CNUTCon 2017 全球运维技术大会

Kubernetes 与 AI 相结合架构、落地解析（从 0 到 1）

赵慧智

技术总监（才云科技）

主办方 Geekbang 极客邦科技 InfoQ

Geekbang.
极客邦科技

极客搜索

全站干货，一键触达，只为技术

s.geekbang.org

扫描二维码立即体验

有没有一种搜索方式，能整合 InfoQ 中文站、极客邦科技旗下12大微信公众号矩阵的全部资源?
极客搜索，这款针对极客邦科技全站内容资源的轻量级搜索引擎，做到了!

扫描上方二维码，极客搜索!

EGO EXTRA GEEKS' ORGANIZATION NETWORKS

这里只有
技术领导者

EGO会员第二季招募季正式开启

报名时间：9月1日-9月15日

扫描添加E小欧，

邀您进入EGO会员预报名群

立即报名

E小欧

TABLE OF CONTENTS

Kubernetes 介绍

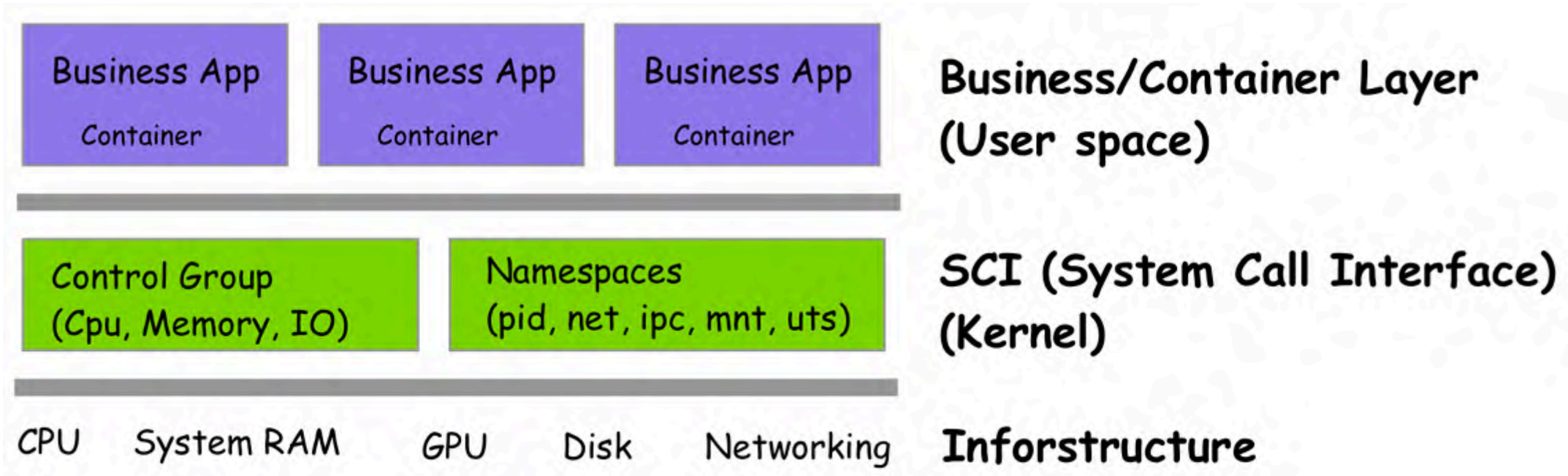
- 是一套集自动部署，弹性扩容，并且基于容器的集群管理工具。
  - 快速部署应用程序
  - 弹性负载均衡
  - 无缝升级应用
  - 硬件隔离

LXC (Linux Container) 介绍

在单一系统的内核层通过一套 API 在应用层提供硬件及软件环境隔离的 Linux 环境（containers）。在内核层，通过 cgroup 来提供硬件环境的隔离（例如 CPU，Memory，Block I/O，网络等等）和通过 namespace 来提供软件层面的隔离（例如 process tree，网络，user IDs 和挂载的文件系统）。

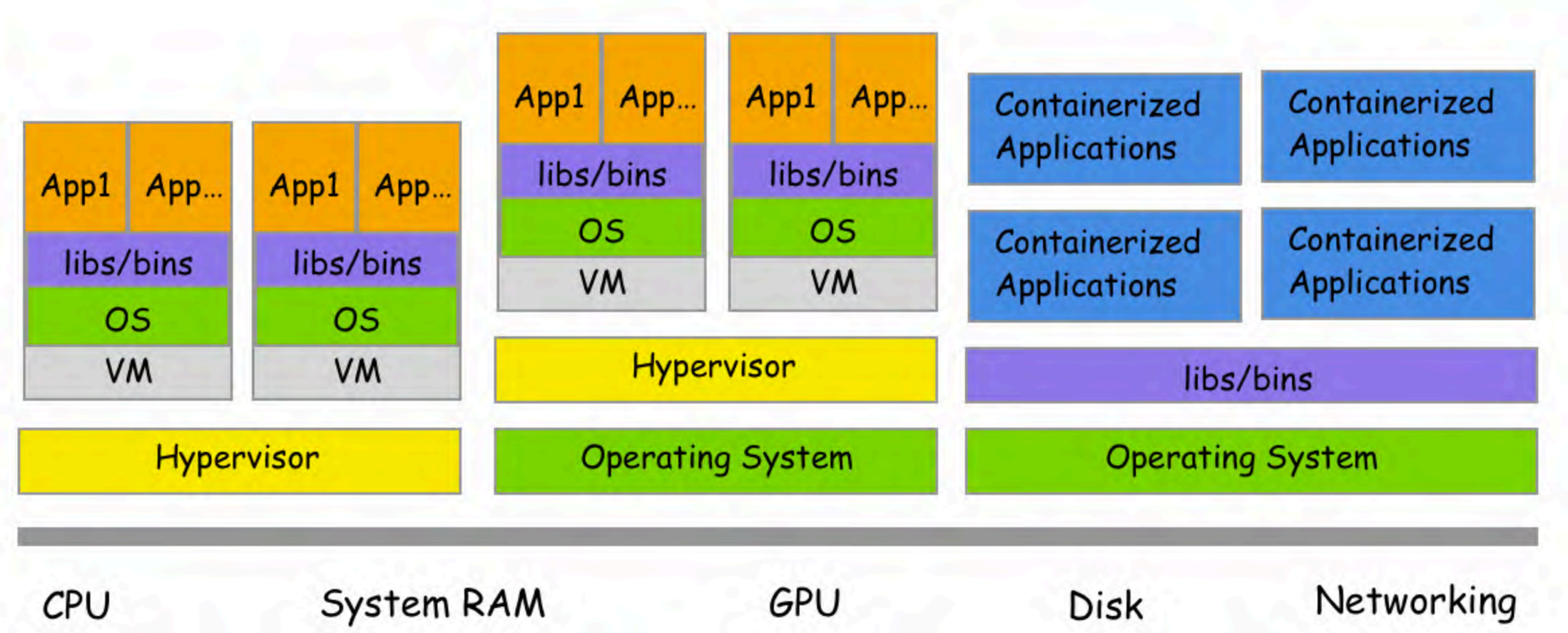
Container 框架

通过在 Kernel 层面提供的 API 来达到上层可以容器化应用程序。



Container VS VM (Virtual Machine)

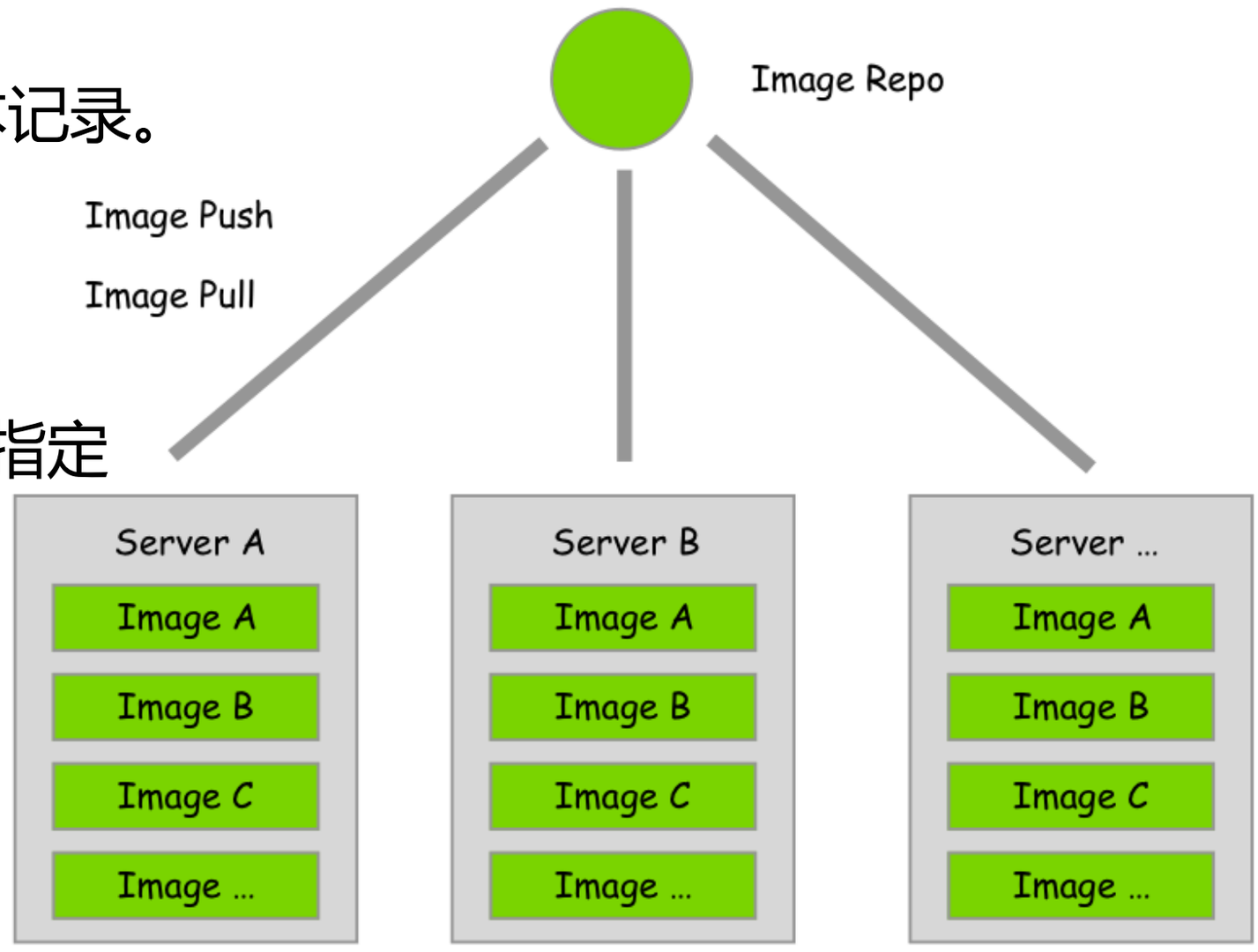
由于采用系统当前的内核，Container会启动加载更快，更轻量，并且更省资源。



Container Image

用来将需要容器化的应用程序及其环境进行打包后存储的镜像。

- 通常会有一个 Image 管理仓库来存储 Image。
- 同一个 Image 会有版本记录。
- 只包含软件环境的配置
- 硬件配置需要运行时去指定



OCI (Open Container Initiative)

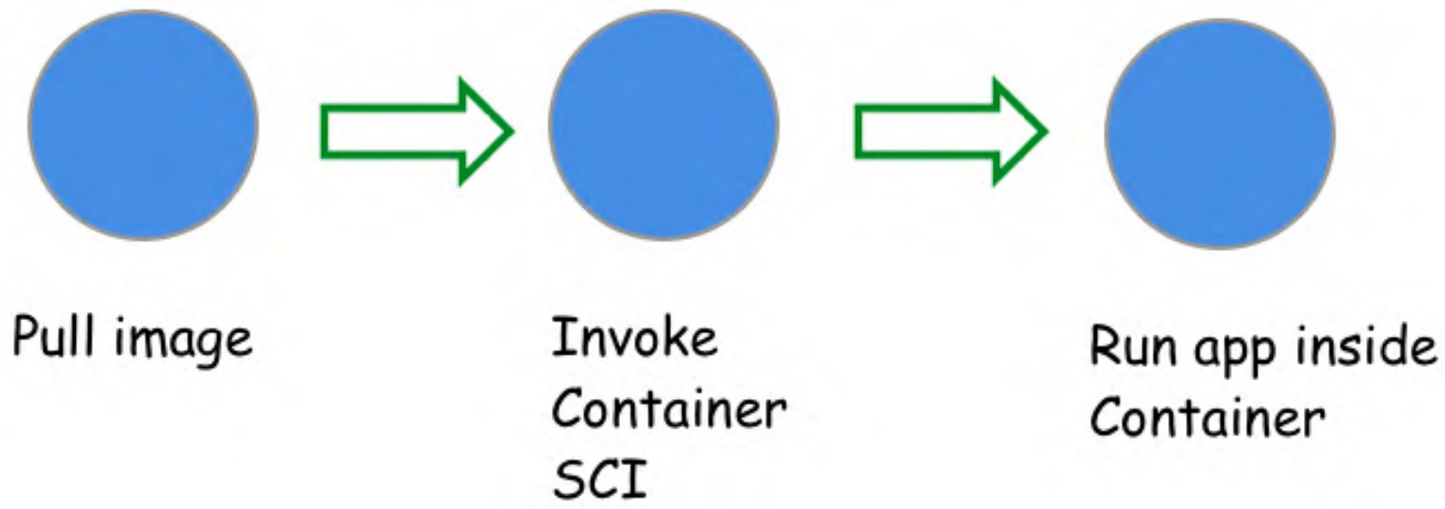
- From Linux Foundation
- 旨在为 Container 格式和运行时创建开放行业标准。
- 该组织于2015年6月22日由Docker，CoreOS等 Container 行业的领导人推出。
- OCI目前包含两个规范：
  - 运行时规范（runtime-spec）
  - Image 规范（image-spec）

Container 管理工具 (User Space)

Software	docker	rkt
Company	docker	CoreOS

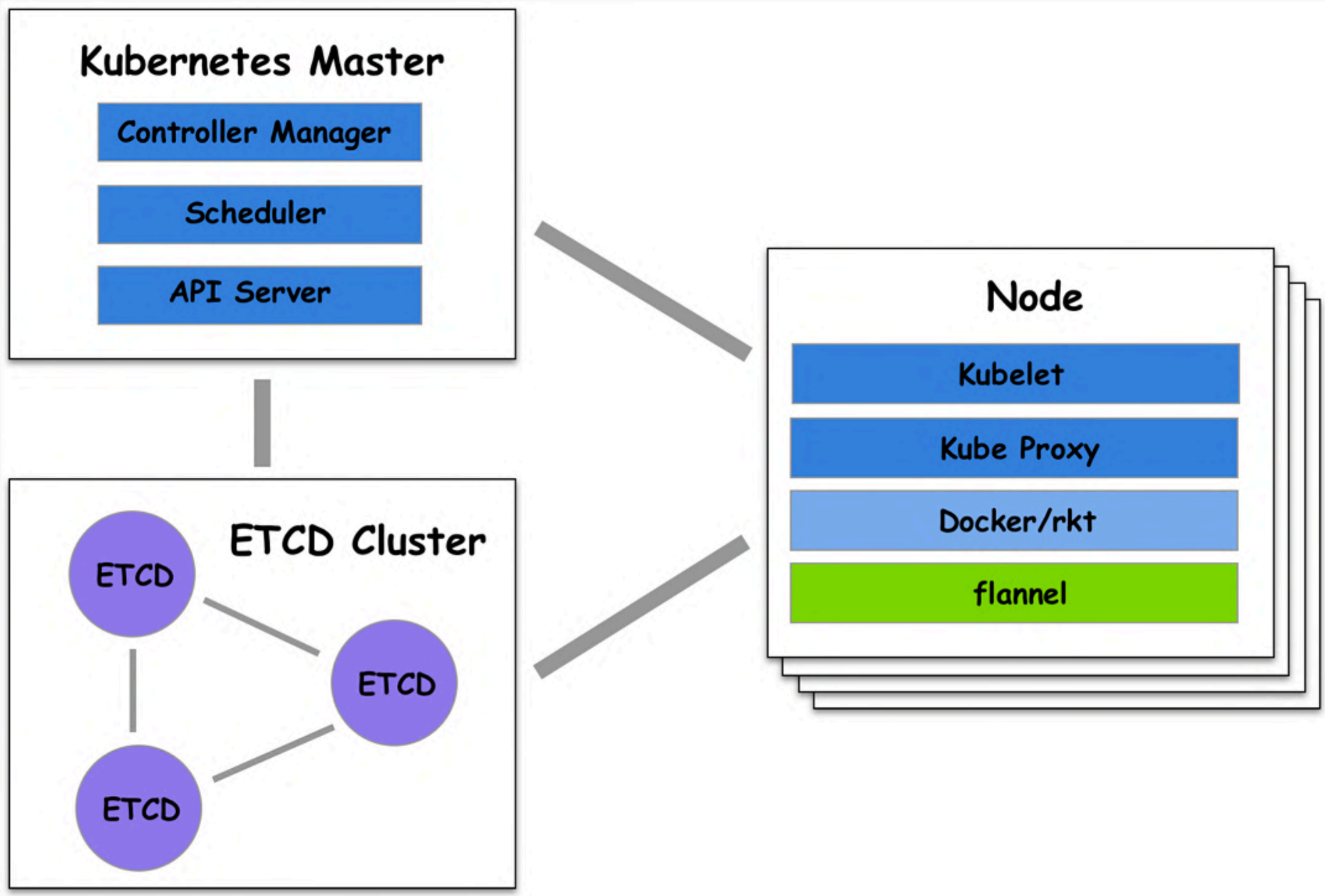
如何通过 Docker 启动 Container 并与硬件绑定

```
$ docker run -it --cpus=".5" ubuntu /bin/bash
```



官方文档参考：
https://docs.docker.com/engine/admin/resource_constraints/

Kubernetes 架构



Kubernetes 基本概念 - Node

一个 Node 是一个 Kubernetes 众多集群中的一个工作节点，一个 Node 可以是一台物理服务器或者虚拟机。一个 Node 的作用是用来运行 pods 的环境，并且被 Master 组件所管理。一个 Node 包含如下组件：

