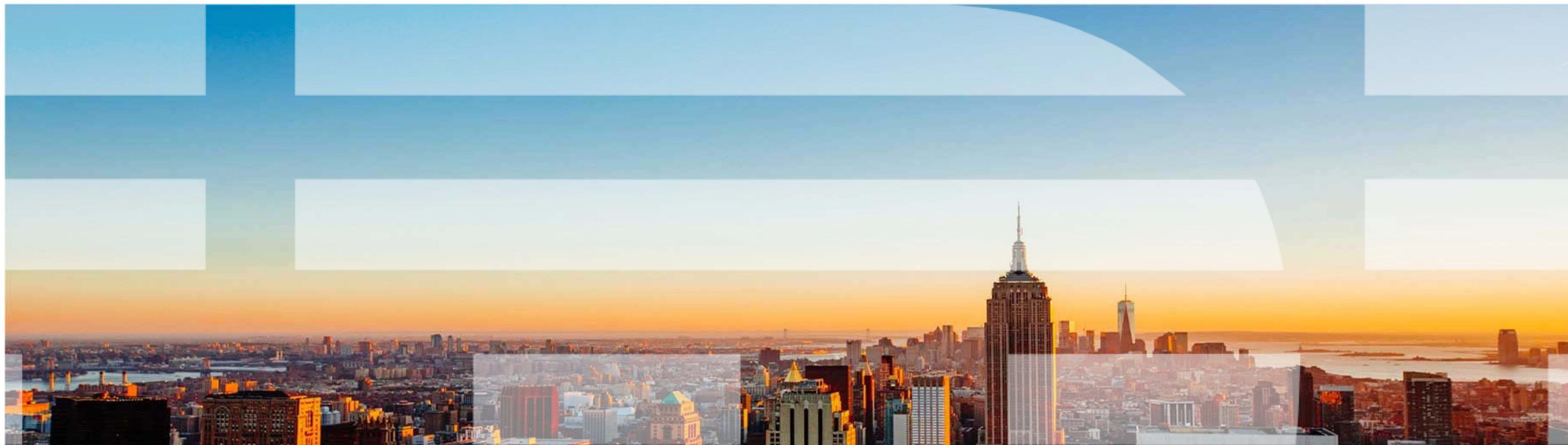


PaaS云平台在企业中的落地实践



1、基于Docker和Kubernetes的平台建设

- ! PaaS平台落地实施关注的问题
- ! Kubernetes核心组件
- ! Kubernetes资源对象
- ! Kubernetes生产环境实施

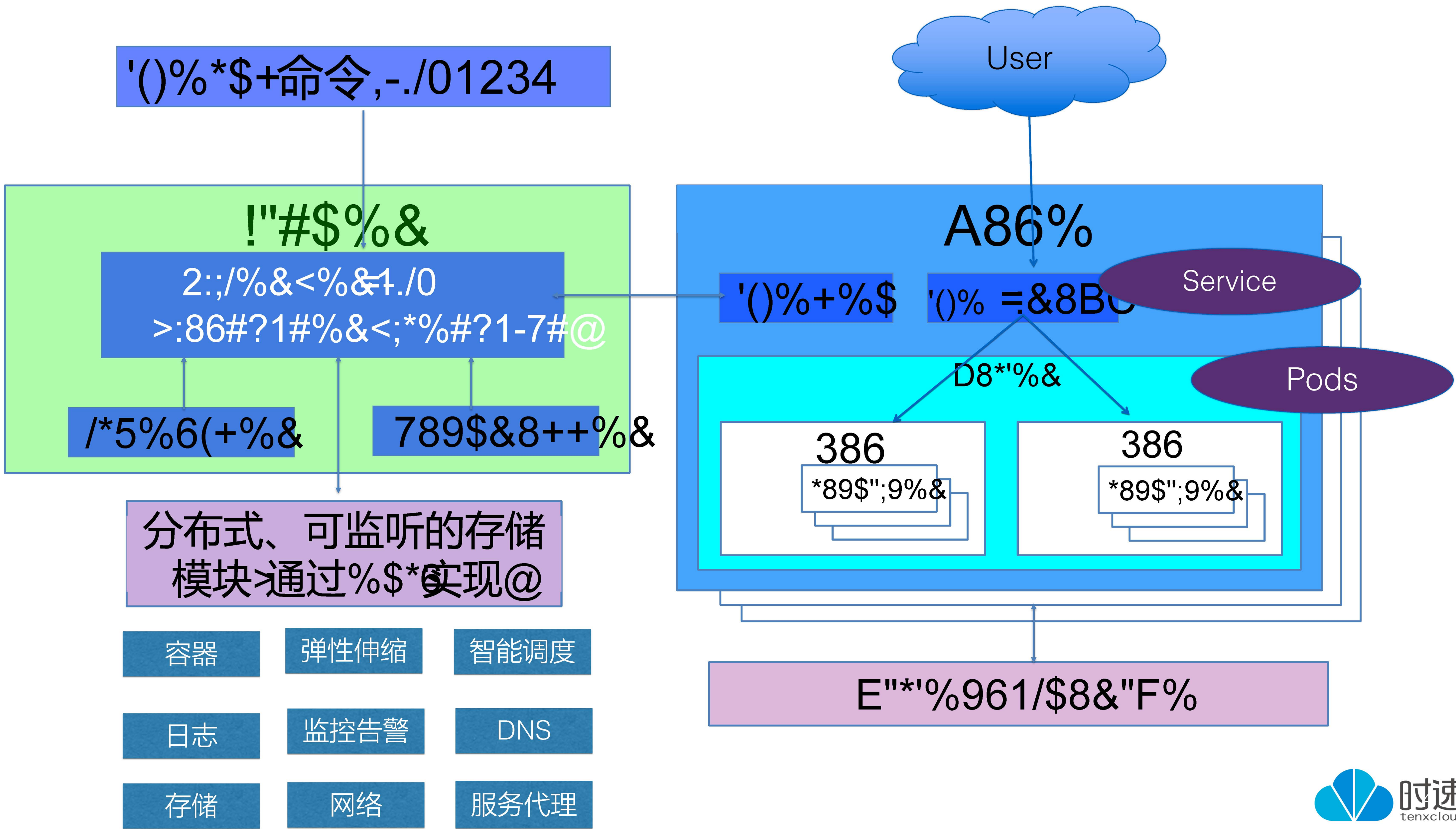
2、容器云平台在企业中的应用场景

- ! 容器云平台功能体系
- ! Devops体系
- ! 测试生产运行体系
- ! 微服务治理

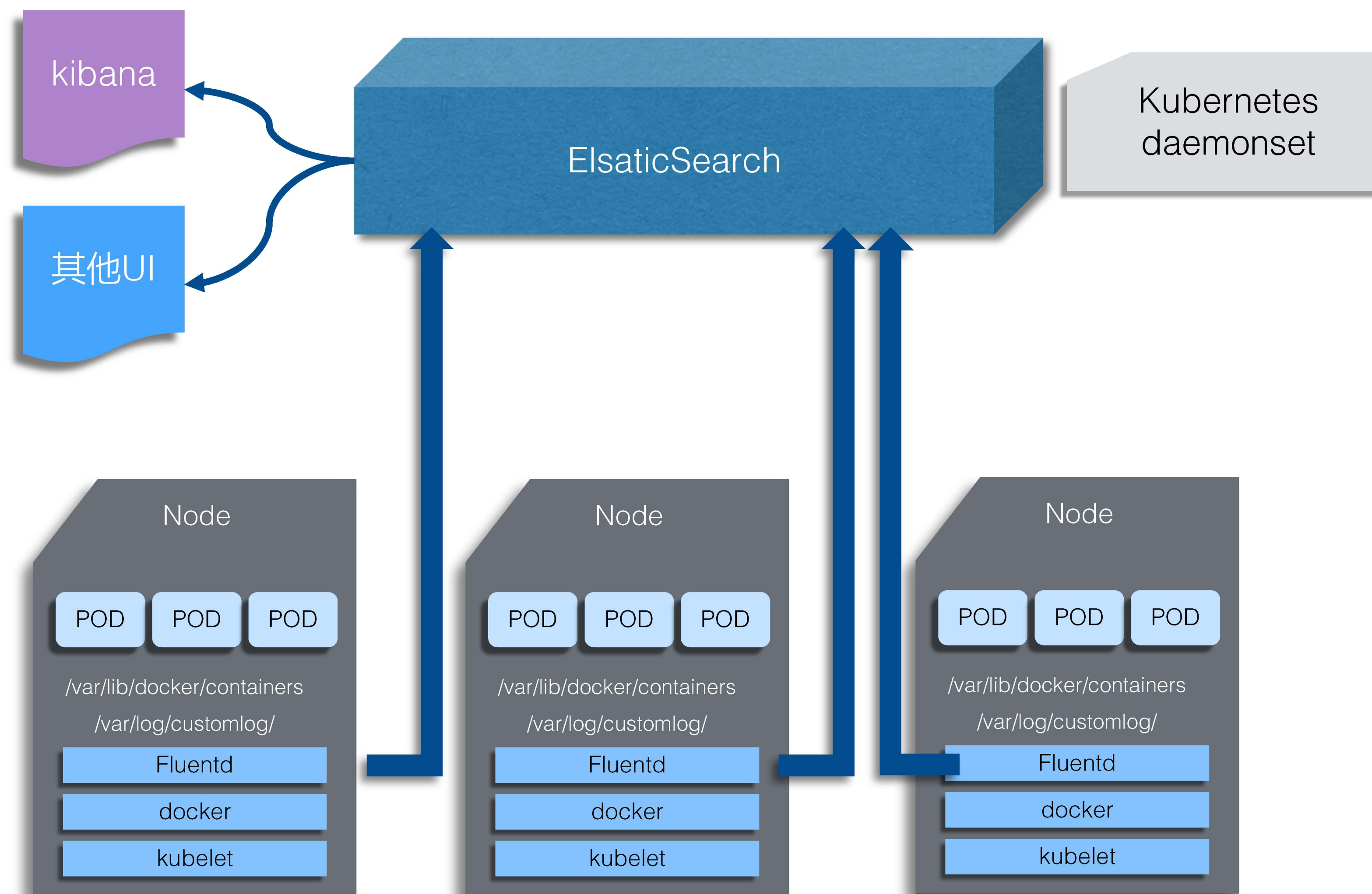
