微博新一代LAMP架构

基于混合云平台的弹性扩容架构

新浪微博 侯青龙

2017 PHP全球开发者大会

关于我

- 2010年加入微博
- 2011年： PC主站，参与微博v2、v3、v4版研发
- 2012年：架构部，负责微博多机房项目、微博多语言
- 2013年：激励团队，负责淘浪项目技术架构工作
- 2014年： PC主站，负责微博v6版整体架构设计
- 2015年至今（PC主站）：负责PC主站研发管理工作

分享大纲

01

背景与挑战

02

DCP平台介绍

03

PHP服务docker化

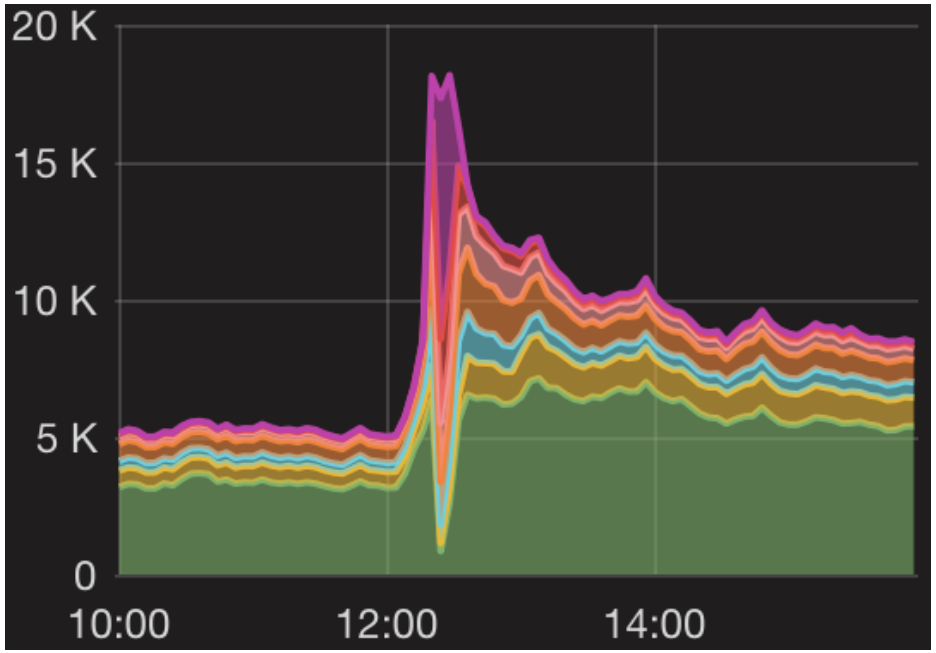
04

弹性扩容

01 背景与挑战

业务现状

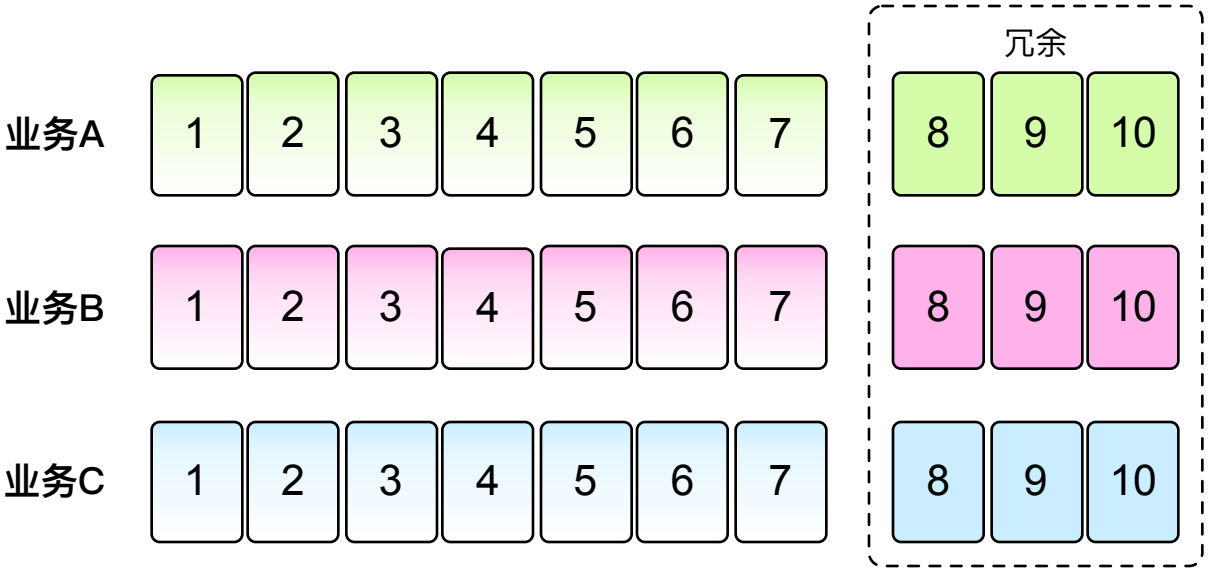
- 突发的热点事件
  - “白百合出轨”、“周一见”、
  - “宝宝离婚”、“女排夺冠”
- 大型活动及三节保障
  - 红包飞
- Push推送
  - 运营的各种站内，站外push



话题业务特点

1. 平时流量稳定，每日峰值波动较小
2. 热点事件push，10分钟内流量可以达到push前的2-3倍
3. 在达到顶峰后，流量在约1小时后恢复到push前水平

传统手段：设备冗余



一些问题

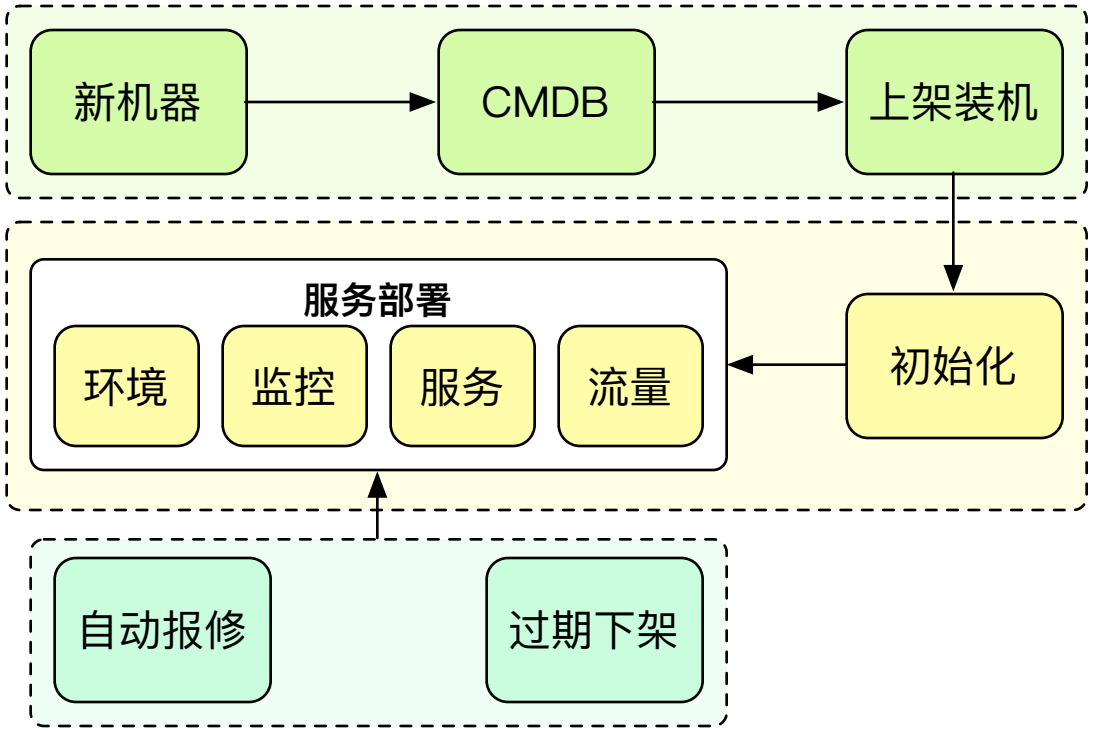
- ✓ 机架位不足，上千台服务器的库存不足。
- ✓ 千万级采购成本
- ✓ 采购周期长、运行三个月只为一晚

