Kubernetes with Containers and DevOps Workshop

Hands-on lab step-by-step

Aralık 2018

## 4.Deploy application to Azure Kubernetes Service

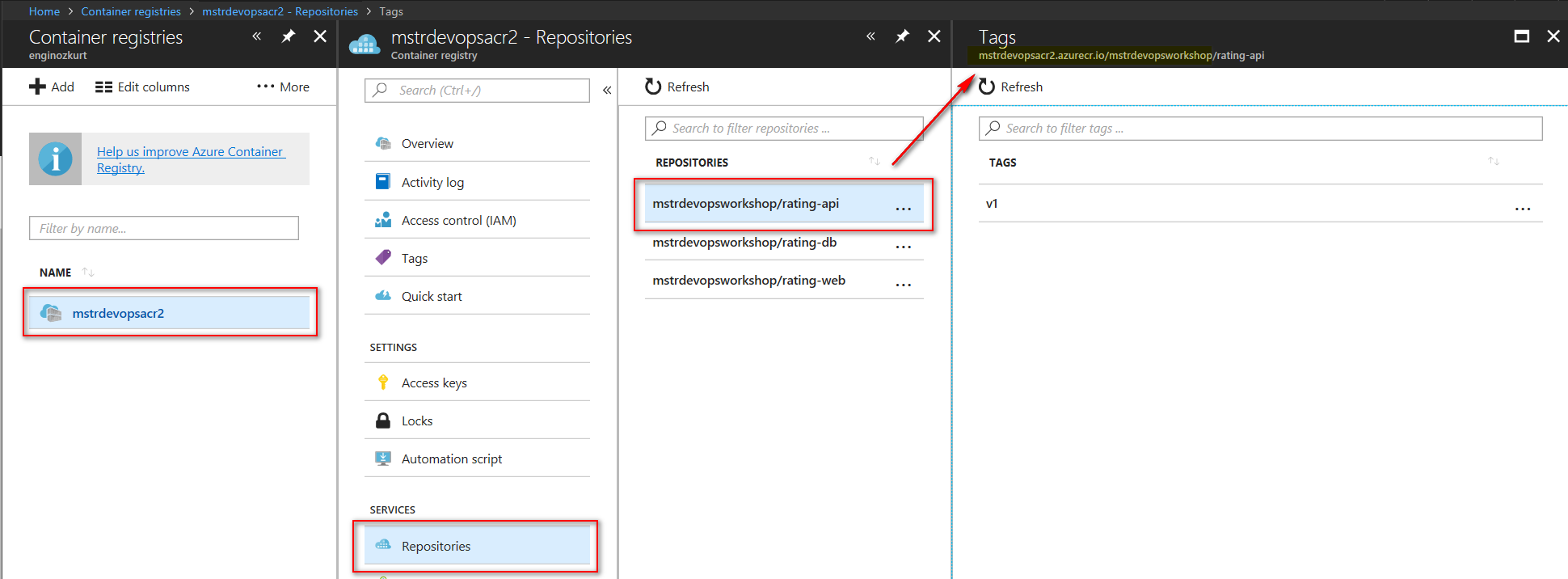
**Review/Edit the YAML Config Files**

* In VS Code (or vi), open the helper-files directory and review the yaml files:

cd ~/cpx-oss-workshop/labs/helper-files



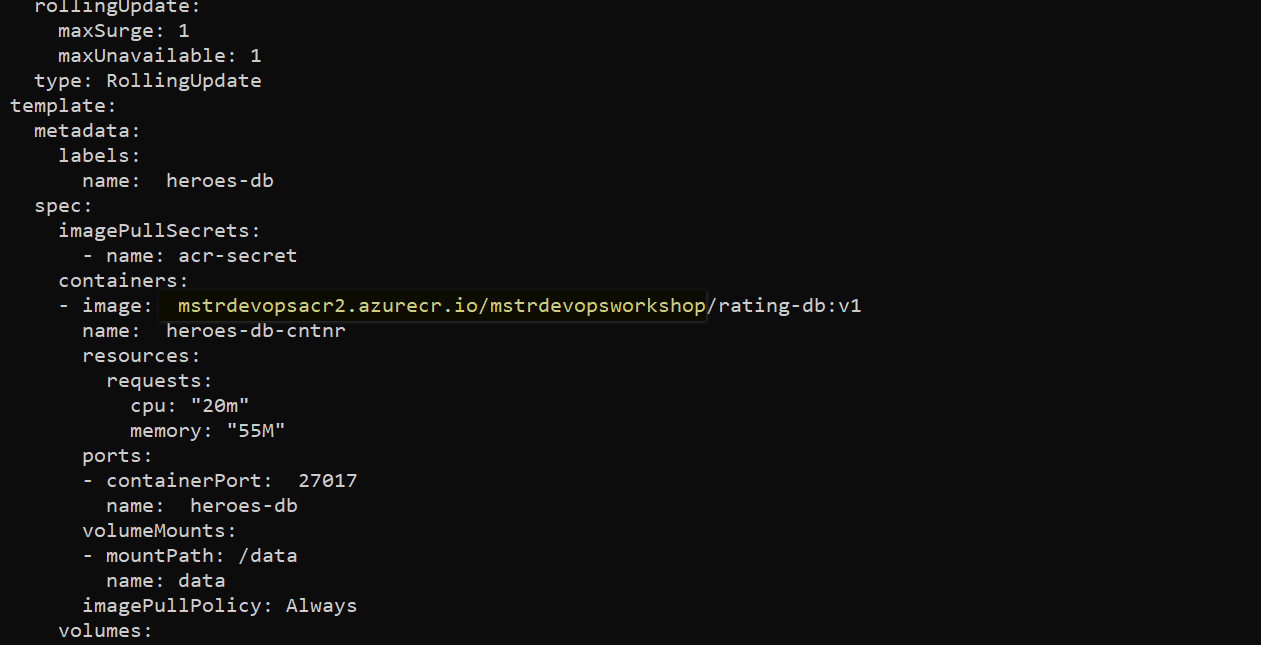
* Note the environment variables that direct each app to other services.
* Update the yaml file for the proper container image names.



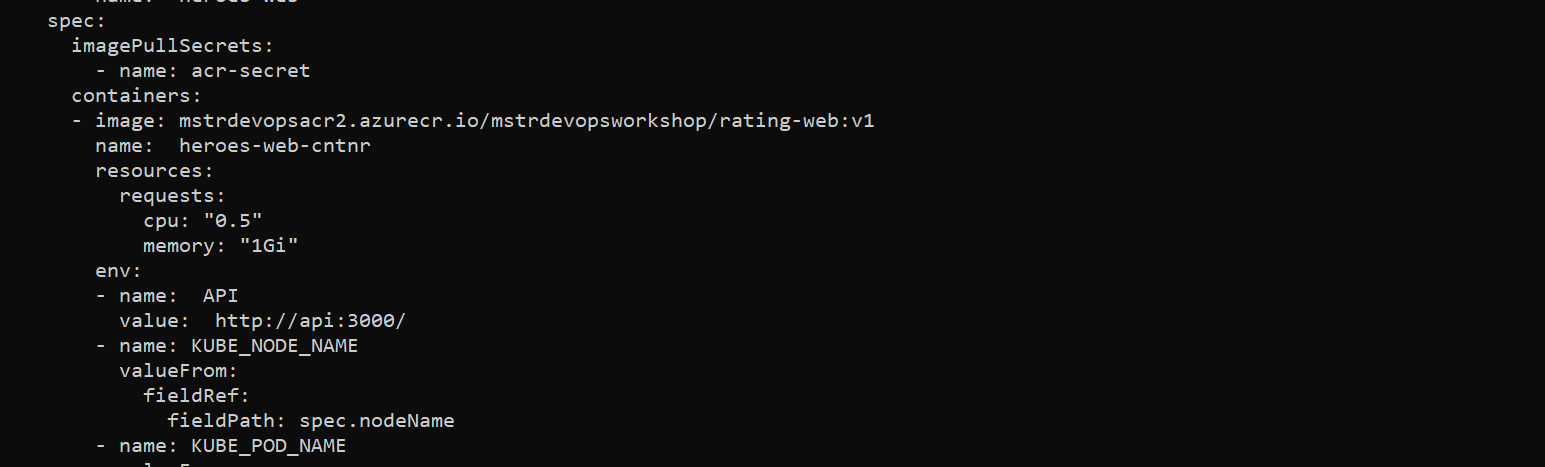
sudo vi heroes-db.yaml

sudo vi heroes-web-api.yaml

* + You will need to replace the <login server> with the ACR login server created in Lab 2.
  + Repeat this **THREE** times in the heroes yaml files (for the web, api, and db images). Example:
  + spec:
  + containers:
  + - image: mstrdevopsacr.azurecr.io/mstrdevopsworkshop/rating-web:v1
  + name: heroes-web-cntnr







