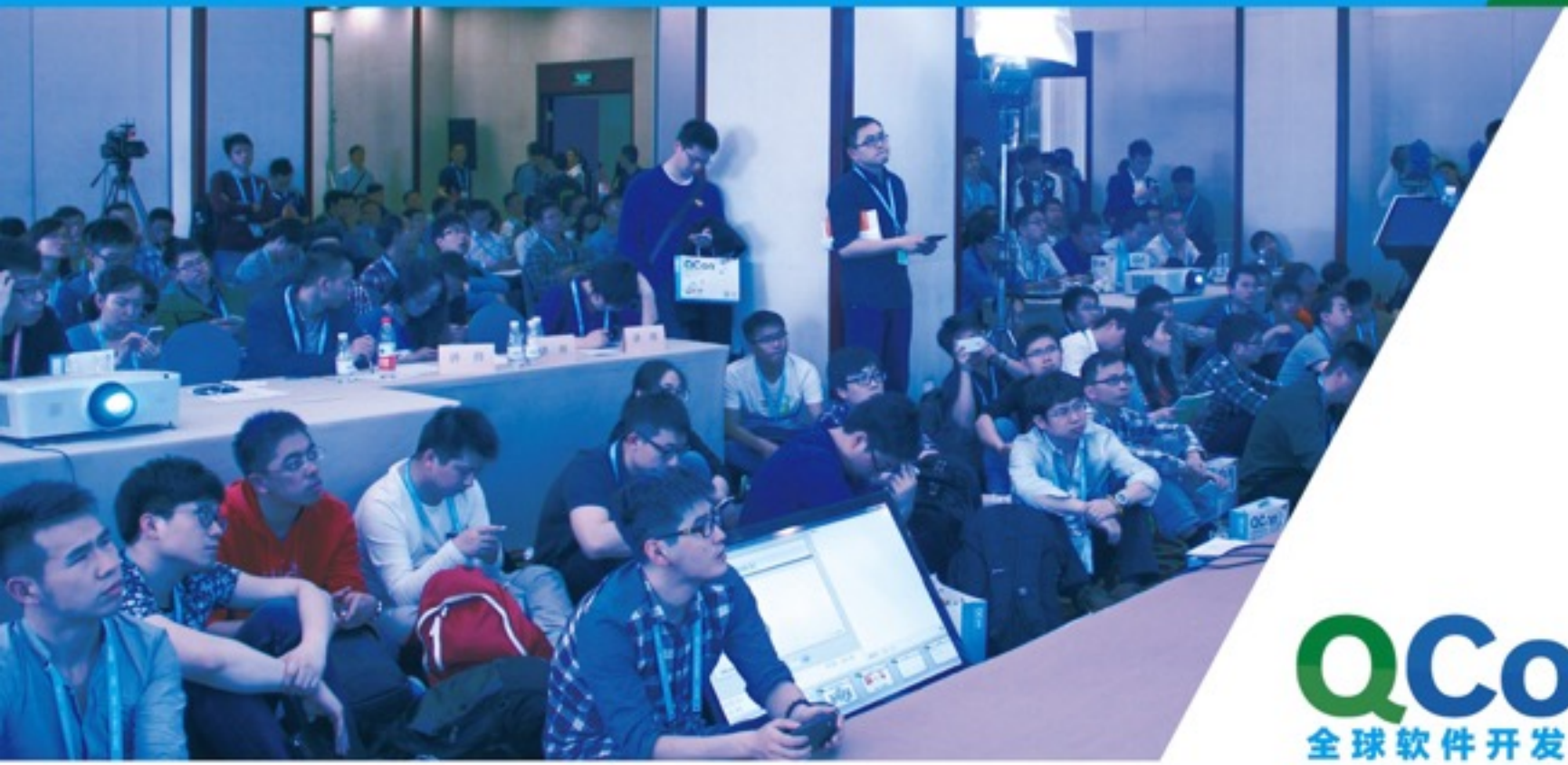
QCon全球软件开发大会

International Software Development Conference



QCon
全球软件开发大会

蚂蚁金服金融云PAAS docker实践

蚂蚁金服基础技术部系统组-知胜(吴峥涛)

为什么选择docker?

