Docker与UCloud DataBase的融合实践

吴斌炜@UCloud
2015-12-19

自我介绍

吴斌炜

UCloud 结构化存储部副经理

2011年浙大毕业后加入腾讯云平台部，

2013年加入UCloud从事云数据库和分布式缓存的研发工作，设计和研发了国内一个服务化的分布式redis系统，第一个支持Mongodb的云数据库系统

UCloud DataBase为什么选择Docker

UCloud DataBase Docker应用实践

谈谈如何更好的使用Docker

UCloud DataBase为什么选择Docker

UCloud DataBase Docker应用实践

谈谈如何更好的使用Docker

先从为什么需要云数据库说起

用户的需求

- 降低运维门槛【操作界面化、API化、自动化 完善的监控和告警】
- 提升服务的可用性和数据安全性
- 更好的性能和资源的隔离

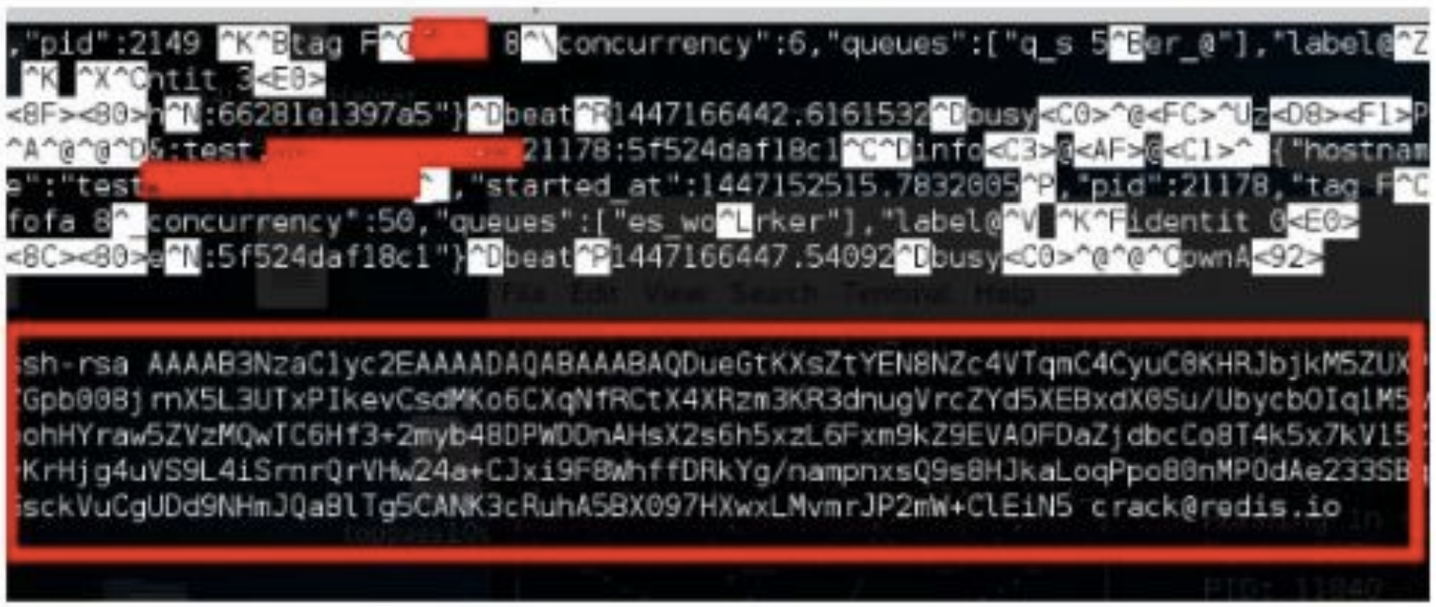
IAAS厂商的需求

- 避免数据库对宿主机io的占用
- 减少虚拟机在线迁移的代价
- 减少用户使用数据库出错的可能

云数据库运行环境的选择

物理机

