

# Kubernetes 完全教程

## Kubernetes 架构概述

王渊命 @jolestar

# Agenda

1. Kubernetes 为何而生
2. Kubernetes 的架构

# Kubernetes 为何而生 - 云发展到一个新阶段

## IaaS 云解决了哪些问题

按需购买

接管硬件资源的运维

提供可编程接口来管理资源

提供 SDN, SDS 模拟硬件网络以及存储

## 特点

对应用无侵入

面向资源

| 用户从关注资源的运维转向关注应用的开发运维成本

# Kubernetes 为何而生 - 容器的成熟奠定了基础

容器（Docker/Moby）解决了哪些问题

应用安装包的标准化（Image）

应用进程的标准化（Container）

特点

单进程标准化

# 容器编排系统应运而生

我们需要一种 面向应用（Application Oriented） 的系统来降低服务端应用的开发部署和运维成本

We wanted people to be able to program for the data center just like they program for their laptop --Ben Hindman

我们再引申一下，从开发延伸到部署运维

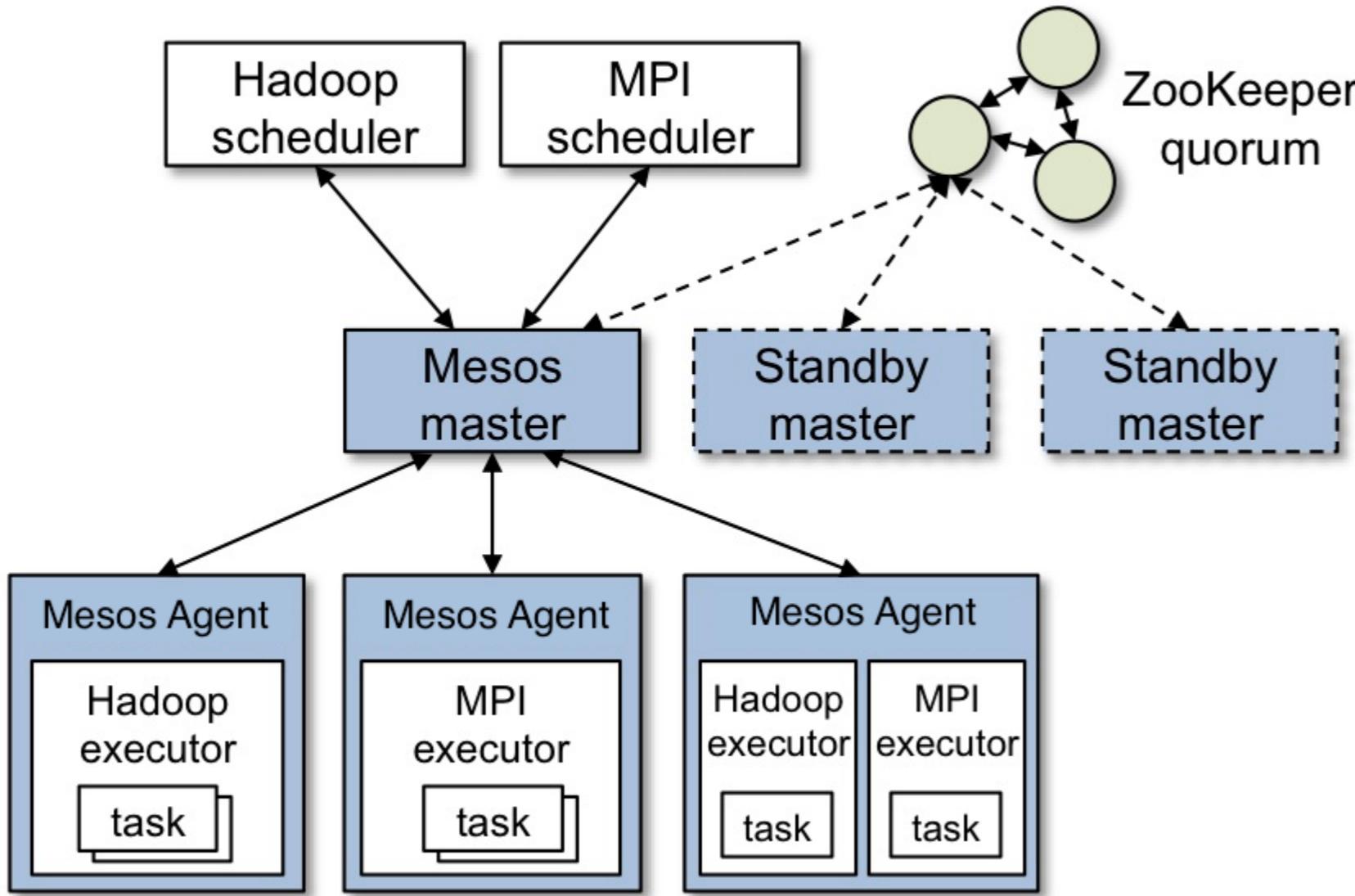
We wanted people to be able to manager app for the data center just like they manager app on their laptop