大型网站所面对的问题

大型网站所面对的问题

负载均衡

缓存

SOA服务化

水平扩展

多机房多

数据分库分表

数据一致性

……

高性能通讯组件

可靠消息中心

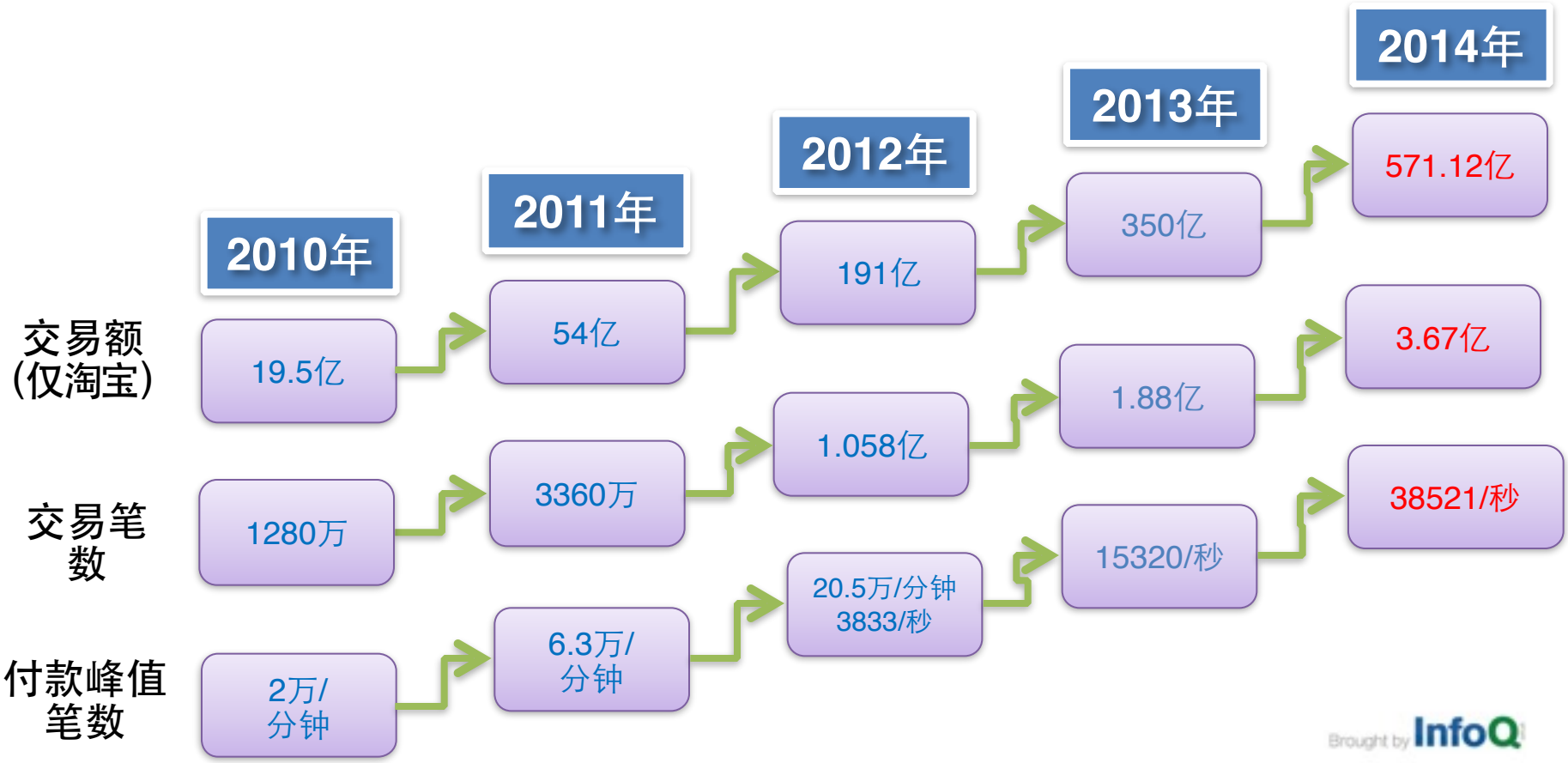
服务发现

异地灾备

蚂蚁中间件地图

请参考蚂蚁金融云网站

有效应对日益增长的业务压力



以PAAS方式打包输出中间件

- 与阿里云整合
- 帮助用户去IOE
- 降低用户研发/运维成本
- 帮助金融用户解决技术问题，让用户专注于业务逻辑

上帝的归上帝，凯撒的归凯撒

蚂蚁金融云架构

网商银行 | SAAS | 神秘应用 | 不可说 | ……

金融云PaaS平台

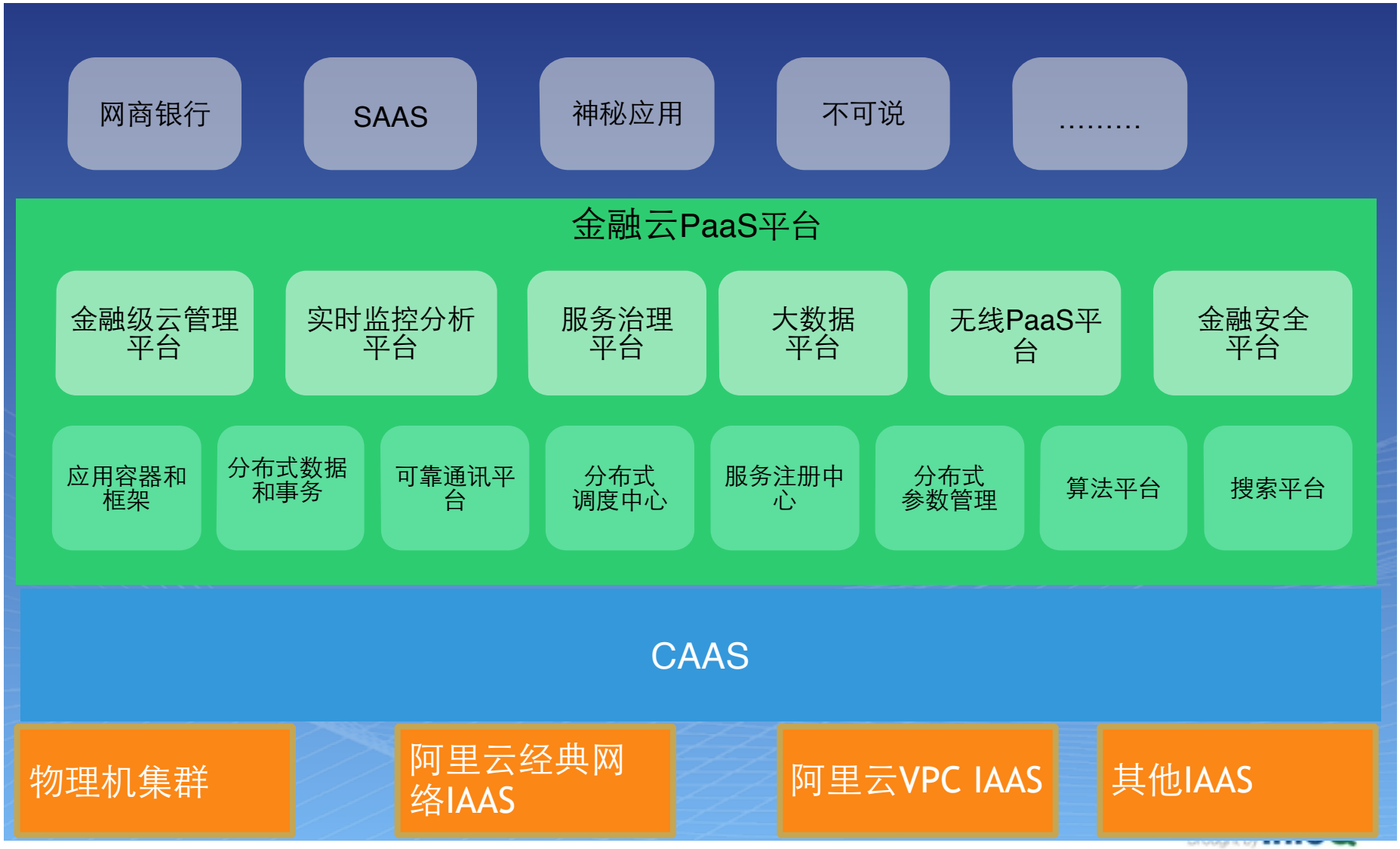
金融级云管理平台 | 实时监控分析平台 | 服务治理平台 | 大数据平台 | 无线PaaS平台 | 金融安全平台