各业务提前申请足够的设备保证冗余。

···●传统手段：服务降级●···

- ✓降级非核心以及周边业务。
- ✓极端情况PC主站只保留主feed。

扩容繁琐

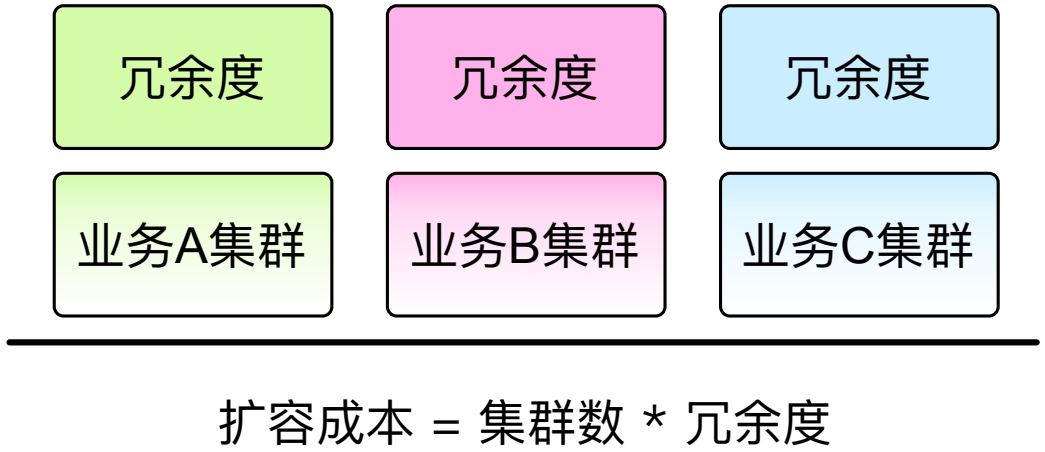


涉及团队

- 采购
  - 机器采购
- 基础运维
  - 操作系统
  - 网络
- 业务运维
  - 环境
  - 监控
  - 服务
  - 流量

设备运营成本高

- ✓各业务利用率不同，导致设备未能得到充分利用。
- ✓各业务模型不同，峰值时间不同，不能进行错峰使用。
- ✓每个业务池都有自己的冗余，多个业务池会造成极大的成本压力。



