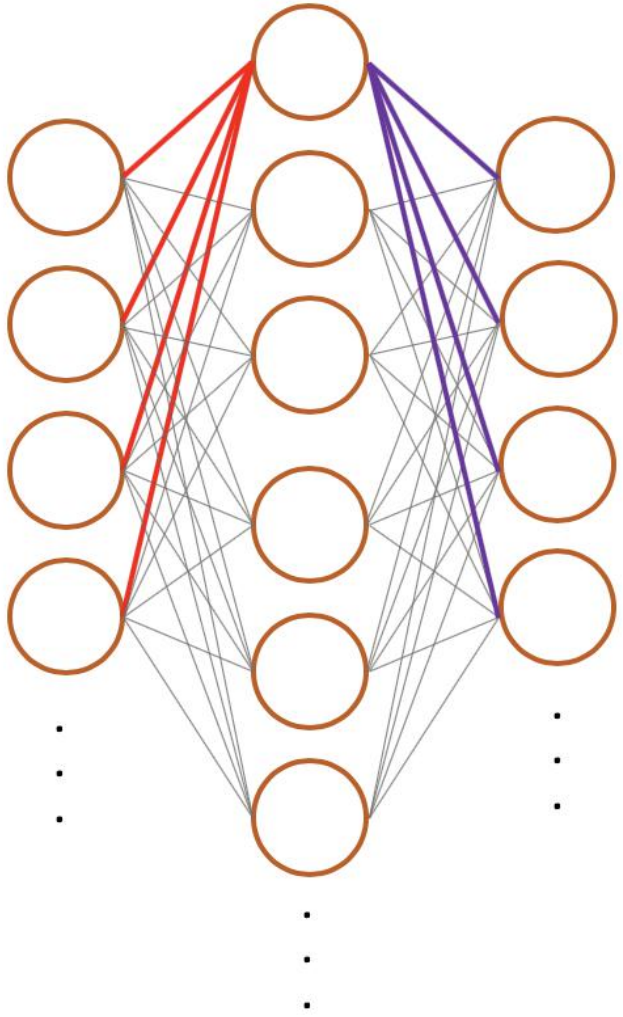
- The normalization factor may therefore be important when initializing deep networks because of the multiplicative effect through layers, and we suggest the following initialization procedure to approximately satisfy our objectives of maintaining activation variances and back-propagated gra dients variance as one moves up or down the network. We call it the normalized initialization

Xavier Initialization

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

based on the assumption that the activations are linear

Xavier Initialization



fan-in is the number of inputs to a hidden unit (*in this simplified diagram*, 4)

fan-out is the number of outputs to a hidden unit (*in this simplified diagram*, 4)

- For MLPs, fan-in was the number of units in the layer below.
- For CNNs however, we have to take into account the number of input feature maps and the size of the receptive fields like so:

```
fan_in = n_feature_maps_in *
receptive_field_height * receptive_field_width
```

```
fan_out = n_feature_maps_out *
receptive_field_height * receptive_field_width
/ max_pool_area
```

where `receptive_field_height` and `receptive_field_width` correspond to those of the conv layer under consideration and `max_pool_area` is the product of the height and width of the max pooling that follows the convolution layer.

activation function

- The choice of activation function ends up playing an important role in determining how effective the initialization method is.

activation function

- Sigmoid
- Softmax
- Tanh
- ReLU
- Leaky ReLU

Kaiming Initialization

- Xavier在tanh中表现的很好，但在ReLU激活函数中表现很差，Kaiming He提出了针对ReLU的初始化方法

Kaiming Initialization

- Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification
- https://arxiv.org/abs/1502.01852

Kaiming Initialization

$$\text{std} = \sqrt{\frac{2}{(1 + a^2) \times \text{fan\_in}}}$$

a=0 if ReLU

Note

Deployment

- Tensorflow Serving
- 独立部署

Tensorflow Serving

- TensorFlow Serving is a flexible, high-performance serving system for machine learning models, designed for production environments. TensorFlow Serving makes it easy to deploy new algorithms and experiments, while keeping the same server architecture and APIs. TensorFlow Serving provides out-of-the-box integration with TensorFlow models, but can be easily extended to serve other types of models and data.

