有容云
Yourun Cloud

有容云产品化 Kubernetes 容器云平台实践

-- 邓绍军（有容云高级软件工程师）

深圳市云舒网络技术有限公司

目录

Kubernetes介绍

kubernetes起源

来源：Google Borg

用于资源管理	平均cell大小有10k个节点 cell内设备可以是异构的 cell组成cluster cluster组成site
用于任务调度	以job为核心 高利用率 高可用性 隐藏资源管理和故障处理细节

kubernetes

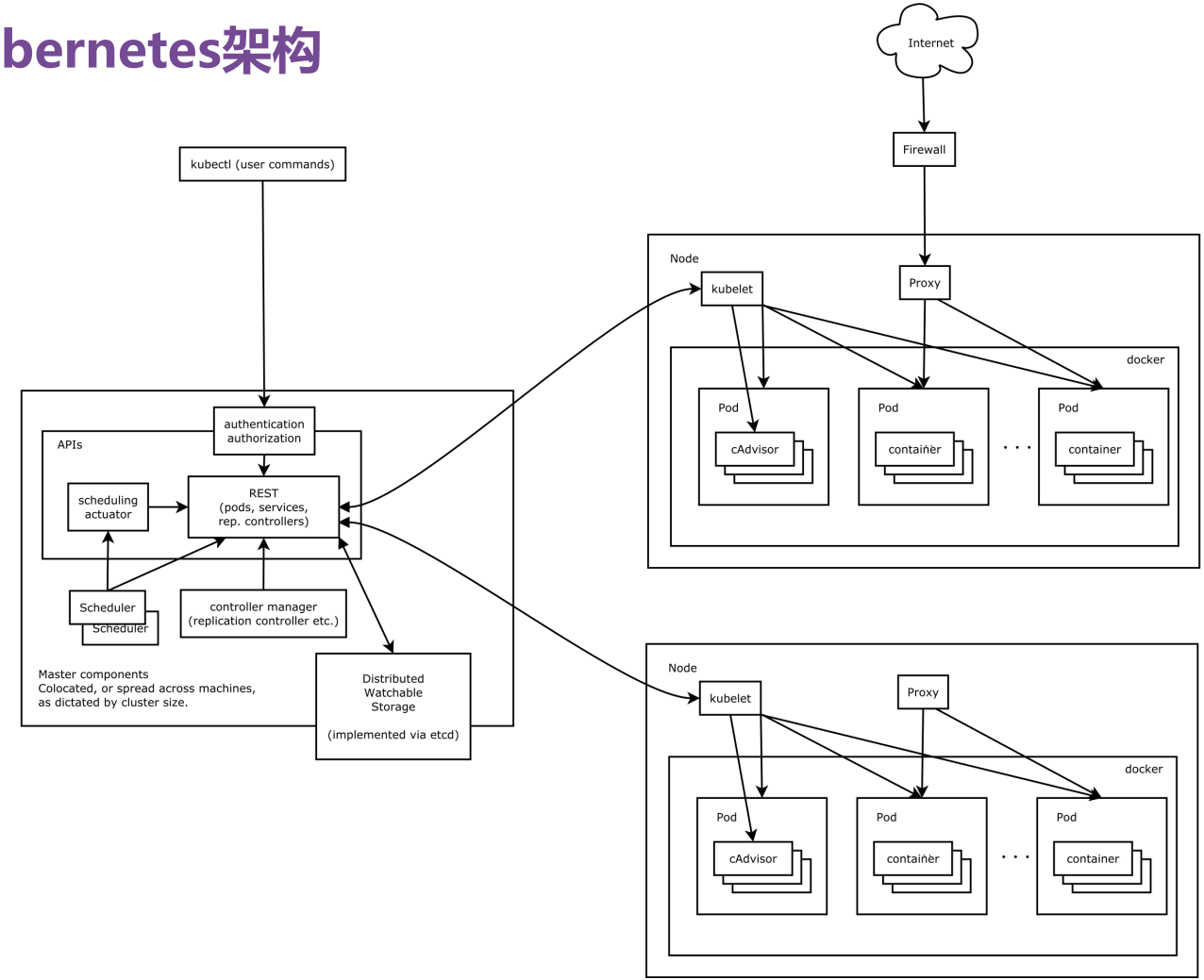
是什么: 开源容器集群管理系统

开源	社区推动,Google,Redhat,CoreOS,Microsoft, VMWare
容器	以容器为中心,提供对应用容器的部署,规模控制,操作
集群	跨主机,多主机所组成的集群

不是什么: 传统的,完整的PAAS平台

不限制支持的应用类型,应用在容器中可运行即可
无内嵌中间件、框架、数据库、集群存储
无一键部署的应用商店
不负责CI流程
允许用户自选监控、日志、告警方案进行集成
不负责主机设备的配置、维护、管理

