71Kubernetes 系列（六十五）Kubernetes 中 DNS 查询的生命周期

在 Kubernetes 中，DNS 查询遵循特定路径来解析主机名的 IP 地址。在这篇博文中，您将逐步了解 Kubernetes 中 DNS 查询的生命周期。

Kubernetes 中的服务发现是整体架构的重要组成部分。它允许将传入请求路由到集群中运行的正确工作负载。 DNS 在此过程中起着关键作用。

了解 DNS 和服务发现在 Kubernetes 中的工作方式有助于调试问题。它使您可以更好地了解集群中的流量并诊断可能出现的任何问题。

当 pod 执行 DNS 查找时，查询首先发送到运行 pod 的节点上的 DNS 缓存。如果缓存不包含所请求主机名的 IP 地址，则将查询转发到集群 DNS 服务器。该服务器处理 Kubernetes 中的服务发现。

集群 DNS 服务器通过查询 Kubernetes 服务注册表来确定 IP 地址。此注册表包含服务名称与其对应 IP 地址的映射。这允许集群 DNS 服务器将正确的 IP 地址返回给请求的 pod。

Services

如前所述，DNS 是服务发现的必要组成部分。服务发现在 Kubernetes 中的工作方式是通过使用 **Service** 资源。Service 有一个 IP，当访问该 IP 时，会将连接重定向到支持该 Service 的健康 pod。

在 Kubernetes 中创建 Service 时，集群 DNS 服务器会为该服务创建一条 A 记录。此记录将服务的 DNS 名称映射到其 IP 地址。这允许 pod 使用其 DNS 名称访问该服务。只要服务的 IP 地址发生变化，DNS 服务器也会更新 A 记录。这可确保 DNS 名称始终指向正确的 IP 地址。

示例服务定义如下所示：

```
apiVersion: v1
kind: Service
metadata:
  name: foo
  namespace: bar
spec:
  ports:
    - port: 80
      name: http
```

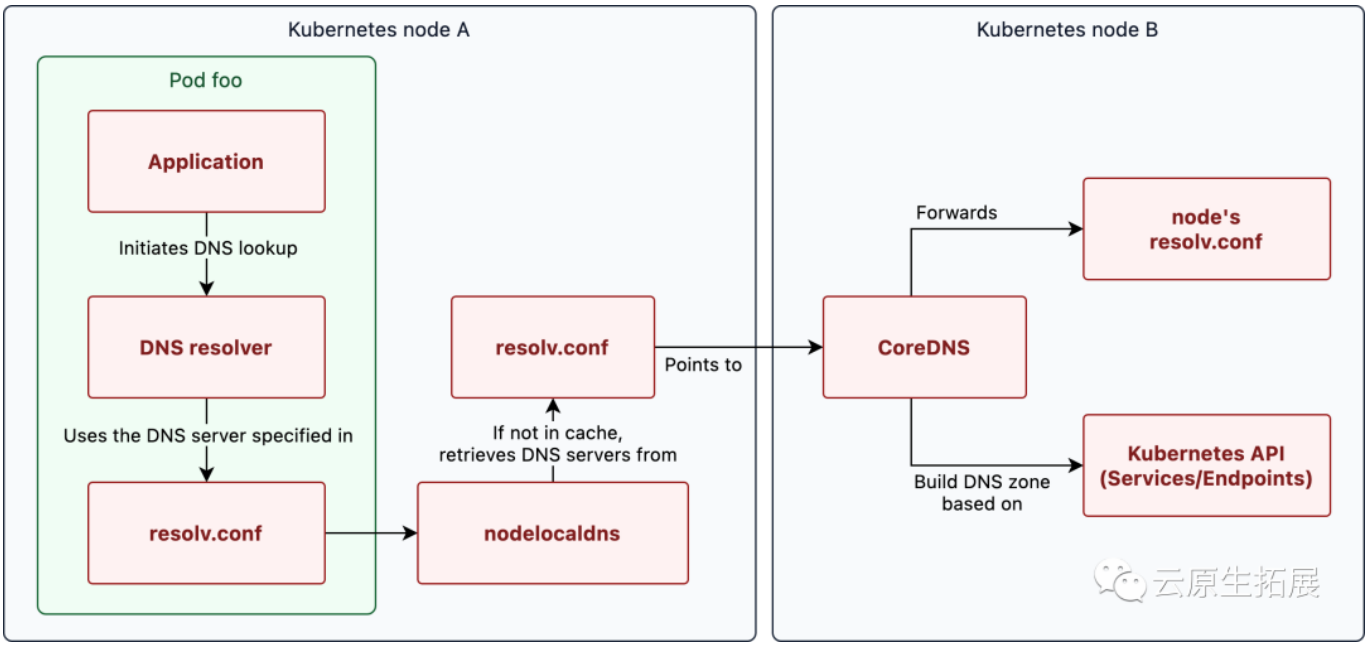
在此实例中创建的 A 记录和 SRV 记录（在本文后面讨论）如下所示：

```
foo.bar.svc.cluster.local                 30   A   10.129.1.26
_http._tcp.nginx.default.svc.cluster.local 3600 SRV 0 100 80 10-129-1-26.foo.bar.svc.cluster.local.
```

