

---

# MESOS

## 数据中心操作 系统的核心

## 演讲者简介： 俞捷



- **Mesosphere** 技术总监
- **Apache Mesos Committer**, 项目管理委员会委员
- 主要负责容器，网络和存储等项目的研发
- 曾任**Twitter**高级工程师，参与**Mesos**的开发与运维
- **University of Michigan** 计算机科学与工程博士
- 本科毕业于复旦大学

Twitter: @jie\_yu

Github: jieyu

---

# 演讲简介

- Mesos的起源及架构简介
- 选择Mesos的理由
- Mesos最新功能介绍

# Mesos: 数据中心操作系统的内核

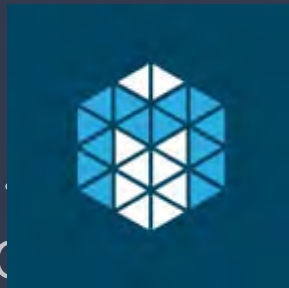
- 传统的操作系统内核

- 资源管理： 单机CPU, 内存, 磁盘, ...
- 编程抽象： POSIX API：进程, 线程, ...
- 安全与隔离： 虚拟内存, 用户, ...



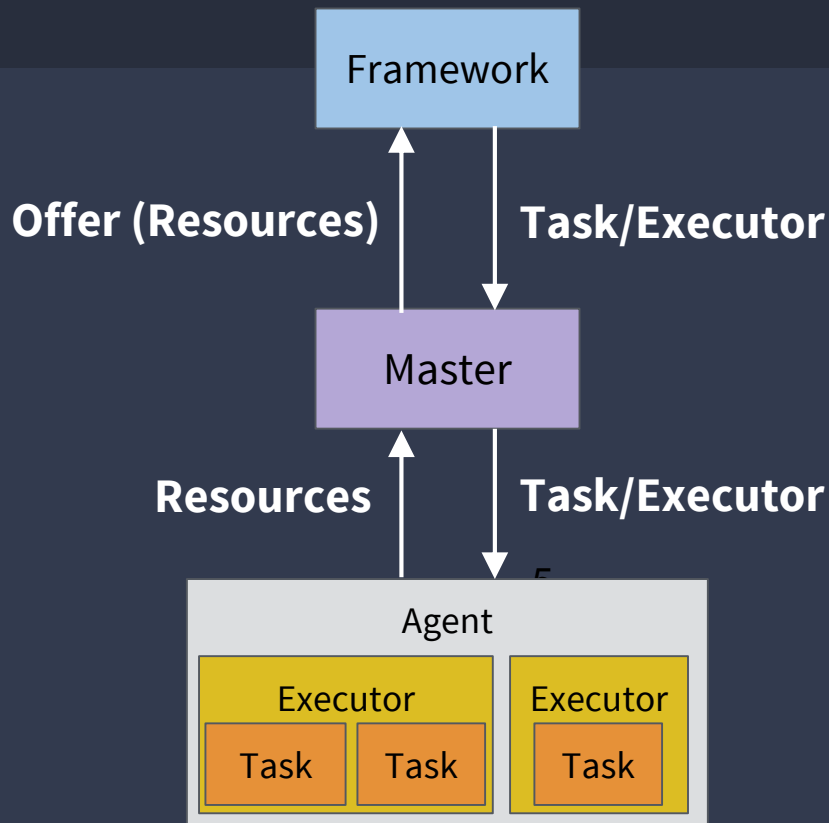
- 数据中心操作系统的内核

- 资源管理： 集群CPU, 内存, 磁盘, ...
- 编程抽象： Mesos API：Resource, Container, ...
- 安全与隔离： 容器化

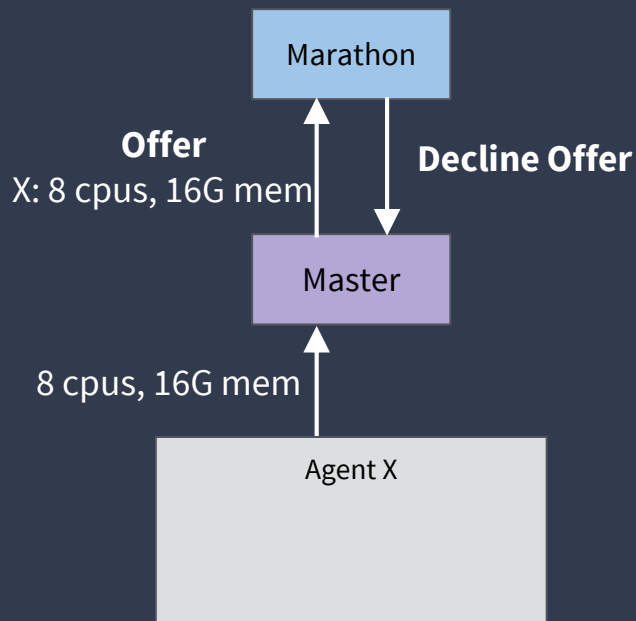


# Mesos的编程抽象

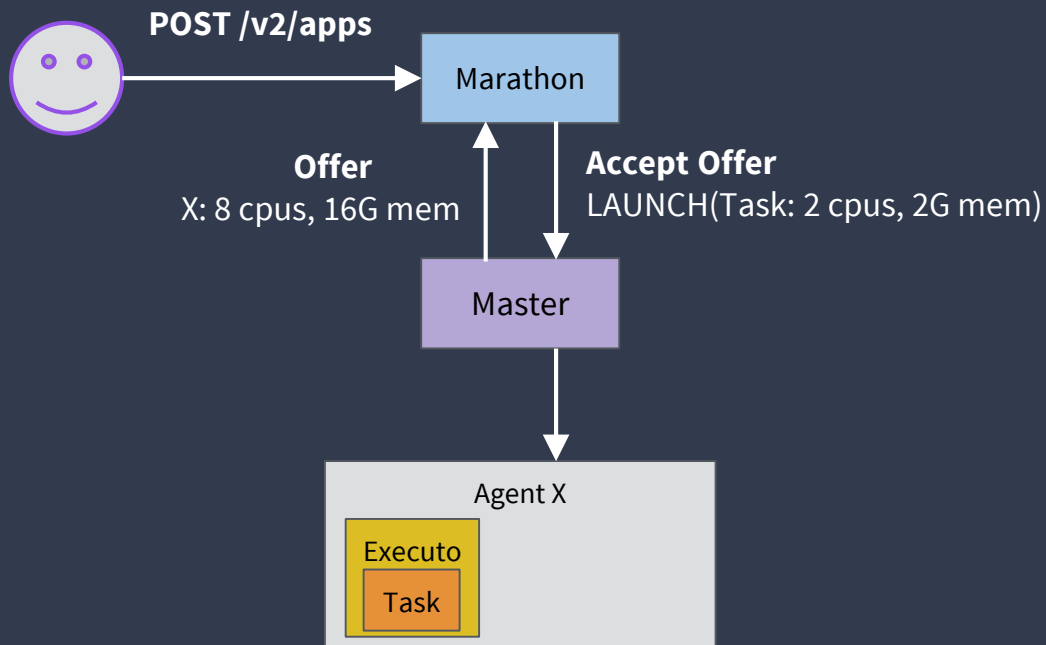
- Framework
- Resource/Offer
- Task
- Executor



