

基础设施服务的微服务化

王渊命 @jolestar



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

个人介绍

- 青云QingCloud 架构师，负责容器平台
- 关注领域：容器，分布式系统，研发工具
- 擅长语言：Java / Go / Python
- 个人博客: <http://jolestar.com>



SegmentFault
Developer Conference



QINGCLOUD 青云

多微的服务才能叫微服务？



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

“In short, the microservice architectural style is an approach to developing a single application as a suite of **small services**, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and **independently deployable** by **fully automated deployment machinery**. ”

–James Lewis and Martin Fowler



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

微服务 (Microservice)

拆

- 更小的服务粒度
- 远程调用解决依赖

合

- 集群化设计思路 (分布式系统)
- 服务间协作机制 (自动化)



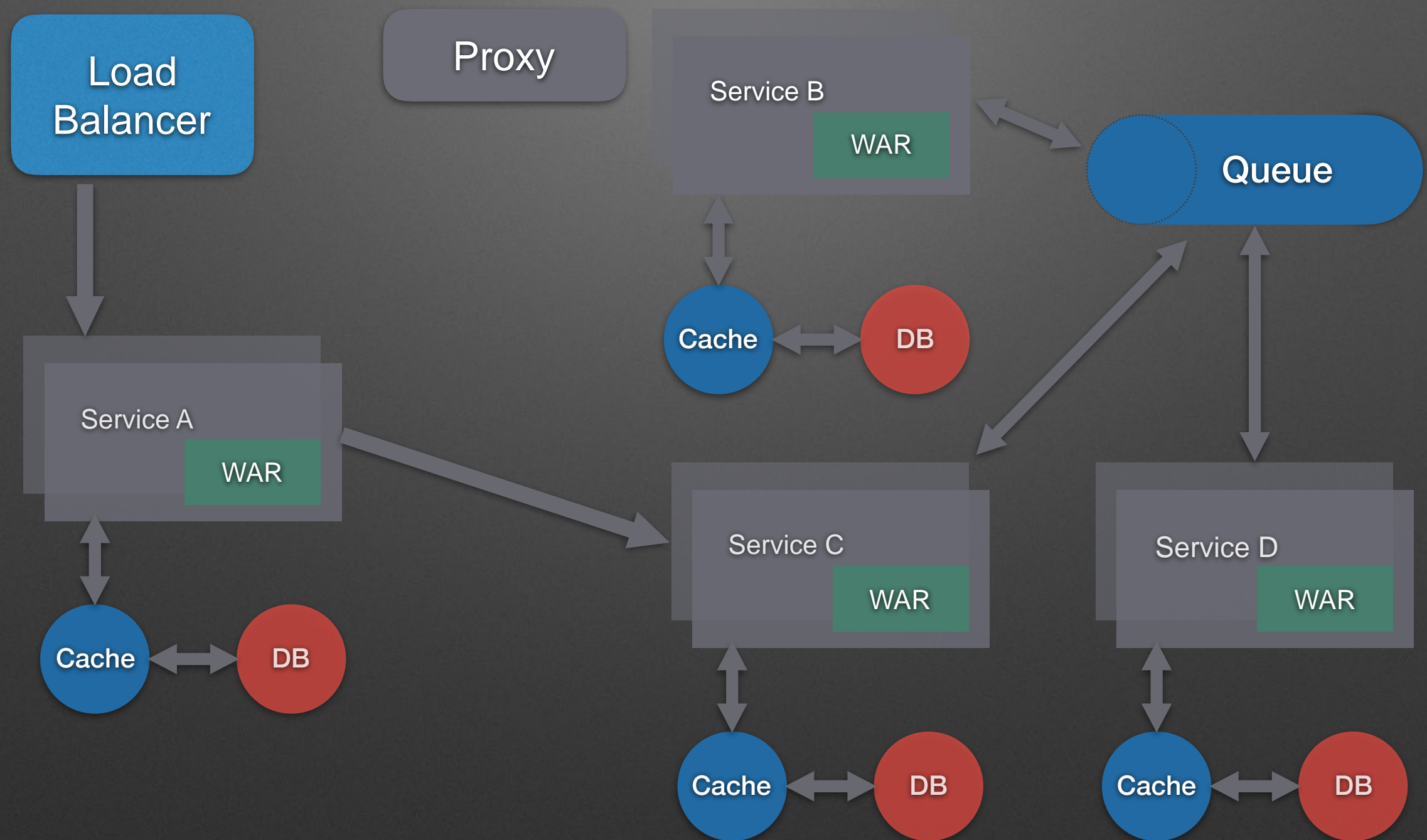
SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