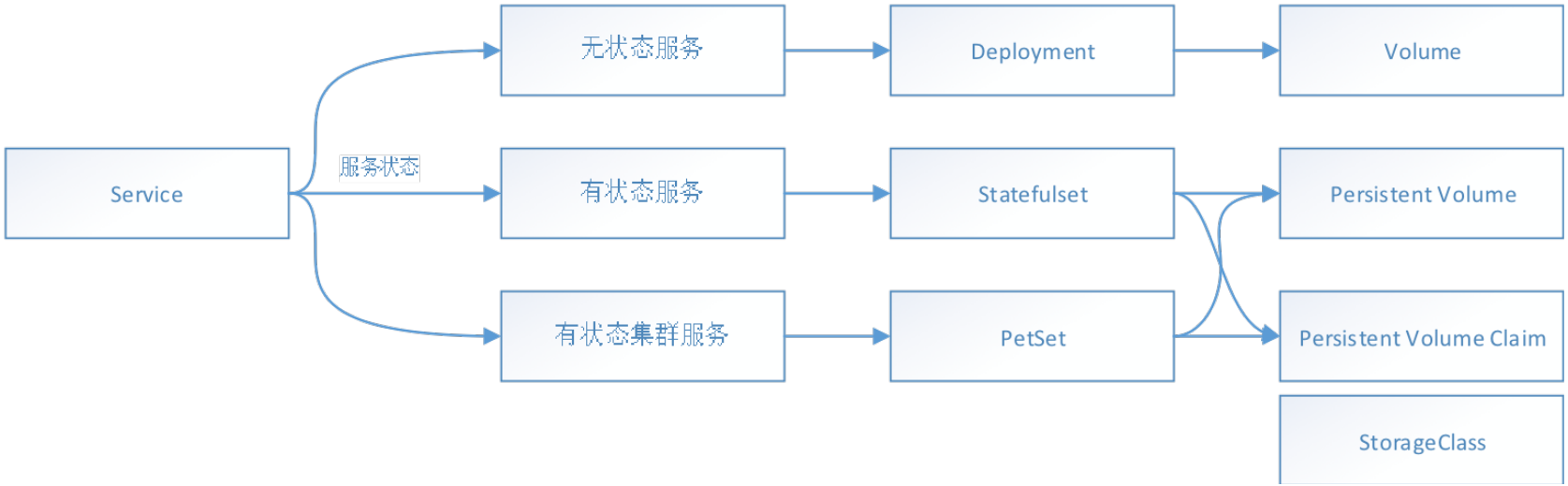
kubernetes架构



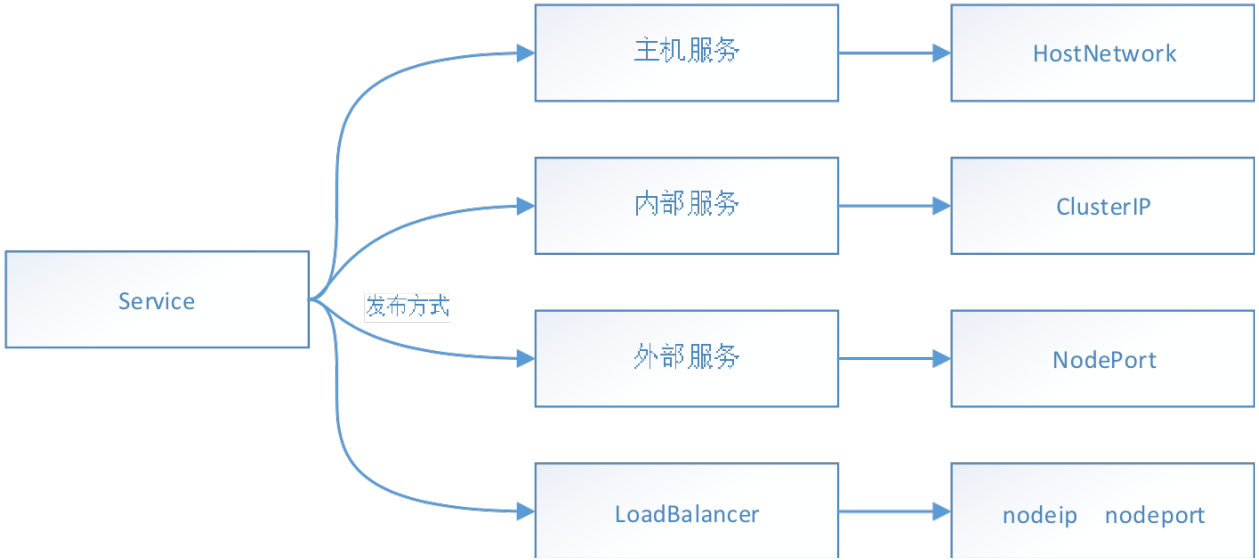
架构说明

etcd	KV store
api server	提供kubernetes api，以RESTFul接口方式提供给外部和内部组件调用，封装操作对象，并将操作对象持久化到etcd中。
scheduler	集群的调度器，负责pod在集群节点中的调度分配。
controller-manager	负责各种集群控制器的执行。
kubelet	管理某节点上的pod的生命周期，同时汇报node的相关信息
kube-proxy	提供网络代理和负载均衡，对应service，并根据service创建代理。