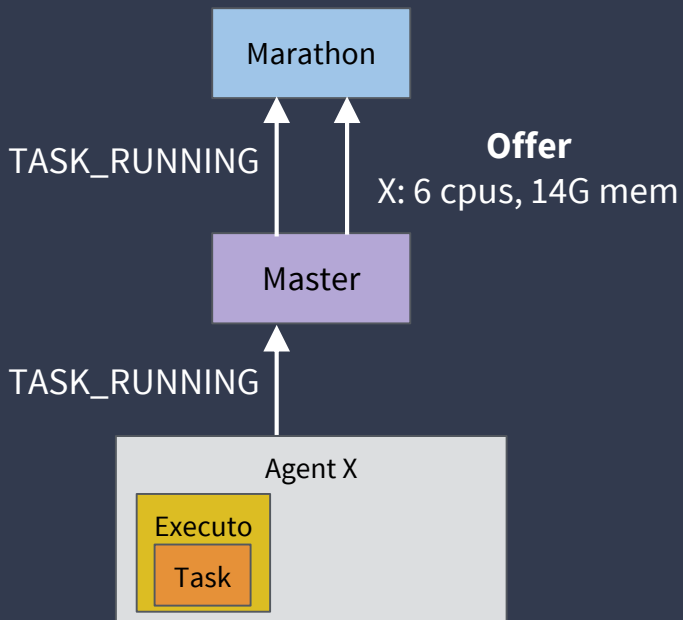
# 用例分析: Marathon+Mesos



# 用Marathon启动一个容器

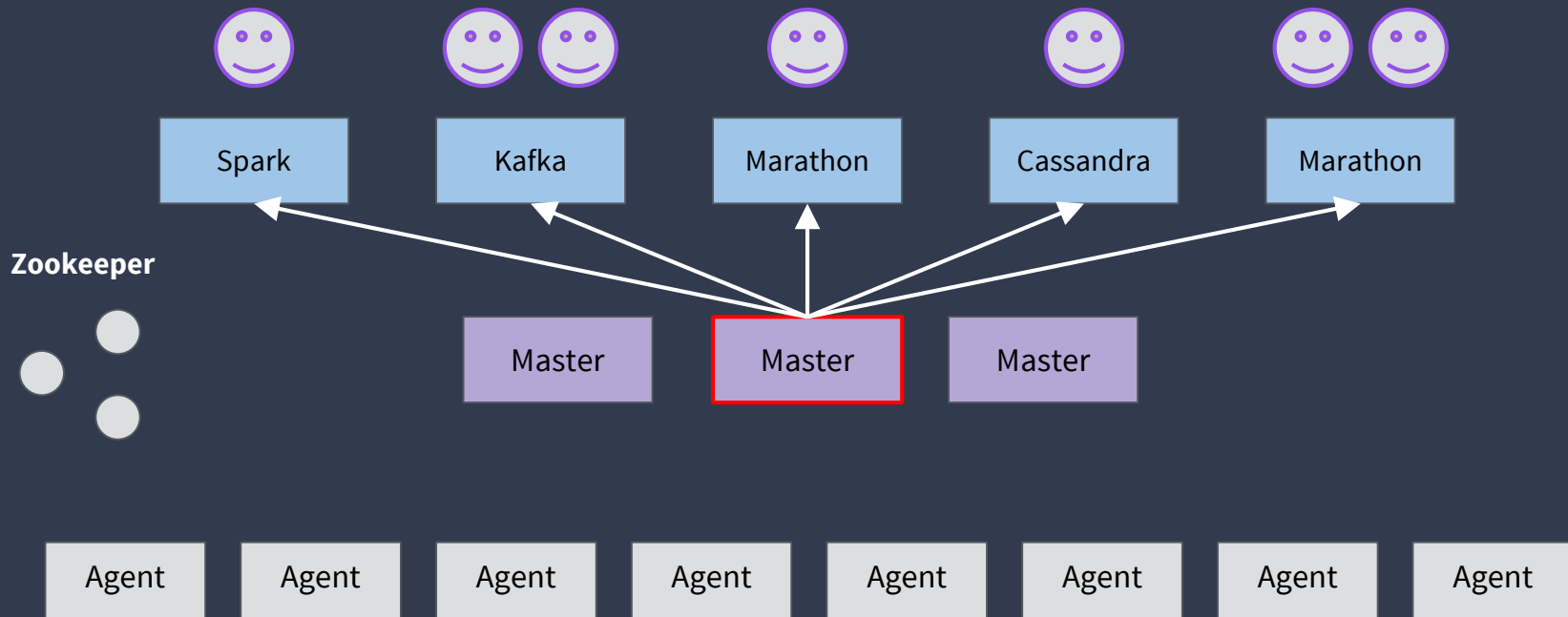


# 用Marathon启动一个容器

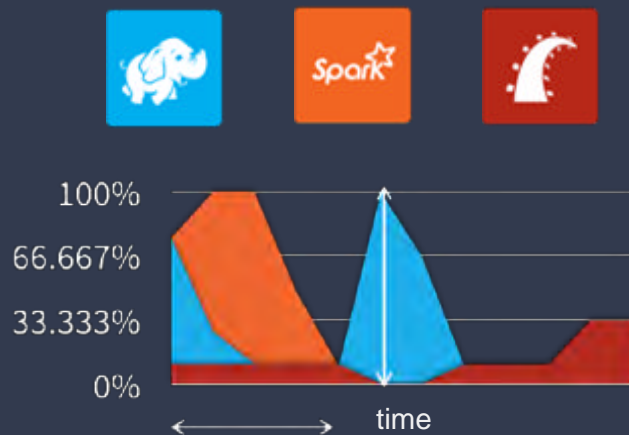
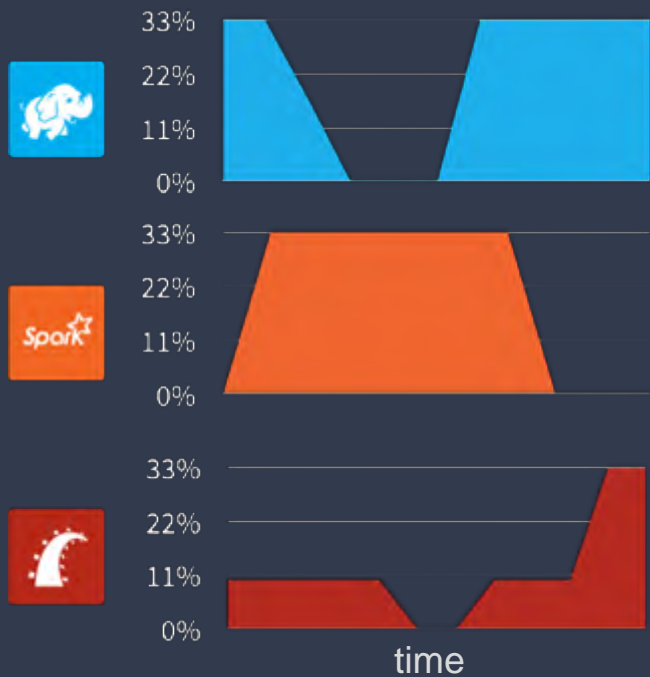




