DBA From The Cold

Ramblings on working as a SQL Server DBA

Differences between using a Load Balanced Service and an Ingress in Kubernetes

Nov 23, 2020 ~ dbafromthecold

What is the difference between using a load balanced service (https://kubernetes.io/docs/concepts/services-networking/service/) and an ingress (https://kubernetes.io/docs/concepts/services-networking/ingress/) to access applications in Kubernetes?

Basically, they achieve the same thing. Being able to access an application that's running in Kubernetes from outside of the cluster, but there are differences!

The key difference between the two is that ingress operates at networking layer 7 (the application layer) so routes connections based on http host header or url path. Load balanced services operate at layer 4 (the transport layer) so can load balance arbitrary tcp/udp/sctp services.

Ok, that statement doesn't really clear things up (for me anyway). I'm a practical person by nature… so let's run through examples of both (running everything in Kubernetes for Docker Desktop).

What we're going to do is spin up two nginx pages that will serve as our applications and then firstly use load balanced services to access them, followed by an ingress.

So let's create two nginx deployments from a custom image (available on the GHCR): –

```
kubectl create deployment nginx-page1 --image=ghcr.io/dbafromthecold/r
kubectl create deployment nginx-page2 --image=ghcr.io/dbafromthecold/r
```

And expose those deployments with a load balanced service: –

```
kubectl expose deployment nginx-page1 --type=LoadBalancer --port=8000
kubectl expose deployment nginx-page2 --type=LoadBalancer --port=9000
```

Confirm that the deployments and services have come up successfully: –

```
kubectl get all
```

```
PS C:\> kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx-page1-587c46b847-5nl2g    1/1     Running   0          20s
pod/nginx-page2-cf9d667cf-ssgdd     1/1     Running   0          18s

NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/kubernetes    ClusterIP      10.96.0.1       <none>        443/TCP          11m
service/nginx-page1   LoadBalancer   10.106.36.48    localhost     8000:31412/TCP   4s
service/nginx-page2   LoadBalancer   10.111.165.18   localhost     9000:32705/TCP   4s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-page1    1/1     1            1           20s
deployment.apps/nginx-page2    1/1     1            1           18s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-page1-587c46b847    1         1         1       20s
replicaset.apps/nginx-page2-cf9d667cf     1         1         1       18s
```

(https://dbafromthecold.files.wordpress.com/2020/11/1.nginx-exposed-with-load-balanced-services-1.png)

Ok, now let's check that the nginx pages are working. As we've used a load balanced service in k8s in Docker Desktop they'll be available as localhost:PORT: –

```
curl localhost:8000
curl localhost:9000
```

```
PS C:\> curl localhost:8000
<!DOCTYPE html>
<html>
<head>
<title>Greetings from dbafromthecold!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>This is demo page 1!</h1>

<p>This is demo page of one dbafromthecold's ingress test</p>

<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For further information, please visit
<a href="www.dbafromthecold.com">www.dbafromthecold.com</a>.<br/>

<p><em>Thank you for reading!</em></p>
</body>
</html>
PS C:\>
```

```
PS C:\> curl localhost:9000
<!DOCTYPE html>
<html>
<head>
<title>Greetings from dbafromthecold!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>This is demo page 2!</h1>

<p>This is demo page of two dbafromthecold's ingress test</p>

<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For further information, please visit
<a href="www.dbafromthecold.com">www.dbafromthecold.com</a>.<br/>

<p><em>Thank you for reading!</em></p>
</body>
</html>
PS C:\>
```

(https://dbafromthecold.files.wordpress.com/2020/11/1.-curl-against-load-balanced-services.png)

Great! So we're using the external IP address (local host in this case) and a port number to connect to our applications.

Now let's have a look at using an ingress.

