Setting up Ingress on Minikube

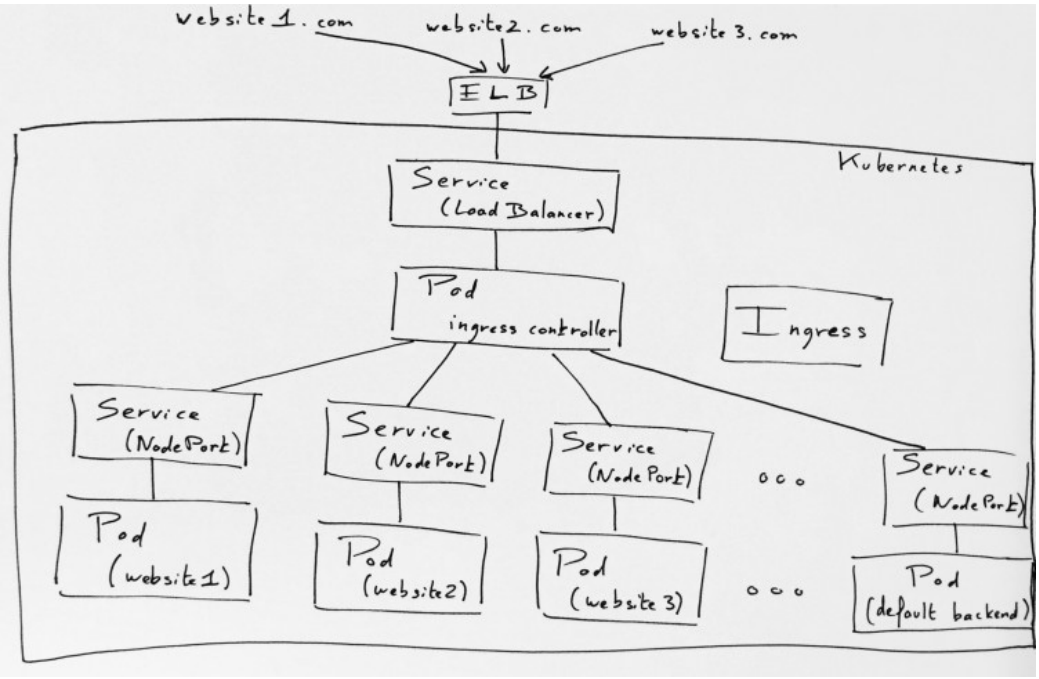


photo courtesy: http://www.sandtable.com/a-single-aws-elastic-load-balancer-for-several-kubernetes-services-using-kubernetes-ingress/

Hosted Kubernetes instances, especially on GCE, come with a certain number of features and configurations out of the box unlike Minikube, the local development platform for kubernetes.

Although, minikube supports (almost) everything you would expect from a kubernetes cluster but since it's running locally, certain cloud provider features will not work out of the box. One such feature is Ingress.

What's Ingress?

An ingress is a set of rules that allows inbound connections to reach the kubernetes cluster services.

Typically, the kubernetes cluster is *firewalled* from the internet. It has edge routers enforcing the firewall. Kubernetes resources like **services, pods,** have IPs only routable by the cluster

network and are not (directly) accessible outside the cluster. All traffic that ends up at an edge router is either dropped or forwarded elsewhere. So, submitting an ingress to the cluster defines a set of rules for routing external traffic to the kubernetes endpoints.

Let's skip ahead and view what these rules look like. Here's one will use for this article
https://gist.github.com/0sc/77d8925cc378c9a6a92890e7c08937ca

As you might have guessed, the rule is:

- all requests to `myminikube.info/` should be routed to the service in the cluster named **echoserver**.
- requests mapping to `cheeses.all/stilton` should be routed to the **stilton-cheese** service.
- and finally, requests mapping to `cheeses.all/cheddar` should be routed to the **cheddar-cheese** service.

Of course, there's more to it; like the `backend` tag which implies that unmatched requests should be routed to the `default-http-backend` service and there's also the familiar kubernetes tags; for example the `apiVersion` tag which clearly marks ingress as a beta feature.

Note the annotation:

```
ingress.kubernetes.io/rewrite-target: /
```

This is necessary if the target services expect requests from the root URL i.e `cheeses.all` and not `cheeses.all/stilton` . The ingress mapping by default will pass along the trailing path on to the service *(e.g /stilton)* and if the service doesn't accept request on that path you get a 403 error response. Thus with `rewrite-target` annotation, the request path is rewritten with the given path before the request get's forwarded to the target backend.

The ingress controller

In order for the Ingress resource to work, the cluster must have an Ingress controller running. When a user requests an ingress by POSTing an Ingress resource (such as the one above) to the API server, the Ingress controller is responsible for fulfilling the Ingress, usually with a loadbalancer. Though it may also configure the edge routers or additional frontends to help handle the traffic.

As such without an Ingress controller to satisfy the ingress, merely creating the ingress resource will have no effect.

You can write your own controller but you need not to. There are readily available third-party ingress controllers like the Nginx, Traefik, HAproxy controllers which you could easily leverage. *I will be using the Nginx controller for this demo but feel free to try out any other*.