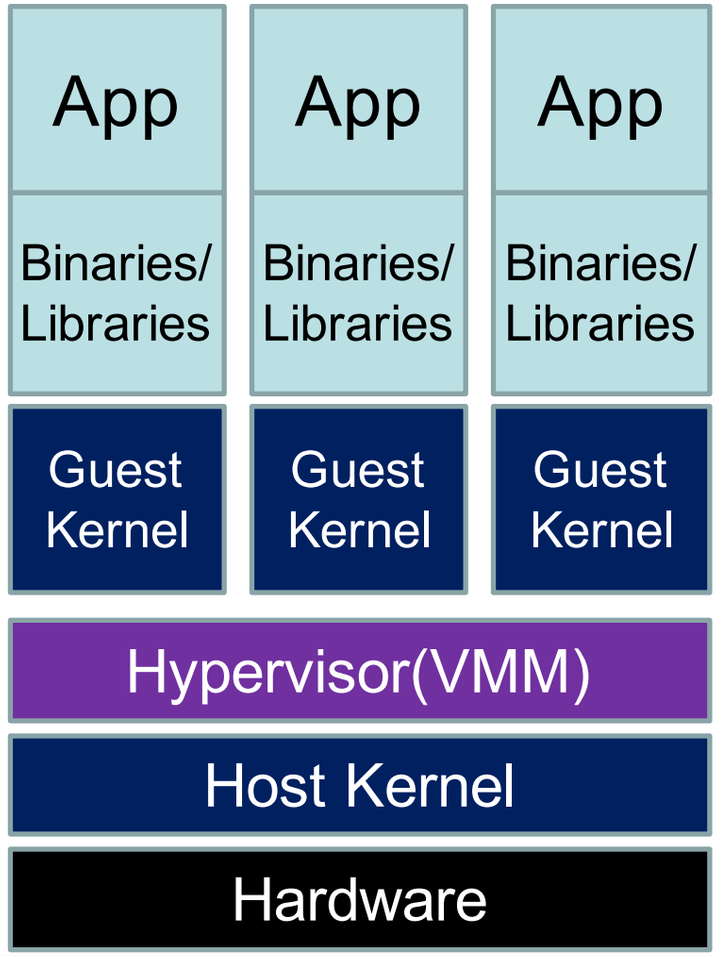
- 优势：资源利用率高，性能好
- 不足：隔离性和安全性较差



云数据库运行环境的选择

虚拟机

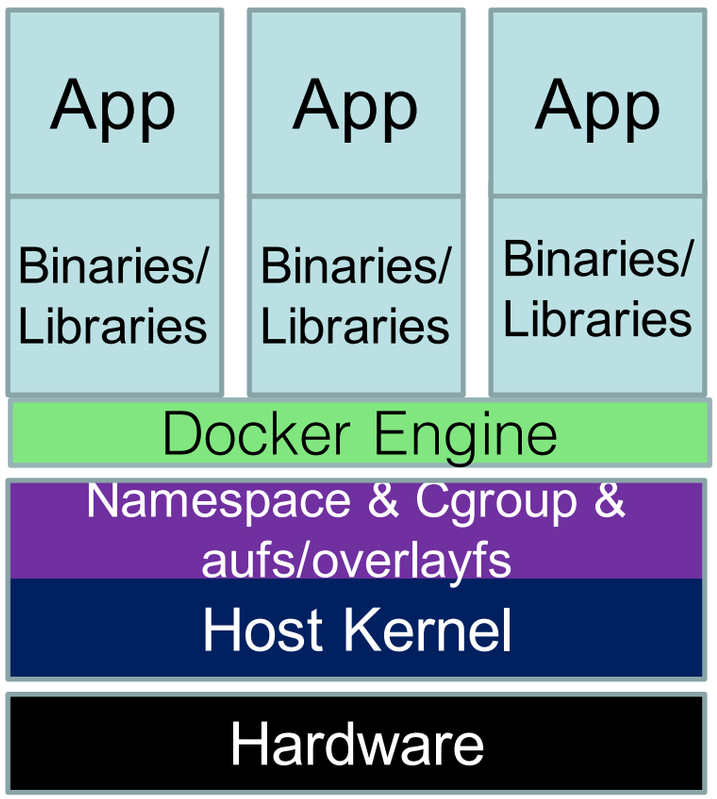
- 优势：隔离性好，安全性高
- 不足：资源利用率低、 镜像大 安装启动时间长



云数据库运行环境的选择

Docker

- 比虚拟机占用了更少的资源，比物理机隔离性更好



Docker

Docker原理

- namespace提供虚拟机的**假象**
- cgroup资源度量
- aufs精简镜像

namespace

Namespace	系统调用参数	隔离内容
UTS	CLONE_NEWUTS	主机名与域名
IPC	CLONE_NEWIPC	信号量、消息队列和共享内存
PID	CLONE_NEWPID	进程编号
Network	CLONE_NEWNET	网络设备、网络栈、端口等等
Mount	CLONE_NEWNS	挂载点（文件系统）
User	CLONE_NEWUSER	用户和用户组

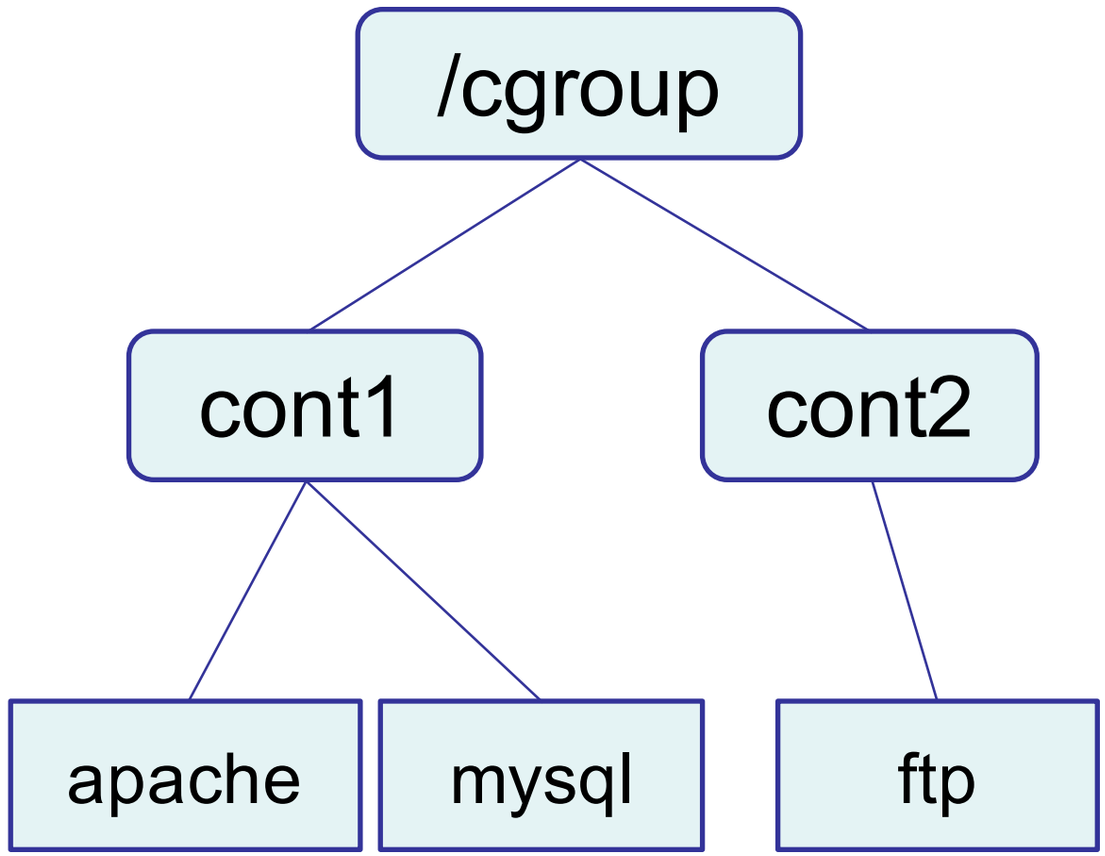
cgroup

有哪些cgroup?