# 一个典型的Mesos集群架构

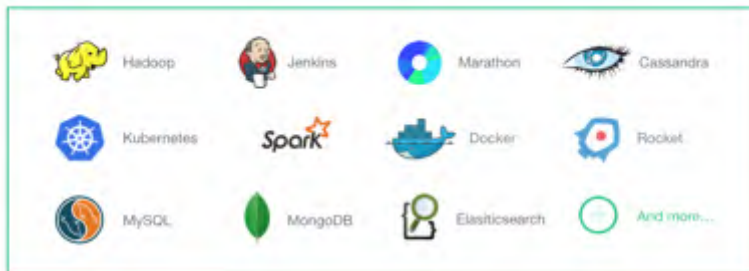


# Mesos可以帮助提升集群资源利用效率



# Mesosphere DC/OS - Mesos的一个发行版

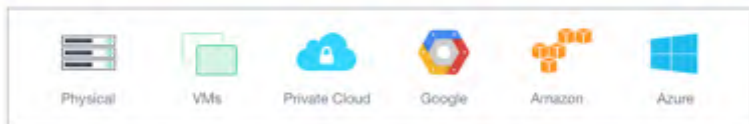
## 容器和服务



## Mesosphere DC/OS



## 任意基础架构 公有云 私有云



- 只有内核是不够的！
- DC/OS: Mesos最好的发行版
  - 命令行和图形界面
  - 包裹管理
  - 服务发现
  - 负载均衡
  - 调试，日志和监控
  - 用户管理，安全性加强
  - Framework SDK
- 是开源的！