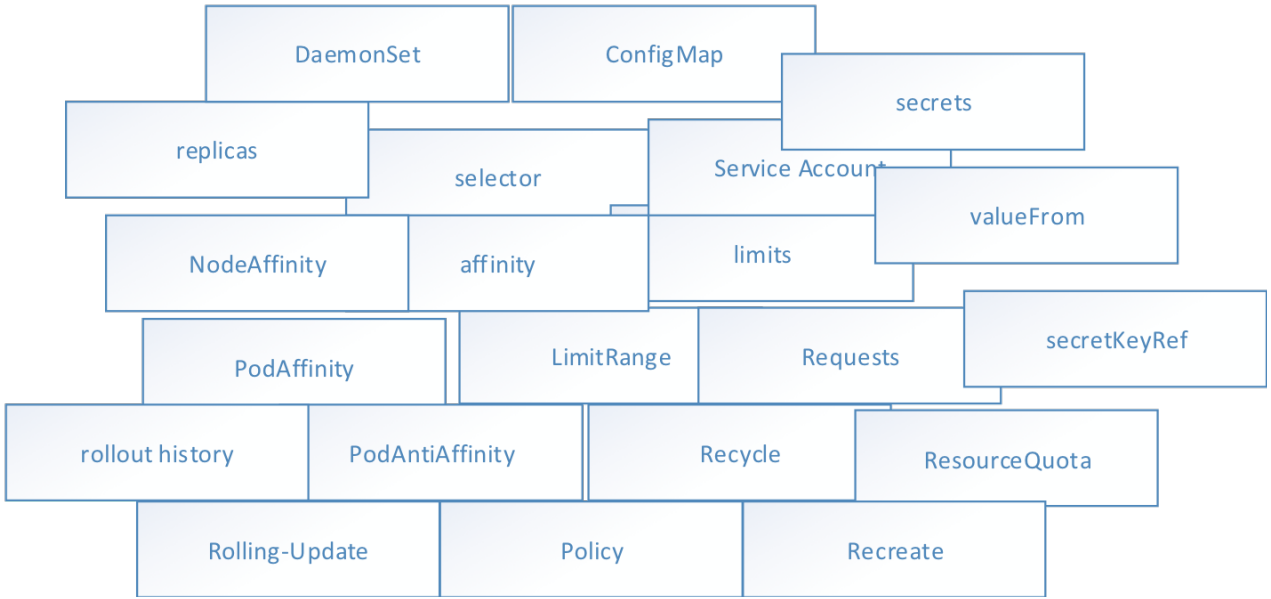
主要概念



主要概念



主要概念



Kubernetes网络

Kubernetes网络

容器网络发展到现在，已经是双雄会的格局。双雄会其实指的就是Docker的CNM和Google、CoreOS、Kuberenetes主导的CNI。首先明确一点，CNM和CNI并不是网络实现，他们是网络规范和网络体系，从研发的角度他们就是一堆接口，你底层是用Flannel也好、用Calico也好，他们并不关心，CNM和CNI关心的是网络管理的问题。

双雄会

CNM

Container Network Model

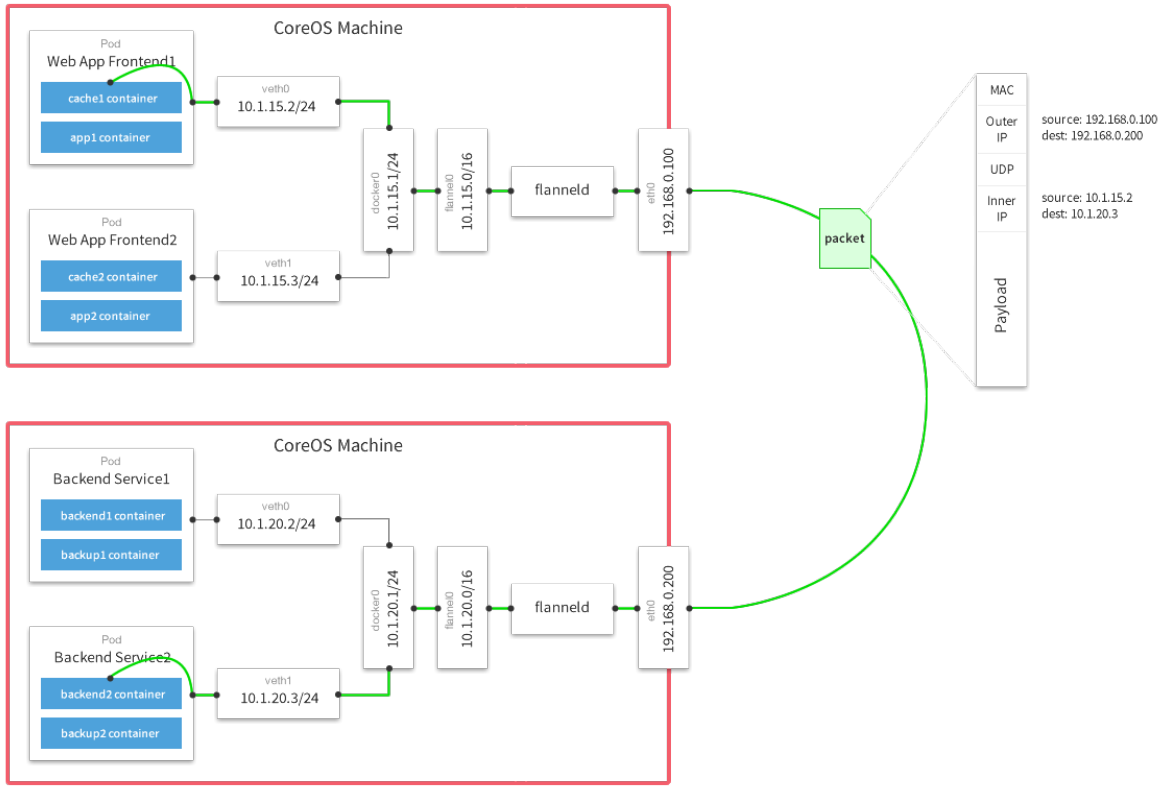
- ✓ Docker自带
- ✓ Docker 命令行直接支持
- ✓ 插件化，Docker提供家长式管理，灵活度不高

CNI

Container Network Interface

- ✓ Docker 不原生支持
- ✓ 需要主动式激活或是用Tricky的办法搞定
- ✓ 插件化，灵活度高

Kubernetes网络-Flannel



Flannel原理：

- 数据从源容器中发出后，经由所在主机的docker0虚拟网卡转发到flannel0虚拟网卡，这是个P2P的虚拟网卡，flanneld服务监听在网卡的另外一端（Flannel通过Etcd服务维护了一张节点间的路由表）；
- 源主机的flanneld服务将原本的数据内容UDP封装后根据自己的路由表投递给目的节点的flanneld服务，数据到达以后被解包，然后直接进入目的节点的flannel0虚拟网卡，然后被转发到目的主机的docker0虚拟网卡；
- 最后就像本机容器通信一下的有docker0路由到达目标容器，这样整个数据包的传递就完成了。

