kubernetes能力提升

实施服务部-系统安全组

2021-9-10

目 录

01

运行环境

Lorem ipsum dolor sit amet, consectetur adipisicing elit.Lorem ipsum dolor sit amet, consectetur

集群规划
（规模大小）

类型	规格	其他需求
Master	8C16G	高可用
Work	16C32G	
镜像仓库	8C16G	磁盘（持久化）
Prometheus		磁盘
日志		磁盘

依赖环境

操作系统选型：内核版本选择3.10+

持久化（可选）：NFS、ceph、longhor、openebs

运行时：docker、containerd

推荐工具

- 集群搭建：sealos、RKE、kubesphere（kk）、kubeadm
- 客户端工具：k9s、lens、kubectl

02

应用发布

工作负载（deployment、satefulset、daemonset）

网络服务、负载均衡（NodePort、ingress）

存储：storageClass

灵魂拷问

01

应用发布选什么类型合适?

02

应用对外用什么方式暴露?

03

应用中的数据如何存储?

04

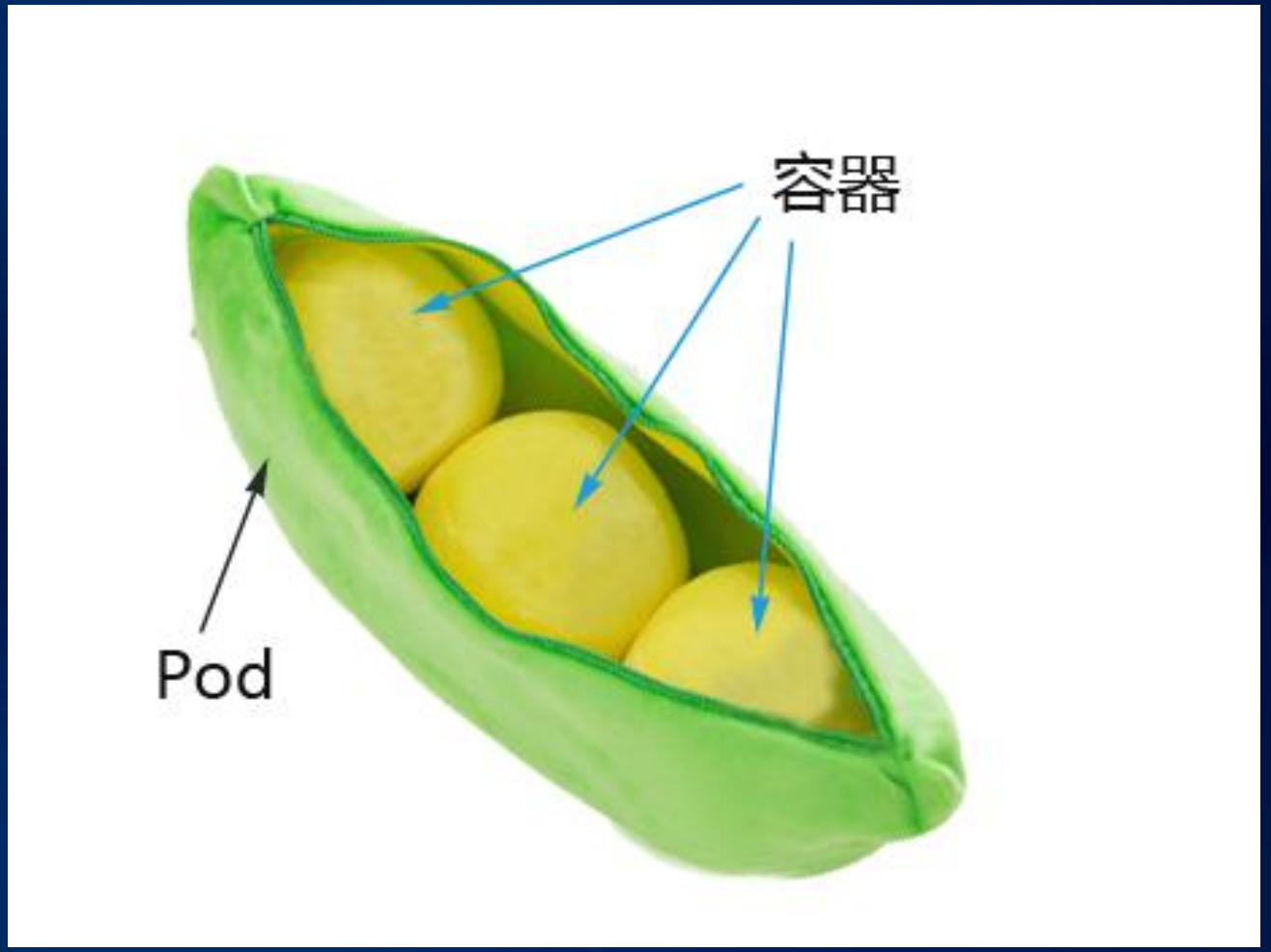
应用怎么保证其可用性?