---

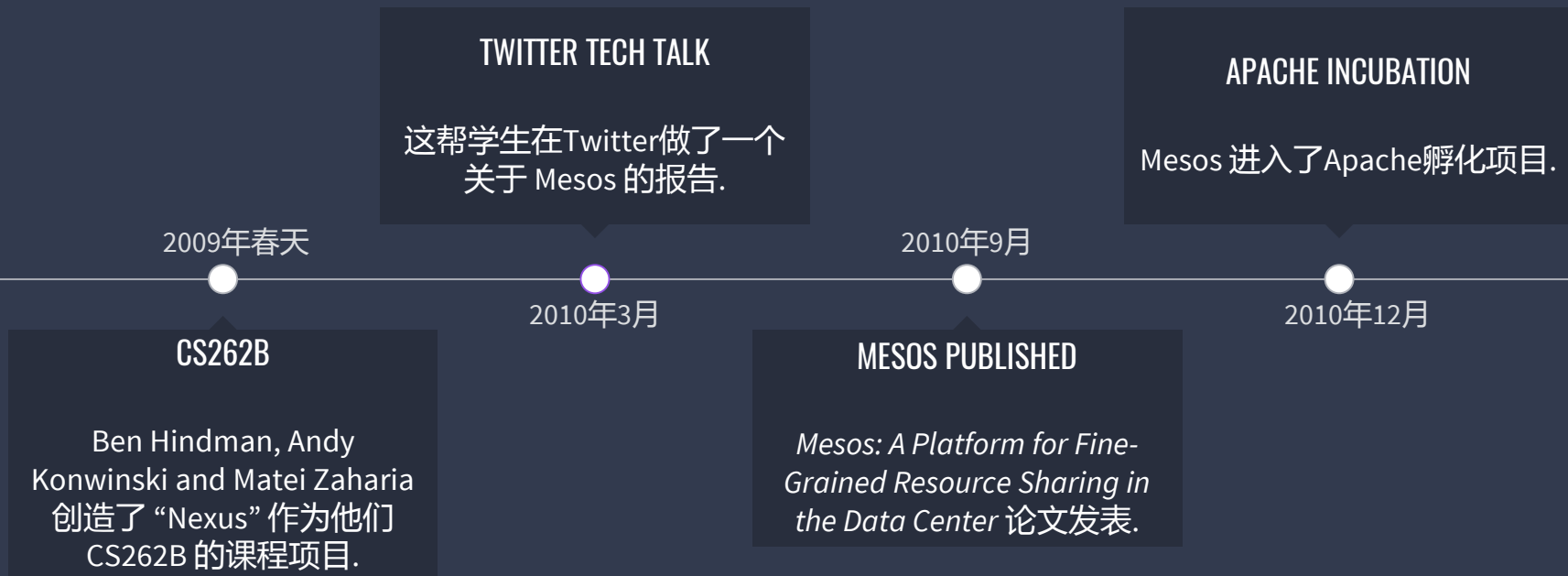
## 选择Mesos的理由

- 在生产环境中被验证过
- 支持超大规模
- 支持模块化自定义和扩展

---

在生产环境  
中被验证过

# Mesos的起源： 它从生产环境中来



# 在生产环境中被验证



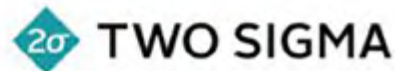
# Mesos的用户



Bloomberg



NETFLIX





---

# 支持超大规模

# Twitter



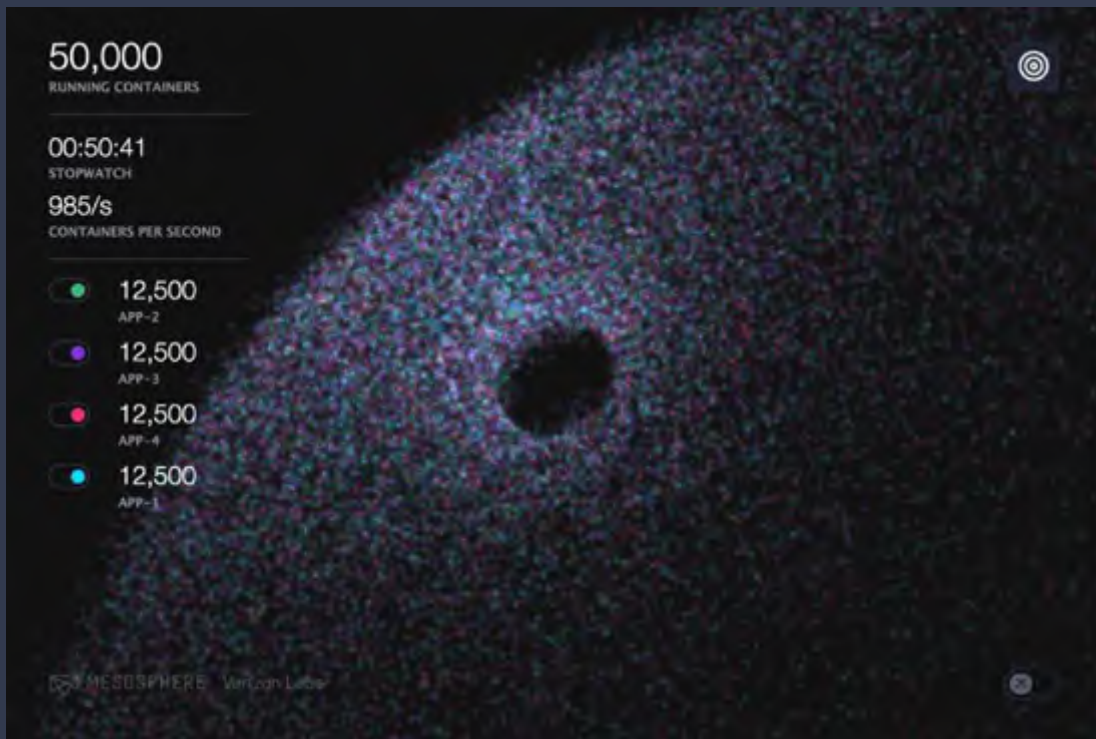
- 多个Mesos集群
- 每个Mesos集群规模
  - > 3万个节点
  - > 25万个容器

# Apple



- 整个Siri平台运行在 Mesos

# Verizon



- 50秒完成5万个容器的调度

# Mesos为什么能够支撑如此大规模部署

- 无状态的主从Master设计
  - 借鉴GFS(Google Filesystem)的设计理念
  - Agents存储运行任务的状态(分布式)
  - Master的状态可以通过Agent注册重构
- 简单的就是好的，只关心
  - 资源分配和隔离
  - 任务管理
- 用C++写的
  - 没有虚拟机
  - 没有垃圾收集问题

## 在选型中，这些意味着什么？

- 知道Mesos可以支持到像Twitter和Apple那样的规模
  - 功能容易添加, 支持大规模并不那么容易
- 免费的软件测试和验证
  - 想象一个有3万节点的真实生产测试环境
- 稳定的接口，对于向后兼容性足够重视
  - We don't want to break their production environment

---

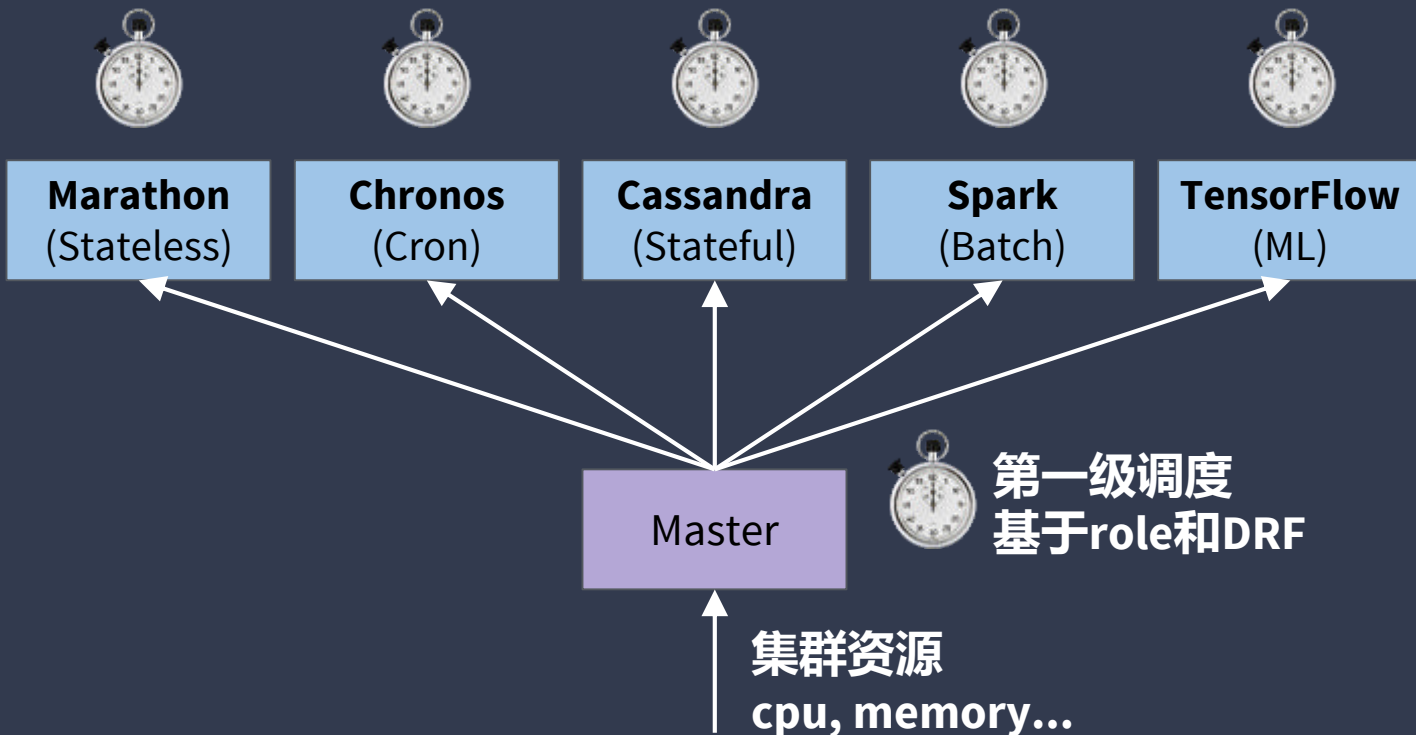
# 支持模块化自 定义和扩展

# Mesos的理念: 只关注核心抽象

- 其他功能通过模块化扩展，因为每个公司的环境和要求都不同
  - 调度算法
  - 服务发现
  - 容器格式
  - 容器网络
  - 容器存储
  - 特殊硬件和加速器 (e.g., GPU, FPGA)



# 自定义可扩展的调度 -- Mesos的二级调度



# 自定义服务发现

- 每个公司的服务发现实现并不相同
  - 基于DNS，用于小规模部署
  - 基于ZK/Etcd/Chubby和统一客户端，用于大规模部署 (Twitter, Google)
  - 自定义的服务发现实现
- Mesos本身不提供服务发现功能，但提供功能支持自定义
  - DiscoveryInfo - 每个任务可以设置一个服务发现的标签
  - Mesos提供接口使得服务发现系统能够获得这些标签

# Mesos支持各种容器插件标准

- 容器镜像
  - OCI (Open Container Initiative)
  - Docker
  - Appc
- 容器网络
  - Container Network Interface (CNI)
- 容器存储
  - Docker卷插件 (DVEDI)
  - Container Storage Interface (CSI) (进行时)