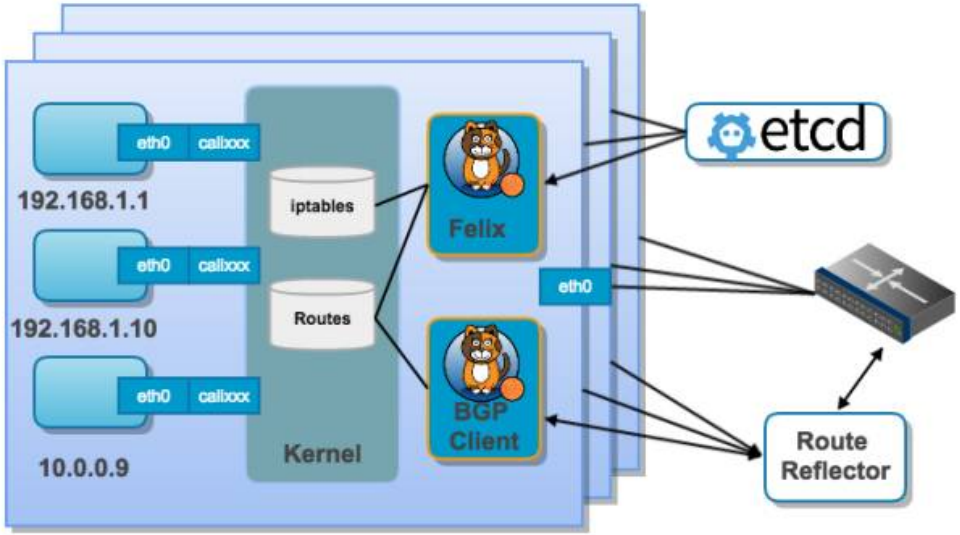
Kubernetes网络-Flannel

Flannel技术特点：

- 每台主机一个CIDR，三层互通
- 主机间有多种封包方式， udp、vxlan、host-gw、aws-vpc、gce和alloc等
- 容器间IP联通但对外服务需要映射到主机IP和端口
- 设计主机CIDR不够灵活，造成大量的IP浪费

Kubernetes网络-Calico

Calico架构图



Felix，Calico Agent，跑在每台需要运行Workload的节点上，主要负责配置路由及ACLs等信息来确保Endpoint的连通状态；

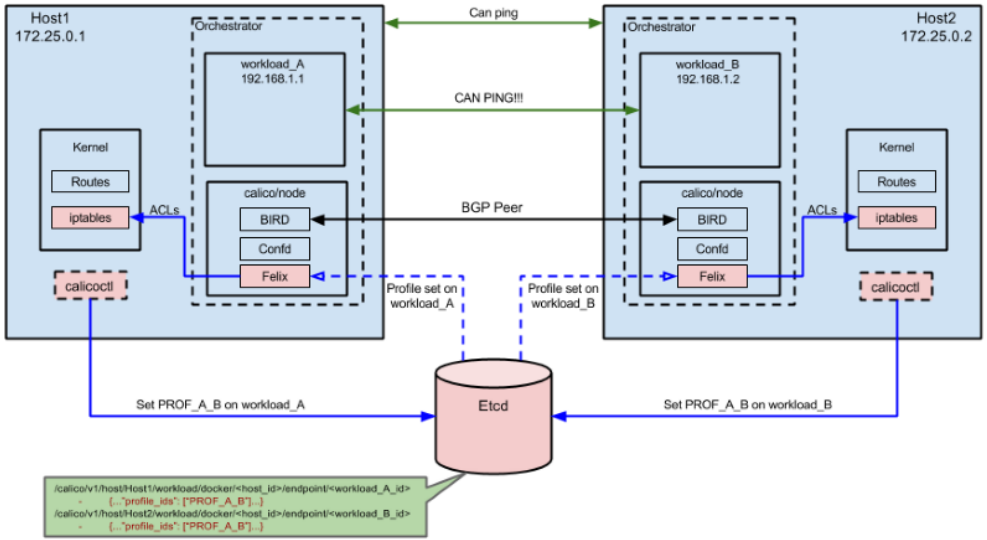
etcd，分布式键值存储，主要负责网络元数据一致性，确保Calico网络状态的准确性；

BGP Client（BIRD），主要负责把Felix写入Kernel的路由信息分发到当前Calico网络，确保Workload间的通信的有效性；

BGP Route Reflector（BIRD），大规模部署时使用，摒弃所有节点互联的 mesh 模式，通过一个或者多个BGP Route Reflector来完成集中式的路由分发。

Kubernetes网络-Calico

技术特点：
✓纯粹的三层实现
✓通过BGP路由
✓扩展性非常好，没有隧道技术，性能好
✓可用于容器、虚拟机、物理机
✓外界可以直接通过路由访问IP，也可以主机上做端口映射
✓需要数据中心内路由器做配置适应



Kubernetes存储

Kubernetes存储定义

Kubernetes对存储抽象了两种存储卷：Volume，Persistent Volume

- Volume： 与Pod之间是静态绑定

  与Docker的存储卷类似，使用的是Pod所在Kubernetes节点的本地目录，分为两种：

  1. emptyDir：一个匿名的空目录，在创建Pod时创建，删除Pod时删除。
  2. hostPath：在Pod之外独立存在，由用户指定宿主机上的文件或目录。
  3. 跨节点卷： 独立于Kubernetes节点存在，可在多个节点上访问使用。

Kubernetes存储定义

Kubernetes对存储提供三个层次上的定义：Volume，Persistent Volume 和动态存储供应(dynamic provisioning)。

- Persistent Volume(PV)：与Pod之间动态绑定

PV是kubernetes的资源对象，可以单独创建PV，借助Persistent Volume Claim(PVC)与Pod实现动态绑定，PV先创建分类，PVC请求已创建的某个类（StorageClass）的资源。