工作负载资源（workload）

- Pod
- Deployment（无状态）
- StaefulSet（有状态）
- DaemonSet（守护模式）
- Job

Pods

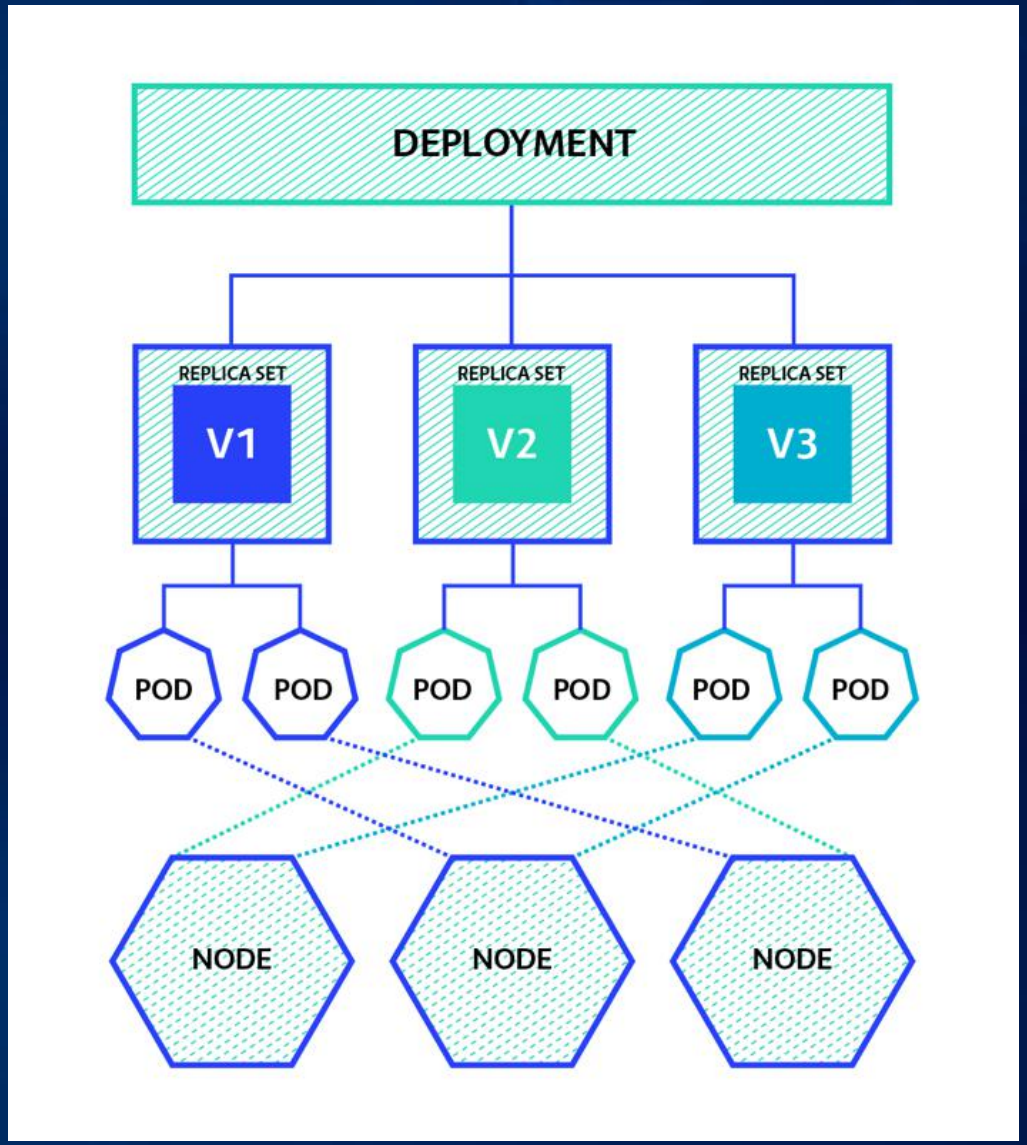
Pod 是可以在 Kubernetes 中创建和管理的、最小的可部署的计算单元。



Deployment

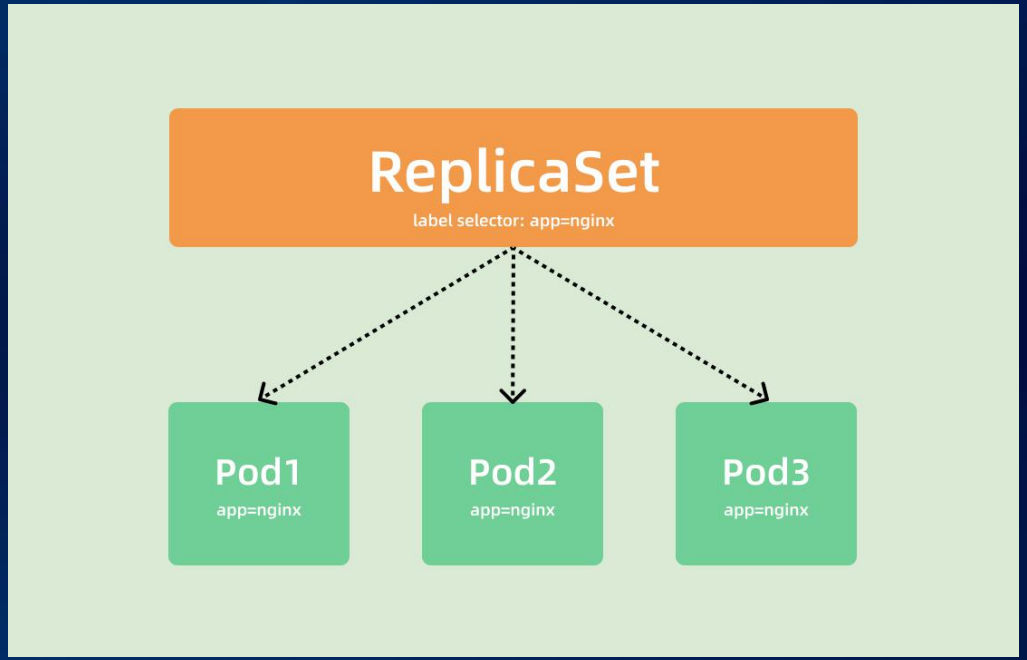
Deployment 为 Pods 和 ReplicaSets 提供声明式的更新能力。