Tensorflow Serving

- https://github.com/tensorflow/serving
- https://www.tensorflow.org/tfx/tutorials/serving/rest_simple

SavedModel

- A SavedModel contains a complete TensorFlow program, including weights and computation. It does not require the original model building code to run, which makes it useful for sharing or deploying (with TFLite, TensorFlow.js, TensorFlow Serving, or TensorFlow Hub).

saved_model_cli

```
!saved_model_cli show --dir {export_path} --all
```

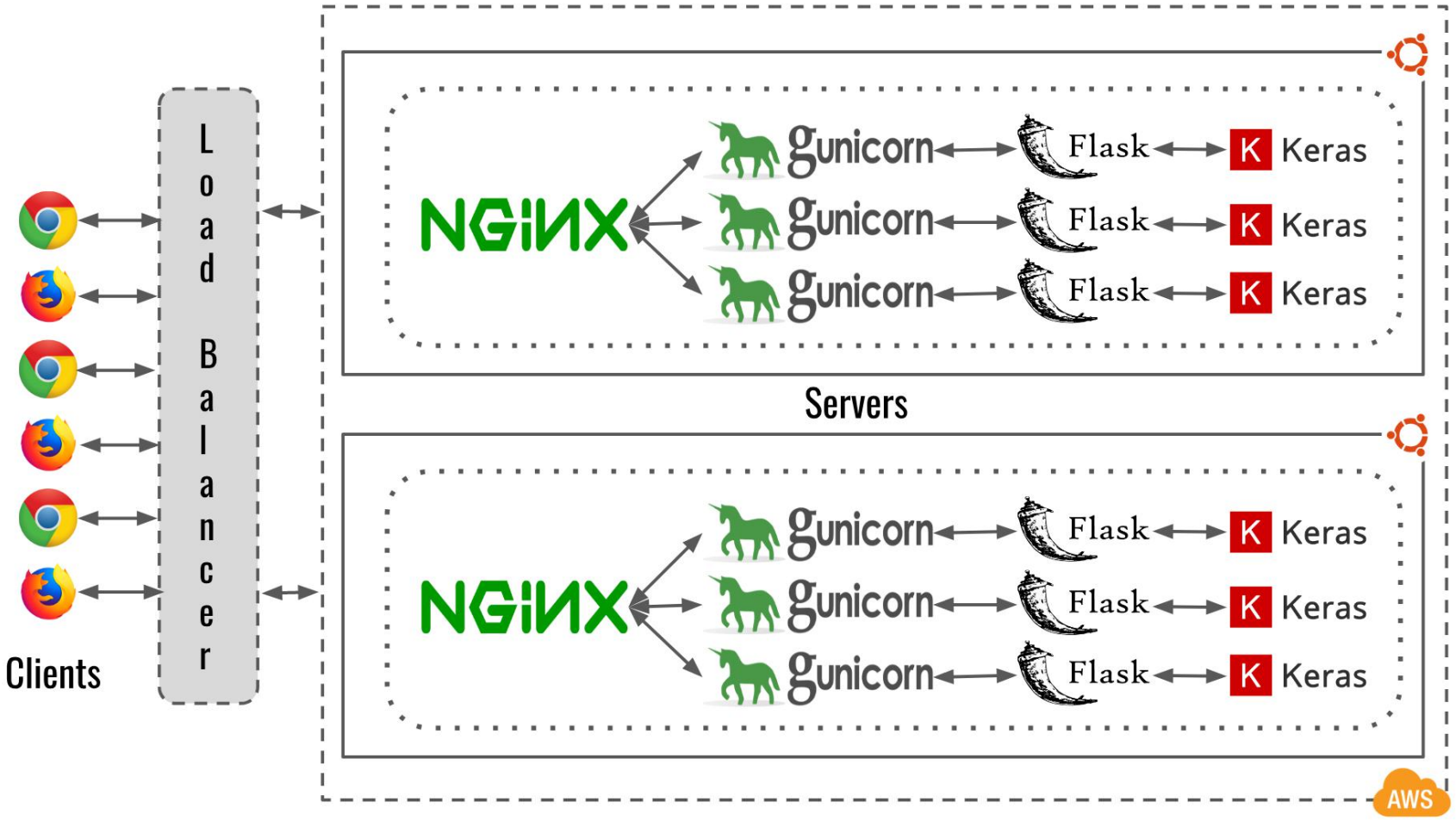
```
MetaGraphDef with tag-set: 'serve' contains the following SignatureDefs:

signature_def['serving_default']:
  The given SavedModel SignatureDef contains the following input(s):
    inputs['input_image'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 28, 28, 1)
        name: Conv1_input:0
  The given SavedModel SignatureDef contains the following output(s):
    outputs['Softmax/Softmax:0'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 10)
        name: Softmax/Softmax:0
  Method name is: tensorflow/serving/predict
```

