Customizing DNS Service

This page explains how to configure your DNS Pod(s) and customize the DNS resolution process in your cluster.

Before you begin

You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. It is recommended to run this tutorial on a cluster with at least two nodes that are not acting as control plane hosts. If you do not already have a cluster, you can create one by using minikube or you can use one of these Kubernetes playgrounds:

- Killercoda
- Play with Kubernetes

Your cluster must be running the CoreDNS add-on.

Your Kubernetes server must be at or later than version v1.12. To check the version, enter `kubectl version`.

Introduction

DNS is a built-in Kubernetes service launched automatically using the *addon manager* cluster add-on.

> **Note:** The CoreDNS Service is named `kube-dns` in the `metadata.name` field.
> The intent is to ensure greater interoperability with workloads that relied on the legacy `kube-dns` Service name to resolve addresses internal to the cluster. Using a Service named `kube-dns` abstracts away the implementation detail of which DNS provider is running behind that common name.

If you are running CoreDNS as a Deployment, it will typically be exposed as a Kubernetes Service with a static IP address. The kubelet passes DNS resolver information to each container with the `--cluster-dns=<dns-service-ip>` flag.

DNS names also need domains. You configure the local domain in the kubelet with the flag `--cluster-domain=<default-local-domain>`.

The DNS server supports forward lookups (A and AAAA records), port lookups (SRV records), reverse IP address lookups (PTR records), and more. For more information, see DNS for Services and Pods.

If a Pod's `dnsPolicy` is set to `default`, it inherits the name resolution configuration from the node that the Pod runs on. The Pod's DNS resolution should behave the same as the node. But see Known issues.

If you don't want this, or if you want a different DNS config for pods, you can use the kubelet's `--resolv-conf` flag. Set this flag to "" to prevent Pods from inheriting DNS. Set it to a valid file path to specify a file other than `/etc/resolv.conf` for DNS inheritance.

CoreDNS

CoreDNS is a general-purpose authoritative DNS server that can serve as cluster DNS, complying with the DNS specifications.

CoreDNS ConfigMap options

CoreDNS is a DNS server that is modular and pluggable, with plugins adding new functionalities. The CoreDNS server can be configured by maintaining a Corefile, which is the CoreDNS configuration file. As a cluster administrator, you can modify the ConfigMap for the CoreDNS Corefile to change how DNS service discovery behaves for that cluster.

In Kubernetes, CoreDNS is installed with the following default Corefile configuration:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns
  namespace: kube-system
data:
  Corefile: |
    .:53 {
        errors
        health {
            lameduck 5s
        }
        ready
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
            ttl 30
        }
        prometheus :9153
        forward . /etc/resolv.conf
        cache 30
        loop
        reload
        loadbalance
    }
```

The Corefile configuration includes the following plugins of CoreDNS:

- errors: Errors are logged to stdout.
- health: Health of CoreDNS is reported to `http://localhost:8080/health`. In this extended syntax `lameduck` will make the process unhealthy then wait for 5 seconds before the process is shut down.
- ready: An HTTP endpoint on port 8181 will return 200 OK, when all plugins that are able to signal readiness have done so.
- kubernetes: CoreDNS will reply to DNS queries based on IP of the Services and Pods. You can find more details about this plugin on the CoreDNS website.
  - `ttl` allows you to set a custom TTL for responses. The default is 5 seconds. The minimum TTL allowed is 0 seconds, and the maximum is capped at 3600 seconds. Setting TTL to 0 will prevent records from being cached.
  - The `pods insecure` option is provided for backward compatibility with `kube-dns`.
  - You can use the `pods verified` option, which returns an A record only if there exists a pod in the same namespace with a matching IP.
  - The `pods disabled` option can be used if you don't use pod records.
- prometheus: Metrics of CoreDNS are available at `http://localhost:9153/metrics` in the Prometheus format (also known as OpenMetrics).
- forward: Any queries that are not within the Kubernetes cluster domain are forwarded to predefined resolvers (/etc/resolv.conf).
- cache: This enables a frontend cache.
- loop: Detects simple forwarding loops and halts the CoreDNS process if a loop is found.
- reload: Allows automatic reload of a changed Corefile. After you edit the ConfigMap configuration, allow two minutes for your changes to take effect.
- loadbalance: This is a round-robin DNS loadbalancer that randomizes the order of A, AAAA, and MX records in the answer.

You can modify the default CoreDNS behavior by modifying the ConfigMap.

Configuration of Stub-domain and upstream nameserver using CoreDNS

CoreDNS has the ability to configure stub-domains and upstream nameservers using the forward plugin.

Example

If a cluster operator has a Consul domain server located at "10.150.0.1", and all Consul names have the suffix ".consul.local". To configure it in CoreDNS, the cluster administrator creates the following stanza in the CoreDNS ConfigMap.

```
consul.local:53 {
    errors
    cache 30
    forward . 10.150.0.1
}
```

To explicitly force all non-cluster DNS lookups to go through a specific nameserver at 172.16.0.1, point the `forward` to the nameserver instead of `/etc/resolv.conf`

```
forward .  172.16.0.1
```

The final ConfigMap along with the default `Corefile` configuration looks like:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns
  namespace: kube-system
data:
  Corefile: |
    .:53 {
        errors
        health
        kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           fallthrough in-addr.arpa ip6.arpa
        }
        prometheus :9153
        forward . 172.16.0.1
        cache 30
        loop
        reload
        loadbalance
    }
    consul.local:53 {
        errors
        cache 30
        forward . 10.150.0.1
    }
```

Note: CoreDNS does not support FQDNs for stub-domains and nameservers (eg: "ns.foo.com"). During translation, all FQDN nameservers will be omitted from the CoreDNS config.

What's next

- Read Debugging DNS Resolution

Feedback

Was this page helpful?

Yes No

Last modified January 11, 2023 at 11:12 AM PST: Update page weights in /tasks/administer-cluster section (b1202c78ff)