安装配置 DNS

DNS 组件作为 Kubernetes 中服务注册和发现的一个必要组件，起着举足轻重的作用，是我们在安装好 Kubernetes 集群后部署的第一个容器化应用。

安装配置kube-dns

在我们安装Kubernetes集群的时候就已经安装了kube-dns插件，这个插件也是官方推荐安装的。通过将 Service 注册到 DNS 中，Kuberentes 可以为我们提供一种简单的服务注册发现与负载均衡方式。
CoreDNS作为CNCF中的托管的一个项目，在Kuberentes1.9版本中，使用kubeadm方式安装的集群可以通过以下命令直接安装CoreDNS。

```
kubeadm init --feature-gates=CoreDNS=true
```

您也可以使用CoreDNS替换Kubernetes插件kube-dns，可以使用 Pod 部署也可以独立部署，请参考Using CoreDNS for Service Discovery，下文将介绍如何配置kube-dns。

kube-dns

kube-dns是Kubernetes中的一个内置插件，目前作为一个独立的开源项目维护，见 https://github.com/kubernetes/dns。
下文中给出了配置 DNS Pod 的提示和定义 DNS 解析过程以及诊断 DNS 问题的指南。

前提要求

- Kubernetes 1.6 及以上版本。
- 集群必须使用 `kube-dns` 插件进行配置。

系统预定义的 RoleBinding

预定义的 RoleBinding system:kube-dns 将 kube-system 命名空间的 kube-dns ServiceAccount 与 system:kube-dns Role 绑定， 该 Role 具有访问

kube-apiserver DNS 相关 API 的权限；

```
[root@vlnx251101 ~]# kubectl get clusterrolebindings system:kube-dns -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: 2018-08-10T09:07:09Z
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:kube-dns
  resourceVersion: "97"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/system%3Akube-dns
  uid: c2dd7cfd-9c7c-11e8-af44-000c29526d85
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:kube-dns
subjects:
- kind: ServiceAccount
  name: kube-dns
  namespace: kube-system
```

kubedns-controller.yaml 中定义的 Pods 时使用了 kubedns-sa.yaml 文件定义的 kube-dnsServiceAccount，所以具有访问 kube-apiserver DNS 相关 API 的权限。

配置 kube-dns

```
[root@vlnx251101 kubedns]# kubectl create -f  kube-dns.yaml
```

检查 kubedns 功能

新建一个 Deployment

```
[root@vlnx251101 kubedns]# vim my-nginx.yml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: my-nginx
spec:
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
[root@vlnx251101 kubedns]# kubectl create -f my-nginx.yml
```

Export 该 Deployment, 生成 my-nginx 服务

```
[root@vlnx251101 kubedns]# kubectl expose deploy my-nginx
```

```
[root@vlnx251101 ~]# kubectl get services --all-namespaces | grep
my-nginx
default       my-nginx              ClusterIP   10.254.152.223
<none>        80/TCP          15s
```

创建另一个 Pod，查看 /etc/resolv.conf 是否包含 kubelet 配置的 --cluster-dns 和 --cluster-domain，是否能够将服务 my-nginx 解析到 Cluster IP 10.254.152.223。

```
[root@vlnx251101 kubedns]# vim nginx-pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.9
    ports:
    - containerPort: 80

[root@vlnx251101 kubedns]# kubectl create -f nginx-pod.yml

[root@vlnx251101 kubedns]# kubectl exec  nginx -i -t -- /bin/bash
root@nginx:/#
root@nginx:/#
root@nginx:/# cat /etc/resolv.conf
nameserver 10.254.0.2
search default.svc.cluster.local. svc.cluster.local. cluster.local.
zyg.com
options ndots:5
root@nginx:/#
root@nginx:/# ping my-nginx
PING my-nginx.default.svc.cluster.local (10.254.152.223): 56 data
bytes
^C--- my-nginx.default.svc.cluster.local ping statistics ---
9 packets transmitted, 0 packets received, 100% packet loss
root@nginx:/#
root@nginx:/#
root@nginx:/#
root@nginx:/# ping kubernetes
PING kubernetes.default.svc.cluster.local (10.254.0.1): 56 data
bytes
^C--- kubernetes.default.svc.cluster.local ping statistics ---
```

