evernote uuid: 2019-Mar-20-Wed@05:04:11

cd ~/Desktop
git clone git@github.com:flavio-fernandes/knative_demo.git knative_demo.git && cd knative_demo.git
git clone git@github.com:flavio-fernandes/flaskapp.git flaskapp.git

```
cd ~/Desktop/knative_demo.git  ; #0

time vagrant up  ;  # takes about  ~4 minutes

bat Vagrantfile

# take snapshot of vm
vagrant snapshot save freshAndClean
vagrant snapshot list

# so we can easily go back to a fresh clean start  ;  #vsr
vagrant snapshot restore --no-provision freshAndClean
```

Run app in vm

```
cd ~/Desktop/knative_demo.git && vagrant ssh  ; #8
ll

cd /vagrant/flaskapp.git && \
bat src/app.py Dockerfile  ; #zz0

cd /vagrant/flaskapp.git/src && \
TARGET=fromVagrantVM FLASK_DEBUG=1 FLASK_APP=app.py flask run --host 0.0.0.0 --port 8080  ;
 #9

open http://localhost:8080/json
```

Run and build app in docker inside vm

```
cd /vagrant/flaskapp.git && \
docker build -t flaskapp .   ; #1

docker run -e "TARGET=bar" -d --rm -p 8081:5000 --name flaskapp flaskapp  ; #2
docker ps  ; #dps
docker logs --follow flaskapp  ; #3

open http://localhost:8081

cd ~/Desktop/knative_demo.git && vagrant ssh  ; #8
curl http://localhost:8081/json?delay=2000  ; #zz1

# get inside running container
docker exec -it flaskapp /bin/sh  ; #4
netstat -plnt  ; #5
top
```

```
exit

docker stop flaskapp  ; #6
```

Auth with Google Cloud

```
cat << EOT >> /home/vagrant/.bashrc_me

export PROJECT=flaviof-knative
export CLUSTER_NAME=knative1
export CLUSTER_ZONE=us-east1-d
EOT
source /home/vagrant/.bashrc_me ; echo $PROJECT   ; #z1

gcloud auth login  && \
gcloud auth application-default login   ; #z2

# if need to create project:
# gcloud init
# gcloud projects create $PROJECT --set-as-default
# enable billing
# open https://console.cloud.google.com/

echo $PROJECT ; gcloud config set core/project $PROJECT && \
gcloud config set compute/zone $CLUSTER_ZONE    ; #z3

# enable services
gcloud services enable cloudapis.googleapis.com && \
gcloud services enable container.googleapis.com && \
gcloud services enable containerregistry.googleapis.com && \
echo ok    ; #z4

gcloud auth configure-docker --project $PROJECT --quiet  ; #zzd

# no $CLUSTER_NAME yet
gcloud projects list && gcloud container clusters list  ; #gpl
```

Create cluster in GKE

```
CLUSTER_VERSION='latest' ; \
time gcloud container clusters create $CLUSTER_NAME \
--zone=$CLUSTER_ZONE \
--cluster-version=${CLUSTER_VERSION} \
--machine-type=n1-standard-4 \
--enable-autoscaling --min-nodes=1 --max-nodes=10 \
--enable-autorepair \
--scopes=service-control,service-management,compute-rw,storage-ro,cloud-platform,logging-
write,monitoring-write,pubsub,datastore \
--num-nodes=3   ; #z5

# zzzz about 3 minutes... back to slides
```

```
gcloud projects list && gcloud container clusters list  ; #gpl

# get the credentials
grep --quiet "gcloud container clusters get-credentials" /home/vagrant/.bashrc_me || \
cat << EOT >> /home/vagrant/.bashrc_me

gcloud container clusters get-credentials ${CLUSTER_NAME} --zone ${CLUSTER_ZONE} --project
${PROJECT}
EOT
gcloud container clusters get-credentials ${CLUSTER_NAME} --zone ${CLUSTER_ZONE} --project
${PROJECT}  ; #z6

# Set rbac.authorization
kubectl create clusterrolebinding cluster-admin-binding \
--clusterrole=cluster-admin \
--user=$(gcloud config get-value core/account)  ; #z7
```