# 容器网络标准: CNI

- Mesos从1.0开始支持 Container Network Interface (CNI)
  - 一个简单，低耦合的容器网络标准
  - 大部分的网络厂商已经支持这个标准



PROJECT  
CALICO



weave



- 即将加入CNCF！

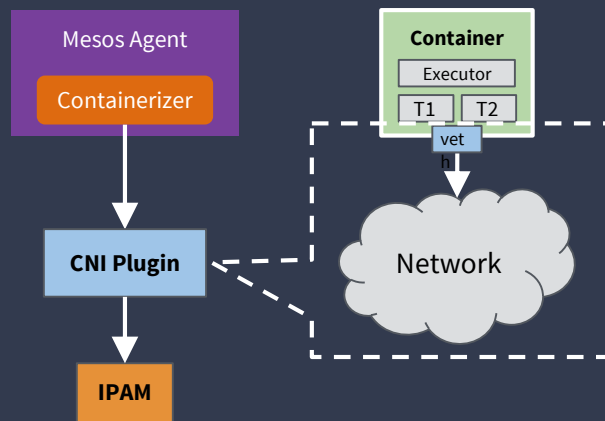


# 容器网络标准: CNI

- 由CoreOS发起

<https://github.com/container networking/cni>

- 介于容器管理器和网络插件之间的一个简单的基于JSON的协议
  - 通过命令行的方式调用插件
  - ADD: 加入网络
  - DEL: 退出网络



# Mesos为什么选择CNI

- 比Docker CNM (libnetwork)简单且低耦合
- 在其他容器社区也被广泛使用 (e.g., Kubernetes, CF)
- 被主流网络提供商支持
- 在容器和网络管理之间有清晰的界限
- 容器IP地址管理由单独的可扩展接口

# 容器存储支持

- Mesos从1.0开始支持Docker卷插件
  - Docker卷插件定义了Docker和存储提供商之间的接口
  - [https://docs.docker.com/engine/extend/plugins\\_volume/](https://docs.docker.com/engine/extend/plugins_volume/)
- 很多厂商支持这个规范
  - Glusterfs
  - Ceph
  - Convoy
  - Flocker
  - Rexray

# 一个新的容器存储标准: CSI



CSI: Towards a more universal storage interface for containers

Benjamin Hindman  
March 30, 2017

Like 9 Share Tweet LinkedIn Share 143 YouTube Vote 3 G+ 1

Standard interfaces can provide many benefits. For technology vendors, they facilitate collaboration, enable interoperability, and save time and resources from building one-off integrations. For customers, standard interfaces accelerate technology adoption, simplify the user experience, and enable choice. The success of the [Container Networking Interface](#) got us to think: can we do the same for container storage?

<https://mesosphere.com/blog/2017/03/30/csi-towards-universal-storage-interface-for-containers/>

## MESOS-7235: Improve Storage Support using Resource Provider and CSI



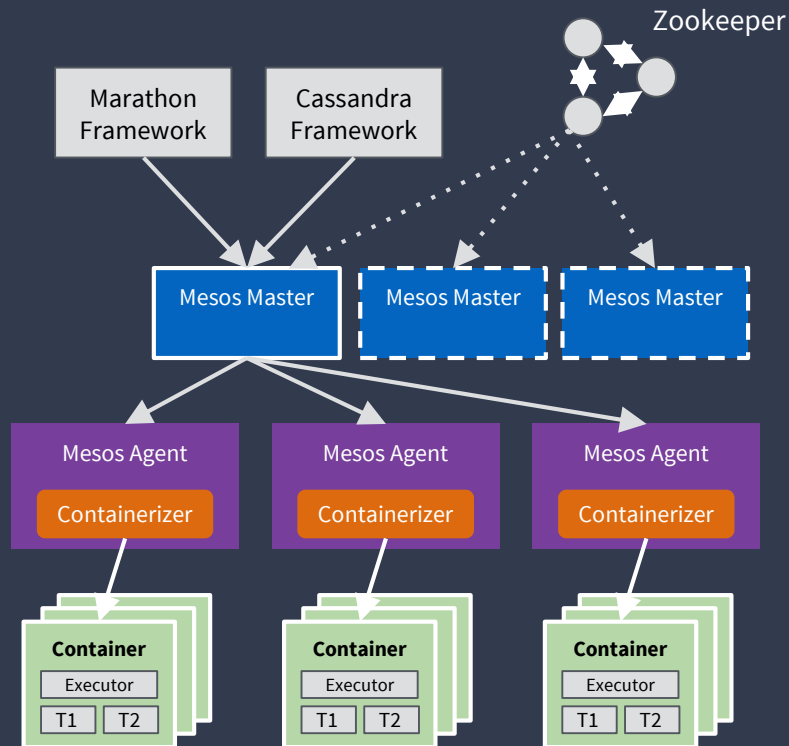
# 为什么要推动CSI

- Docker卷插件存在不少设计缺陷
  - 不支持块设备访问
  - 不支持远程分离(detach)
  - Mount/Unmount不是幂等(idempotent)的
- 其他容器存储标准也存在不少问题
  - Kubernetes Flex Volume
  - Libstorage
- 并没有一个现有标准能够解决大部分问题

# 容器镜像支持

- Mesos支持多种容器镜像格式，且可方便扩展
  - Docker
  - Appc
  - OCI (正在开发)
  - [CVMFS](#) (实验性)
  - 宿主机文件系统 + tar/jar包

