Serverless初探及展望

李华卿 2017.6

- 1.Serverless简介
- 2.Serverless实现原理
- 3.Serverless实战
- 4.Serverless展望

Serverless简介

```java
public class Solution {
    public TreeNode invertTree(TreeNode root) {
        if(root == null){
            return null;
        }
        if(root.left!=null){
            invertTree(root.left);
        }
        if(root.right!=null){
            invertTree(root.right);
        }
        if(root.left!=null&&root.right!=null){
            TreeNode tmp = root.left;
            root.left = root.right;
            root.right =tmp;
        }else if(root.left == null){
            root.left = root.right;
            root.right = null;

        }else if(root.right == null){
            root.right = root.left;
            root.left = null;

        }
         return root;
    }
}
```

Custom Testcase

Contribute Testcase

Shortcut: Command + '

Run Code | Submit Solution

Run Code Status: Judging

Run Code Result:

Your input

[]

Your answer

Judging

Expected answer

Pending

Show Diff

Note: is Run Code inconsistent with Submit Solution? If you are using global variables or C/C++, check this out.

- 传统建站: 购买刀片机、装机(sdk,db,server...)、托管IDC、部署应用
- 目前建站: 购买云主机、装机、部署应用
- 不久的未来?: 开发 -> 生效

- 请求波动怎么办？ -> 添加/删除机器
- 计算资源夜间过剩怎么办？ -> 挖矿？

Serverless指这样的应用架构，服务逻辑由应用开发者实现，与传统架构不同在于，他们运行于无状态的容器中。应用可以由事件触发，容器完全被第三方管理。

公有云Serverless解决方案：

- AWS Lambda
- Microsoft Azure Functions
- Google Cloud Functions

热门Serverless框架：

- Fission　基于Kubernetes
- Funktion　基于Kubernetes
- IronFunctions 基于Kubernetes

Serverless原理

AWS is pretty sure not to release anything about the inner most working of it anytime.