Note

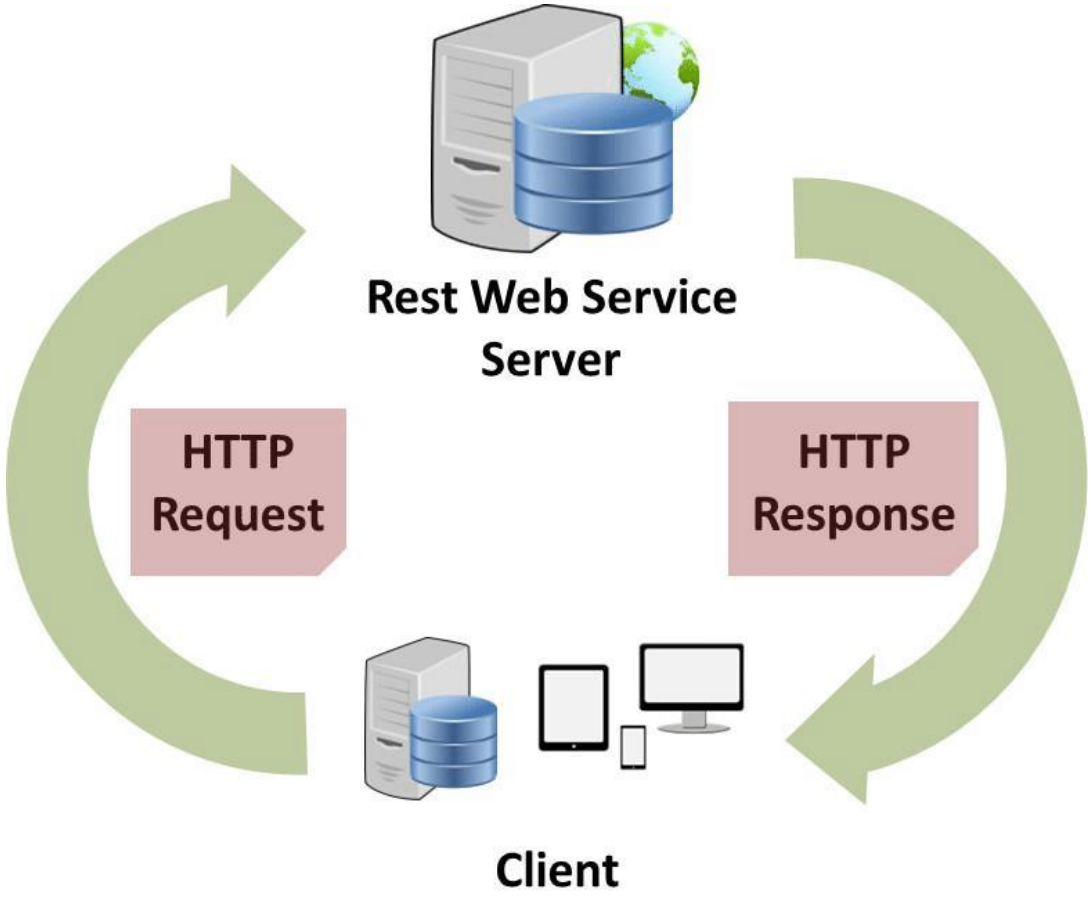
独立部署

- Flask/Django
- Nginx
- Docker
- K8s

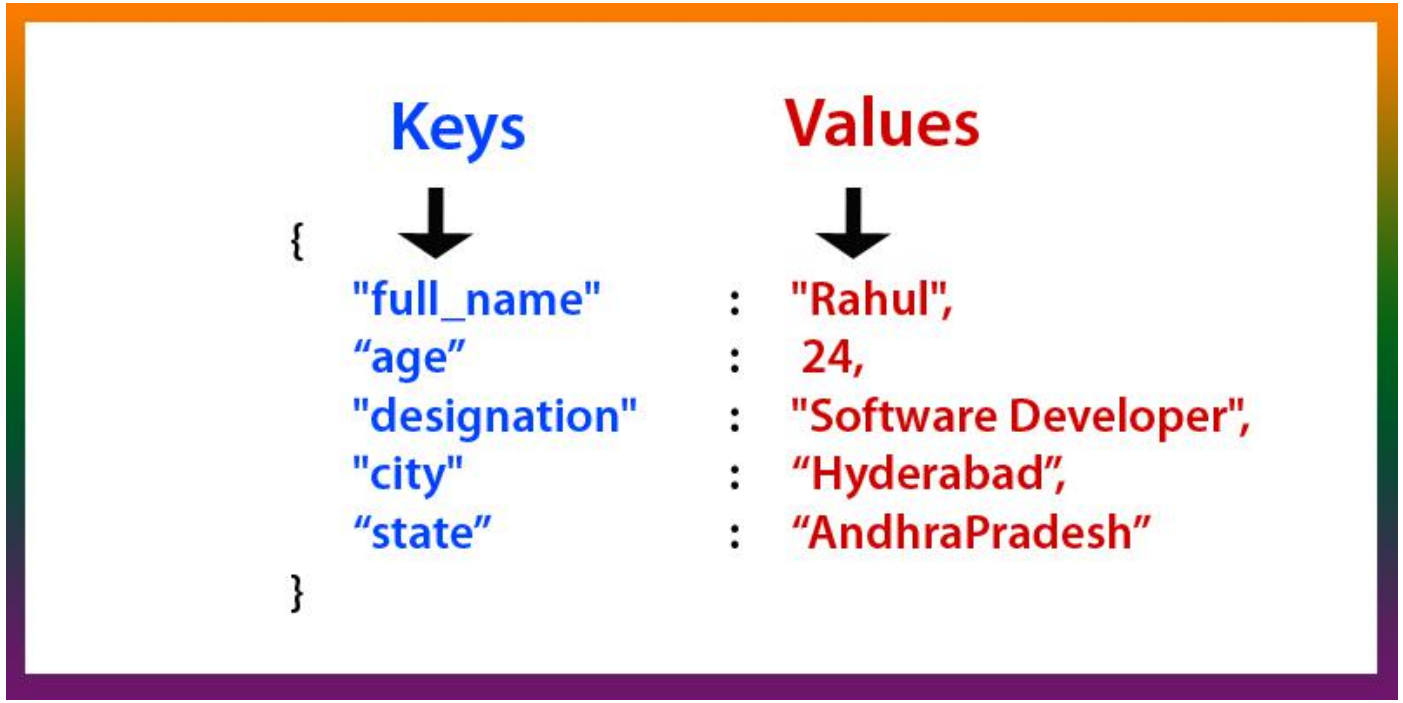
独立部署