# 容器管理接口(Containerizer)



Containerizer

介于Agent和容器之间的组件

启动更新销毁容器

负责容器之间的隔离

汇报容器状态和信息

## 目前支持的容器管理器

- DockerContainerizer
  - 利用Docker daemon
  - 将来可能会利用containerd

**可以同时使用！！！！**

- MesosContainerizer
  - 使用标准OS特性 (e.g., cgroups, namespaces)
  - 模块化设计，支持自定义和扩展

---

# Mesos最新功能介绍

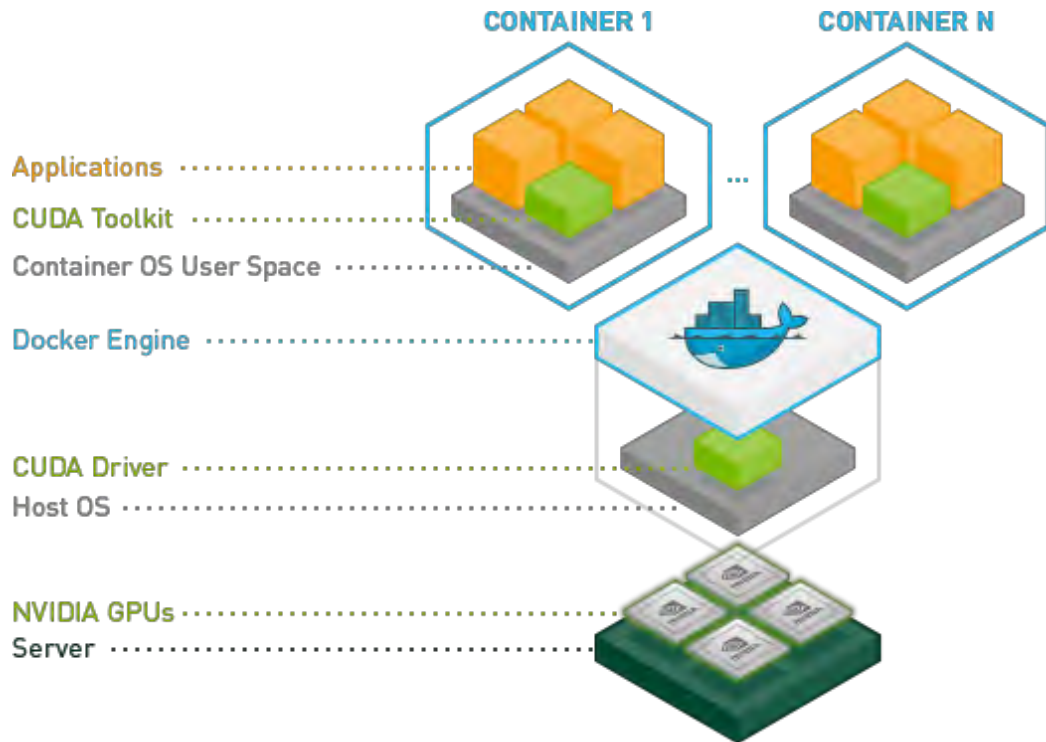
# Mesos最新功能介绍

- 原生Docker镜像支持
- Nvidia GPU支持
- 多容器统一调度(Pod)的支持
- 远程容器调试支持
- 支持网络阻隔时的多种策略
- 多个role和阶级式role的支持

# Nvidia GPU支持

## Nvidia-docker

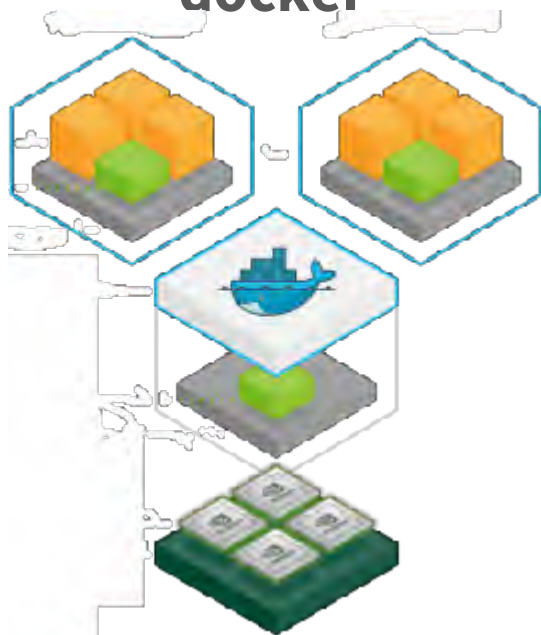
Nvidia开发的一个工具  
使得开发人员能够更容易的在Docker容器里面  
使用GPU



Source: <https://devblogs.nvidia.com/parallelforall/nvidia-docker-gpu-server-application-deployment-made-easy/>

# Nvidia GPU支持

## 本地开发 nvidia-docker

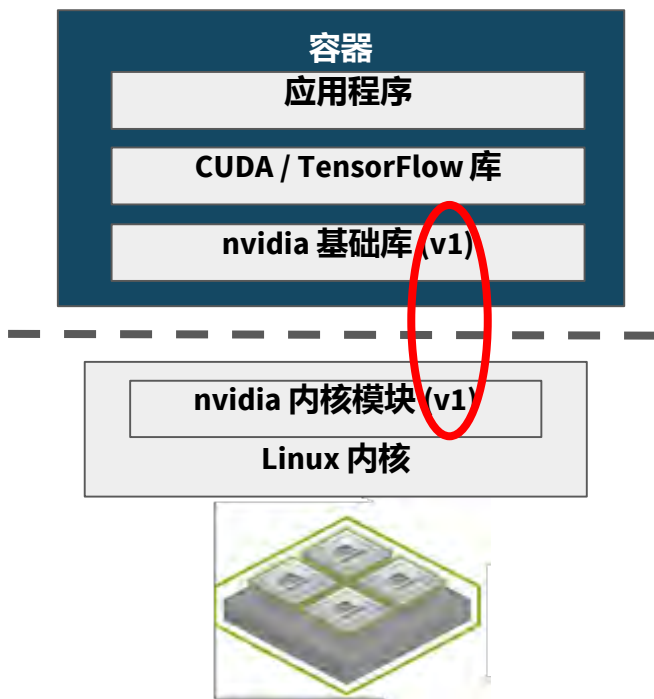


## 生产环境部署 Mesos





# nvidia-docker到底做了些什么



它在Docker客户端上包了一层

寻找Docker镜像是否有一下这个标签

```
com.nvidia.volumes.needed = nvidia_driver
```

如果有，就自动把nvidia的基础库(nvidia\_XXX.XX)通过卷的方式加载到容器里面

```
/usr/local/nvidia
```

# Mesos的GPU支持

- 模拟了nvidia-docker的功能
  - 支持docker容器的nvidia标签
  - 挂载nvidia基础库到容器内的 /usr/local/nvidia
  - 自动发现宿主机上的GPU，并且注入到容器之中
- 容器之间GPU设备的隔离
  - 只使用nvidia-docker的话无法达到
- 和Mesos的资源管理完美结合
  - GPU和CPU一样，是一种资源