# Borg, Mesos, Omega, and Kubernetes

1. [《Borg, Omega, and Kubernetes》](#)

2. Mesos vs Kubernetes

# Mesos



Mesos 架构以及源码浅析

@jolestar

# Mesos vs Kubernetes

1. 编程框架 vs 运行平台
2. 资源共享 vs 定义状态
3. 分布式调度 vs 状态控制器

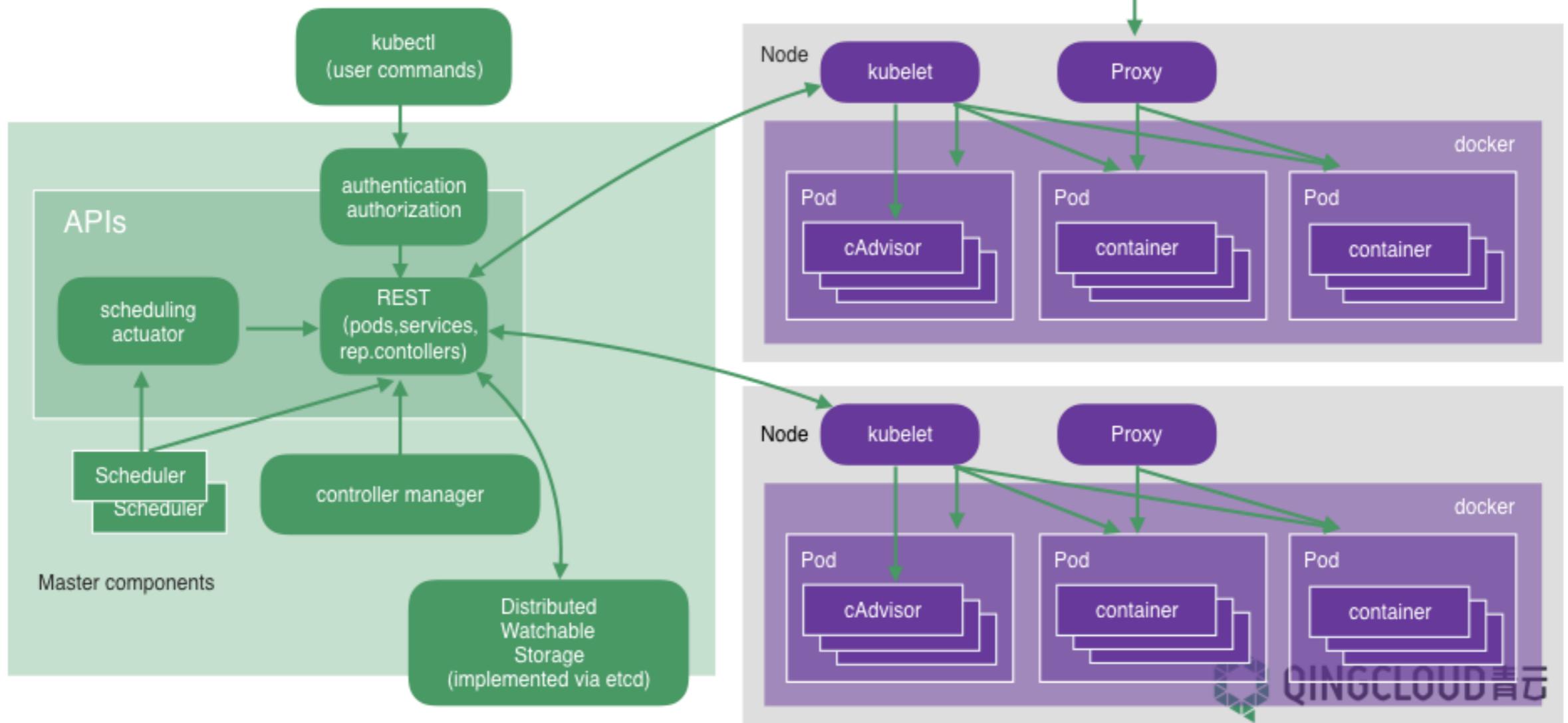
define a minimal interface that enables efficient resource sharing across frameworks, and otherwise push control of task scheduling and execution to the frameworks -- mesos

Kubernetes is not a mere “orchestration system”; it eliminates the need for orchestration. The technical definition of “orchestration” is execution of a defined workflow: do A, then B, then C. In contrast, Kubernetes is comprised of a set of independent, composable control processes that continuously drive current state towards the provided desired state. It shouldn’t matter how you get from A to C: make it so. -- kubernetes