Access Modes（访问模式）

ReadWriteOnce：该卷能够以读写模式被加载到一个节点上。

ReadOnlyMany：该卷能够以只读模式加载到多个节点上。

ReadWriteMany：该卷能够以读写模式被多个节点同时加载。

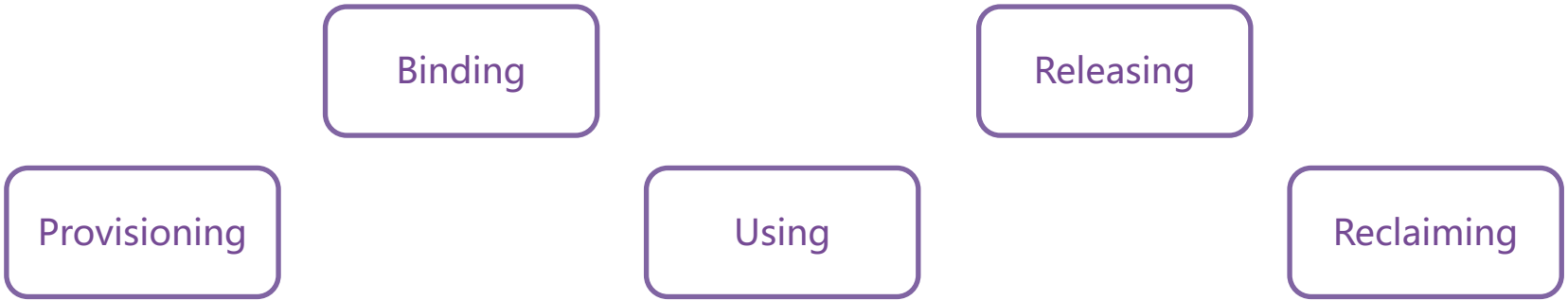
Kubernetes存储定义

Kubernetes对存储提供三个层次上的定义：Volume，Persistent Volume 和动态存储供应(dynamic provisioning)。

- Dynamic provisioning：动态存储供应

  动态卷供给是一个 Kubernetes 独有的功能，这一功能允许按需创建存储卷。动态方式是通过StorageClass来完成的。

PV 和 PVC 的生命周期



平台安全

集群安全机制——Authentication认证

Kubernetes 中的所有资源访问和变更都是通过kube-apiserver提供的REST API实现的，所以集群安全的关键是如何识别并**认证客户端身份**（Authentication）和**访问权限的授权**（Authorization）

Kubernetes提供了以下三种客户端认证方式

- HTTPS证书认证：基于CA证书签名的双向数字证书认证方式。
- HTTP Token认证：通过一个Token来识别用户。
- HTTP Base认证：用户名+密码的方式认证。

集群安全机制——Authorization授权

API Server 授权支持以下几种：

- AlwaysDeny 拒绝所有请求，一般用于测试。
- AlwaysAllow 表示接受所有的请求。
- ABAC 基于属性的访问控制。

当授权模式为ABAC是需要在kube-apiserver启动命令行中加入--authorization-policy-file=FILE

示例：alice拥有所有的操作权限

```
{
    "apiVersion": "abac.authorization.kubernetes.io/v1beta1",
    "kind": "Policy",
    "spec": {
        "user": "alice",
        "namespace": "*",
        "resource": "*",
        "apiGroup": "*"
    }
}
```