- **cpu**
- memory
- blkio
- cpuset, freezer, net_cls, net_prio, devices, perf, cpuacct, hugetlb

mount –t cgroup none /cgroup

“对Container进行ulimit限制”



group1: 内存上限3072M，cpu占80%
group2: 内存上限1024M，cpu占20%

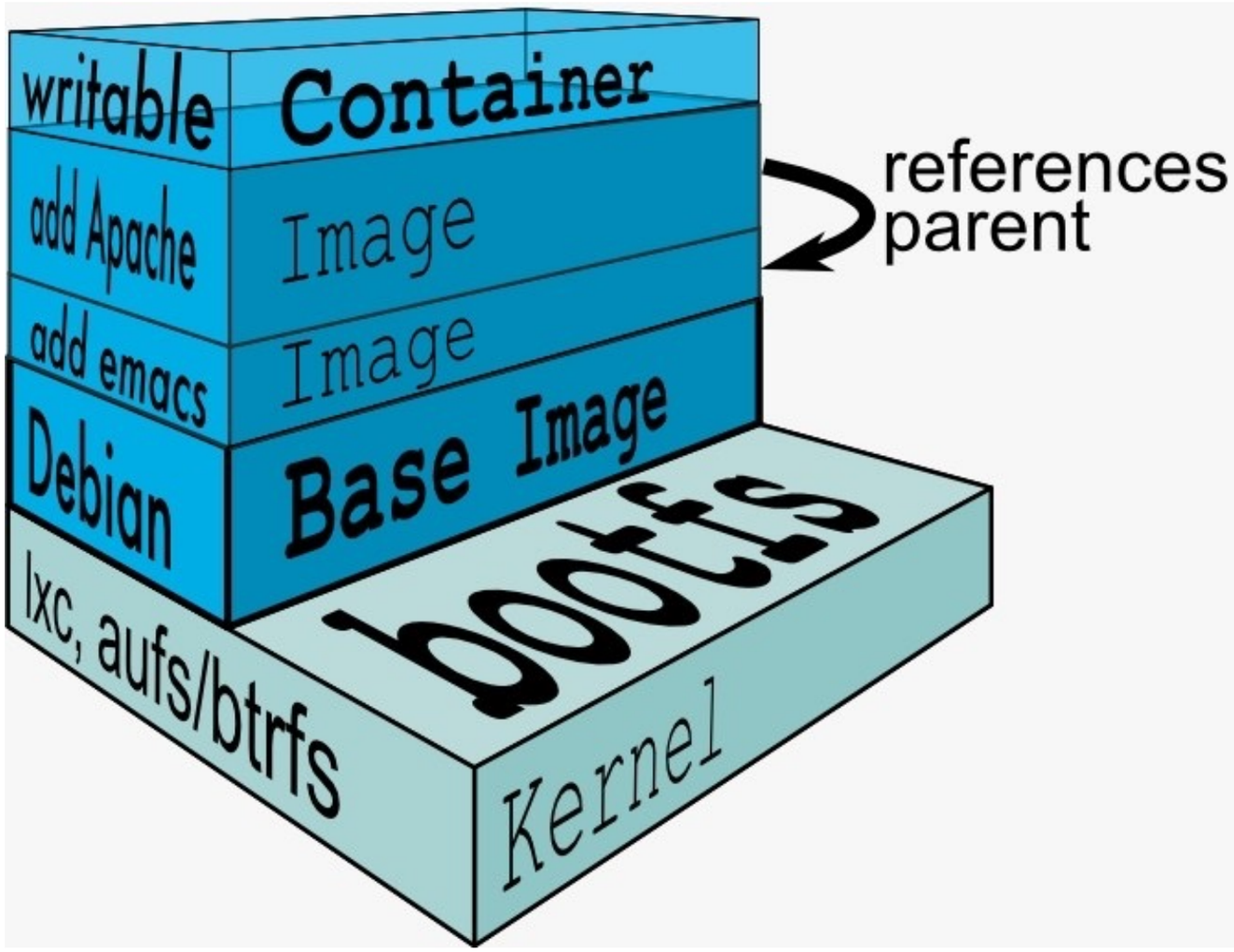
aufs overview

- Another Union File System　同类：Overlayfs

/base
boot
bin
lib
lib64
usr
sbin

/app1
emacs

/Container1
boot
bin
lib
lib64
usr
sbin
emacs



mount -t aufs -o br=/base=ro:/app1=rw　none /Container1

Docker & VM

特性	虚拟机	Container
启动速度	分钟	百毫秒
镜像大小	1G	10M
运行内存	1G、8G、32G	?
性能开销	CPU：1% 磁盘IO：20% 网卡: 20%	CPU：~0% 磁盘IO：~0% 网卡：同VM
迁移	在线	离线
安全隔离	严格隔离	共用内核

