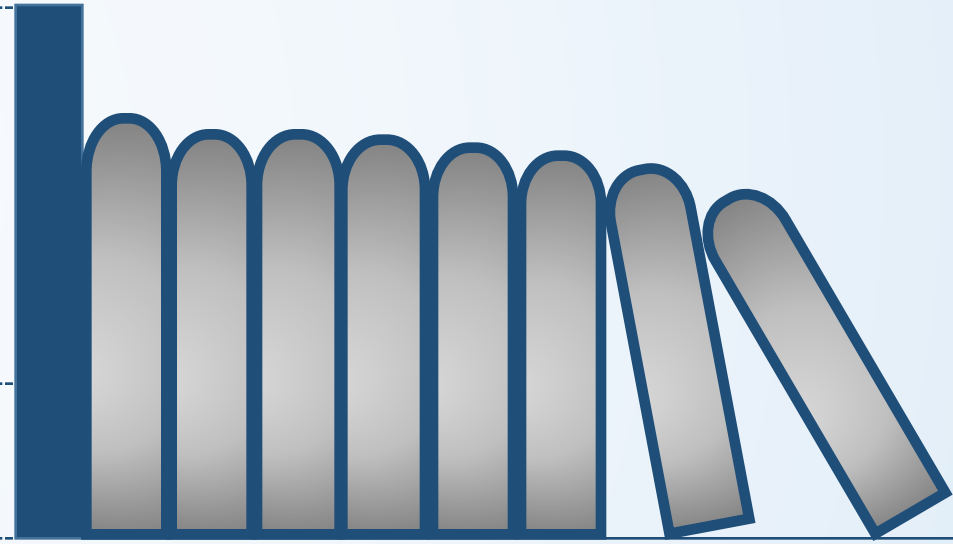
容器平台架构设计案例分析

主讲人：宋小金



目录

预期收获

- 认知容器技术的价值
- 理解容器平台的架构设计思路，能结合业务场景输出解决方案
- 了解容器技术的关键点和坑

容器的优势

1. 更高效的虚拟化 --- 性能损耗少，调度颗粒细
2. 更快速的启动时间 --- 进程形式，秒级启动
3. 一致的运行环境 --- 镜像
4. 更快速的交付和部署
5. 更轻松的迁移
6. 更轻松的维护和扩展

兼具IaaS的灵活和PaaS的便利

总结：

任何IT技术兴起都可从三个维度来分析：提高稳定性，提升效率，节省资源成本

Kubernetes的核心 —— 编排

- 资源弹性，命名空间，多租户，额度管理，资源隔离
- 应用编排，服务发现，副本数控制，探针机制，生命周期管理及钩子
- Deployment/StatefullSet/DaemonSet等，发布部署机制，支持有/无状态应用上线，更新，回滚，多种发布策略
- 存储自动挂载，资源自动选择，网络策略控制，拓扑亲和/反亲和
- 轻量化部署，高部署密度，应用驱逐机制，提升资源利用率

前景 —— all in k8s

现状：目前kubernetes+docker主要用于构建微服务管理平台及AI训练深度学习平台，ToB的容器平台，大厂，创业公司基本上都是PaaS平台及AI训练平台

前景：all in k8s，kubernetes会演变成一个资源调度引擎，统管资源池，快速资源交付，充分资源共享，提升资源利用率

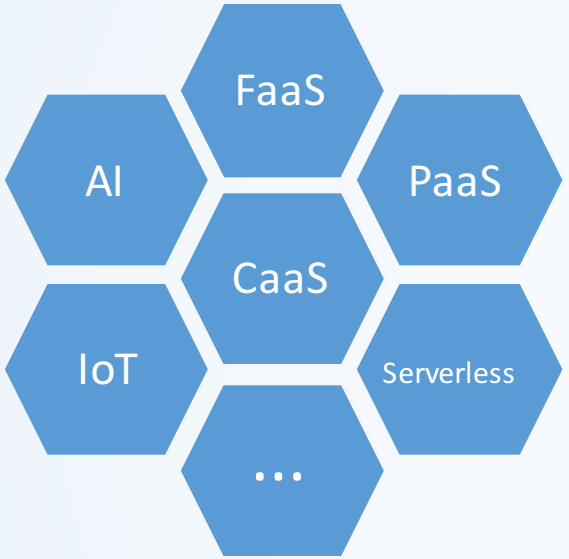
时代背景：

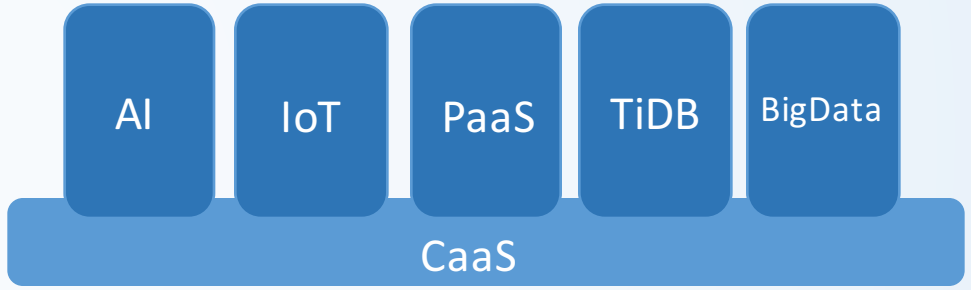
AI， 5 G，物联网，边缘计算 -- 新一代基础设施，连接方式转换

消费互联网转向产业互联网，ToB产业兴起，技术进一步分工

IT架构从大中台转向工字型架构，即前端 / 数据层厚，中台薄，终端多样化，数据层在资源弹性，横向扩展能力的赋能下，承接更多中间件能力，这需要Kubernetes的加持