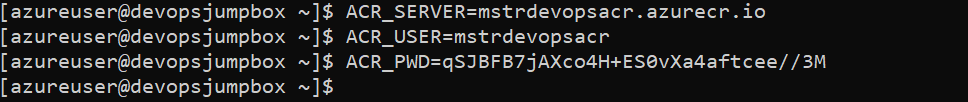
**Setup AKS with access to Azure Container Registry**

There are a few ways that AKS clusters can access your private Azure Container Registry. Generally the service account that kubernetes utilizes will have rights based on its Azure credentials. In our lab config, we must create a secret to allow this access.

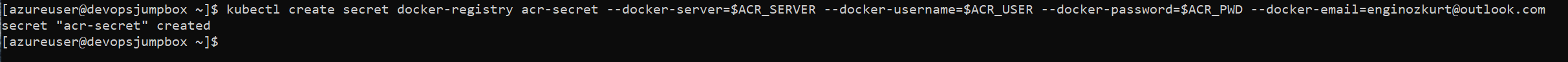
ACR\_SERVER=mstrdevopsacr.azurecr.io

ACR\_USER=mstrdevopsacr

ACR\_PWD=qSJBFB7jAXco4H+ES0vXa4aftcee//3M



kubectl create secret docker-registry acr-secret --docker-server=$ACR\_SERVER --docker-username=$ACR\_USER --docker-password=$ACR\_PWD --docker-email=azureuser@gmail.com

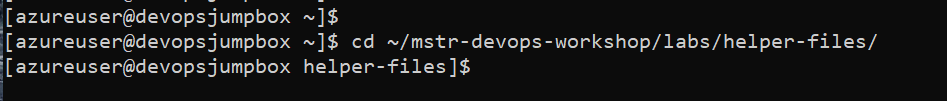


Note: You can review the heroes-db.yaml and heroes-web-api.yaml to see where the imagePullSecrets are configured.

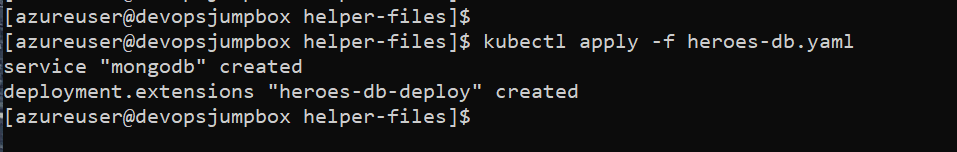
**Deploy database container to AKS**

* Use the kubectl CLI to deploy each app

cd ~/cpx-oss-workshop/labs/helper-files/

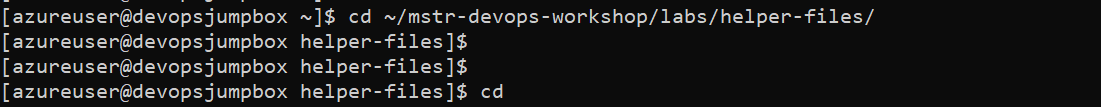


kubectl apply -f heroes-db.yaml

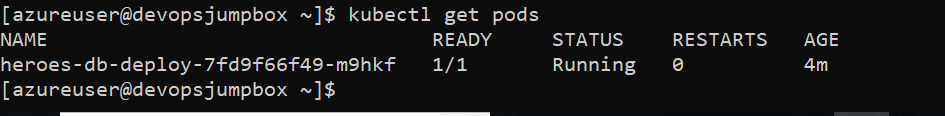


* Get mongodb pod name. Wait your pod to get ready.