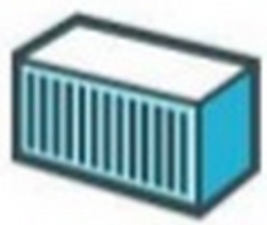
云数据库运行环境的选择

Docker

- 安装+配置+运行 -> 复制+运行

Build

通过容器构建app，不管语言工具链



Ship

完整地ship容器化的app到任何地方-QA、云平台

Run

随时run容器化的app在各类云平台、虚拟机、个人PC、移动设备上

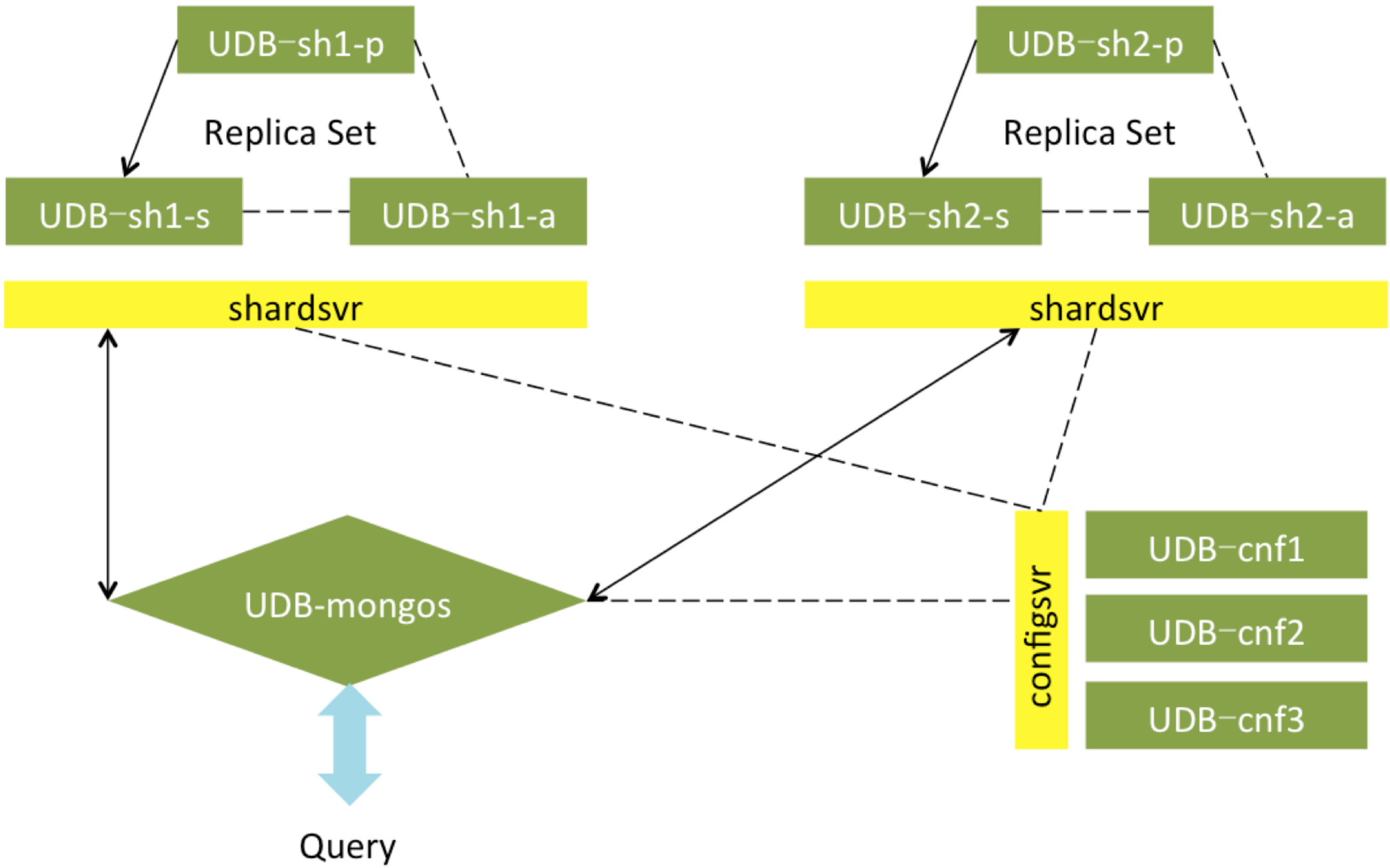
UCloud DataBase为什么选择Docker

UCloud DataBase Docker应用实践

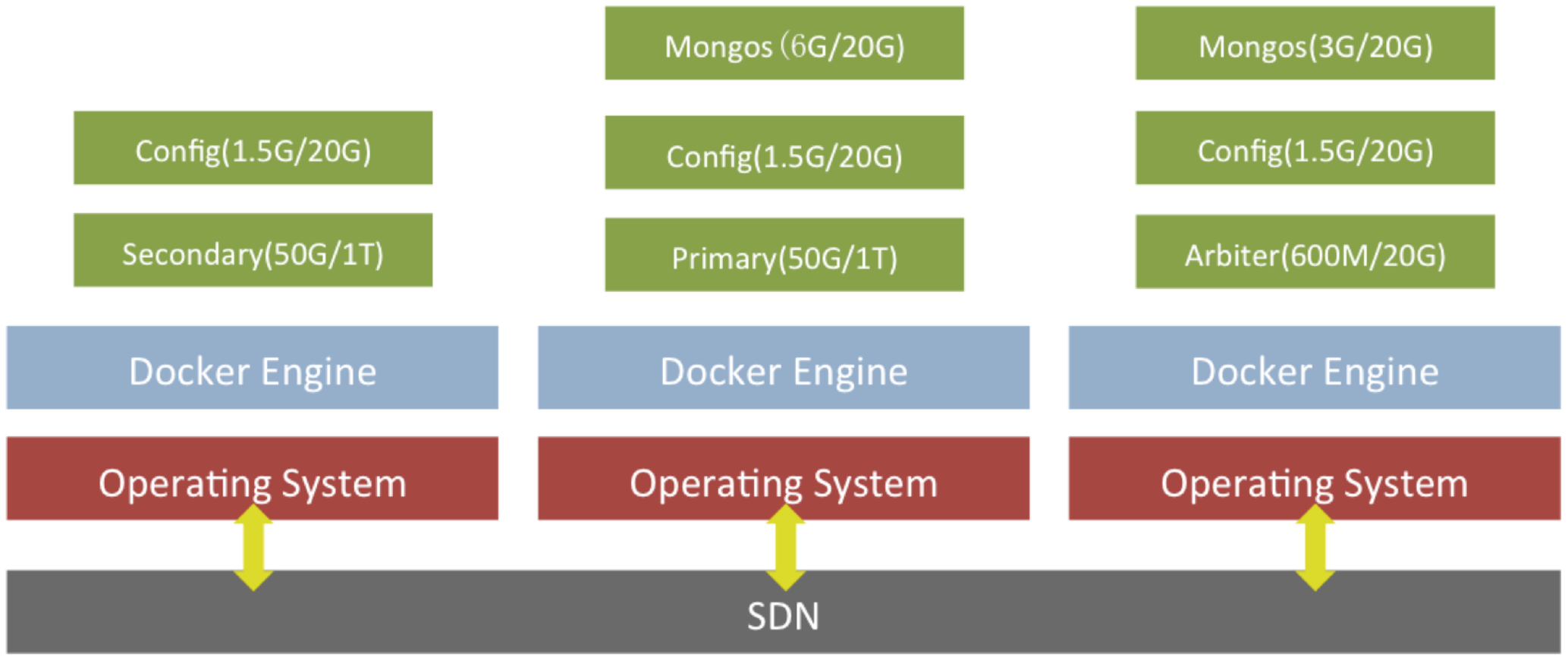
谈谈如何更好的使用Docker

首先选择了Mongodb

- 具备分布式 高可用能力
- 模块较多 较复杂
- 相比Mysql用户较少


UDB-sh1-p
Replica Set
UDB-sh1-s
UDB-sh1-a
shardsvr
UDB-sh2-p
Replica Set
UDB-sh2-s
UDB-sh2-a
shardsvr
UDB-mongos
configsvr
UDB-cnf1
UDB-cnf2
UDB-cnf3
Query

Mongodb Docker化



Docker和Mongodb优势互补

Feature	Docker	MongoDB
High Performance	√	√
Scale-up	√	-
Scale-out	-	√
High Availability	-	√
Cost	√	-
Deployment/Maintenance	√	-
Security	√	-