应用容器和框架 | 分布式数据和事务 | 可靠通讯平台 | 分布式调度中心 | 服务注册中心 | 分布式参数管理 | 算法平台 | 搜索平台

IAAS

碰到的问题

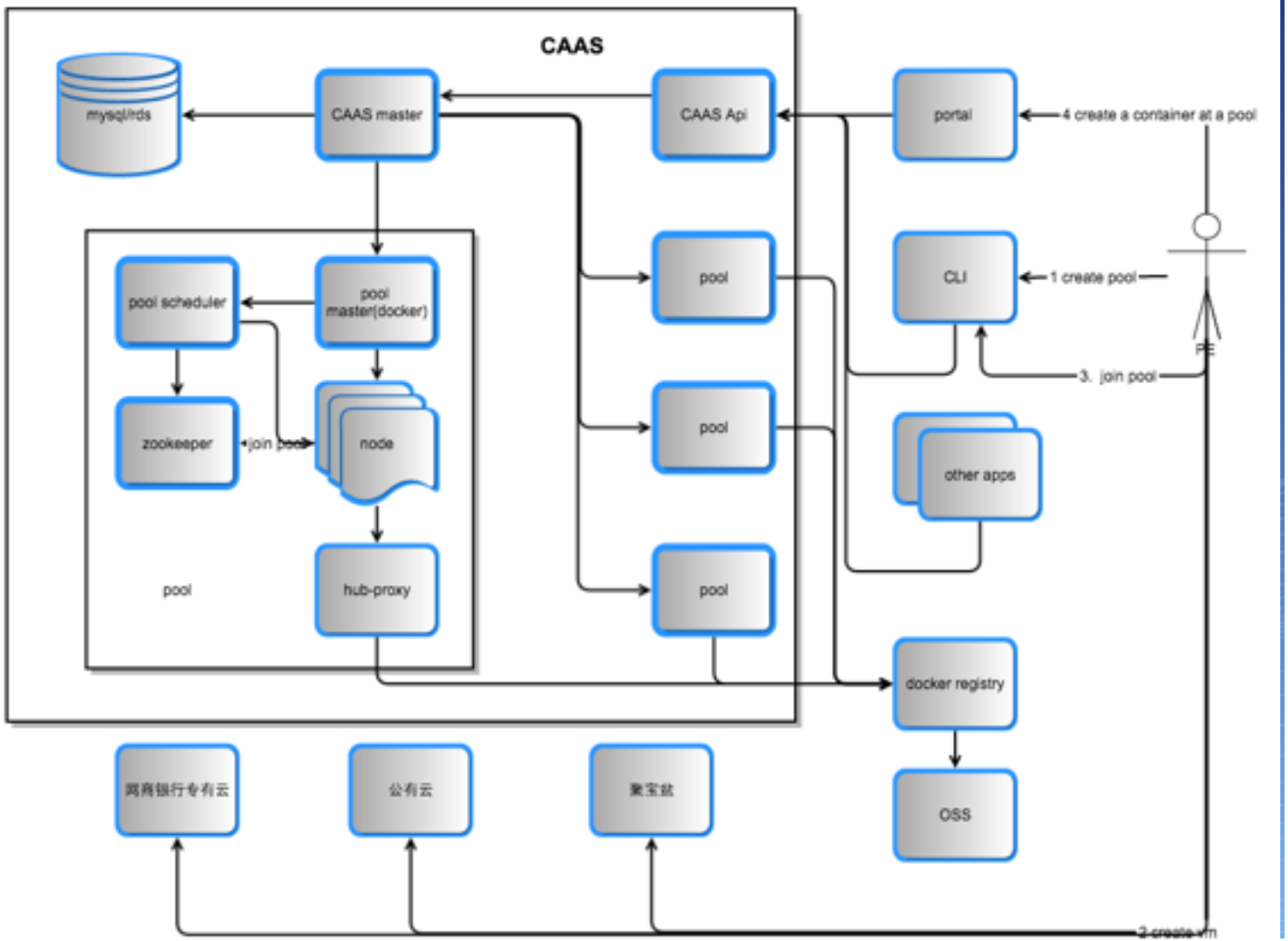
- SOA服务化架构导致的模块数量快速增长与资源粒度的矛盾
- 自动部署/扩容/缩容对资源交付速度的要求
- 租户间安全隔离的需求
- 自定义组网需求
- 兼容多种IAAS


网商银行
SAAS
神秘应用
不可说
.....
金融云PaaS平台
金融级云管理平台
实时监控分析平台
服务治理平台
大数据平台
无线PaaS平台
金融安全平台
应用容器和框架
分布式数据和事务
可靠通讯平台
分布式调度中心
服务注册中心
分布式参数管理
算法平台
搜索平台
CAAS
物理机集群
阿里云经典网络IAAS
阿里云VPC IAAS
其他IAAS