微服务的一个例子



微服务的基础设施服务

- 负载均衡 (HAProxy, Nginx)
- 数据库 (MySQL)
- 缓存 (Memcached, Redis)
- 队列 (Kafka)
- NoSQL (Elasticsearch, MongoDB)
- 服务发现 (Zookeeper, Etcd, Consul)



SegmentFault
Developer Conference



QINGCLOUD 青云

基础设施服务是否需要微服务化？



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

基础设施服务当前的主要解决方案

- 托管给基础设施部门，由基础设施部门管理和运维
- 托管给云厂商，使用云厂商的基础设施服务



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

问题

- 开发测试流程中的基础设施服务如何自动化部署？
- 基础设施服务迁移、伸缩、故障恢复时应用如何感知？



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

微服务的要求

- 集群化设计
- 自动化部署/伸缩



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

基础设施服务属于微服务系统中的一部分，
需要和业务服务互相感知， 需要被微服务化。



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

基础设施服务如何微服务化？



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

难点

- 基础设施服务种类多样，配置异构
- 基础设施服务集群机制多样



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

案例-Zookeeper

zoo.cfg

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/data/zookeeper
clientPort=2181
maxClientCnxns=1000
server.1=192.168.1.11:2888:3888
server.2=192.168.1.12:2888:3888
server.3=192.168.1.13:2888:3888
```

myid

192.168.1.11	1
<hr/>	
192.168.1.12	2
<hr/>	
192.168.1.13	3



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

案例-Haproxy

```
frontend testserver_gw
bind *:8081 ssl crt /etc/haproxy/certs/
ssl.chained.pem
mode tcp
option tcplog
default_backend testserver_gw_backend

backend testserver_gw_backend
balance roundrobin
mode tcp
server s1 192.168.1.11:8081 check
server s2 192.168.1.12:8081 check
server s3 192.168.1.13:8081 check
```



SegmentFault
Developer Conference



QINGCLOUD 青云

案例-Redis

```
./redis-trib.rb create --replicas 1 127.0.0.1:7000 127.0.0.1:7001 \  
127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005
```

```
./redis-trib.rb add-node 127.0.0.1:7006 127.0.0.1:7000
```

```
./redis-trib.rb add-node --slave 127.0.0.1:7006 127.0.0.1:7000
```

```
./redis-trib del-node 127.0.0.1:7000 `<node-id>`
```

```
./redis-trib.rb reshard --from <node-id> --to <node-id> --slots <number of slots> --  
yes <host>:<port>
```



案例-Kafka