2 packets transmitted, 0 packets received, 100% packet loss

root@nginx:/#

root@nginx:/#

root@nginx:/# ping kube-dns.kube-system.svc.cluster.local

PING kube-dns.kube-system.svc.cluster.local (10.254.0.2): 56 data bytes

^C--- kube-dns.kube-system.svc.cluster.local ping statistics ---

4 packets transmitted, 0 packets received, 100% packet loss

从结果来看，service名称可以正常解析。

注意：直接ping ClusterIP是ping不通的，ClusterIP是根据**IPtables**路由到服务的endpoint上，只有结合ClusterIP加端口才能访问到对应的服务。

kube-dns 介绍

从 Kubernetes v1.3 版本开始，使用 cluster add-on 插件管理器回自动启动内置的DNS。

Kubernetes DNS pod 中包括 3 个容器：

- kubedns：kubedns 进程监视 Kubernetes master 中的 Service 和 Endpoint 的变化，并维护内存查找结构来服务DNS请求。
- dnsmasq：dnsmasq 容器添加 DNS 缓存以提高性能。
- sidecar：sidecar 容器在执行双重健康检查（针对 dnsmasq 和 kubedns）时提供单个健康检查端点（监听在10054端口）。

DNS pod 具有静态 IP 并作为 Kubernetes 服务暴露出来。该静态 IP 分配后，kubelet 会将使用 --cluster-dns = <dns-service-ip> 标志配置的 DNS 传递给每个容器。

DNS 名称也需要域名。本地域可以使用标志 --cluster-domain = <default-local-domain> 在 kubelet 中配置。

Kubernetes集群DNS服务器基于 SkyDNS 库。它支持正向查找（A 记录），服务查找（SRV 记录）和反向 IP 地址查找（PTR 记录）

kube-dns 支持的 DNS 格式

kube-dns 将分别为 service 和 pod 生成不同格式的 DNS 记录。

Service

- A记录：生成my-svc.my-namespace.svc.cluster.local域名，解析成 IP 地址，分为两种情况：
  - 普通 Service：解析成 ClusterIP
  - Headless Service：解析为指定 Pod 的 IP 列表
- SRV记录：为命名的端口（普通 Service 或 Headless Service）生成 _my-port-name._my-port-protocol.my-svc.my-namespace.svc.cluster.local 的域名

Pod

- A记录：生成域名 pod-ip.my-namespace.pod.cluster.local

kube-dns 存根域名

可以在 Pod 中指定 hostname 和 subdomain：hostname.custom-subdomain.default.svc.cluster.local，例如：

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  labels:
    name: busybox
spec:
  hostname: busybox-1
  subdomain: busybox-subdomain
  containers:
  name: busybox
  - image: busybox
    command:
    - sleep
    - "3600"
```

该 Pod 的域名是 busybox-1.busybox-subdomain.default.svc.cluster.local。

继承节点的 DNS

运行 Pod 时，kubelet 将预先配置集群 DNS 服务器到 Pod 中，并搜索节点自己的 DNS 设置路径。如果节点能够解析特定于较大环境的 DNS 名称，那么 Pod 应该也能够解析。请参阅下面的已知问题以了解警告。
如果您不想要这个，或者您想要为 Pod 设置不同的 DNS 配置，您可以给 kubelet 指定 `--resolv-conf`标志。将该值设置为 "" 意味着 Pod 不继承 DNS。将其设置为有效的文件路径意味着 kubelet 将使用此文件而不是 `/etc/resolv.conf` 用于 DNS 继承。

配置存根域和上游 DNS 服务器

通过为 kube-dns （`kube-system:kube-dns`）提供一个 ConfigMap，集群管理员能够指定自定义存根域和上游 nameserver。
例如，下面的 ConfigMap 建立了一个 DNS 配置，它具有一个单独的存根域和两个上游 nameserver：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-dns
  namespace: kube-system
data:
  stubDomains: |
    {“acme.local”: [“1.2.3.4”]}
  upstreamNameservers: |
    [“8.8.8.8”, “8.8.4.4”]
```