负责描述 Deployment 中的 目标状态，而 Deployment 控制器（Controller）以受控速率更改实际状态， 使其变为期望状态。你可以定义 Deployment 以创建新的 ReplicaSet，或删除现有 Deployment， 并通过新的 Deployment 收养其资源。



ReplicaSet

ReplicaSet 确保任何时间都有指定数量的 Pod 副本在运行。 然而，Deployment 是一个更高级的概念，它管理 ReplicaSet，并向 Pod 提供声明式的更新以及许多其他有用的功能。 因此，我们建议使用 Deployment 而不是直接使用 ReplicaSet，除非 你需要自定义更新业务流程或根本不需要更新。

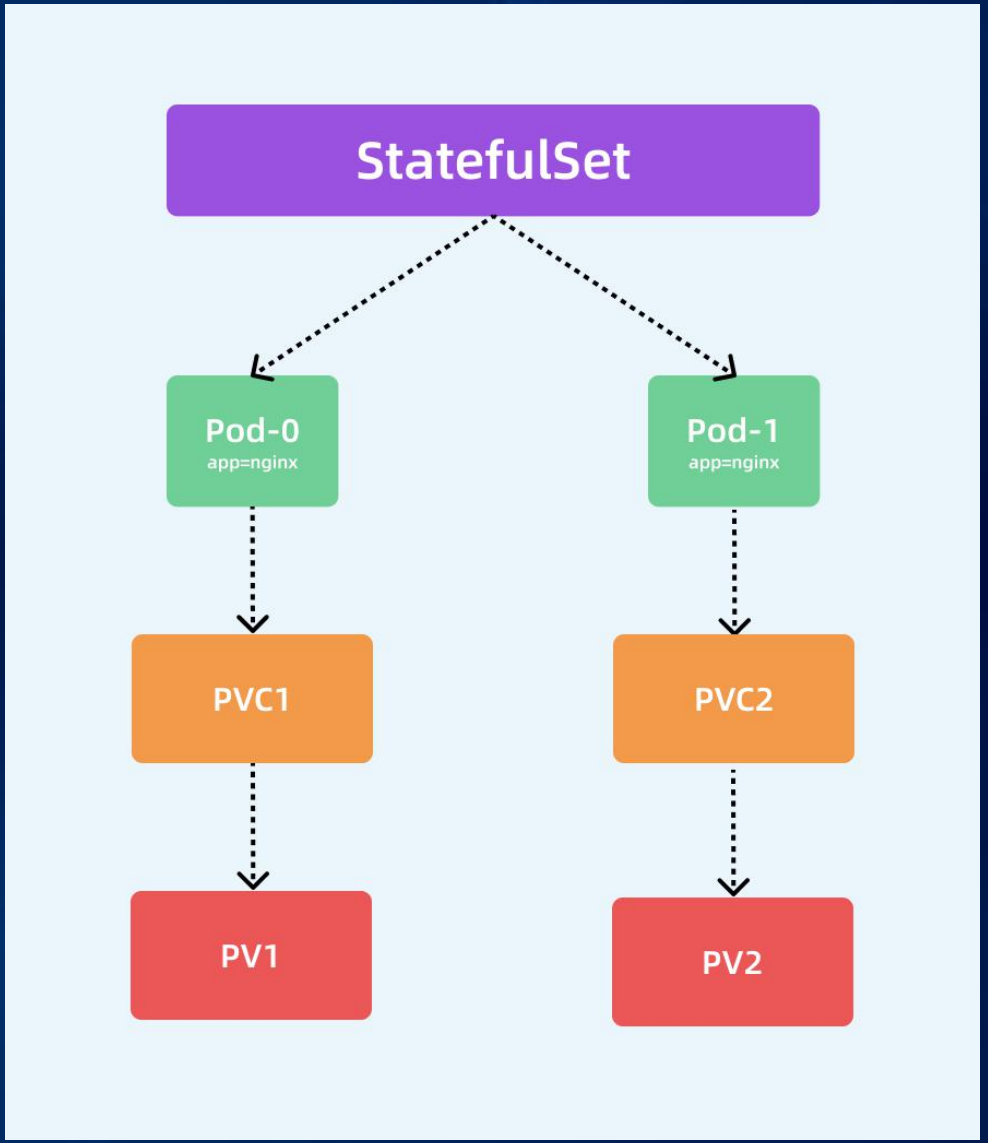


StatefulSet

StatefulSet 是用来管理有状态应用的工作负载 API 对象。

StatefulSet 用来管理某 Pod 集合的部署和扩缩，并为这些 Pod 提供持久存储和持久标识符。

和 Deployment 类似，StatefulSet 管理基于相同容器规约的一组 Pod。但和 Deployment 不同的是，StatefulSet 为它们的每个 Pod 维护了一个有粘性的 ID。这些 Pod 是基于相同的规约来创建的，但是不能相互替换：无论怎么调度，每个 Pod 都有一个永久不变的 ID。