```
zookeeper.connect=192.168.1.11:2181,192.168.1.12:2181,192.168.1.13:2181  
host.name=192.168.1.20
```



服务注册发现以及配置变更方案

- **理想：**应用内部自发现，自动配置变更
- **现实：**非侵入的第三方注册，以及配置变更



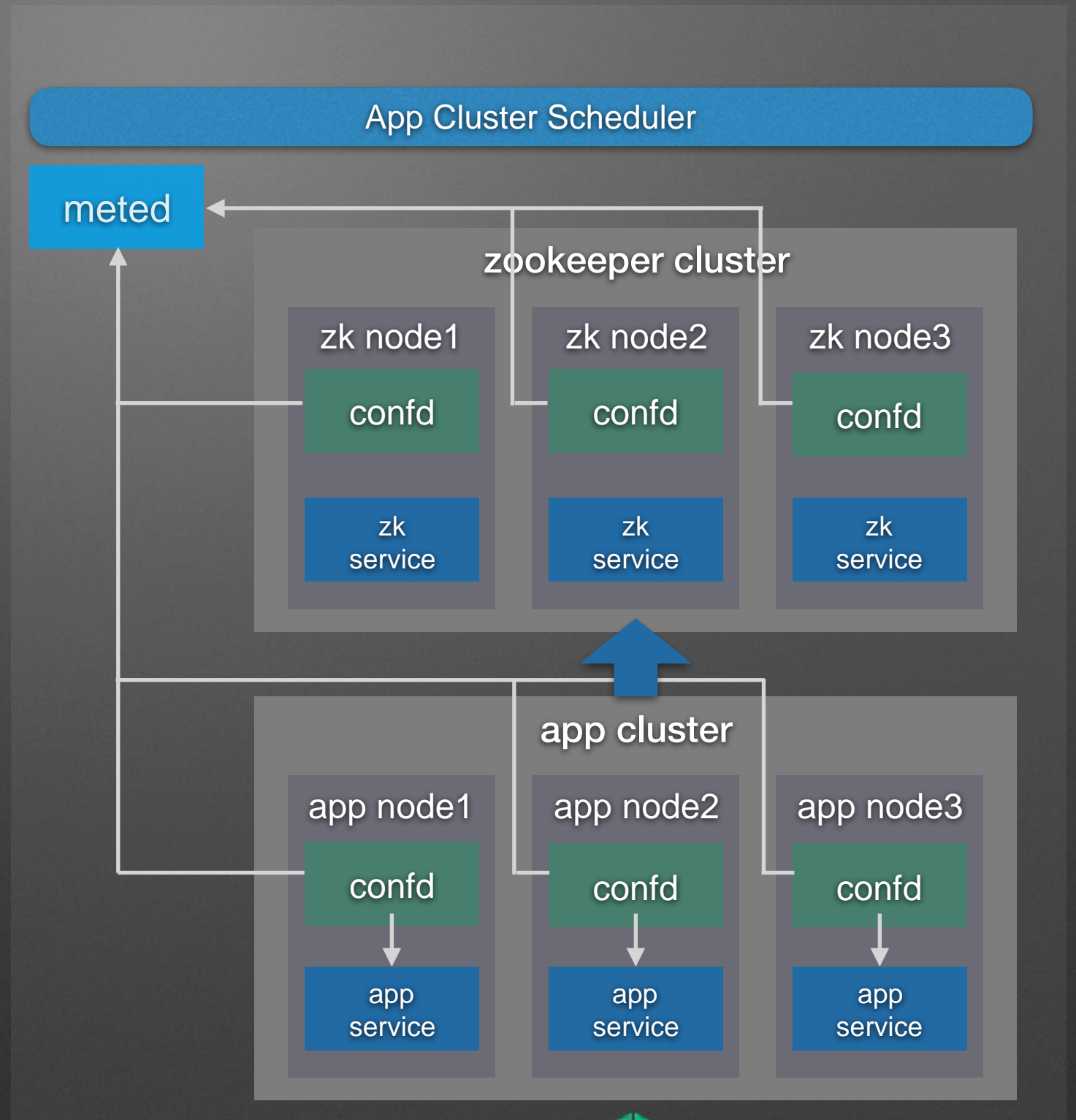
SegmentFault
Developer Conference



QINGCLOUD 青云

QingCloud 应用调度系统实践

- 调度系统注册
- etcd/metad 元信息服务
- confd 配置变更



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

etcd

- 分布式，一致性 kv 存储，提供元信息的持久化
- 提供 watch 机制，可以实现变更推送



SegmentFault
Developer Conference



QINGCLOUD 青云

metad

- 独立的元信息服务，后端对接 etcd，与 etcd数据同步
- 提供 /self 映射机制，客户端实现配置自发现



confd

- 后端对接 metad
- 通过监听 metad的变更，进行应用的配置更新，服务管理



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云


```
{
  "app_id": "app-zkv33646",
  "app_version": "1.0",
  "name": "ZK",
  "node": {
    "container": {
      "type": "kvm",
      "image": "img-zookeeper"
    },
    "instance_class": 0,
    "count": 3,
    "cpu": 1,
    "memory": 512,
    "volume": {
      "size": 3,
      "mount_point": "/data",
      "filesystem": "xfs"
    },
    "server_id_upper_bound": 255,
    "service": {
      "start": {
        "cmd": "/opt/zookeeper/bin/zkServer.sh start"
      },
      "stop": {
        "cmd": "/opt/zookeeper/bin/zkServer.sh stop"
      }
    }
  },
  "advanced_action": ["change_vxnet", "scale_horizontal"]
}
```

案例: Zookeeper



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

案例: Zookeeper

```
[template]
src = "zoo.cfg.tmpl"
dest = "/opt/zookeeper/conf/
zoo.cfg"
keys = [
    "/",
```

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/data/zookeeper
clientPort=2181
maxClientCnxns=1000
{{range $dir := lsdirent "/hosts"}}
{{$sid := printf "/hosts/%s/sid" $dir}}
{{$sip := printf "/hosts/%s/ip" $dir}}
server.{{getv $sid}}={{getv $sip}}:2888:3888
{{end}}
```

zoo.cfg.tmpl

案例: Zookeeper

```
[template]
src = "myid.tmpl"
dest = "/data/zookeeper/myid"
keys = [
    "/",
]
```

myid.tmpl

```
{{getv "/host/
```



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

Docker支持

- Node的 Container Type 支持 kvm和Docker
- Docker镜像默认启动的是 confd, 应用服务通过 confd启动



为什么不用 Docker默认的方式

- **理想：**应用的静态配置都通过环境变量，动态配置通过配置中心
- **现实：**大多数应用的配置还是配置文件



业界方案比较 - Ansible/Puppet

- 纯静态配置变更
- 服务间的依赖变更不好解决
- 没有自动故障迁移以及容灾机制



业界方案比较 - Kubernetes

- DNS,VIP 方式支持服务发现和负载均衡
- 目标是通用的容器集群
- 支持映射全局配置文件到容器，但不支持配置变更（只能静态配置）
- 动态配置以及节点发现可通过应用插件实现（比如Elasticsearch）