Linux Container/ Docker

计算资源 CPU、内存

容器

运行库

字节码/脚本

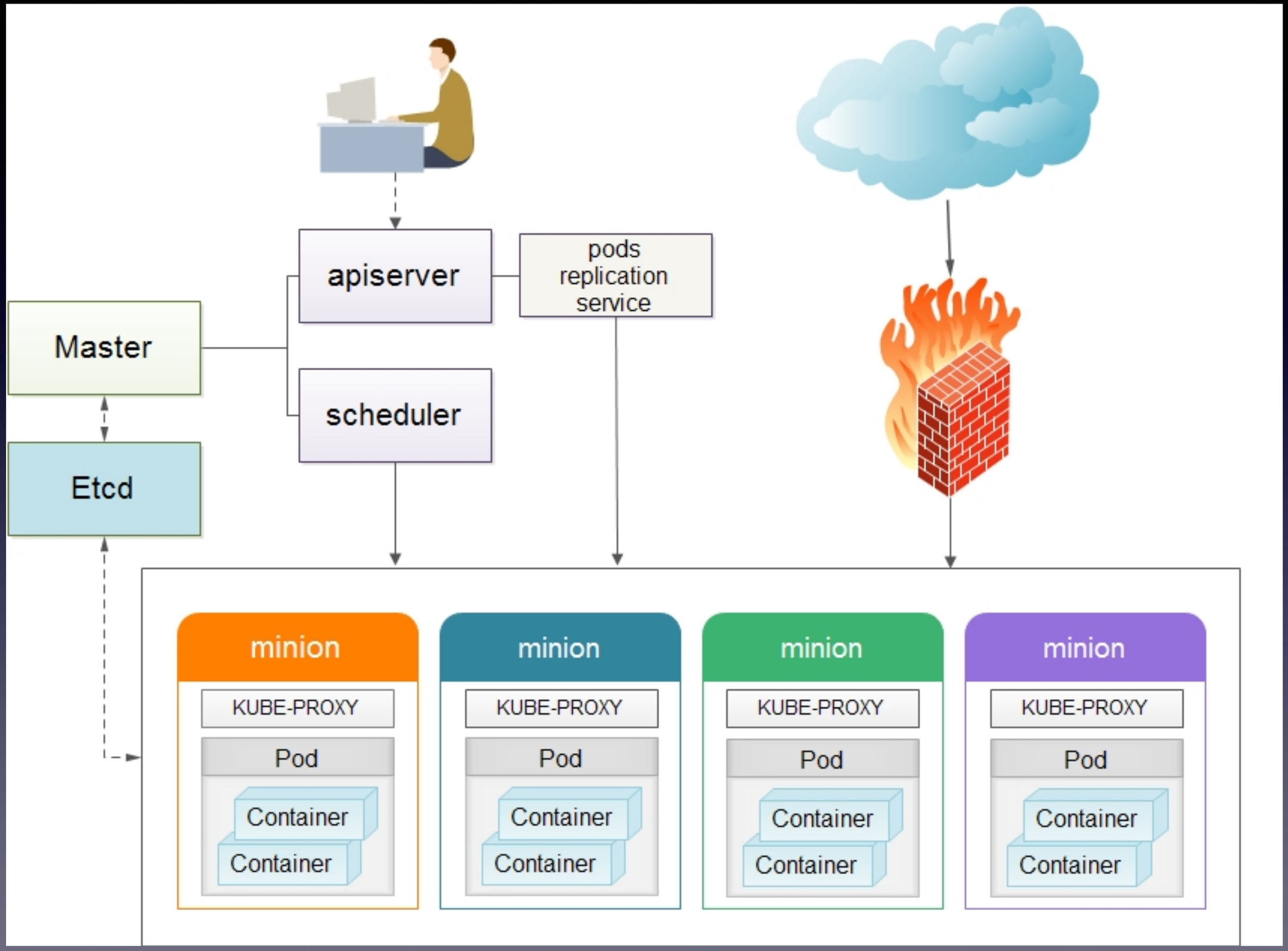
容器创建/销毁

容器状态监控/管理

调度 AWS-Dispatcher/Kubernetes/ swarm

请求接收&分发

容器弹性自适应引擎


apiserver
pods
replication
service
Master
scheduler
Etcd
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container

Request(HTTP API or Any Trigger)