容器平台集成模式

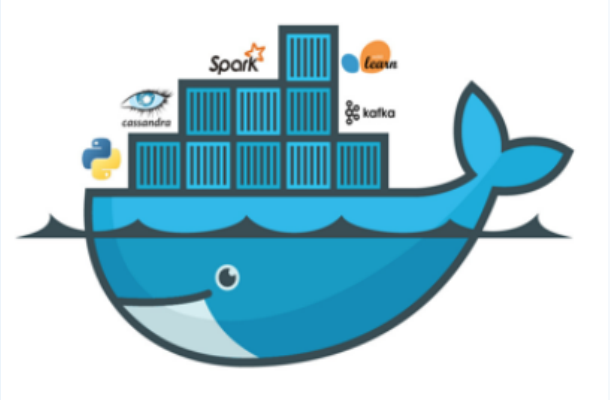




基于容器平台构建其它平台产品，Kubernetes作为编排引擎，CaaS作为IaaS和PaaS的中间层

容器服务设计理念

能力分层



DevOps

PaaS

CaaS

EKS集群服务

IaaS服务

满足不同产品，不同用户的业务需求

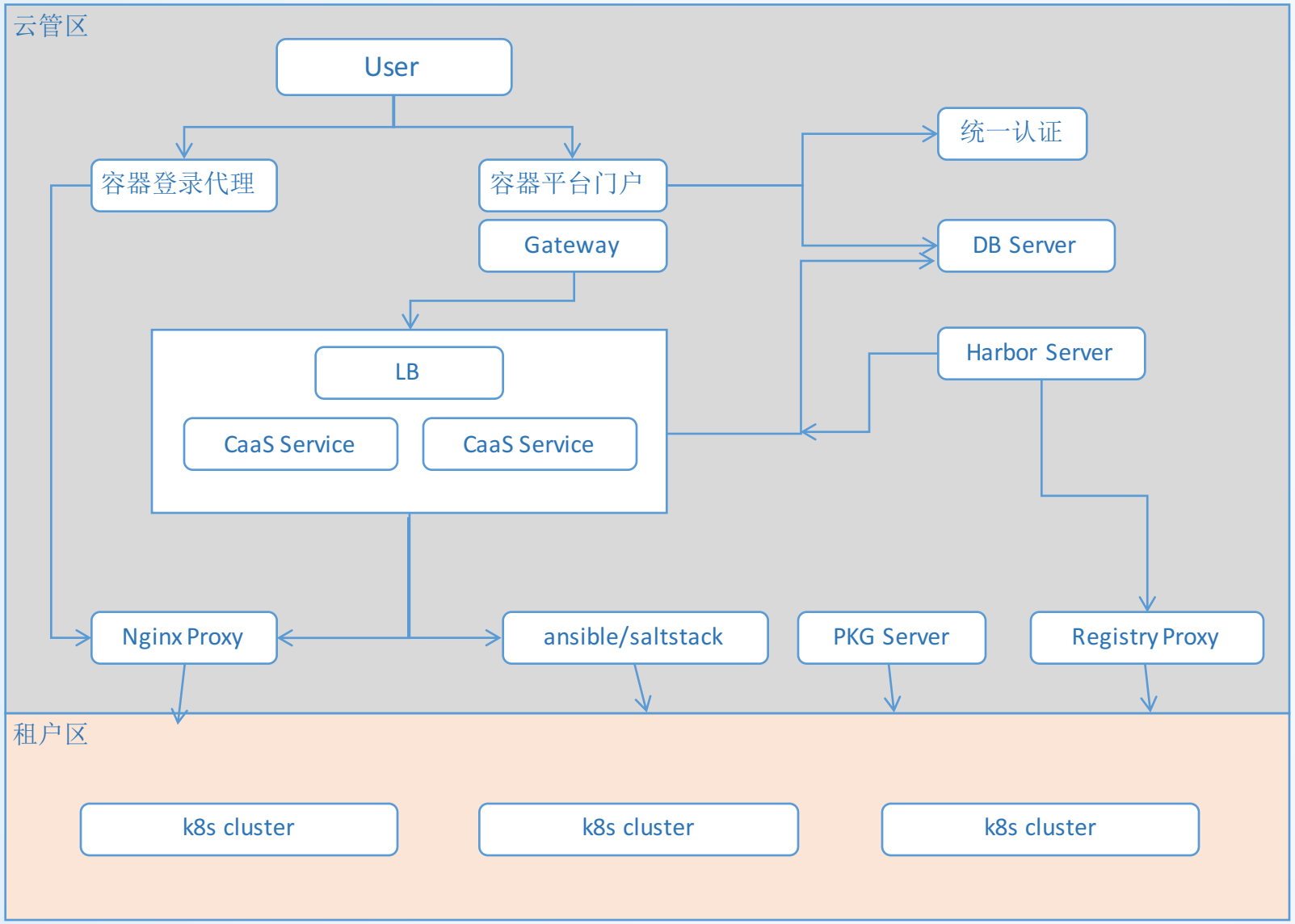
容器平台功能明细

	能力层	功能	备注
1	EKS	集群管理	集群创建删除，版本升级，额度管理，集群弹性扩容，证书管理，命名空间管理
2		节点管理	添加删除结点，登录配置节点，标签/污点管理
3		网络管理	Service / Ingress，网络策略（NetPolicy）,VPC / CIDR
4		存储管理	PV / PVC / StorageClass
5		插件管理	Addon部署升级，即可CoreDNS，存储插件，监控插件，日志采集，Ingress结点
6		监控日志	容器集群的监控告警，日志采集，数据展示
7	CaaS	镜像仓库	镜像生命周期/版本管理，多租户，灾备
8		编排调度	Helm chart模板化部署，调度策略
9		模板商城	私有模板，公有模板
10		应用市场	一键应用容器化部署
11		配置中心	ConfigMap / Secret / 集中式配置中心

容器平台功能明细

	能力层	功能	备注
1	PaaS	持续集成 / 流水线	编译构建打包，打镜像，单元测试，代码检查，安全扫描
2		应用管理	应用生命周期管理，应用规格，启停，环境变量，健康检查，优雅关闭
3		部署策略	滚动，蓝绿，灰度
4		弹性伸缩	资源池弹性（Cluster AutoScale），水平实例弹性（HPA）
5		服务治理	调用链，熔断限流，流量调度，注册发现
6		权限管理	基于角色的权限管理
7		操作审计	操作记录留存
8			
9			
10			
11			

容器部署架构图



部署模式：

1、传统模式

2、k8s on k8s模式

传统模式基于命令通道来部署kubernetes二进制组件，创建集群

k8s on k8s模式是将组件容器化，Master相关组件直接利用k8s集群管理直接部署

集群管理

1 服务选型 — 2 创建节点 — 3 安装插件 — 4 规格确认 — 5 完成

* 计费模式：包年/包月 | 按需计费

* 区域：华南-广州

不同区域的云服务产品之间内网互不相通；请就近选择靠近您业务的区域，可减少网络时延，提高访问速度。

* 集群名称：test-cluster

* 版本：v1.11.7 | v1.13.10　集群版本升级说明

* 集群管理规模：50节点 | 200节点 | 1,000节点

* 高可用：是 | 否　集群创建完成后，高可用模式及普通模式之间不可变更，请按实际使用场景选择。

* 虚拟私有云：vpc-k8s(192.168.0.0/16)　创建虚拟私有云 完成创建后点击刷新按钮。

了解虚拟私有云、子网和集群的关系

* 所在子网：subnet-k8s(192.168.1.0/24)　创建子网 完成创建后点击刷新按钮。

当前子网剩余可用IP数：250。高可用集群控制节点会额外使用4个IP。

* 网络模型：容器隧道网络 | VPC网络

* 容器网段：自动选择

172 . 16 . 0 . 0 / 16

当前容器网段最多支持65533个实例（估算值，实际存在偏差），此参数在集群创建后不可更改，请谨慎选择。

建议使用网段：10.0.0.0/12~19，172.16.0.0/16~19，192.168.0.0/16~19

服务网段：使用默认网段 | 手动设置网段

10 . 247 . 0 . 0 / 16

鉴权方式：RBAC

开启RBAC能力后，设置了细粒度权限的子用户使用集群下资源将受到权限控制

我已知晓上述限制并阅读CCE权限管理说明

认证方式：认证能力增强　默认开启X509认证模式

集群描述：选填，输入相应的描述

0/60

高级设置

多可用区　多可用区模式支持集群管理面多可用区容灾，但是对于集群性能有所损耗。

服务转发模式：iptables | ipvs

集群管理

集群管理 › **集群详情 (test-cluster)**

删除

test-cluster

混合集群

正常

基本信息

集群ID	7f13016b-f635-11e9-82...
集群版本	v1.13.10-r0
集群规格	cce.s1.small | 50节点 | 通用
Docker 版本	18.09.0.15
创建时间	2019/10/24 16:08:43 GMT+08:00

网络

网络模型	容器隧道网络
所在VPC	vpc-k8s
所在子网	subnet-k8s
服务转发模式	ipvs
服务网段	10.247.0.0/16
容器网段	172.16.0.0/16

其他

计费模式	按需计费
认证方式	X509证书
鉴权方式	RBAC
描述	--

CPU分配率

剩余：2.86核

内存分配率

剩余：3.12GB

更多：

控制节点

test-cluster-master-1 监控

可用

CPU使用率	13.60%
内存使用率	23.00%
规格	4核 | 8GB

事件 弹性扩容 kubectl

– 请输入事件名称

产生时间	事件名称	备注	操作
2019/10/24 16:14:09 GMT+08:00	为控制节点创建备份策略成功	【集群管理】创建集群备份策略	查看详情
2019/10/24 16:13:59 GMT+08:00	插件实例[AllAddons]安装成功	【插件管理】插件安装	查看详情
2019/10/24 16:13:53 GMT+08:00	创建集群成功	创建集群	查看详情
2019/10/24 16:13:53 GMT+08:00	安装用户节点Kubernetes软件成功	安装用户节点Kubernetes软件	查看详情
2019/10/24 16:13:43 GMT+08:00	安装用户节点Kubernetes软件成功	安装用户节点Kubernetes软件	查看详情
2019/10/24 16:13:16 GMT+08:00	安装控制节点Kubernetes软件成功	安装控制节点Kubernetes软件	查看详情
2019/10/24 16:11:53 GMT+08:00	开始安装用户节点Kubernetes软件	安装用户节点Kubernetes软件	查看详情
2019/10/24 16:11:53 GMT+08:00	创建用户节点虚拟机成功	创建用户节点虚拟机	查看详情
2019/10/24 16:11:53 GMT+08:00	开始安装用户节点Kubernetes软件	安装用户节点Kubernetes软件	查看详情

集群管理

节点管理

用户指南　购买节点

删除节点　转包周期　集群：test-cluster　全部状态　节点名称或私有IP

名称	状态	所属节点池	规格	可分配CPU...	可分配内存...	私有IP地址	弹性IP地址	可用区	节点来源	操作
test-cluster-26949-cps16	可用	DefaultPool	2 核 | 4 GB	1.43	1.55	192.168.1.233	--	可用区1	创建节点	监控 标签管理 更多
test-cluster-26949-jjbl4	可用	DefaultPool	2 核 | 4 GB	1.43	1.55	192.168.1.85	--	可用区1	创建节点	监控 标签管理 更多