First, let's get rid of those load balanced services: –

```
kubectl delete service nginx-page1 nginx-page2
```

And create two new cluster IP services: –

```
kubectl expose deployment nginx-page1 --type=ClusterIP --port=8000 --t
kubectl expose deployment nginx-page2 --type=ClusterIP --port=9000 --t
```

So now we have our pods running and two cluster IP services, which aren't accessible from outside of the cluster: –

```
PS C:\> kubectl delete service nginx-page1 nginx-page2
service "nginx-page1" deleted
service "nginx-page2" deleted
PS C:\> kubectl expose deployment nginx-page1 --type=ClusterIP --port=8000 --target-port=80
service/nginx-page1 exposed
PS C:\> kubectl expose deployment nginx-page2 --type=ClusterIP --port=9000 --target-port=80
service/nginx-page2 exposed
PS C:\> kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
nginx-page1-587c46b847-5nl2g   1/1     Running   0          6m25s
nginx-page2-cf9d667cf-ssgdd    1/1     Running   0          6m23s
PS C:\> kubectl get services
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP    17m
nginx-page1   ClusterIP   10.105.135.237   <none>        8000/TCP   10s
nginx-page2   ClusterIP   10.98.25.213     <none>        9000/TCP   9s
PS C:\>
```

(https://dbafromthecold.files.wordpress.com/2020/11/3.nginx-exposed-with-cluster-ip-services.png)

The services have no external IP so what we need to do is deploy an ingress controller.

An ingress controller will provide us with one external IP address, that we can map to a DNS entry. Once the controller is up and running we then use an ingress resources to define routing rules that will map external requests to different services within the cluster.

Kubernetes currently supports GCE and nginx controllers, we're going to use an nginx ingress controller (https://github.com/kubernetes/ingress-nginx).

To spin up the controller run: –

```
1 kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-
```

```
PS C:\> kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.40.2/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
configmap/ingress-nginx-controller created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
service/ingress-nginx-controller-admission created
service/ingress-nginx-controller created
deployment.apps/ingress-nginx-controller created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
serviceaccount/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
PS C:\>
```

(https://dbafromthecold.files.wordpress.com/2020/11/4.deploy-ingress-controller.png)

We can see the number of resources that's going to create its own namespace, and to confirm they're all up and running: –

```
1 kubectl get all -n ingress-nginx
```

```
PS C:\> kubectl get all -n ingress-nginx
NAME                                        READY   STATUS      RESTARTS   AGE
pod/ingress-nginx-admission-create-hmzh6    0/1     Completed   0          76s
pod/ingress-nginx-admission-patch-c4bxr     0/1     Completed   1          76s
pod/ingress-nginx-controller-98cb87fb7-w8zjj   1/1     Running     0          76s

NAME                                         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
service/ingress-nginx-controller             LoadBalancer   10.102.74.201   localhost     80:32023/TCP,443:30028/TCP   76s
service/ingress-nginx-controller-admission   ClusterIP      10.96.249.44    <none>        443/TCP                      76s

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller   1/1     1            1           76s

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/ingress-nginx-controller-98cb87fb7    1         1         1       76s

NAME                                       COMPLETIONS   DURATION   AGE
job.batch/ingress-nginx-admission-create   1/1           3s         76s
job.batch/ingress-nginx-admission-patch    1/1           4s         76s
PS C:\>
```

(https://dbafromthecold.files.wordpress.com/2020/11/5.-ingress-controller-resources.png)

Note the external IP of "localhost" for the ingress-nginx-controller service.

Ok, now we can create an ingress to direct traffic to our applications. Here's an example ingress.yaml file: –

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-testwebsite
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: www.testwebaddress.com
    http:
      paths:
       - path: /pageone
         pathType: Prefix
         backend:
           service:
             name: nginx-page1
             port:
               number: 8000
       - path: /pagetwo
         pathType: Prefix
         backend:
           service:
             name: nginx-page2
             port:
               number: 9000
```

Watch out here. In Kubernetes v1.19 (https://kubernetes.io/docs/setup/release/notes/) ingress went GA so the apiVersion changed. The yaml above won't work in any version prior to v1.19.

Anyway, the main points in this yaml are: –

```yaml
annotations:
  kubernetes.io/ingress.class: "nginx"
```

Which makes this ingress resource use our ingress nginx controller.