RESTful API



Json

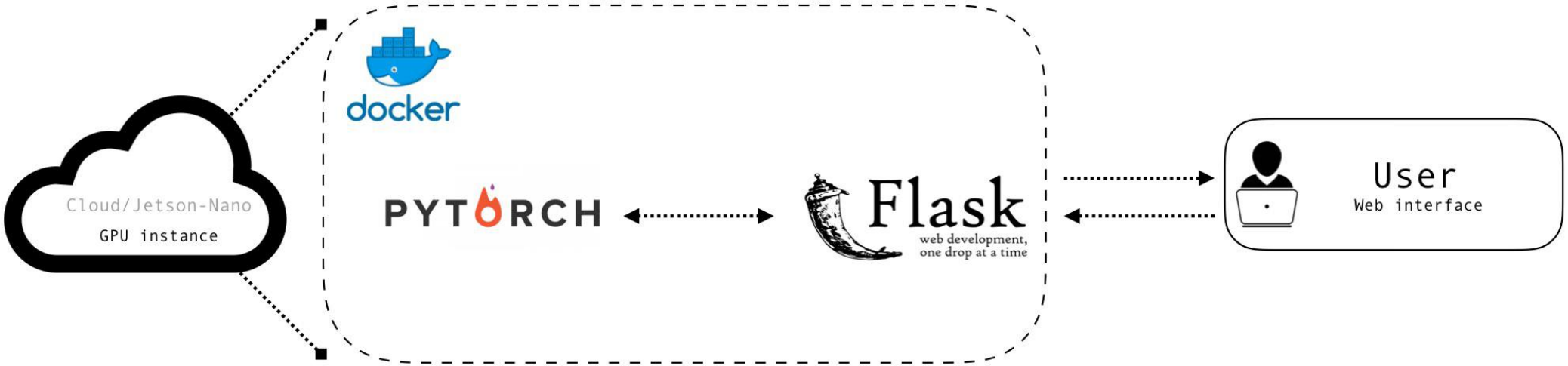


Flask

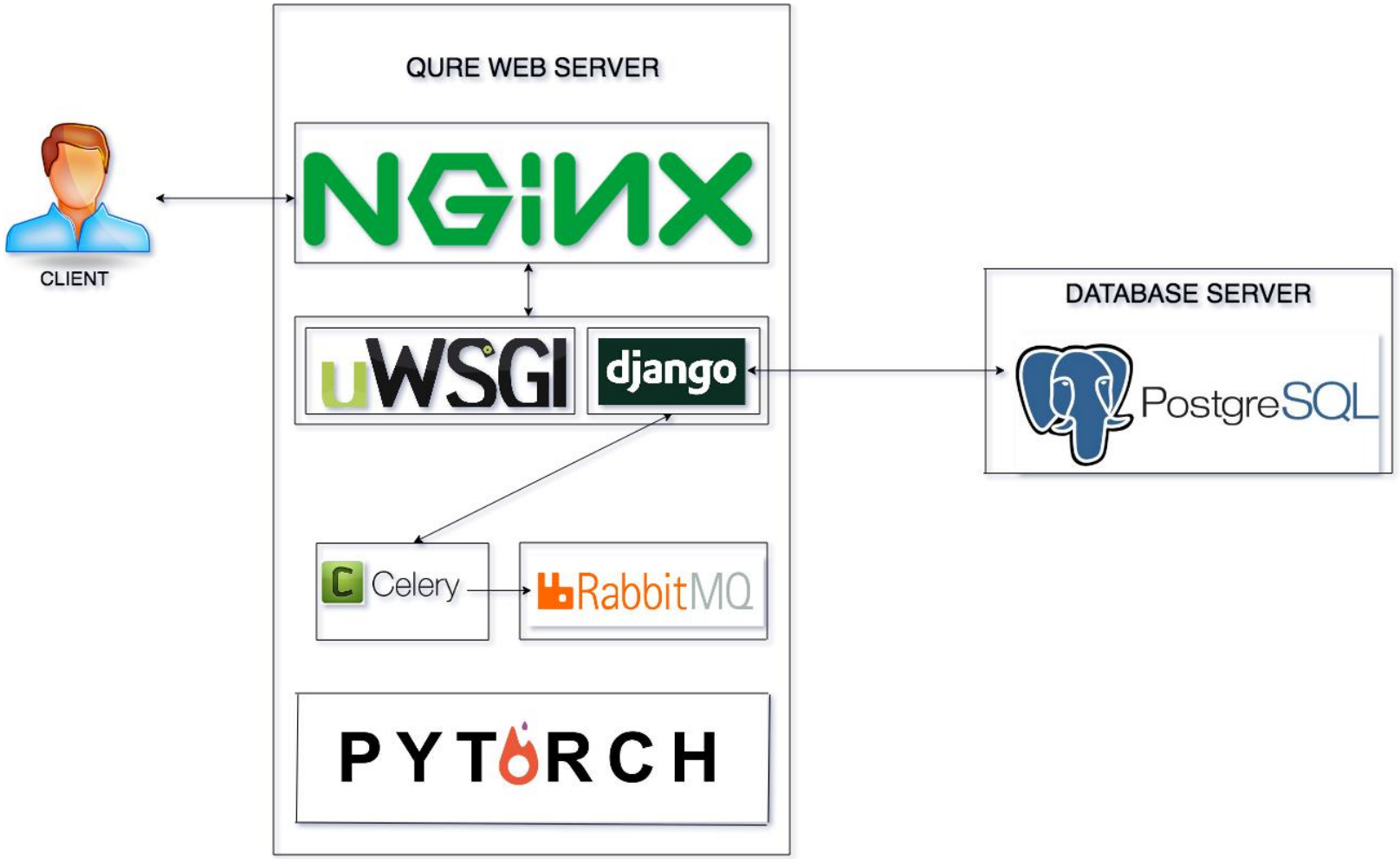
```python
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

```
$ env FLASK_APP=hello.py flask run
 * Serving Flask app "hello"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Flask