cd



kubectl get pods



MONGO\_POD=heroes-db-deploy-2357291595-k7wjk



* Import data into MongoDB using script
* # ensure the pod name variable is set to your pod name
* # once you exec into pod, run the `**import.sh`** script

kubectl exec -it $MONGO\_POD bash

root@heroes-db-deploy-2357291595-xb4xm:/# **./import.sh**

2018-01-16T21:38:44.819+0000 connected to: localhost

2018-01-16T21:38:44.918+0000 imported 4 documents

2018-01-16T21:38:44.927+0000 connected to: localhost

2018-01-16T21:38:45.031+0000 imported 72 documents

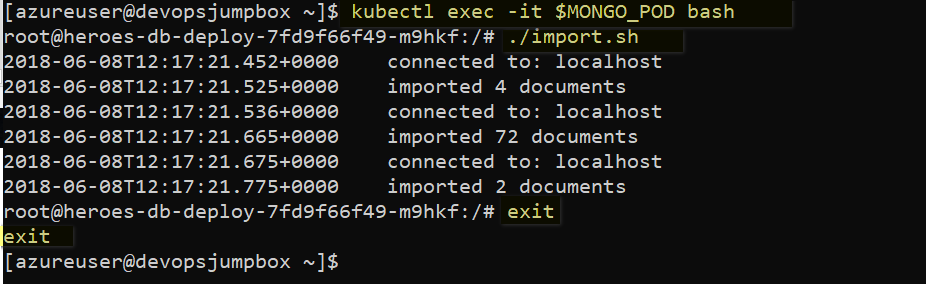
2018-01-16T21:38:45.040+0000 connected to: localhost

2018-01-16T21:38:45.152+0000 imported 2 documents

root@heroes-db-deploy-2357291595-xb4xm:/# exit

# be sure to exit pod as shown above

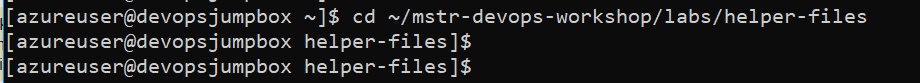
exit



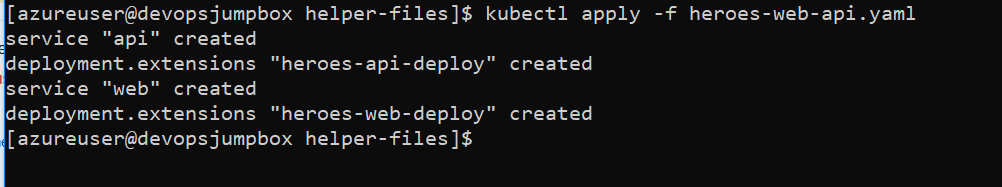
**Deploy the web and api containers to AKS**

* Use the kubectl CLI to deploy each app

cd ~/cpx-oss-workshop/labs/helper-files



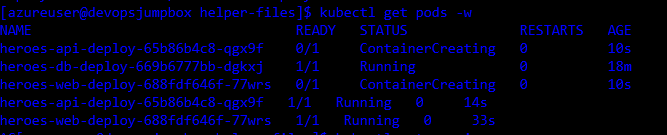
kubectl apply -f heroes-web-api.yaml



**Validate**

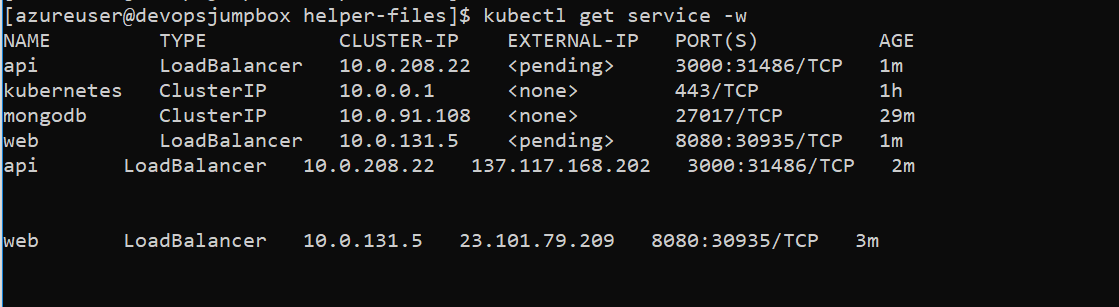
Check to see if pods are running in your cluster