要为此服务创建完全限定域名，我们使用服务名称 (foo)、命名空间 (bar) 和集群域 (cluster.local)。

集群中运行的任何工作负载现在都可以使用此 DNS 名称解析服务的 IP 地址。

Services 的 DNS 查找



当 pod 执行 DNS 查找时，查询首先发送到 pod 中的本地 DNS 解析器。此解析器使用 resolv.conf 配置文件。在此文件中，nodelocaldns 服务器设置为默认的递归 DNS 解析器，充当缓存。

如果此缓存不包含所请求主机名的 IP 地址，则查询将转发到集群 DNS 服务器 (CoreDNS)。

此 DNS 服务器通过咨询 Kubernetes 服务注册表来确定 IP 地址。此注册表包含服务名称与其对应 IP 地址的映射。这允许集群 DNS 服务器将正确的 IP 地址返回给请求的 pod。

任何被查询但不在 Kubernetes 服务注册表中的域都将转发到上游 DNS 服务器。

我们将逐步更详细地介绍这些组件中的每一个。

Pod foo

当 Pod 向同一个 Kubernetes 集群内的服务发送 API 请求时，它必须首先解析该服务的 IP 地址。为此，pod 使用在其 `/etc/resolv.conf` 配置文件中指定的 DNS 服务器执行 DNS 查找。

该文件由 Kubelet 提供，定义了 pod 中 DNS 查找的设置。它包含对集群 DNS 服务器的引用。

默认情况下，此配置文件如下所示：

```
search namespace.svc.cluster.local svc.cluster.local cluster.local
nameserver 10.123.0.10
options ndots:5
```

Pod config

默认情况下，Kubelet 提供的 `/etc/resolv.conf` 文件会将所有 DNS 查询转发到集群的 DNS 服务器（上例中的 10.123.0.10）。 Kubelet 还定义了搜索域和 DNS 查询的 ndots 选项。

搜索域指定在给出不完整域（非 FQDN）时应搜索哪些域后缀。 ndots 选项确定何时直接查询绝对域而不是首先附加搜索域。

为了更好地理解这是如何工作的，让我们看一个例子。假设名为 foo 的 pod 执行 bar.other-ns 的 DNS 查找。如果 ndots 选项设置为 5（默认值），解析器将计算域中的点数。

如果少于 5 个点，将在 DNS 服务器上执行 DNS 查找之前附加搜索域。如果有 5 个或更多点，则将按原样查询域而不附加搜索域。在此示例中， bar.other-ns 少于 5 个点，因此将在执行 DNS 查找之前附加搜索域。

默认情况下，搜索域是：

- <请求者命名空间>.svc.cluster.local
- svc.cluster.local
- cluster.local

在找到有效响应之前，这些搜索域将附加到域中并进行查询。解析器将一一尝试以下查询：

- bar.other-ns.<请求者命名空间>.svc.cluster.local
- bar.other-ns.svc.cluster.local（⇐ 找到匹配项！）