# Kubernetes 的架构 - 始于编排而超越编排

1. Kubernetes 的部署架构
2. Kubernetes 的逻辑架构

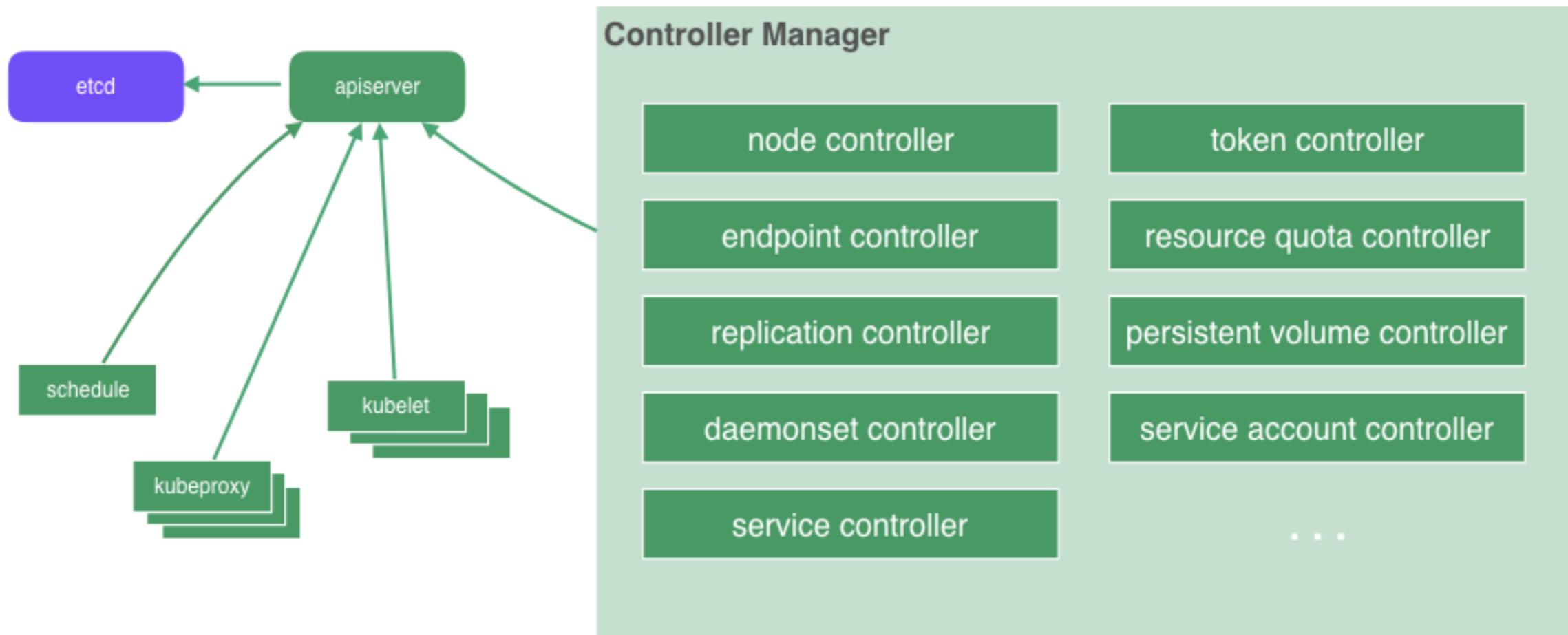
# Kubernetes 架构



# Kubernetes 逻辑架构

1. Declare, Observe, React
2. 一个状态存储
3. 多个控制器

# 一个状态存储，多个控制器



# Kubernetes 的架构优势

1. 自愈（最终一致）
2. 组合（低级组件组合成高级组件）
3. 面向未来（API 定义目标，而不是过程）

```
apiVersion: vx
kind: API-Sla
metadata:
  name: myservice-sla
spec:
  percent: 99%
  latency: 200ms
  cost: x$/h
  selector:
    app: myapp
```

# Kubernetes 的架构劣势

1. 同步操作需求比较难实现
2. 非状态操作不好实现

# Kubernetes 尚未解决的问题（开放式问题）

1. 配置管理 ([Declarative application management in Kubernetes](#))
2. 服务依赖

# 作业

1. 阅读本次课程中给出的论文，以及 Kubernetes 的官方文档，结合自己的编程运维经验，思考 Kubernetes 能解决你遇到的哪些问题，哪些可能解决不了。
2. 手动搭建一个 Kubernetes 进行试验，初步熟悉 Kubernetes 的命令以及 API。（推荐通过 QingCloud 青云的 appcenter 进行部署。<https://appcenter.qingcloud.com/apps/app-u0llx5j8>）

## 关于我

个人博客: <http://jolestar.com>



午夜咖啡

工具 · 架构 · 成长 · 思考

公众号: jolestar-blog

个人博客: <http://jolestar.com>

☞ 微信扫描关注 午夜咖啡

# 关注我们



QingCloud-IaaS



青云QingCloud

[www.qingcloud.com](http://www.qingcloud.com)