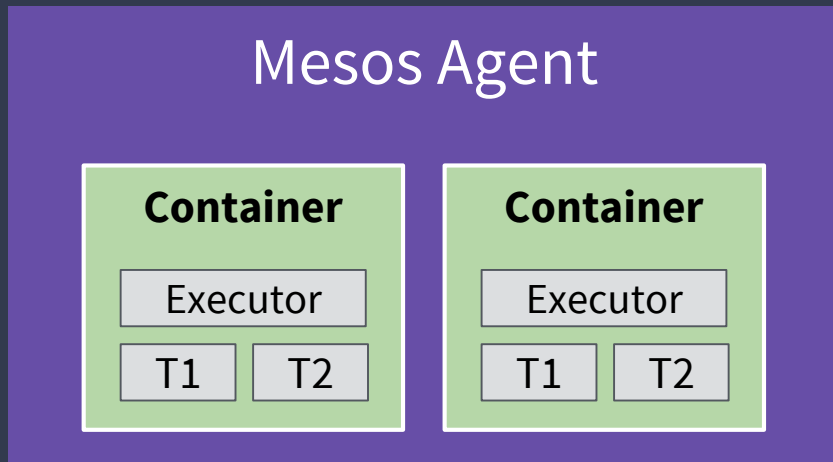
# 多容器统一调度(Pod)的支持

## Frameworks启动Tasks

- Tasks指定executor

## Executors运行tasks

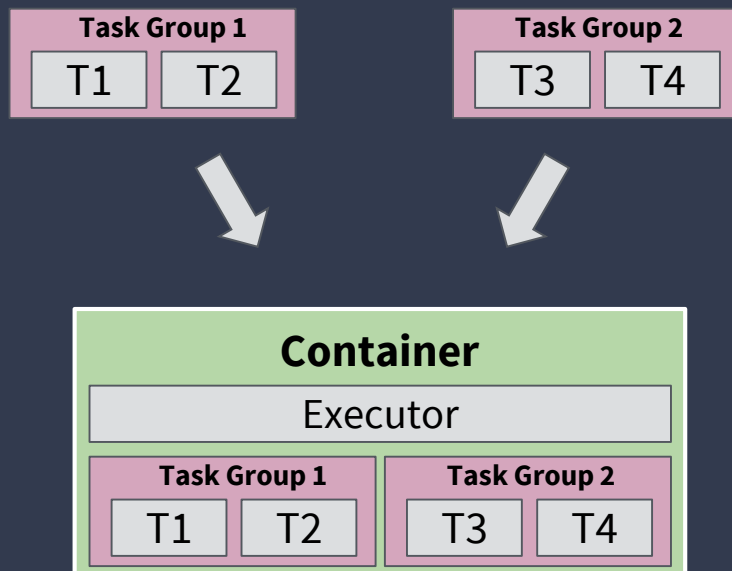
- 每个executor可以有多个tasks
- 每个executor是一个容器



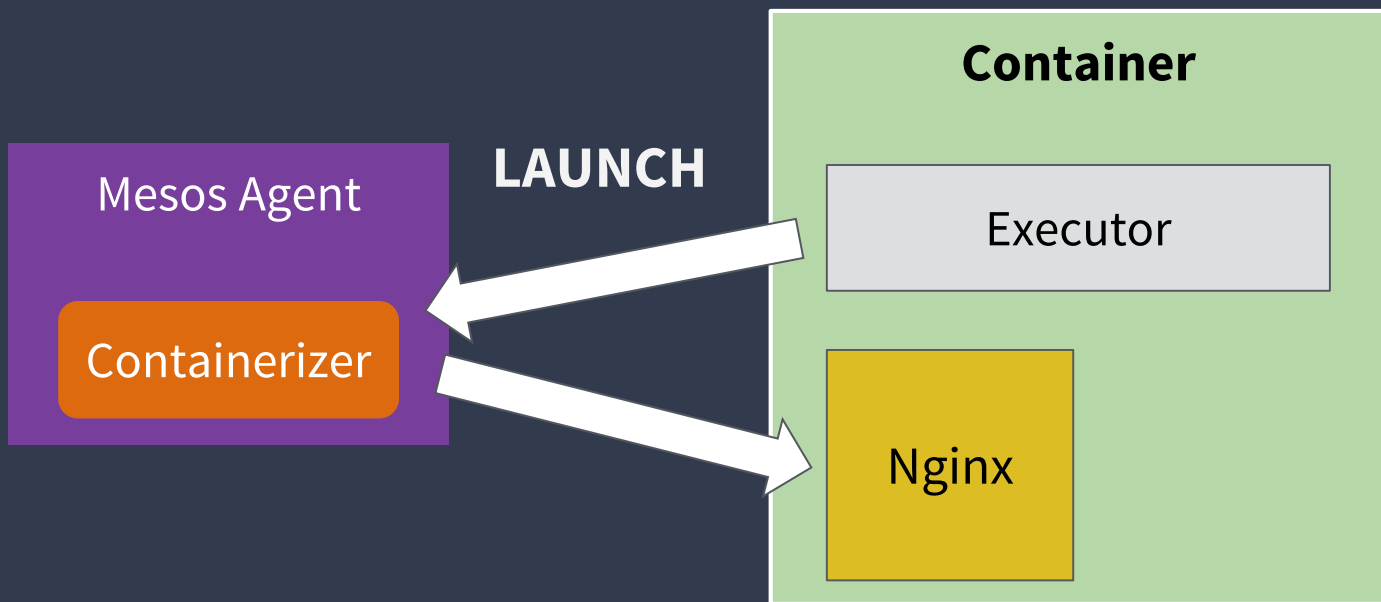
# 新增的功能 #1: 任务组(Task Group)

Framework现在可以启动一  
组tasks (TaskGroup)

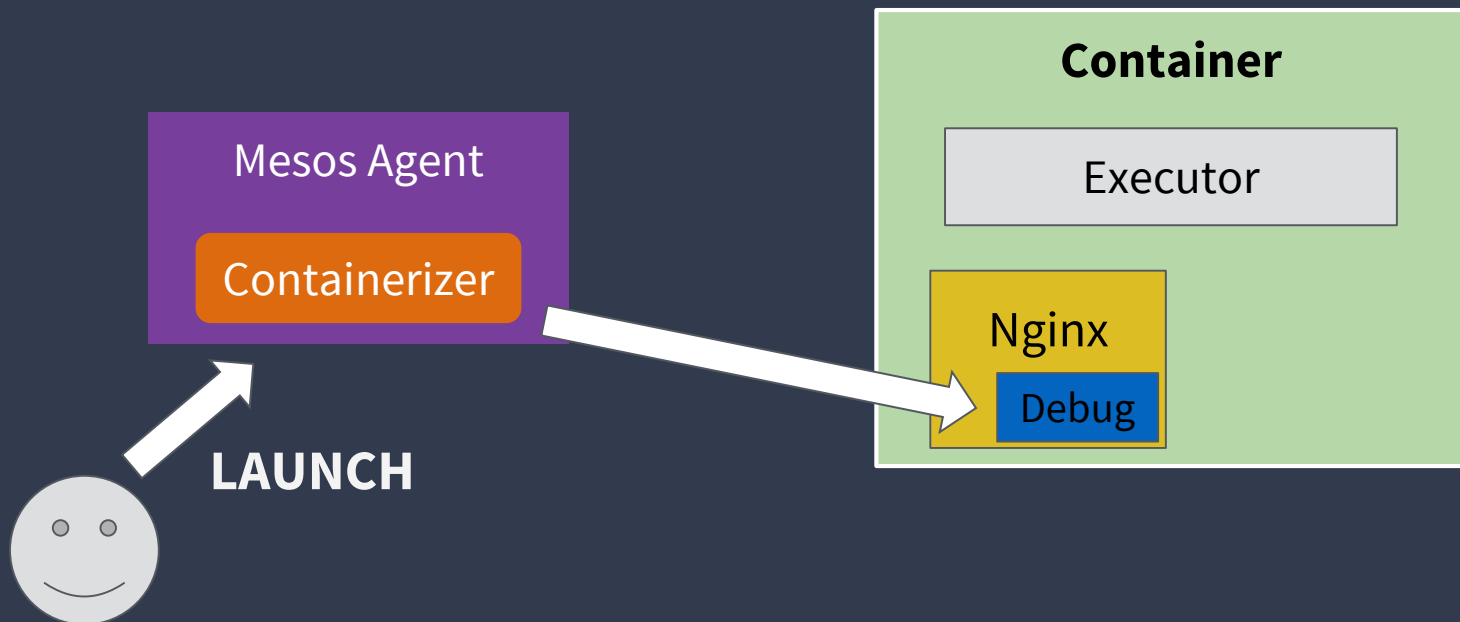
Executor可以原子性地  
(atomically)收到TaskGroup