CAAS职责

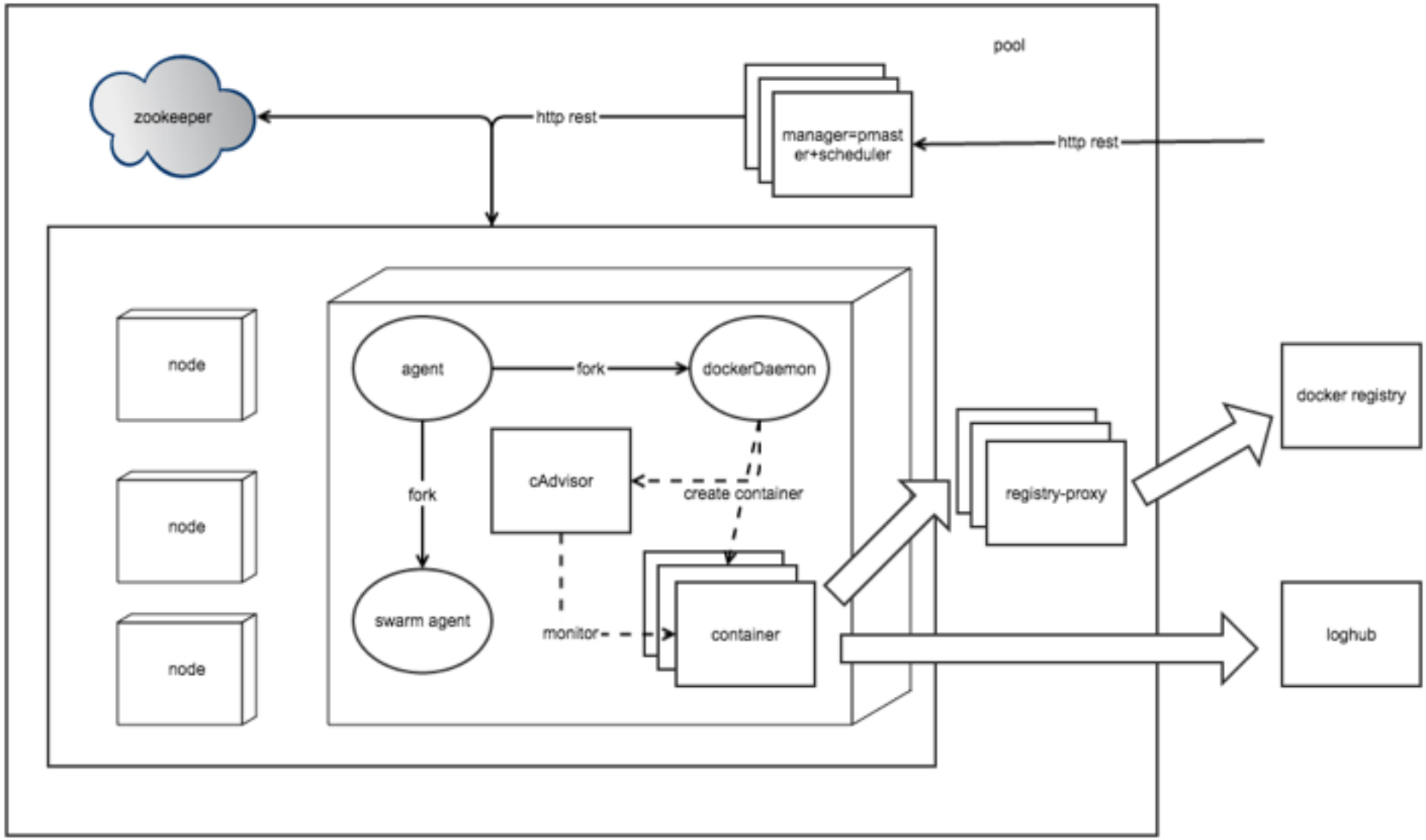
- 以微容器(docker)为载体，为用户按需提供计算/存储/网络资源
- 提高交付速度与资源利用率，降低资源粒度
- 对PAAS屏蔽IAAS，兼容多种IAAS实现
- 集群和应用的标准化/自动化/产品化

AntCAAS 架构


CAAS
mysql/rds
CAAS master
CAAS Api
portal
4 create a container at a pool
pool scheduler
pool master(docker)
pool
CLI
1 create pool
zookeeper
join pool
node
3. join pool
pool
other apps
pool
hub-proxy
pool
docker registry
网商银行专有云
公有云
聚宝盆
OSS
2 create vm

架构特点

- 多个pool解决安全性
- node可以是物理机或者虚拟机
- zk用以监控node和container（可以多个pool共享一套zk，也可以用etcd）
- scheduler 调度器，负载container的创建调度
- manager，对master提供管控的http rest接口
- registry-proxy 保证网络联通性以及cache镜像数据，加速镜像下载
- 在node内起cadvisor监控container


pool
zookeeper
http rest
manager=pmaster+scheduler
http rest
node
node
node
agent
fork
dockerDaemon
fork
cAdvisor
create container
swarm agent
monitor
container
registry-proxy
docker registry
loghub

主要工作

网络扩展

- 在docker默认的bridge/host/none/container4种网络模式的技术上，模拟docker1.8的插件模型，扩展出vlan/vxlan两种drvier，正在实现vpc driver