命名空间

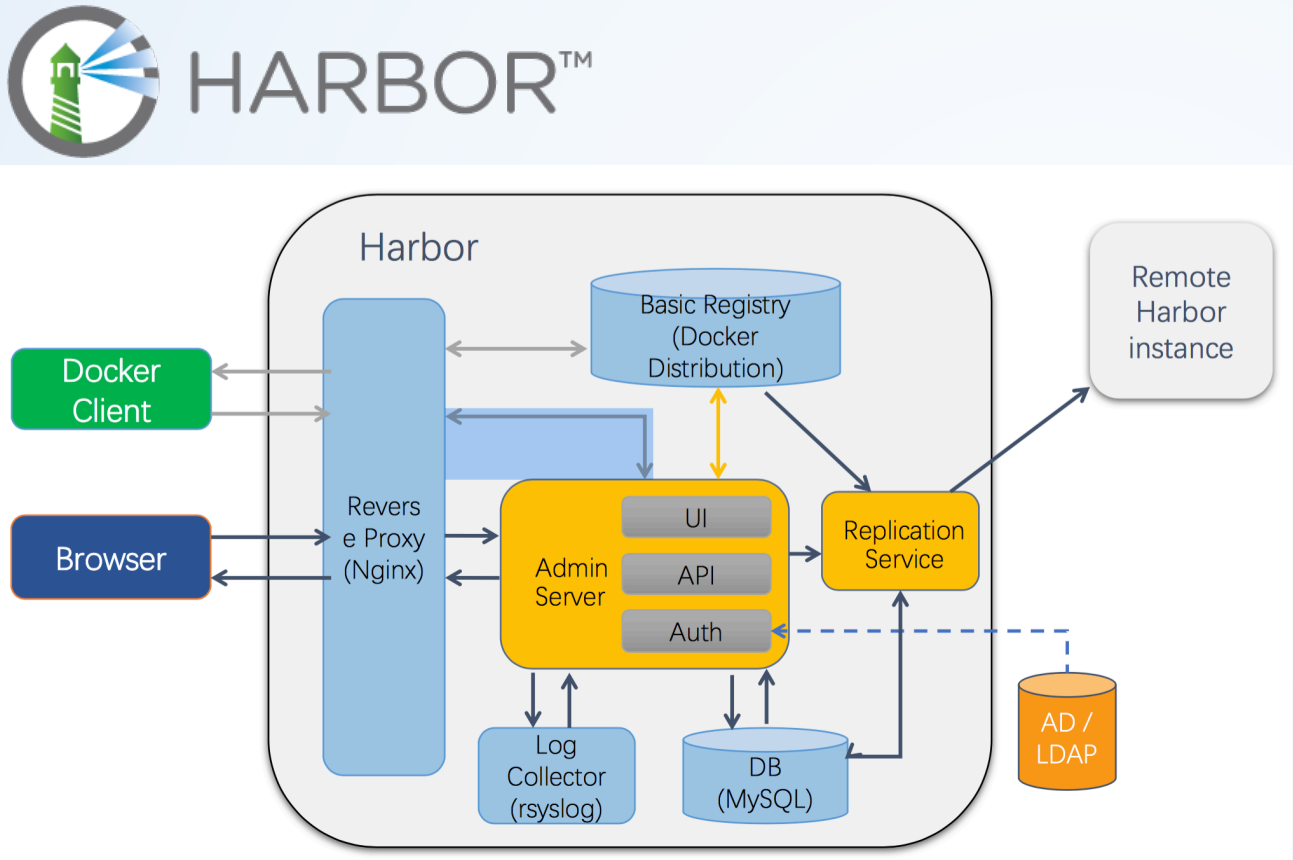
用户指南　创建命名空间

当前您还可以为该集群新增 97 个命名空间。

删除　集群：test-cluster　请输入命名空间的名称

命名空间	状态	命名空间描述	创建时间	网络隔离	节点亲和	操作
kube-system	可用		2019/10/24 16:11:54 GMT+08:00	隔离状态未开启		配额管理 删除
kube-public	可用		2019/10/24 16:11:54 GMT+08:00	隔离状态未开启		配额管理 删除
default	可用		2019/10/24 16:11:54 GMT+08:00	隔离状态未开启		配额管理 删除

镜像仓库

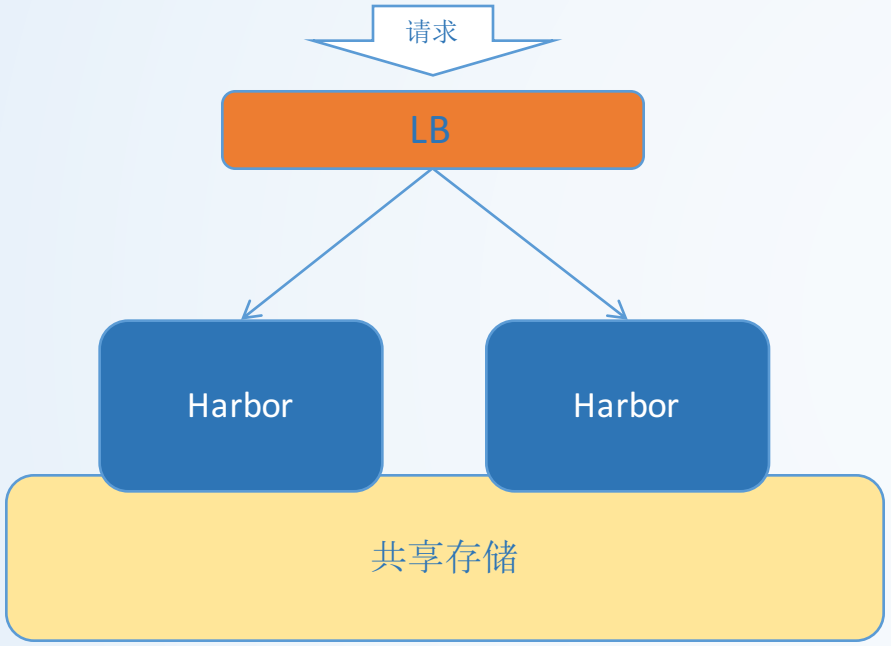


镜像仓库的要点：

- 服务高可用部署
- 镜像文件存储的HA，容量可扩容
- 多租户，支持私有镜像及公共镜像
- 镜像的安全传输，签名检验防篡改
- 异地备份

Harbor除HA部署外都支持，可基于OpenAPI包装，集成到平台内

镜像仓库 -- HA部署

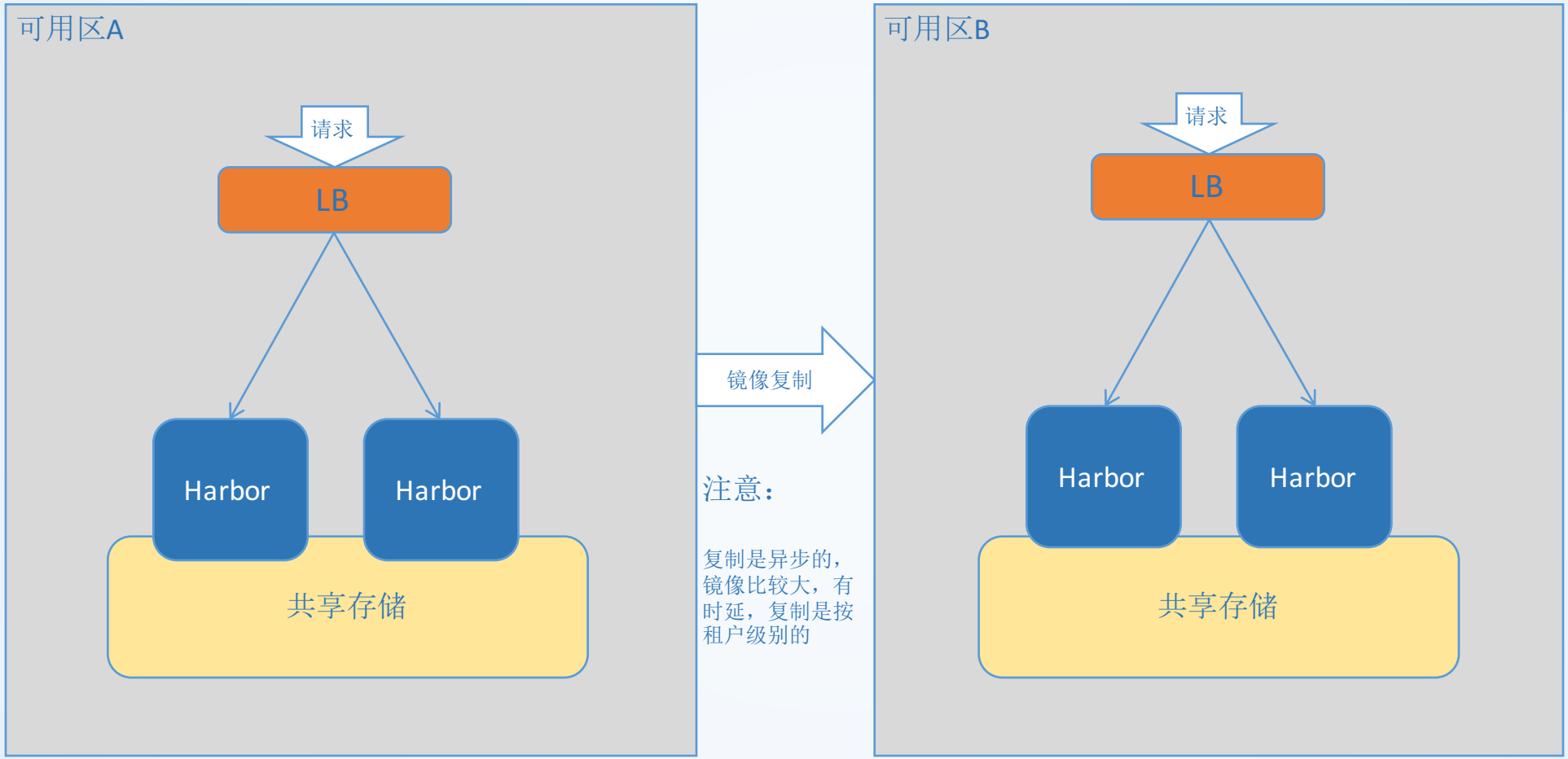


基于Harbor构建镜像服务：

- 分解Harbor原生服务
- MySQL基于Cat做一个高可用集群
- 除Docker Registry是有状态的，其余都是无状态的，可多实例部署
- Docker Registry通过共享存储，变成无状态，HA部署
- 共享存储（NFS）支持双写，存储层做Raid 1，保障数据安全
- 多实例前面做LB

产品形式参照各大厂公有云即可，镜像服务相对标准

镜像仓库 -- 灾备


可用区A
请求
LB
Harbor
Harbor
共享存储
镜像复制
注意：
复制是异步的，镜像比较大，有时延，复制是按租户级别的
可用区B
请求
LB
Harbor
Harbor
共享存储

编排调度

基本信息

名称: zltest0923　　版本号: 0.0.1　　创建人: 张磊　　创建时间: 2019-09-23 17:59:56　　描述:

版本号: 0.0.1

模板内容

- templates
  - ingress.yaml
  - deployment.yaml
  - service.yaml
  - NOTES.txt
  - _helpers.tpl
- Chart.yaml
- values.yaml
- charts

```
{{- if .Values.ingress.enabled -}}
{{- $serviceName := include "zltest0923.fullname" . -}}
{{- $servicePort := .Values.service.externalPort -}}
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: {{ template "zltest0923.fullname" . }}
  labels:
    app: {{ template "zltest0923.name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version | replace "+" "_" }}
    release: {{ .Release.Name }}
    heritage: {{ .Release.Service }}
  annotations:
    {{- range $key, $value := .Values.ingress.annotations }}
      {{ $key }}: {{ $value | quote }}
    {{- end }}
spec:
  rules:
    {{- range $host := .Values.ingress.hosts }}
    - host: {{ $host }}
      http:
        paths:
          - path: /
            backend:
```