云数据库Docker化遇到的挑战

- 数据库有状态
- SDN环境网络通信
- /proc和/cgroup隔离问题

数据库有状态

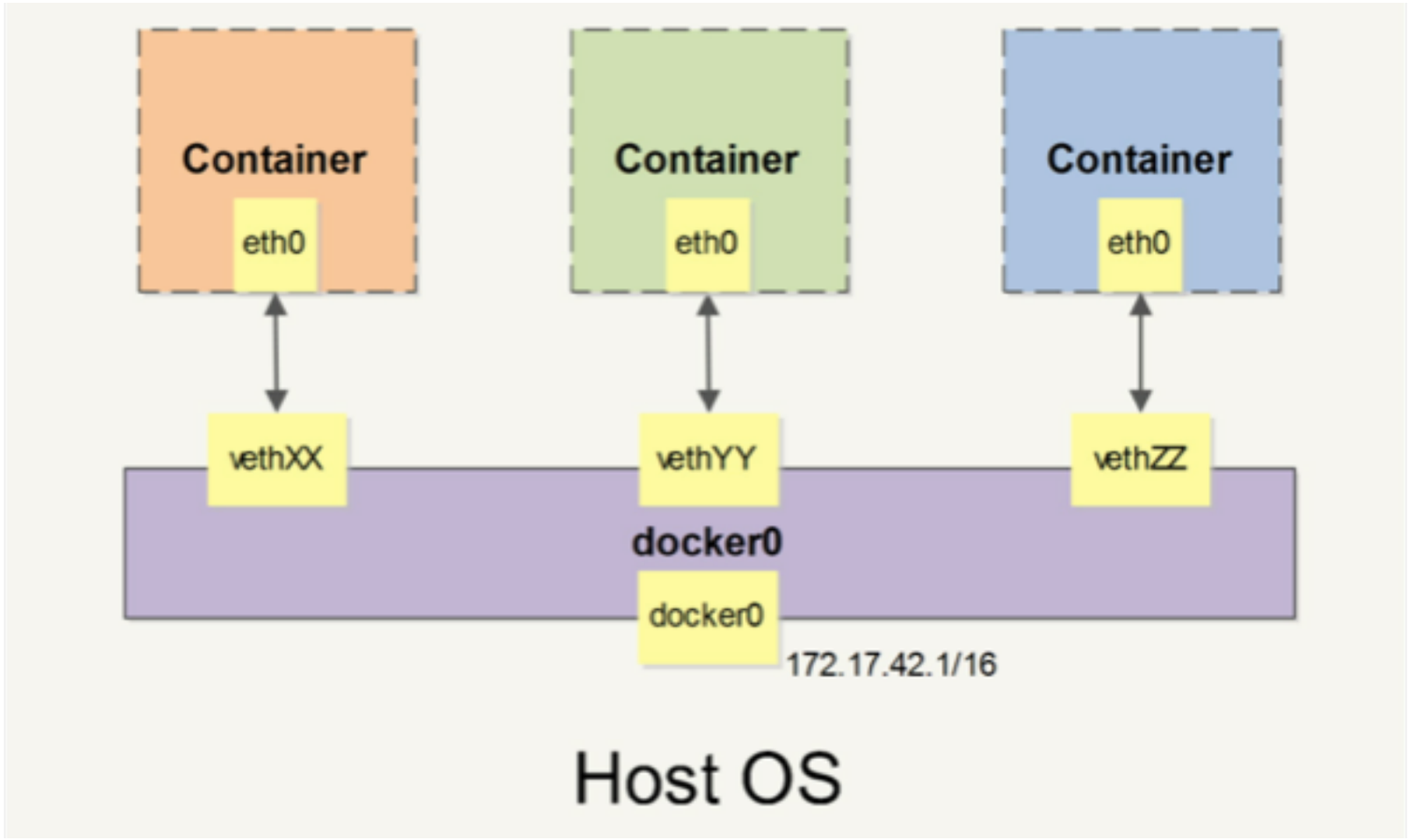
- Docker + LVM
- 离线迁移或者利用数据集群特性做在线迁移

SDN环境的网络通信

pipework br0 instance_name 10.10.101.150/24@10.10.101.254

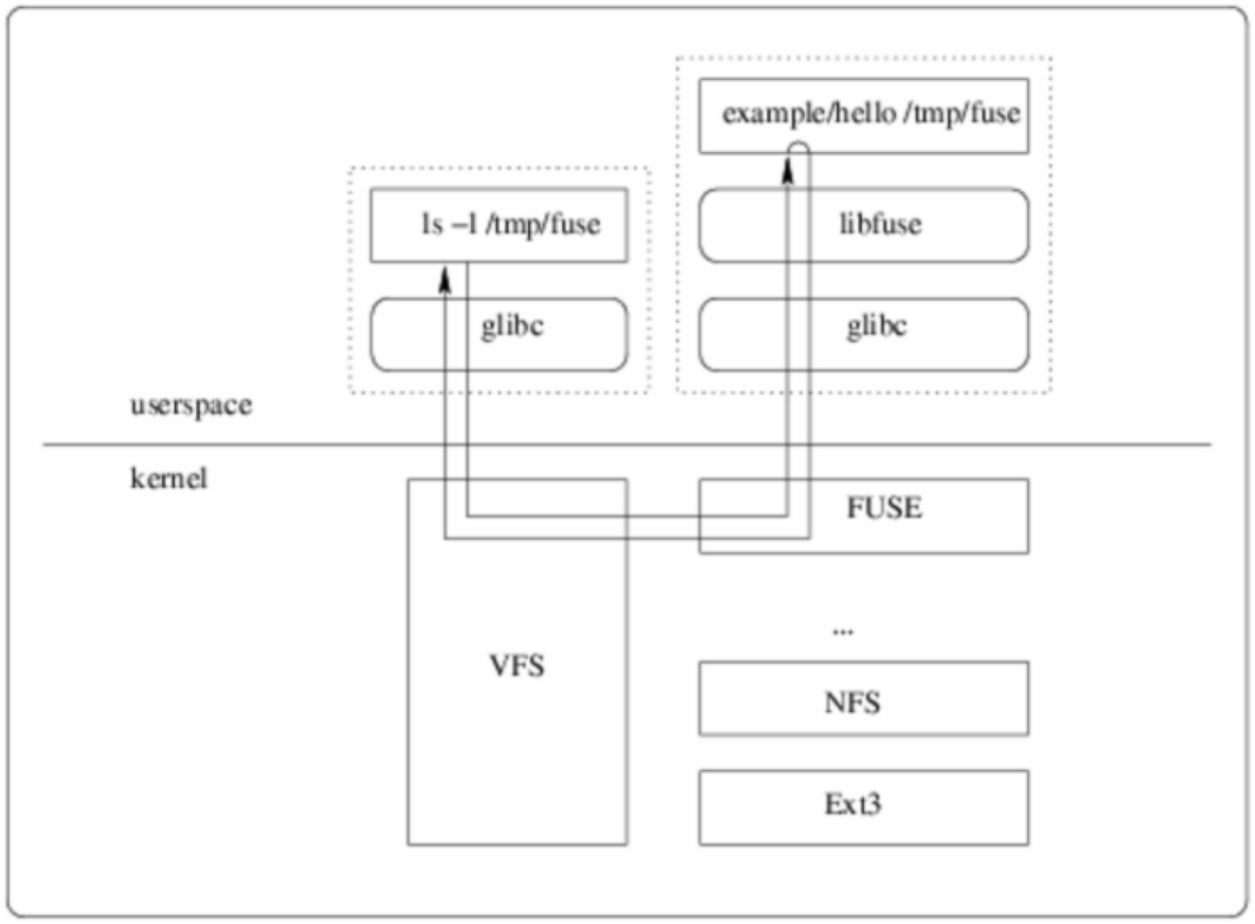
```
#创建br0网桥
#若ovs开头，则创建OVS网桥 ovs-vsctl add-br ovs*
brctl addbr $IFNAME
#创建veth pair,用于连接容器和br0
ip link add name $LOCAL_IFNAME mtu $MTU type veth peer name $GUEST_IFNAME mtu $MTU
#找到Docker容器test1在主机上的PID,创建容器网络命名空间的软连接
DOCKERPID=$(docker inspect --format='{{ .State.Pid }}' $GUESTNAME)
ln -s /proc/$NSPID/ns/net /var/run/netns/$NSPID
#将veth pair一端放入Docker容器中，并设置正确的名字eth1
ip link set $GUEST_IFNAME netns $NSPID
ip netns exec $NSPID ip link set $GUEST_IFNAME name $CONTAINER_IFNAME
#将veth pair另一端加入网桥
#若为OVS网桥则为 ovs-vsctl add-port $IFNAME $LOCAL_IFNAME ${VLAN:+"tag=$VLAN"}
brctl addif $IFNAME $LOCAL_IFNAME
#为新增加的容器配置IP和路由
ip netns exec $NSPID ip addr add $IPADDR dev $CONTAINER_IFNAME
ip netns exec $NSPID ip link set $CONTAINER_IFNAME up
ip netns exec $NSPID ip route delete default
ip netns exec $NSPID ip route add $GATEWAY/32 dev $CONTAINER_IFNAME
```