- Docker/rkt
- kubelet
- kube-proxy

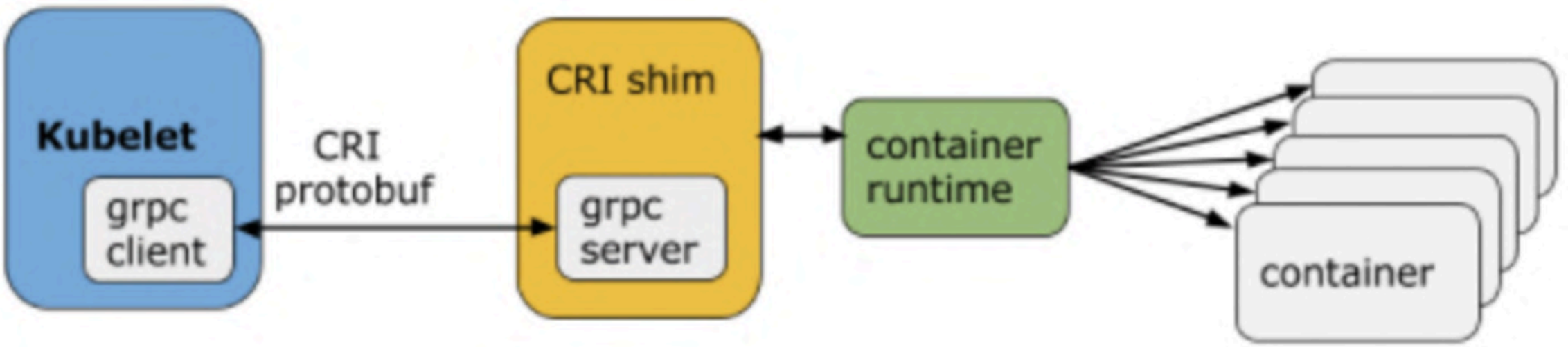
Kubernetes 基本概念 - Master

Master 节点是 Kubernetes 环境中的管理节点，负责整个集群的资源管理 / 分配，容器编排。一个 Master 节点包含如下组件：

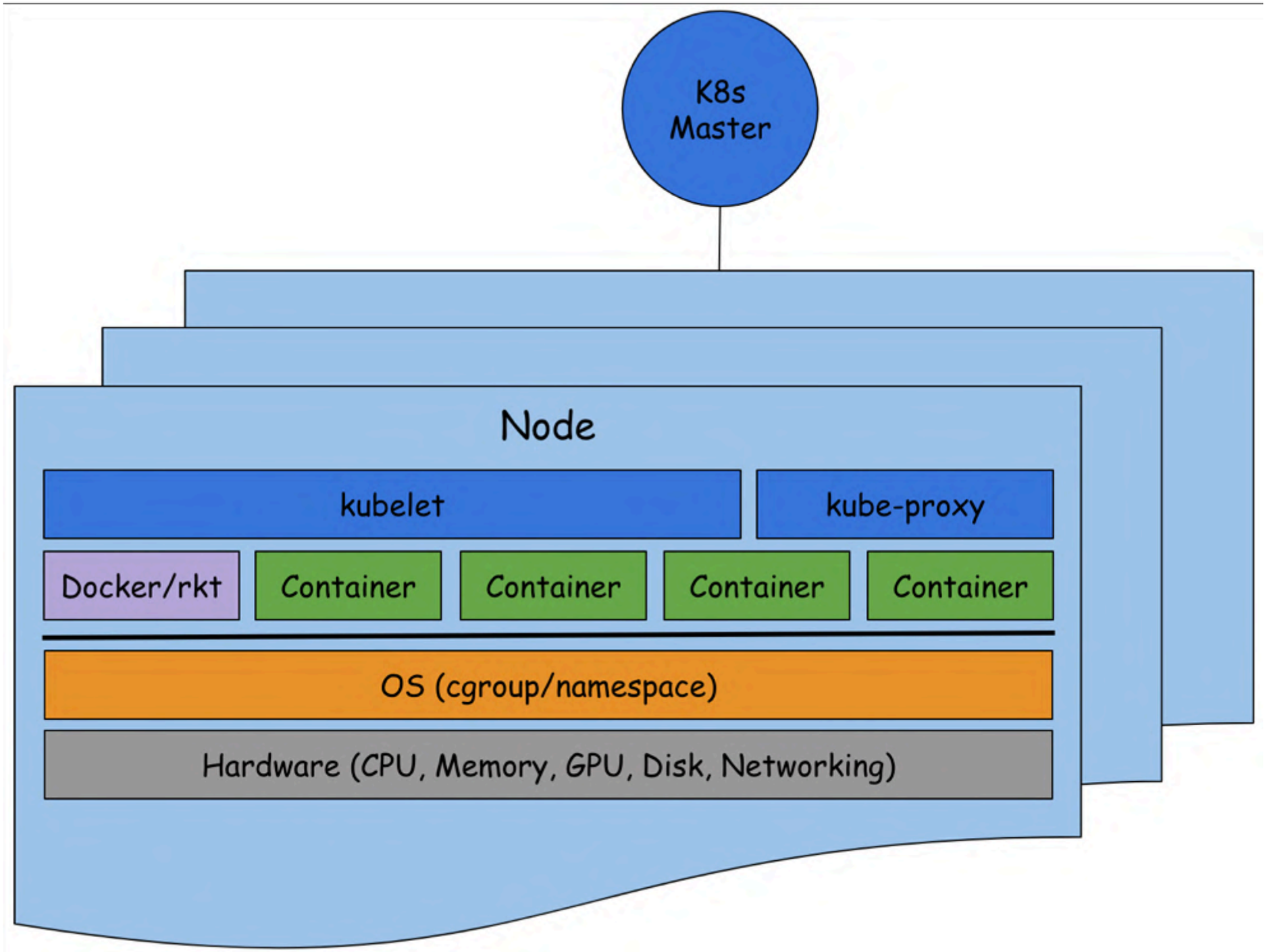
- kube-controller-manager
- kube-apiserver
- kube-scheduler

Kubernetes 与 Container 通信框架

Kubernetes 通过 CRI (Container Runtime Interface) 层将 Kuernetes 与具体的 Container 管理工具隔离，并且可以进行 Container 的操作。

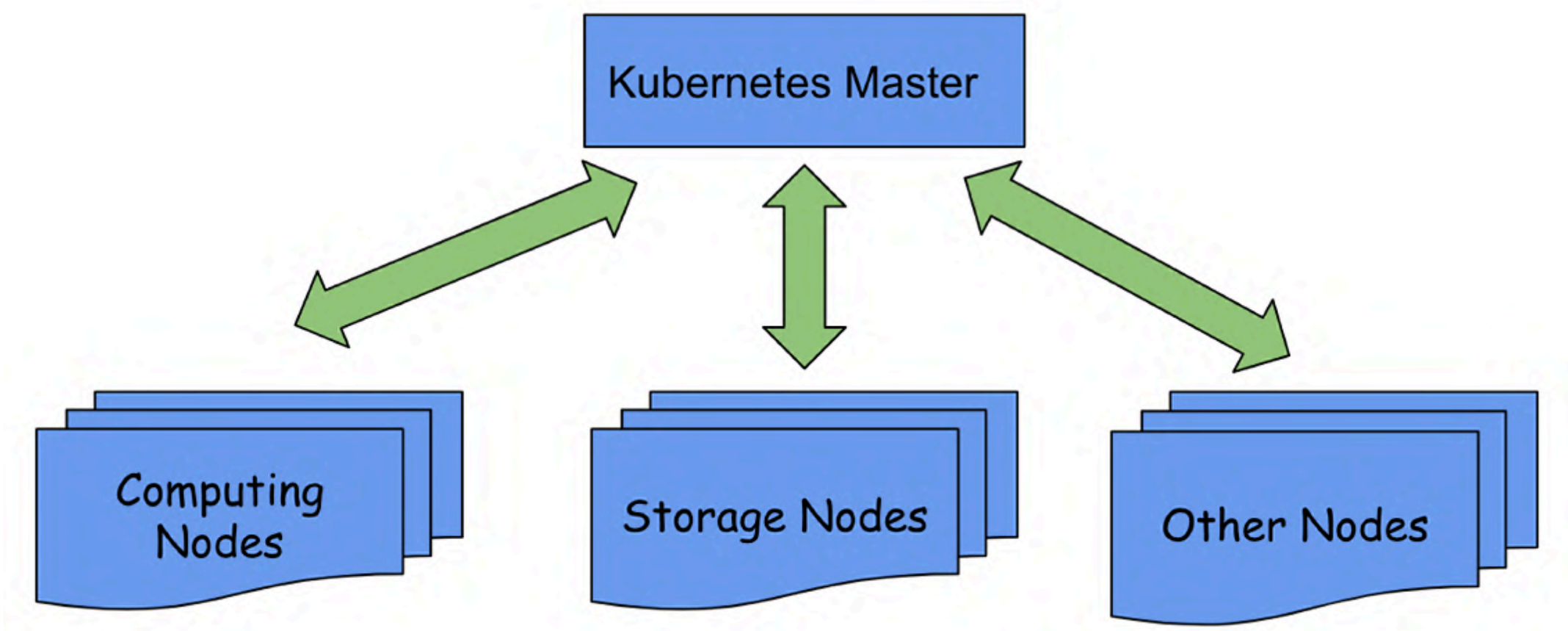


在 Node 上的层次关系



通过 Label 的方式将 Node 分类

在企业环境中 Node 很有可能需要进行不同类别的区分，而每一类 Node 上的环境都有可能会不同。



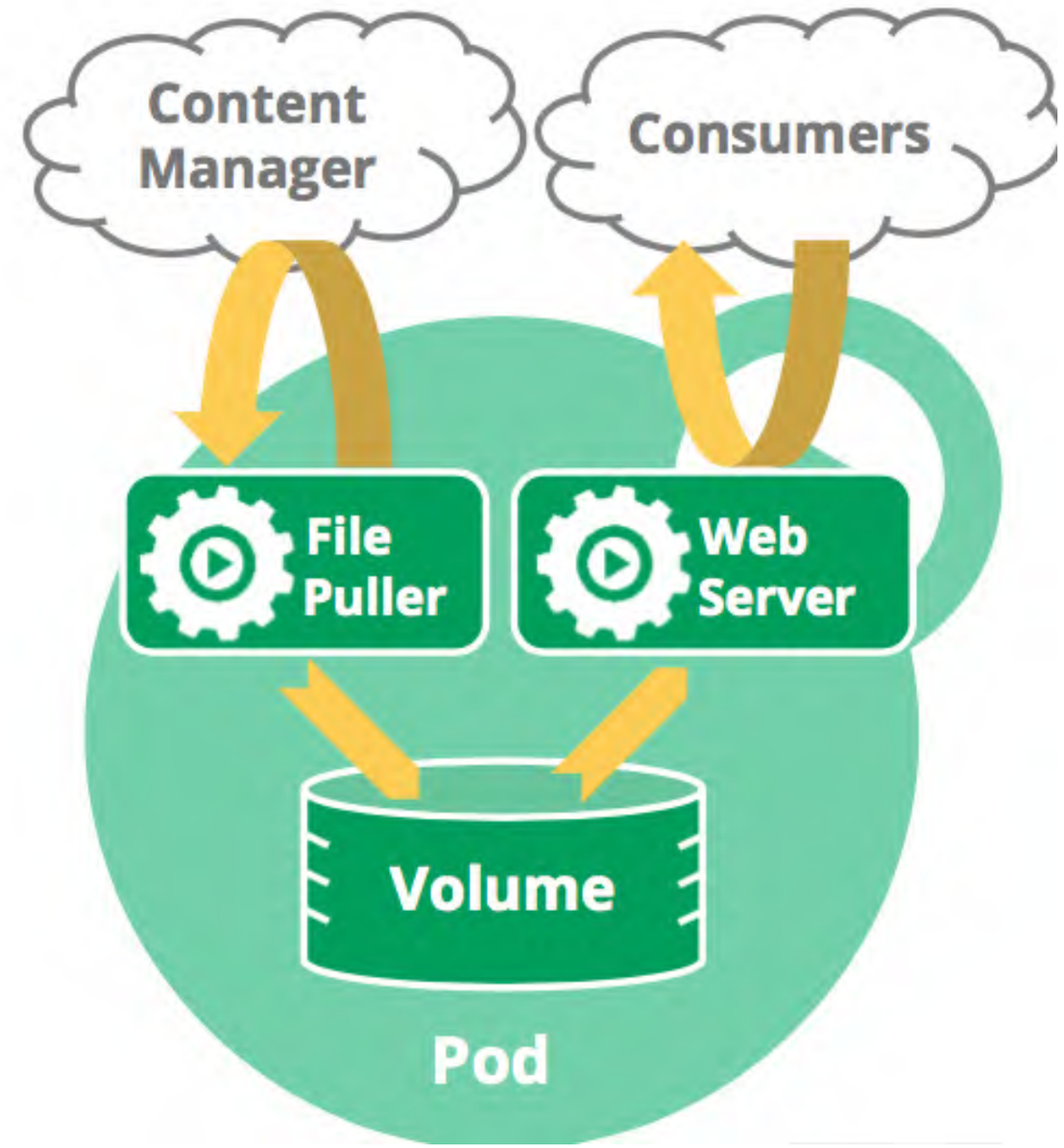
关键点回顾

- Kubernetes 是什么：容器编排工具
- Container 与 VM 比较的优势：更轻量化，更快，更易部署
- Container Image 的管理
- Container 运行与 Image 的标准：OCI
- Kubernetes 的基本架构：Master 与 Node
- Node 上的软件架构：Kubernetes 与 container 通信，Node 的分类

TABLE OF CONTENTS

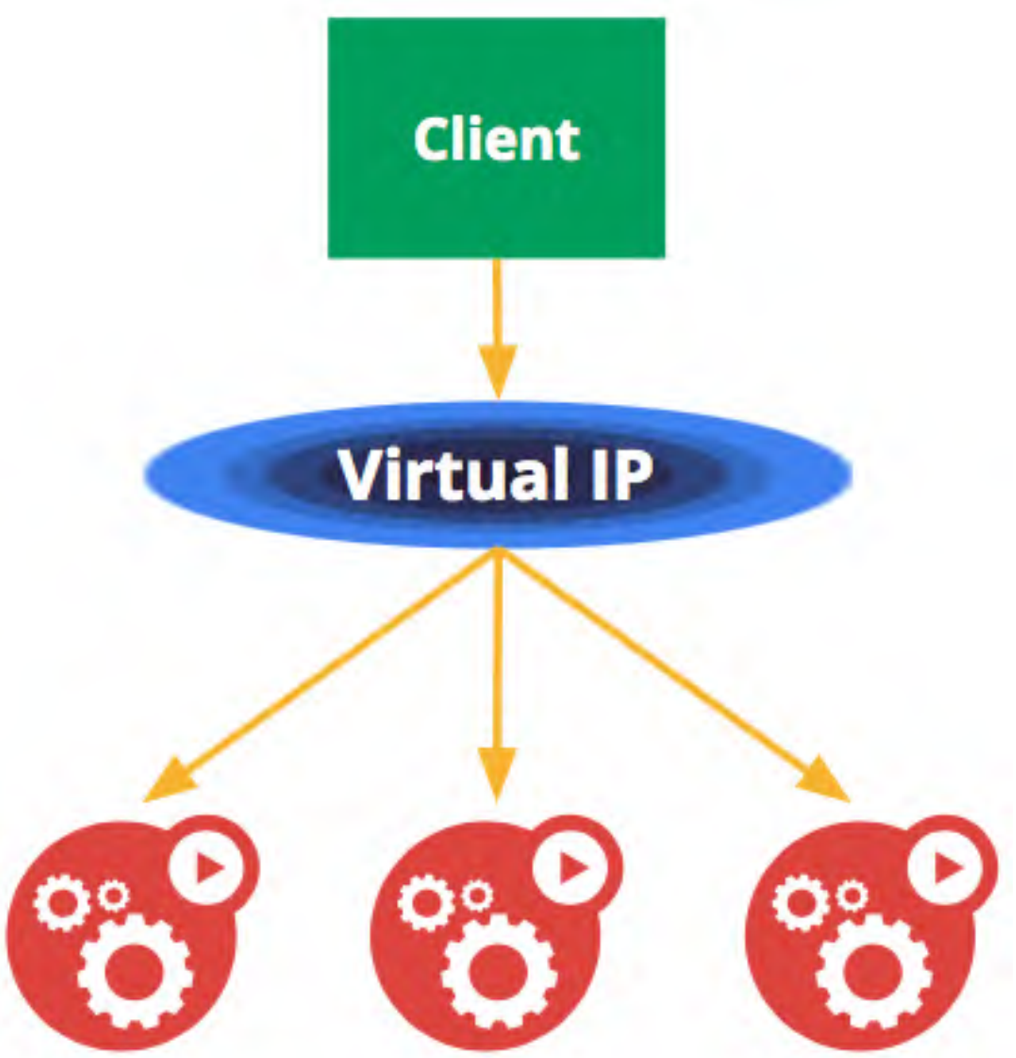
Kubernetes 基本概念 - Pod

- Pod 是 Kubernetes 中最小的资源
- 一个 Pod 包含一个或多个 Container
- Pod 内的 Containers 可以共享网络和存储



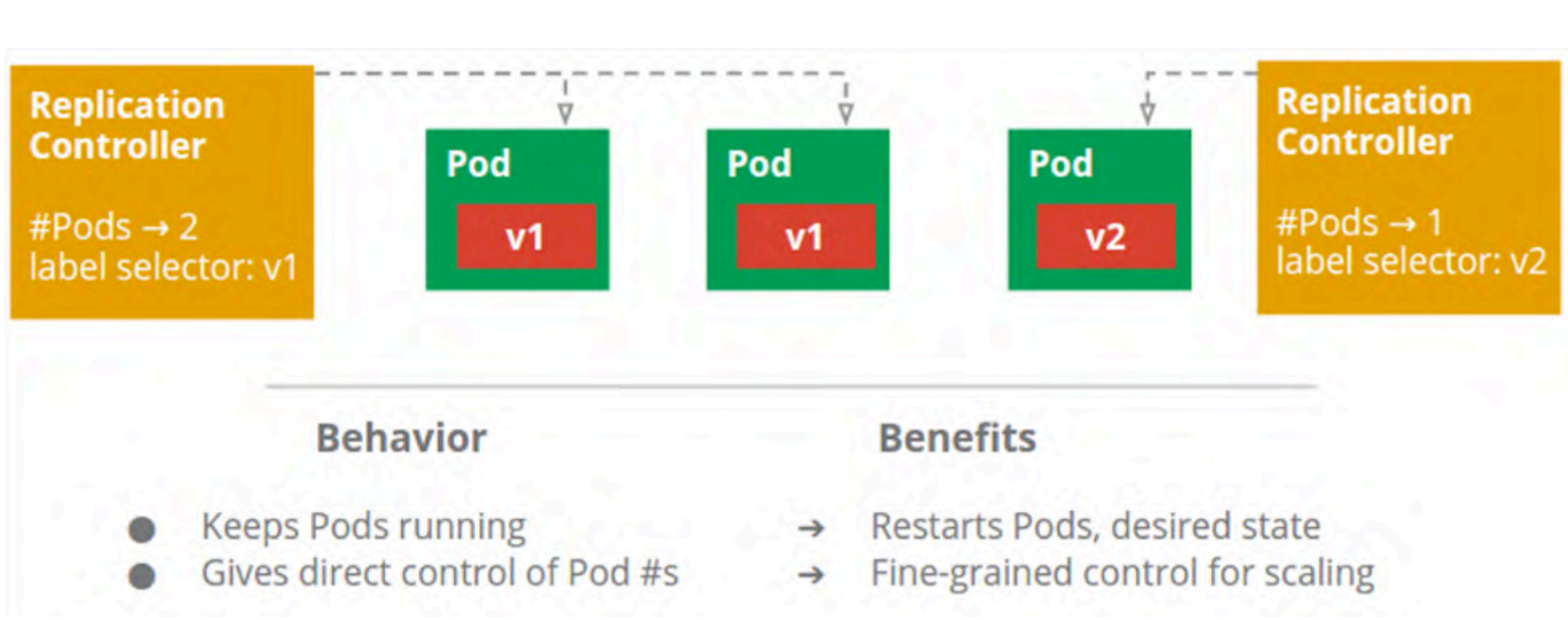
Kubernetes 基本概念 - Service

- Service 是一组 Pod 协同工作。
- 有时也被称为微服务。
- 可以给 Service 添加标签（label）来标识其业务属性。
- 通过 kube-proxy 可以得到固定的 virtual IP 和端口。