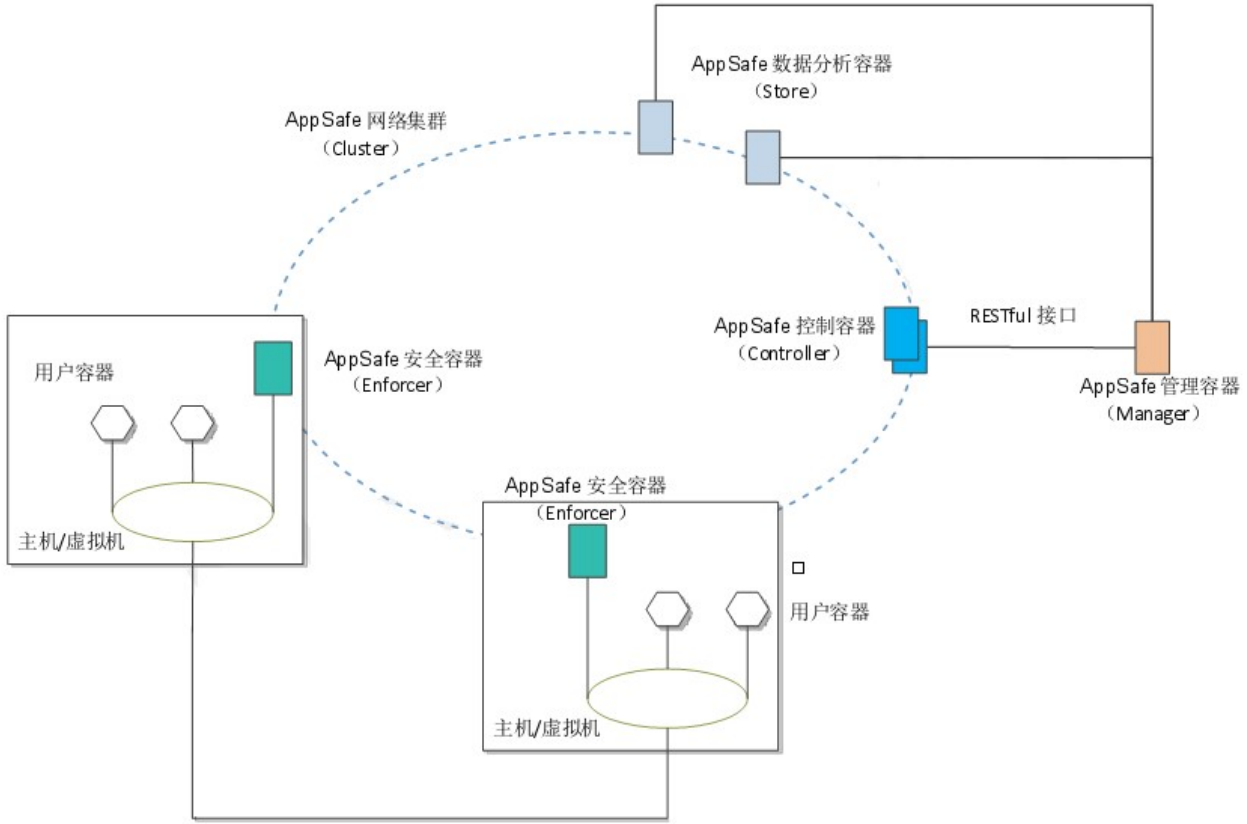
集群安全机制——Admission Control准入控制

当一个请求通过了认证和鉴权之后还需要通过Admission Control才能访问到API Server。

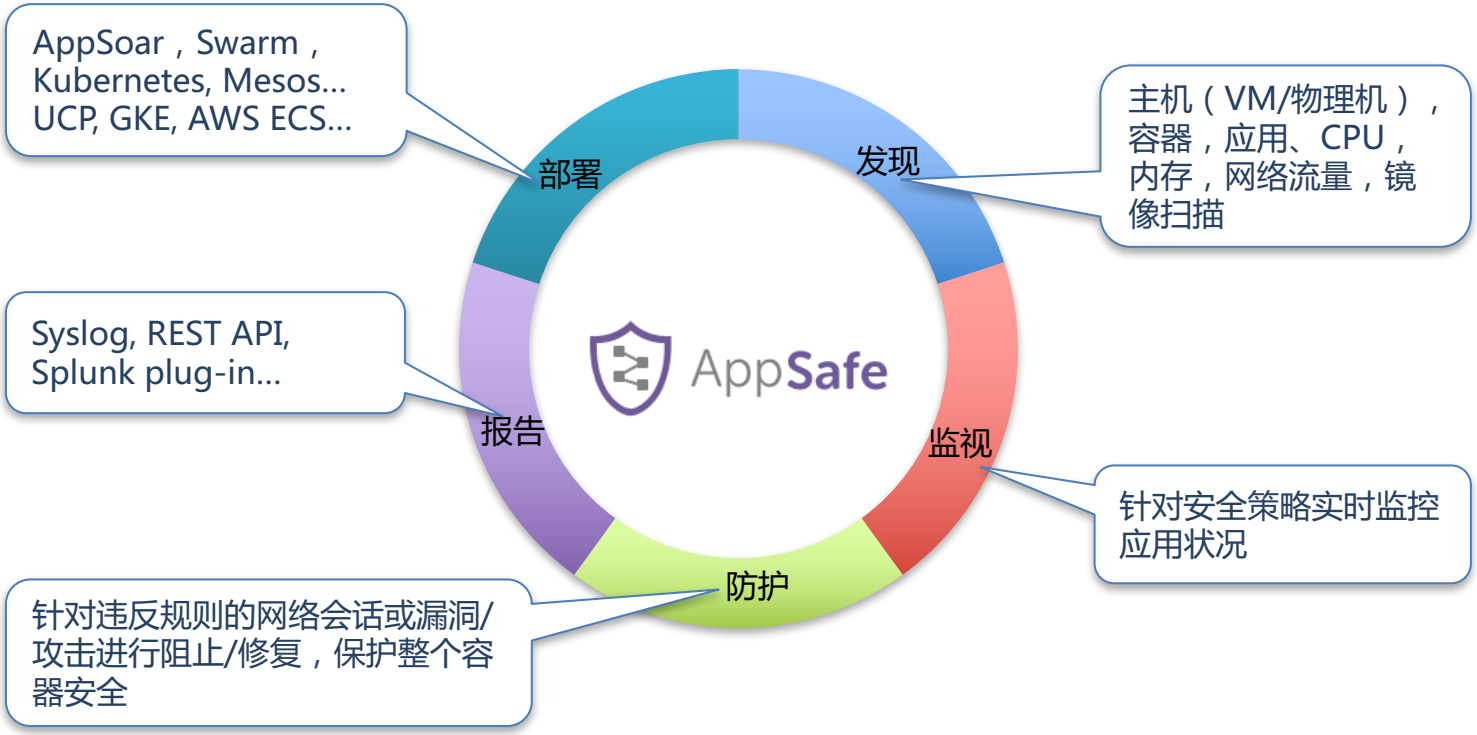
常用的配置项

控制器	说明
AlwaysAdmit	允许所有请求
AlwaysPullImages	在每个新的Pod都会去下载镜像，类容器中配置了imagePullPolicy=Always。
DenyExecOnPrivileged	拦截所有在privileged container执行命令的请求。
Service Account	实现了serviceAccount自动化。如果你需要使用Service Account则需要开启。
SecurityContextDeny	使用了该插件Pod中定义的securityContext（定义了容器在操作系统中的安全设定如：uid,gid,capabilities、SELinux）全部选项将失效。
ResourceQuota	用于配额管理使用，作用于Namasapce上，它会观察所有的请求，确保不会超额，建议配置在参数列表的最后
LimitRanger	用于配额管理，作用于Pod与Container上。确保配额不会超标。
NamespaceLifecycle	如果尝试在一个不存在的namasapce上创建对象，请求会被拒绝。当删除一个Namespace时，会删除Namesapce中所有的资源对象