SDN环境的网络通信


Container
eth0
Container
eth0
Container
eth0
vethXX
vethYY
vethZZ
docker0
docker0
172.17.42.1/16
Host OS

/proc和cgroup的隔离问题

docker --volumns /var/lib/lxcfs/proc/:/docker/proc/



运维中面临的问题和挑战

- 模块数量多，模块间的关联关系复杂，维护和部署成本高
- 多环境，多IDC部署，应用配置维护难
- 迭代速度快，部署交付效率低
- 服务器和应用规模增加迅速，应用管理和运维成本高
- 开发 测试 预发布 线上环境不一致

如何解决这些问题

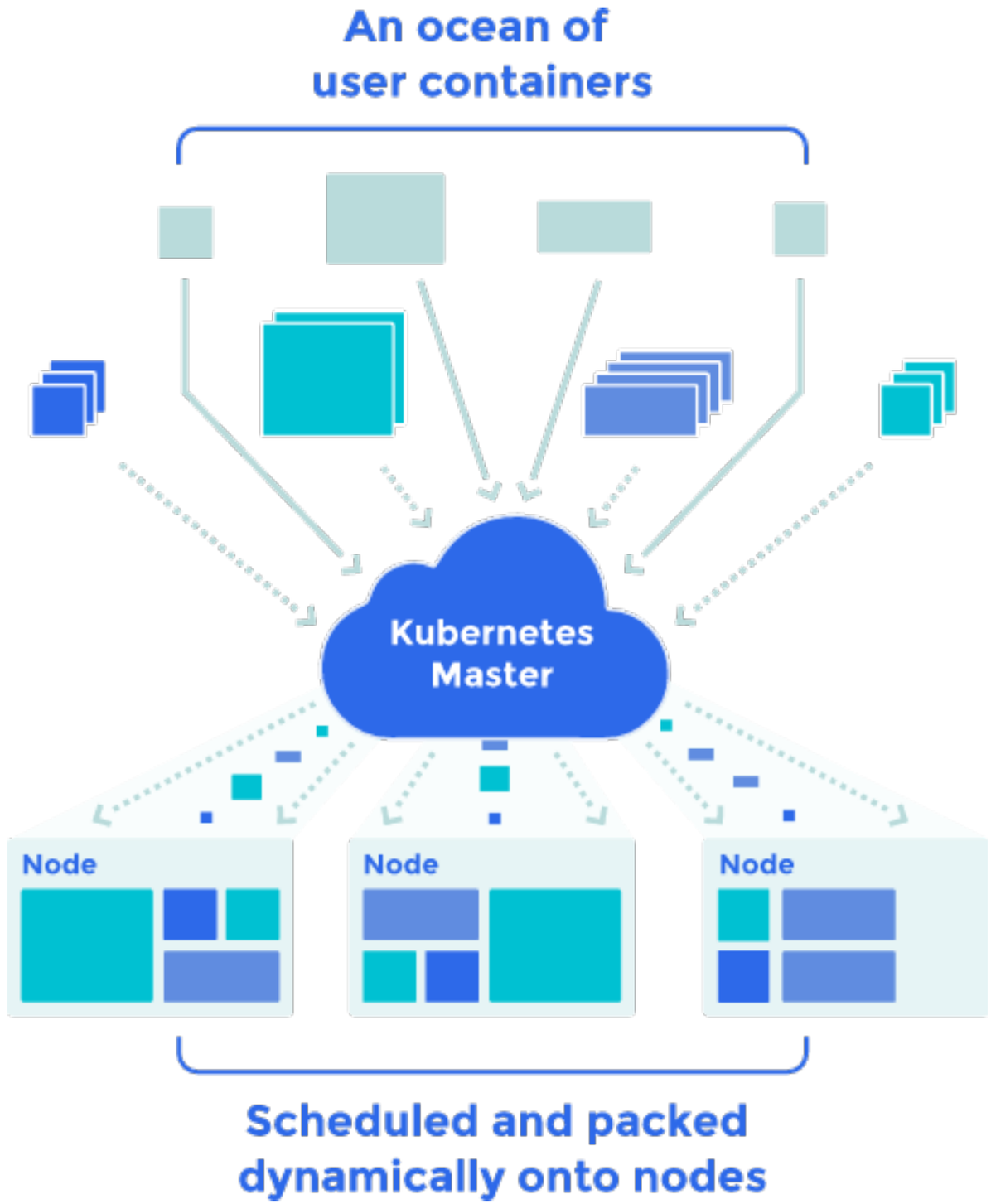
- 采用Docker镜像方式部署
- 基础系统、基本工具、框架，协议分层构建；
- 配置文件剥离环境、IDC等参数
- 开发、测试、线上运行环境均采用docker生成的镜像，保证一致；
- 创建私有镜像库

容器集群管理-kubernetes

集群基本需求

- ☑ 资源调度
- ☑ 负载均衡
- ☑ 健康检查
- ☑ 实例伸缩
- ☑ 服务发现
- ☑ 生命周期

- 程序构建
- 镜像管理
- 存储和网络



容器集群管理-kubernetes

“Pets vs Cattle” (Yes, again)

- Servers are like pets.

Pets are given names, are unique, lovingly hand raised and cared for. When they get ill, you nurse them back to health

- Servers are like cattle.