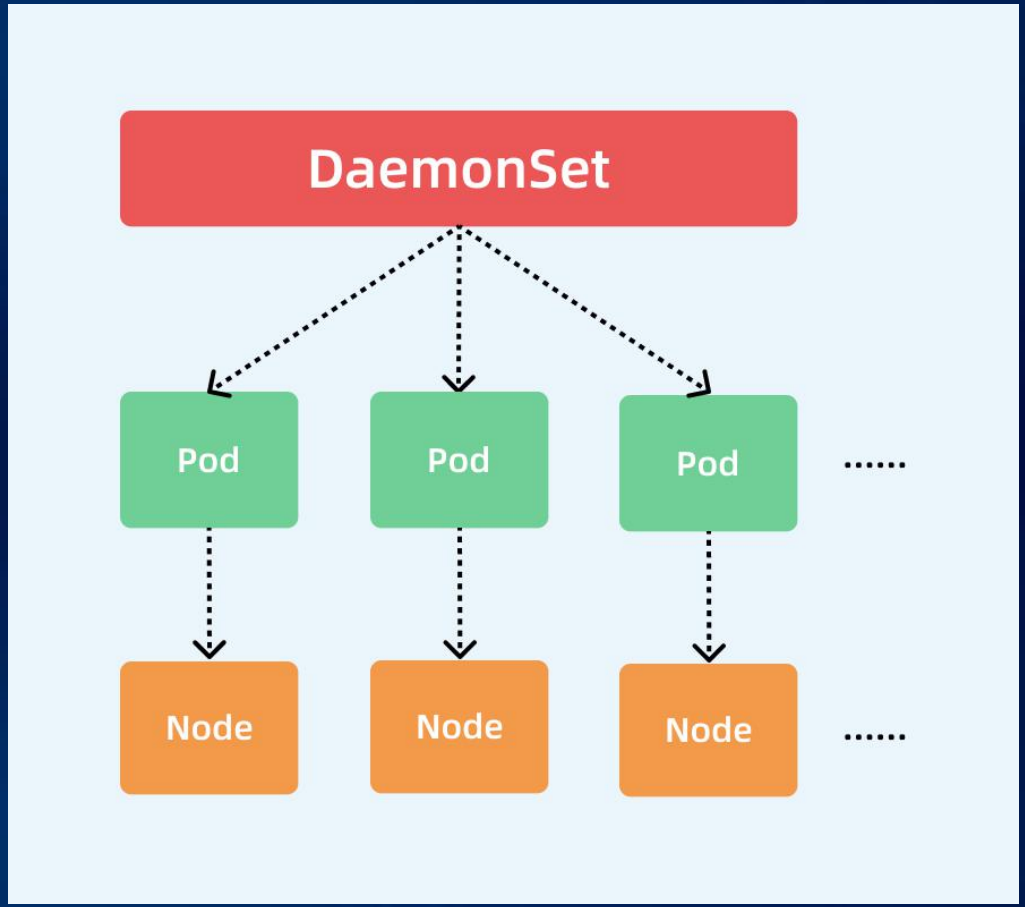


DaemonSet

DaemonSet 确保全部（或者某些）节点上运行一个 Pod 的副本。 当有节点加入集群时， 也会为他们新增一个 Pod 。 当有节点从集群移除时，这些 Pod 也会被回收。删除 DaemonSet 将会删除它创建的所有 Pod。

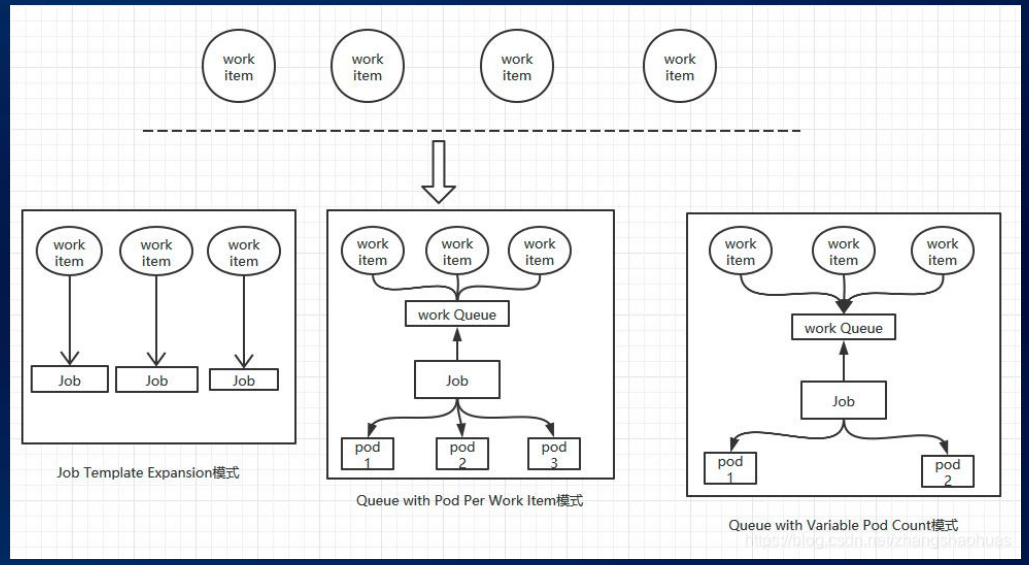
DaemonSet 的一些典型用法：

在每个节点上运行集群守护进程
在每个节点上运行日志收集守护进程
在每个节点上运行监控守护进程



Job&CronJob

Job 对象可以用来支持多个 Pod 的可靠的并发执行。 Job 对象不是设计用来支持相互通信的并行进程的，后者一般在科学计算中应用较多。 Job 的确能够支持对一组相互独立而又有所关联的 工作条目 的并行处理。 这类工作条目可能是要发送的电子邮件、要渲染的视频帧、要编解码的文件、NoSQL 数据库中要扫描的主键范围等等。