!""# 容器云平台落地实施关注的问题

- 架构运行
- 核心组件运行
- 服务代理出口
- 系统高可用性
- 存储类型
- 网络类型
- 持续集成组件运行(子业务产品)
- 应用(有状态)运行与发布

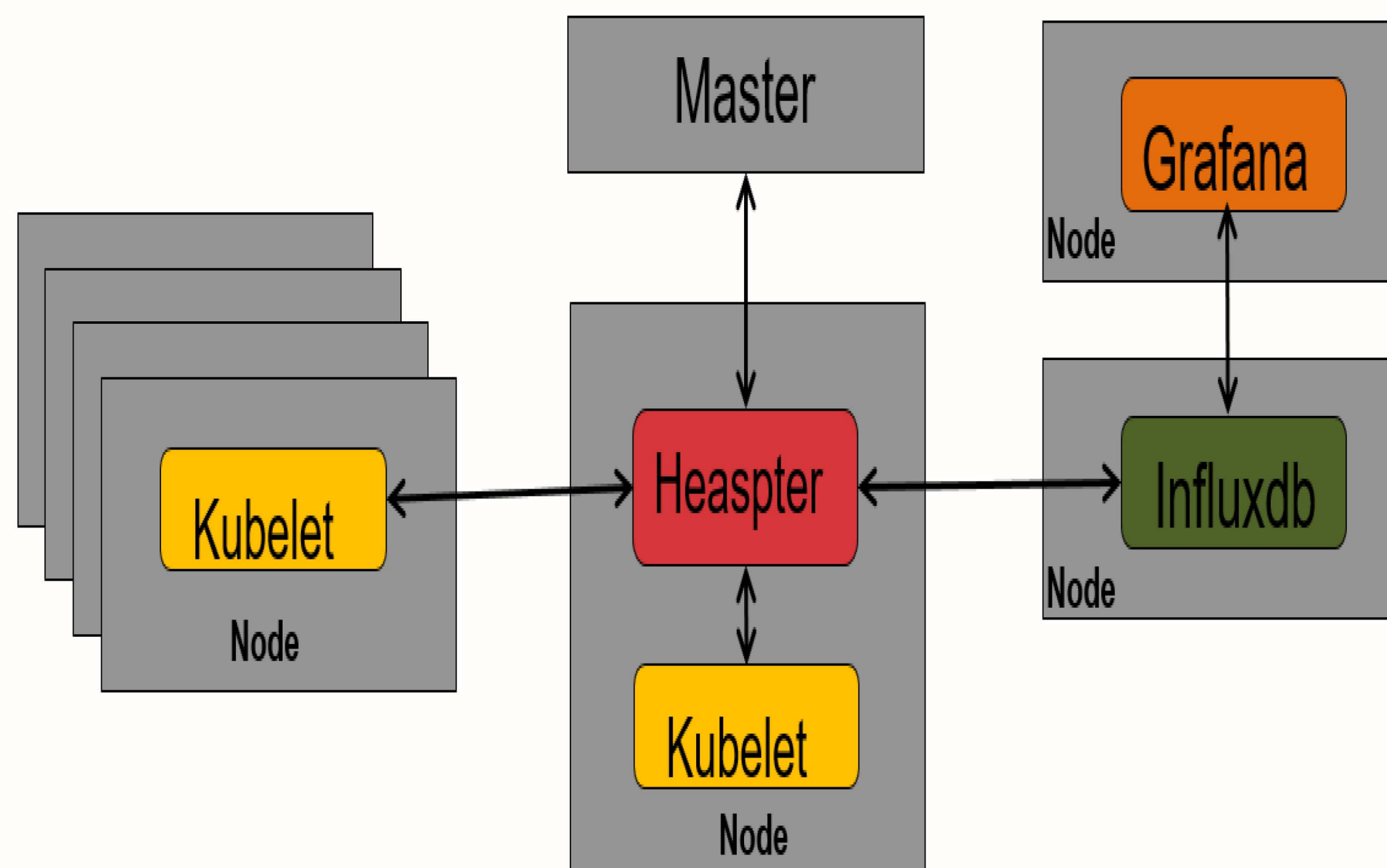
!"#\$%&\$'\$(架构



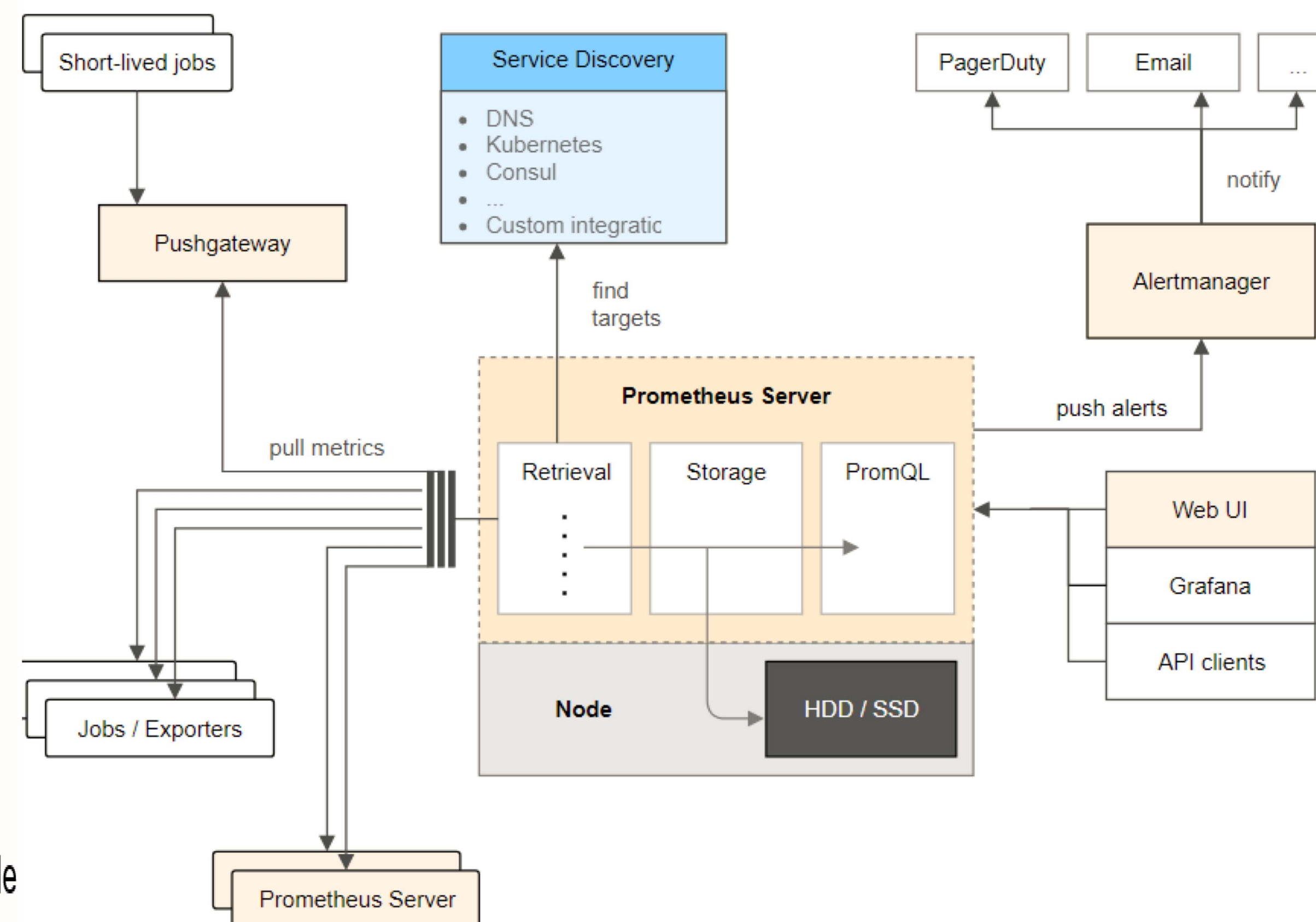
!'"#\$\$%&\$'\$(核心组件)日志收集



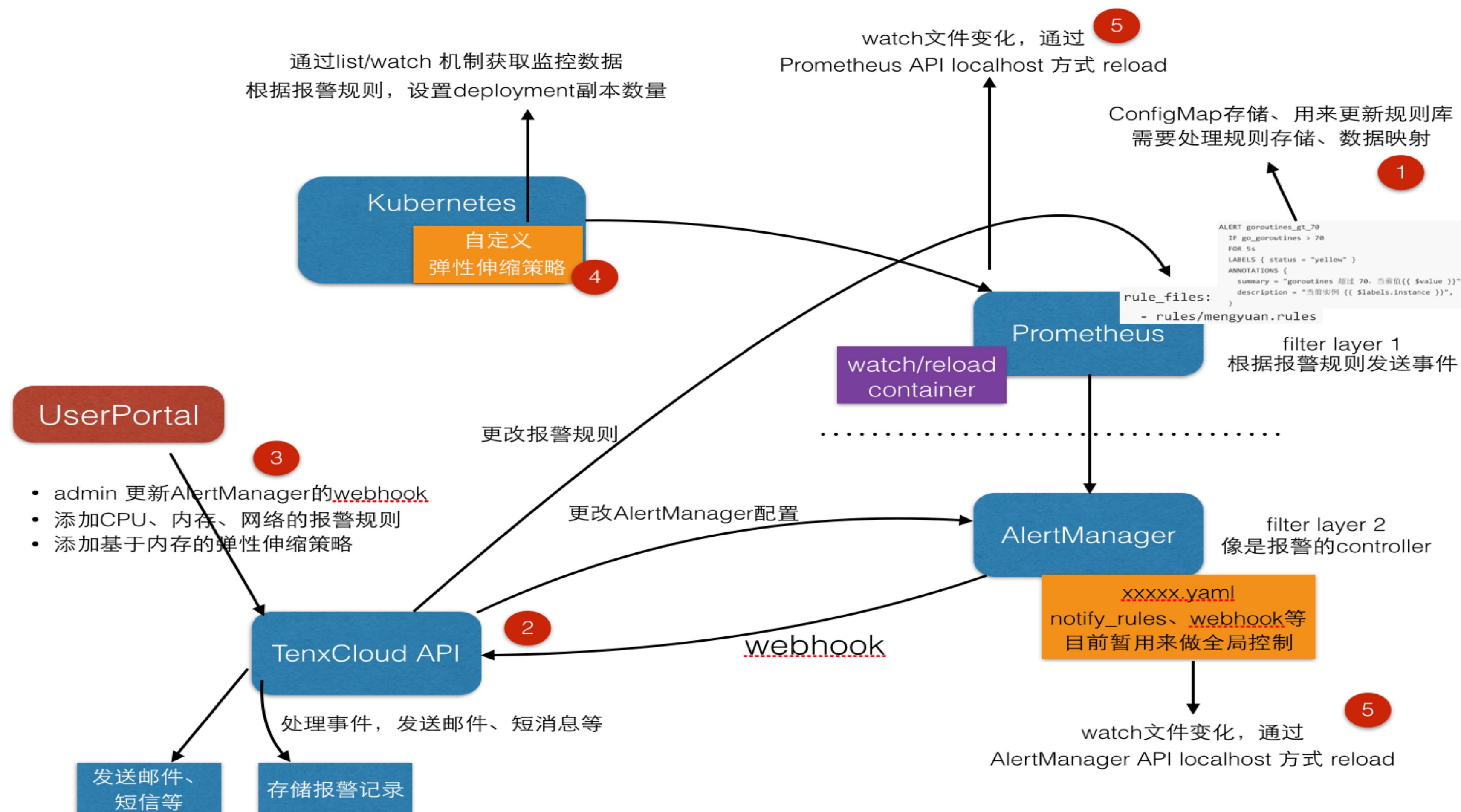
!'"#\$%&\$'\$(核心组件)监控



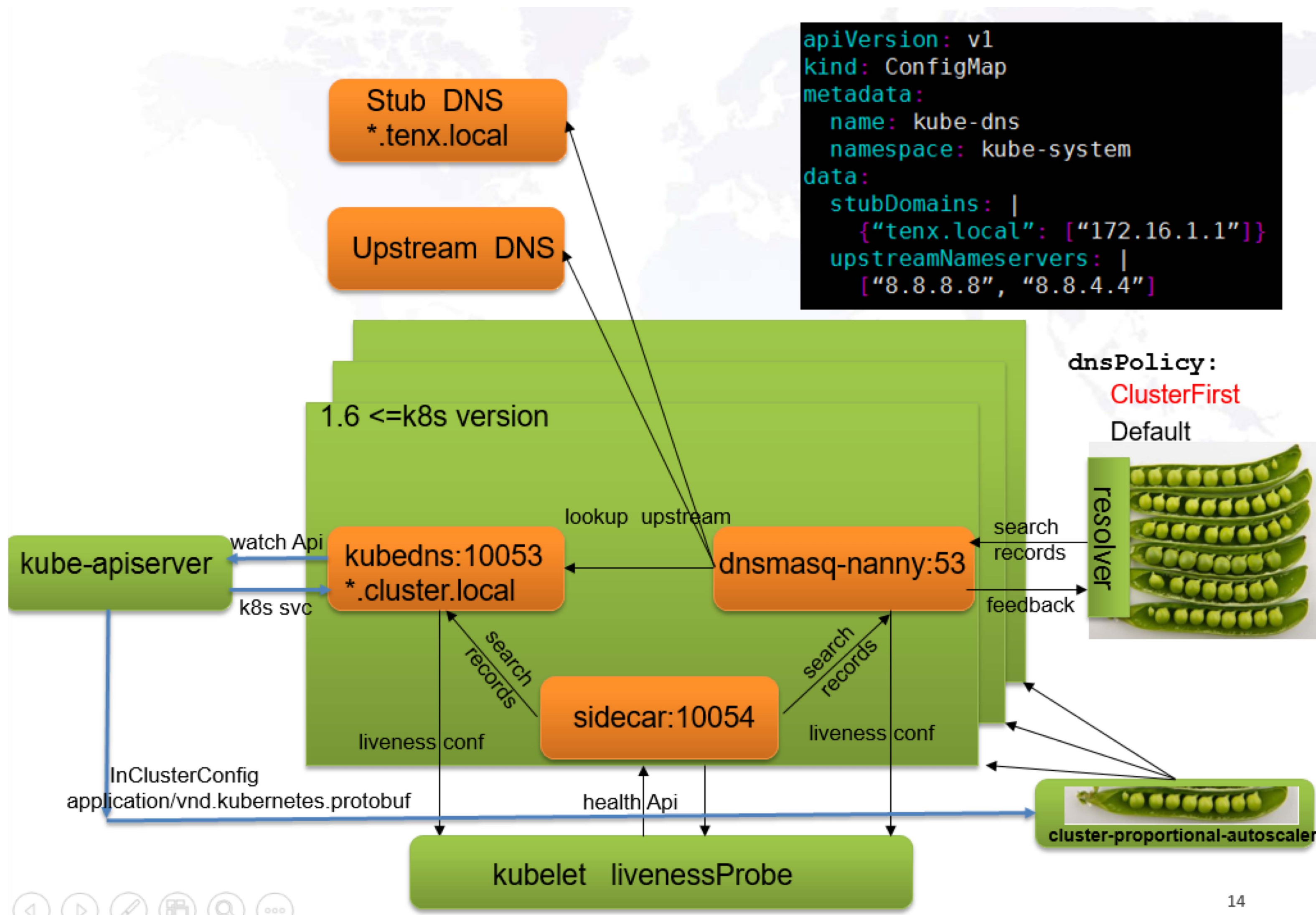
Heapster is a cluster-wide aggregator of monitoring and event data, The Heapster pod discovers all nodes in the cluster and queries usage information from the nodes' Kubelets, This data is then pushed to a configurable backend for storage and visualization. Currently supported backends include Influxdb



