Minikube walkthrough

This document will take you through setting up and trying the sample apiserver on a local minikube from a fresh clone of this repo.

Pre requisites

- Go 1.7.x or later installed and setup. More information can be found at go installation
- Dockerhub account to push the image to Dockerhub

Install Minikube

Minikube is a single node Kubernetes cluster that runs on your local machine. The Minikube docs have installation instructions for your OS.

- minikube installation

Clone the repository

In order to build the sample apiserver image we will need to build the apiserver binary.

```
cd $GOPATH/src
mkdir -p k8s.io
cd k8s.io
git clone https://github.com/kubernetes/sample-apiserver.git
```

Build the binary

Next we will want to create a new binary to both test we can build the server and to use for the container image.

From the root of this repo, where `main.go` is located, run the following command:

```
export GOOS=linux; go build .
```

if everything went well, you should have a binary called `sample-apiserver` present in your current directory.

Build the container image

Using the binary we just built, we will now create a Docker image and push it to our Dockerhub registry so that we deploy it to our cluster. There is a sample `Dockerfile` located in `artifacts/simple-image` we will use this to build our own image.

Again from the root of this repo run the following commands:

```
cp ./sample-apiserver ./artifacts/simple-image/kube-sample-apiserver
docker build -t <YOUR_DOCKERHUB_USER>/kube-sample-apiserver:latest ./artifacts/simple-image
docker push <YOUR_DOCKERHUB_USER>/kube-sample-apiserver
```

Modify the replication controller

You need to modify the artifacts/example/deployment.yaml file to change the `imagePullPolicy` to `Always` or `IfNotPresent`.

You also need to change the image from `kube-sample-apiserver:latest` to `<YOUR_DOCKERHUB_USER>/kube-sample-apiserver:latest`. For example:

```
...
      containers:
      - name: wardle-server
        image: <YOUR_DOCKERHUB_USER>/kube-sample-apiserver:latest
        imagePullPolicy: Always
...
```

Save this file and we are then ready to deploy and try out the sample apiserver.

Deploy to Minikube

We will need to create several objects in order to setup the sample apiserver so you will need to ensure you have the `kubectl` tool installed. Install kubectl.

```
# create the namespace to run the apiserver in
kubectl create ns wardle

# create the service account used to run the server
kubectl create -f artifacts/example/sa.yaml -n wardle

# create the rolebindings that allow the service account user to delegate authz back
to the kubernetes master for incoming requests to the apiserver
kubectl create -f artifacts/example/auth-delegator.yaml -n kube-system
kubectl create -f artifacts/example/auth-reader.yaml -n kube-system

# create rbac roles and clusterrolebinding that allow the service account user to use
admission webhooks
kubectl create -f artifacts/example/rbac.yaml
kubectl create -f artifacts/example/rbac-bind.yaml

# create the service and replication controller
kubectl create -f artifacts/example/deployment.yaml -n wardle
kubectl create -f artifacts/example/service.yaml -n wardle

# create the apiservice object that tells kubernetes about your api extension and
where in the cluster the server is located
kubectl create -f artifacts/example/apiservice.yaml
```

Test that your setup has worked

You should now be able to create the resource type `Flunder` which is the resource type registered by the sample apiserver.

```
kubectl create -f artifacts/flunders/01-flunder.yaml
# outputs flunder "my-first-flunder" created
```

You can then get this resource by running:

```
kubectl get flunder my-first-flunder

#outputs
# NAME               KIND
# my-first-flunder   Flunder.v1alpha1.wardle.example.com
```