CronJob 创建基于时隔重复调度的 Jobs。

网络资源

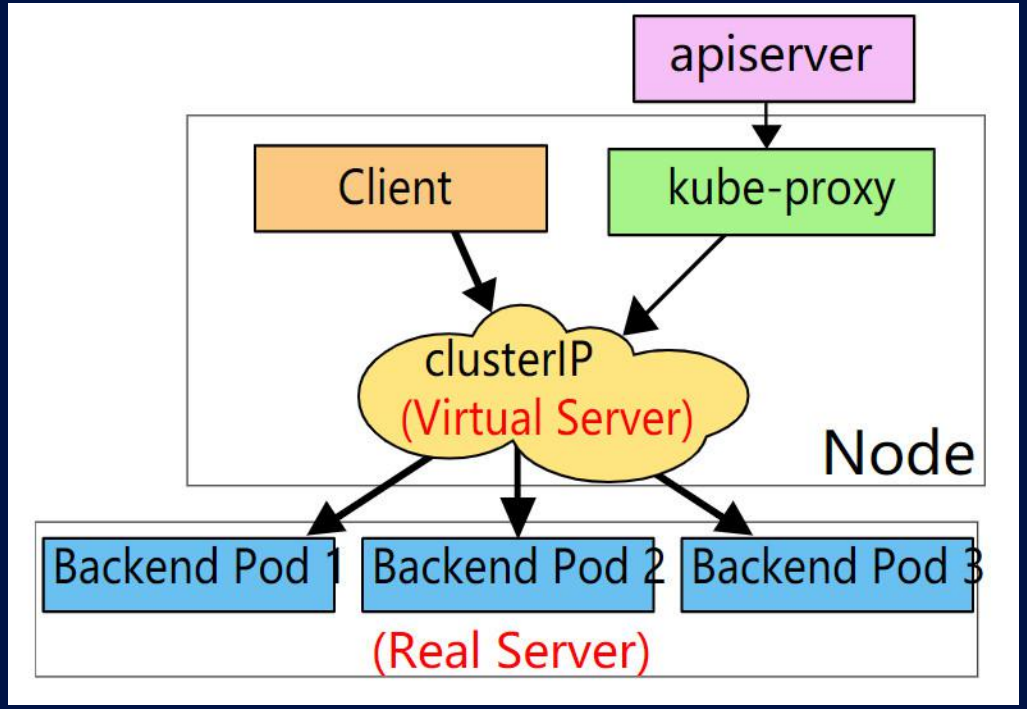
service、ingress

Service

Kubernetes Service 定义了这样一种抽象：逻辑上的一组 Pod，一种可以访问它们的策略 —— 通常称为微服务。 Service 所针对的 Pods 集合通常是通过选择算符来确定的。 要了解定义服务端点的其他方法，请参阅不带选择算符的服务。

举个例子，考虑一个图片处理后端，它运行了 3 个副本。这些副本是可互换的 —— 前端不需要关心它们调用了哪个后端副本。 然而组成这一组后端程序的 Pod 实际上可能会发生变化， 前端客户端不应该也没必要知道，而且也不需要跟踪这一组后端的状态。

Service 定义的抽象能够解耦这种关联。

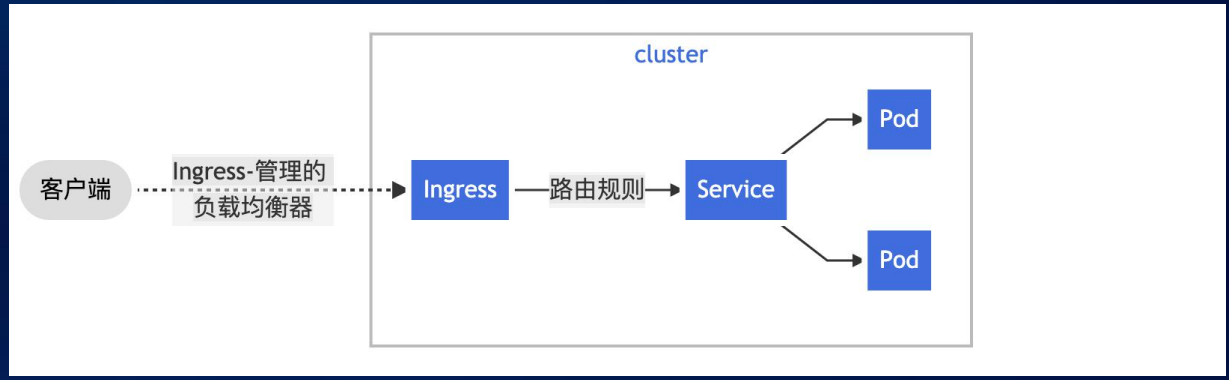


Ingress

Ingress 公开了从集群外部到集群内服务的 HTTP 和 HTTPS 路由。 流量路由由 Ingress 资源上定义的规则控制。

Ingress 是对集群中服务的外部访问进行管理的 API 对象，典型的访问方式是 HTTP。

Ingress 可以提供负载均衡、SSL 终止和基于名称的虚拟托管。



Ingress controller

可以在集群中部署任意数量的 ingress 控制器。 创建 ingress 时，应该使用适当的 ingress.class 注解每个 Ingress 以表明在集群中如果有多个 Ingress 控制器时，应该使用哪个 Ingress 控制器。

