Creating Ingress Resources Based on Domains

In this lesson, we will learn to create Ingress Resources based on domains.

WE'LL COVER THE FOLLOWING

Refactoring the Definition

We'll try to refactor our `devops-toolkit` Ingress definition so that the Controller forwards requests coming from the `devopstoolkitseries.com` domain. The change should be minimal, so we'll get down to it right away.

```
cat ingress/devops-toolkit-dom.yml
```

When compared with the previous definition, the **only difference** is in the additional entry `host: devopstoolkitseries.com`. Since that will be the only application accessible through that domain, we also removed the `path: /` entry.

Applying the New Definition

Let's `apply` the new definition.

```
kubectl apply \
  -f ingress/devops-toolkit-dom.yml \
  --record
```

What would happen if we send a similar domain-less request to the Application? We're sure you already know the answer, but we'll check it out anyways.

```
curl -I "http://$IP"
```

The **output** is as follows.

```
HTTP/1.1 404 Not Found
Server: nginx/1.15.9
Date: Wed, 19 Jun 2019 11:12:42 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 21
Connection: keep-alive
```

There is **no** Ingress resource defined to listen to `/`. The updated Ingress will forward requests only if they come from `devopstoolkitseries.com`.

Since it's not feasible to give you access to the DNS registry of `devopstoolkitseries.com`. So you cannot configure it with the IP of your Minikube cluster. Therefore, we won't be able to test it by sending a request to `devopstoolkitseries.com`.

What we can do is to "fake" it by adding that domain to the request header.

```
curl -I \
    -H "Host: devopstoolkitseries.com" \
    "http://$IP"
```

The **output** is as follows.

```
HTTP/1.1 200 OK
Server: nginx/1.15.9
Date: Wed, 19 Jun 2019 11:13:28 GMT
Content-Type: text/html
Content-Length: 6109
Connection: keep-alive
Vary: Accept-Encoding
Last-Modified: Wed, 10 Apr 2019 22:06:08 GMT
ETag: "5cae68d0-17dd"
Accept-Ranges: bytes
```

Now that Ingress received a request that looks like it's coming from the domain `devopstoolkitseries.com`, it forwarded it to the `devops-toolkit` Service which, in turn, load balanced it to one of the `devops-toolkit` Pods. As a result, we got the response `200 OK`.

Just to be on the safe side, we'll verify whether `go-demo-2` Ingress still works.

```
curl -H "Host: acme.com" \
    "http://$IP/demo/hello"
```

We got the famous `hello, world!` response, thus confirming that both Ingress resources are operational. Even though we "faked" the last request as if it's coming from `acme.com`, it still worked. Since the `go-demo-2` Ingress does not have any `host` defined, it accepts any request with the `path` starting with `/demo`.

We're still missing a few things. One of those is a setup of a default backend. We'll go through it in the next lesson.