Debugging DNS Resolution

This page provides hints on diagnosing DNS problems.

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 configured to use the CoreDNS addon or its precursor, kube-dns.

Your Kubernetes server must be at or later than version v1.6. To check the version, enter `kubectl version`.

Create a simple Pod to use as a test environment

admin/dns/dnsutils.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dnsutils
  namespace: default
spec:
  containers:
  - name: dnsutils
    image: registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3
    command:
      - sleep
      - "infinity"
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
```

Note: This example creates a pod in the `default` namespace. DNS name resolution for services depends on the namespace of the pod. For more information, review DNS for Services and Pods.

Use that manifest to create a Pod:

```
kubectl apply -f https://k8s.io/examples/admin/dns/dnsutils.yaml
```

```
pod/dnsutils created
```

...and verify its status:

```
kubectl get pods dnsutils
```

```
NAME      READY     STATUS    RESTARTS   AGE
dnsutils   1/1       Running   0          <some-time>
```

Once that Pod is running, you can exec `nslookup` in that environment. If you see something like the following, DNS is working correctly.

```
kubectl exec -i -t dnsutils -- nslookup kubernetes.default
```

```
Server:    10.0.0.10
Address 1: 10.0.0.10

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

If the `nslookup` command fails, check the following:

Check the local DNS configuration first

Take a look inside the resolv.conf file. (See Customizing DNS Service and Known issues below for more information)

```
kubectl exec -ti dnsutils -- cat /etc/resolv.conf
```

Verify that the search path and name server are set up like the following (note that search path may vary for different cloud providers):

```
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
```

Errors such as the following indicate a problem with the CoreDNS (or kube-dns) add-on or with associated Services:

```
kubectl exec -i -t dnsutils -- nslookup kubernetes.default
```

```
Server:    10.0.0.10
Address 1: 10.0.0.10

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

or

```
kubectl exec -i -t dnsutils -- 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'
```

Check if the DNS pod is running

Use the `kubectl get pods` command to verify that the DNS pod is running.

```
kubectl get pods --namespace=kube-system -l k8s-app=kube-dns
```

```
NAME                       READY     STATUS    RESTARTS   AGE
...
coredns-7b96bf9f76-5hsxb   1/1       Running   0          1h
coredns-7b96bf9f76-mvmmt   1/1       Running   0          1h
...
```

Note: The value for label `k8s-app` is `kube-dns` for both CoreDNS and kube-dns deployments.

If you see that no CoreDNS Pod is running or that the Pod has failed/completed, the DNS add-on may not be deployed by default in your current environment and you will have to deploy it manually.

Check for errors in the DNS pod

Use the `kubectl logs` command to see logs for the DNS containers.

For CoreDNS:

```
kubectl logs --namespace=kube-system -l k8s-app=kube-dns
```

Here is an example of a healthy CoreDNS log:

```
.:53
2018/08/15 14:37:17 [INFO] CoreDNS-1.2.2
2018/08/15 14:37:17 [INFO] linux/amd64, go1.10.3, 2e322f6
CoreDNS-1.2.2
linux/amd64, go1.10.3, 2e322f6
2018/08/15 14:37:17 [INFO] plugin/reload: Running configuration MD5 = 24e6c59e83ce706f07bcc82c31b1ea1c
```

See if there are any suspicious or unexpected messages in the logs.

Is DNS service up?

Verify that the DNS service is up by using the `kubectl get service` command.

```
kubectl get svc --namespace=kube-system
```

```
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)             AGE
...
kube-dns     ClusterIP   10.0.0.10      <none>        53/UDP,53/TCP        1h
...
```

Note: The service name is `kube-dns` for both CoreDNS and kube-dns deployments.

If you have created the Service or in the case it should be created by default but it does not appear, see debugging Services for more information.

Are DNS endpoints exposed?

You can verify that DNS endpoints are exposed by using the `kubectl get endpoints` command.