扩容成本 = 集群数 * 冗余度

总结

问题

- 申请冗余设备时周期长，服务扩缩容繁琐。
- 设备运营成本高。
- 服务负载过高时，只能进行服务降级。

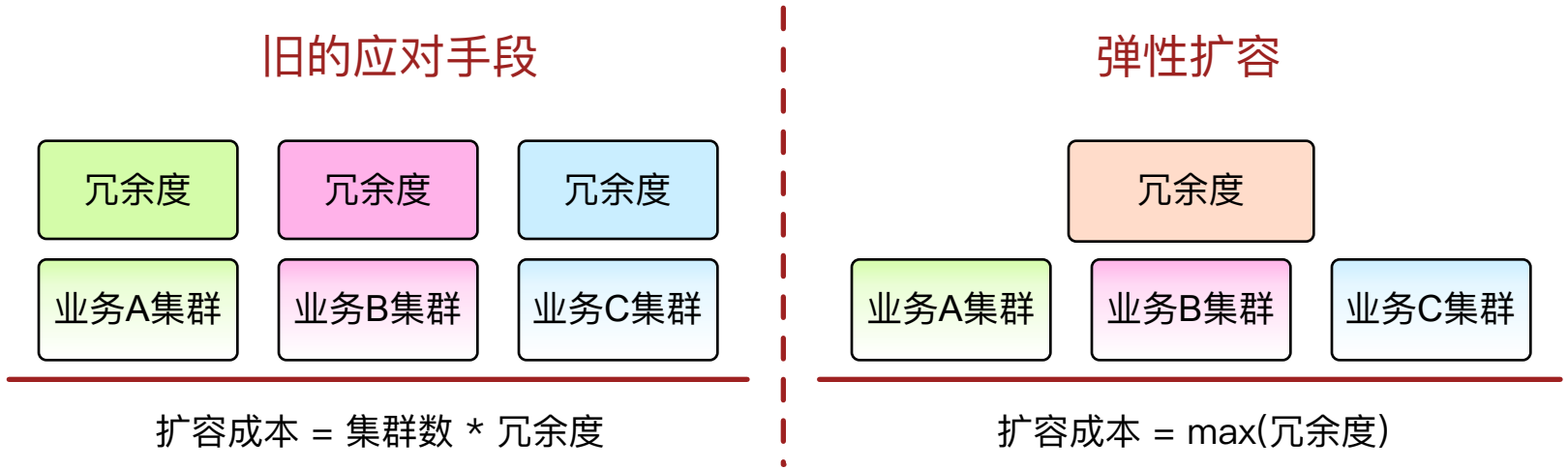
挑战

- 降低设备运营成本
- 实现业务的弹性扩容部署。

02 DCP平台介绍

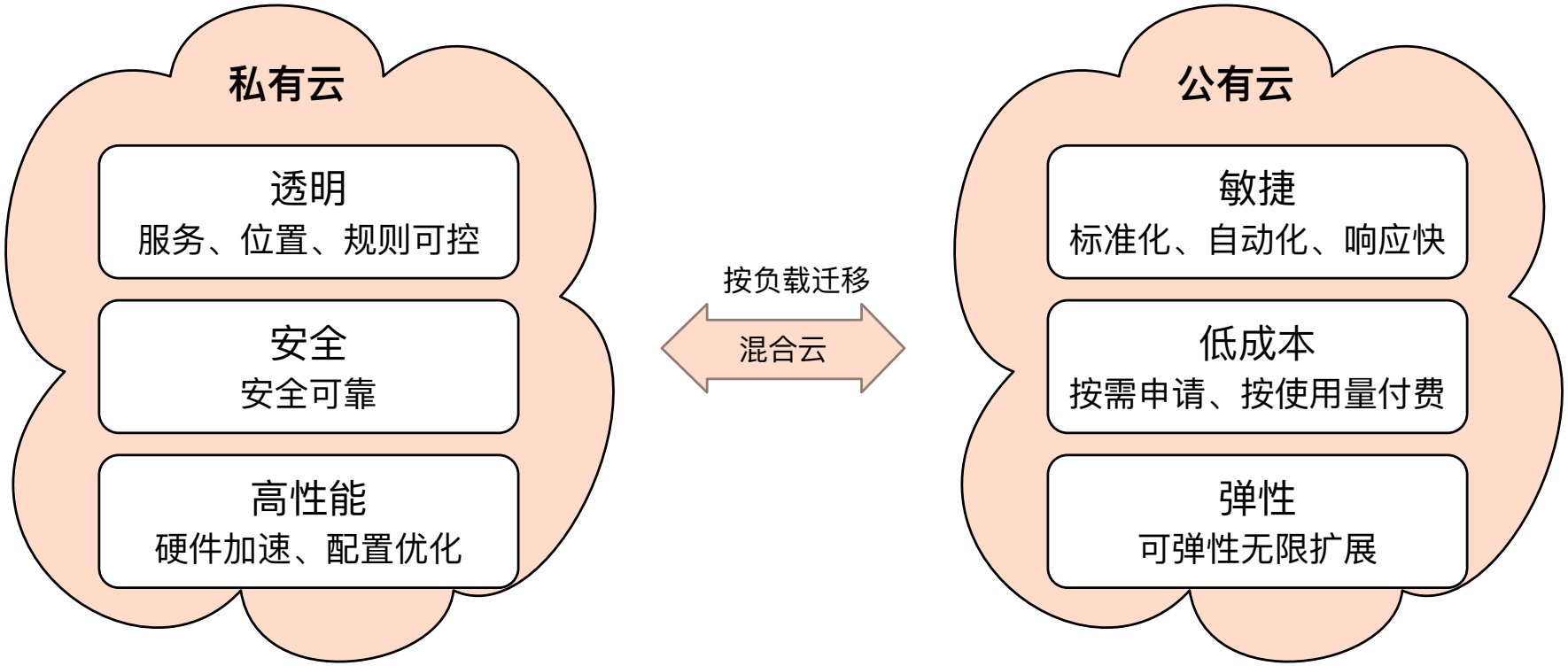
主要思想

业务弹性调度



容器化来抹平运行环境的差异

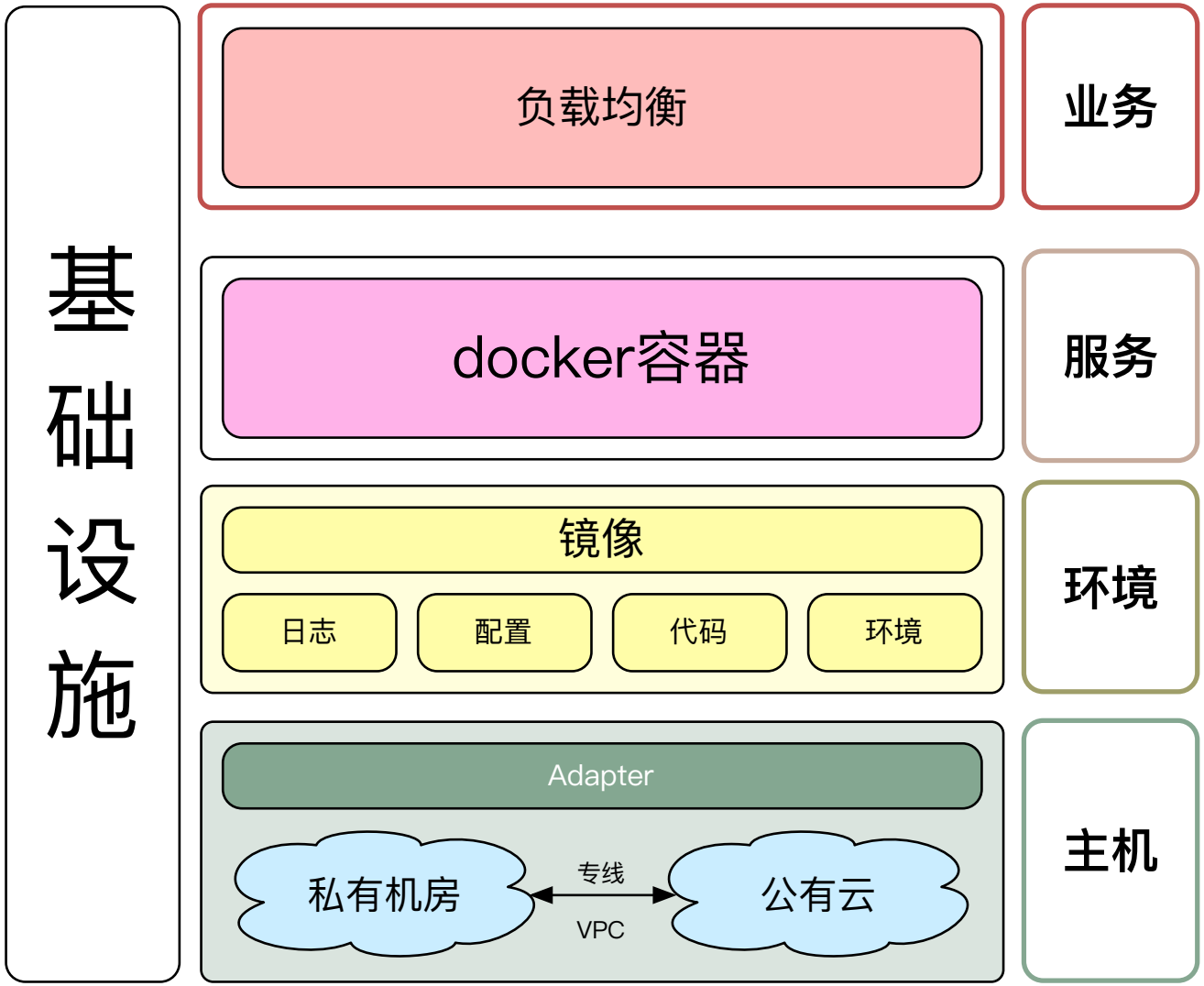
基础设施支持跨云



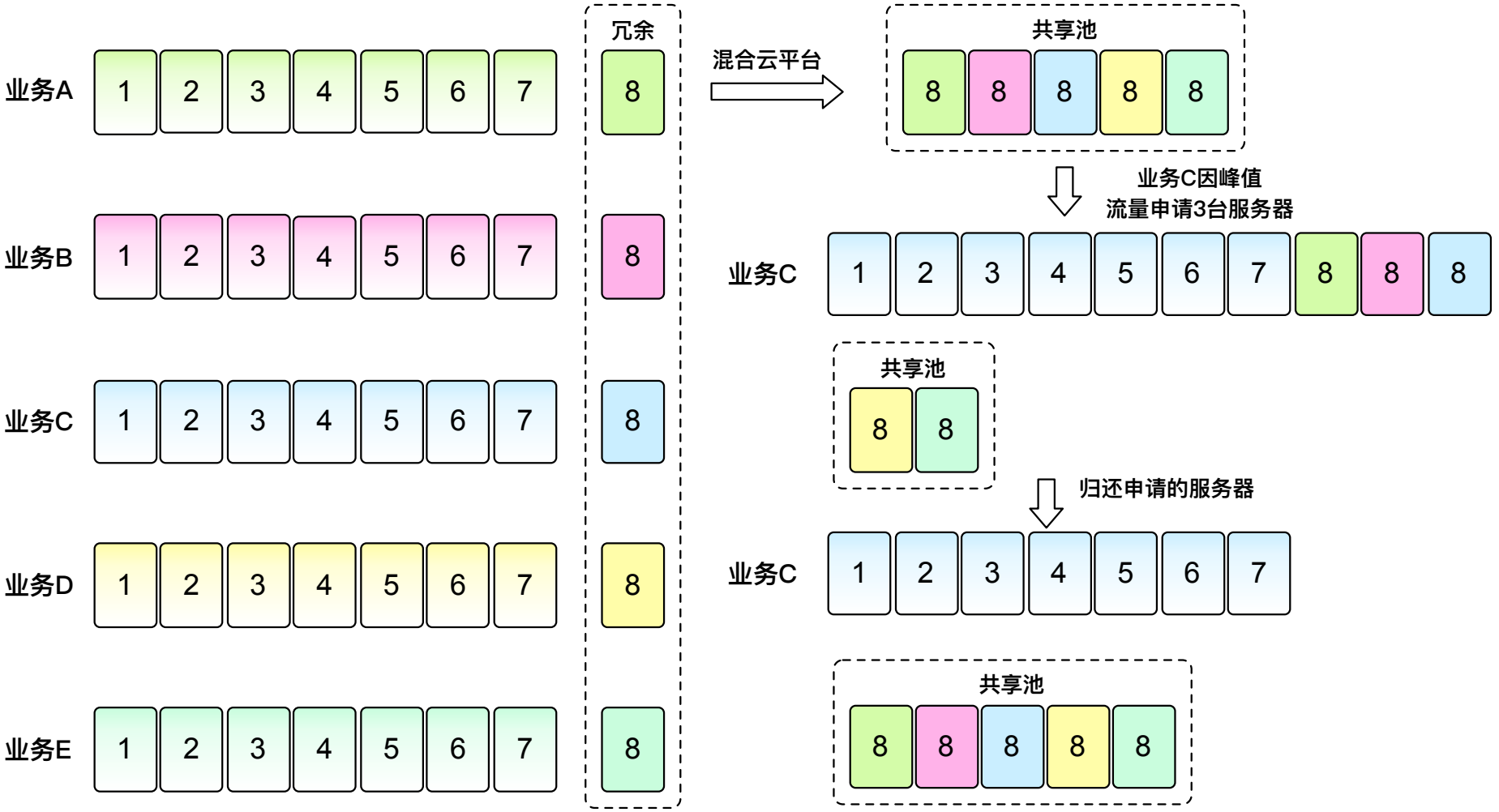
固定工作负载

弹性工作负载

DCP系统架构图


基础设施
负载均衡
业务
docker容器
服务
镜像
日志
配置
代码
环境
环境
Adapter
私有机房
专线
VPC
公有云
主机

私有云“化零为整”