其他控制器

注意： 本部分链接到提供 Kubernetes 所需功能的第三方项目。Kubernetes 项目作者不负责这些项目。此页面遵循CNCF 网站指南，按字母顺序列出项目。要将项目添加到此列表中，请在提交更改之前阅读内容指南。

- AKS 应用程序网关 Ingress 控制器 是一个配置 Azure 应用程序网关 的 Ingress 控制器。
- Ambassador API 网关是一个基于 Envoy 的 Ingress 控制器。
- Apache APISIX Ingress 控制器 是一个基于 Apache APISIX 网关 的 Ingress 控制器。
- Avi Kubernetes Operator 使用 VMware NSX Advanced Load Balancer 提供第 4 到第 7 层的负载均衡。
- Citrix Ingress 控制器 可以用来与 Citrix Application Delivery Controller 一起使用。
- Contour 是一个基于 Envoy 的 Ingress 控制器。
- EnRoute 是一个基于 Envoy API 网关， 可以作为 Ingress 控制器来执行。
- Easegress IngressController 是一个基于 Easegress API 网关，可以作为 Ingress 控制器来执行。
- F5 BIG-IP 的 用于 Kubernetes 的容器 Ingress 服务 让你能够使用 Ingress 来配置 F5 BIG-IP 虚拟服务器。
- Gloo 是一个开源的、基于 Envoy 的 Ingress 控制器，能够提供 API 网关功能，
- HAProxy Ingress 针对 HAProxy 的 Ingress 控制器。
- 用于 Kubernetes 的 HAProxy Ingress 控制器 也是一个针对 HAProxy 的 Ingress 控制器。
- Istio Ingress 是一个基于 Istio 的 Ingress 控制器。
- 用于 Kubernetes 的 Kong Ingress 控制器 是一个用来驱动 Kong Gateway 的 Ingress 控制器。
- 用于 Kubernetes 的 NGINX Ingress 控制器 能够与 NGINX Web 服务器（作为代理） 一起使用。
- Skipper HTTP 路由器和反向代理可用于服务组装，支持包括 Kubernetes Ingress 这类使用场景， 设计用来作为构造你自己的定制代理的库。
- Traefik Kubernetes Ingress 提供程序 是一个用于 Traefik 代理的 Ingress 控制器。
- Tyk Operator 使用自定义资源扩展 Ingress，为之带来 API 管理能力。Tyk Operator 使用开源的 Tyk Gateway & Tyk Cloud 控制面。
- Voyager 是一个针对 HAProxy 的 Ingress 控制器。

存储

持久化存储、storageClass

存储概述

- 存储类：nfs、ceph、openebs、Local
- 持久卷（Persistent Volume，PV）
- 持久卷申领（PersistentVolumeClaim，PVC）
- 临时卷：emptyDir、

怎么使用呢？重点

- 1、根据集群中的资源选择合适的存储类(storageClass)，例如NFS（简单）、Longhorn\openebs（一般）、ceph（复杂）等
- 2、根据存储类配置持久类PV（PersistentVolume）
- 3、从配置好的PV（PersistentVolume）中申领PVC（PersistentVolumeClaim）
- 4、workload中应用volumes挂载PVC,volumeMounts挂载到目录或block

- 注意：PV是全局的，面向整个集群
- PVC是面向命名空间

03

应用运维

如何在kubernetes里保障容器的健康状态?

当遇到突发访问容器该如何应对?

服务如何保障不被击垮

应用扩缩容

scale、autoscale

scale&autoscale&HorizontalPodAutoscaler

- 扩容注意事项：
- Pod resource配置

```
        resources:
          requests: #调度时，最小资源
            cpu: 100m
            memory: 100Mi
          limits: #运行时
            cpu: 100m
            memory: 200Mi
```

- 手动扩容
- kubectl scale deployment nginx --replicas 2
- 自动扩容
- kubectl autoscale deployment nginx --min=2 --max=10 --cpu-percent=80
- 手动缩容
- kubectl scale deployment nginx --replicas 1

服务可用性

限流、熔断

服务调度

- nodeSelector（标签）
- affinity（亲和性）
- nodeAffinity（节点亲和性）
- 污点和容忍度

健康检查

- livenessProbe(服务存活状态检查)
- startupProbe(启动探测保护慢启动容器)
- readinessProbe(服务就绪检查)