  vlan driver用于在物理机集群上，需要在交换机上为container分配一个独立的vlan网段；

  vxlan driver用于解决经典网络ECS集群的overlay网络问题，是一个轻量级的私有网络解决方案；

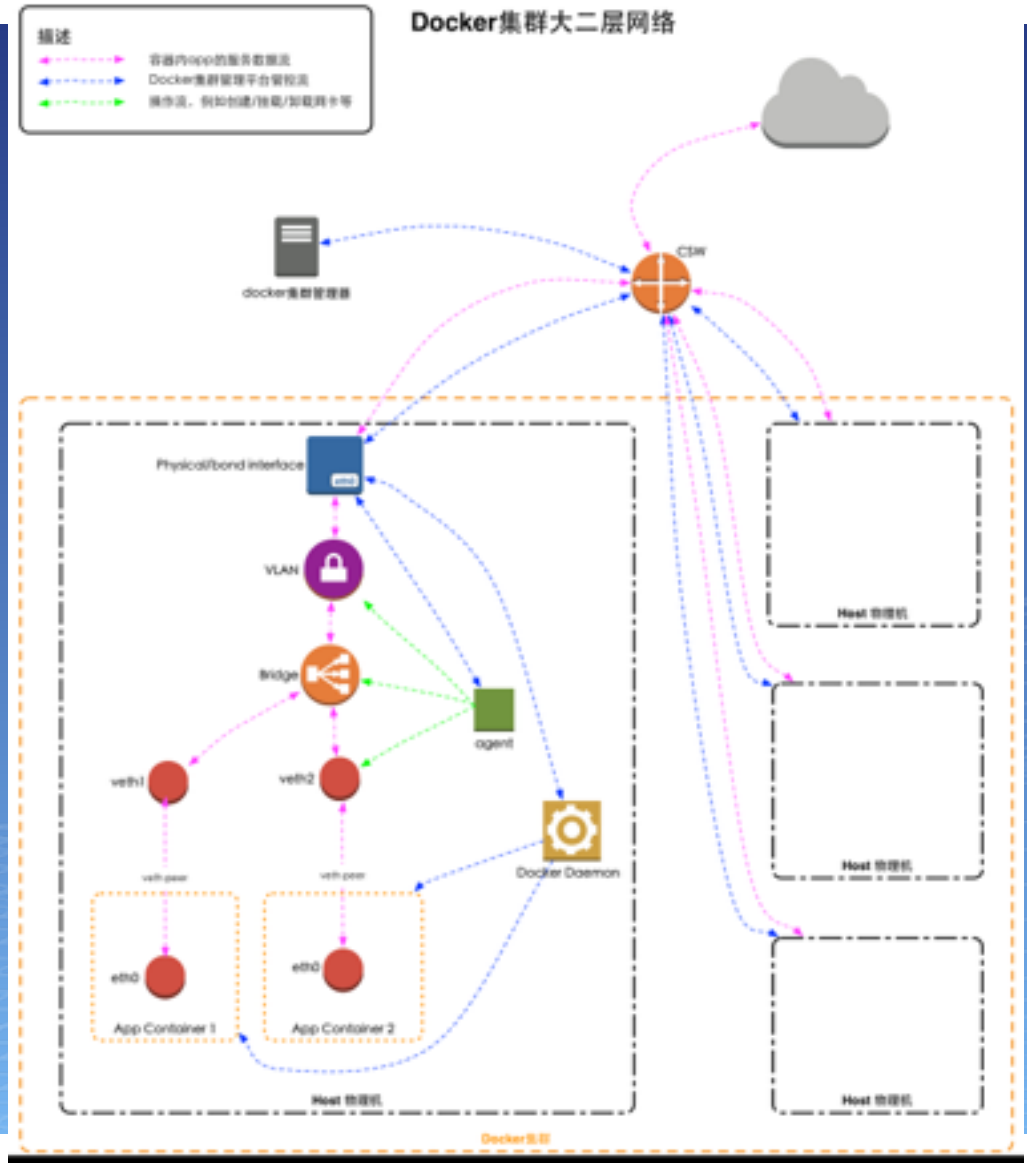
vlan driver

- 运维先创建一个vlan
  cli --pool=pool2 network create myvlan – d vlan -l 10.209.164.0/22 -l Gateway=10.209.164.1

- 动态创建一个container
  cli --pool=pool2 container create --publich-service=.myvlan -l ip=10.209.164.8