编排调度：

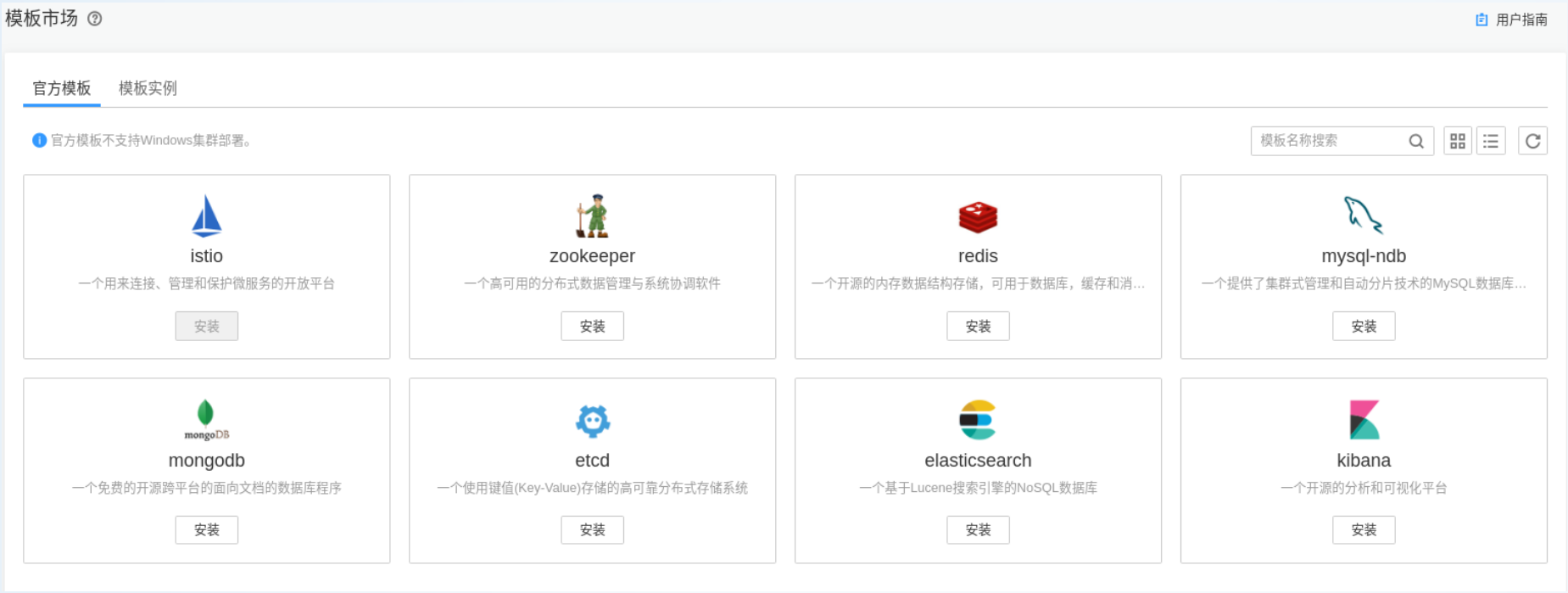
1、模板，k8s的资源对象

2、Helm部署，支持YAML对象，服务，版本，产品多维度的部署

本质上是定义部署模板，通过Helm服务，并发下发给集群，集群根据模板内描述的要求，调度生成应用实例

模板可以直接编写，也可以全UI化自动生成

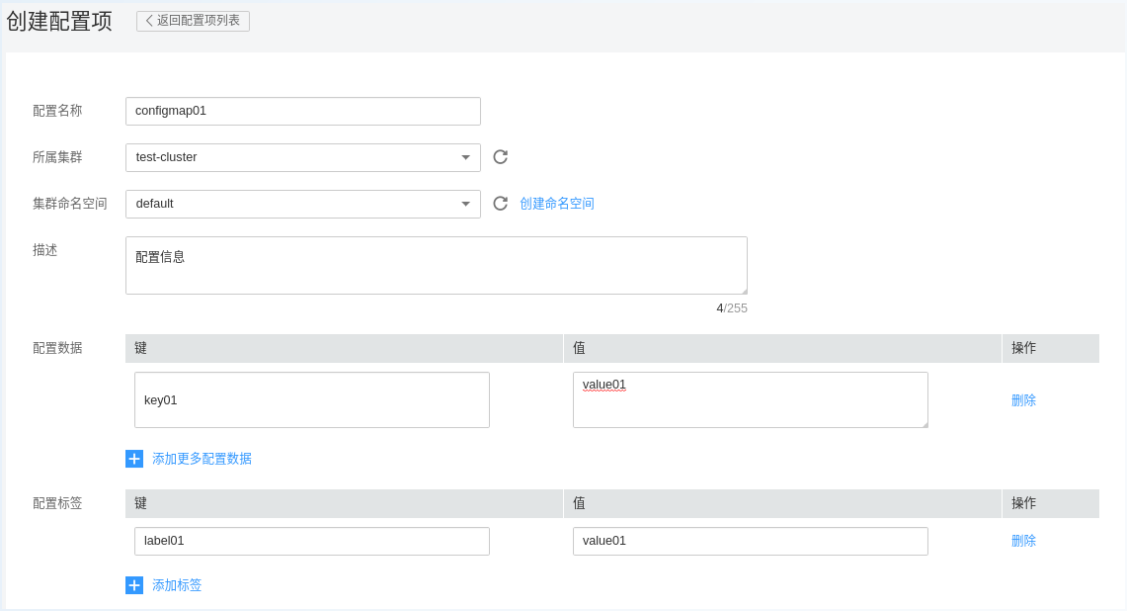
模板商店



模板是应用/中间件在集群内部署的规则
不同应用，不同版本，只需改镜像，端口域名，IP等，就可以部署各种应用服务

私有模板（可公开）/ 公有模板

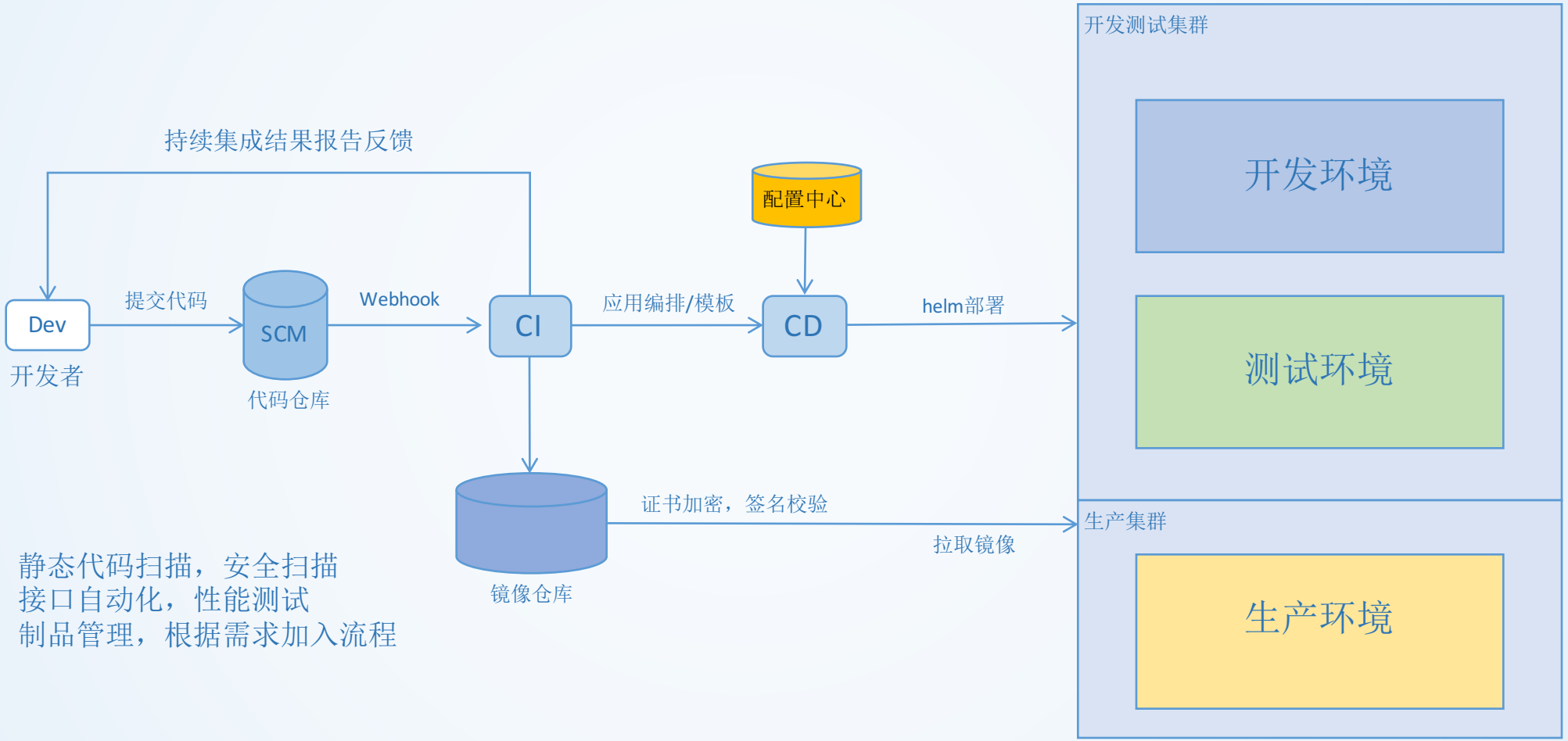
配置中心




```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  special.how: very
  special.type: charm
```