!'"#\$%&\$'\$(核心组件)监控



!"#\$%&\$'\$(核心组件)*+,

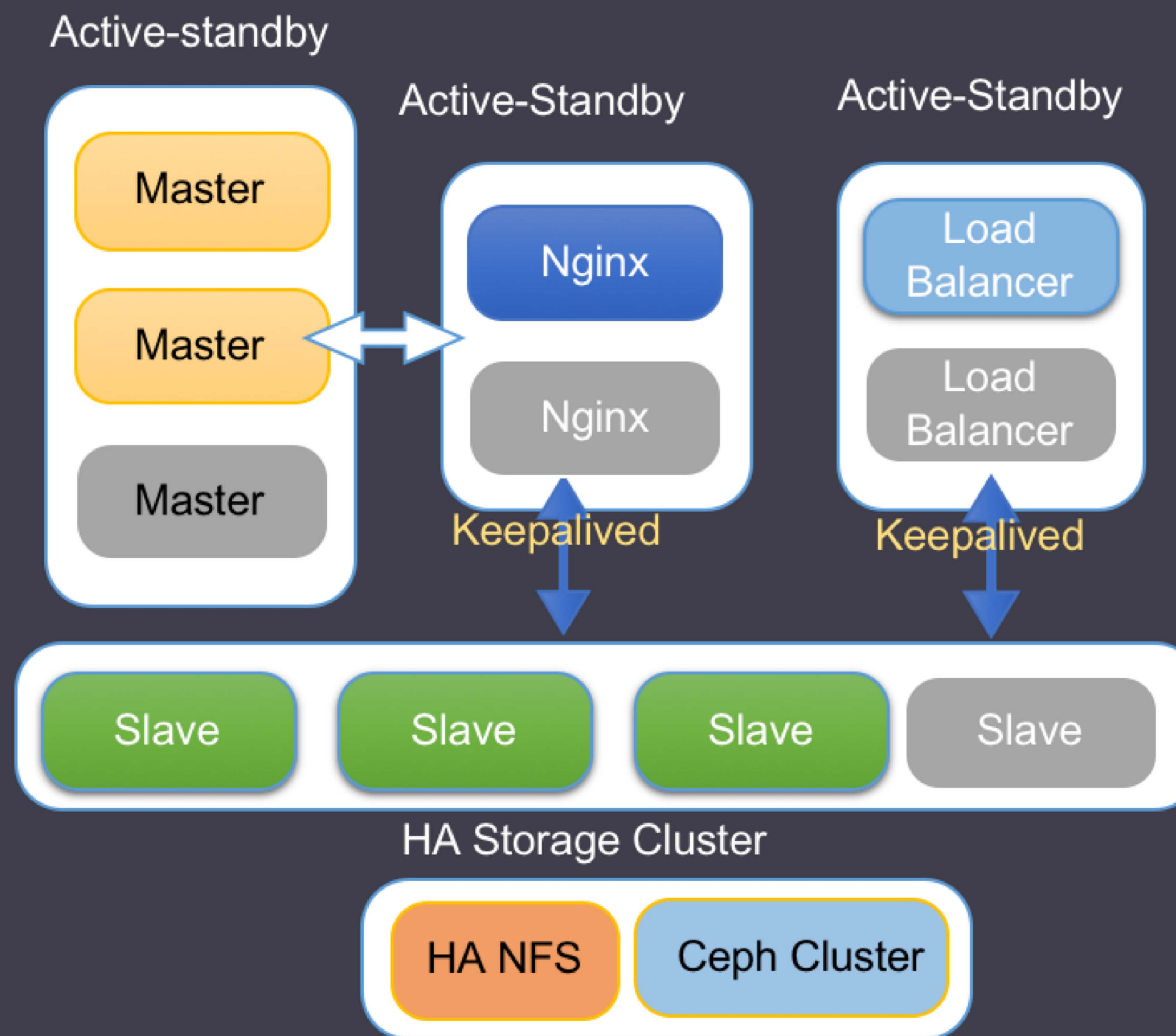


!'"#\$\$%&\$'\$()高可用

System High Availability

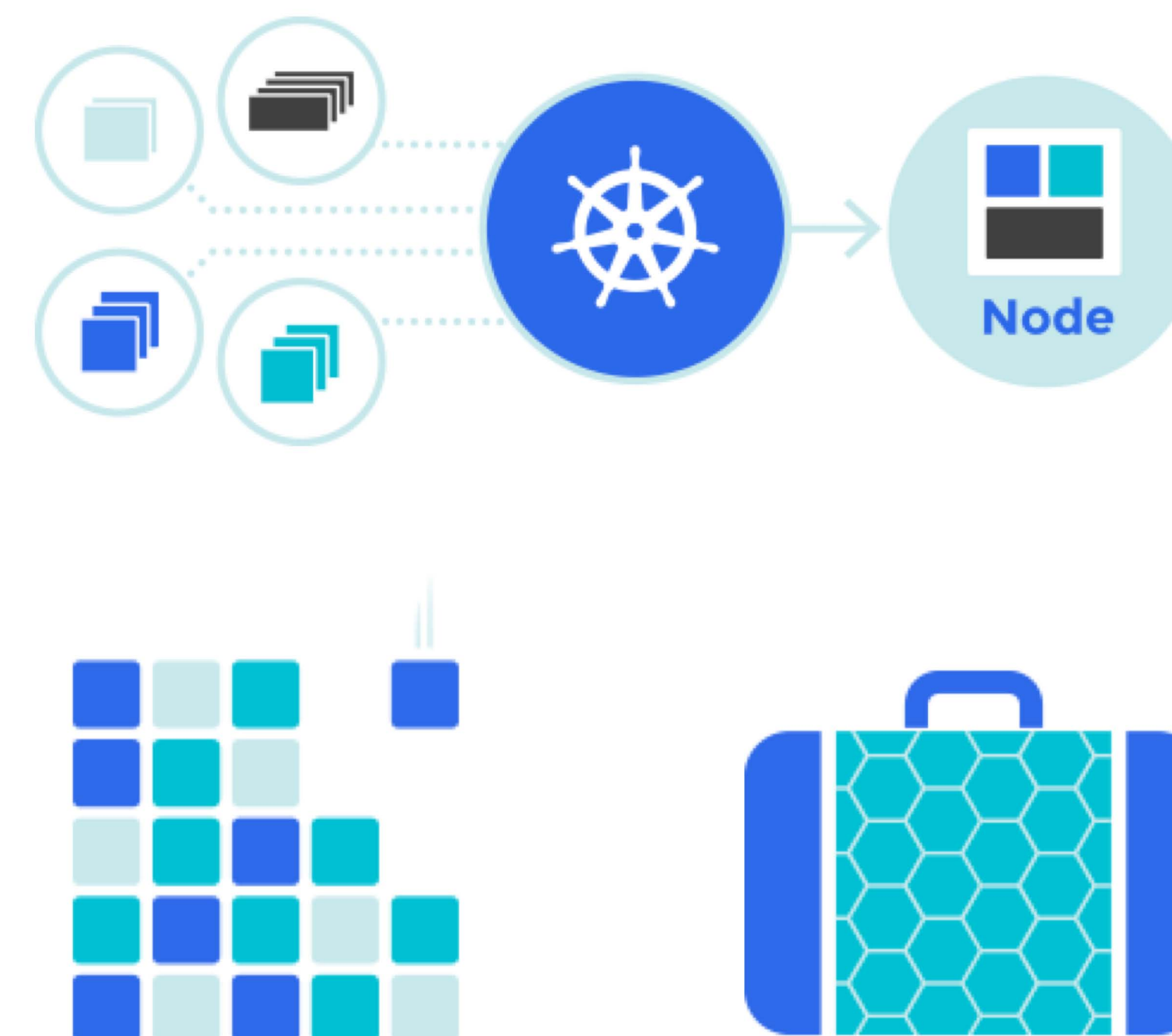
1. Master nodes
2. Master Router
3. Distributed Storage
4. Service Load Balancer
5. Container Cluster

We can power off any nodes for each component and make sure each component still works well



!"#\$%&\$'\$(资源对象

- !"#\$%最小部署单元，容器集合，共享存储、网络&
- '()*+,-./0123456789:;@A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 生命周期控制器，2,-*(\$ 资源伸缩&
- 3()*456789:;@A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 提供了一个声明式定义&
- 6(17+,\$%抽象服务出口，通过)1"84\$对多个容器负载均衡，内部服务发现&
- 9-:(*2\$%标签，用于分类，查询筛选，规则由用户指定&
- 0"/;+<=-) (保存配置数据的键值对>)
- 6(,1((解决了密码、."?(/ 、密钥等敏感数据的配置问题)
- @":A01"/B": (任务A定时任务，就类似于9+/C&系统的,1"/.-:)
- D!E (水平伸缩)
- 6.-. (;C*%解决有状态服务的问题&
- F"*C5(2%磁盘存储&
- G/<1(22(负载均衡，代理)



!"#\$%&\$'\$(资源对象 - Stateful

StatefulSet是为了解决有状态服务的问题（对应Deployments和ReplicaSets是为无状态服务而设计），其应用场景包括：

- ！ 稳定的持久化存储，即Pod重新调度后还是能访问到相同的持久化数据，基于PVC来实现
- ！ 稳定的网络标志，即Pod重新调度后其PodName和HostName不变，基于Headless Service（即没有Cluster IP的Service）来实现
- ！ 有序部署，有序扩展，即Pod是有顺序的，在部署或者扩展的时候要依据定义的顺序依次依次进行（即从0到N-1，在下一个Pod运行之前所有之前的Pod必须都是Running和Ready状态），基于init containers来实现
- ！ 有序收缩，有序删除（即从N-1到0）

StatefulSet由以下几个部分组成：

- ！ 用于定义网络标志（DNS domain）的Headless Service
- ！ 用于创建PersistentVolumes的volumeClaimTemplates
- ！ 可定义应用，定义实例数的StatefulSet

DNS命名：

```
statefulSetName-{0..N-1}.serviceName.namespace.svc.cluster.local  
dig @10.96.0.10 web-1.nginx.default.svc.cluster.local
```