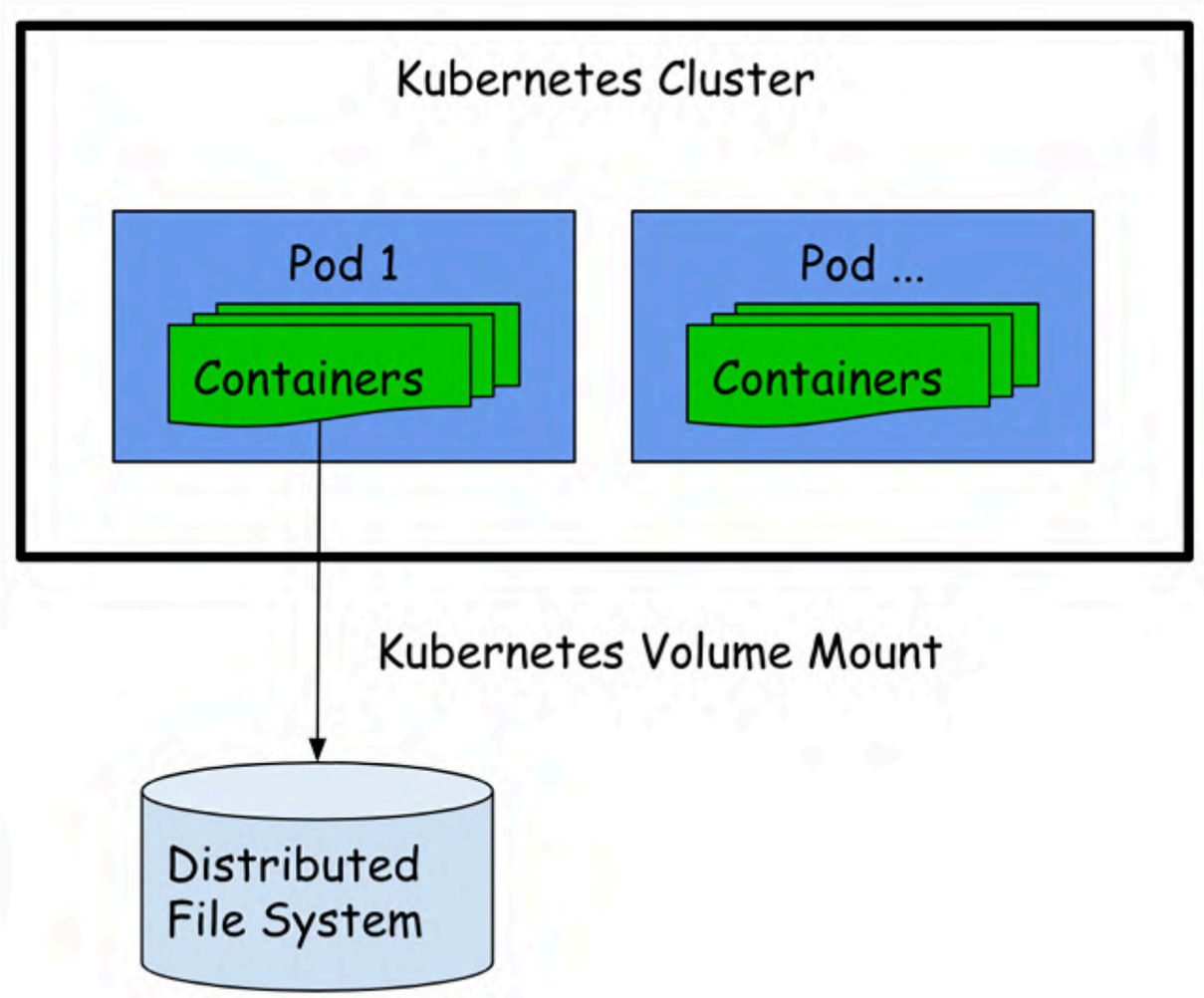
Kubernetes 基本概念 - Controller

- 一个 controller 可以创建和管理多个 pod
- 可以决定同时可以多少个相同的pod在同时运行
- 维护pod的状态



Kubernetes 基本概念 – Storage（ volume ）

- 达到非易失存储的方式
- 可以视作一个目录
- 将存储挂在到 Pod 及其内部 containers 的一种方式。



快速部署一个应用 - kubectl

- kubectl 是一个命令行操作 Kubernetes 的工具，可以在 Kubernetes 平台上进行容器编排。

```
// Create a service using the definition in example-service.yaml.
$ kubectl create -f example-service.yaml
```

```
// List all pods in plain-text output format.
$ kubectl get pods
```

```
// Display the details of the node with name <node-name>.
$ kubectl describe nodes <node-name>
```

```
// Return a snapshot of the logs from pod <pod-name>.
$ kubectl logs <pod-name>
```

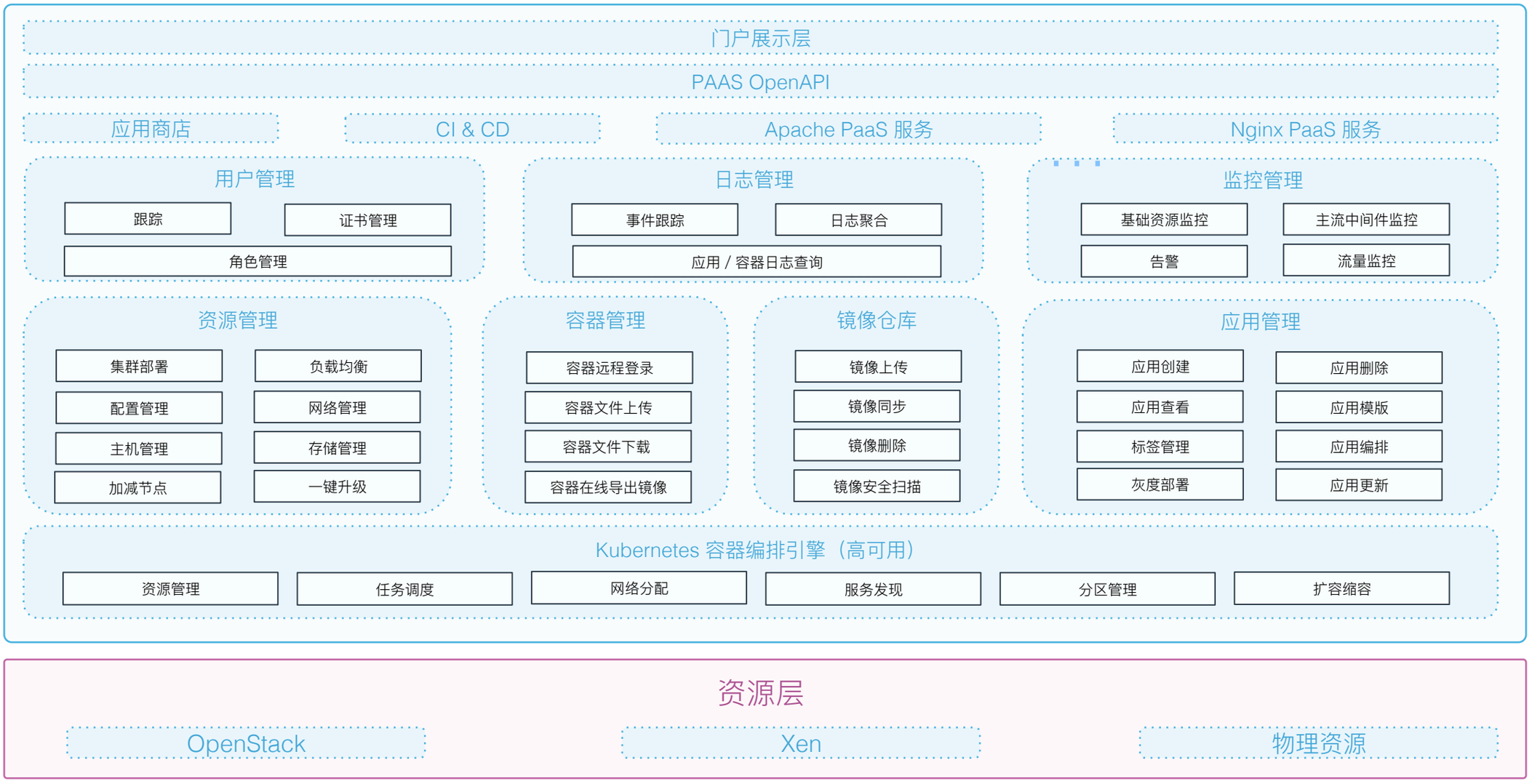
关键点回顾

- Pod
- Service
- Controller
- Volume
- Kubectl

TABLE OF **CONTENTS**

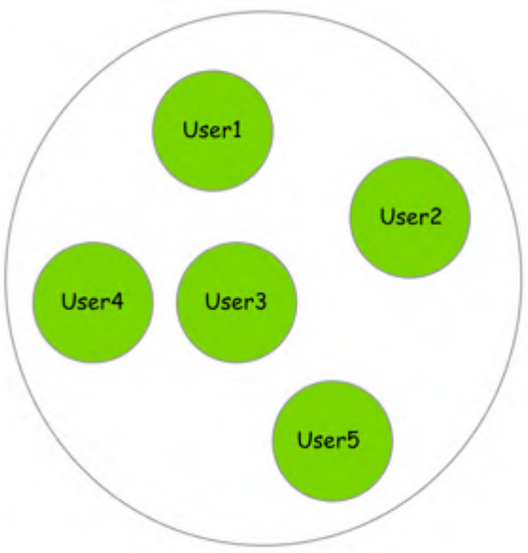
Kubernetes 企业级架构

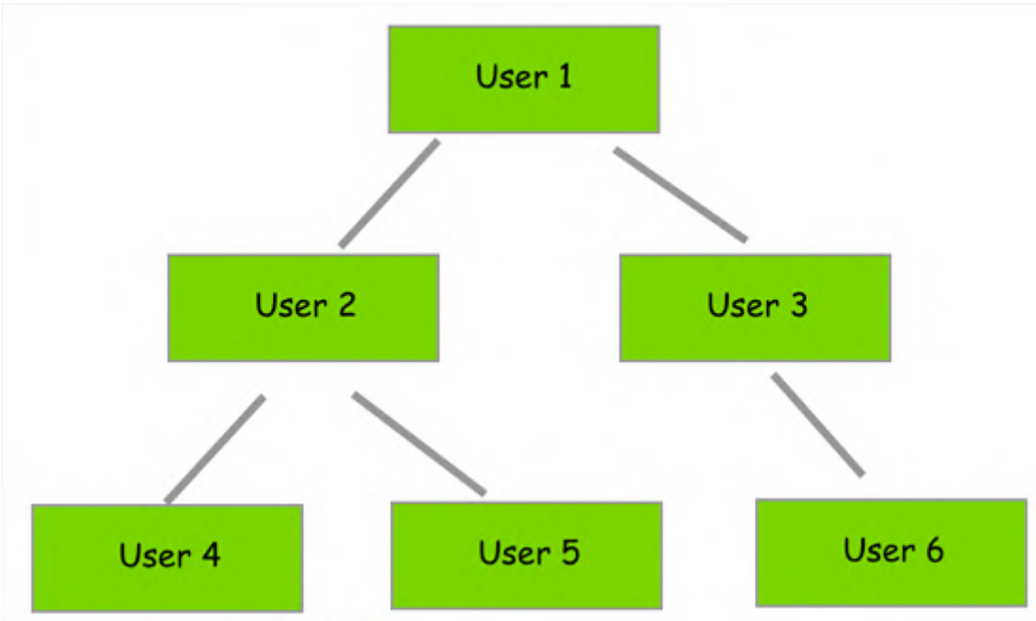
在企业级架构中，Kubernetes 更多是作为一个核心组件进行部署。在这个核心组件的外围，我们还有其他的功能需要提供。