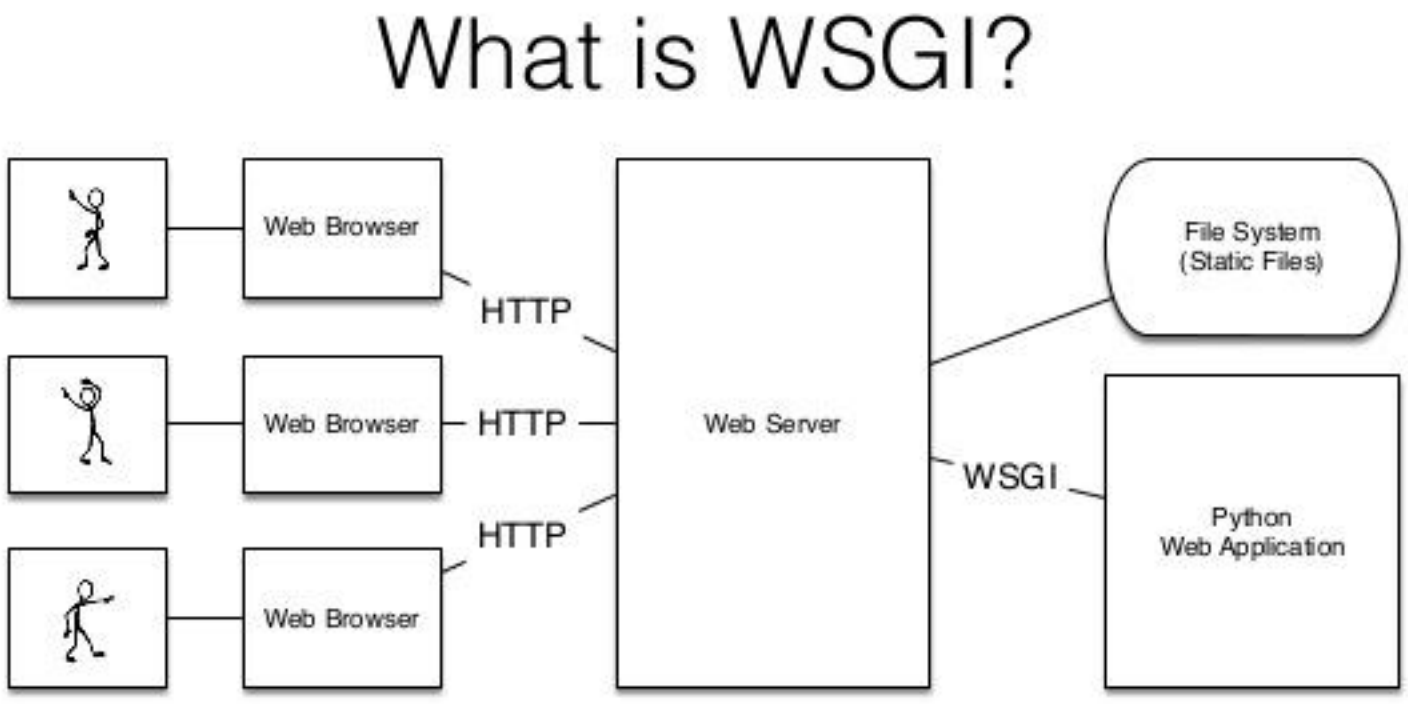

Cloud/Jetson-Nano
GPU instance
docker
PYTORCH
Flask
web development,
one drop at a time
User
Web interface

Django



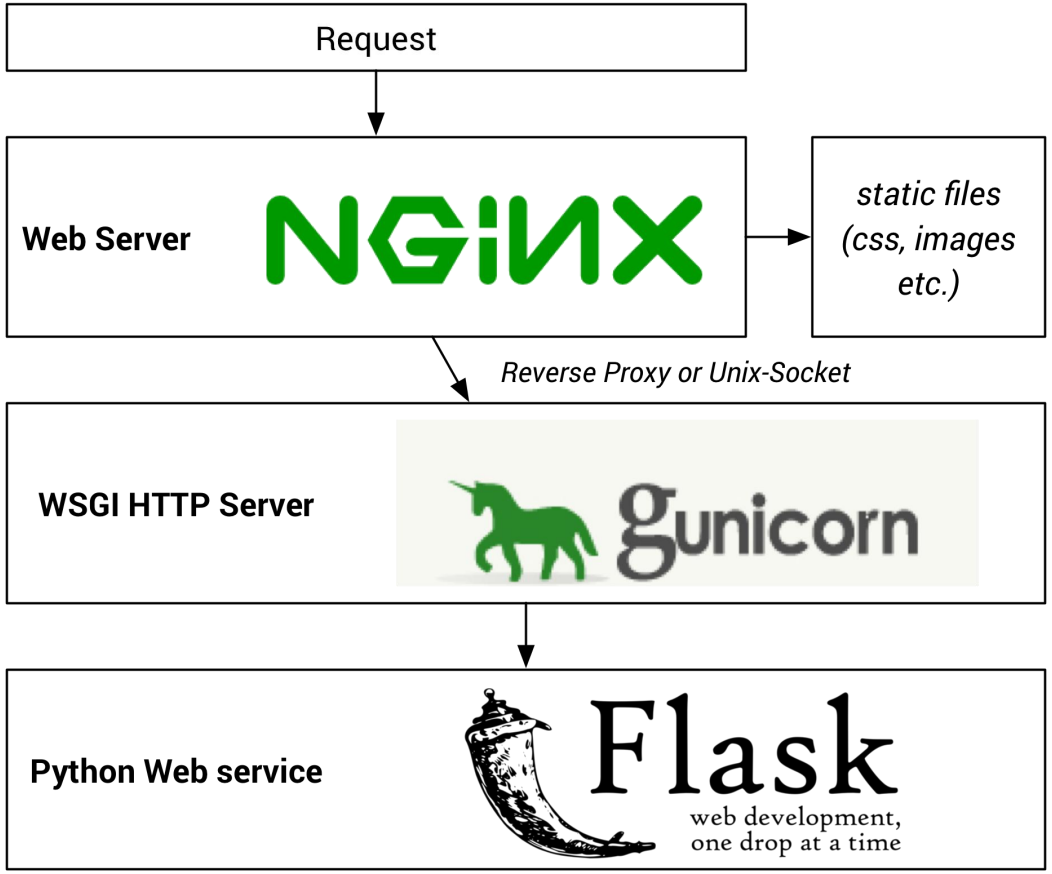
https://www.djangoproject.com/

WSGI HTTP Server



WSGI == Web Server Gateway Interface (PEP 3333)

