Understanding how Kubernetes DNS services work

Kubernetes allows you to create container groups and define services on top of them. Kubernetes assigns each service a virtual static IP address routable within the cluster, so any connection that reaches this IP address will be automatically routed to one of the containers in the group.

The benefit of using services is that you are able to access the functionality provided by the containers without knowing their identity. This is basically an easy to discover load balancer. And to make things even easier, Kubernetes also generates an internal DNS entry that resolves to this IP address.

Deploying a Simple Service

We are going to deploy a simple service called `backend` within a namespace `critical-app`. The `backend` service is the entry point to 3 Nginx pods grouped in what Kubernetes calls a deployment.

```
kubectl create namespace critical-app
```

```
namespace/critical-app created
```

```
cat << EOF > backend.yaml
apiVersion: v1
kind: Service
metadata:
  name: backend
  namespace: critical-app
  labels:
    app: critical-app
    role: backend
spec:
  ports:
    - port: 80
  selector:
    app: critical-app
    role: backend
---
apiVersion: extensions/v1beta1
kind: Deployment
```

```
metadata:
  name: backend
  namespace: critical-app
spec:
  replicas: 3
  strategy:
    rollingUpdate:
      maxUnavailable: 0
      maxSurge: 3
  template:
    metadata:
      labels:
        app: critical-app
        role: backend
    spec:
      containers:
        - name: backend
          image: nginx:1.10
EOF

kubectl create -f backend.yaml
```

```
service/backend created
deployment.extensions/backend created
```

Now we can use the describe command to show the IP address assigned to the service:

```
kubectl describe service backend --namespace critical-app
```

```
Name:              backend
Namespace:         critical-app
Labels:            app=critical-app
                   role=backend
Annotations:       <none>
Selector:          app=critical-app,role=backend
Type:              ClusterIP
IP:                10.7.241.247
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.4.0.55:80,10.4.1.25:80,10.4.2.57:80
Session Affinity:  None
Events:            <none>
```

Here we can also identify the 3 underlying pods IP addresses as the Endpoints.

Let's create an interactive container with `curl` installed to ping the `backend` entry point:

```
kubectl run -it --image=tutum/curl client --namespace critical-app --restart=Never
```

```
If you don't see a command prompt, try pressing enter.
root@client:/#
```

```
curl backend # or curl backend.critical-app.svc.cluster.local
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Welcome to nginx!</title>
    <style>
      body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Welcome to nginx!</h1>
    <p>
      If you see this page, the nginx web server is successfully installed and
      working. Further configuration is required.
    </p>

    <p>
      For online documentation and support please refer to
      <a href="http://nginx.org/">nginx.org</a>.<br />
      Commercial support is available at
      <a href="http://nginx.com/">nginx.com</a>.
    </p>

    <p><em>Thank you for using nginx.</em></p>
  </body>
</html>
```

Automagically I get a response from one of the underlying Nginx containers. I didn't have to know the identity, Kubernetes DNS did everything for me.

```
apt update && apt install dnsutils -y
```

```
nslookup backend
```

```
Server:         10.39.240.10
Address:        10.39.240.10#53

Name:   backend.critical-app.svc.cluster.local
Address: 10.39.241.167
```

```
dig +search backend
```

```
; <<>> DiG 9.9.5-3ubuntu0.18-Ubuntu <<>> +search backend
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54095
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;backend.critical-app.svc.cluster.local.        IN A

;; ANSWER SECTION:
backend.critical-app.svc.cluster.local. 30 IN A 10.39.241.167

;; Query time: 1 msec
;; SERVER: 10.39.240.10#53(10.39.240.10)
;; WHEN: Sat Jan 05 09:47:44 UTC 2019
;; MSG SIZE  rcvd: 72
```