Install gloo (instead of istio)
Ref: https://www.knative.dev/docs/install/knative-with-gloo/

```
glooctl --version && time glooctl install knative && echo ok  ; #z8

kubectl get pods --namespace gloo-system ; \
kubectl get pods --namespace knative-serving ; #z9

# wait for external ip to be available from load balancer
kubectl get services --all-namespaces   ;  #kcs

grep --quiet CLUSTERINGRESS_URL /home/vagrant/.bashrc_me || \
echo 'export CLUSTERINGRESS_URL=$(glooctl proxy url --name clusteringress-proxy)'
>> /home/vagrant/.bashrc_me ;
[ -z "$CLUSTERINGRESS_URL" ] && source /home/vagrant/.bashrc_me ; \
echo "CLUSTERINGRESS_URL: $CLUSTERINGRESS_URL" ; #za


# Map load balancer public ip to my dynamic DNS via postman
```

Pushing foo app into google docker registry

```
open https://console.cloud.google.com/gcr/images/flaviof-knative?project=flaviof-knative

cd /vagrant/flaskapp.git/ && docker build -t foo .  ; #zb

# gcr.io/flaviof-knative/foo:latest
docker tag foo gcr.io/${PROJECT}/foo:latest && \
docker push gcr.io/${PROJECT}/foo:latest  ; #zc
```

Deploying it simple app on k8 (outside knative) -- SO BORING!!! :^)

```
cd /vagrant/k8s && bat foo.yaml  ; #zd

# Start watching pods and service on default namespace
watch kubectl get pod,service  ; #kw
```

```
kubens default && \
kubectl create -f foo.yaml    ; #ze

kubectl expose deployment foo-example --type LoadBalancer --port 80 --target-port 5000  ;
#zf

# hit foo from inside cluster
kubectl run -i --tty --rm ubuntu --image=ubuntu:16.04 --restart=Never -- bash -il ; #kuu

apt-get update >/dev/null 2>&1 && apt install --quiet -y curl >/dev/null 2>&1 && echo ok  ;
#zh

export CLUSTER_IP=${CLUSTER_IP:-X} ; export PORT=${PORT:-8080} ; #zi  [clipboard]

curl http://${CLUSTER_IP}:${PORT}/json -H 'cache-control: no-cache'  ; #zj

# break it
curl http://${CLUSTER_IP}:${PORT}/json?boom -H 'cache-control: no-cache'  ; # zzk

while : ; do curl http://${CLUSTER_IP}:${PORT}/json?delay=750 -H 'cache-control: no-cache' ;
done ; #zk
#exit

# clean up
cd /vagrant/k8s && { \
kubectl delete -f foo.yaml ; \
kubectl delete service foo-example --ignore-not-found=true ; \
kubectl delete service foo-np --ignore-not-found=true; } ;  #zl
```

Deploying it simple, but using knative way

```
cd /vagrant/k8s && bat foo.yaml foo-knative-1.yaml ; #zm

kubectl apply -f foo-knative-1.yaml    ; #zn

printf "ksvc\n" ; kubectl get ksvc ; \
printf "\nroutes\n" ; kubectl get routes.serving.knative.dev ; \
printf "\nconfiguration\n" ; kubectl get configurations.serving.knative.dev  ; #zu

curl -v -H "Host: foo-example-knative.default.example.com" ${CLUSTERINGRESS_URL}/json  ; #zp
http ${CLUSTERINGRESS_URL}/json 'Host:foo-example-knative.default.example.com' -v -s fruity
 ; #zq

# do it from postman
glooctl proxy url --name clusteringress-proxy ; \
dig +short foo.flaviof.dev ; # ki

# clean up... or not :^)
cd /vagrant/k8s && \
kubectl delete -f foo-knative-1.yaml    ; #zr
```