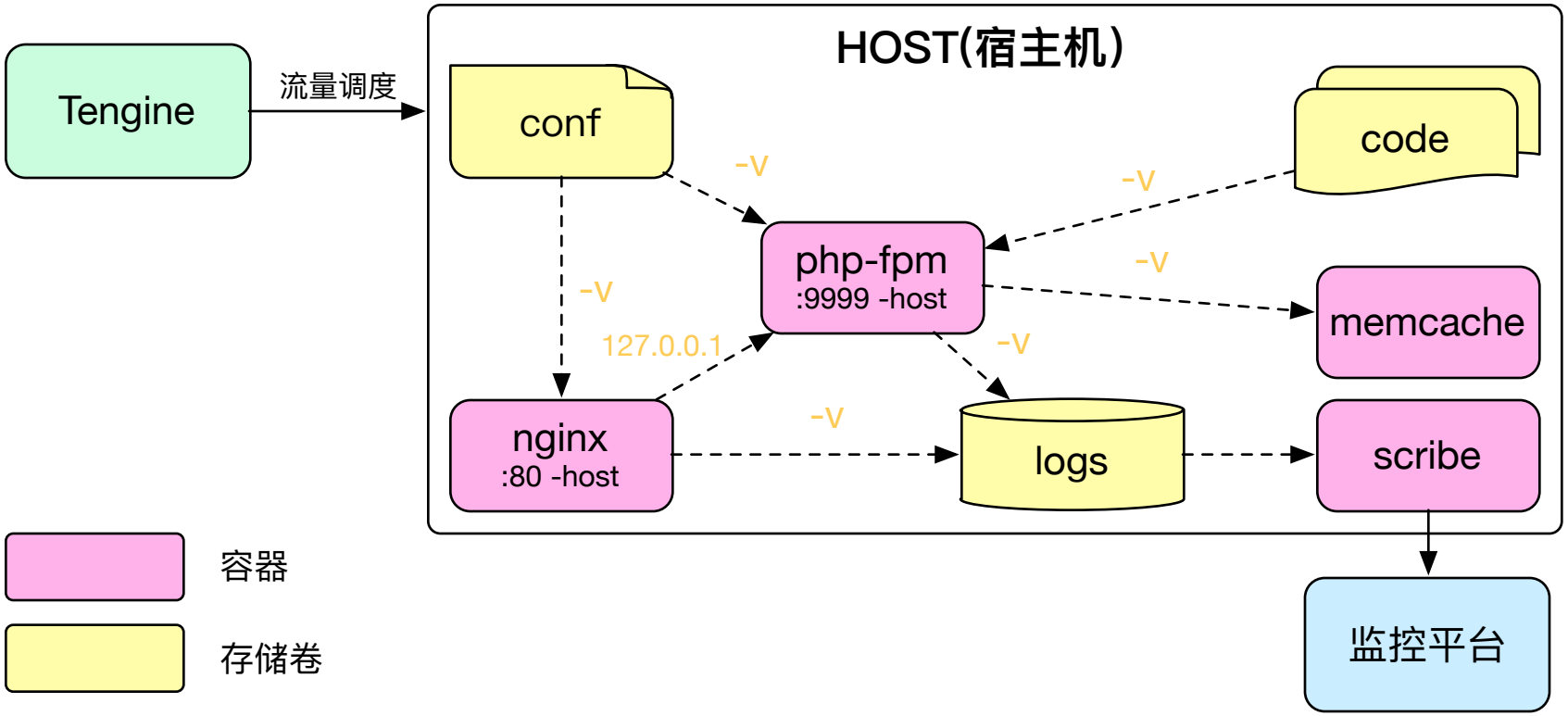

业务A
1
2
3
4
5
6
7
冗余
8
业务B
1
2
3
4
5
6
7
8
业务C
1
2
3
4
5
6
7
8
业务D
1
2
3
4
5
6
7
8
业务E
1
2
3
4
5
6
7
8
混合云平台
共享池
8
8
8
8
8
业务C因峰值
流量申请3台服务器
业务C
1
2
3
4
5
6
7
8
8
8
共享池
8
8
归还申请的服务器
业务C
1
2
3
4
5
6
7
共享池
8
8
8
8
8

03 PHP服务docker化

服务docker化

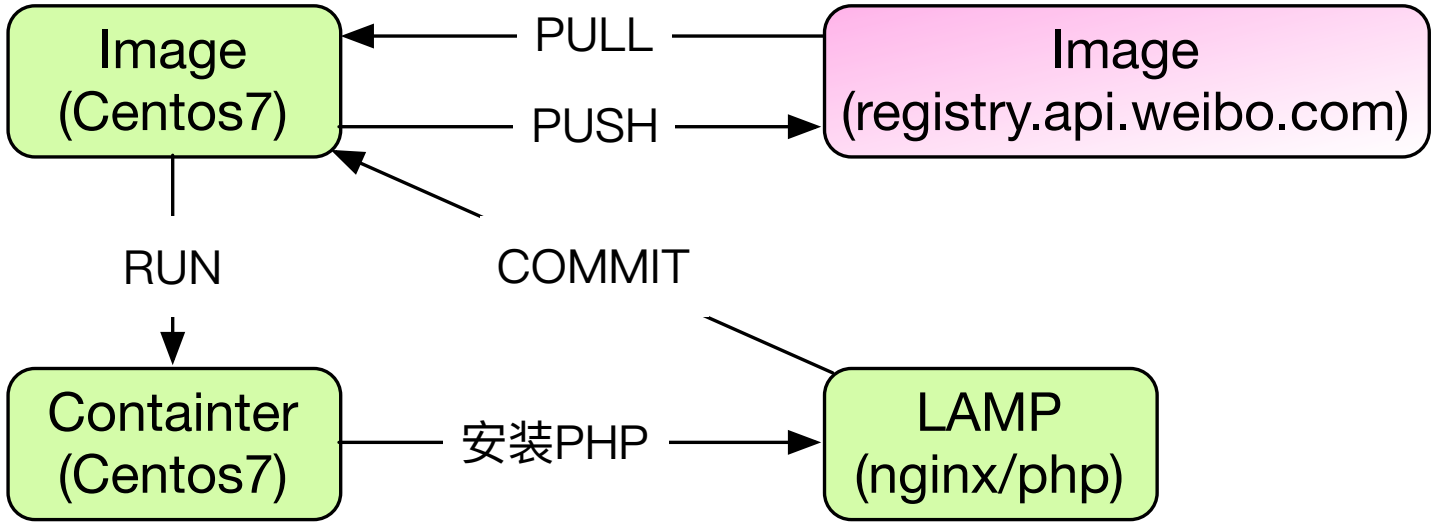
- docker服务启动快。
- Docker镜像一次制作，多次快速部署。
- 尤其适合动态扩容部署。

部署方案设计



1. php服务包括nginx、php-fpm、memcache、scribe等几大组件。
2. php组件容器单独部署。
3. 代码、配置、日志等经常变更部分通过挂载的方式和docker容器互动。

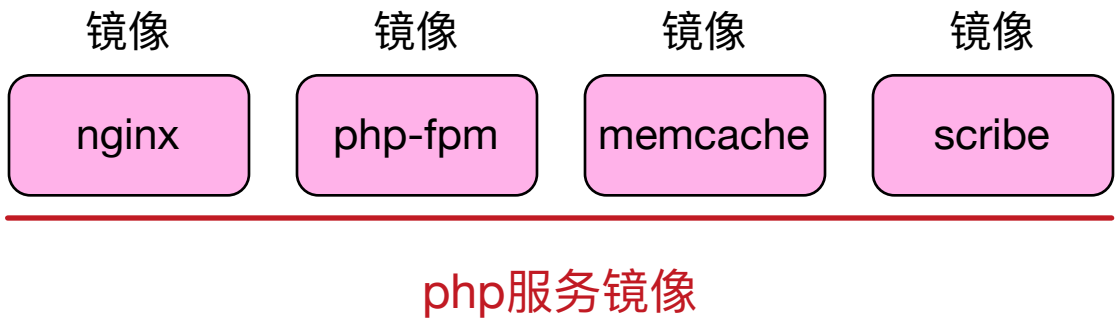
镜像制作



镜像制作步骤

1. 从镜像仓库里拉去CentOS作为基础镜像。
2. 运行镜像
3. 在运行容器中安全PHP环境相关软件包。
4. 提交修改并推送至仓库。
5. PHP服务镜像制作完毕。

