jotak Better step-by-step with comments

6dcae8e on 25 Jul

1 contributor

251 lines (177 sloc) 6.82 KB

Mesh Arena

This is a step-by-step guide to run the demo.

Slides

This demo was presented at DevopsDday in the Velodrome, Marseilles' famous stadium, and then at RivieraDev 2019. Check the slides, in French (a bit outdated) or in English.

Pre-requisite

- Kubernetes or OpenShift cluster running (ex: minikube 0.27+ / minishift)
- Istio with Kiali installed

Example of Istio + Kiali install:

```
curl -L https://git.io/getLatestIstio | ISTIO_VERSION=1.1.5 sh -
# Don't forget to export istio-1.1.5/bin to your path (as said in terminal output)
cd istio-1.1.5
for i in install/kubernetes/helm/istio-init/files/crd*yaml; do kubectl apply -f $i; done
kubectl apply -f install/kubernetes/istio-demo.yaml

# Remove old Kiali:
kubectl delete deployment kiali-operator -n kiali-operator
kubectl delete deployment kiali -n istio-system

bash <(curl -L https://git.io/getLatestKialiOperator)
```

In a new terminal, you can forward Kiali's route:

```
kubectl port-forward svc/kiali 20001:20001 -n istio-system
```

Open https://localhost:20001/kiali

(Might be an insecure connection / invalid certificate, to allow in Chrome go to chrome://flags/#allow-insecure-localhost)

Install dashboards

From there: https://github.com/kiali/kiali/tree/master/operator/roles/kiali-deploy/templates/dashboards

```
kubectl apply -f dashboards
```

Get the yml files locally

- Clone this repo locally, `cd` to it.

```
git clone git@github.com:jotak/demo-mesh-arena.git
cd demo-mesh-arena
```

For OpenShift users, you may have to grant extended permissions for Istio, logged as admin:

```
oc new-project mesh-arena
oc adm policy add-scc-to-user privileged -z default
```

Jaeger

Tracing data generated from microservices and Istio can be viewed in Jaeger by port-forwarding `jaeger-query` service.

```
kubectl port-forward svc/jaeger-query 16686:16686 -n istio-system
```

AI service generates trace named `new_game` for each game. This way we are able to trace player's movement on the stadium.

The other interesting trace is from `ui` service called `on-start` it captures all initialization steps performed at the beginning of the game.

Deploy microservice UI

```
kubectl apply -f <(istioctl kube-inject -f ./services/ui/Deployment.yml)
kubectl create -f ./services/ui/Service.yml
kubectl apply -f mesh-arena-gateway.yaml
```

Open in browser

(Wait a little bit because port-forward?)

```
kubectl port-forward svc/istio-ingressgateway 8080:80 -n istio-system
```

Open http://localhost:8080 in a browser.

Deploy stadium & ball

```
kubectl apply -f <(istioctl kube-inject -f ./services/stadium/Deployment-Smaller.yml)
kubectl create -f ./services/stadium/Service.yml
kubectl apply -f <(istioctl kube-inject -f ./services/ball/Deployment.yml)
kubectl create -f ./services/ball/Service.yml
```

Deploy 2x2 players

```
kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-2-locals.yml)
kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-2-visitors.yml)
kubectl create -f ./services/ai/Service.yml
```

▶ Kiali TIP

Second ball

```
kubectl apply -f <(istioctl kube-inject -f ./services/ball/Deployment-v2.yml)
```

In this state, the usual K8S load balancer is in use. Players can't decide whether to go to ball v1 or v2.

▶ Kiali TIP

Ponderate ball v1 and v2

```
kubectl apply -f ./services/ball/destrule.yml
kubectl apply -f ./services/ball/virtualservice-75-25.yml
```

Players know a little bit better where to go, but still unsure.

▶ Kiali TIP

Messi / Mbappé

```
kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-Messi.yml)
kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-Mbappe.yml)
```

Two new players.

Each his ball

```
kubectl apply -f ./services/ball/virtualservice-by-label.yml
```

Now they know. Clean state.

▶ Kiali TIP

Reset

```
kubectl delete -f ./services/ai/Deployment-Messi.yml
kubectl delete -f ./services/ai/Deployment-Mbappe.yml
kubectl delete -f ./services/ball/virtualservice-by-label.yml
kubectl delete -f ./services/ball/Deployment-v2.yml
```

Burst ball (500 errors) with shadowing

```
kubectl apply -f ./services/ball/virtualservice-mirrored.yml
kubectl apply -f <(istioctl kube-inject -f ./services/ball/Deployment-burst.yml)
```

A new ball, v2 is deployed "for fake": all requests sent to v1 are duplicated to v2. But the players don't "know" about that: from their PoV their requests are for v1 only. They don't get responses from v2.

The new ball sometimes (randomly) returns errors. When it does so, it turns red.

▶ Kiali TIP

Remove shadowing, put circuit breaking

```
kubectl delete -f ./services/ball/virtualservice-mirrored.yml
kubectl apply -f ./services/ball/destrule-outlier.yml
```

CB is configured to evict failing workload for 10s upon error. Then it's put back into the LB pool, and will be evicted again, and again, and again.

When a ball receives no request, it turns darker. So it's grey when it's evicted by the circuit breaker.

▶ Kiali TIP

To clean up everything

```
kubectl delete deployments -l project=mesh-arena
kubectl delete svc -l project=mesh-arena
kubectl delete virtualservices -l project=mesh-arena
kubectl delete destinationrules -l project=mesh-arena
```