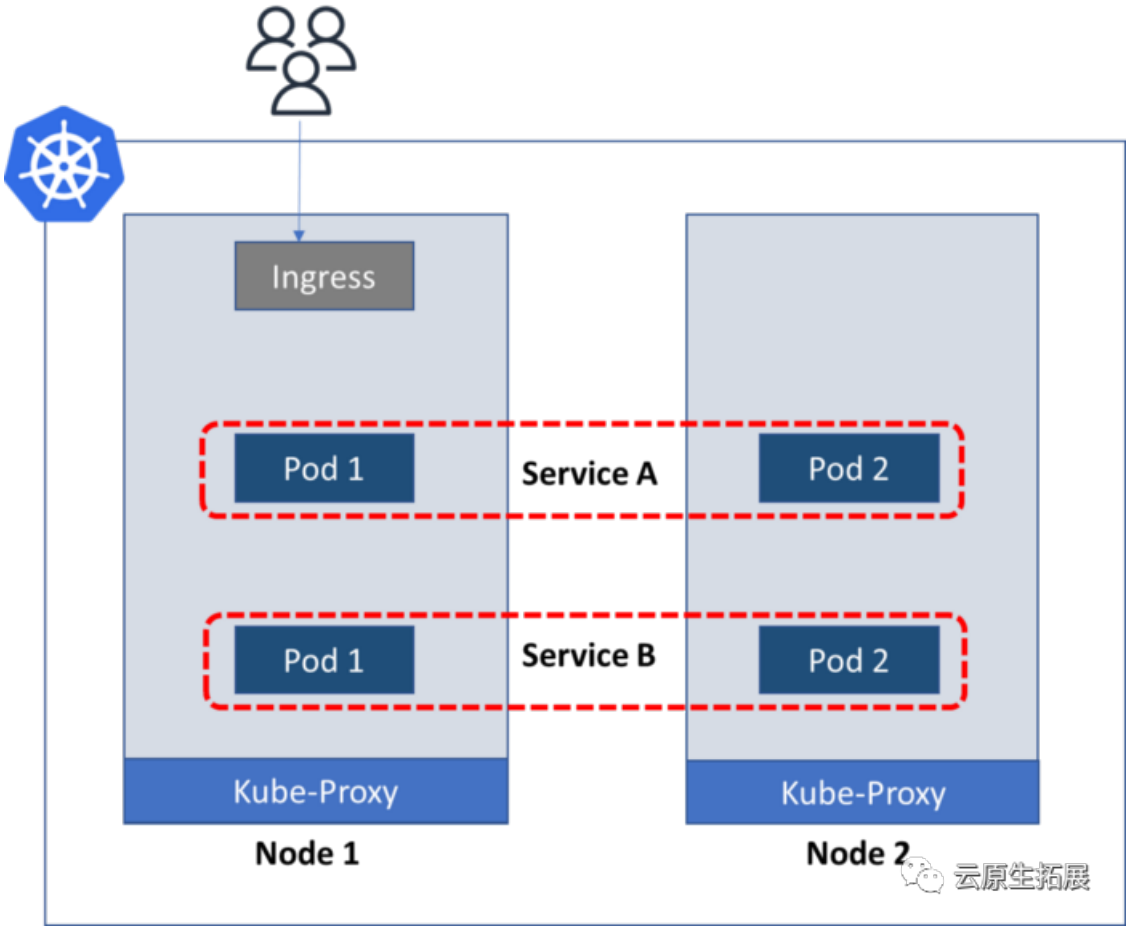
49Kubernetes 系列（四十六）Kubernetes 中东西向通信（Service to Service）

Kubernetes 系列（四十六）Kubernetes 中东西向通信（Service to Service）

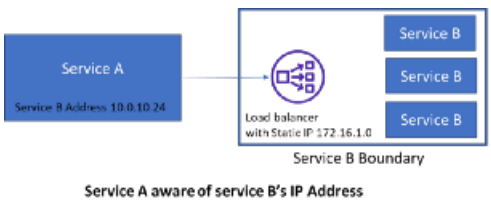
详细了解3个涉及 service to service 通信的原生k8s对象：ClusterIP Service 、DNS 和 Kube Proxy。