用户管理

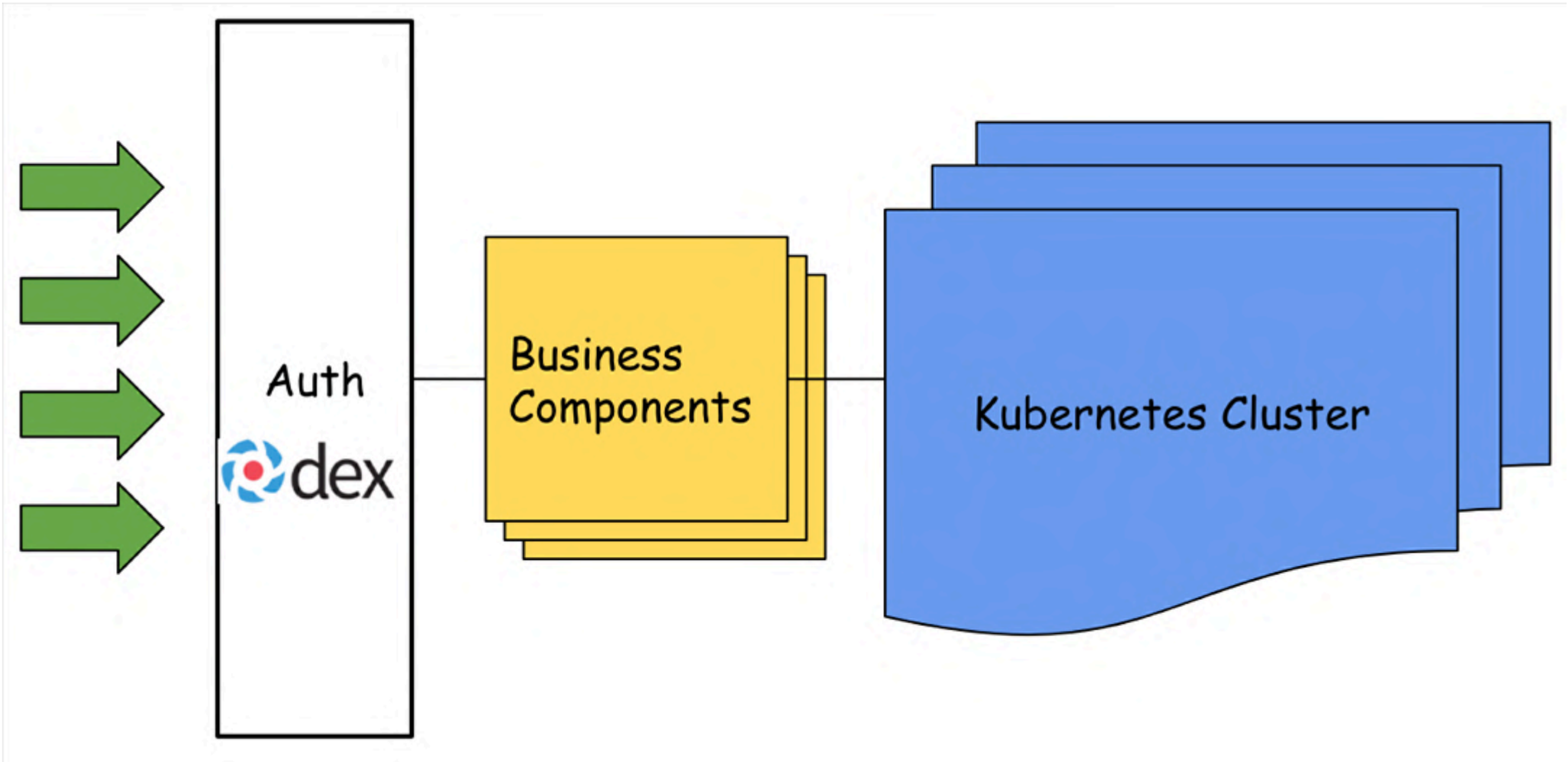
- 按资源组和层级用户的区分





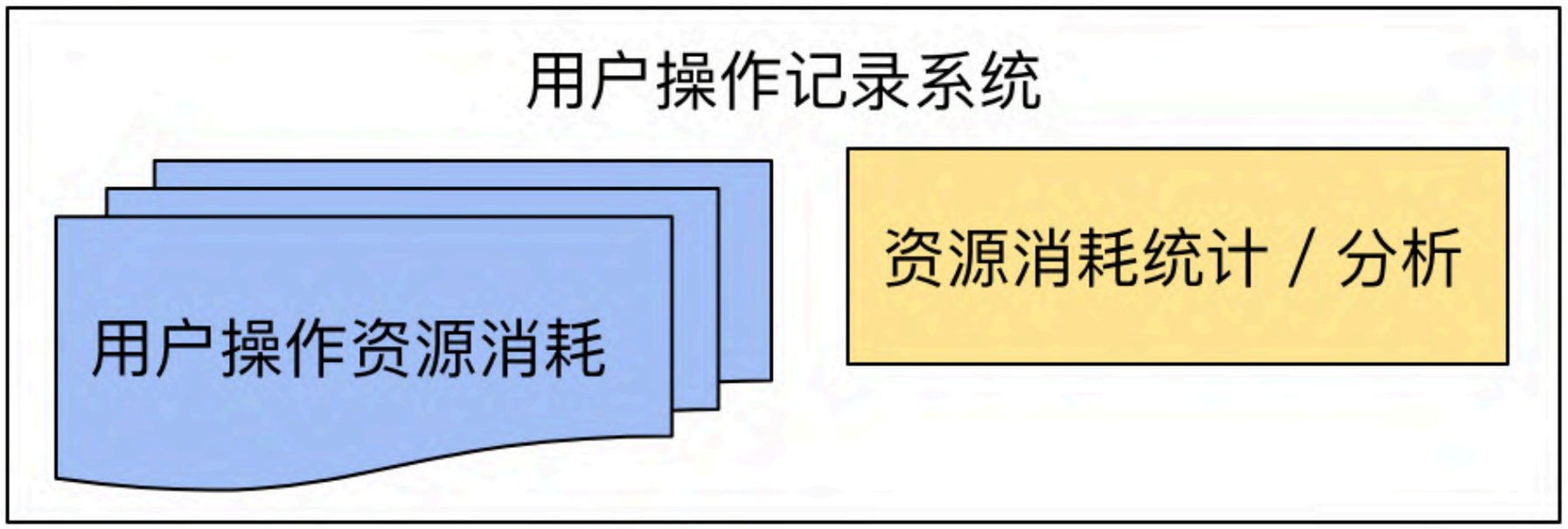
用户访问权限管理

根据用户功能组进行访问验证



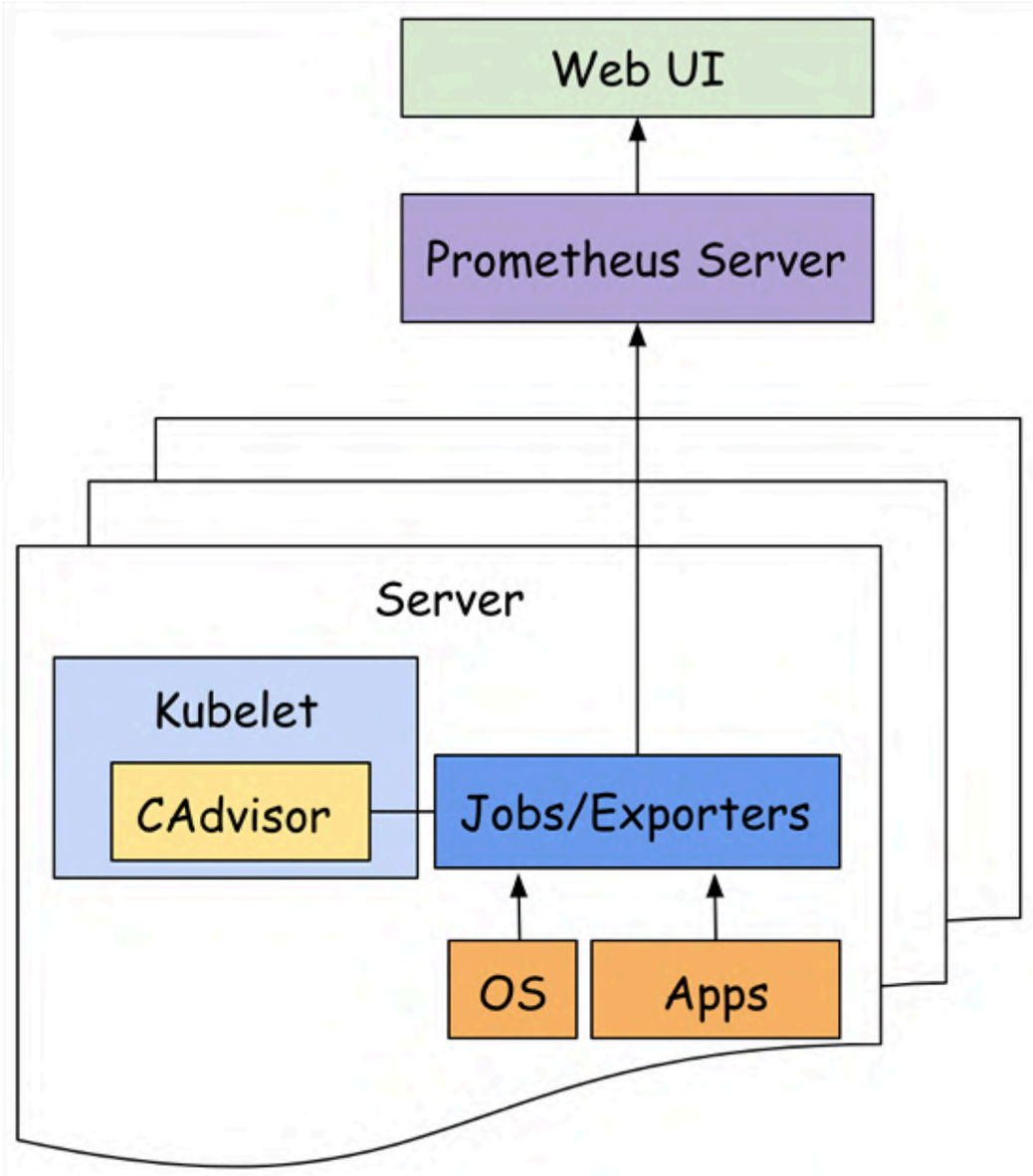
用户操作日志

- 对用户的操作进行记录
- 对用户的资源使用状况进行统计
- 对多用户的资源使用状况进行统计



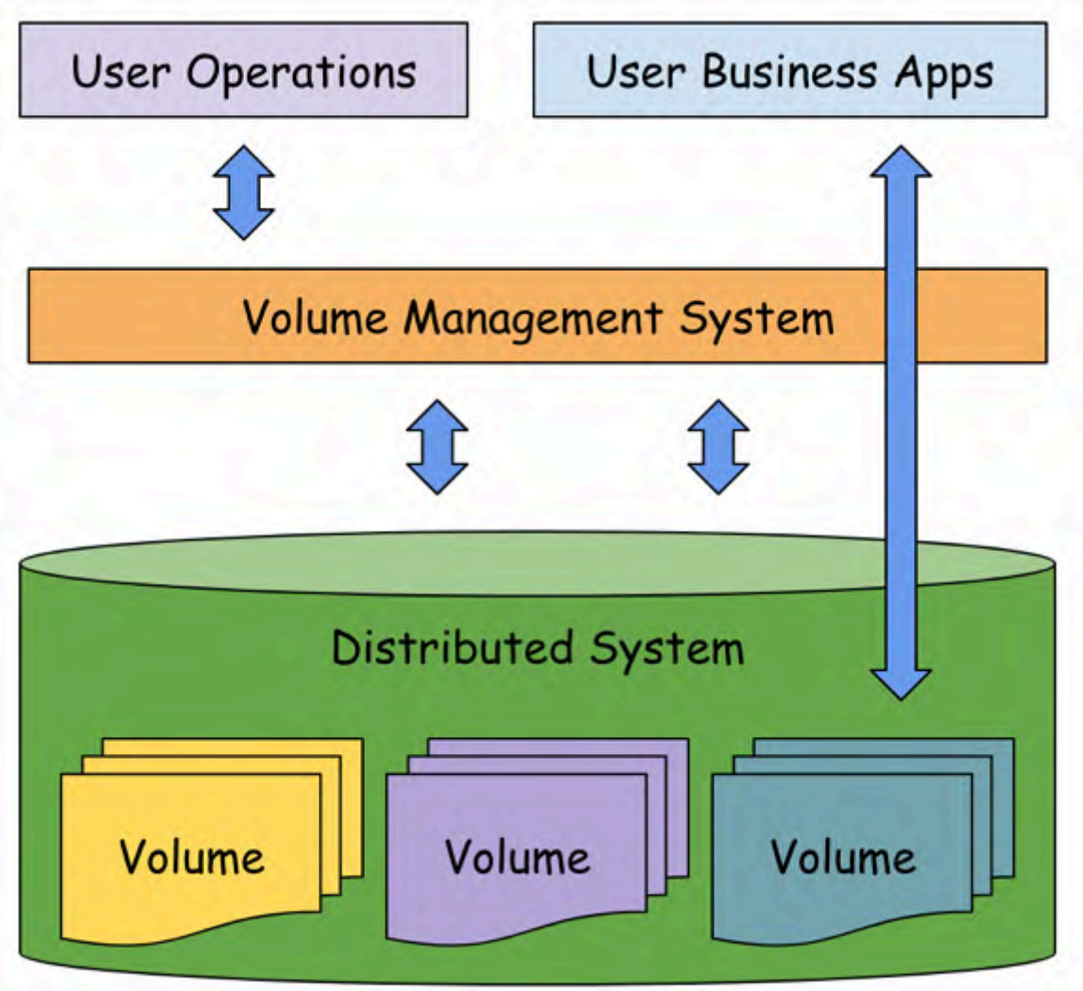
监控

- 实时的集群及 Containers 信息监控（包括CPU，Memory，BIO，Networking）
- 监控预警的阀值设置
- 收集监控日志



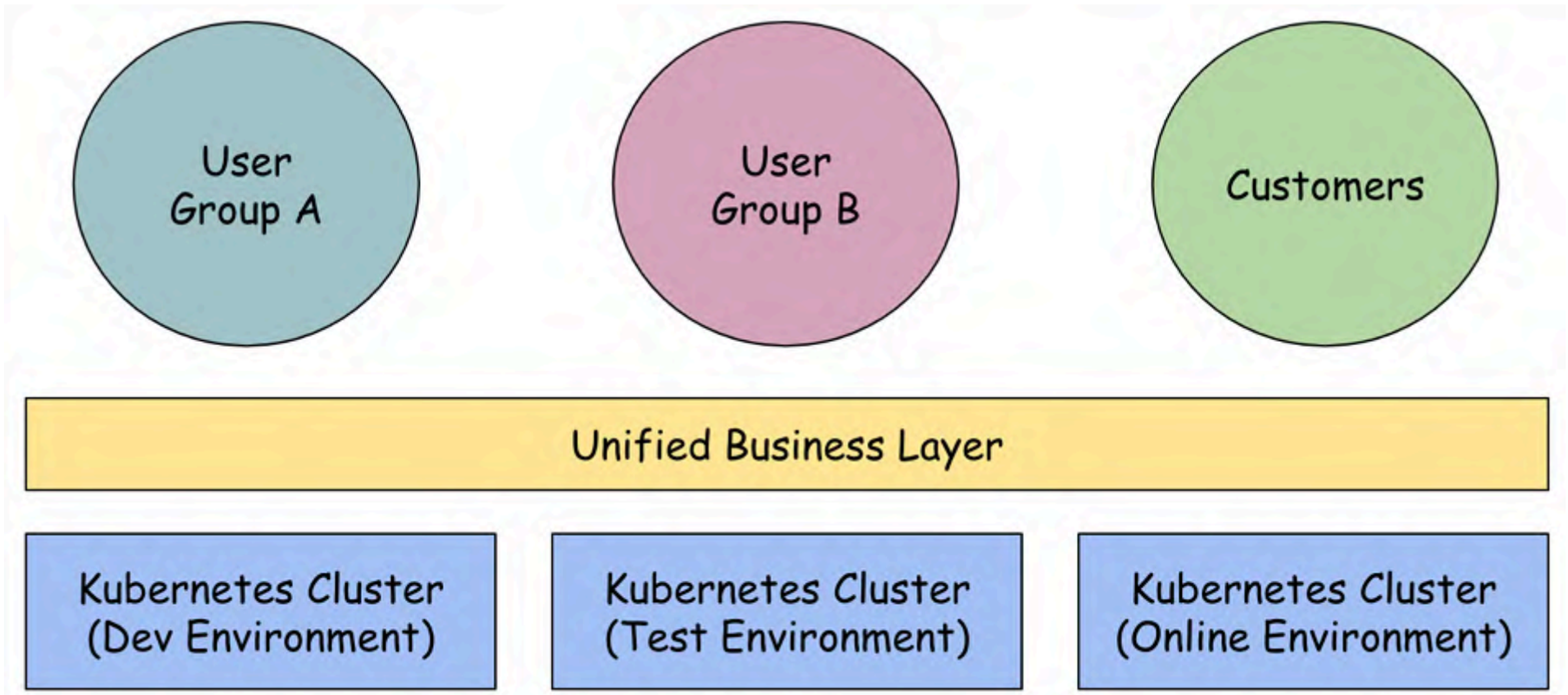
存储管理

- 对于所有 Kubernetes Volume 按照业务类型提供统一的管理
- 用户在进行Volume操作的时候根据业务进行中间层处理，包括但不限于访问权限，大小申请，读写操作等

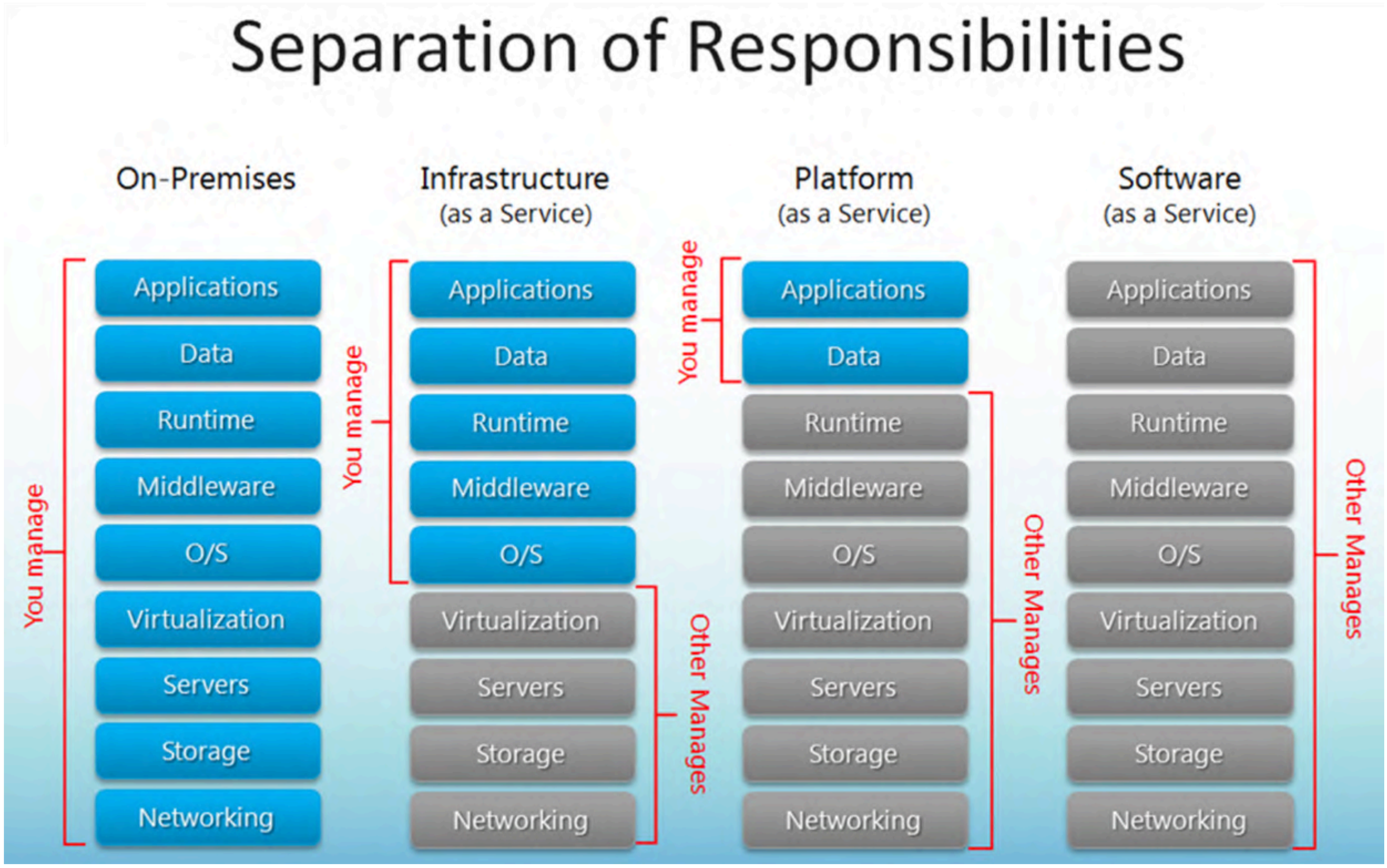


多集群管理

- 根据用户不同的角色进行集群不同的环境选择
- 在业务层面让用户感知为统一的环境



云提供服务形式


Separation of Responsibilities
On-Premises
Infrastructure (as a Service)
Platform (as a Service)
Software (as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You manage
Other Manages

关键点回顾

- 企业级架构
- 用户管理
- 用户访问权限管理
- 用户操作日志
- 硬件监控
- 存储管理
- 多集群管理
- 根据客户需求不同进行不同层次的封装

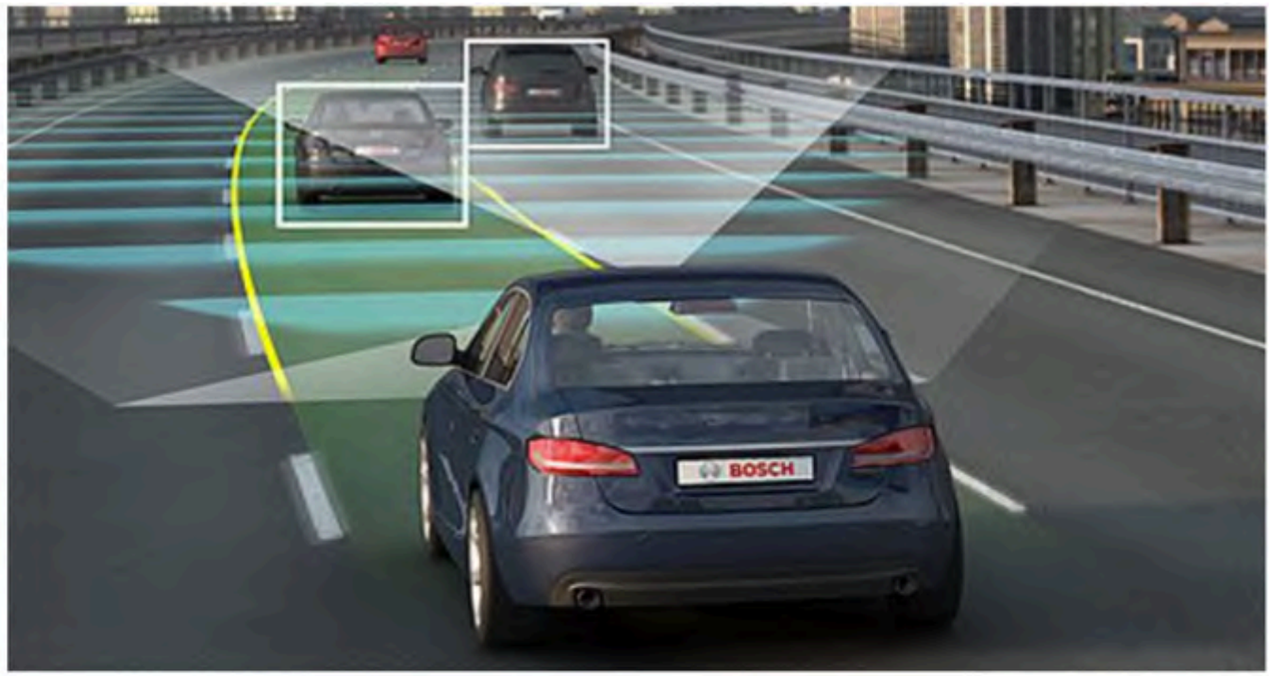
TABLE OF **CONTENTS**

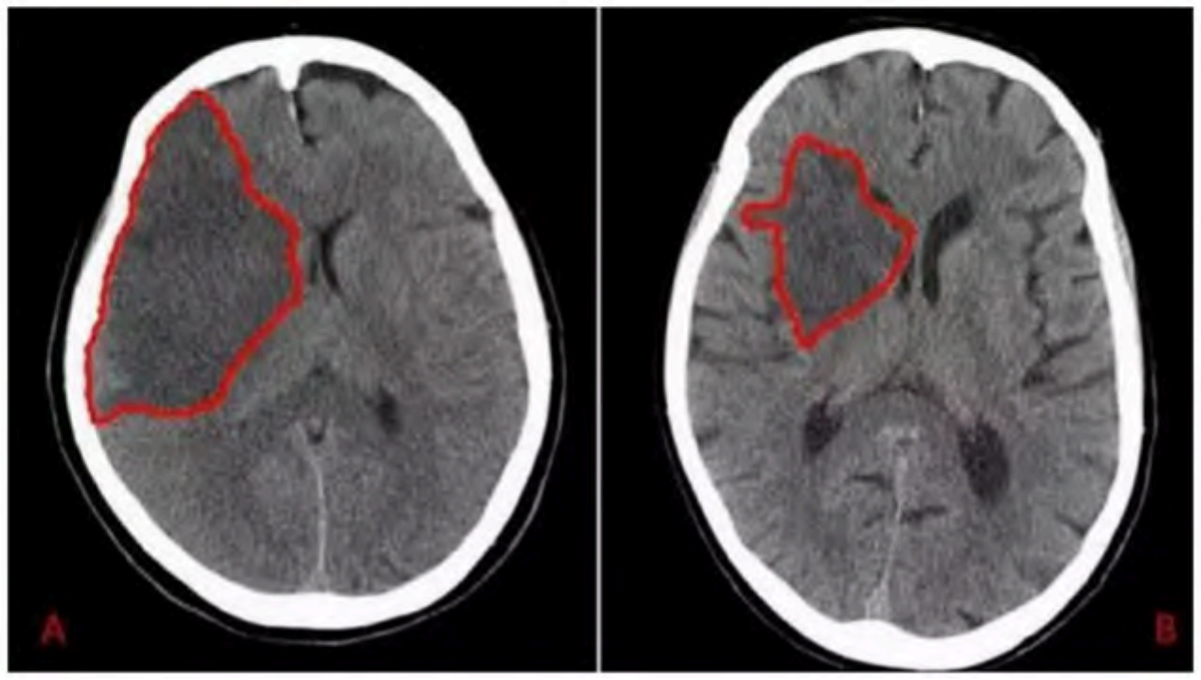
AI (Artificial Intelligence)