Setup

Minikube v0.14.0 (and above) ships with Nginx ingress setup as an add-on (as requested here). It can be easily enabled by executing

```
minikube addons enable ingress
```

Enabling the add-on provisions the following:

- a configMap for the Nginx loadbalancer
- the Nginx ingress controller
- a service that exposes a default Nginx backend pod for handling unmapped requests.

If you are using an older version of minikube (and insist on not updating) you might need to manually deploy the ingress controller (and default backend service).

The layout of our cluster for this demo is:

- A backend that will receive requests for **myminikube.info** and displays some basic information about the cluster and the request**.**
- A pair of backends that will receive the request for **cheeses.all** .One whose path begins with `/stilton` and another whose path begins with `/cheddar` .

Nginx already provides a default backend so we need not worry about that.

Let's set up the echoserver deployments and expose them:

```
kubectl run echoserver --
image=gcr.io/google_containers/echoserver:1.4 --port=8080
kubectl expose deployment echoserver --type=NodePort
```

Then confirm that requests can get to the service

```
minikube service echoserver
```

This should open the service in your default browser.
Next we setup the backend for `/stilton` cheese

```
kubectl run stilton-cheese --image=errm/cheese:stilton --
port=80
kubectl expose deployment stilton-cheese --type=NodePort
```

You can also check out the service:

```
minikube service stilton-cheese
```

Finally the `/cheddar` cheese

```
kubectl run cheddar-cheese --image=errm/cheese:cheddar --
port=80
kubectl expose deployment cheddar-cheese --type=NodePort
minikube service cheddar-cheese
```

Thus far, we can access the services via the [minikube ip]:[node port] address. Our aim, however, is to access them via `myminikube.info` , `cheeses.all/stilton` and `cheeses.all/cheddar` . And that's where ingress comes in.

To setup ingress, enable the minikube add-on

```
minikube addons enable ingress
```

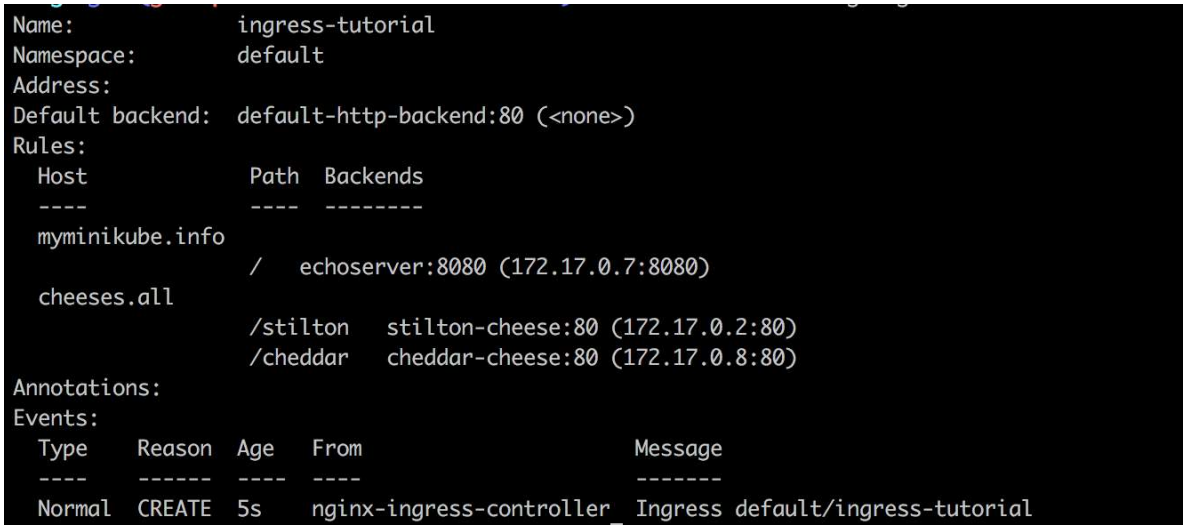
Copy the ingress definition above and save to a file `ingress-tutorial`. Then we create the ingress resource

```
kubectl create -f ingress-tutorial.yaml
```

or create directly from the gist

```
kubectl create -f
https://gist.githubusercontent.com/0sc/77d8925cc378c9a6a92890e
7c08937ca/raw/84ceff5f03da4a2d0a4d2afabd30a1cf3d61fbd1/ingress
-tutorial.yaml
```

You can run `kubectl describe ing ingress-tutorial` for information on the requested ingress.



Now, the last bit is to update our `/etc/hosts` file to route requests from `myminikube.info` and `cheeses.all` to our minikube instance.

Execute

```
echo "$(minikube ip) myminikube.info cheeses.all" | sudo tee -a /etc/hosts
```

to add the following lines to your `/etc/hosts` file.

```
[minikube ip] myminikube.info cheeses.all
```

[minikube ip] *will be replaced with the actual IP of your minikube instance.*

And our work is done.
Test it out by
visiting `myminikube.info` , `cheeses.all/stilton` , `cheeses.all/cheddar` from your browser.