vlan 网络模型

下一步优化
1 mac nat
2 arp proxy



vlan driver 进一步优化

- 新问题
  交换机能纪录的mac地址和路由表项有限
  container的arp请求消耗了大量交换机的cpu

基于ovs进一步优化

经典网络ecs的网络问题

经典网络的ECS是一个大二层的虚拟网络，结构与vlan网络基本一致，用户不能自己指定ip，没有dns服务，通过安全组进行网络隔离。

如果在经典网络的vm上起container，则不能为之分配内网/公网ip。

docker默认的bridge/host/none/container四种模式都无法满足蚂蚁关键中间件配置中心的需求。

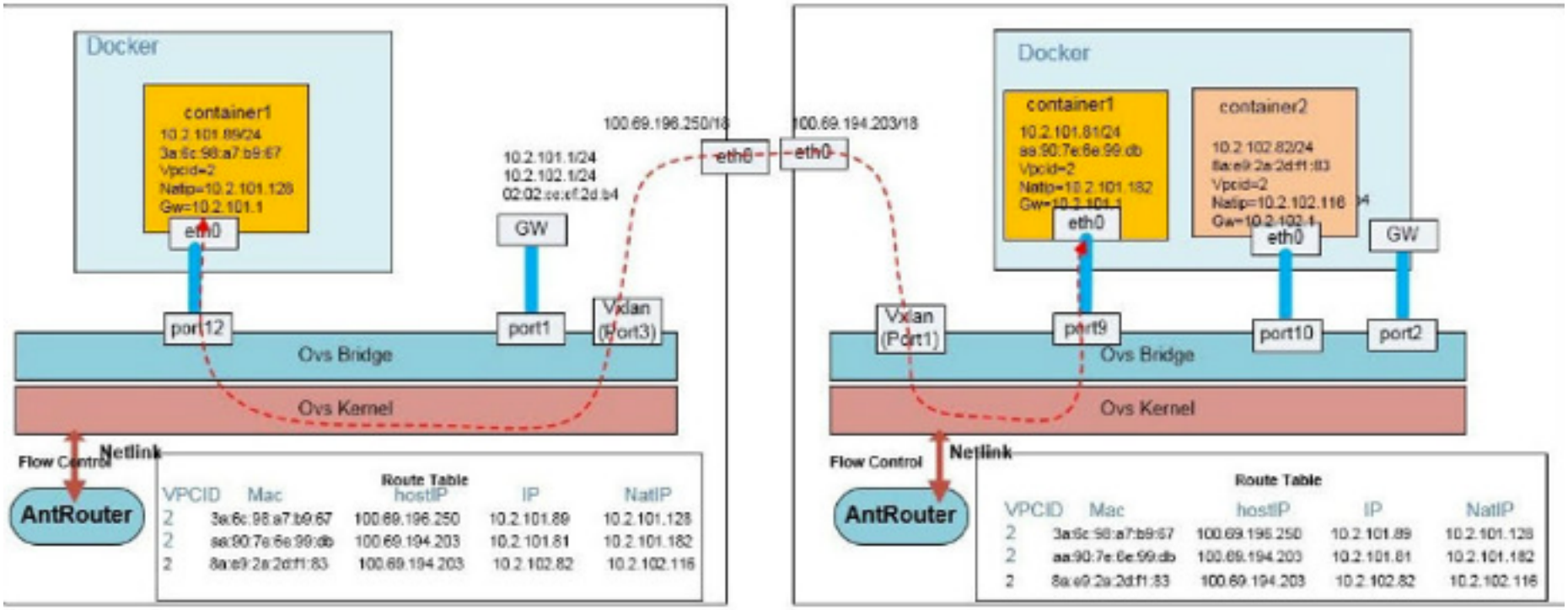
vxlan driver解决的问题

- 跨node的container自定义ip直连，
- 网络隔离
- container能够访问公网/node 网络
- 负载均衡
- 内部dns服务

vxlan driver example

- cli --pool=pool2 network create myvxlan – d vxlan -l Subnet=192.168.0.0/19 -l Gateway=192.168.31.254
- cli --pool=pool2 dns create dns1 -x myvxlan
- cli --pool=pool2 container create --publich-service=.myvxlan -l Ip=192.168.31.3 -l Hostname=core-1
- cli --pool=pool2 lb add_container 580cfe378bb --type=slb -l slb.config=/etc/acs/slb.config -l slb.lb=150240ff0a2-cn-hangzhou-dg-a0 -l container=24573203f308