镜像方案



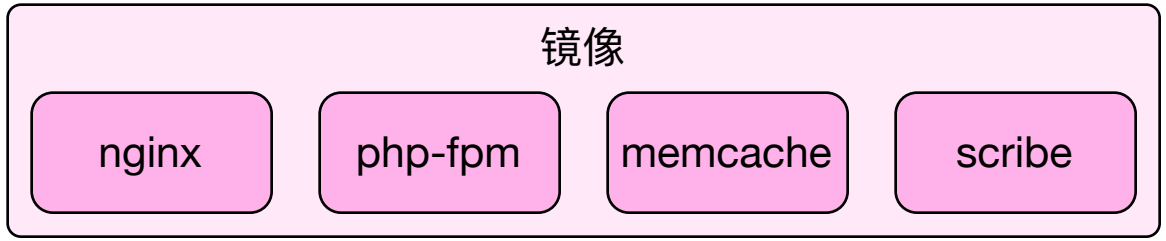
方案

1. 基于CentOS6.7来制作镜像。
2. 将PHP服务组件拆成了独立的镜像。

缺点

1. 镜像占用空间太大，每个镜像都超过1G大小。
2. 拉去镜像耗时太久，占用带宽较高。

镜像方案

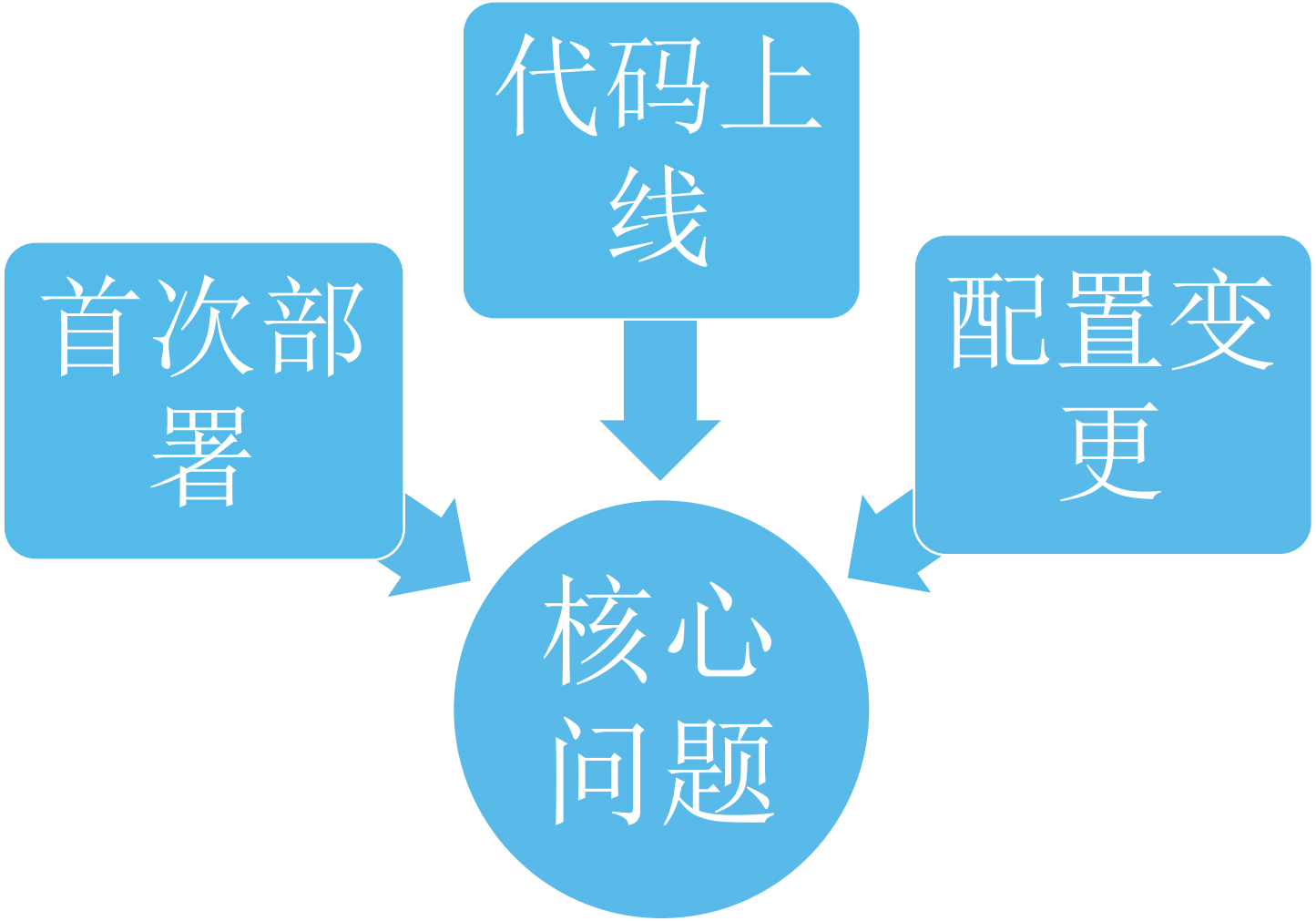


php服务镜像

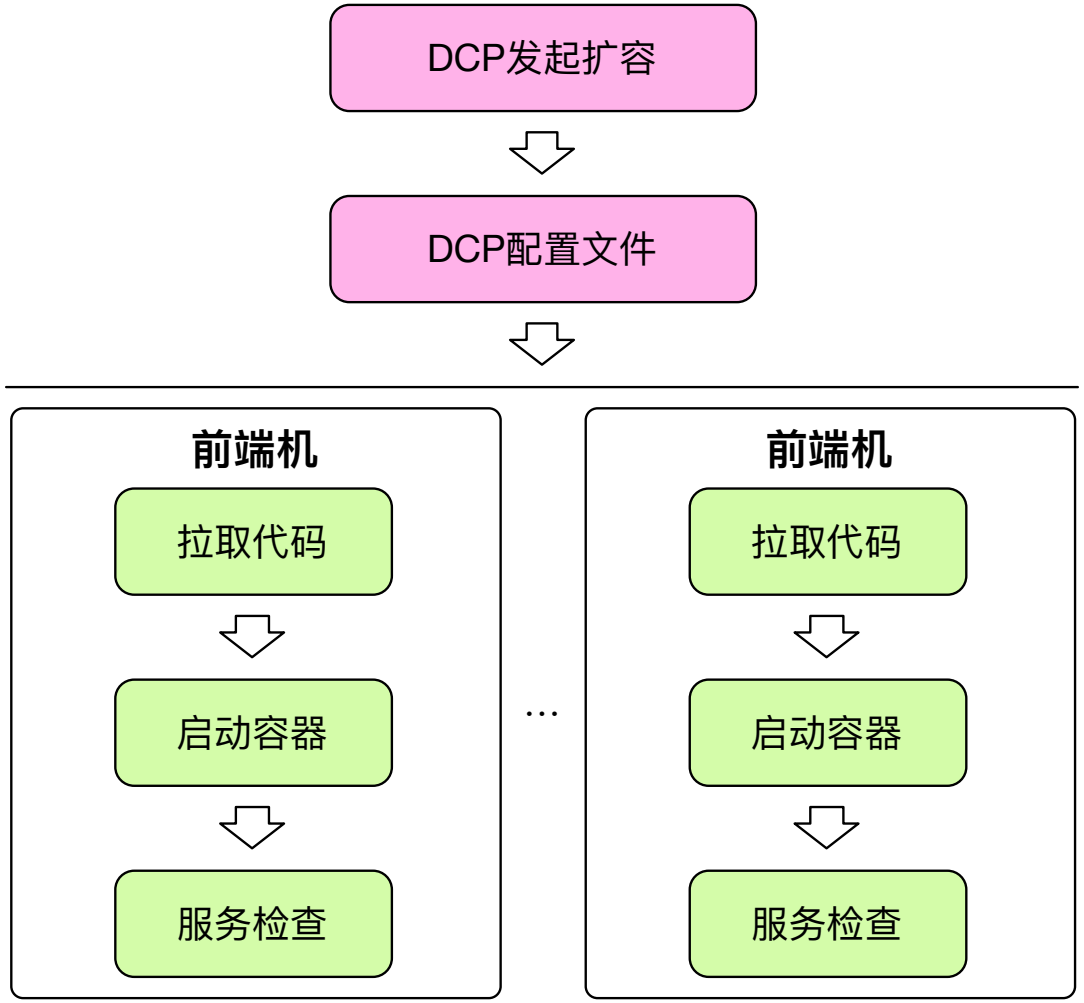
1. 将PHP服务相关的组件制作成一个镜像。
2. 服务通过容器命令来启动。

```
docker run php7.img:1.6.3 --name nginx /usr/local/sinasrv2/sbin/nginx
docker run php7.img:1.6.3 --name php /usr/local/sinasrv2/sbin/php-fpm
```