如上面指定的那样，带有“.acme.local”后缀的 DNS 请求被转发到 1.2.3.4 处监听的 DNS。Google Public DNS 为上游查询提供服务。
下表描述了如何将具有特定域名的查询映射到其目标DNS服务器：

域名	响应查询的服务器
kubernetes.default.svc.cluster.local	kube-dns
foo.acme.local	自定义 DNS (1.2.3.4)

widget.com	上游 DNS (8.8.8.8 或 8.8.4.4)

查看 ConfigMap 选项 获取更多关于配置选项格式的详细信息。

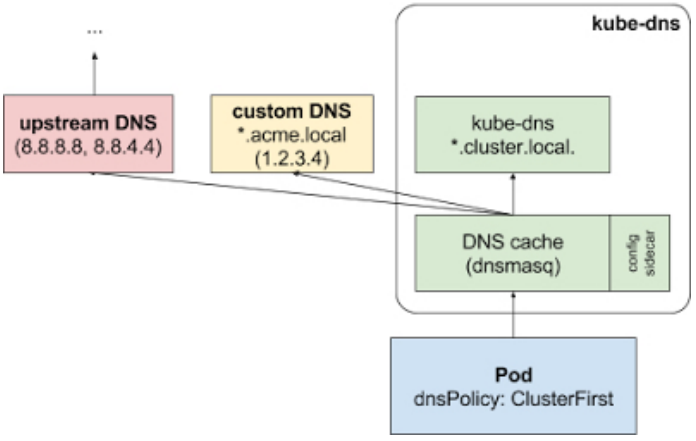
对 Pod 的影响

自定义的上游名称服务器和存根域不会影响那些将自己的 `dnsPolicy` 设置为 `Default` 或者 `None` 的 Pod。
如果 Pod 的 `dnsPolicy` 设置为 "`ClusterFirst`"，则其名称解析将按其他方式处理，具体取决于存根域和上游 DNS 服务器的配置。

未进行自定义配置：没有匹配上配置的集群域名后缀的任何请求，例如 "www.kubernetes.io"，将会被转发到继承自节点的上游 nameserver。

进行自定义配置：如果配置了存根域和上游 DNS 服务器（和在 前面例子 配置的一样），DNS 查询将根据下面的流程进行路由：

1. 查询首先被发送到 kube-dns 中的 DNS 缓存层。
2. 从缓存层，检查请求的后缀，并转发到合适的 DNS 上，基于如下的示例：
    - *具有集群后缀的名字*（例如 "`.cluster.local`"）：请求被发送到 kube-dns。
    - *具有存根域后缀的名字*（例如 "`.acme.local`"）：请求被发送到配置的自定义 DNS 解析器（例如：监听在 `1.2.3.4`）。
    - *不具有能匹配上后缀的名字*（例如 "widget.com"）：请求被转发到上游 DNS（例如：Google 公共 DNS 服务器，`8.8.8.8` 和 `8.8.4.4`）。



图片 - DNS lookup flow

ConfigMap 选项

kube-dns `kube-system:kube-dns` ConfigMap 的选项如下所示：

字段	格式	描述
`stubDomains`（可选）	使用 DNS 后缀 key 的 JSON map（例如 “acme.local”），以及 DNS IP 的 JSON 数组作为 value。	目标 nameserver 可能是一个如，可以运行自己的 dnsmasq ClusterDNS namespace 中。
`upstreamNameservers`（可选）	DNS IP 的 JSON 数组。	注意：如果指定，则指定的值会替的 /etc/resolv.conf中获取到的 n以指定三个上游 nameserver。

示例

示例：存根域

在这个例子中，用户有一个 Consul DNS 服务发现系统，他们希望能够与 kube-dns 集成起来。 Consul 域名服务器地址为 10.150.0.1，所有的 Consul 名字具有后缀 .consul.local。 要配置 Kubernetes，集群管理员只需要简单地创建一个 ConfigMap 对象，如下所示：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-dns
  namespace: kube-system
data:
  stubDomains: |
    {“consul.local”: [“10.150.0.1”]}
```

注意，集群管理员不希望覆盖节点的上游 nameserver，所以他们不会指定可选的 `upstreamNameservers` 字段。

示例：上游 nameserver

在这个示例中，集群管理员不希望显式地强制所有非集群 DNS 查询进入到他们自己的 nameserver 172.16.0.1。 而且这很容易实现：他们只需要创建一个 ConfigMap，`upstreamNameservers` 字段指定期望的 nameserver 即可。

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: kube-dns
  namespace: kube-system
data:
  upstreamNameservers: |
    ["172.16.0.1"]
```