SegmentFault
Developer Conference



QINGCLOUD 青云

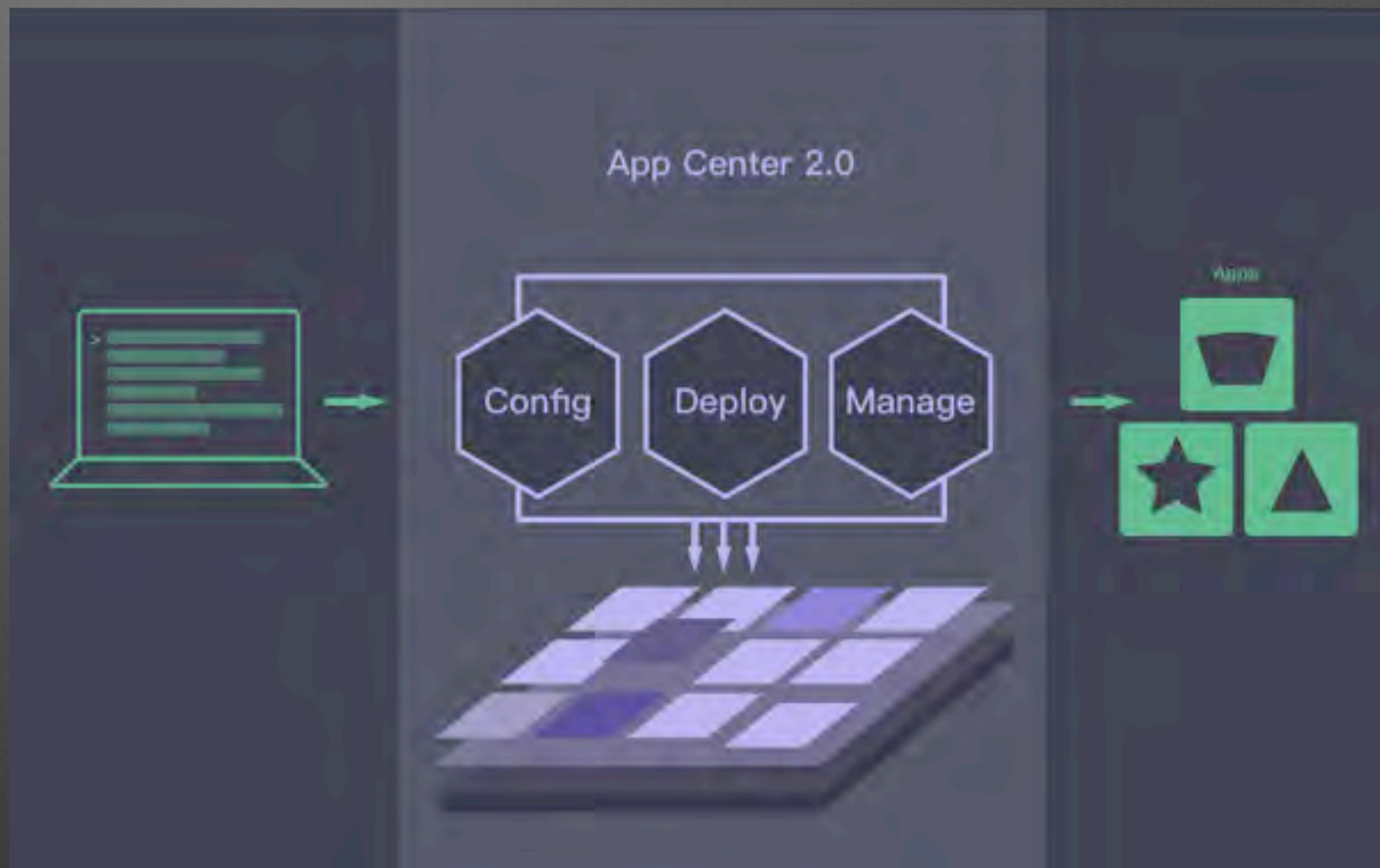
业界方案比较 - Mesos

- DNS,VIP 方式支持服务发现和负载均衡
- 目标是通用的资源调度分配系统
- 特殊的应用通过自定义 Framework实现，开发成本比较高
- 通用的容器应用，配置变更以及节点发现机制不完备



QingCloud 应用中心介绍

- 基于新的应用调度系统
- 提供企业应用标准化开发平台
- 快速将应用云化，实现应用的秒级部署和弹性伸缩
- 提供计费服务以及客服平台，让企业应用快速商业化



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

对开发者的意义

- 基础研发运维部门在企业中的角色转换
- 服务器端研发人员的春天



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云

Q&A



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 —



SFDC

SegmentFault
Developer Conference



QINGCLOUD 青云