kubectl get pods

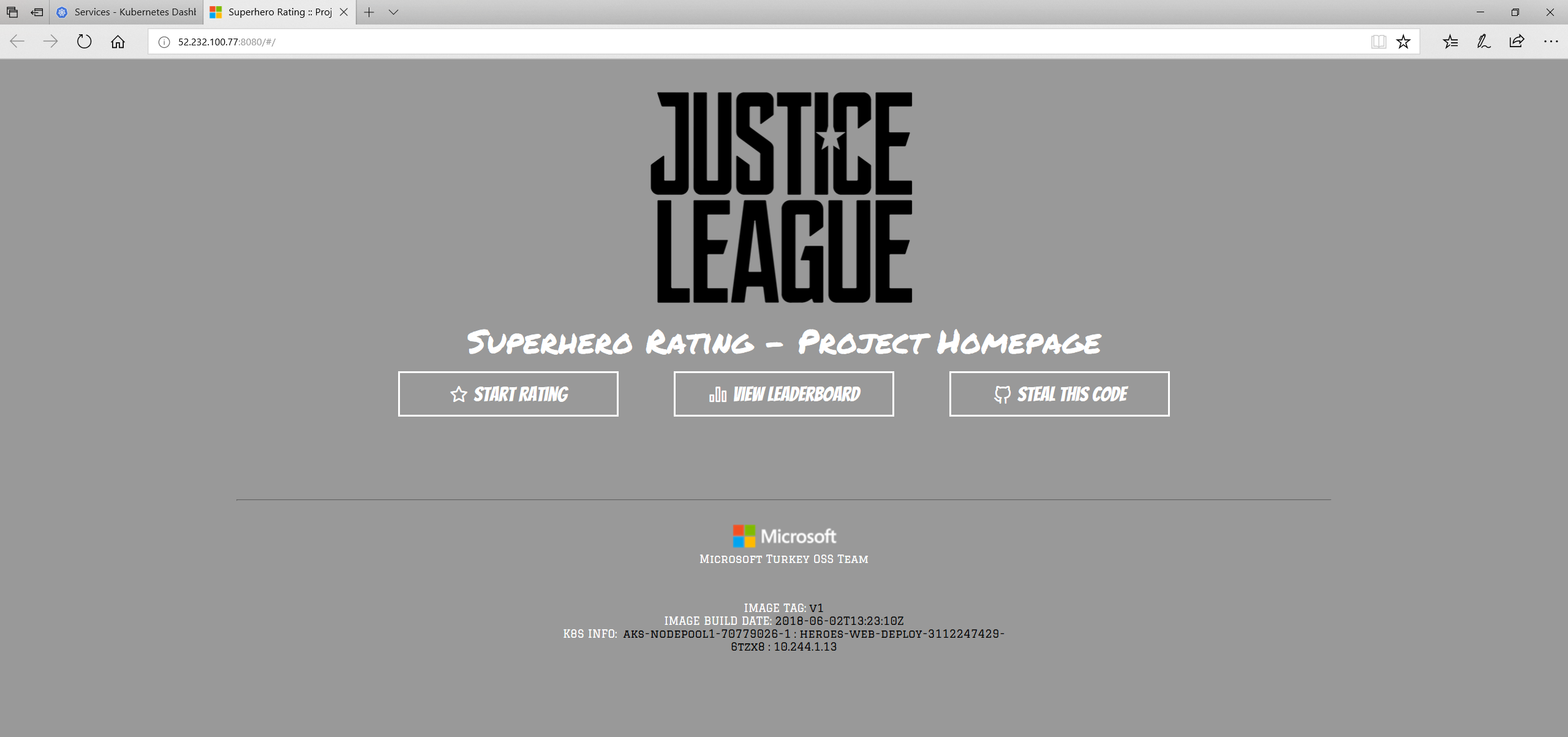


* Check to see if services are deployed.

kubectl get service -w



* Browse to the External IP for your web application (on port 8080) and try the app



The public IP can take a few minutes to create with a new cluster. Sit back and relax. Maybe check twitter.