人工智能（英语：Artificial Intelligence, AI）亦称机器智能，是指由人工制造出来的系统所表现出来的智能。通常人工智能是指通过普通电脑实现的智能。同时也指研究这样的智能系统是否能够实现，以及如何实现的科学领域。

------ wiki

AI 具体应用场景



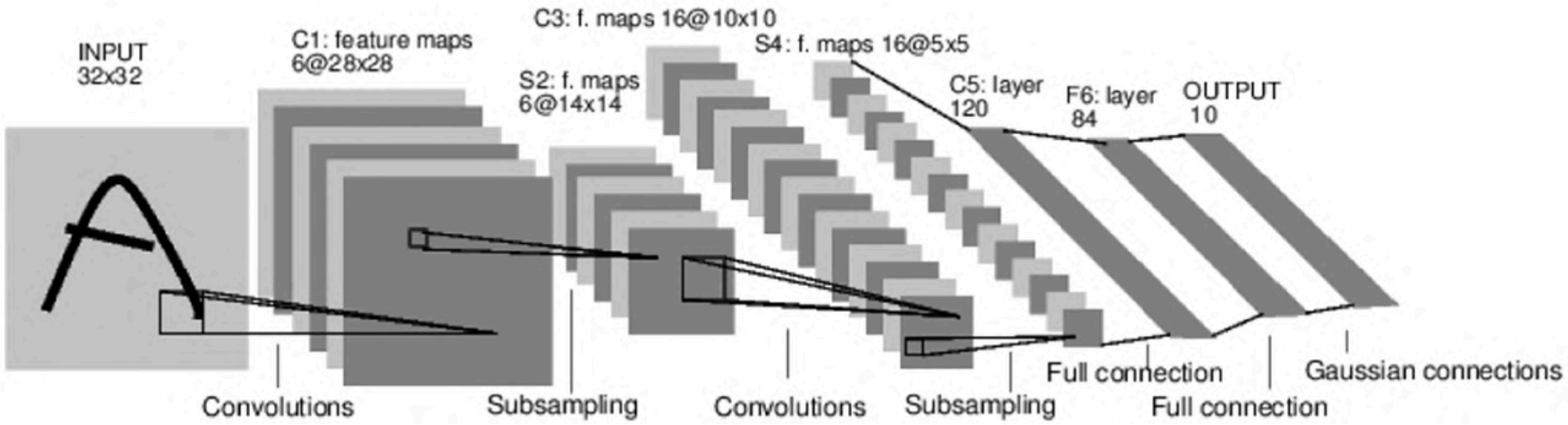


机器学习 vs 深度学习

- 机器学习是指通过算法，对数据进行分析，最终得出判断。
- 深度学习是指通过人工神经网络，通过对数据进行分析学习，最终得出判断。
- 机器学习最难的地方在于特征的提取，而深度学习认为特征提取是可以通过人工神经网络学习而得出结论的。深度学习在非结构化数据方面有很大的优势。

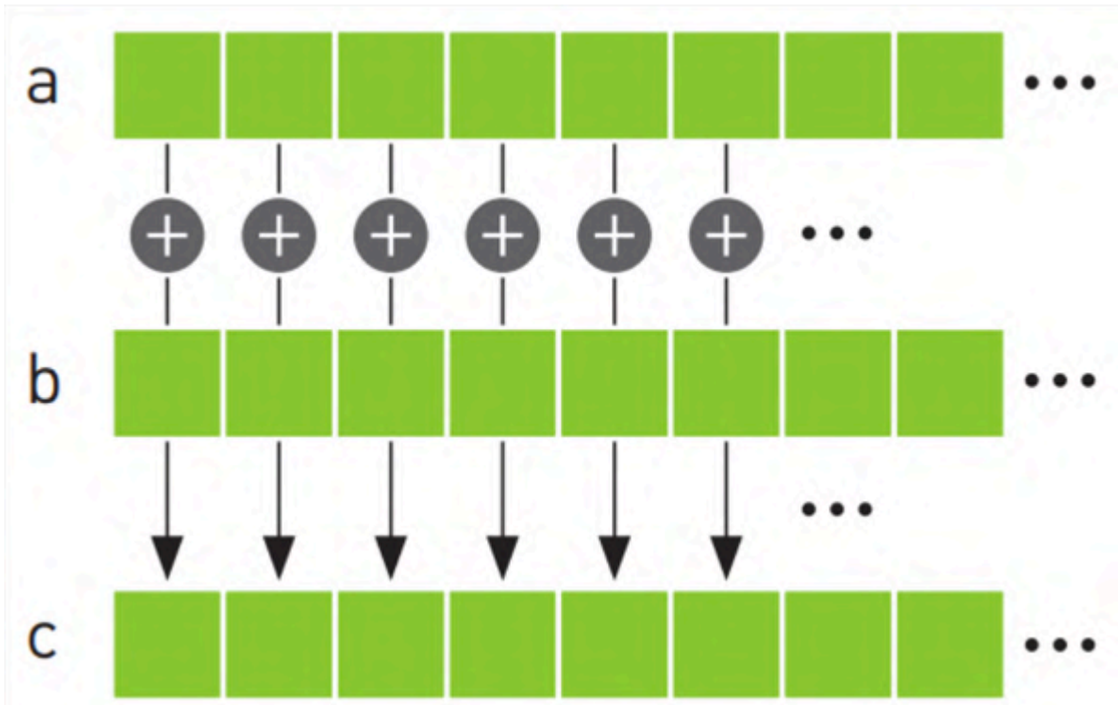
卷积神经网络 - CNN

通过卷基层和池化层的网络结构进行不断的对图像的特征提取



数组运算并行化 – CUDA by Example

将数组 a 和数组 b 相加并将计算结果放入数组 c 中。




```
void add( int *a, int *b, int *c ) {
    for (i=0; i < N; i++) {
        c[i] = a[i] + b[i];
    }
}
```

数组运算并行化 – CUDA by Example

当我们有多个 core 的时候

CPU CORE 1

```
void add( int *a, int *b, int *c )
{
    int tid = 0;
    while (tid < N) {
        c[tid] = a[tid] + b[tid];
        tid += 2;
    }
}
```

CPU CORE 2

```
void add( int *a, int *b, int *c )
{
    int tid = 1;
    while (tid < N) {
        c[tid] = a[tid] + b[tid];
        tid += 2;
    }
}
```

深度学习对于并行化硬件的依赖 - GPU

- Core 的多少往往决定真正并行化运算的数量

TESLA M60 FEATURES AND BENEFITS

- Two high-end NVIDIA Maxwell™ GPUs
- Up to 32 users per board
- 4096 NVIDIA CUDA® cores (2048 per GPU)
- 16 GB of GDDR5 memory (8 per GPU)
- 36 H.264 1080p30 streams
- ECC protection for increased reliability
- Server optimization to deliver the best throughput in the data center

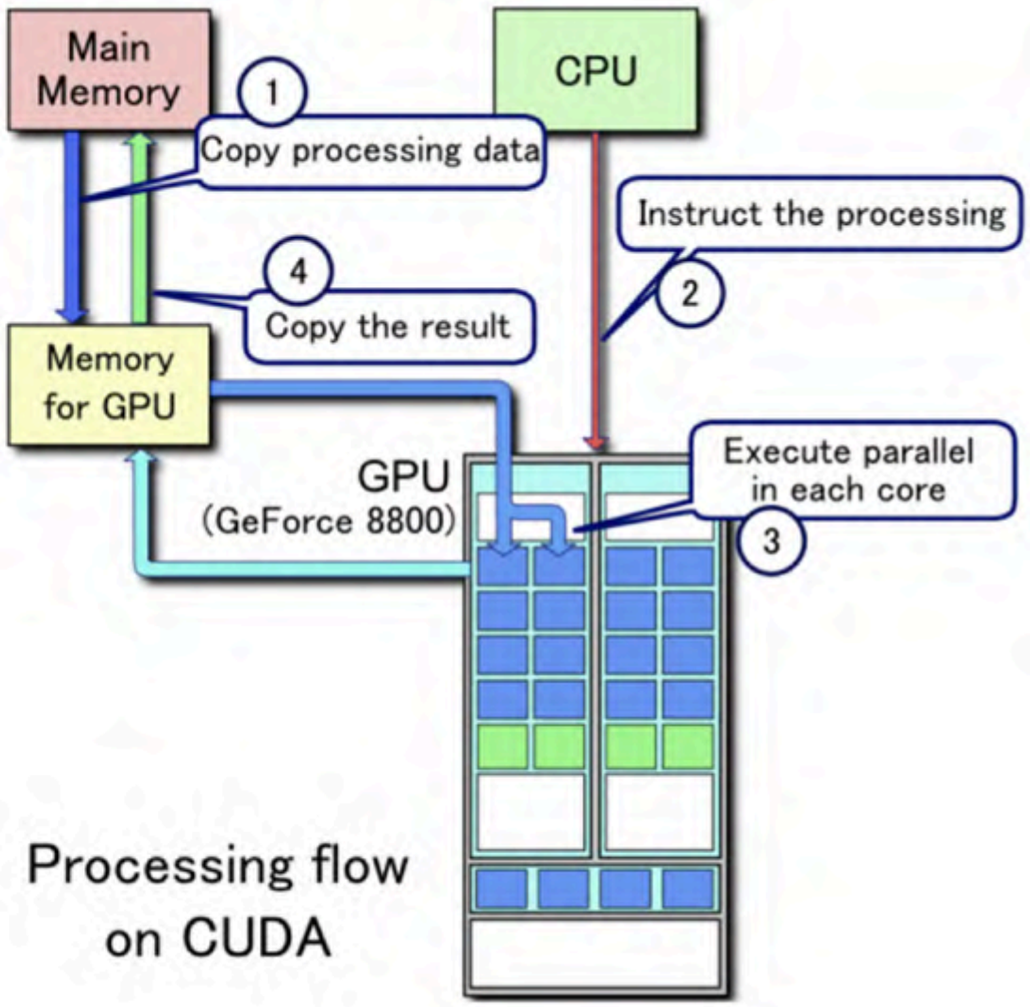
GPU 硬件使用流程

```
// allocate the memory on the GPU
HANDLE_ERROR( cudaMalloc( (void**)&dev_a, N * sizeof(int) ) );
// copy the arrays 'a' and 'b' to the GPU
HANDLE_ERROR( cudaMemcpy( dev_a, a, N * sizeof(int),
                          cudaMemcpyHostToDevice ) );
add<<<N,1>>>( dev_a, dev_b, dev_c );
// copy the array 'c' back from the GPU to the CPU
HANDLE_ERROR( cudaMemcpy( c, dev_c, N * sizeof(int),
                          cudaMemcpyDeviceToHost ) );
```

