Enabling Ingress Controllers

In this lesson, we will enable the Ingress controller and play around with it.

WE'LL COVER THE FOLLOWING

Why Ingress Controllers are Required?

We need a mechanism that will accept requests on pre-defined ports (e.g., `80` and `443`) and forward them to Kubernetes Services. It should be able to distinguish requests based on paths and domains as well as to be able to perform SSL offloading.

Kubernetes itself does not have a ready-to-go solution for this. Unlike other types of Controllers that are typically part of the `kube-controller-manager` binary, Ingress Controller needs to be installed separately. Instead of a Controller, `kube-controller-manager` offers *Ingress resource* that other third-party solutions can utilize to provide requests forwarding and SSL features. In other words, Kubernetes only provides an *API*, and we need to set up a Controller that will use it.

Fortunately, the community already built a myriad of Ingress Controllers. We won't evaluate all of the available options since that would require a lot of space, and it would mostly depend on your needs and your hosting vendor. Instead, we'll explore how Ingress Controllers work through the one that is already available in Minikube.

Exploring Minikube Addons

Let's take a look at the list of the Minikube addons.

```
minikube addons list
```

The **output** is as follows.

```
- addon-manager: enabled
- dashboard: enabled
- default-storageclass: enabled
- efk: disabled
- freshpod: disabled
- gvisor: disabled
- heapster: disabled
- ingress: enabled
- logviewer: disabled
- metrics-server: disabled
- nvidia-driver-installer: disabled
- nvidia-gpu-device-plugin: disabled
- registry: disabled
- registry-creds: disabled
- storage-provisioner: enabled
```

We can see that `ingress` is available as one of the Minikube addons and it is enabled by default

> ℹ If the `ingress` addon is not already enabled. You can enable it by executing the command that follows.

```
minikube addons enable ingress
```

> ℹ Once enabled, minikube addons will be added to every new cluster we create. We will continue enabling the required addons in every chapter only to allow readers to skip them at their convenience.

Playing Around with the Controller

Now that `ingress` addon is enabled, we'll check whether it is running inside our cluster.

```
kubectl get pods -n kube-system \
    | grep ingress
```

Ignore the `-n` argument. We did not yet explore Namespaces. For now, please note that the output of the command should show that `nginx-ingress-controller-...` Pod is running.

If the output is empty, you might need to wait for a few moments until the containers are pulled, and re-execute the `kubectl get all --namespace ingress-nginx` command again.

> ℹ The Ingress Controller that ships with Minikube is based on the gcr.io/google_containers/nginx-ingress-controller image hosted in Google Cloud Platform (GCP) Container Registry. The image is based on NGINX Ingress Controller. It is one of the only two currently supported and maintained by the Kubernetes community. The other one is GLBC that comes with Google Compute Engine (GCE) Kubernetes hosted solution.

By default, the Ingress Controller is configured with only two endpoints.

If we'd like to check Controller's health, we can send a request to `/healthz`.

```
curl -i "http://$IP/healthz"
```

The **output** is as follows.

```
HTTP/1.1 200 OK
Server: nginx/1.15.9
Date: Mon, 10 Jun 2019 12:02:11 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
```

It responded with the status code `200 OK`, thus indicating that it is healthy and ready to serve requests. There's not much more to it so we'll move to the second endpoint.

The Ingress Controller has a default catch-all endpoint that is used when a request does not match any of the other criteria. Since we did not yet create any Ingress Resource, this endpoint should provide the same response to all requests except `/healthz`.

```
curl -i "http://$IP/something"
```

The **output** is as follows.

```
HTTP/1.1 404 Not Found
Server: nginx/1.13.5
Date: Sun, 24 Dec 2017 15:36:23 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 21
Connection: keep-alive

default backend - 404
```

We got the response indicating that the requested resource could not be found.

Now that we're ready to create our first Ingress Resource, let's go for it in the next lesson.