Gunicorn



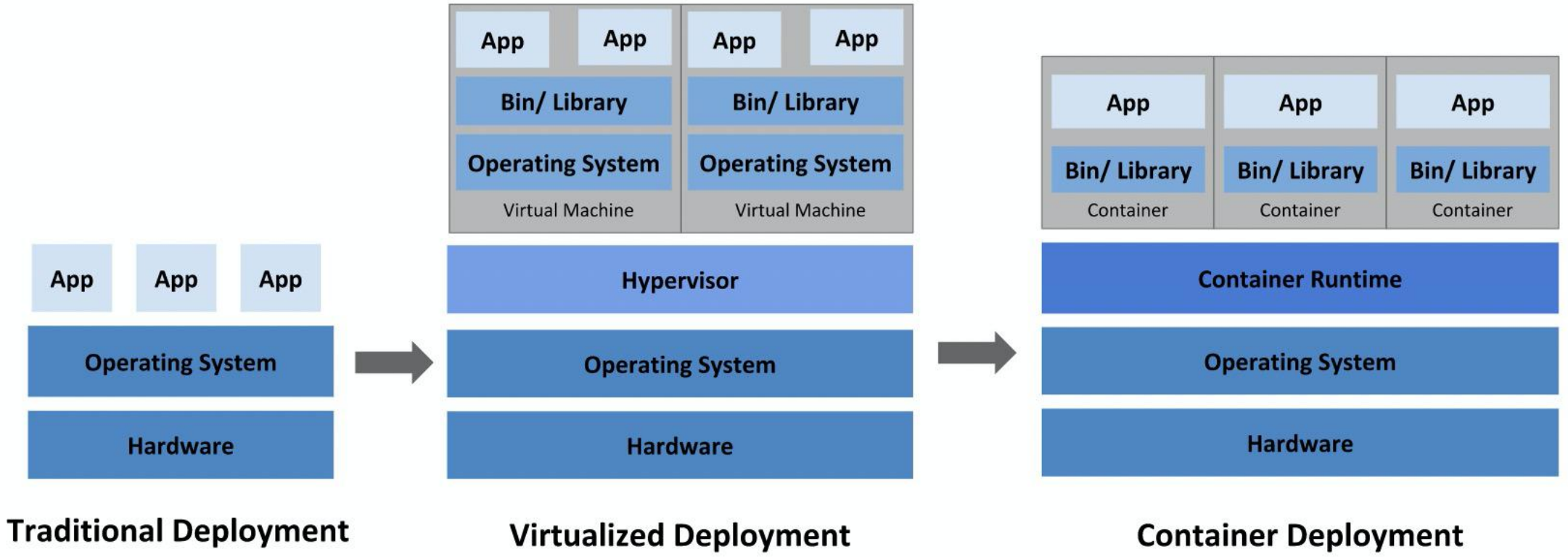
https://gunicorn.org/

Ngnix

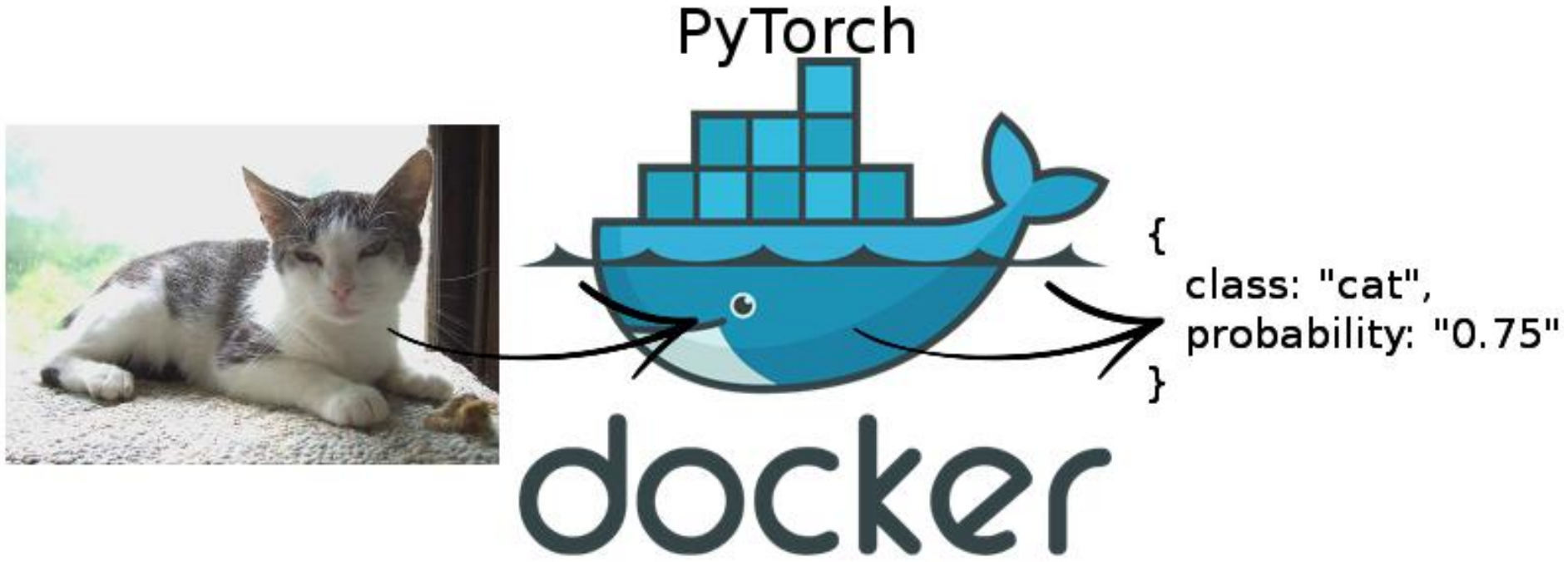
Improve the performance, reliability, and security of your applications