NVIDIA CUDA

Example of CUDA processing flow

1.复制存储器至GPU
2. CPU指令驱动GPU
3. GPU每一核心并行处理
4. GPU将结果传回主存



Processing flow on CUDA

AI 模型

- AI 模型会决定最终使用资源的多少
- AI 模型的服务性能还与网络相关
- 并不是所有 AI 模型都适合通过 GPU 加速

TABLE OF **CONTENTS**

AI 云平台的价值

- 为 AI 工程师提供一体化的研发工作环境
- 为 AI 对于硬件资源需求提供弹性伸缩
- 为 AI 模型在生产环境中部署及运行提供保障

AI 模型实现工具及其框架举例

- 不同的框架和工具都有其优点长处。

dmlc XGBoost

BVLC / caffe

mxnet

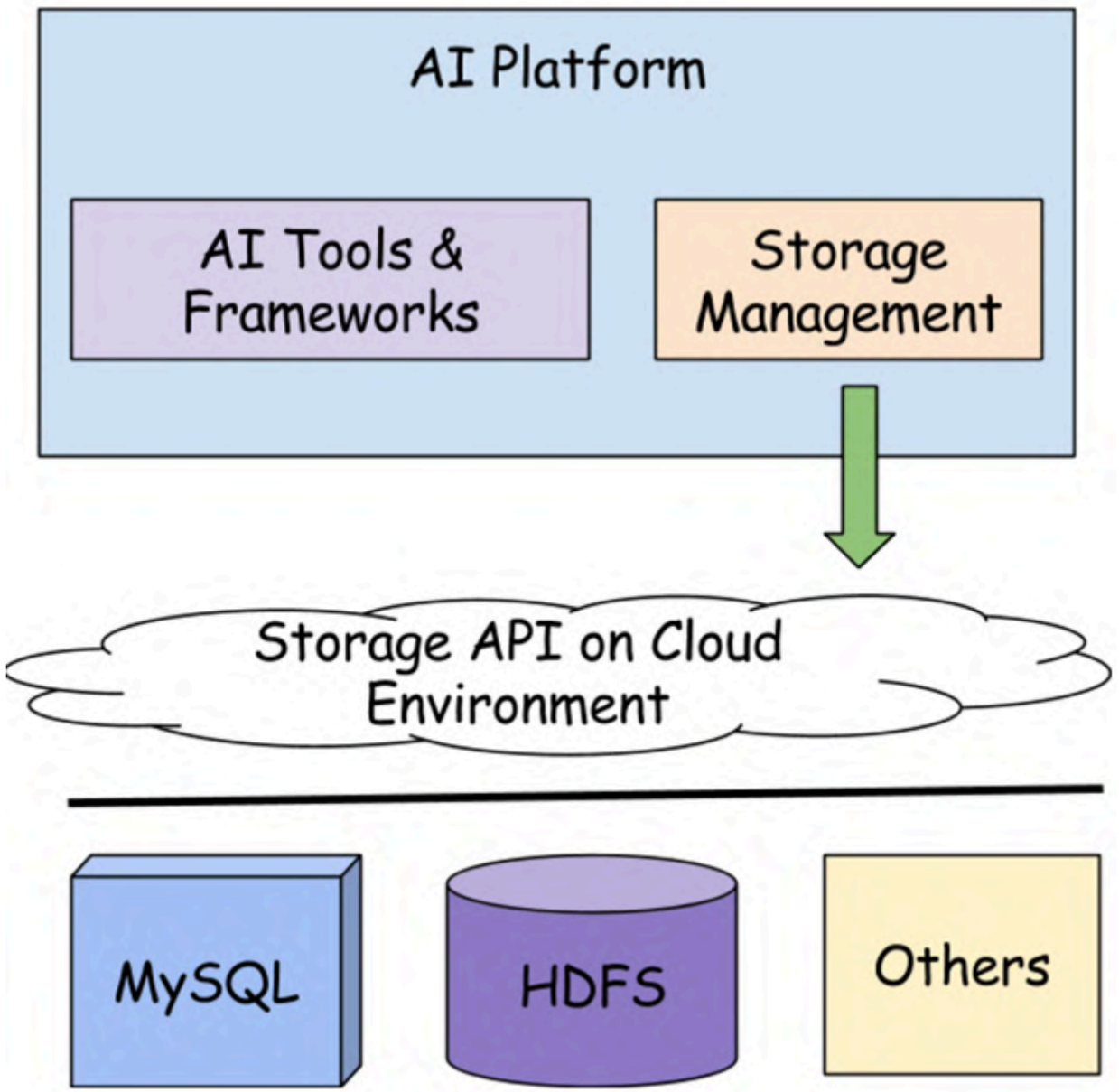
TensorFlow

Cognitive Toolkit

AI 工程师工作流程

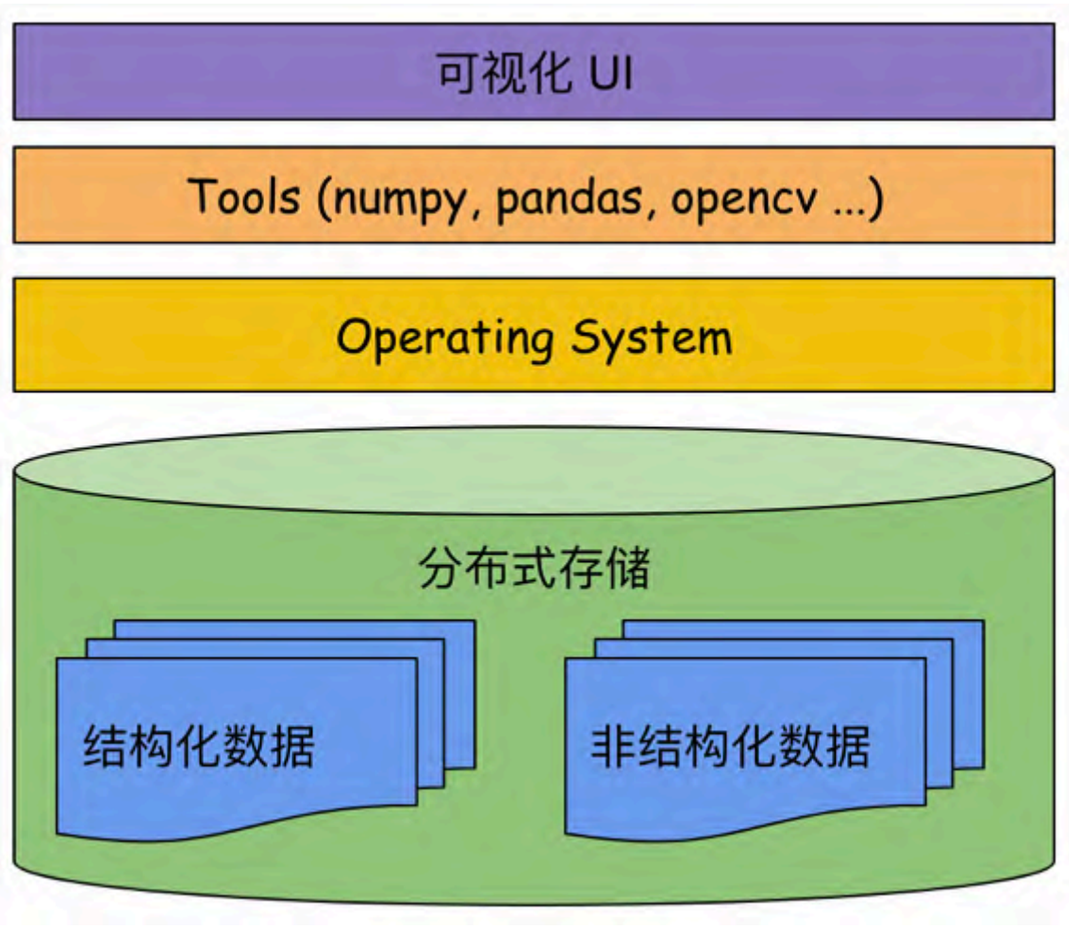
存储对接

- 存储通常都由云平台通过SaaS服务提供
- 在 AI 平台中对于存储进行统一的管理和操作

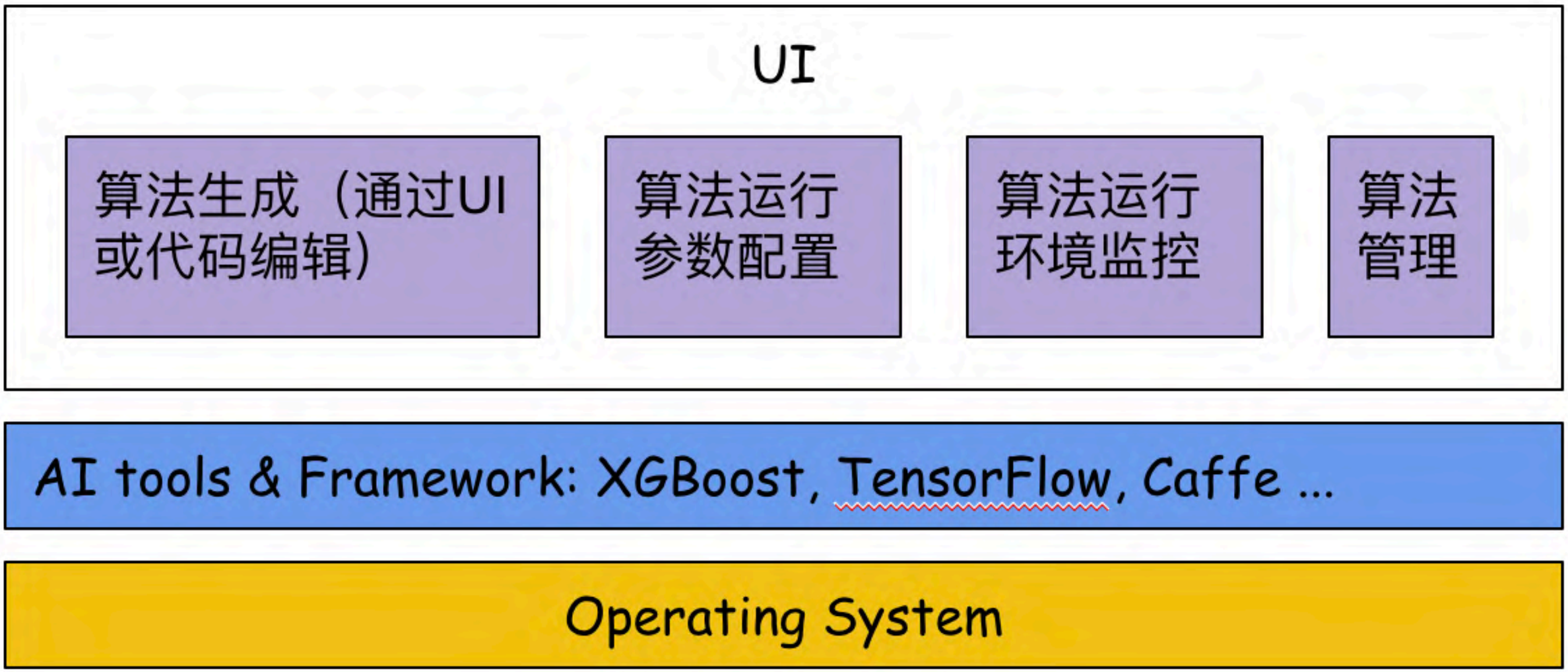


数据预处理

AI 工程师的大部分工作都是在做数据预处理的部分，而可视化 UI 则可以大大减少 AI 工程师的工作量。



AI 工具及其研发框架的整合



模型训练资源池管理

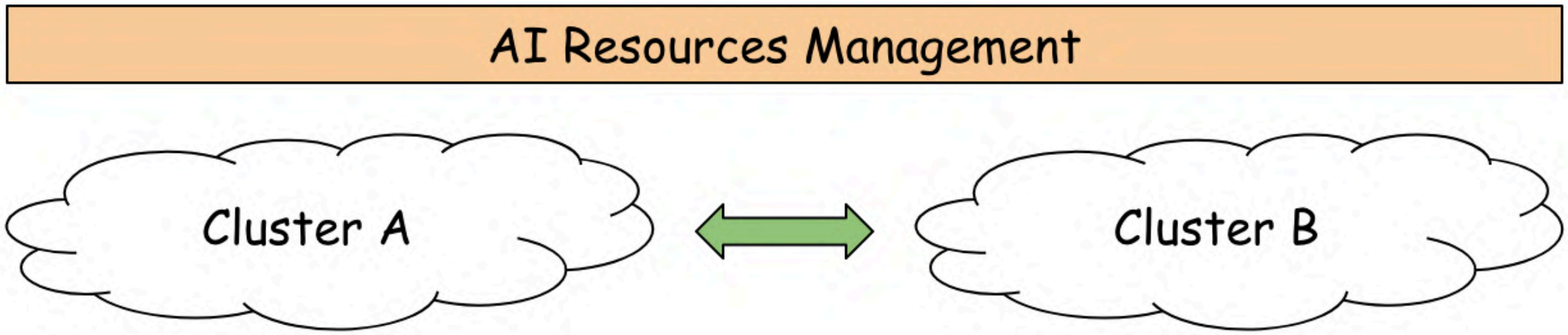
- AI 模型训练会耗费巨大的资源并且长时间占用
- 多个用户在模型训练时需要通过队列的方式来解决资源短缺问题
- 需要对不同用户进行资源池划分

模型管理与发布

- 模型发布：
  - 模型服务的负载均衡
  - 硬件资源的规划
- 模型管理：
  - 模型的版本
  - 模型的类型

研发环境与生产环境隔离

- 硬件资源互相隔离
- 网络资源相互连通



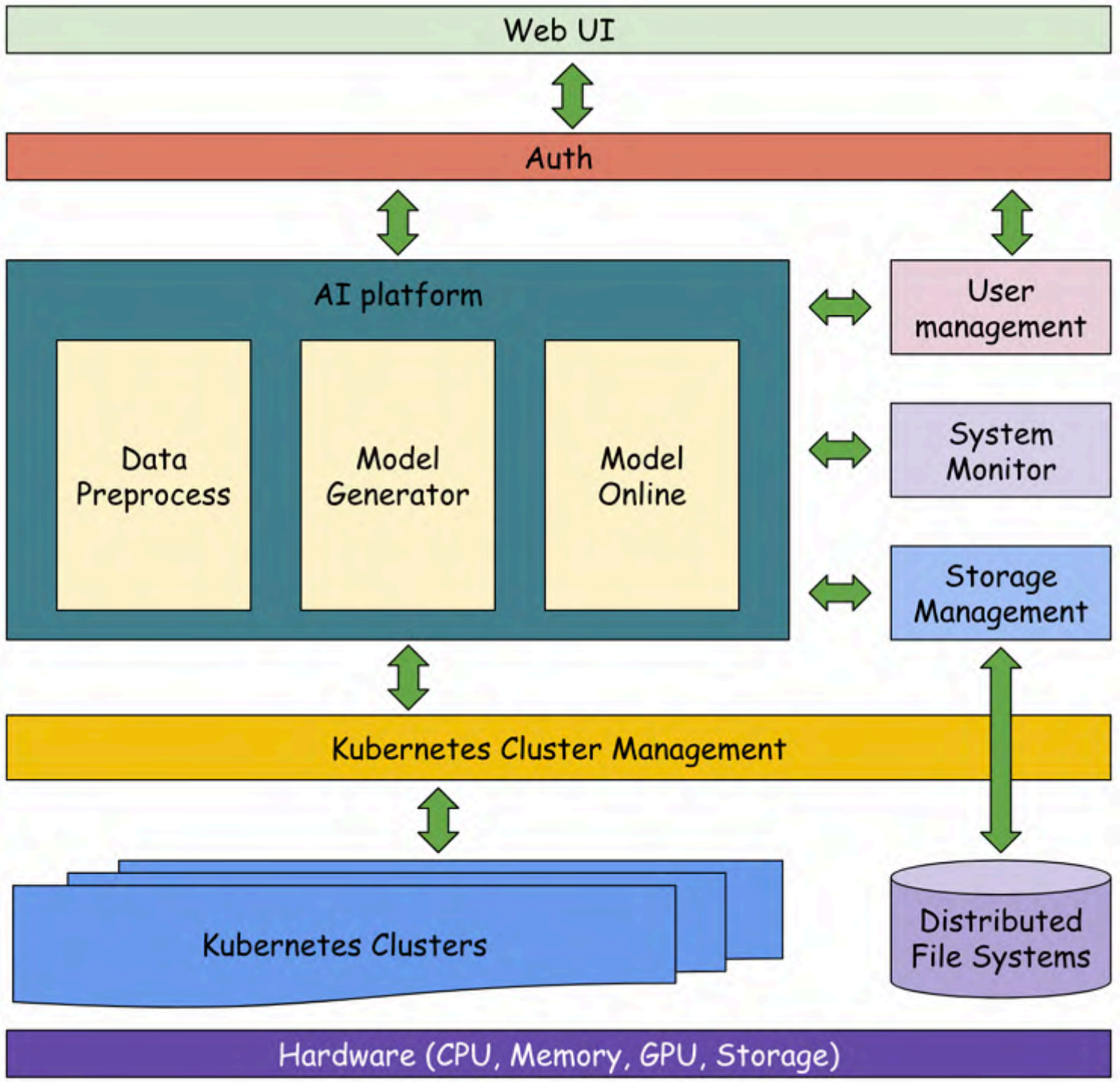
资源监控及分配策略

- 用户资源的限制
- - 创建训练任务的限制
- - 创建模型服务的限制
- 资源使用的负载均衡状况
- - 总体资源使用的状况
- - 单个用户的资源使用状况

TABLE OF **CONTENTS**

AI 云平台与 Kubernetes 融合

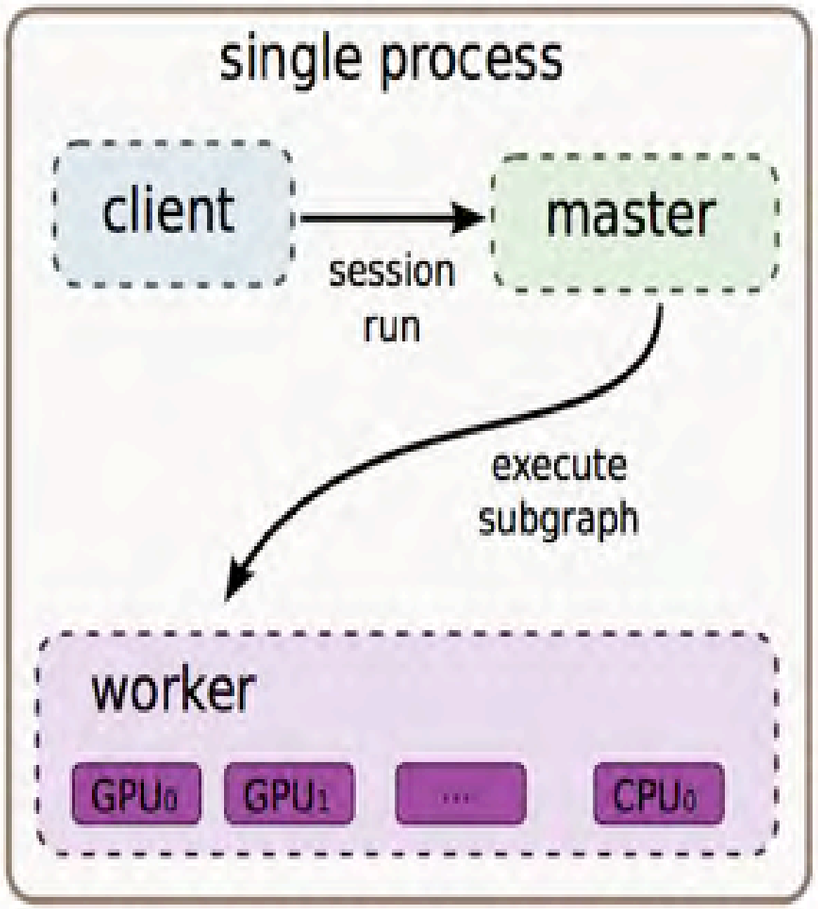
- Kubernetes 作为 AI 平台的一个核心调度和任务管理平台
- AI 业务层负责将具体的业务逻辑实现，并与 Kubernetes 层对接
- 在底层硬件选型需要注意适合 AI 模型训练和在线服务的类型，例如10G及以上的networking和GPU

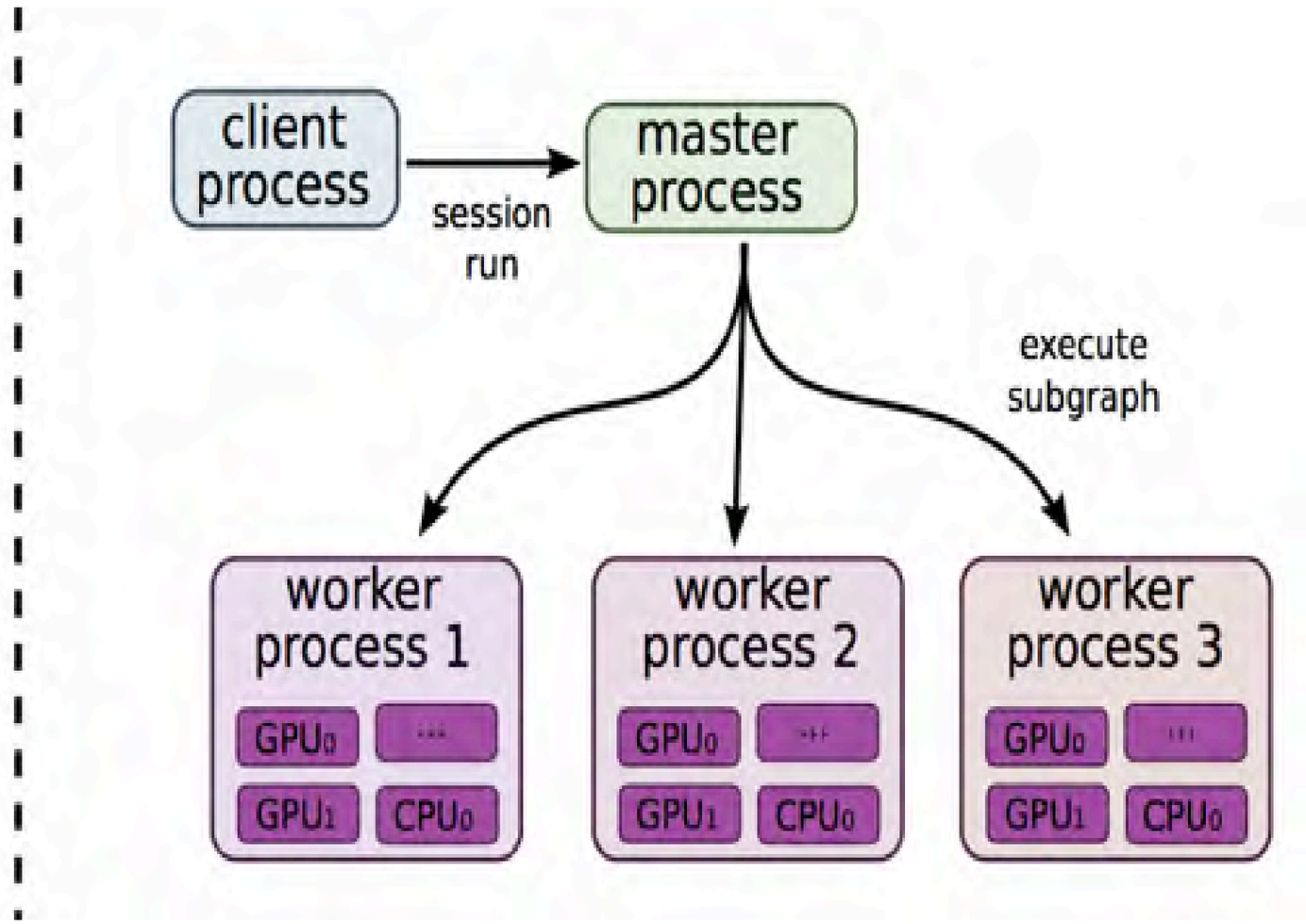


TensorFlow 介绍

- TensorFlow™ 是一个使用数据流图进行数值计算的开源软件库。图中的节点代表数学运算， 而图中的边则代表在这些节点之间传递的多维数组（张量）。这种灵活的架构可让您使用一个 API 将计算工作部署到桌面设备、服务器或者移动设备中的一个或多个 CPU 或 GPU。 TensorFlow 最初是由 Google 机器智能研究部门的 Google Brain 团队中的研究人员和工程师开发的，用于进行机器学习和深度神经网络研究， 但它是一个非常基础的系统，因此也可以应用于众多其他领域。