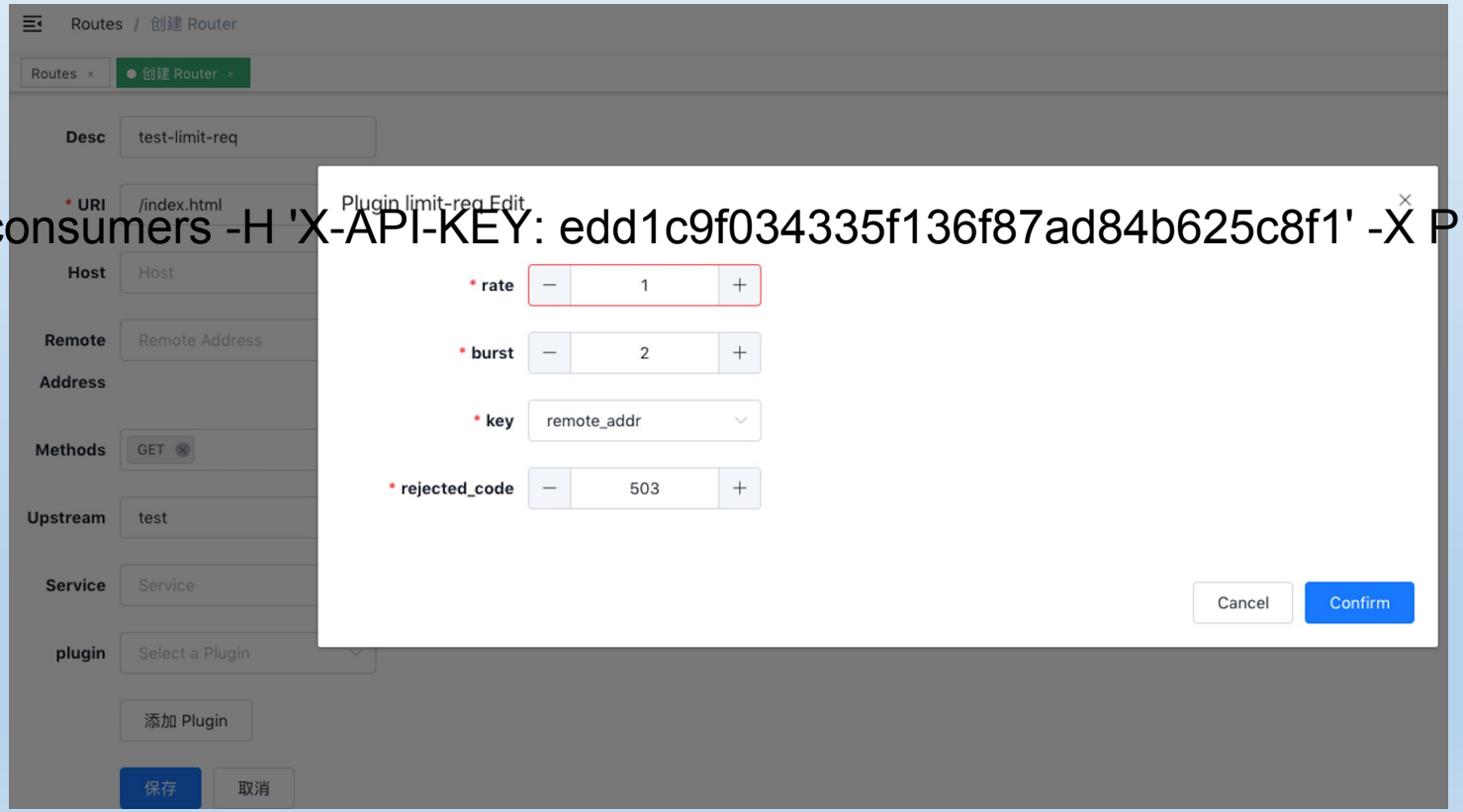
服务限流

- 注意：NodePort不支持7层限流

- 官方ingress限流：
- nginx.ingress.kubernetes.io/limit-connections：来自单个IP地址的并发连接数。
- nginx.ingress.kubernetes.io/limit-rps：每秒可从给定IP接受的连接数。
- nginx.ingress.kubernetes.io/limit-rpm：每分钟可从给定IP接受的连接数。
- nginx.ingress.kubernetes.io/limit-rate-after：设置初始金额，在此之后，对客户的进一步传输响应将受到速率限制。
- nginx.ingress.kubernetes.io/limit-rate：每秒从客户接受的请求率。

APISIX限流

- Apache apisix （国产api网关）基于openresty+lua




```
curl http://127.0.0.1:9080/apisix/admin/consumers -H 'X-API-KEY: edd1c9f034335f136f87ad84b625c8f1' -X PUT -d '
{
    "username": "consumer_jack",
    "plugins": {
        "key-auth": {
            "key": "auth-jack"
        },
        "limit-req": {
            "rate": 1,
            "burst": 1,
            "rejected_code": 403,
            "key": "consumer_name"
        }
```

traefik限流

- apiVersion: traefik.containo.us/v1alpha1
- kind: Middleware
- metadata:
- name: flask-k8s-traffic
- namespace: default
- spec:
- rateLimit:
- # 1s 内接收的请求数的平均值不大于500个，高峰最大1000个请求
- burst: 1000
- average: 500

服务安全性

TLS、Pod securityContext

Pod安全上下文

- securityContext

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Pod网络安全性

- 说明： 除非选择支持网络策略的网络解决方案(CNI)，否则上述示例没有任何效果。
- 目前支持networkpolicy的网络解决方案组件如下：calico、cilium、weave、romana、kube-route
- 默认情况下，Pod 是非隔离的，它们接受任何来源的流量。
- Pod 在被某 NetworkPolicy 选中时进入被隔离状态。 一旦名字空间中有 NetworkPolicy 选择了特定的 Pod，该 Pod 会拒绝该 NetworkPolicy 所不允许的连接。 （名字空间下其他未被 NetworkPolicy 所选择的 Pod 会继续接受所有的流量）

- 网络策略不会冲突，它们是累积的。 如果任何一个或多个策略选择了一个 Pod, 则该 Pod 受限于这些策略的 入站（Ingress）/出站（Egress）规则的并集。因此评估的顺序并不会影响策略的结果。

