Authenticating inside the cluster

This example shows you how to configure a client with client-go to authenticate to the Kubernetes API from an application running inside the Kubernetes cluster.

client-go uses the Service Account token mounted inside the Pod at the `/var/run/secrets/kubernetes.io/serviceaccount` path when the `rest.InClusterConfig()` is used.

Running this example

First compile the application for Linux:

```
cd in-cluster-client-configuration
GOOS=linux go build -o ./app .
```

Then package it to a docker image using the provided Dockerfile to run it on Kubernetes.

If you are running a Minikube cluster, you can build this image directly on the Docker engine of the Minikube node without pushing it to a registry. To build the image on Minikube:

```
eval $(minikube docker-env)
docker build -t in-cluster .
```

If you are not using Minikube, you should build this image and push it to a registry that your Kubernetes cluster can pull from.

If you have RBAC enabled on your cluster, use the following snippet to create role binding which will grant the default service account view permissions.

```
kubectl create clusterrolebinding default-view --clusterrole=view --
serviceaccount=default:default
```

Then, run the image in a Pod with a single instance Deployment:

```
kubectl run --rm -i demo --image=in-cluster

There are 4 pods in the cluster
There are 4 pods in the cluster
There are 4 pods in the cluster
...
```

The example now runs on Kubernetes API and successfully queries the number of pods in the cluster every 10 seconds.

Clean up

To stop this example and clean up the pod, press `Ctrl+C` on the `kubectl run` command and then run:

```
kubectl delete deployment demo
```