```
rules:
- host: www.testwebaddress.com
```

Which sets the URL we'll be using to access our applications to ***http://www.testwebaddress.com (http://www.testwebaddress.com)***

```
- path: /pageone
  pathType: Prefix
  backend:
    service:
      name: nginx-page1
      port:
        number: 8000
- path: /pagetwo
  pathType: Prefix
  backend:
    service:
      name: nginx-page2
      port:
        number: 9000
```

Which routes our requests to the backend cluster IP services depending on the path (e.g. – ***http://www.testwebaddress.com/pageone (http://www.testwebaddress.com/pageone)*** will be directed to the nginx-page1 service)

You can create the ingress.yaml file manually and then deploy to Kubernetes or just run: –

```
kubectl apply -f https://gist.githubusercontent.com/dbafromthecold/a68
```

Confirm that the ingress is up and running (it'll take a minute to get an address): –

```
kubectl get ingress
```

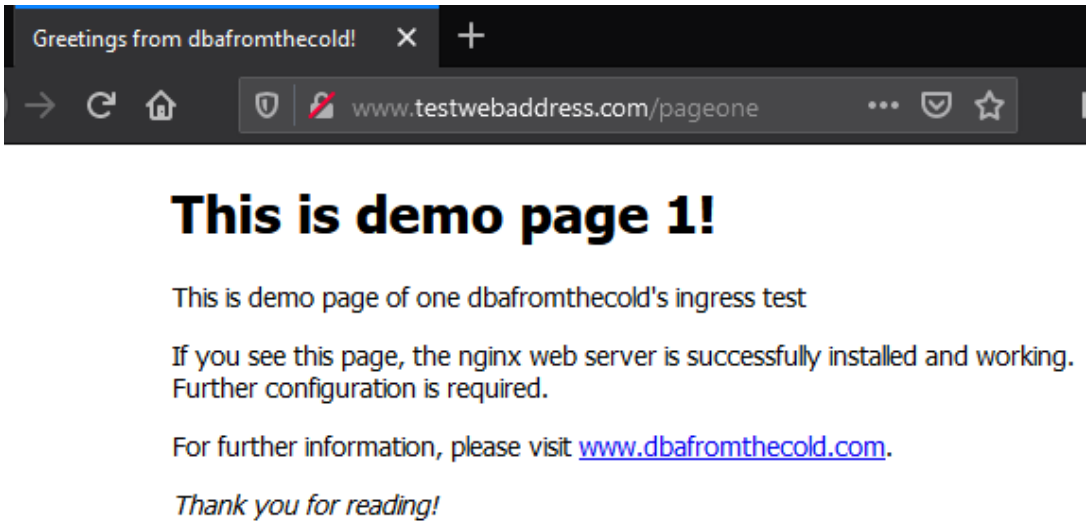
```
PS C:\> kubectl get ingress
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
NAME                 CLASS    HOSTS                   ADDRESS     PORTS   AGE
ingress-testwebsite  <none>   www.testwebaddress.com  localhost   80      30s
PS C:\>
```

(https://dbafromthecold.files.wordpress.com/2020/11/6.-show-ingress-resource.png)
N.B. – Ignore the warning (if you get one like in the screen shot above), we're using the correct API version

Finally, we now also need to add an entry for the web address into our hosts file (simulating a DNS entry): –

```
127.0.0.1 www.testwebaddress.com
```

And now we can browse to the web pages to see the ingress in action!



(https://dbafromthecold.files.wordpress.com/2020/11/7.-browse-to-website.png)

And that's the differences between using load balanced services or an ingress to connect to applications running in a Kubernetes cluster. The ingress allows us to only use the one external IP address and then route traffic to different backend services whereas with the load balanced services, we would need to use different IP addresses (and ports if configured that way) for each application.

Thanks for reading!

This entry was posted in Kubernetes, Linux. Bookmark the permalink.

WordPress.com.