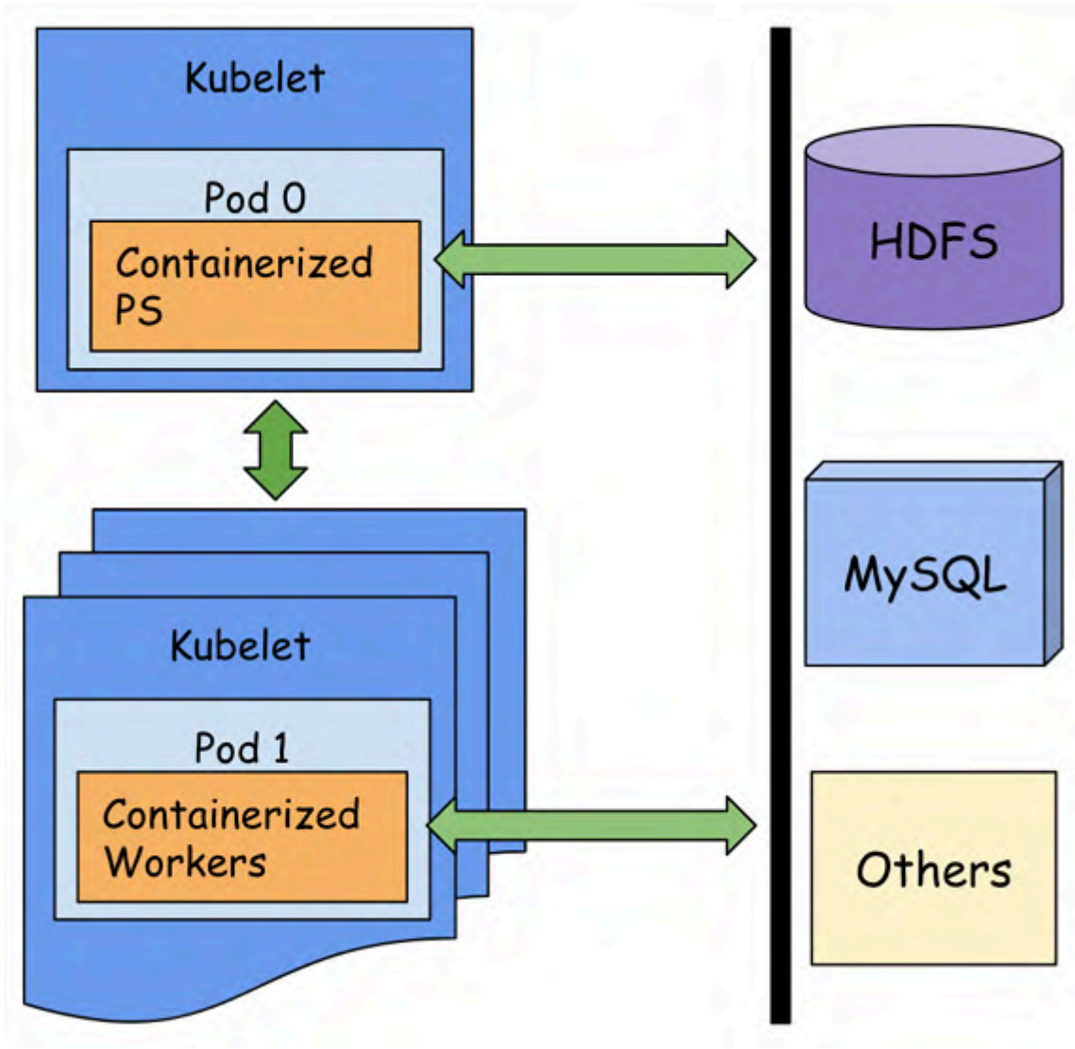
分布式 TensorFlow



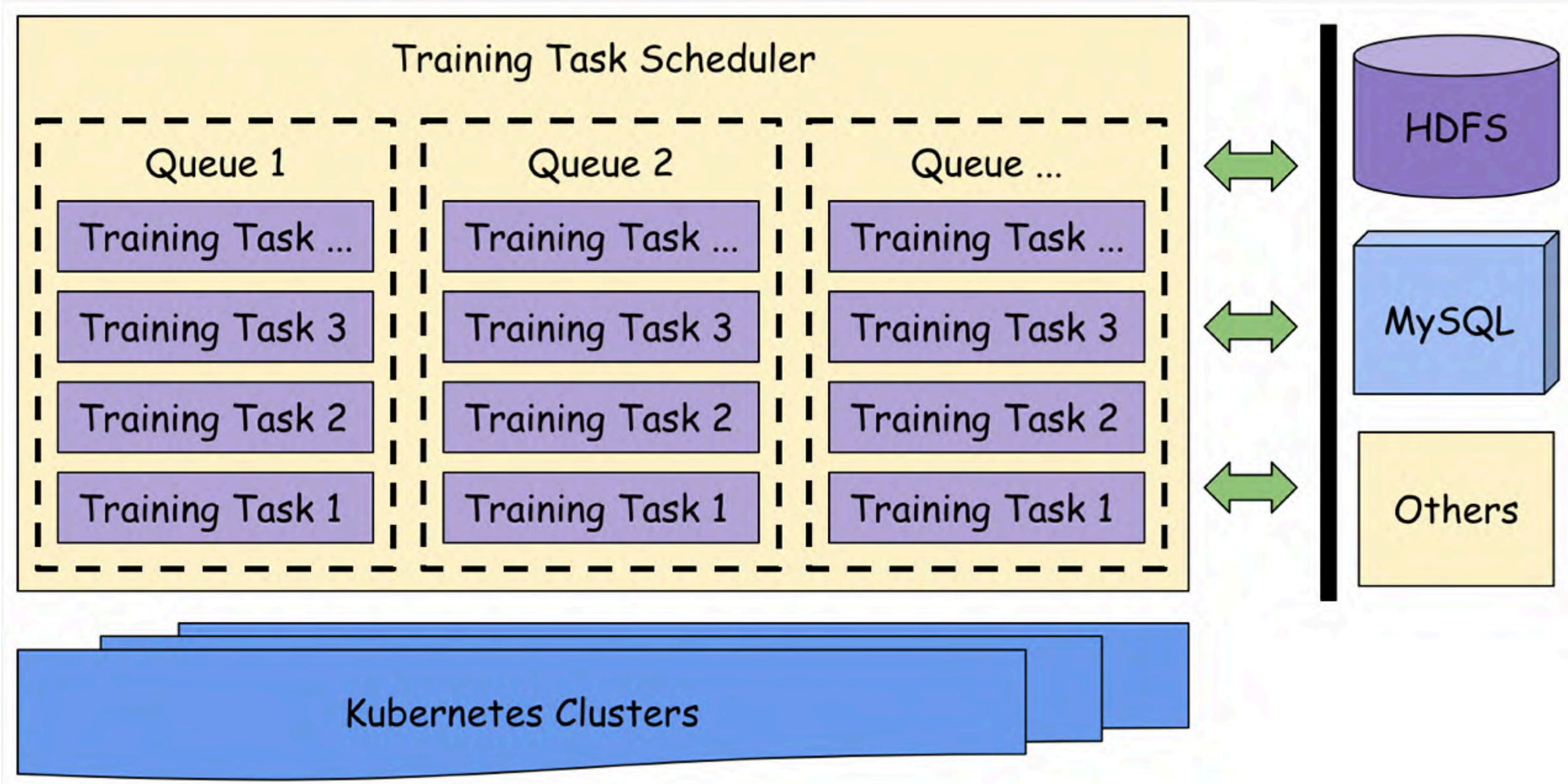


存储对接

通过 Kubernetes 的 Volume 机制将分布式存储的具体地址 mount 到 TensorFlow PS 和 Worker 的 Container 内部，使 Container 可以得到需要处理的数据

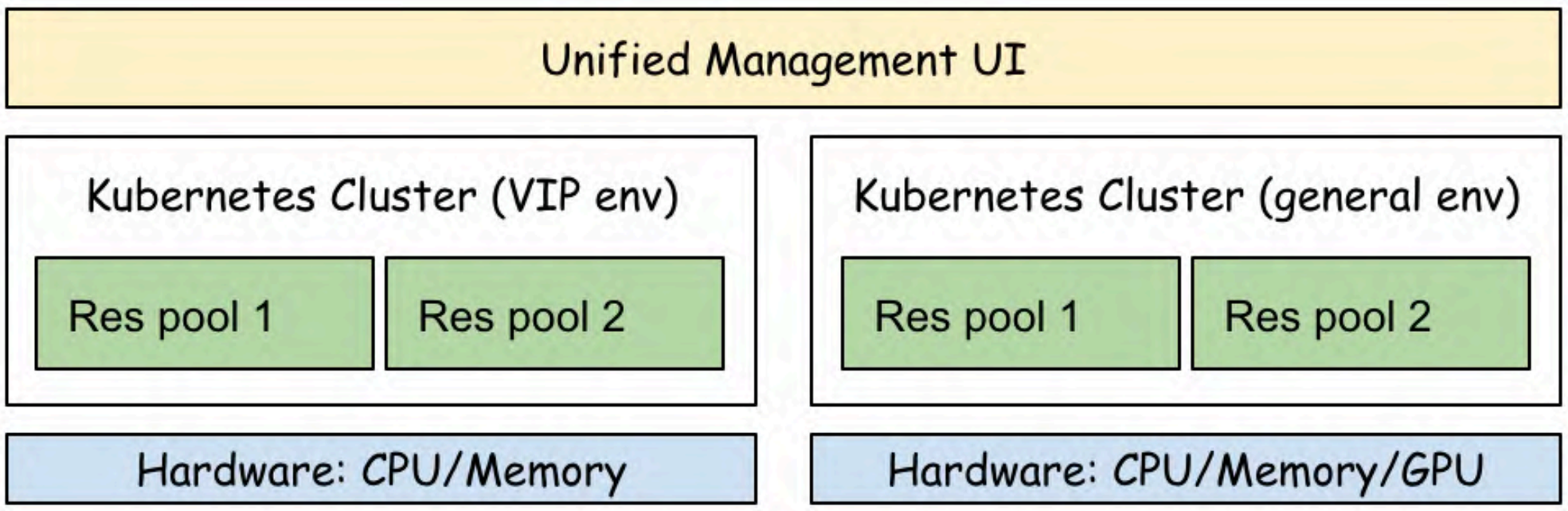


资源队列

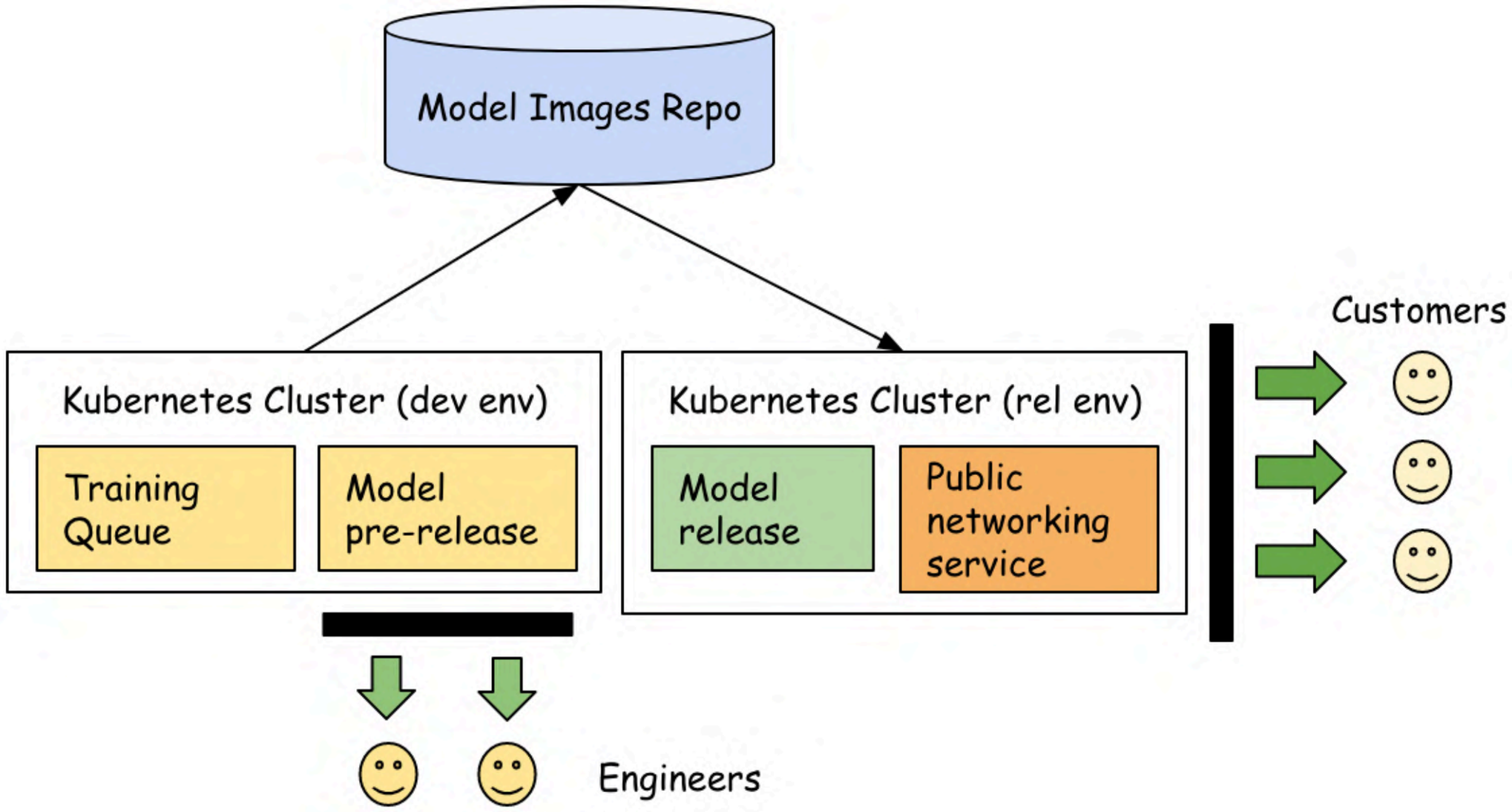

Training Task Scheduler
Queue 1
Queue 2
Queue ...
Training Task ...
Training Task ...
Training Task ...
Training Task 3
Training Task 3
Training Task 3
Training Task 2
Training Task 2
Training Task 2
Training Task 1
Training Task 1
Training Task 1
HDFS
MySQL
Others
Kubernetes Clusters

资源队列在不同的集群分布

- 针对不同的用户设置不同的资源池
- 资源池存在硬件分配原则的差异，例如资源的多少，类型。



模型发布


Model Images Repo
Kubernetes Cluster (dev env)
Training Queue
Model pre-release
Kubernetes Cluster (rel env)
Model release
Public networking service
Customers
Engineers

AI Platform 本身的运行

- 1. 容器化
- 2. 对接 Kubernetes API，实现资源的分配与调度
- 3. 进行多 Kubernetes Cluster 的配置（需要进行跨集群的操作）

Caicloud TaaS 云平台

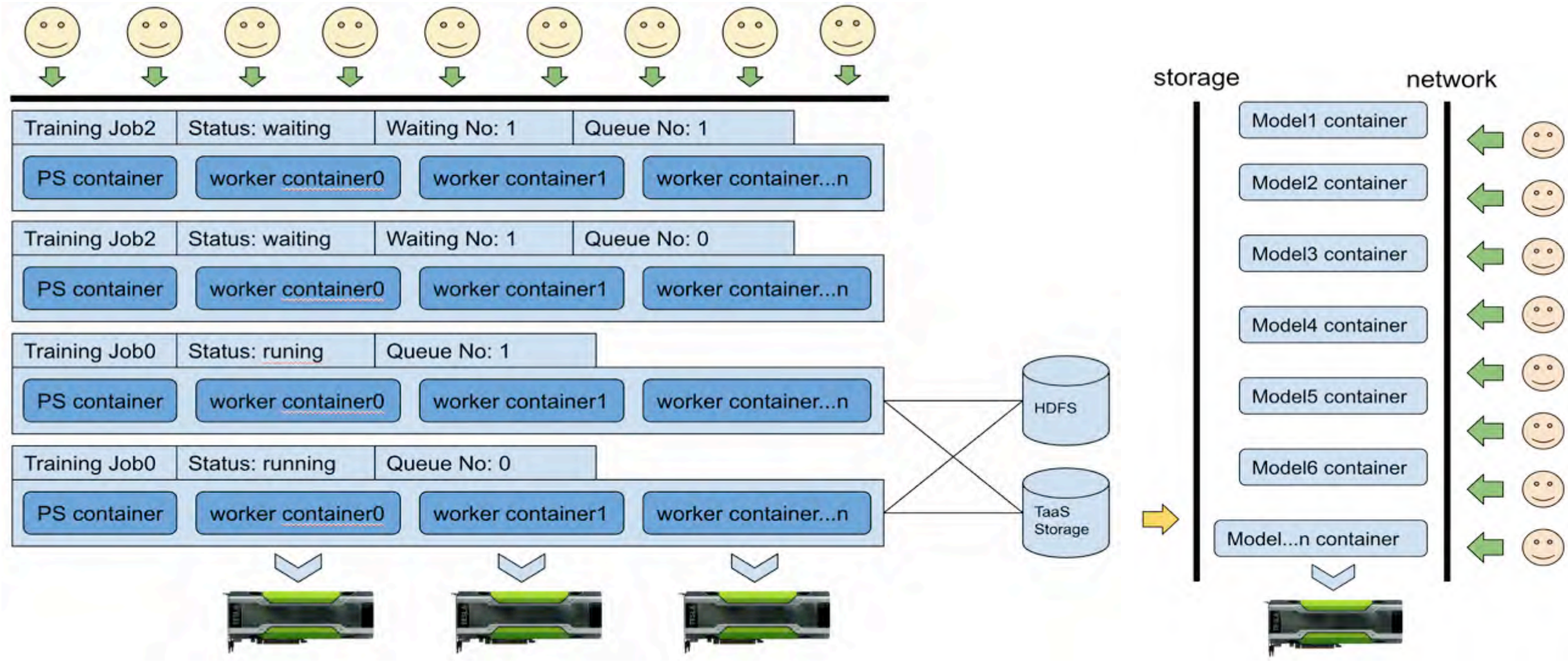
TensorFlow as a Service (TaaS) 平台专注于将Google级AI 模型生成及服务工具TensorFlow打造成一个成熟的商业级人工智能平台为个人和企业解决计算资源短缺、环境配置复杂、资源管理缺失等问题.

