Multi-node Deployment of a Stateless Application on Minikube

This folder contains the configuration for the deployment and services of a web application that exposes everything via Ingress.

Explanation

The deployment creates 2 pod replicas with a Hello-World container. The Service then groups all the pods and exposes them with a ClusterIP. Finally, the Ingress exposes the service externally, always on port 80.
The affinity section prevents the pods from being created on the same node at deployment time. Note that with `requiredDuringSchedulingIgnoredDuringExecution` it is allowed for the pods to end up on the same node at a later time, for example after a restart.

Usage

Prerequisites

- Minikube installed and working
- Kubectl installed and working
- Ingress addon enabled

Starting Minikube

To start Minikube with multiple nodes, run the following command:

```
minikube start --nodes 2 --driver=docker -p multinode-demo
```

A Minikube profile is also created (not sure what it's for, but the tutorial included it). You can verify the presence of the nodes with the command:

```
kubectl get nodes
```

and

```
minikube status -p multinode-demo
```

Deployment

To create the cluster, run the following command:

```
kubectl apply -f manifests/
```

Verifying the Deployment

To check that the deployment was successful, run the following command:

```
kubectl get pods -o wide
```

and use other commands to check services and ingress.

Accessing the Service

To access the service, it is not enough to connect to the cluster IP because it is not exposed.

```
curl  http://192.168.49.2:31000
```

In fact, we need to open the tunnel for the Ingress.

```
minikube tunnel -p multinode-demo
```

After that, you can access the service with the command:

```
curl --resolve "www.example.com:80:127.0.0.1" -i http://www.example.com
```

Cleaning up

To clean up use the `cleanup.sh` in the `scripts` directory:

```
./cleanup.sh
```