调试 DNS 解析

创建一个简单的 Pod 用作测试环境

创建一个名为 busybox.yaml 的文件，其中包括以下内容：

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
  - name: busybox
    image: busybox
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
```

使用该文件创建 Pod 并验证其状态：

```
$ kubectl create -f busybox.yaml
$ kubectl get pods busybox
```

该 Pod 运行后，您可以在它的环境中执行 `nslookup`。如果您看到类似如下的输出，表示 DNS 正在正确工作。

```bash
$ kubectl exec -ti busybox -- nslookup kubernetes.default
Server:    10.0.0.10
```

```
Address 1: 10.0.0.10

Name:      kubernetes.default
Address 1: 10.0.0.1
```

如果 `nslookup` 命令失败，检查如下内容：

首先检查本地 DNS 配置

查看下 `resolv.conf` 文件。（参考集成节点的 DNS和 下面的已知问题获取更多信息）

```
$ kubectl exec busybox cat /etc/resolv.conf
```

验证搜索路径和名称服务器设置如下（请注意，搜索路径可能因不同的云提供商而异）：

```
search default.svc.cluster.local svc.cluster.local cluster.local
google.internal c.gce_project_id.internal
nameserver 10.0.0.10
options ndots:5
```

如果看到如下错误表明错误来自 `kube-dns` 或相关服务：

```
$ kubectl exec -ti busybox -- nslookup kubernetes.default
Server:    10.0.0.10
Address 1: 10.0.0.10

nslookup: can't resolve 'kubernetes.default'
```

或者

```
$ kubectl exec -ti busybox -- nslookup kubernetes.default
Server:    10.0.0.10
Address 1: 10.0.0.10 kube-dns.kube-system.svc.cluster.local

nslookup: can't resolve 'kubernetes.default'
```

检查 DNS pod 是否在运行

使用 `kubectl get pods` 命令验证 DNS pod 是否正在运行。

```
$ kubectl get pods --namespace=kube-system -l k8s-app=kube-dns
NAME                        READY     STATUS    RESTARTS   AGE
...
kube-dns-v19-ezo1y          3/3       Running   0           1h
...
```

如果您看到没有 Pod 运行或者 Pod 处于 失败/完成 状态，DNS 插件可能没有部署到您的当前环境中，您需要手动部署。

检查 DNS pod 中的错误

使用 `kubectl logs` 命令查看 DNS 守护进程的日志。

```
$ kubectl logs --namespace=kube-system $(kubectl get pods --
namespace=kube-system -l k8s-app=kube-dns -o name) -c kubedns
$ kubectl logs --namespace=kube-system $(kubectl get pods --
namespace=kube-system -l k8s-app=kube-dns -o name) -c dnsmasq
$ kubectl logs --namespace=kube-system $(kubectl get pods --
namespace=kube-system -l k8s-app=kube-dns -o name) -c sidecar
```

看看有没有可疑的日志。以字母“`W`”，“`E`”，“`F`”开头的代表警告、错误和失败。请搜索具有这些日志级别的条目，并使用 kubernetes issues来报告意外错误。

DNS 服务启动了吗？

使用 `kubectl get service` 命令验证 DNS 服务是否启动。

```
$ kubectl get svc --namespace=kube-system
NAME          CLUSTER-IP      EXTERNAL-IP    PORT(S)                  AGE
...
kube-dns      10.0.0.10       <none>         53/UDP,53/TCP            1h
```

...

如果您已经创建了该服务或它本应该默认创建但没有出现，参考调试服务获取更多信息。

DNS 端点暴露出来了吗？

您可以使用`kubectl get endpoints`命令验证 DNS 端点是否被暴露。

```
$ kubectl get ep kube-dns --namespace=kube-system
NAME        ENDPOINTS                       AGE
kube-dns    10.180.3.17:53,10.180.3.17:53    1h
```

如果您没有看到端点，查看调试服务文档中的端点部分。