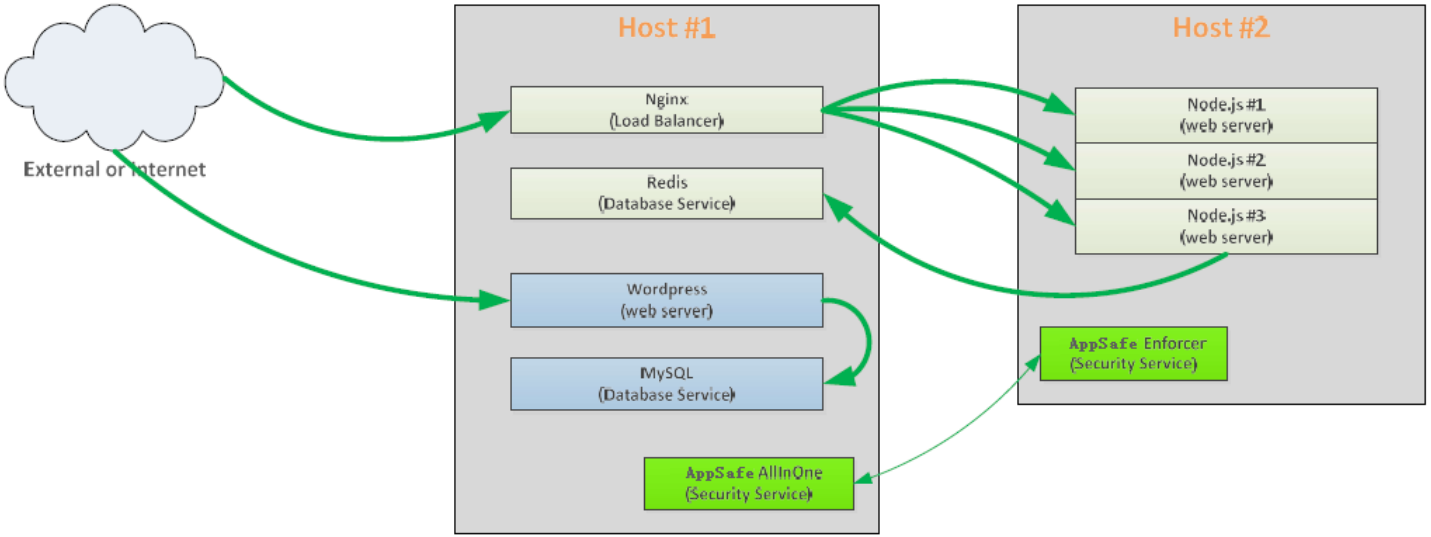
应用安全机制——AppSafe架构



应用安全机制——AppSafe工作流程



应用安全机制——AppSafe解决方案



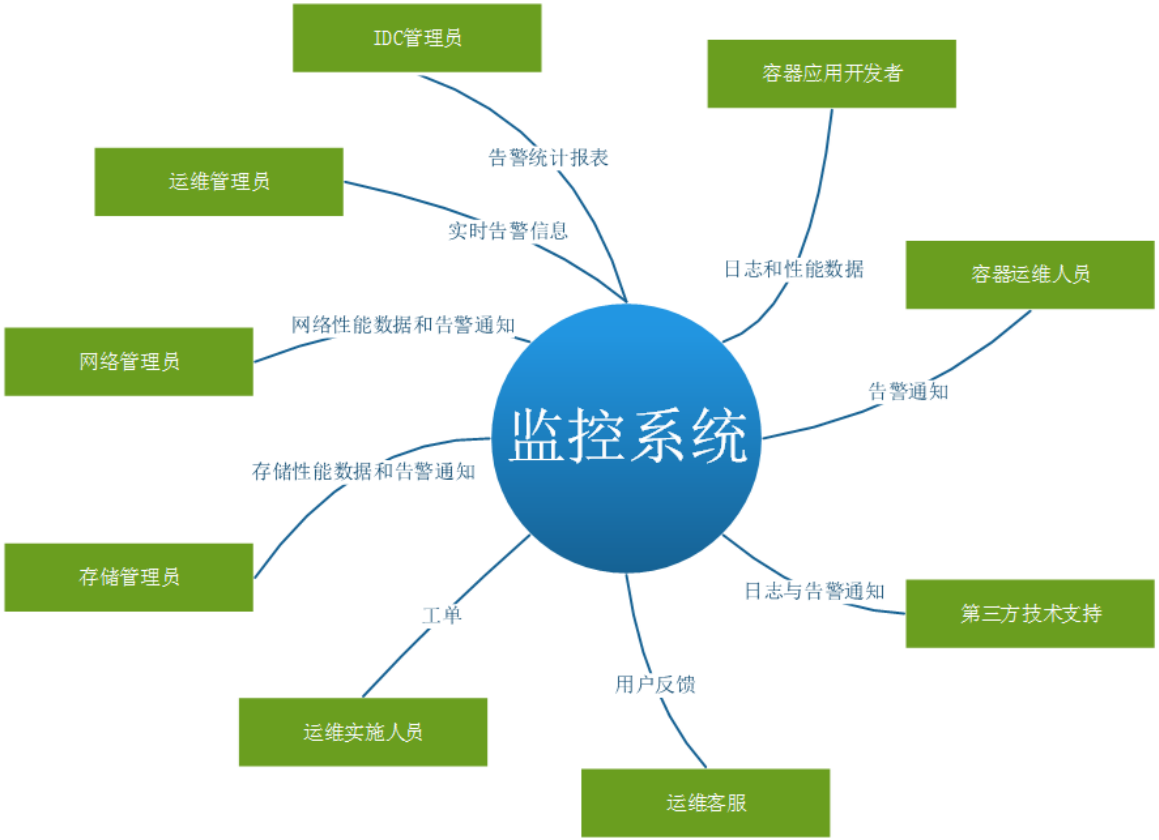
1、Node.js应用（3个Node.js Web服务、1个Nginx代代理服务、一个数据库服务