部署的核心问题

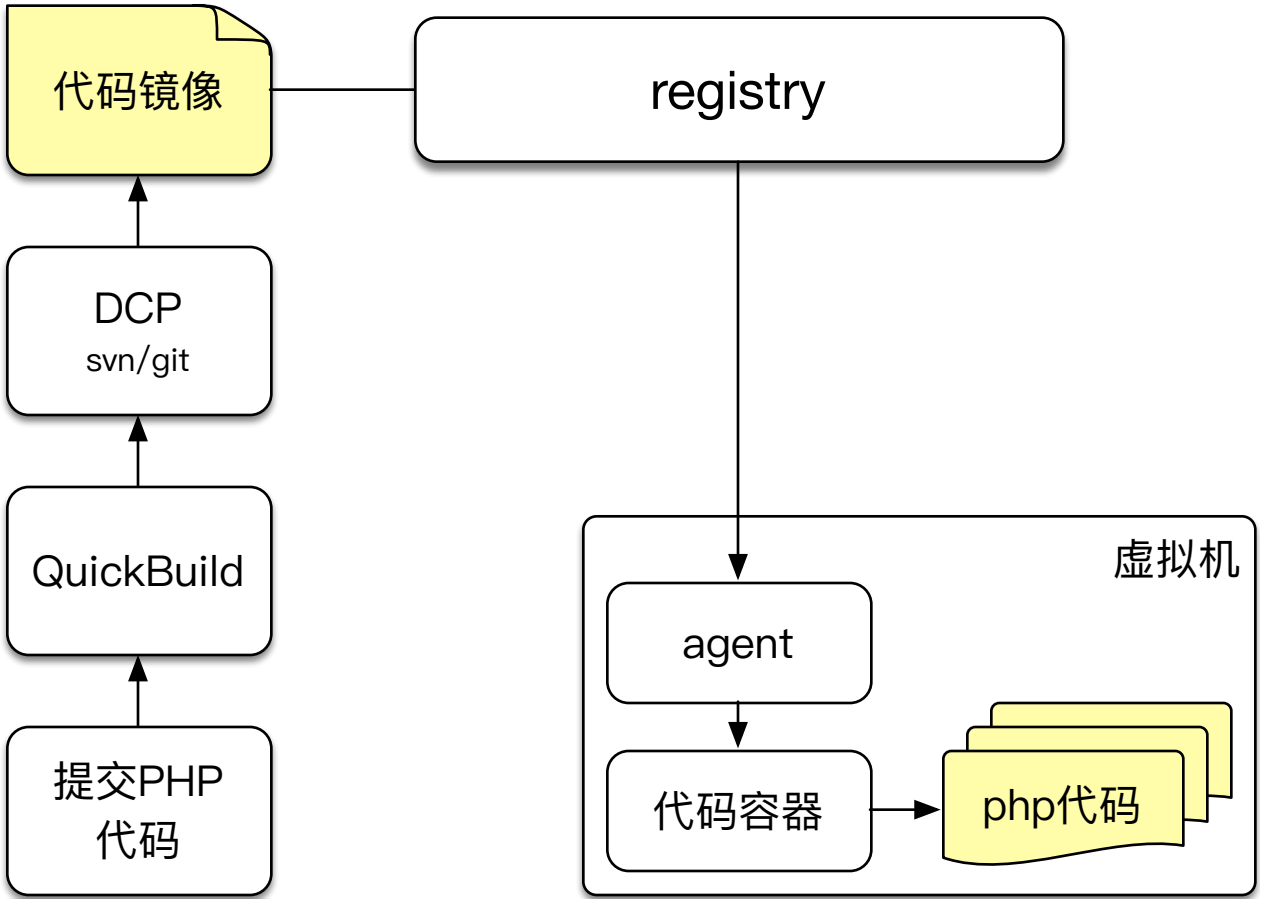


首次部署服务



通过下发配置文件、上线代码、启动容器完成服务部署。

代码上线



- ✓通过镜像完成上线
- ✓代码镜像使用busybox为基础，大小仅1M

创建代码镜像

1. Dockerfile

```
FROM registry.x.weibo.com/qinglong/busybox
RUN mkdir -p /code/x.weibo.com
ADD x.weibo.com  /code/x.weibo.com
```

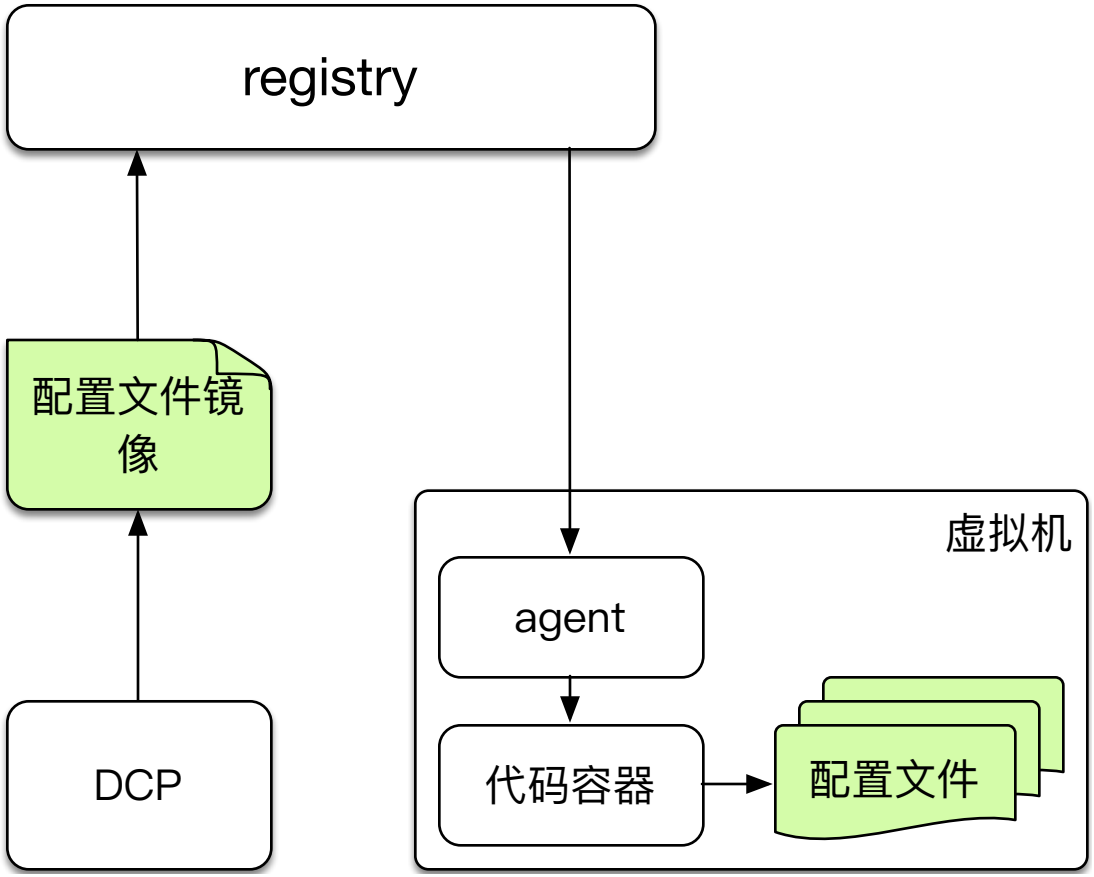
2. Build

```
registry.x.weibo.com/codeimg_x_weibo_com_git:324234
```

3. 下载代码镜像、启动容器、拷贝代码

```
docker pull registry.x.weibo.com/codeimg_x_weibo_com_git:324234
docker run -name=code_container -t -i -d /phpcode codeimg_x_weibo_com_git:324234
docker exec code_container cp -R /phpcode /code/x.weibo.com
```

配置文件更新



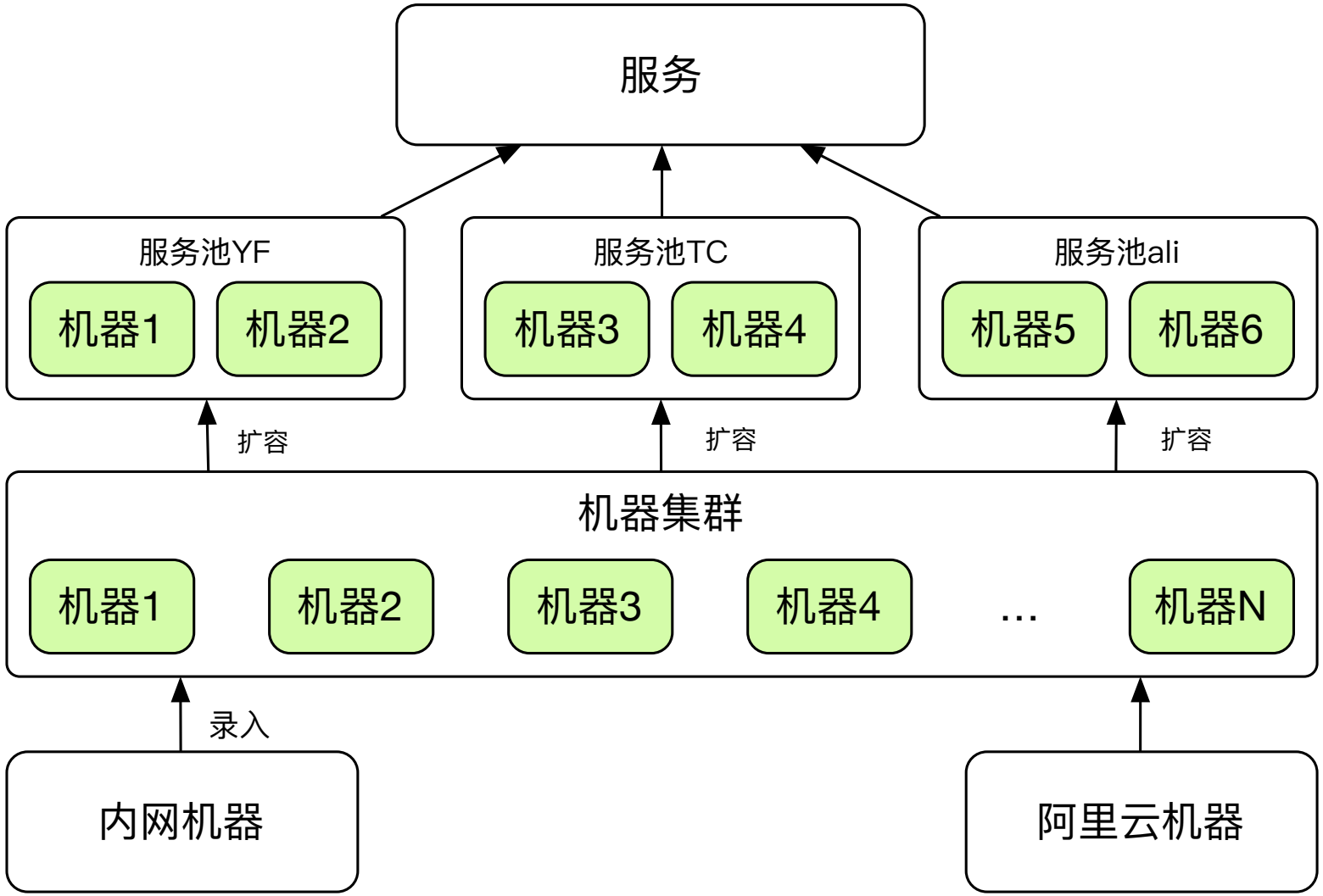
- ✓ 配置文件制作成docker镜像
- ✓ 每台机器拉取镜像，替换配置文件
- ✓ 自定义脚本执行reload

