fzu_huang RP 147 发布于 kubernetes+docker集群管理 关注专栏
2018-03-19 发布

浅析k8s service的应用

原创 kubernetes 1.8k 次阅读 · 读完需要 9 分钟

前言

最近在产品新版本的服务发现和负载均衡方案上遇到了一个问题，在尽量不改动原生k8s使用方式和代码前提下，对service又重新复习了一遍，略有体会。

Service

k8s中service是一个面向微服务架构的设计，它从k8s本身解决了容器集群的负载均衡，并开放式地支持了用户所需要的各种负载均衡方案和使用场景。

通常，一个service被创建后会在集群中创建相应的endpoints，随后，controller-manager中的endpointsController会去检查并向该endpoints填入关于这个service，符合下述所有条件的后端端点（即pod）：

1. 相同的namespace；

2. pod的labels能满足service.Spec.selector（除非service.Spec.selector为空，这种情况下不会自动创建endpoints）；
3. 如果service开放了port，且是以字符串名字的形式（targetPort=[字符串]），则相应的pod的某个container中必须有配置同名的port；

当endpoints被更新后，kube-proxy会感知，并根据更新后的endpoints，在宿主机上做转发规则的配置，kube-proxy目前支持iptables、ipvs两种负载均衡方式，默认是iptables。

DNS

绝大部分使用k8s的人都会使用kubedns做服务发现和service domain的解析。kubedns会建立长连接即时检查service的变化，一旦发现有service被创建，会根据service的类型，在数据库中构建service domain 到指定的CNAME或IP（cluster-IP）的映射，作为对该service domain 的dns解析结果。

service 的类型

ClusterIP

最普遍的service类型，也是默认类型。ClusterIP类型的service创建时，k8s会通过etcd从可分配的IP池中分配一个IP，该IP全局唯一，且不可修改。所有访问该IP的请求，都会被iptables转发到后端的endpoints中。

该类型下，service的cluster-ip会作为kube-dns的解析结果，返回给客户端。基本上这是私有云中服务内部常用的方案，但是也只能集群内服务互相访问。

NodePort

通过node的IP进行地址转换实现服务的可访问性，外部访问集群中任意一个node：port即可以访问服务。
举个例子:一个单副本的deployment，其pod运行在node2上，通过nodePort方式开放服务，外部client访问node1_ip:port即可访问到容器服务。这个过程中原理是：

- client访问node1_ip:port
- node1对数据包做SNAT，将来源地址改成node1_ip
- node1对数据包做DNAT,将目的地址改成node2_ip
- node2收到数据包，根据port查找自身的端口，这个端口在service创建时就会与自身运行的port做映射，所以这里数据包会被转发到真实的容器中
- 数据响应由容器发给node1，node1再返回给client

这种方案下，node上会产生端口的占用，所以要确保端口可用性。
另外，通过指定service.spec.externalTrafficPolicy=Local可以设置node上kube-proxy不转发到其他node，参照上面的例子，由于node1没有响应的pod在运行，所以node1上会直接drop数据包。

使用这种方案的原因，不外乎是：**外部无法访问容器服务。**

LoadBalancer

需要外部支持（GCP and Azure），用户访问service.spec.external-ip,该IP对应到一个外部负载均衡的vip，外部服务对这个vip的请求，会被loadbalancer通过健康检查和转发，发送到一个运行着该服务pod的node上，并同样通过nodePort里的端口映射，发送给容器。

上述是几种比较普遍的service，随着社区发展，又出现了一些新的类型，可以做更灵活的适配。

ExternalName

用户可以指定一个任意的名字，作为该service被解析的CNAME,这种类型的servcie不用指定clusterIP，因此kube-proxy不会管理这类service，这类service需要使用1.7版本以上的kubedns。比如用户想创建一个service，但要让所有容器访问该service的请求都引导到用户自己定义的外部域名服务，就可以通过这个方式实现。可以说这个是最自由的一个方式：你希望服务被如何解析，服务就能被如何解析。你甚至可以给多个service配置同一个externalName。

headless service

headless service是一个特殊的ClusterIP类service，这种service创建时不指定clusterIP(--cluster-ip=None)，因为这点，kube-proxy不会管这种service，于是node上不会有相关的iptables规则。

当headless service有配置selector时，其对应的所有后端节点，会被记录到dns中，在访问service domain时kube-dns会将所有endpoints返回，选择哪个进行访问则是系统自己决定；

当selector设置为空时，headless service会去寻找相同namespace下与自己同名的pod作为endpoints。这一点被应用到statefulset中，当一个三副本的statefulset（mysql1，mysql2,mysql3）运行在不同节点时，我们可以通过域名的方式对他们分别访问。