本质上就是生成一个模板

持续集成/流水线



CI流程：

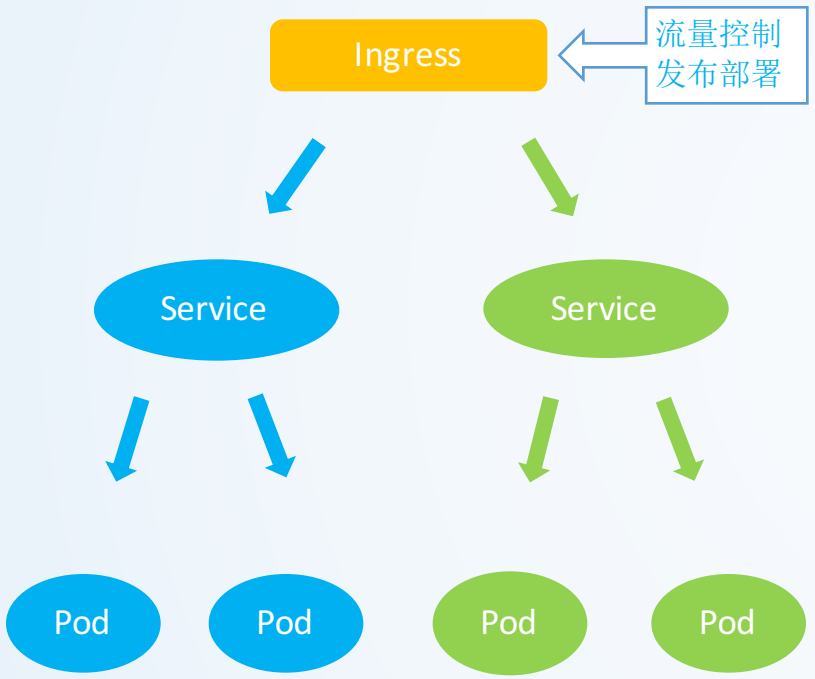
编译打包，静态代码扫描，安全扫描
单元测试，接口自动化，性能测试
审核卡点，制品管理，根据需求加入流程

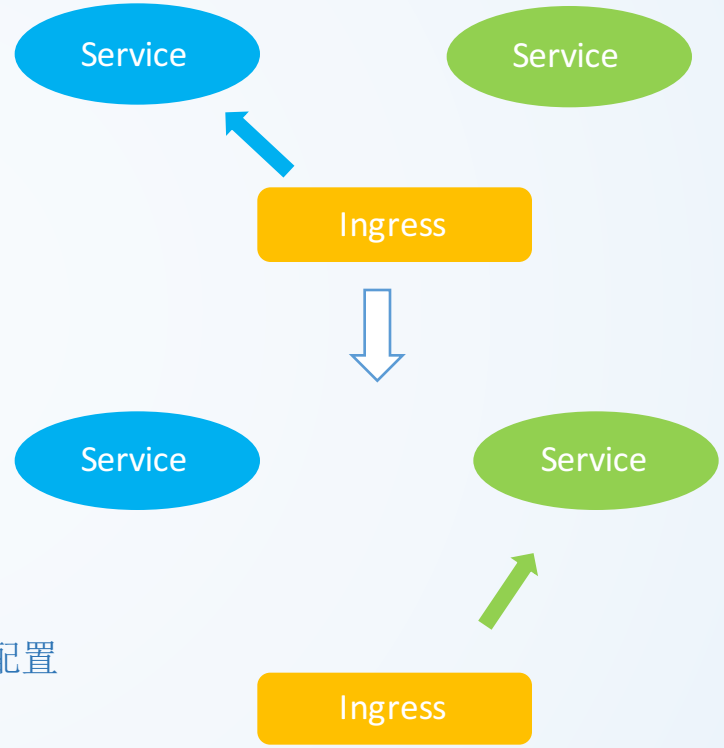
部署策略

部署策略：

- 滚动部署，Deployment/StatefulSet原生支持滚动策略，Deployment无序滚动，StatetulSet有序，中间可以暂时和再启
- 蓝绿部署，通过流量导入控制的方式来发布已部署好的服务
- 灰度部署（Canary）
  - 基于Cookie的灰度，用于PC / H5形式
  - 基于Header的灰度，用于APP / 黑的名单 / 地域等形式
  - 基于Weight的灰度，多版本在线，用于验证新版本效果或不能完全功能测试版本验证

部署策略 -- 蓝绿部署





流量控制：

Ingress的Weight配置

蓝： 1 0 0 —> 0

绿： 0 —> 1 0 0

部署策略 -- 灰度部署

灰度部署有多种维度，各种方案、目前和业务配合定制的居多，有基于Nginx+lua，也有kong方案，下面以Nginx型的Ingress为例，云原生的灰度方案，都是配置ingress的模板

1、基于Cookie的灰度

在ingress的模板里配置

annotations:

kubernetes.io/ingress.class ：nginx

nginx.ingress.kubernetes.io/canary ：true

nginx.ingress.kubernetes.io/canary-by-cookie ：canary-cookie

curl -H “Host:my-app.com” -b “canary-cookie=always” https://masterIP

2、基于Header的灰度

在ingress的模板里配置

annotations:

kubernetes.io/ingress.class ：nginx

nginx.ingress.kubernetes.io/canary ：true

nginx.ingress.kubernetes.io/canary-by-header ：canary-header

curl -H “Host:my-app.com” -H “canary-header=always” https://masterIP

3、基于Weight的灰度

在ingress的模板里配置

annotations:

kubernetes.io/ingress.class ：nginx

nginx.ingress.kubernetes.io/canary ：true

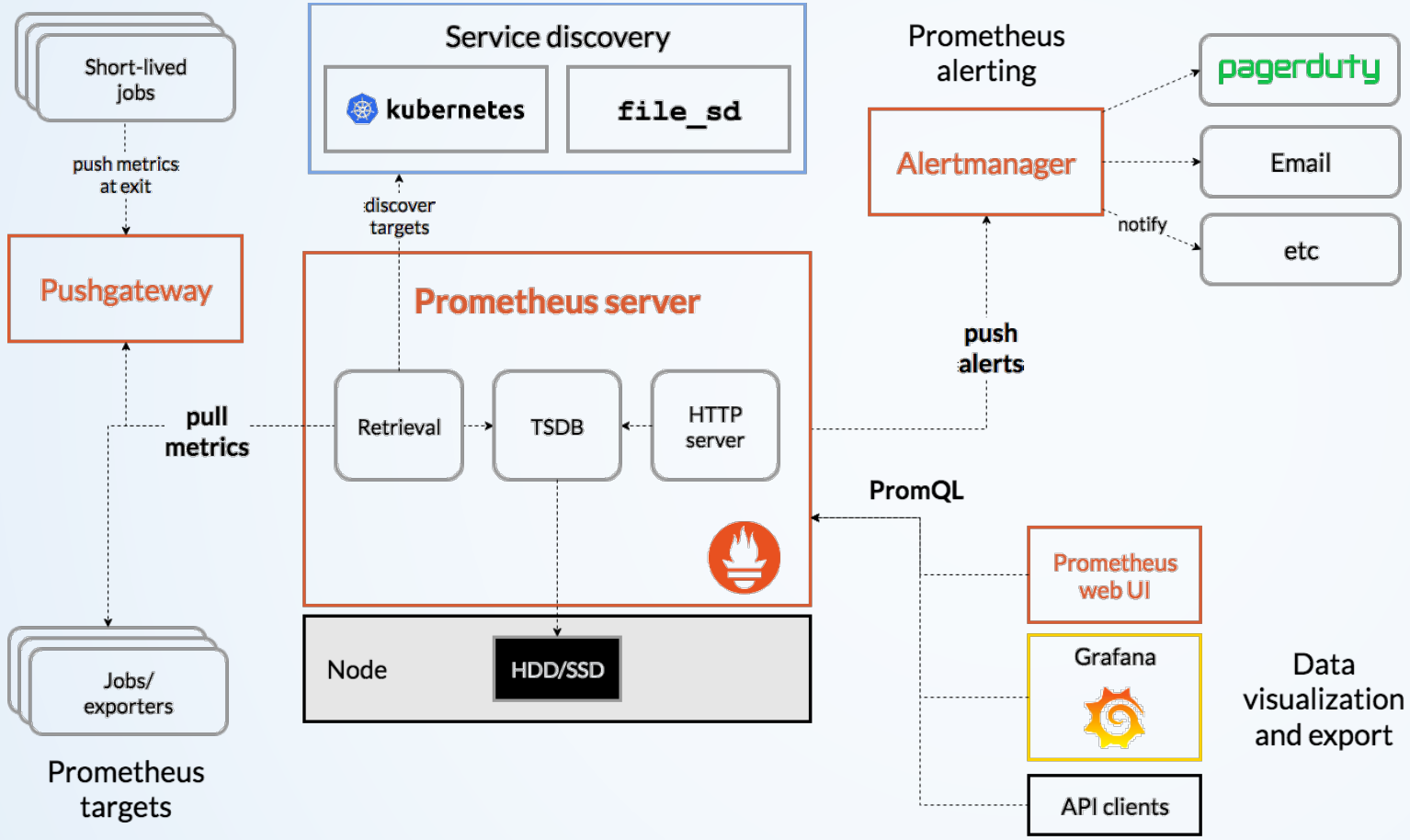
nginx.ingress.kubernetes.io/canary-weight ：10 即10%