一些细节

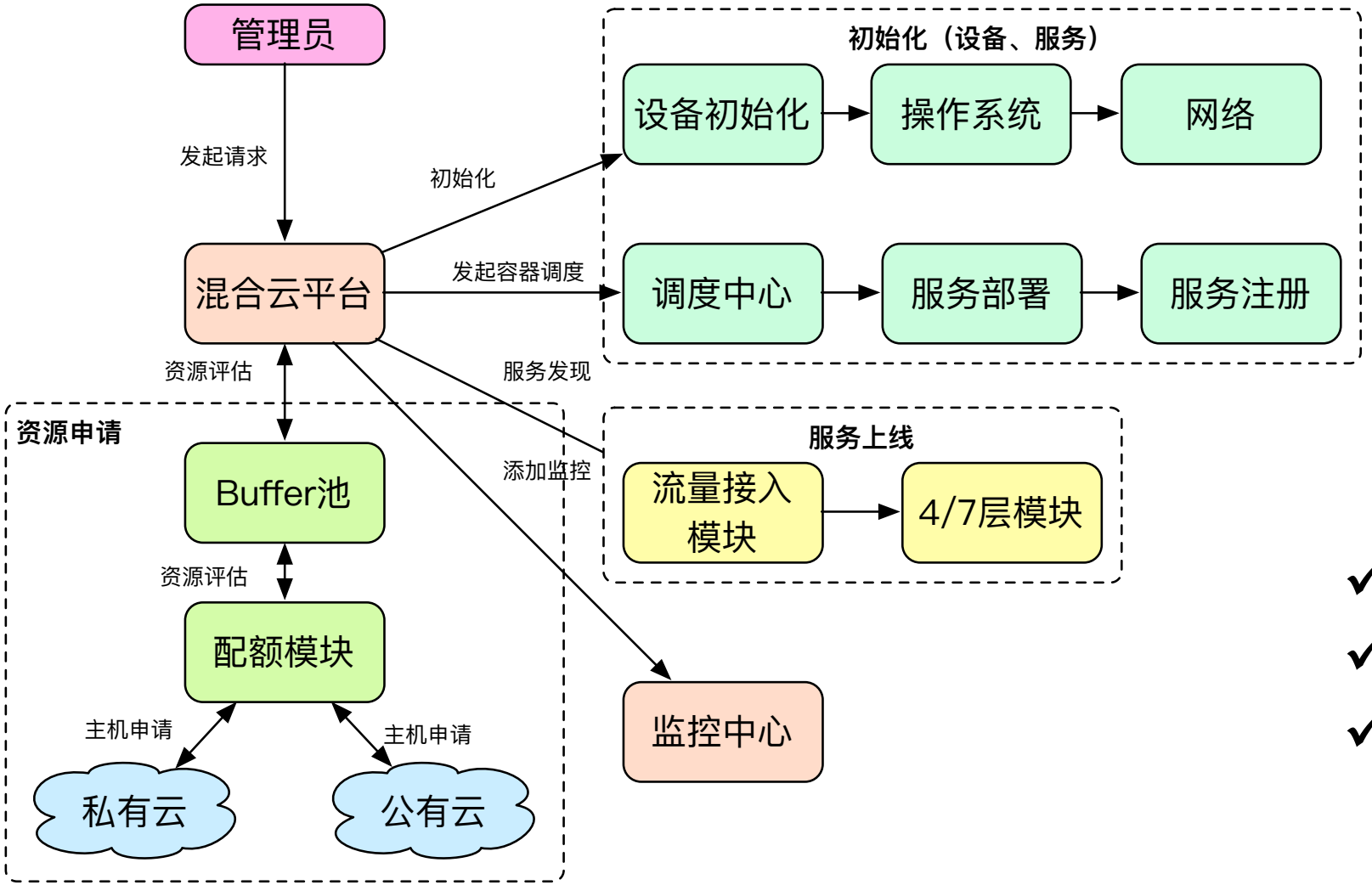
- ✓docker化后宿主机运行在centos 7
- ✓内核升级到3.10
- ✓容器中的启动命令要是前台启动
- ✓经常变更的部分放在镜像外通过volume挂接容器
- ✓网络模式选用host网络模式
- ✓容器的reload或优雅重启采用docker exec xx reload方式

04 弹性扩容

服务、服务、集群


服务
服务池YF
机器1
机器2
服务池TC
机器3
机器4
服务池ali
机器5
机器6
扩容
扩容
扩容
机器集群
机器1
机器2
机器3
机器4
...
机器N
录入
内网机器
阿里云机器