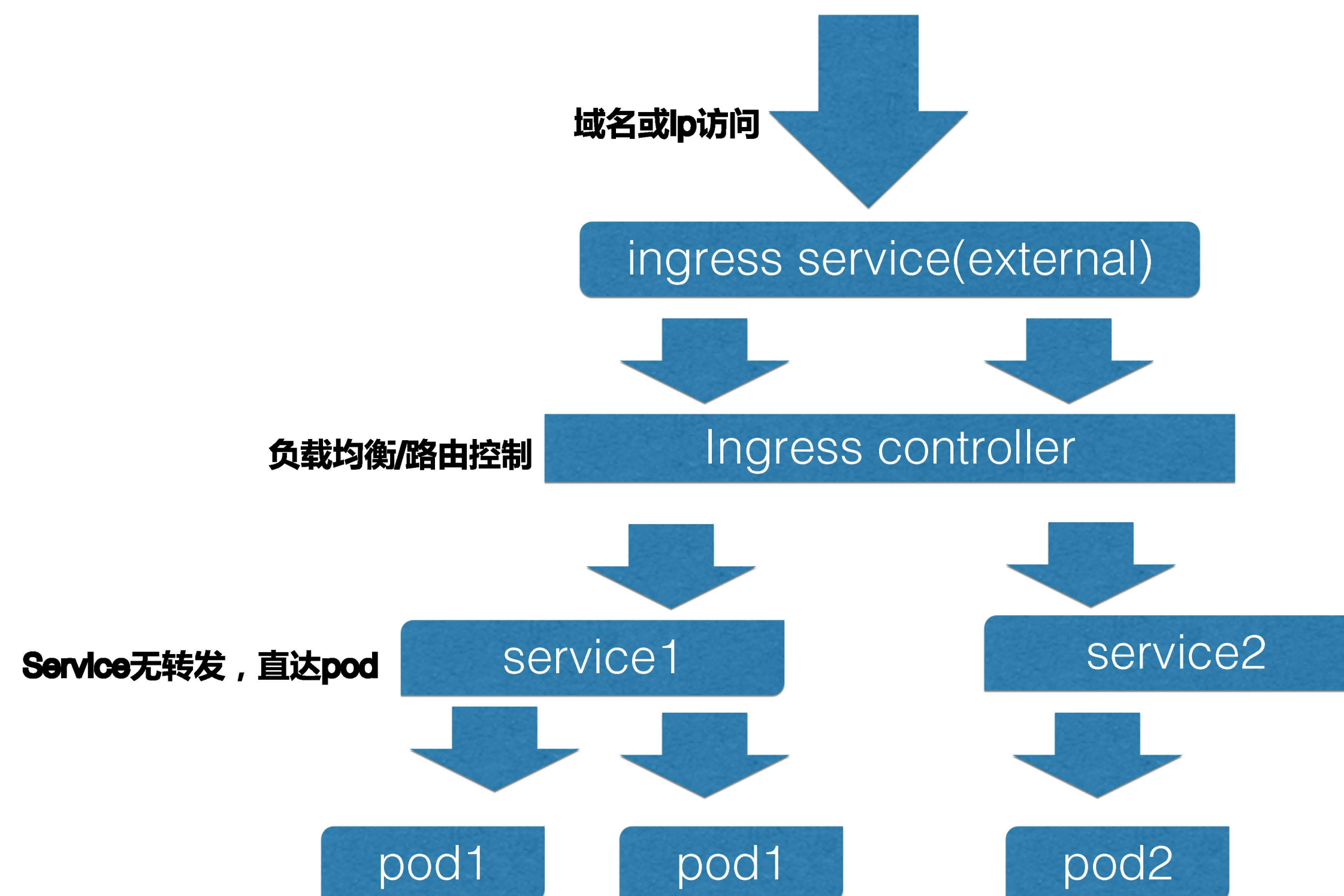

!"#\$%&\$'\$(资源对象 - Ingress

通常情况下，service和pod的IP仅可在集群内部访问。集群外部的请求需要通过负载均衡转发到service在Node上暴露的NodePort上，然后再由kube-proxy将其转发给相关的Pod。

Ingress可以给service提供集群外部访问的URL、负载均衡、SSL终止、HTTP路由等。为了配置这些Ingress规则，集群管理员需要部署一个[Ingress controller](#)，它监听Ingress和service的变化，并根据规则配置负载均衡并提供访问入口。

重点:

- 1.需要创建ingress-controller的deployment,该ingress-controller需要监听服务和ingress对象的变动，并可依据规则进行路由代理，例如Nginx,haproxy。
- 2.需要集群外部访问时，需要为ingress-controller创建service并将可代理到外面。
- 3.需要创建ingress对象，ingress中指明了入口的域名(如果需要),和后段service名称和端口
- 4.正常创建一个需要代理出去的服务



!"#\$%&\$'\$(资源对象 -)*+",\$(\$

目前支持以下类型的Volumes：

- emptyDir 随Pod删除，适用于临时存储、灾难恢复、共享运行时数据，支持 RAM-backed filesystem
- hostPath 类似于Docker的本地Volume, 用于访问一些本地资源（比如本地Docker）
- gcePersistentDisk GCE disk - 只有在 Google Cloud Engine 平台上可用
- awsElasticBlockStore 类似于GCE disk, 节点必须是 AWS EC2的实例
- nfs - 支持网络文件系统
- rbd - Rados Block Device - Ceph
- secret 用来通过Kubernetes API 向Pod 传递敏感信息，使用 tmpfs (a RAM-backed filesystem)
- persistentVolumeClaim - 从抽象的PV中申请资源，而无需关心存储的提供方
- glusterfs
- iscsi
- gitRepo
- Resources 规范emptyDir, hostPath的空间使用大小和类型

!'"#\$%&\$'\$(特性解析 - 网络

☑ !'"#\$%&\$'\$(网络模型

- 集群中所有) *&' +, &\$% 可以直接和其他任意) *&' +, &\$% 通信，不通过 . / 0
- 任意 &*1\$ 主机可以和任意) *&' +, &\$% 通信，不通过 . / 0
- 任意) *&' +, &\$% 自己本身拥有的, 2, 和外界) *&' +, &\$% 看到它的, 2 一致
- 3452\$%2*1 模式：一个 2*1 内的) *&' +, &\$% 可以共享一个, 2 和一套 2*%' (
- 高度耦合容器之间的通信，通过 2*1 和 6*) + 67*(' 方式解决