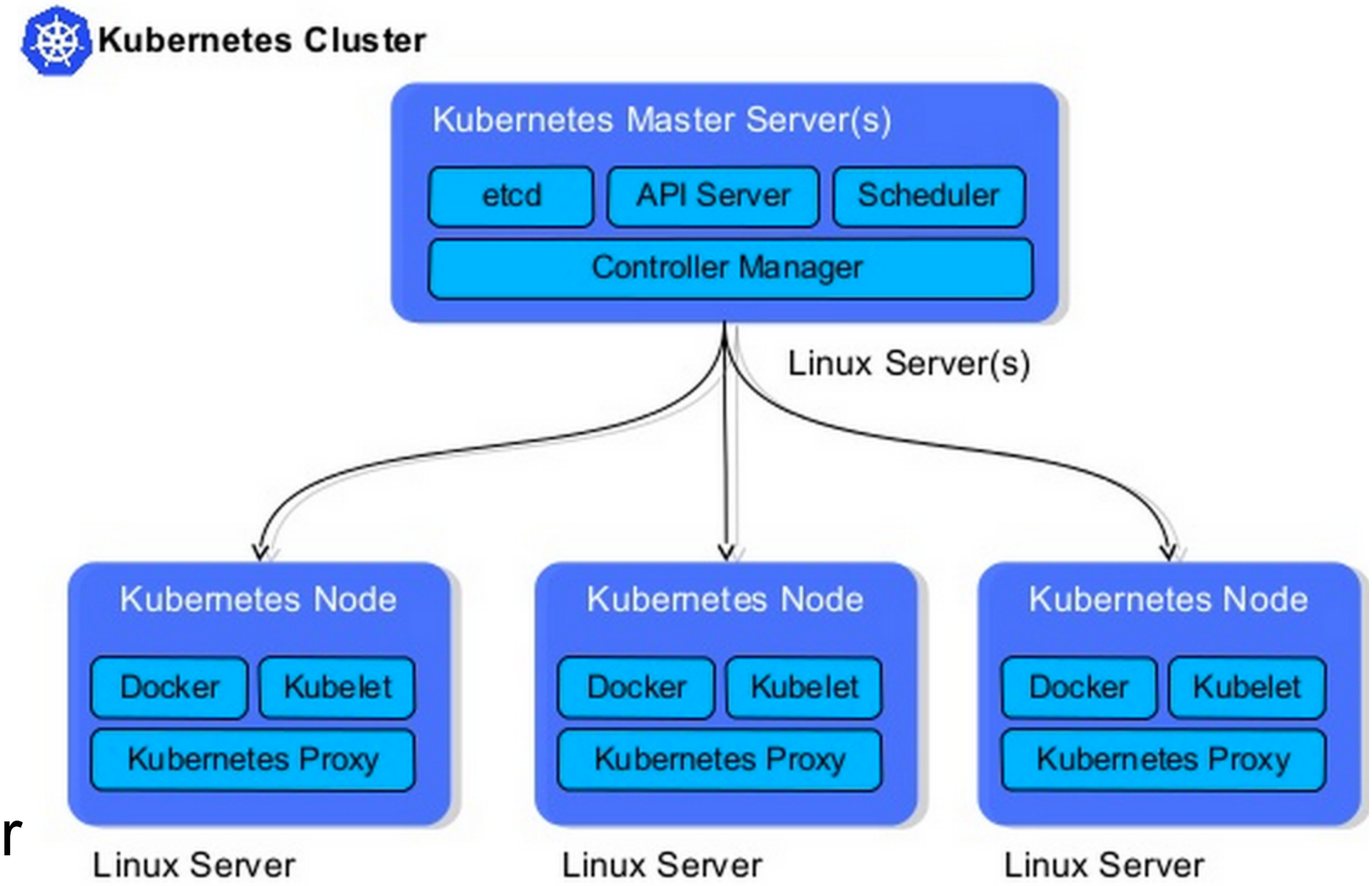
Cattle are given numbers and are almost identical to each other. When they get ill, you get another one.

> “Future application architectures should use Cattle but Pets with strong configuration management are viable and still needed”
> - Tim Bell, CERN

The above adapted from Tim Bell, CERN
http://www.slideshare.net/noggin143/20121017-openstack-cern-accelerating-cience

容器集群管理-kubenetes

- 模块化
- 可扩展
- 轻量级



- container
- pod
- controller
- service
- labels & selector
- scheduler

容器集群管理收益明显

- **部署效率提升**
  - 部署时间减少：小时级别到分钟级
  - 部署效果能有效保证，运行副本数满足期望
- **容器生命周期管理**
  - 自动故障恢复，减少故障处理运维成本

UCloud DataBase为什么选择Docker

UCloud DataBase Docker应用实践

谈谈如何更好的使用Docker

一些经验

- **Docker日志**
  - 最好关闭logdrive，将日志打印到后台
- **Docker Daemon**
  - centos 6.3 service stop耗时长，需要5min，init-scripts的bug
  - 退出Daemon前先退出container
- **Docker网络**
  - NAT模式下会启用启用nf_conntrack造成性能下降：调节内核参数

一些经验

- **Docker Image**
  - 制作基础镜像，要求是干净、安全的，推荐是来自官网
  - 使用Dockerfile或者其他工具，安装必要的工具

- **操作系统**
  - 注意内核版本与Docker版本适配
  - 注意Cgroup挂载
  - 提高最大文件数限制，影响到DB的连接数（1.6以前）

- **数据持久化**
  - 数据卷挂载
  - 为数据卷设置合理读写权限

一些经验

- **网络设置**
  - 自定义网桥，限制docker0
  - 内网按多租户隔离
  - 不作DB端口映射
  - 按需配置DNS

- **安全加强**
  - 启动SELinux/GRSEC等安全机制
  - 启用能力机制，控制某些超级权限

- **Docker Daemon防护**
  - 禁止宿主机的根目录映射
  - 禁止滥用root权限

THANKS

微信号：weixinhao2015