```
kubectl get endpoints kube-dns --namespace=kube-system
```

```
NAME       ENDPOINTS                       AGE
kube-dns   10.180.3.17:53,10.180.3.17:53    1h
```

If you do not see the endpoints, see the endpoints section in the debugging Services documentation.

For additional Kubernetes DNS examples, see the cluster-dns examples in the Kubernetes GitHub repository.

Are DNS queries being received/processed?

You can verify if queries are being received by CoreDNS by adding the `log` plugin to the CoreDNS configuration (aka Corefile). The CoreDNS Corefile is held in a ConfigMap named `coredns`. To edit it, use the command:

```
kubectl -n kube-system edit configmap coredns
```

Then add `log` in the Corefile section per the example below:

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

After saving the changes, it may take up to minute or two for Kubernetes to propagate these changes to the CoreDNS pods.

Next, make some queries and view the logs per the sections above in this document. If CoreDNS pods are receiving the queries, you should see them in the logs.

Here is an example of a query in the log:

```
.:53
2018/08/15 14:37:15 [INFO] CoreDNS-1.2.0
2018/08/15 14:37:15 [INFO] linux/amd64, go1.10.3, 2e322f6
CoreDNS-1.2.0
linux/amd64, go1.10.3, 2e322f6
2018/09/07 15:29:04 [INFO] plugin/reload: Running configuration MD5 = 162475cdf272d8aa601e6fe67a6ad42f
2018/09/07 15:29:04 [INFO] Reloading complete
172.17.0.18:41675 - [07/Sep/2018:15:29:11 +0000] 59925 "A IN kubernetes.default.svc.cluster.local. udp 54 false 512
```

Does CoreDNS have sufficient permissions?

CoreDNS must be able to list service and endpoint related resources to properly resolve service names.

Sample error message:

```
2022-03-18T07:12:15.699431183Z [INFO] 10.96.144.227:52299 - 3686 "A IN serverproxy.contoso.net.cluster.local. udp 5
```

First, get the current ClusterRole of `system:coredns`:

```
kubectl describe clusterrole system:coredns -n kube-system
```

Expected output:

```
PolicyRule:
  Resources                   Non-Resource URLs  Resource Names  Verbs
  ---------                   -----------------  --------------  -----
  endpoints                   []                 []              [list watch]
  namespaces                  []                 []              [list watch]
  pods                        []                 []              [list watch]
  services                    []                 []              [list watch]
  endpointslices.discovery.k8s.io  []                 []              [list watch]
```

If any permissions are missing, edit the ClusterRole to add them:

```
kubectl edit clusterrole system:coredns -n kube-system
```

Example insertion of EndpointSlices permissions:

```
...
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
...
```

Are you in the right namespace for the service?

DNS queries that don't specify a namespace are limited to the pod's namespace.

If the namespace of the pod and service differ, the DNS query must include the namespace of the service.

This query is limited to the pod's namespace:

```
kubectl exec -i -t dnsutils -- nslookup <service-name>
```

This query specifies the namespace:

```
kubectl exec -i -t dnsutils -- nslookup <service-name>.<namespace>
```

To learn more about name resolution, see DNS for Services and Pods.

Known issues

Some Linux distributions (e.g. Ubuntu) use a local DNS resolver by default (systemd-resolved). Systemd-resolved moves and replaces `/etc/resolv.conf` with a stub file that can cause a fatal forwarding loop when resolving names in upstream servers. This can be fixed manually by using kubelet's `--resolv-conf` flag to point to the correct `resolv.conf` (With `systemd-resolved`, this is `/run/systemd/resolve/resolv.conf`). kubeadm automatically detects `systemd-resolved`, and adjusts the kubelet flags accordingly.

Kubernetes installs do not configure the nodes' `resolv.conf` files to use the cluster DNS by default, because that process is inherently distribution-specific. This should probably be implemented eventually.

Linux's libc (a.k.a. glibc) has a limit for the DNS `nameserver` records to 3 by default and Kubernetes needs to consume 1 `nameserver` record. This means that if a local installation already uses 3 `nameserver`s, some of those entries will be lost. To work around this limit, the node can run `dnsmasq`, which will provide more `nameserver` entries. You can also use kubelet's `--resolv-conf` flag.

If you are using Alpine version 3.17 or earlier as your base image, DNS may not work properly due to a design issue with Alpine. Until musl version 1.24 didn't include TCP fallback to the DNS stub resolver meaning any DNS call above 512 bytes would fail. Please upgrade your images to Alpine version 3.18 or above.

What's next

- See Autoscaling the DNS Service in a Cluster.
- Read DNS for Services and Pods

Feedback

Was this page helpful?

Yes No

Last modified August 24, 2023 at 6:38 PM PST: Use code_sample shortcode instead of code shortcode (e8b136c3b3)