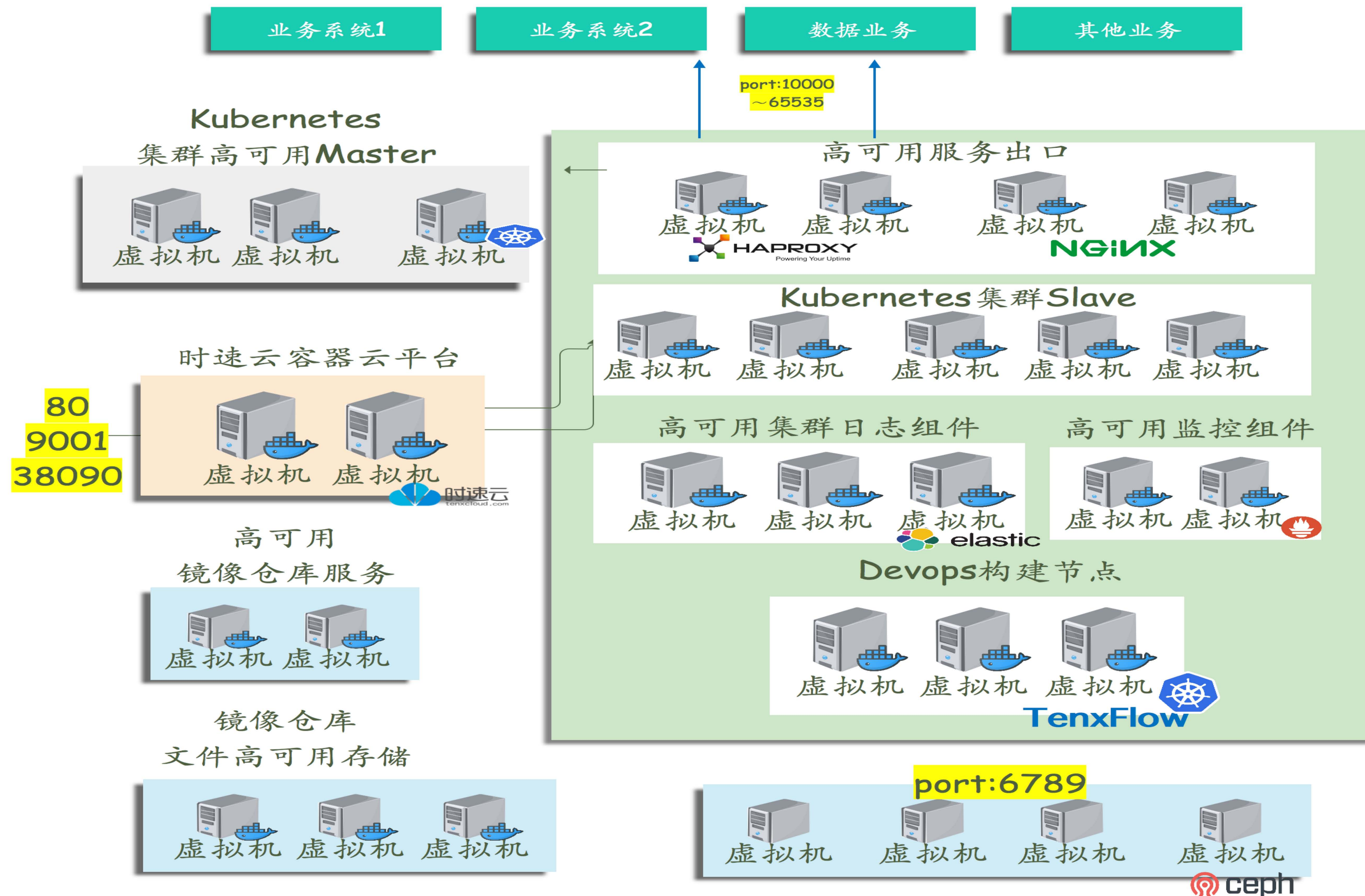
★ 实现方式

- 隧道、路由，calico、86+&&\$6 92\$& ::<,')7 、=\$+:\$ 等

★ 注意事项

- 内网给每个 2*1 动态分配 34 地址，地址可能会因 2*1 的重建而变动，通过; \$%:,)\$- 环境变量或者 1&(方式的方式解决
- 随着集群规模的扩大，注意调整 (\$%:,)\$56"(' \$%525% + &>\$ 网段设置