- 为了允许两个 Pods 之间的网络数据流，源端 Pod 上的出站（Egress）规则和 目标端 Pod 上的入站（Ingress）规则都需要允许该流量。 如果源端的出站（Egress）规则或目标端的入站（Ingress）规则拒绝该流量， 则流量将被拒绝。

ingress添加TLS

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: testsecret-tls
  namespace: default
data:
  tls.crt: base64 编码的 cert
  tls.key: base64 编码的 key
type: kubernetes.io/tls
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  tls:
  - hosts:
      - https-example.foo.com
    secretName: testsecret-tls
  rules:
  - host: https-example.foo.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 80
```

04

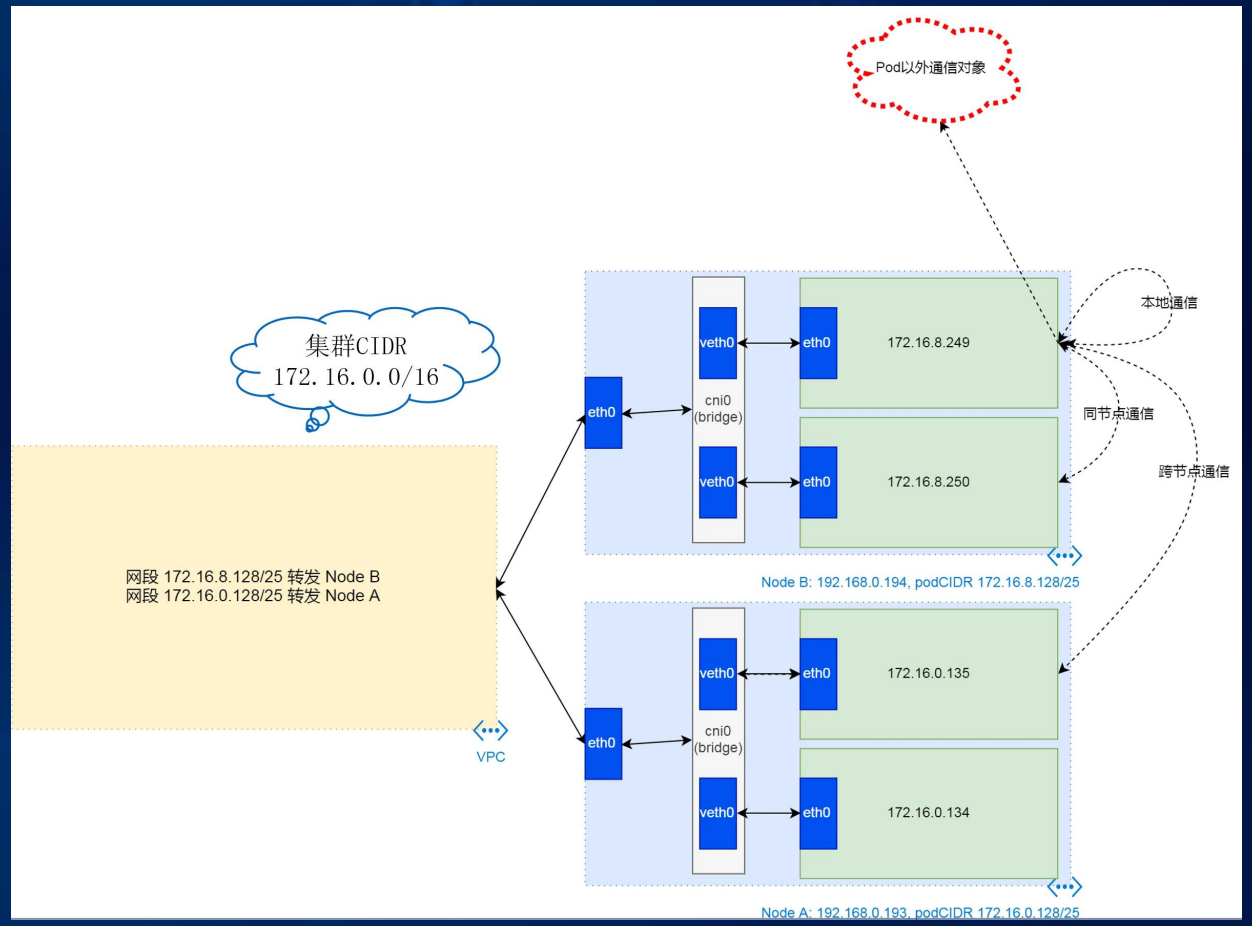
故障排查

集群类故障

应用类故障

pod怎么与集群外的主机通信?

本地->lookback
同节点->cni0
跨节点：cni0-宿主机路由-cni0
非集群：snat



Pod 状态不是running状态怎么排查?

- kubectl describe pod-id
- kubectl logs pod-id
- 检查所在节点磁盘空间是否满了
- 检查私有镜像仓库是否能连接
- 内存碎片化问题
- 系统 OOM

Pod 健康检查失败

- 健康检查配置不合理
- 节点负载过高
- 容器进程被木马进程停止
- 容器内进程端口监听故障
- SYN backlog 设置过小

国产化容器改造问题

- CPU架构不一样了，在X86下的容器镜像不能适配到国产化系统中
- 解决办法：换台国产化机器，其他的都一样。

应用该如何交付?

- 1、首选CICD系统，如jenkins、gitlab
- 2、其次手动构建镜像，如dockerfile->build->run->push
- 3、其他高级玩意，如kubevela

05

FAQ

https://shimo.im/docs/9PxccKdVk6JjjHVV/ 《kubernetes能力提升问题收集》

感谢您的聆听与观看

文档:https://kubernetes.io/zh/docs/concepts/