Set up Ingress on Minikube with the NGINX Ingress Controller

An Ingress is an API object that defines rules which allow external access to services in a cluster. An Ingress controller fulfills the rules set in the Ingress.

This page shows you how to set up a simple Ingress which routes requests to Service 'web' or 'web2' depending on the HTTP URI.

Before you begin

This tutorial assumes that you are using `minikube` to run a local Kubernetes cluster. Visit Install tools to learn how to install `minikube`.

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 Kubernetes server must be at or later than version 1.19. To check the version, enter `kubectl version`. If you are using an older Kubernetes version, switch to the documentation for that version.

Create a minikube cluster

If you haven't already set up a cluster locally, run `minikube start` to create a cluster.

Enable the Ingress controller

1. To enable the NGINX Ingress controller, run the following command:
2. `minikube addons enable ingress`
3. Verify that the NGINX Ingress controller is running
4. `kubectl get pods -n ingress-nginx`

Note: It can take up to a minute before you see these pods running OK.

The output is similar to:

```
NAME                                        READY   STATUS
RESTARTS    AGE
ingress-nginx-admission-create-g9g49        0/1     Completed   0
11m
ingress-nginx-admission-patch-rqp78         0/1     Completed   1
11m
ingress-nginx-controller-59b45fb494-26npt   1/1     Running     0
11m
```

Deploy a hello, world app

1. Create a Deployment using the following command:
2. `kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0`

   The output should be:

   ```
   deployment.apps/web created
   ```
3. Expose the Deployment:
4. `kubectl expose deployment web --type=NodePort --port=8080`

   The output should be:

   ```
   service/web exposed
   ```
5. Verify the Service is created and is available on a node port:
6. `kubectl get service web`

   The output is similar to:

   ```
   NAME     TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)
   AGE
   web      NodePort   10.104.133.249   <none>        8080:31637/TCP
   12m
   ```
7. Visit the Service via NodePort:
8. `minikube service web --url`

   The output is similar to:

   ```
   http://172.17.0.15:31637
   curl http://172.17.0.15:31637
   ```

   The output is similar to:

   ```
   Hello, world!
   Version: 1.0.0
   Hostname: web-55b8c6998d-8k564
   ```

You can now access the sample application via the Minikube IP address and NodePort. The next step lets you access the application using the Ingress resource.

Create an Ingress

The following manifest defines an Ingress that sends traffic to your Service via `hello-world.info`.

1. Create `example-ingress.yaml` from the following file:

service/networking/example-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - host: hello-world.info
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
```

2. Create the Ingress object by running the following command:
3. `kubectl apply -f https://k8s.io/examples/service/networking/example-ingress.yaml`

   The output should be:

```
ingress.networking.k8s.io/example-ingress created
```

4. Verify the IP address is set:
5. `kubectl get ingress`

   Note: This can take a couple of minutes.

   You should see an IPv4 address in the `ADDRESS` column; for example:

```
NAME                CLASS    HOSTS              ADDRESS        PORTS
AGE
example-ingress     <none>   hello-world.info   172.17.0.15    80
38s
```

6. Verify that the Ingress controller is directing traffic:

```
7. curl --resolve "hello-world.info:80:$( minikube ip )" -i
   http://hello-world.info
```

   You should see:

```
Hello, world!
Version: 1.0.0
Hostname: web-55b8c6998d-8k564
```

   You can also visit `hello-world.info` from your browser.

   - **Optionally** Look up the external IP address as reported by minikube:
   - `minikube ip`

     Add line similar to the following one to the bottom of the `/etc/hosts` file on your computer (you will need administrator access):

     ```
     172.17.0.15 hello-world.info
     ```

     Note: Change the IP address to match the output from `minikube ip`.

     After you make this change, your web browser sends requests for `hello-world.info` URLs to Minikube.

Create a second Deployment

1. Create another Deployment using the following command:

```
2. kubectl create deployment web2 --image=gcr.io/google-samples/hello-
   app:2.0
```

   The output should be:

```
deployment.apps/web2 created
```

3. Expose the second Deployment:

```
4. kubectl expose deployment web2 --port=8080 --type=NodePort
```

   The output should be:

```
service/web2 exposed
```

Edit the existing Ingress

1. Edit the existing `example-ingress.yaml` manifest, and add the following lines at the end:

```yaml
- path: /v2
  pathType: Prefix
  backend:
    service:
      name: web2
      port:
        number: 8080
```

9. Apply the changes:

```
kubectl apply -f example-ingress.yaml
```

   You should see:

```
ingress.networking/example-ingress configured
```

Test your Ingress

1. Access the 1st version of the Hello World app.

```
curl --resolve "hello-world.info:80:$( minikube ip )" -i
http://hello-world.info
```

   The output is similar to:

```
Hello, world!
Version: 1.0.0
Hostname: web-55b8c6998d-8k564
```

3. Access the 2nd version of the Hello World app.

```
curl --resolve "hello-world.info:80:$( minikube ip )" -i
http://hello-world.info/v2
```

   The output is similar to:

```
Hello, world!
Version: 2.0.0
Hostname: web2-75cd47646f-t8cjk
```

   Note: If you did the optional step to update `/etc/hosts`, you can also visit `hello-world.info` and `hello-world.info/v2` from your browser.

What's next

- Read more about Ingress
- Read more about Ingress Controllers
- Read more about Services