配置多个ingress对应同一域名，配置不同权重

curl -H “Host:my-app.com” https://masterIP

按权重切分流量

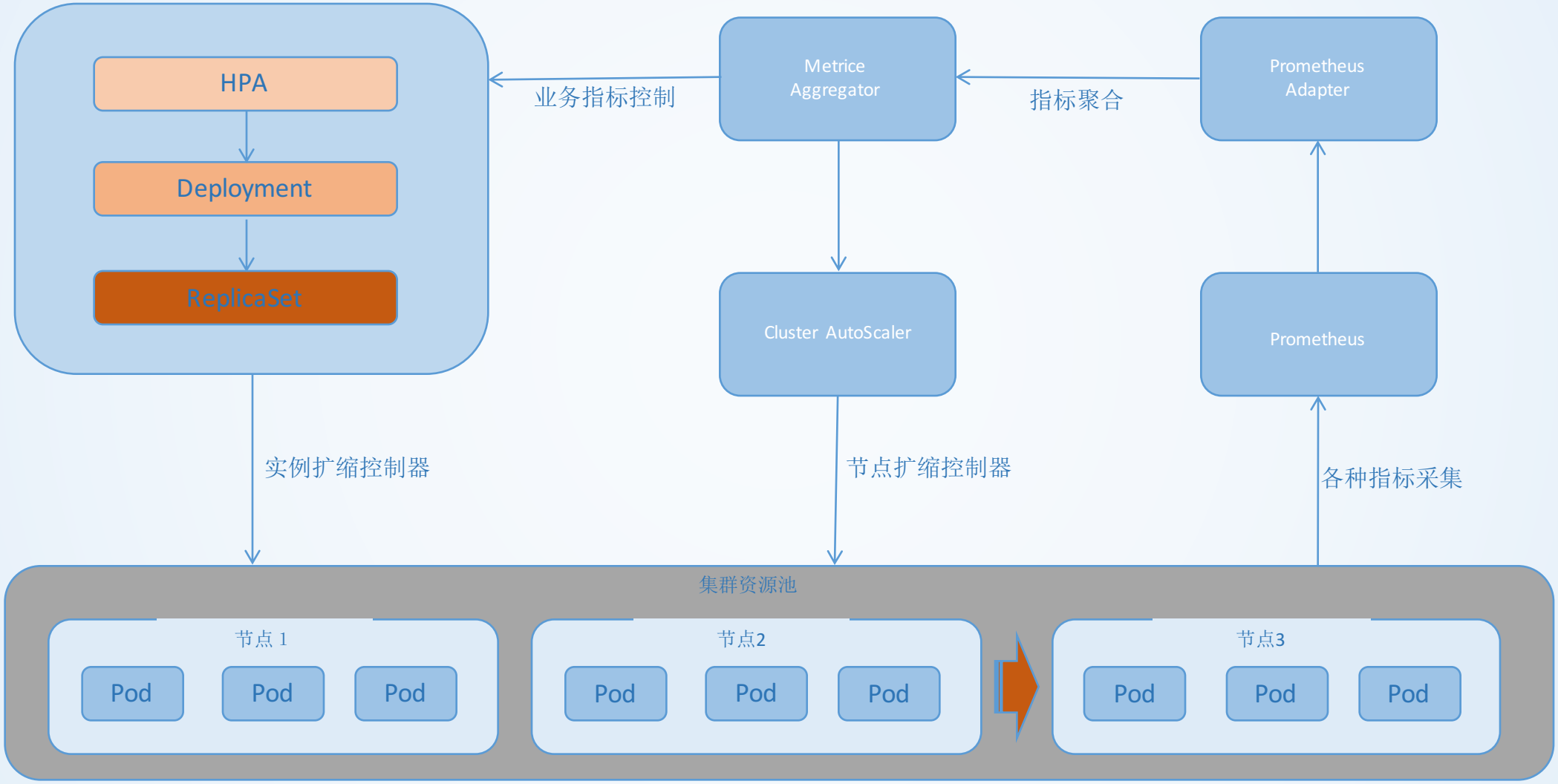
监控告警



监控的要点：

- 使用Prometheus云原生方案
- 直接原生部署，合理根据集群规模配置资源，注意存储性能，选择高IO的设备，保存监控数据的周期
- 告警系统对接Alertmanagersa模块，支持邮件，短信，电话，微信，钉钉等告警方式
- 展示面板可使用Grafana，也可自定义开发，满足需求

弹性伸缩


HPA
Deployment
ReplicaSet
业务指标控制
Metrice
Aggregator
指标聚合
Prometheus
Adapter
Cluster AutoScaler
Prometheus
实例扩缩控制器
节点扩缩控制器
各种指标采集
集群资源池
节点 1
Pod
Pod
Pod
节点2
Pod
Pod
Pod
节点3
Pod
Pod
Pod

双活灾备

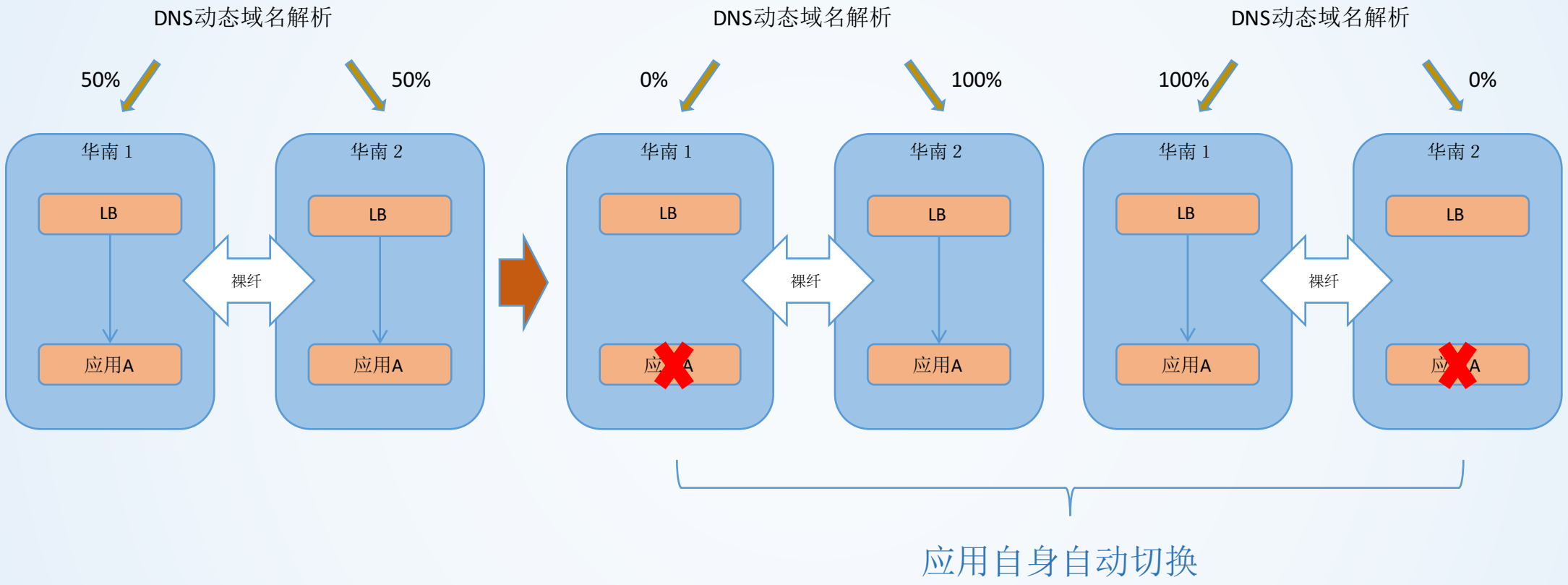
双活是个很大范围的概念，涉及应用双活，网络双活，存储双活，DB双活，DNS/出口网关/公网IP等，又有同城双活，异地双活，异地多活，除业务简单无状态外，异地形式一般都会侵入业务，光靠平台层无法实现，金融业务一般都是两地三中心布局，即同城双活，异地灾备

同城双活：
双IDC在附近的地域，IDC间通过双裸纤连接，带宽无限制，近似单IDC，但物理上，计算设备，网络设备，出口网关，带宽，公网IP，电信运营商都是分开的

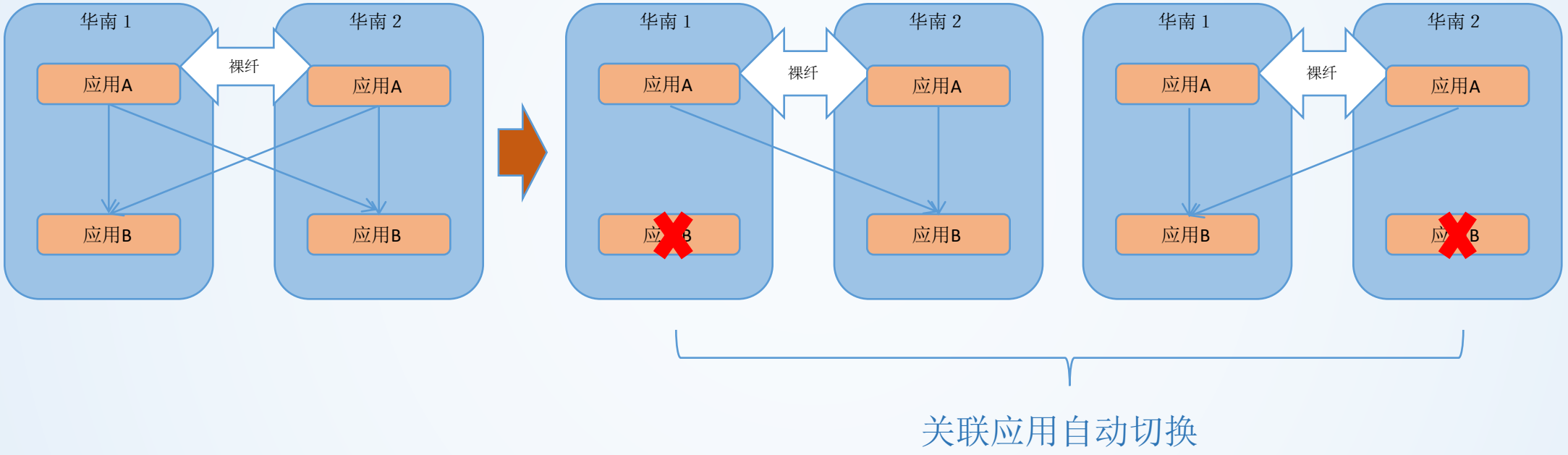
应用双活：
容器平台纳管多可用区，双活部署时，选择两同城可用区，创建集群，可按权重比例配置资源和应用实例
按服务的维度对集群外暴露业务，IDC内做LB，双IDC间通过内网DNS动态域名解析的方式，监控服务的可用性，自动切换，服务间通过域名调用，非Kubernetes的服务发现方式，DNS能智能解析，优先调用本IDC内的可用服务，不可用才调另IDC的同一服务