https://www.nginx.com/

Docker

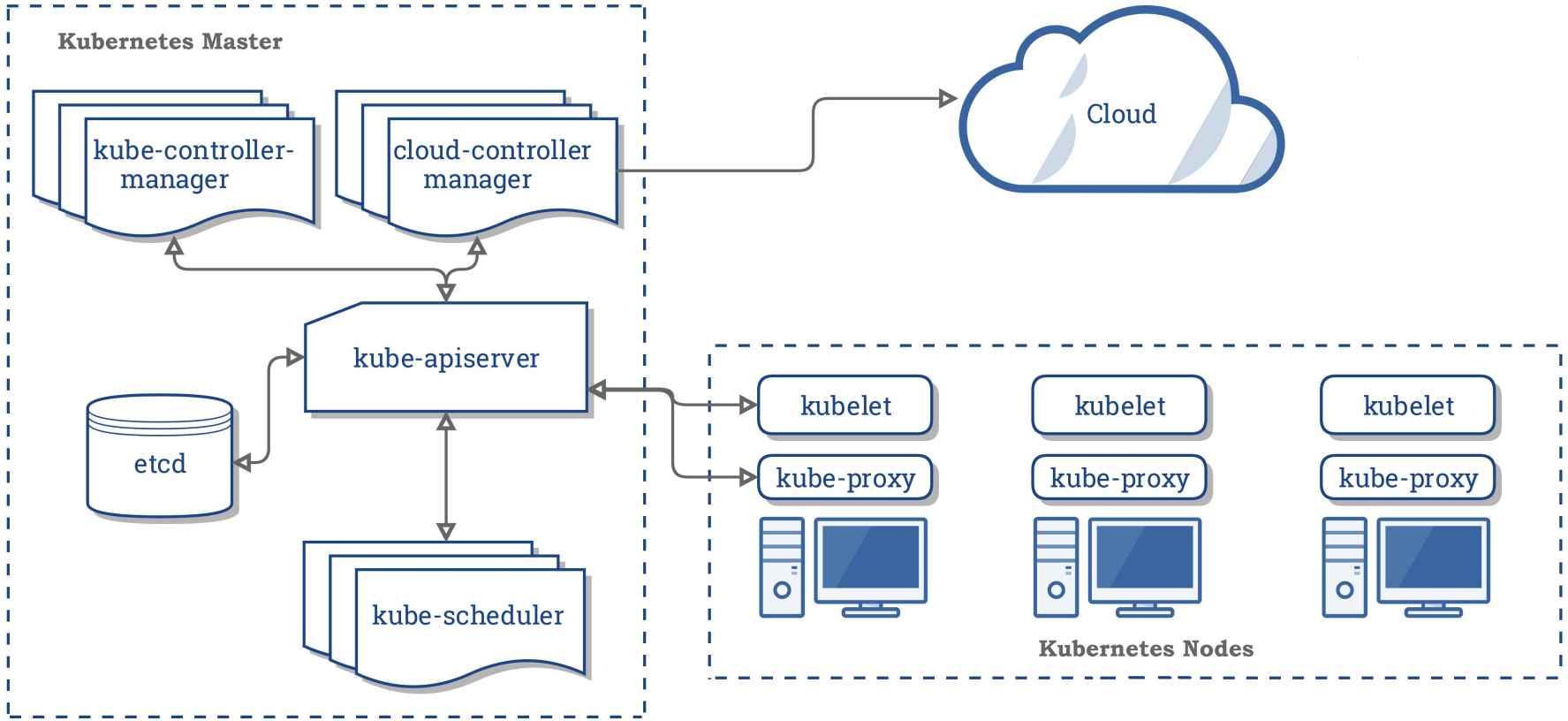


Docker



https://www.docker.com/

K8s



https://kubernetes.io/

Note

课程总结

- 权重初始化方法
- 模型部署方法

重难点

- 权重初始化不能为零的原因
- Kaiming初始化和Xavier初始化的区别
- Tensorflow Serving的部署流程
- Flask/Ngnix/Docker/K8s

课程作业

- 提交项目的训练结果

后厂理工学院

houchangtech.com

一所专注前沿互联网技术领域的创新实战大学