Init or Dispatch Container

Execute function

Return result retry on error

self-adaption

MQ、DB

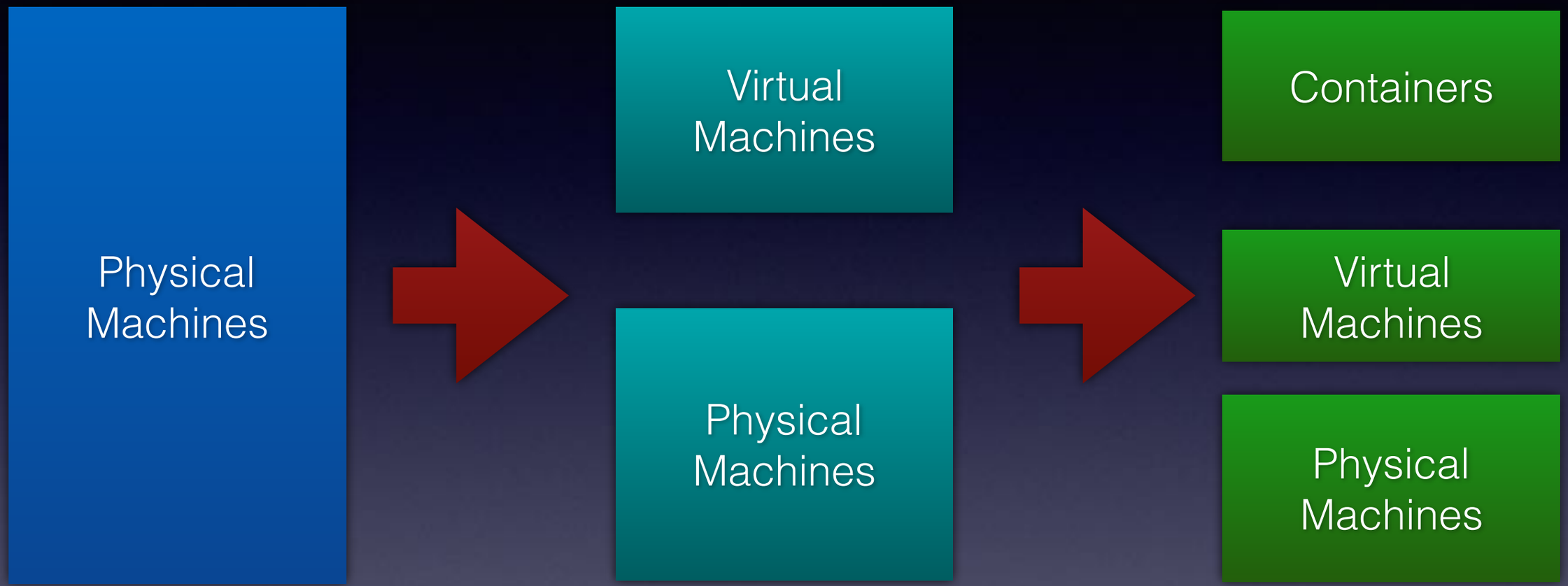
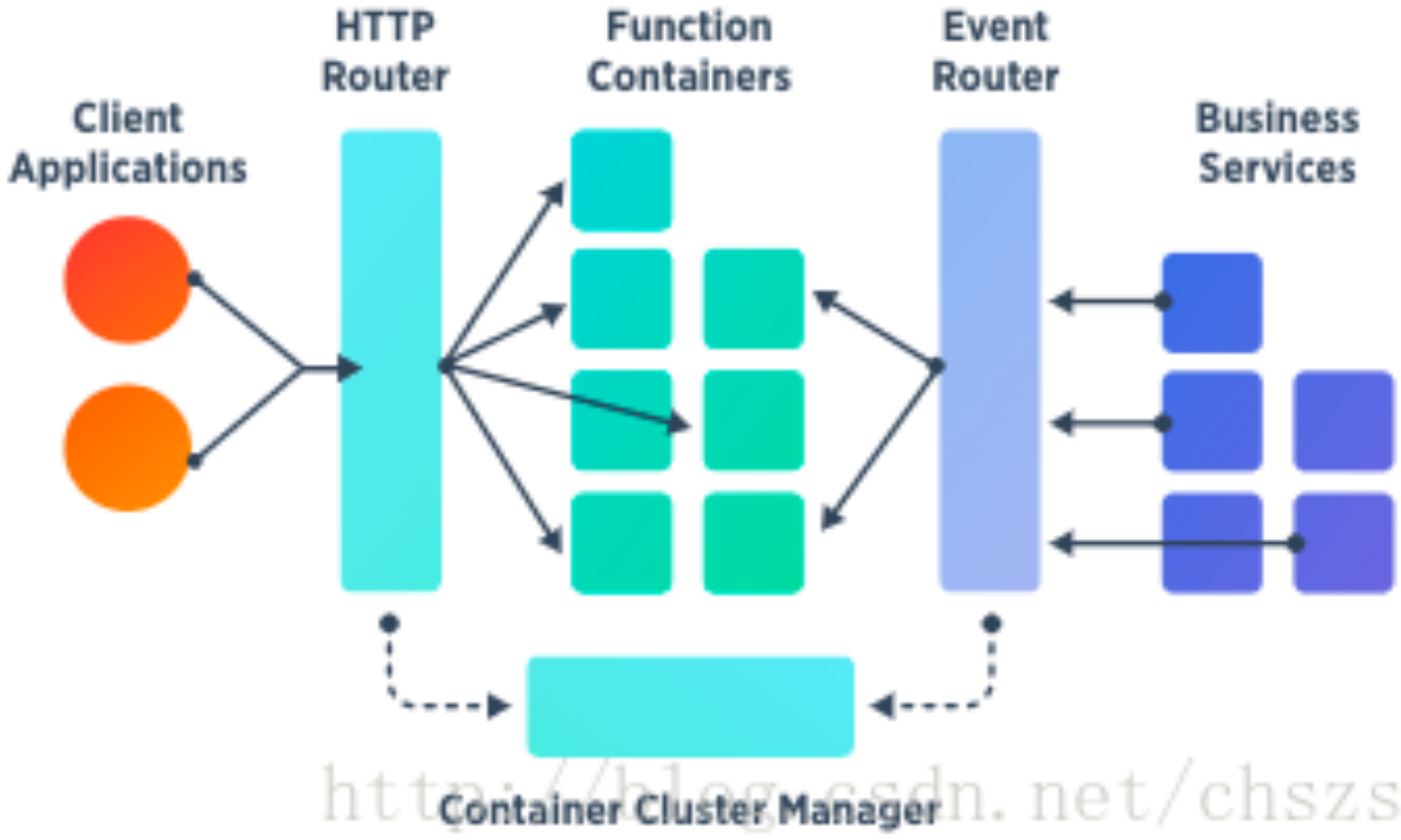

Physical Machines
Virtual Machines
Physical Machines
Containers
Virtual Machines
Physical Machines

FIGURE 3: SERVERLESS ARCHITECTURE OVERVIEW



- 弹性扩容
- 按需计费、节约成本
- 专注应用
- 微服务化
- FaaS/CaaS
- ……

· 在定义功能的范围时应遵循单一责任的原则。

· 通过优化功能来实现以毫秒为单位的执行效率。

· 坚持无状态的协议，以能够在功能上无缝地扩展。

· 使用外部的服务来实现服务的发现、状态和缓存管理。

· 使用环境变量来读取配置而非不依赖于文件。

· 鉴于容器被设计为是短暂的，我们应该避免使用文件系统去保持数据的持久性。

Serverless实战

Serverless展望

- 微服务 ->FaaS
- API层薄化
- 分层治理（RD/SRE解耦）
- 去中间件化
- 高可用弹性
- 培训成本降低，易上手

- 功能拆分过细
- 略强于微服务的分布式事物管理
- 服务依赖增加