基于ovs解决跨node的container互通



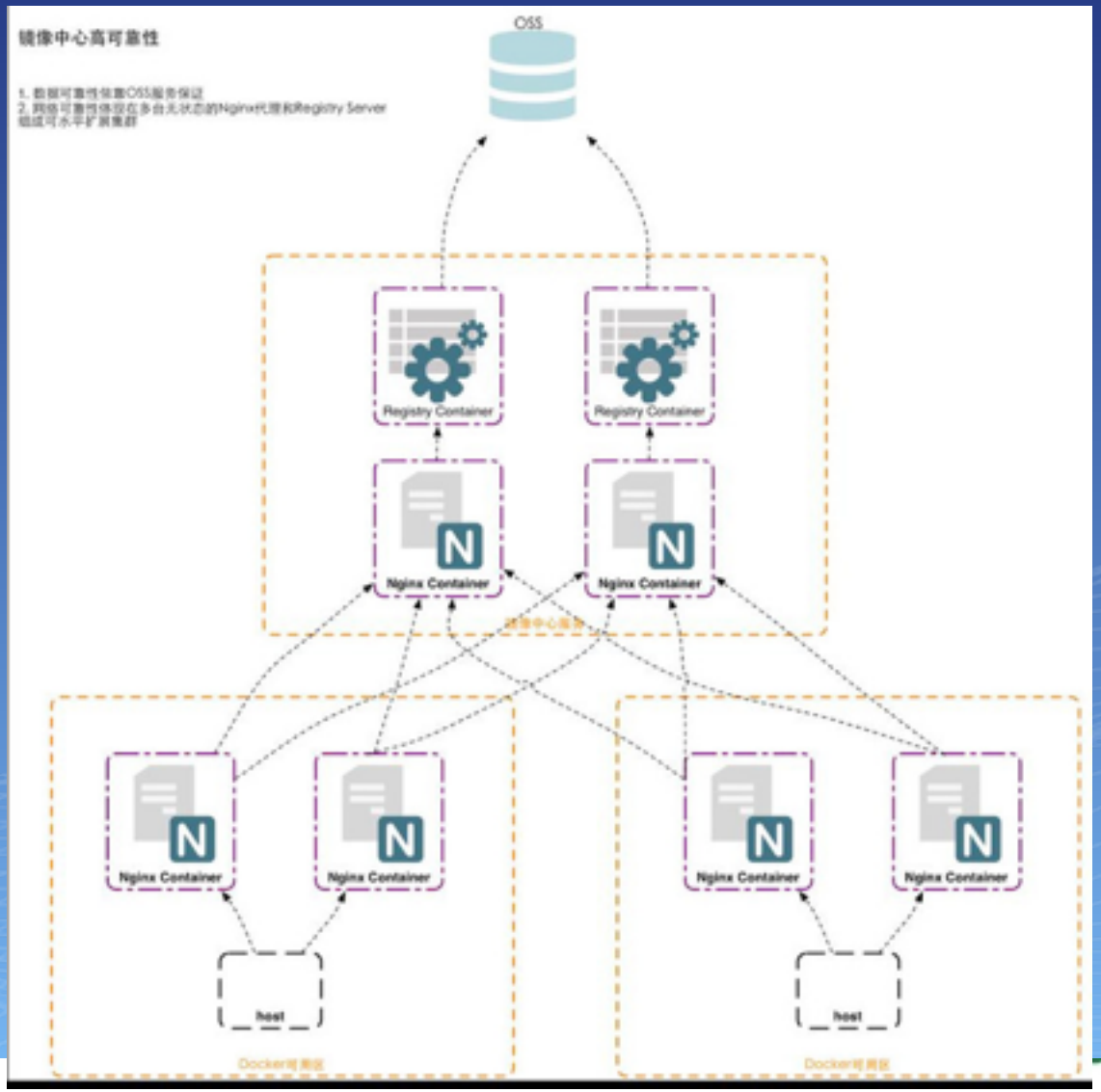
统一镜像中心

- 全局一份镜像中心
  镜像数据保存在oss上
  registry 无状态，可以水平扩展
  registry前加一个tenginx做缓存

- 在异地机房/pool部署registry-proxy
  减少跨机房流量
  加快镜像下载速度
  对镜像进行预热

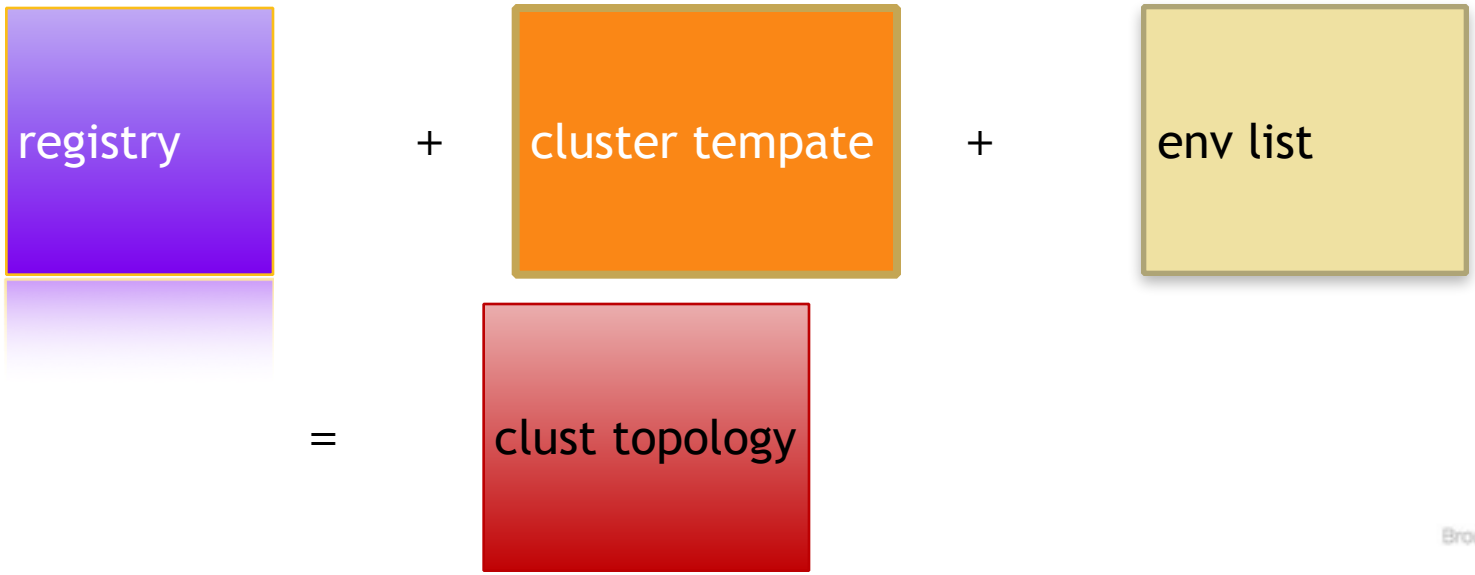
统一镜像中心架构

进一步优化：
1. 升级到docker registry v2
2. 以镜像为粒度管理缓存



集群复制与快速搭建

- 在生产实践中，经常需要快速搭建一些复杂的集群，这些集群内部角色众多，相互调用关系复杂，对外依赖复杂
- 不同环境下集群拓扑总有细微变化



镜像制作

- build once， run anywhere
  镜像需要和环境无关，和环境相关的参数需要在启动时作为参赛传入
- 启动容器时候尽量不使用mount

cluster template

- cluster template是集群静态关系的描述，静态关系之与环境无关，例如角色间依赖关系

```
bsoc:
    service:
        start: *service-start-sofa
    option:
        start:
            port: 80
            appname: bsoc
            env:
            {{ range $app.Env }}
              - {{ . }} {{ end }}
              - db_schema={{ $resources.Rds.socboss.Schema }}
              - db_url={{ $resources.Rds.socboss.Url }}
              - db_username={{ $resources.Rds.socboss.User }}
              - db_password={{ $resources.Rds.socboss.Password }}
              - auth_server_name={{ $apps.auth.Vip }}
              - core_server_name={{ $apps.core.Vip }}
              - boss_domain_name={{ $depends.ruly_url }}
              - boss_server_name={{ $apps.ruly.Vip }}
              - oss_endpoint={{ $resources.Oss.Schema}}{{ $resources.Oss.Endpoints.inn
              - oss_bucket_soc={{ $resources.Oss.Instances.bsoc.Name }}
              - odps_endpoint={{ $resources.Odps.Schema }}{{ $resources.Odps.Endpoints
              - odps_project_soc={{ $resources.Odps.Instances.bsoc.Name }}
              - sls_endpoint={{ $resources.Sls.Schema }}{{ $resources.Sls.Endpoints.da
              - sls_project_soc={{ $resources.Sls.Instances.bsoc.Name }}
              - zone={{ $info.zone }}
              - domainname={{ $info.domainname }}
              - log_level={{ $info.log_level }}
              - inner_domain={{ $info.inner_domain }}
            <<: *option-sofa
            image: {{ $dockerhub }}/{{ $app.Image }}
```

环境属性

- 外部资源
  集群运行所依赖的外部资源 - depends.yaml
- 集群内部资源
  为了搭建这个集群需要新建的rds/ocs/oss/slb - resources.yaml
- 应用特定配置