DB双活：
这个要给合服务的特性来做，传统的MySQL，PG等目前要做到数据完全一致的双活还比较困难，但新型的DB，计算与存储解耦的，如TiDB可以通过Kubernetes实现，即Kubernetes跨IDC单集群部署，TiKV在Kubernetes Opertor的加持下，在不同IDC间的结点间做到数据的多副本，数据一致的效果，前提是网络带宽得够

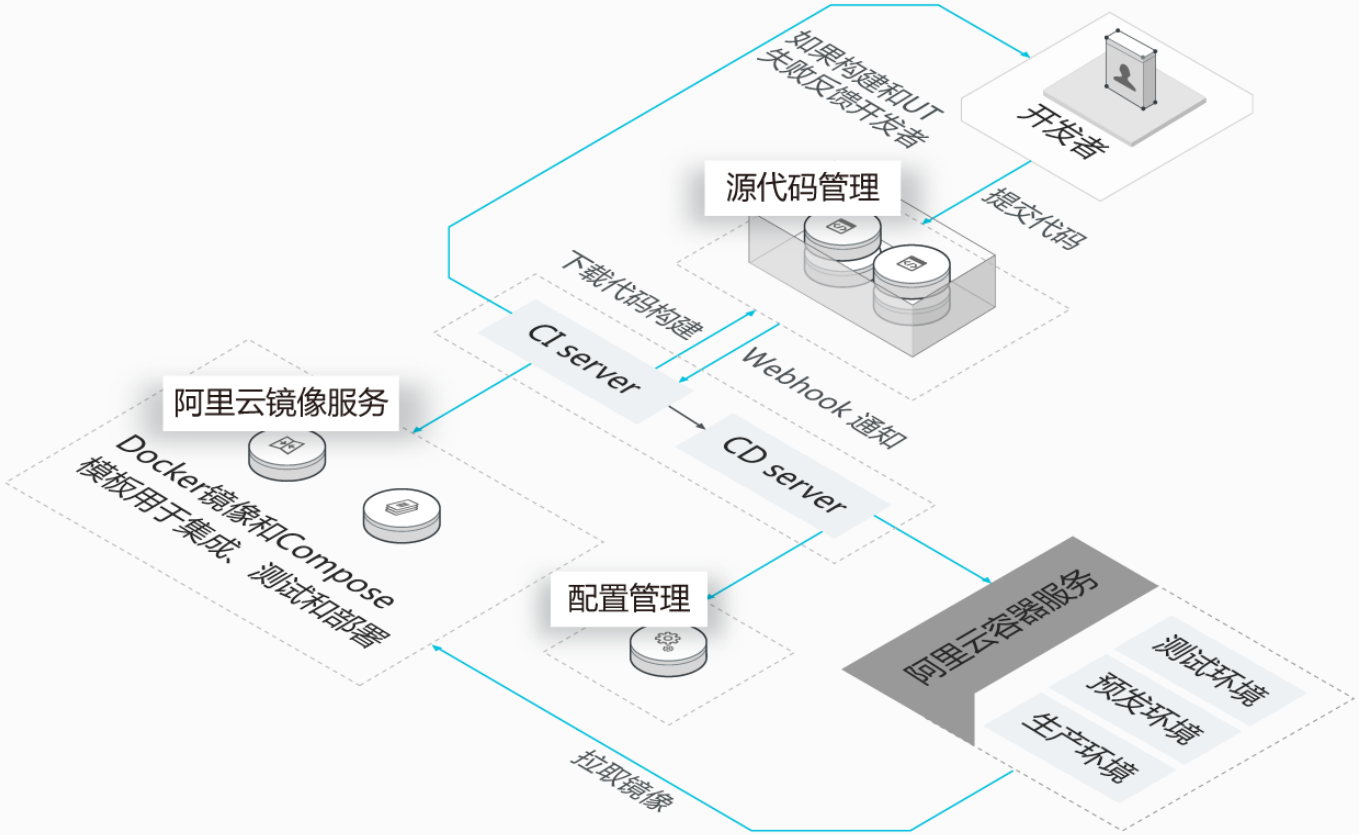
双活灾备


DNS动态域名解析
50%
50%
华南 1
LB
应用A
裸纤
华南 2
LB
应用A
DNS动态域名解析
0%
100%
华南 1
LB
应用A
裸纤
华南 2
LB
应用A
DNS动态域名解析
100%
0%
华南 1
LB
应用A
裸纤
华南 2
LB
应用A
应用自身自动切换

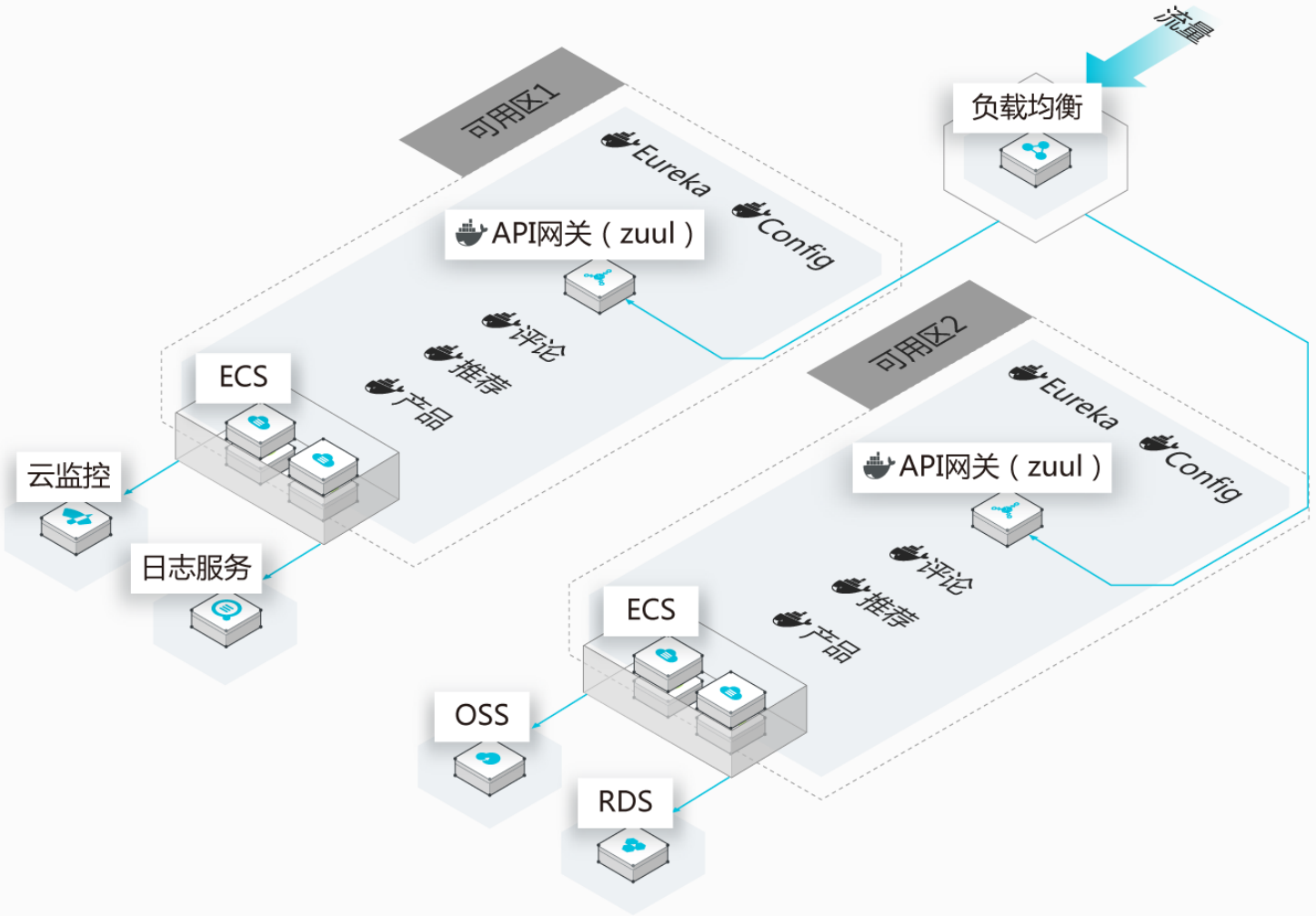
双活灾备


华南 1
裸纤
华南 2
应用A
应用A
应用B
应用B
华南 1
裸纤
华南 2
应用A
应用A
应用B
应用B
华南 1
裸纤
华南 2
应用A
应用A
应用B
应用B
关联应用自动切换