在我上一篇文章 “Kubernetes中的南北通信” 中，我写到了客户机如何在集群中访问服务。一旦进入集群，我们现在就可以看到后端服务如何在集群内相互通信。

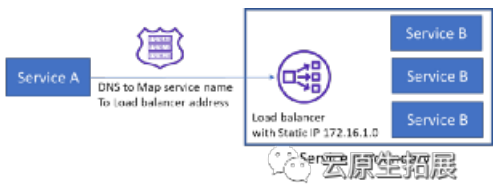


传统 service-to-service 通信

在进入Kubernetes生态系统之前，先快速了解一下传统的服务到服务通信——通信是通过IP地址进行的，因此为了让服务 A 调用服务 B，一种方法是为服务B分配一个静态IP地址。现在，要么服务A已经知道这个IP地址（这在处理非常少的服务时可能有效），要么服务B使用域名注册自己，服务A通过 DNS 查找获得服务 B 的联系地址。



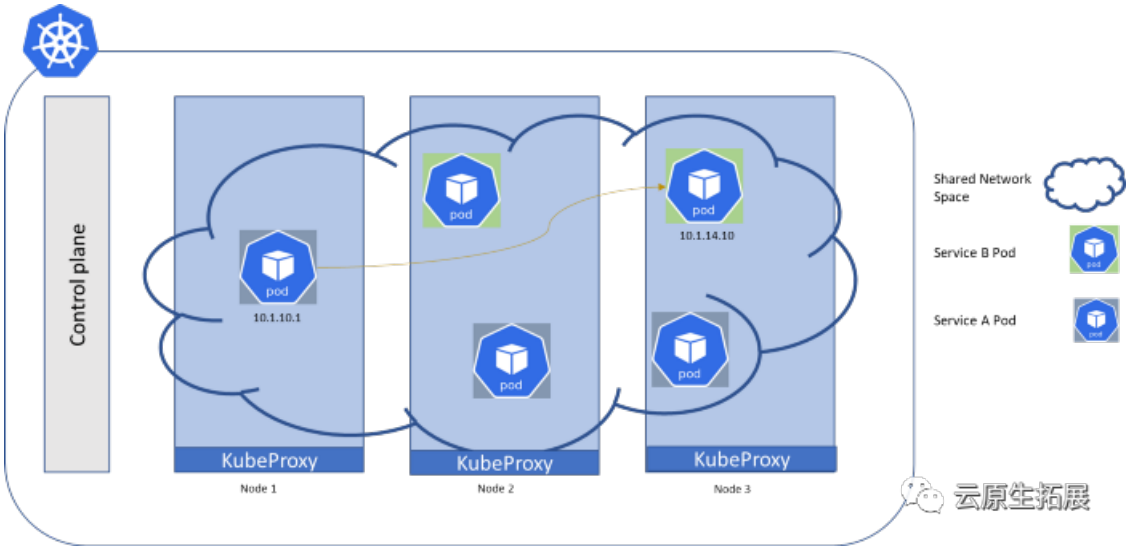
Service A aware of service B's IP Address



Service A fetches Service B's IP Address using DNS lookup

Kubernetes 网络模型

现在在Kubernetes集群中，我们有一个控制平面，它构成了集群管理组件和一组称为节点的工作机器。这些节点托管Pods，这些Pods将后端微服务作为容器化服务运行。集群内的Pod到Pod通信就是 k8s 的网络模型。

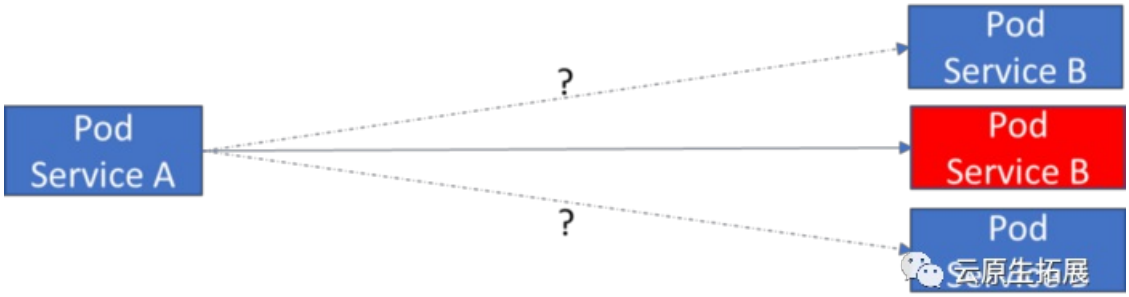


根据Kubernetes网络模型-

1. 集群中的每个 pod 都有自己独特的集群范围内的 IP 地址
2. 集群内所有 pod 可以互相通信
3. 通信在没有 NAT 的情况下进行，这意味着目标 pod 可以看到源 pod 的真实 IP 地址。Kubernetes 认为容器网络或其上运行的应用程序是可信的，不需要在网络级别进行身份验证，当然集群也支持类似的 NetworkPolicy 进行访问控制。