扩容流程



- ✓ 资源申请
- ✓ 环境初始化
- ✓ 服务上线

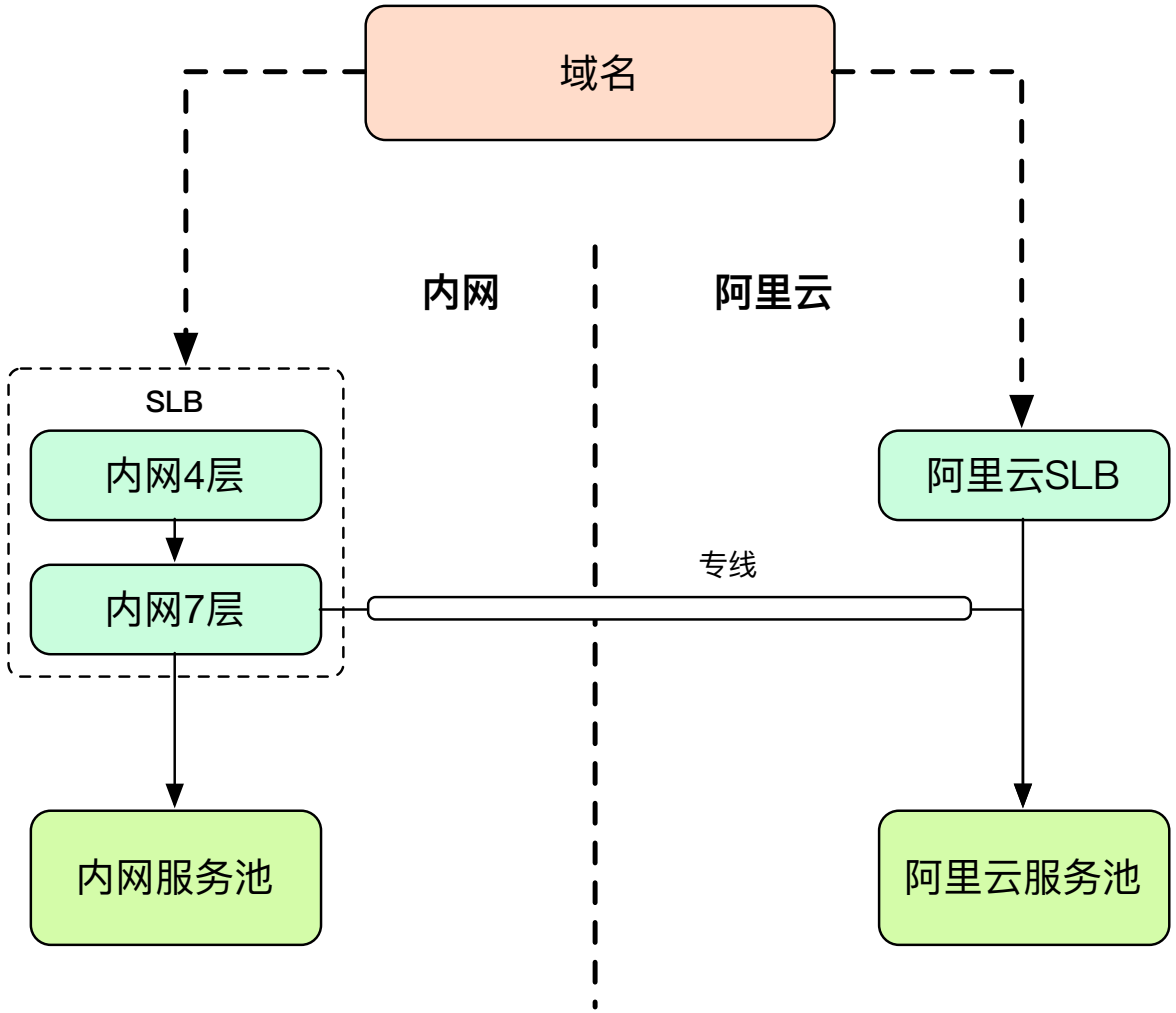
扩容模板

镜像地址	image:1.6.1
容器参数	-v /data1:/data1 -v /data1/vhost/:/usr/ /etc/vhost -v /data1/fpm.d:/usr/lo /etc
容器命令	/usr/local/ n/nginx -g 'daemon off;'
容器名	nginx_container

镜像地址	bo_tech_pop.image:1.6.1
容器参数	-v /data1:/data1 -v /data1/vhost/:/ tc/vhost -v /data1/fpm.d:/ /etc
容器命令	/usr/loca bin/php-fpm -F
容器名	php_container

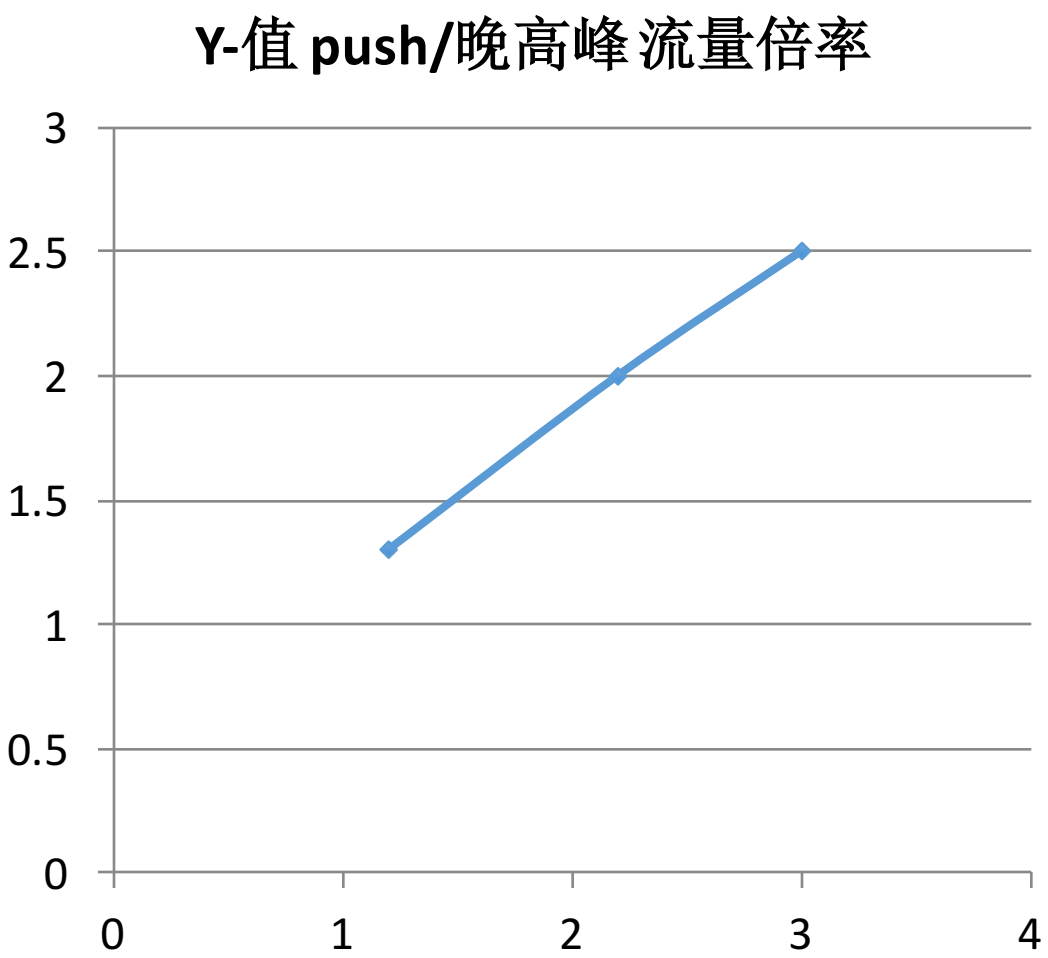
镜像地址	_tech_pop.image:1.6.1
容器参数	-d --net=host
容器命令	/usr/local/ emcached -u www -m 4096 -l 127.0.0.1 -p 6666 -c 20000
容器名	memcached_container

流量切换



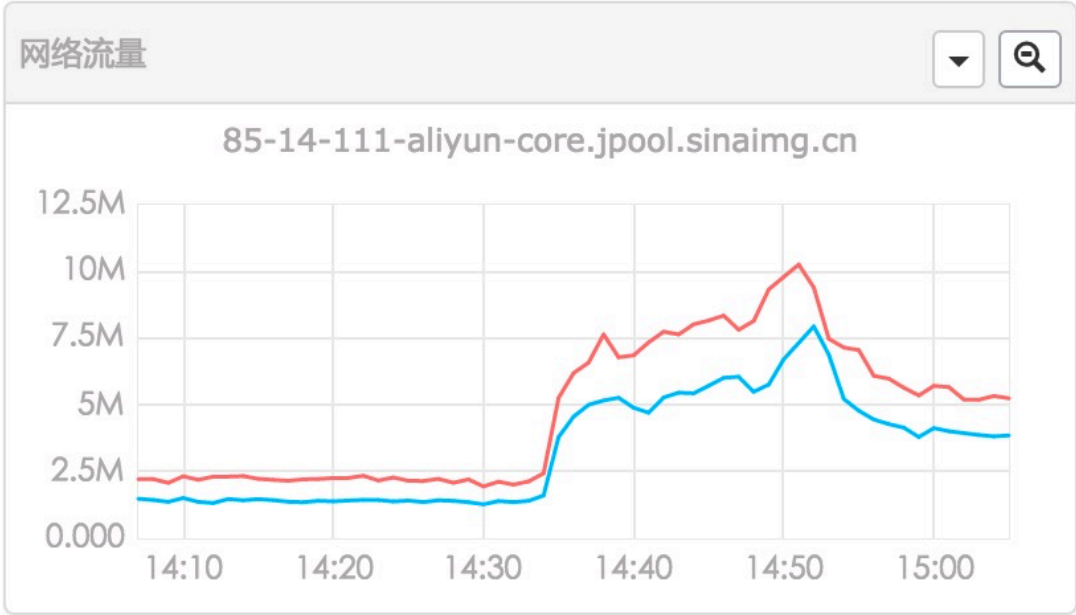
- ✓内网7层添加阿里云设备
- ✓域名解析到阿里云SLB

弹性容量的考虑



- 以统计的打开率数据为依据，push打开率和流量较晚高峰倍率存在对应关系
- “宝宝”事件的push打开率为3%，对应流量会上涨到晚高峰的2.5倍，差额需要约30台服务器
- 留出冗余，申请50台预算，可以做到对打开率达到约4%的超热点事件push的运营支持

扩容效果



✓一键式扩容
✓15分钟、峰值被明显削平

总结

- LNMP服务docker化，制作PHP服务相关镜像。
- 结合DCP平台完成PHP服务的首次部署、配置更改、代码上线等。
- 目前主站TV视频站、头条问答、话题、红包飞、通行证等LNMP项目已全量部署，方便弹性扩容。
- PC主站部分服务已部署完成。

谢谢！