## 新增的功能 #2: 容器嵌套(Nested Container)



# 支持任意层数的嵌套



## 多个role和阶级式role的支持

### Mesos是一个资源管理系统

- 在用户之间提供资源的共享
- 给某些用户提供最低资源保障
- 提供资源统计
  - 有多少资源已经被分配了，多少被利用了？
  - 每个用户有资源统计

## 多个role和阶级式role的支持

### Mesos是一个资源管理系统

- 在用户role之间提供资源的共享
- 给某些用户role提供最低资源保障
- 提供资源统计
  - 有多少资源已经被分配了，多少被利用了？
  - 每个用户role有资源统计

**“Role”：资源消费者**

**(e.g. 用户, 单位, 服务账号, etc).**



# 基于Role的资源分配



28 CPUs  
56GB mem  
100GB disk

14 CPUs  
28GB mem  
50GB disk

14 CPUs  
48GB mem  
800GB disk

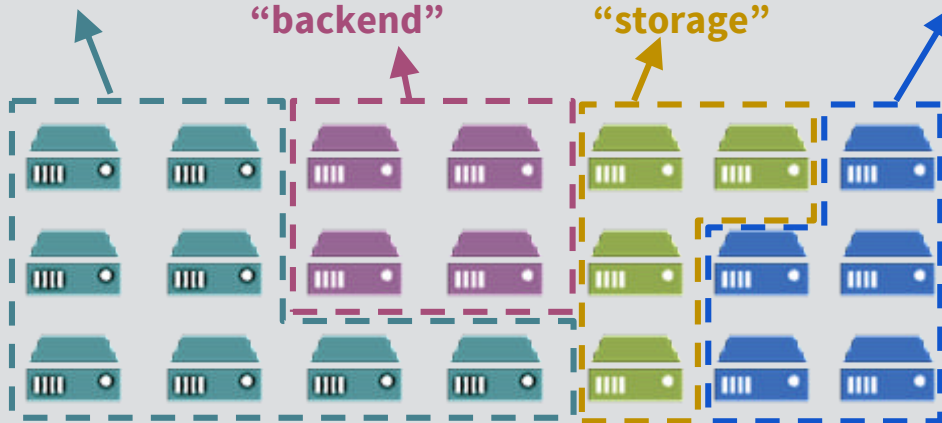
72 CPUs  
128GB mem  
100GB disk

“frontend”

“backend”

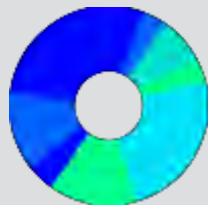
“storage”

“analytics”

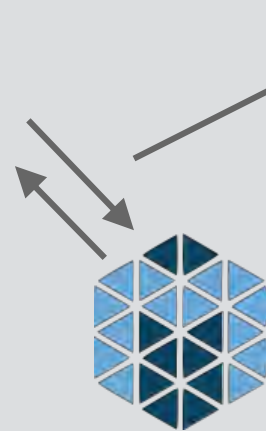


# Framework 可以支持 多个role

“backend”  
“frontend” “reporting”



MARATHON



Framework现在可以有多个role

- Quota
- Reservation

“frontend”

“backend”

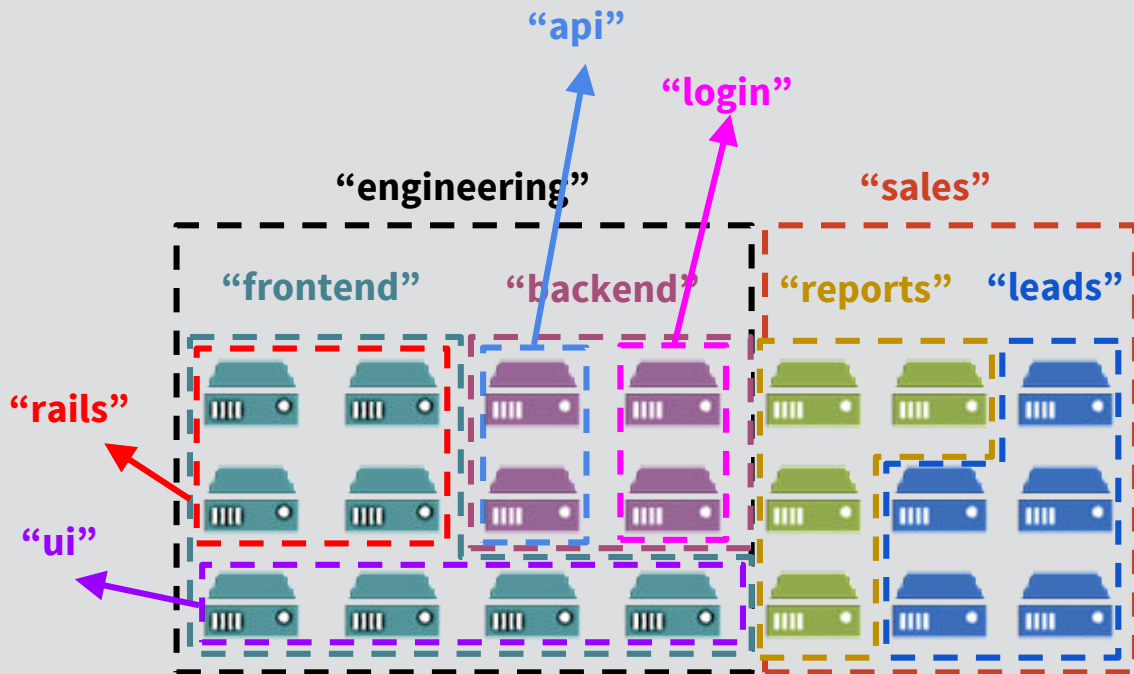
“storage”...



# 阶梯式role的支持

阶梯式role的支持，大部分组织结构都是阶梯式的:

- `engineering/frontend/ui`
- `engineering/backend/api`



# 总结

Mesos是一个很强大且稳定的数据  
中心操作系统内核

在生产环境中被验证过  
支持超大规模  
支持模块化自定义和扩展

最近在保持稳定性的同时又加入  
了很多新功能