- 集群资源弹性管理
- 用户 / 硬件资源管理
- 用户 / 硬件资源监控
- 易用的模型 / 服务部署
- 商业级的模型服务平台

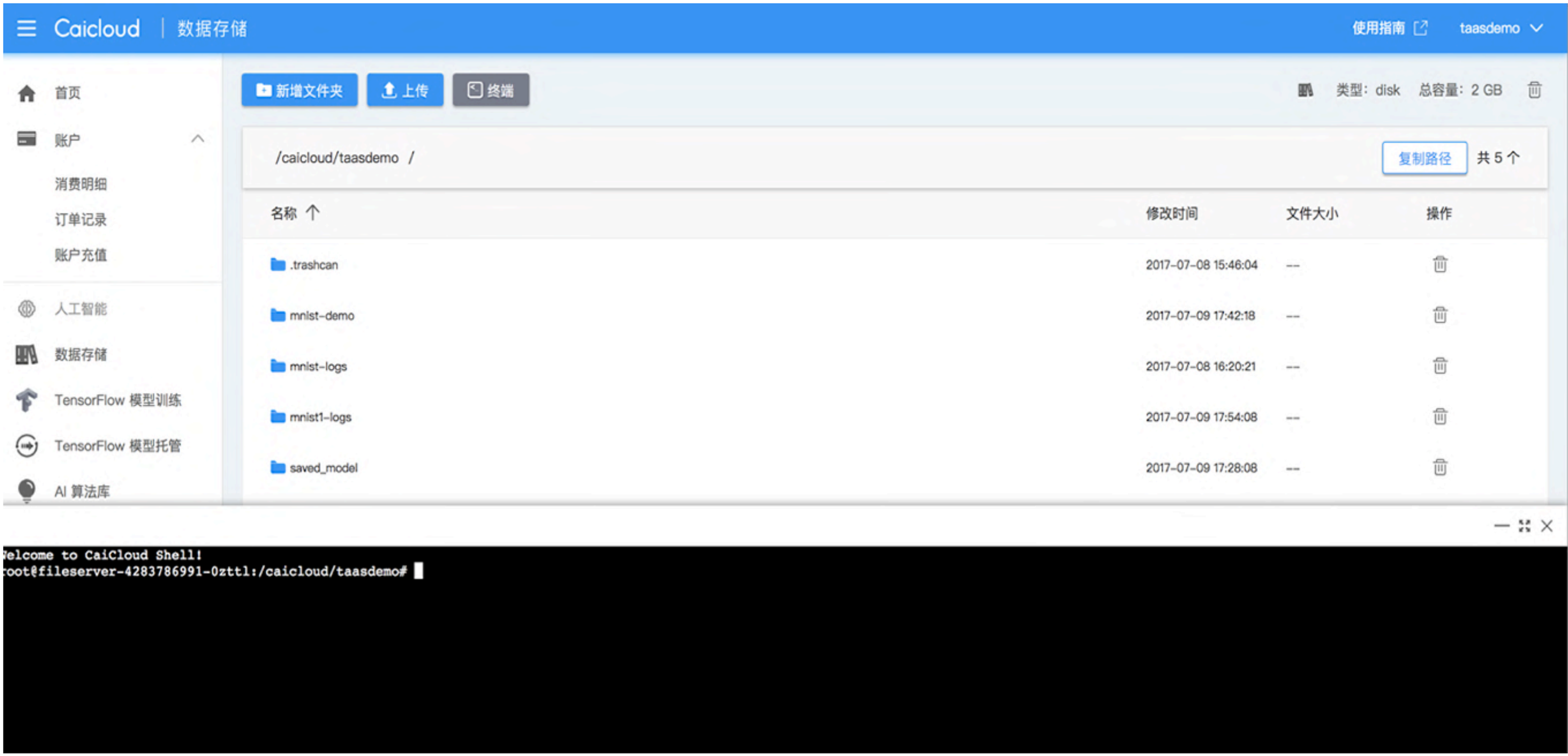
TaaS 主要功能列表

功能类别	功能	公有云	私有云
企业级功能	用户权限管理	N/A	有
	存储管理	N/A	有
	资源管理	N/A	有
个人研发功能	模型训练	有	有
	模型上线服务	即将上线	有
	原生态 TensorBoard 支持	即将上线	有
	训练项目管理	有	有
	代码编辑	有	有
	GPU 支持	即将上线	有
	远程命令行支持	有	有

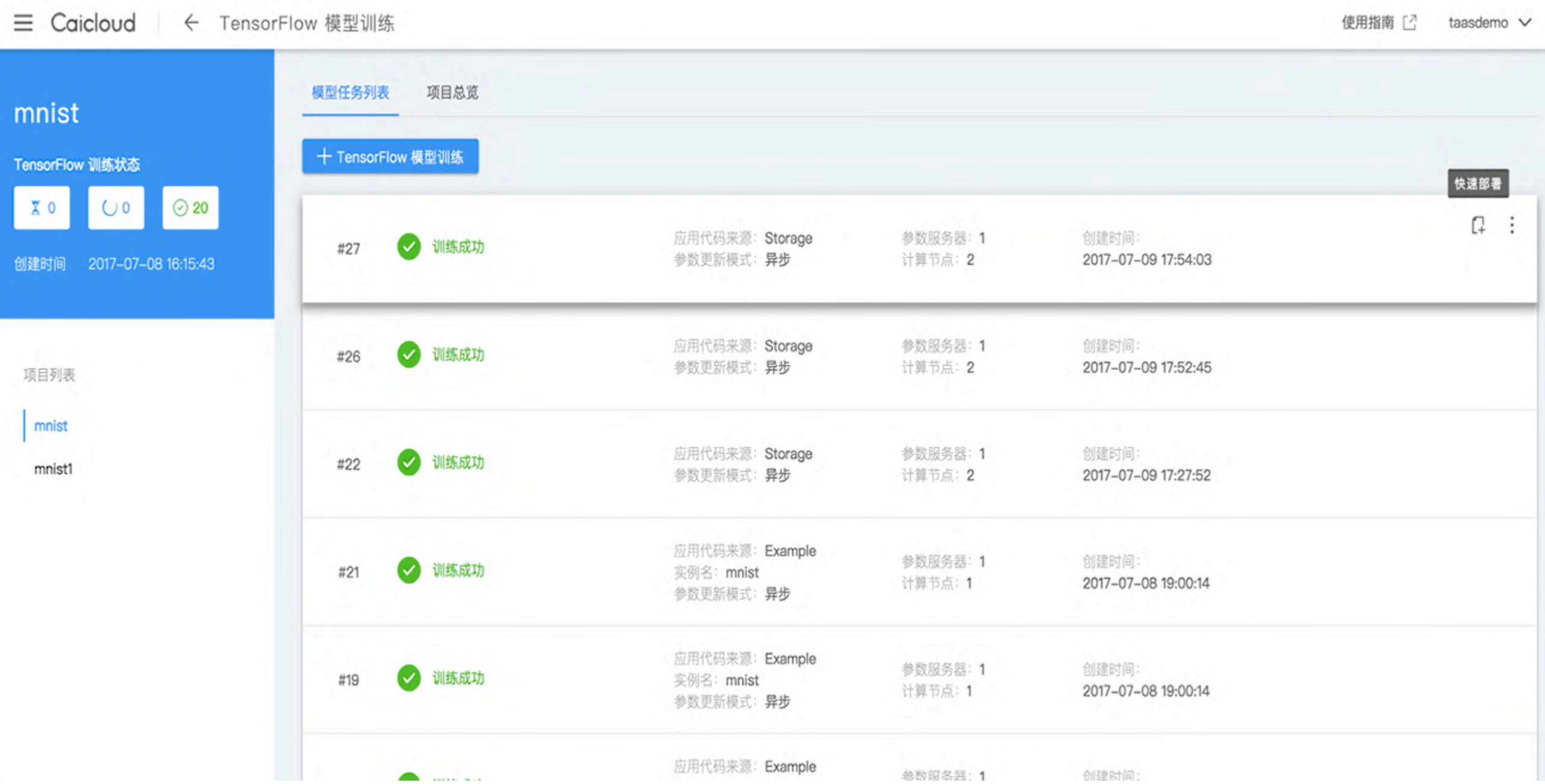
TaaS 运行原理


Training Job2
Status: waiting
Waiting No: 1
Queue No: 1
PS container
worker container0
worker container1
worker container...n
Training Job2
Status: waiting
Waiting No: 1
Queue No: 0
PS container
worker container0
worker container1
worker container...n
Training Job0
Status: runing
Queue No: 1
PS container
worker container0
worker container1
worker container...n
Training Job0
Status: running
Queue No: 0
PS container
worker container0
worker container1
worker container...n
HDFS
TaaS Storage
storage
network
Model1 container
Model2 container
Model3 container
Model4 container
Model5 container
Model6 container
Model...n container

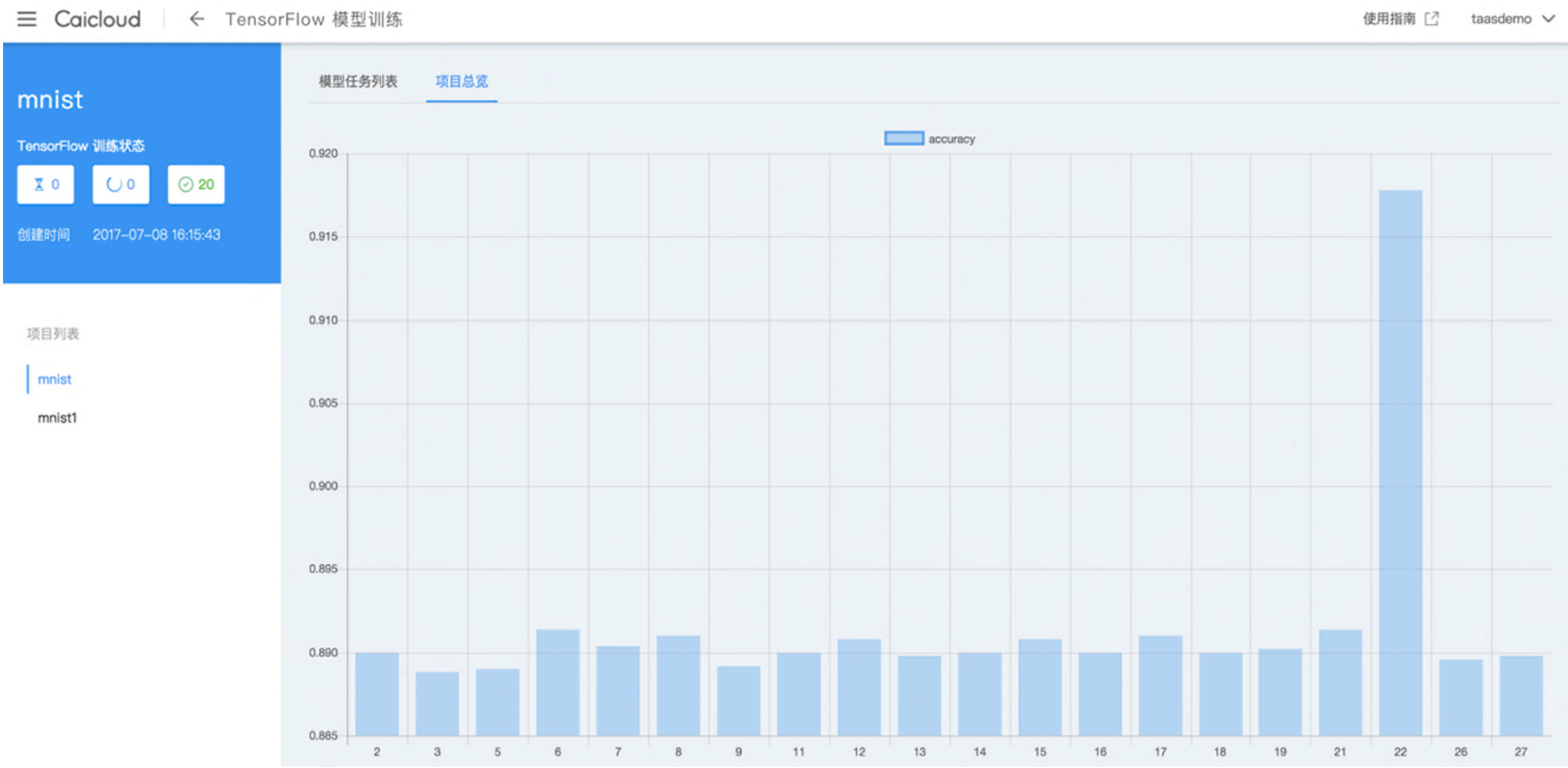
TaaS 主要功能 – 数据存储



TaaS 主要功能 – 项目管理

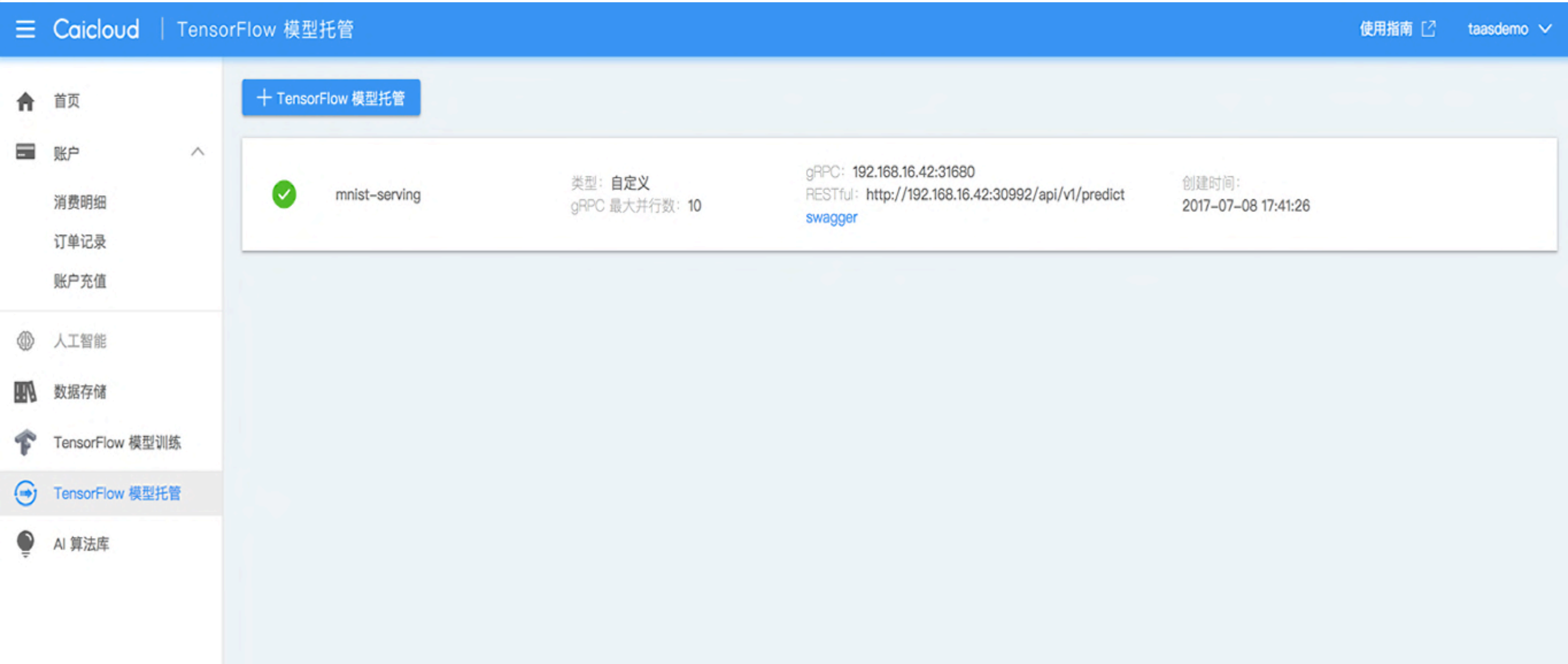


TaaS 主要功能 – 项目精准度分析及汇总

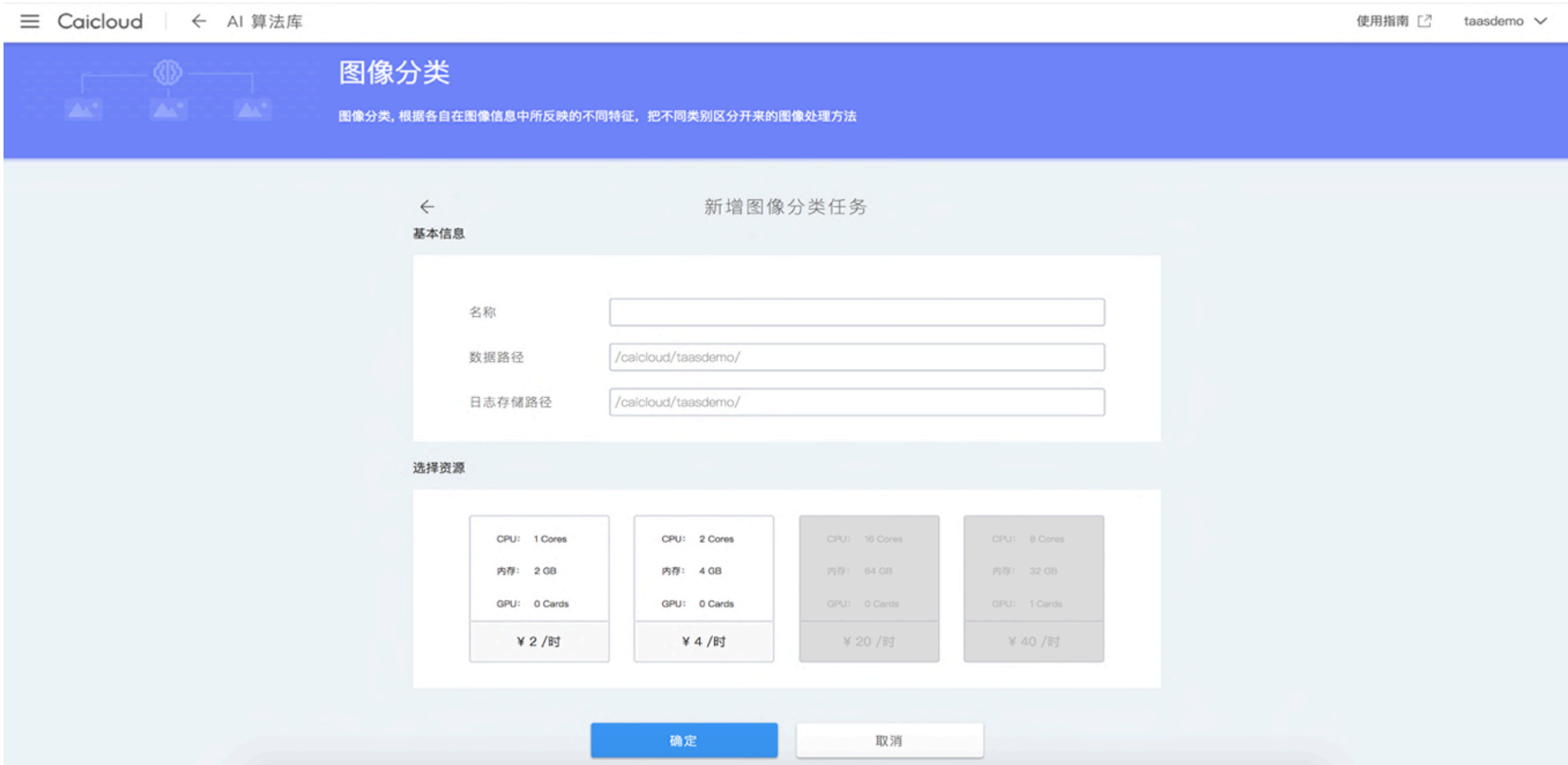


TaaS 主要功能 – 模型训练

TaaS 主要功能 – 模型上线服务



TaaS 主要功能 – AI 算法库



THANKS!

智 能 时 代 的 新 运 维

CNUTCon 2017