2、Wordpress应用（1个Wordpress服务、一个数据库服务）

3、自动分组：发现 — 监视 — 防护

容器与应用监控

监控系统——介绍



有容云
Yourun Cloud

监控系统——方案

组合	告警	特点	适用
docker stats	N	最简单的命令行而已	容器监控
Sysdig	N	命令行工具	OS监控与系统故障排查
sematext	Y	服务好 日志按处理量收 云形态的是免费	容器和应用监控
Nagios	Y	轻量，配置复杂，侧重告警	只需捕获告警的场景
Zabbix	Y	大量定制工作是必须的	大部分的泛互联网企业
open-falcon	Y	吸收了Zabbix的长处，功能模块分解比较细，显得比较复杂	OS监控和部分应用监控
cAdvisor+InfluxDB+Grafana	N	做数据收集和存档很强大	容器监控
cAdvisor+Prometheus+Grafana	Y	部署方便	容器监控
exporter+Prometheus+Grafana	Y	扩展性非常好的监控方案	应用监控，容器监控，主机监控都有需求的企业

监控系统——Prometheus介绍

Prometheus是一个开源的系统监控和报警工具

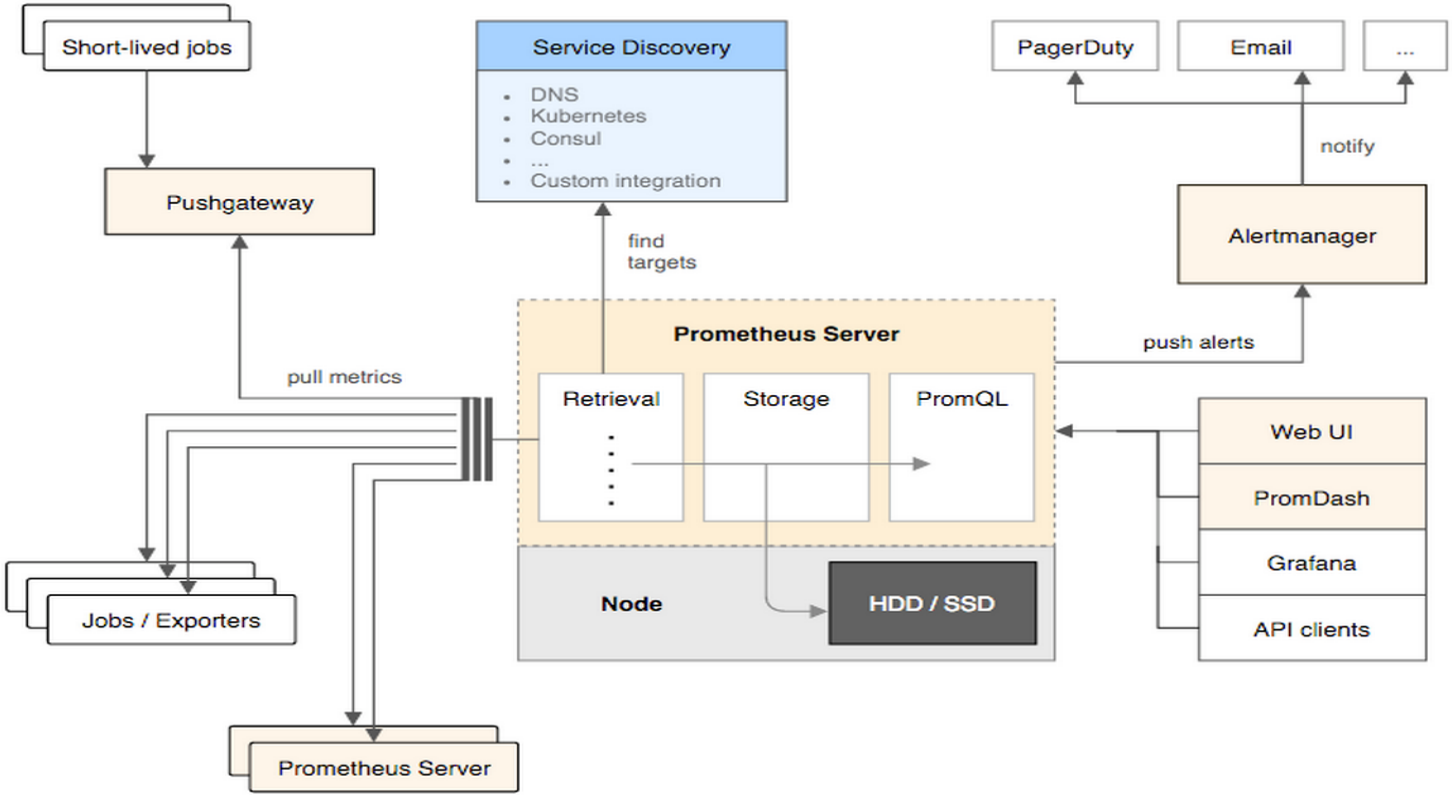
特点：

- ✓ 多维数据模型（时序列数据由metric名和一组key/value组成）
- ✓ 在多维度上灵活的查询语言(PromQl)
- ✓ 不依赖分布式存储，单主节点工作
- ✓ 通过基于HTTP的pull方式采集时序数据
- ✓ 可以通过push gateway进行时序列数据推送(pushing)
- ✓ 可以通过服务发现或者静态配置去获取要采集的目标服务器
- ✓ 多种可视化图表及仪表盘支持

监控系统——Prometheus组件

- ✓ Prometheus server
  主要负责数据采集和存储，提供PromQL查询语言的支持
- ✓ 客户端sdk 官方提供的客户端类库
  go、java、scala、python、ruby，其他还有很多第三方开发的类库，支持nodejs、php、erlang等
- ✓ Push Gateway
  支持临时性Job主动推送指标的中间网关
- ✓ PromDash
  使用rails开发的dashboard，用于可视化指标数据
- ✓ exporters
  支持其他数据源的指标导入到Prometheus，支持数据库、硬件、消息中间件、存储系统、http服务器、jmx等
- ✓ alertmanager
  实验性组件、用来进行报警
- ✓ prometheus_cli
  命令行工具
- ✓ 其他辅助性工具

监控系统——Prometheus架构



有容云
Yourun Cloud

有容云 - 构筑企业容器云

www.youruncloud.com