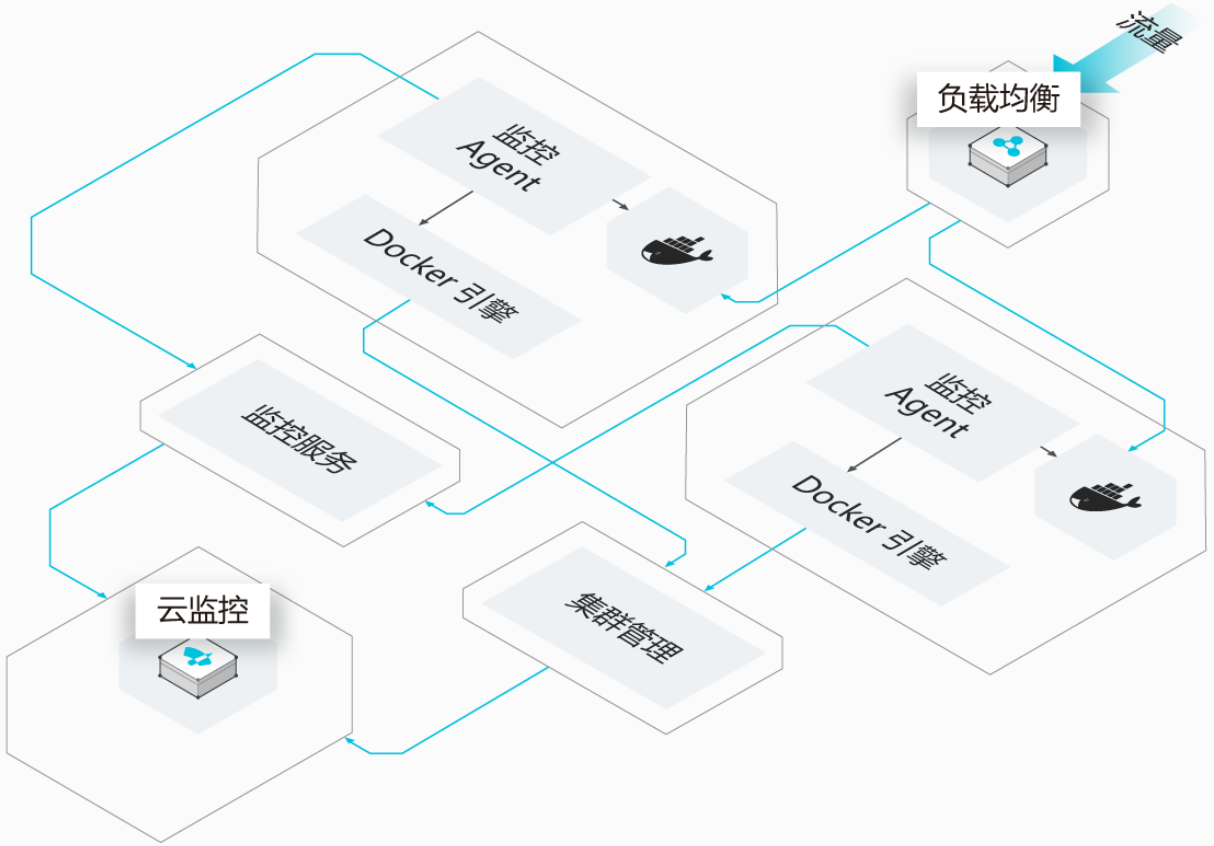
主要使用场景 ---- DevOps持续交付



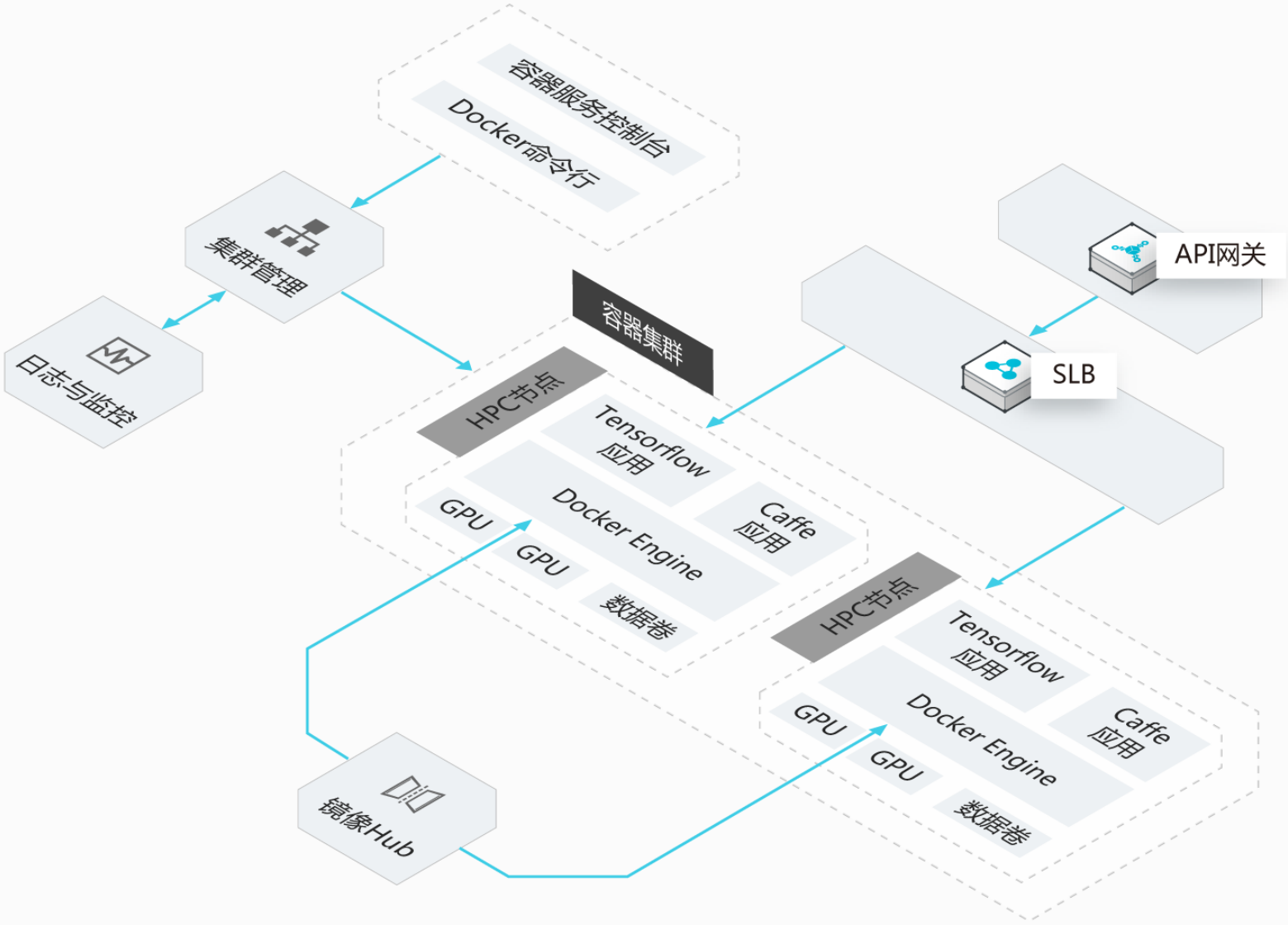
主要使用场景 ---- 微服务


流量
负载均衡
可用区1
Eureka
Config
API网关（zuul）
评论
推荐
产品
ECS
云监控
日志服务
可用区2
Eureka
Config
API网关（zuul）
评论
推荐
产品
ECS
OSS
RDS

主要使用场景 ---- 弹性伸缩



主要使用场景 ---- AI计算


容器服务控制台
Docker命令行
集群管理
日志与监控
容器集群
HPC节点
Tensorflow
应用
Caffe
应用
Docker Engine
GPU
GPU
数据卷
API网关
SLB
HPC节点
Tensorflow
应用
Caffe
应用
Docker Engine
GPU
GPU
数据卷
镜像Hub

最佳实践

镜像制作：

- 不要从不可信任的外部下载镜像，特别是基础镜像，最好自己定制，镜像尽可能的小，镜像间的依赖，继承关系，比如CentOS—JDK—TOMCAT—APP
- 开发人员得熟悉dockerfile，怎么写镜像可以小，且构建快速
- 镜像中按需集成些常用命令工具，方便Debug，如：ifconfig,telnet,netstat,dig等
- 镜像里尽量不使用root启动进程，减少攻击面
- 非必要，镜像里最好跑一个进程，避免产生僵尸容器

集群管理：

- 根据业务需求量评估及业务特性，合理选择集群规模及节点的规格（CPU/MEM/DISK）
- 宿主机OS定制加固，轻量化，稳定的内核，参照kubernetes官方
- Etcd集群使用全闪（SSD）的节点部署，做好存储灾备和快速恢复预案
- 系统和kubernetes组件预留合理的资源配置，确保集群稳定性
- Master不调度业务，打上污点，禁止调度，参数调优，插件指定节点部署，不与业务混部
- Node打上多维度标签，使用应用编排及高可用反亲和/亲和部署
- 按命名空间配置好额度管理及LimitRanger

最佳实践

应用部署：

- 应用部署前设计定义多维度标签
- 应用资源隔离设置合理的Limit/Request
- 应用强制配置健康检查探针，最好是HTTP，不推荐TCP
- 应用最好配置PreStop LifeCycle，实现程序优雅关闭
- 应用重启策略最好不要配置为Restart on Failure
- 除开发环境外，镜像标签不要使用latest或无标签
- 应用的配置信息最好不要打到镜像里，使用Configmap/Secret/外部配置中心
- 应用不要部署单个实例，避免发版时断服，多实例建议考虑亲和性策略

流量转发/负载均衡：

- 使用IPVS，不推荐Iptables
- Ingress节点做多租户，多节点做HA，不部署业务，只做流量转发
- 尽量不用或少用NodePort做对集群外服务暴露
- 基于LVS/HaProxy/Nginx,自研loadBalance型Service，比较简单，减少转发层

其它：

- 角色定义分明，权限清晰，审计记录完整
- 安全，SSL证书，网络策略，基线管理

Q&A

谢谢！

Q&A