!'"#\$%&\$'\$(生产环境实施



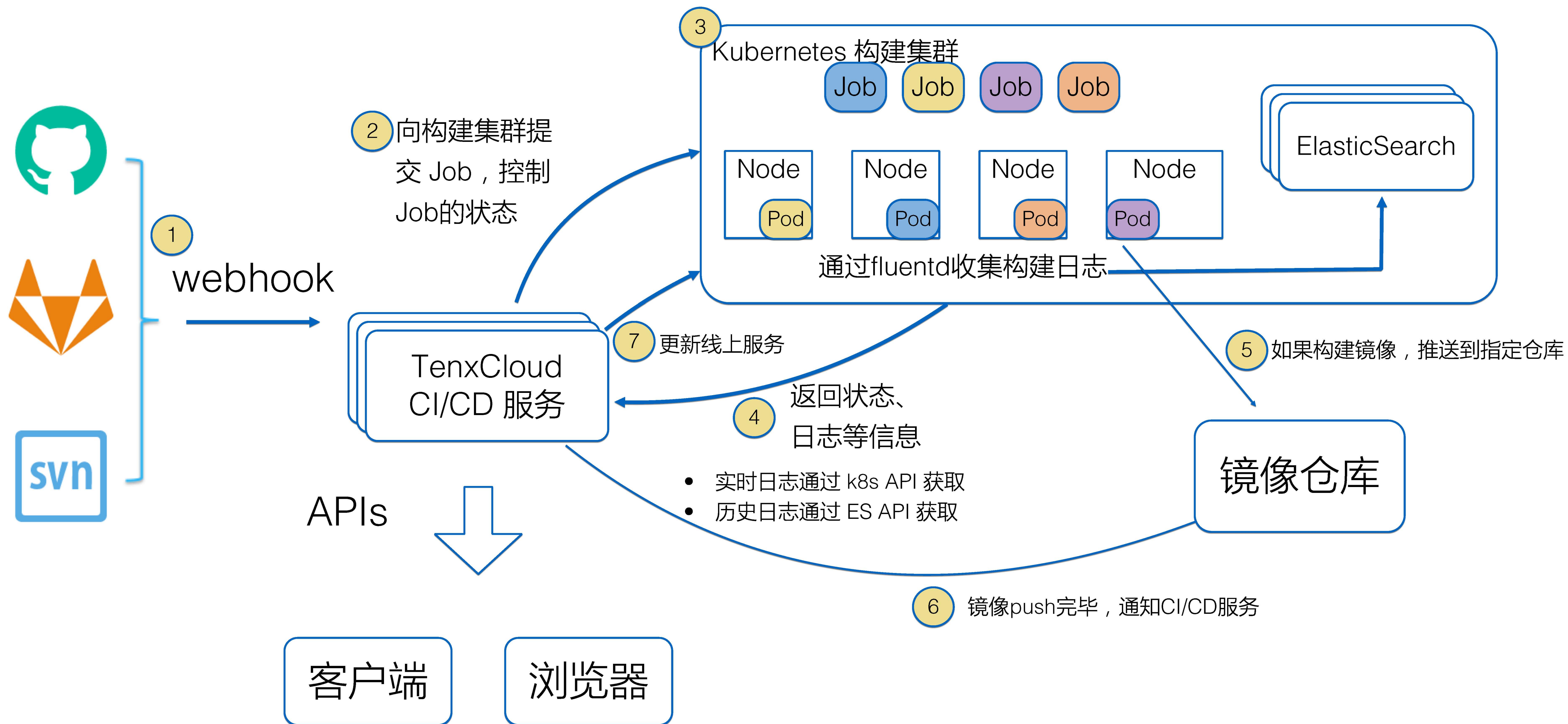
1、基于Docker和Kubernetes的平台建设

- ! PaaS平台落地实施关注的问题
- ! kubernetes核心组件
- ! kubernetes资源对象
- ! kubernetes生产环境实施

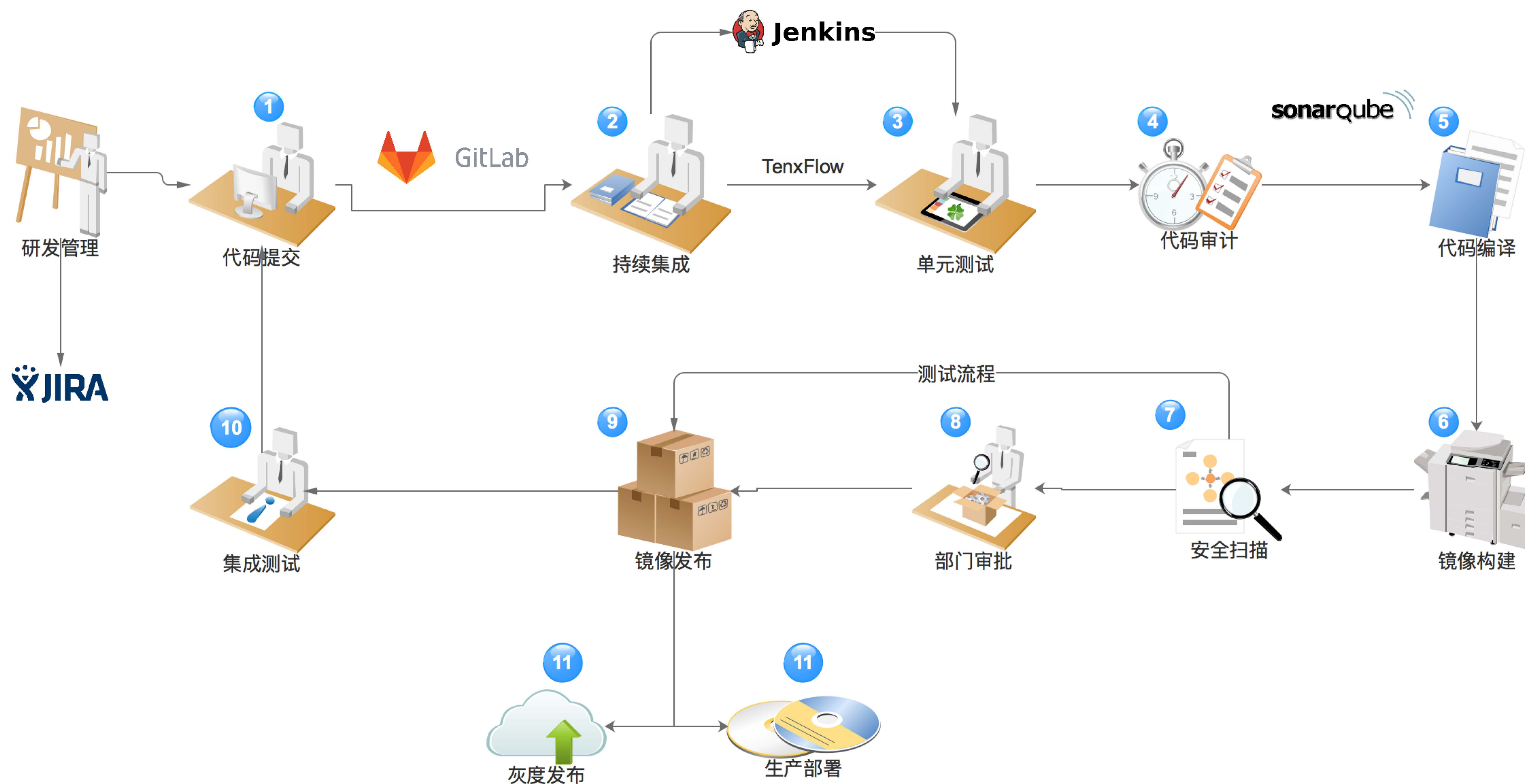
2、容器云平台在企业中的应用场景