bar Service 将在 bar.other-ns.svc.cluster.local 上侦听，因此找到匹配项并返回正确的 A 记录。

要更改 pod 的 DNS 解析器的行为，您可以更改 pod 的 DNS 配置：

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
    - name: test
      image: nginx
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
      - 1.2.3.4
    searches:
      - ns1.svc.cluster-domain.example
      - my.dns.search.suffix
    options:
      - name: ndots
        value: "2"
      - name: edns0
```

在上面的示例中， `dnsPolicy` 设置为“None”，这意味着 pod 将不会使用集群提供的默认 DNS 设置。相反， `dnsConfig` 字段用于指定 pod 的自定义 DNS 设置。

`nameservers` 字段用于指定 pod 应该用于 DNS 查找的 DNS 服务器。搜索字段用于指定应该用于不完整域的搜索域。

`options` 字段用于指定 DNS 解析器的自定义选项，例如上例中的 `ndots` 和 `edns0` 选项。

这些设置将由 pod 的 DNS 解析器使用，而不是集群提供的默认设置。有关 pod DNS 配置的更多信息，请参阅官方文档 (https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/#pod-dns-config)。

权威 DNS 服务器

在 1.13 版本之前的 Kubernetes 集群中，kube-dns 充当 Kubernetes 的权威 DNS 服务器。在 Kubernetes 1.13 版本中，CoreDNS 取代了 kube-dns 作为权威 DNS 查询的默认组件。

DNS 服务器将所有服务添加到其权威的 DNS 区域，以便它可以将域名解析为 Kubernetes 服务的 IP 地址。 Kubernetes 中的权威 DNS 服务器存在各种软件实现。

CoreDNS 是一个流行的选择，因为它支持从 Kubernetes 服务注册表构建 DNS 区域。它还提供额外的功能，例如缓存、转发和日志记录。

一个CoreDNS配置文件的例子：

```
.:53 {
    errors
    health {
        lameduck 5s
    }
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        fallthrough in-addr.arpa ip6.arpa
        ttl 30
    }
    forward . /etc/resolv.conf
    cache 30
}
```

需要特别注意的是 kubernetes 区域和 forward 语句。

有关更改 kube-dns 配置的更多信息，请参阅此文档页面(https://kubernetes.io/docs/tasks/administer-cluster/dns-custom-nameservers/)。

Nodelocaldns

DNS 查询是网络通信中常见且必不可少的部分。它们需要快速处理以避免性能问题。缓慢的 DNS 查询会导致难以诊断和排除故障的问题。

为了提高 Kubernetes 集群中 DNS 查询的性能，可以使用 nodelocaldns 组件在每个节点上添加一个缓存层。该组件缓存对 DNS 查询的响应。

如果在缓存中没有找到响应，它将查询转发到权威名称服务器 (CoreDNS)。响应存储在本地缓存中，以便它可用于为来自同一节点上的相同或其他 pod 的未来查询提供服务。

这减少了 pod 和 DNS 服务器之间的网络流量。这意味着更低的延迟和更快的 DNS 查询性能。 nodelocaldns 的功能通常也由 CoreDNS 完成。

关于 Kubernetes 中记录的 TTL（生存时间）的说明

在 Kubernetes 中，DNS 记录的生存时间 (TTL) 由正在使用的 DNS 服务器实现决定。

默认情况下，CoreDNS 将 DNS 记录的 TTL 设置为 30 秒。这意味着当 DNS 查询得到解析时，响应将被缓存最多 30 秒，然后才会被视为过时。可以使用 CoreDNS 配置文件中的 `ttl` 选项修改 DNS 记录的 TTL。

DNS 记录的 TTL 是一个重要参数，因为它决定了在必须进行新查询之前 DNS 响应将被视为有效的时间长度。

较短的 TTL 可以提高 DNS 响应的准确性，但也会增加 DNS 服务器的负载。较长的 TTL 可以减少 DNS 服务器的负载。但是，如果基础 DNS 记录已更新，它也可能导致 DNS 响应过时或不准确。

因此，应根据集群的具体要求选择合适的TTL。

SRV 记录

到目前为止，我们只讨论了使用 A 记录解析 IP 地址。 Kubernetes 还使用 SRV（服务）记录来解析命名服务的端口号。这允许客户端通过向 DNS 服务器查询适当的 SRV 记录来发现服务的端口号。

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
spec:
  ports:
    - port: 80
      name: http
```

在此服务中，容器端口 80 被公开并被命名为“http”。由于端口已命名，因此 Kubernetes 将生成具有以下名称的 SRV 记录： `_<port>._<proto>.<service>.<ns>.<svc>.<zone>` 。

在本例中，SRV 记录将被命名为 _http._tcp.nginx.default.svc.cluster.local 。对此记录的 DNS 查询将返回指定服务的端口号和 IP 地址：

```
dig +short SRV _http._tcp.nginx.default.svc.cluster.local
0 100 80 10-129-1-26.nginx.default.svc.cluster.local.
```

某些服务（例如 Kerberos）使用 SRV 记录来发现 KDC（密钥分发中心）服务器。