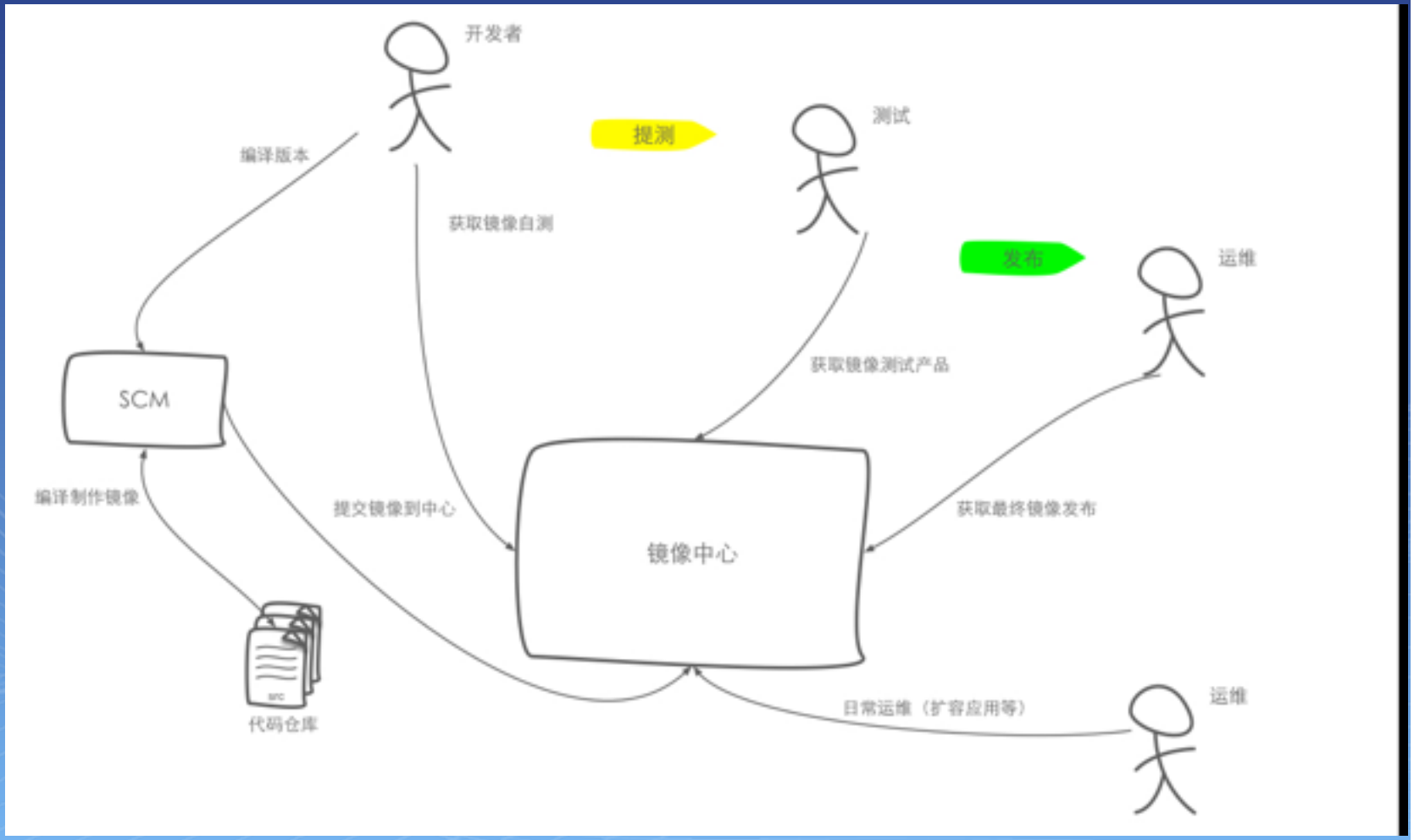
```
bsoc:
      env:
      - bsoc_thread_num=50
      - system_locale=zh_CN
      - system_model=BOSS
      - log_level=info
      image: paas/soc:f966b9-151010000706-d60c282
      instances: 2
```

根据集群拓扑创建集群

- 解析集群拓扑
- 创建rds/slb/oss/ocs等资源
- 根据集群拓扑为每个应用创建container
- 后续会兼容docker-compose

已有应用迁移的策略

理想模式


开发者
测试
提测
编译版本
获取镜像自测
发布
运维
获取镜像测试产品
SCM
编译制作镜像
提交镜像到中心
镜像中心
获取最终镜像发布
src
代码仓库
日常运维（扩容应用等）
运维

老应用迁移的痛

- 谁来写Dockerfile并制作应用镜像
  蚂蚁线上已经有上千应用，几千开发人员，很难一下推动他们都学习docker，切换到新的研发模式下；
  如果需要开发人员写dockerfile，会影响推广效率；

- 蚂蚁原有的运维/监控/SCM/财务等系统都是以vm为纬度的，基于docker的运维发布系统与原有系统对接比较麻烦
  以往运维都是先申请一批机器，测试网络正常后备用，上线前再决定跑什么应用；
  发布应用不重启vm，所以也不希望重启container；

- 怎么尽量保证开发测试环境与生产环境一致

应对策略

- 开发辅助工具帮助研发同学编译应用/自动生成dockerfile/制作镜像并搭建测试环境
- 把CAAS当作轻量级的IAAS，让运维把container当作轻量级vm用，便于和已有系统对接
- 使用通用的sofa4/sofa3 container，可以不需要制作应用镜像

```
FROM alipay.docker.io/alipay/sofa4:ba4040-150812020836

MAINTAINER zhengtao.wuzt <zhengtao.wuzt@alibaba-inc.com>
MAINTAINER smallfish     <xiaoyu.chen@alibaba-inc.com>

COPY ./release/opsware.tgz  /home/admin/release/run/opsware.tgz

RUN chown -R admin.admin /home/admin/release/run

CMD ["/usr/bin/supervisord", "-c", "/etc/supervisor/supervisord.conf"]
```

应对策略

- 在基础镜像中集成sshd，运行运维ssh到container中
- 使用supervisor启动应用和相关监控/运维agent
- 提供webconsole允许开发人员登录container查看日志/进行一定权限的操作
- 使用 data container 避免本地 mount

THANKS