Blue green

ref: https://www.knative.dev/docs/serving/samples/blue-green-deployment/

```
cd /vagrant/k8s && bat blue-green-demo-1.yaml  ; #zs

kubectl apply -f blue-green-demo-1.yaml  ; #zt

printf "ksvc\n" ; kubectl get ksvc ; \
printf "\nroutes\n" ; kubectl get routes.serving.knative.dev ; \
printf "\nconfiguration\n" ; kubectl get configurations.serving.knative.dev  ; #zu

# open a window that grabs the color in a loop
while : ; do \
curl --silent -X GET http://foo.flaviof.dev/ \
  -H 'Host: blue-green-demo.default.example.com' \
  -H 'cache-control: no-cache' | grep 'div class=' ; \
sleep 1 ; \
done  ; #zv

# introduce green as v2
cd /vagrant/k8s && diff blue-green-demo-{1,2}.yaml | bat -l diff ; \
bat blue-green-demo-2.yaml ; #zw

kubectl apply -f blue-green-demo-2.yaml  ; #zx


# via postman, get green, even though the routes still point 100% blue,
# by using v2 in the Host header
# also show that v2 can be reached if asked explicitly

# show revision 000002 via  #zu (above)

# go 50/50
grep -B 2 -A 99 'kind: Route' blue-green-demo-2.yaml > /tmp/blue-green-demo-2Route.yaml ; \
diff /tmp/blue-green-demo-2Route.yaml blue-green-demo-3.yaml | bat -l diff ; \
bat blue-green-demo-3.yaml  ; #K1

kubectl apply -f blue-green-demo-3.yaml  ; #K2

# go 100 green

diff blue-green-demo-{3,4}.yaml | bat -l diff ; \
bat blue-green-demo-4.yaml ;  #K3
kubectl apply -f blue-green-demo-4.yaml ;  #K4


# cleanup blue and green kservice and route
cd /vagrant/k8s && kubectl delete -f blue-green-demo-2.yaml  ;  # K5
```

Auto build

ref: https://github.com/knative/build-templates/tree/master/kaniko
https://www.knative.dev/docs/serving/samples/source-to-url-go/
https://boxboat.com/2018/08/10/building-containers-with-kubernetes-and-knative/
https://content.pivotal.io/practitioners/knative-whittling-down-the-code

```
Adding the Knative Build component
# ref https://www.knative.dev/docs/build/installing-build-component/
kubectl apply --filename https://github.com/knative/build/releases/download/v0.4.0
/build.yaml  ;  #K6

kubectl get pods --namespace knative-build --watch  ; #K7

# Install the kaniko build template
kubectl apply --filename https://raw.githubusercontent.com/knative/build-templates/master
/kaniko/kaniko.yaml  ;  # K8

kubectl get buildtemplates.build.knative.dev   ;  #K9

# Populate k8s/autodeploy/docker-secret.yaml if needed

cd /vagrant/k8s/autodeploy && \
kubectl apply -f docker-secret.yaml && \
kubectl apply -f service-account.yaml  ;  #Ka

bat docker-secret.yaml.example && \
bat service-account.yaml && \
bat build.yaml.standalone ;   #Kb

#kubectl apply -f build.yaml
#kubectl get builds kaniko-build -oyaml
#kubectl delete build kaniko-build

cd /vagrant/k8s && \
diff -u foo-knative-{1,2}.yaml | bat -l diff && \
bat foo-knative-2.yaml  ;  #Kc

cd /vagrant/flaskapp.git && \
git checkout knativeDemo && \
emacs Dockerfile src/app.py  ;  #Kd

git commit -a -m 'changed something' && \
git push  ; #Ke

cd /vagrant/k8s && \
kubectl apply -f foo-knative-2.yaml  ; #Kf

kubectl get builds foo-example-knative-00001 -oyaml | bat -l yaml  ; #Kg

kubectl get ksvc  ; #Kh

check image at: https://cloud.docker.com/u/flaviofdocker/repository/docker/flaviofdocker/foo

http ${CLUSTERINGRESS_URL}/json 'Host:foo-example-knative.default.example.com' -v -s
fruity  ; #zq

# Make a new Revision, using image in google cloud
kubectl apply -f foo-knative-1.yaml  ; #zn

# grab foo output via postman again or via
```

```
curl -v -H "Host: foo-example-knative.default.example.com" ${CLUSTERINGRESS_URL}/json  ; #zq

kubectl get ksvc  ; #Kh

# jump back to image built as revision 000003
kubectl apply -f foo-knative-2.yaml  ; #Kf

kubectl delete -f foo-knative-2.yaml  ; #Ki
```

cleanup
blow cluster away

```
time gcloud container clusters delete $CLUSTER_NAME --zone $CLUSTER_ZONE   ; #KZ
# about 3.5 minutes
```