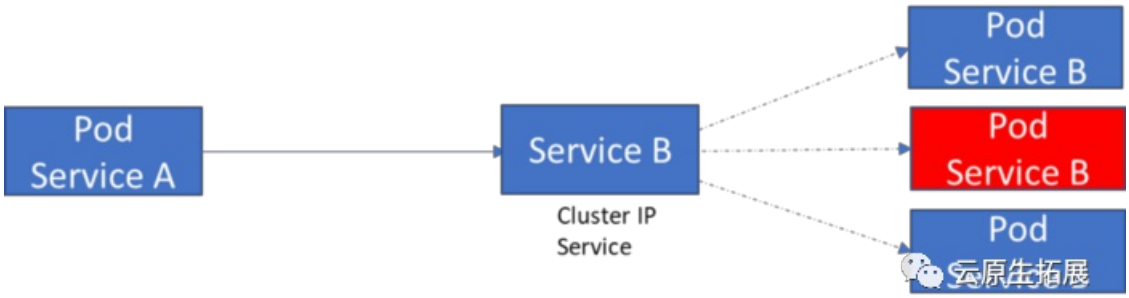
ClusterIP service — 对 Pod 的持久抽象

由于集群中的每个 pod 都有自己的IP地址，所以一个 pod 应该很容易与另一个pod通信？不，因为Pods 是不稳定的，每次创建一个Pod 都会得到一个新的 IP 地址。因此，客户端服务必须以某种方式切换到下一个可用的 pod，这是不可取的。



Pod 直接相互访问需要面临的问题是：pod IP 地址的短暂性以及如何发现其他关联的一组 pod

因此，Kubernetes 可以在一组 Pod 之上创建一个层，该层可以为该组提供单一 IP 地址，并提供基本的负载均衡。

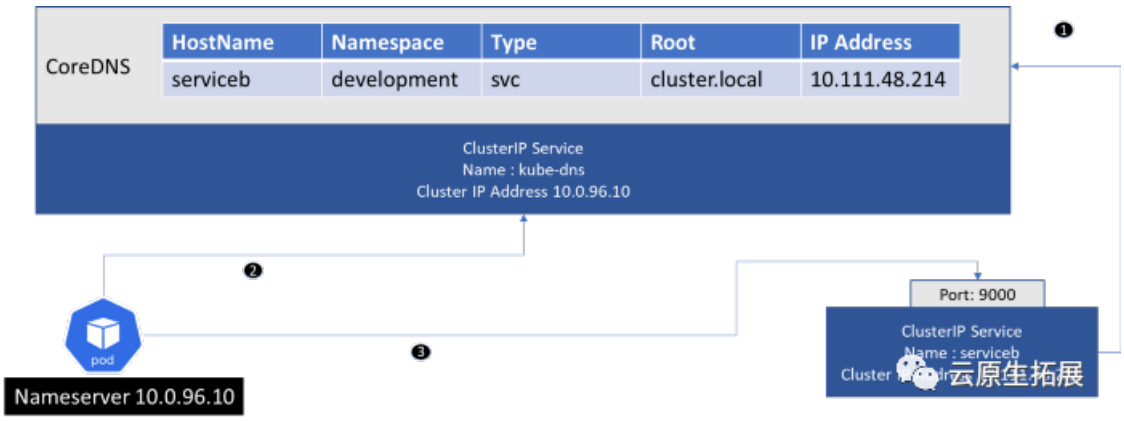


Pods 通过持久 IP 地址上的 Cluster IP 类型服务暴露客户端与服务对话，而不是直接与Pods 对话

这个抽象由 Kubernetes 中名为 **ClusterIP Service** 的服务对象提供。它在多个节点上产生，从而在集群内创建单个服务。它可以在任何端口上接收请求，并将其转发到pod上的任何端口。因此，当应用程序服务 A 需要与服务 B 通信时，它调用服务 B 对象的ClusterIP服务，而不是运行服务的单个pod。ClusterIP 使用 Kubernetes 中标签和选择器的标准模式来持续扫描符合选择标准的pod。Pod 有标签，服务用选择器来查找标签。使用这一点，就有可能实现一个基本的流量分割，即微服务的旧版本和新版本在同一个集群IP服务后面共存。