- ! 容器云平台功能体系
- ! Devops体系
- ! 测试生产运行体系
- ! 微服务治理

Devops体系-持续集成架构



Devops体系-打造研发管理流程体系

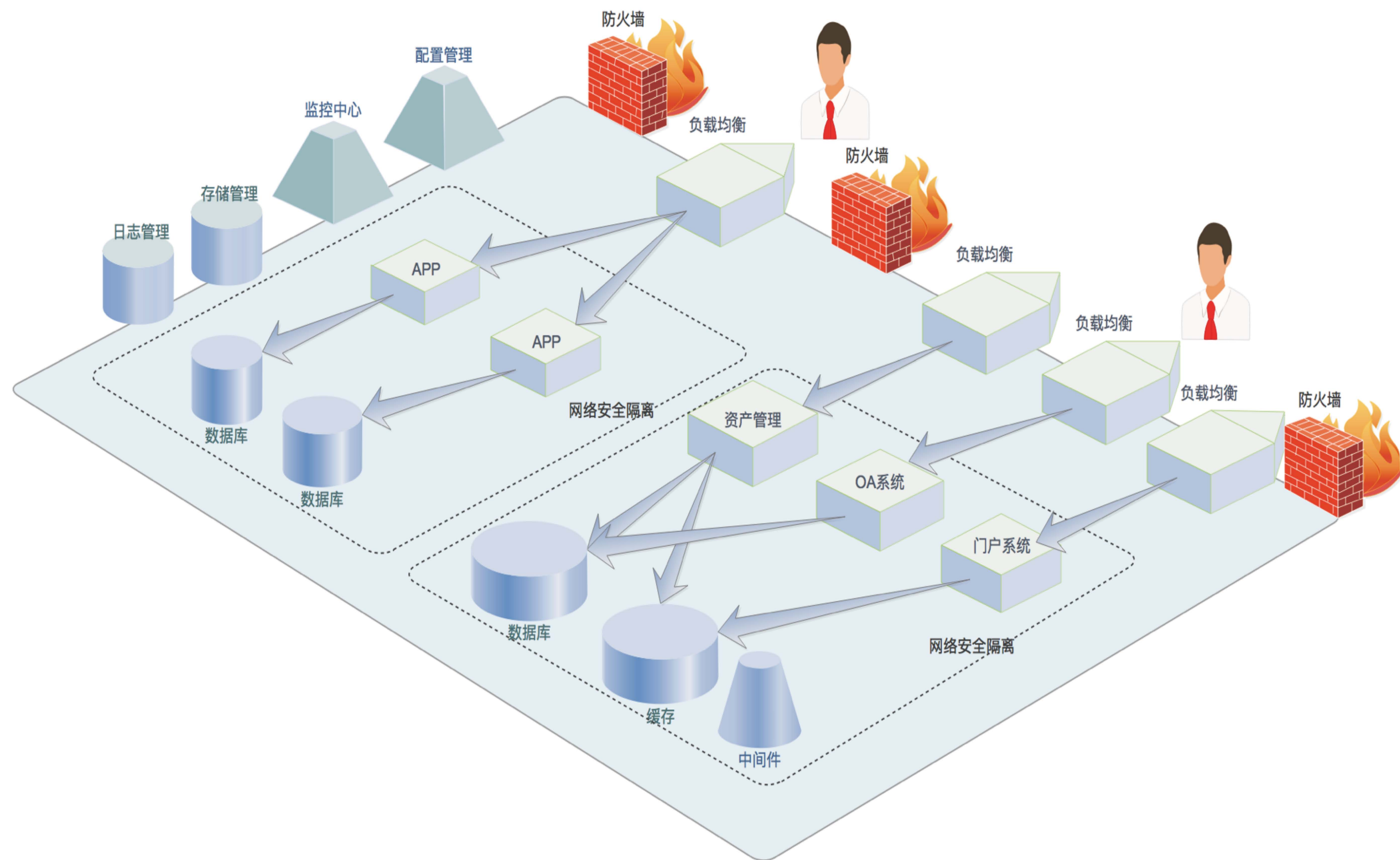


测试生产运行体系



应用部署运行实践

- 1.应用程序容器化
- 2.应用容器生命周期管理
- 3.应用容器间的编排
- 4.持久化存储
- 5.服务发布出口



谢谢！