CoreDNS — 集群中的服务发现

虽然服务 B 已经获得了一个持久的IP地址，服务 A 仍然需要知道这个 IP 地址是什么，然后才能与服务B通信。Kubernetes 支持使用 CoreDNS 进行名称解析。服务 A 应该知道它需要与之通信的 ClusterIP 的名称（&port）。



1. CoreDNS 会扫描集群，每当创建 ClusterIP 服务时，它的条目就会添加到DNS服务器（如果配置了，它还会为每个pod 添加一个条目，但与服务到服务的通信无关）。

2. 接下来，CoreDNS 将自己暴露为一个ClusterIP 服务（默认情况下称为kube dns），该服务被配置为 pods 中的名称服务器。

3. 发起请求的 Pod从 DNS 获取 ClusterIP服务的 IP 地址，然后可以使用IP地址和端口发起请求。

Services 通过 <host name>.<name of namespace>.<type>.<root> 来识别。

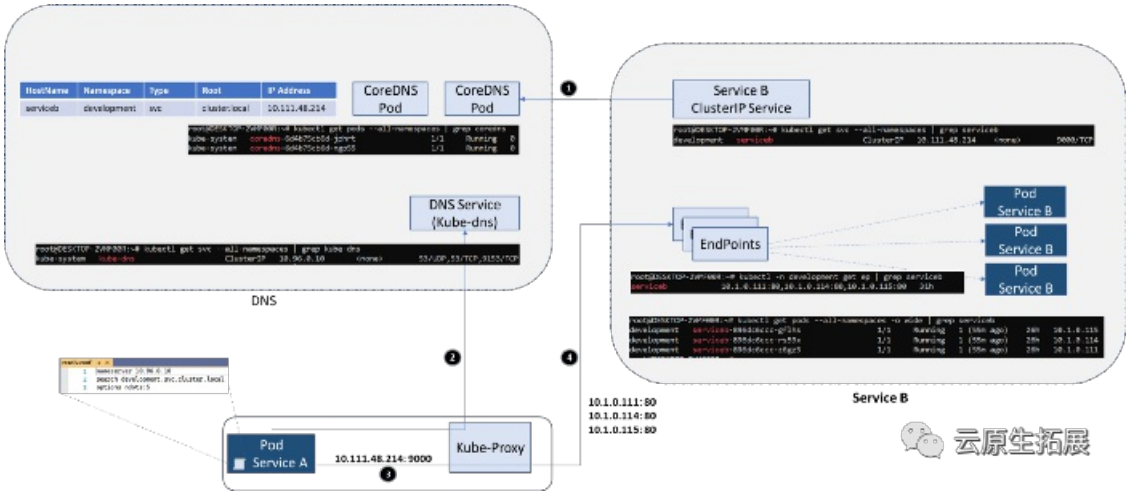
Kube-proxy — ClusterIP 服务和后台Pods之间的链接（目标网络地址转换）

到目前为止，从本文来看，似乎是ClusterIP服务将调用转发到后端Pods。但实际上，这是由Kube-proxy 完成的。Kube-proxy 在每个节点上运行，并监视服务及其后台Pod（实际上是端点对象）。

1. 当节点上运行的 pod 向ClusterIP服务发出请求时，kube-proxy 会拦截它。

2. 通过查看目标 IP 地址和端口，它可以识别目标ClusterIP服务。它将此请求的目的地替换为一个端点的地址，在该端点上存在用于服务请求的实际Pod。

它们是如何正在协同工作的?



ClusterIP 服务、CoreDNS、客户端Pod、Kube-proxy、端点和目标服务Pod的交互

1. 目标的 ClusterIP 服务已在 CoreDNS 中注册

2. DNS解析：每个pod 都有一个 resolve.conf 文件，其中包含 CoreDNS 服务的 IP 地址，pod执行DNS查找。

3. Pod使用从 DNS 接收的 IP 地址和它已经知道的端口来调用 ClusterIP 服务。

4. 目标地址转换：Kube-proxy 将目标 IP 地址更新为服务B的 Pod 可用的地址

总结

我们看到了本地 Kubernetes 对象，这些对象使服务到服务的通信成为可能。虽然这些细节对应用程序层是隐藏的，但最好知道在普通的Kubernetes中有什么可用的，以及在什么地方适合构建在Kubernetes之上的平台/产品。

欢迎关注我的公众号“ **云原生拓展** ”，原创技术文章第一时间推送。