Rate Limiter

Kubernetes Deployment Guide

Prerequisite:

1. Docker
2. Kubernetes Cluster
3. Kubectl configured
4. Rate Limiter Docker Image (nikaris/rate-limiter:latest)

Deployment Step

1. Create a new namespace named ‘rate’ and all next deployment we will be using this namespace

```
kubectl create namespace rate
```

2. Deploy Redis with statefulset and its Service.

```
kubectl apply -f redis-statefulset.yml -n rate
kubectl apply -f redis-service.yml
```

3. Create Configmap and Secret for App Config

```
kubectl apply -f environment-config.yml -n rate
kubectl apply -f secret-redis.yml -n rate
```

4. Deploy Rate Limiter App and its Service.

```
kubectl apply -f rate-limiter-app.yml -n rate
kubectl apply -f rate-service.yml -n rate
```

Interact with the Rate Limiter App

- If you are using minicube, run:

```
minikube service rate-limiter-service
```

- Otherwise, access it at:

```
http://<NodeIP>:30110/<endpoint>
```

example:

```
http://localhost:30110/home
```

Configuration for Rate Limiter

User can adjust the **RATE_LIMIT** and **EXPIRE_TIME** to be their desire value in ‘environment-config.yml’ file.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: environment-config
data:
  RATE_LIMIT: "3"
  EXPIRE_TIME: "5"
```

After changes been made, run:

```
kubectl apply -f environment-config.yml -n rate
kubectl rollout restart deployment -n rate rate-limiter-app
```

If there is no value for RATE_LIMIT and EXPIRE_TIME, the value will be the default value set by application itself as below.

EXPIRE_TIME = 1
RATE_LIMIT = 5

Test the Rate Limiter

1. Run the curl 10 times:

```
# Simulate requests from 192.168.1.123 to /health
for i in {1..10}; do
  curl -H "X-Forwarded-For: 192.168.1.123" -i http://localhost:30110/health
done
```

   when it exceeds RATE_LIMIT, you will get response HTTP 429 TOO MANY REQUESTS with message ‘rate limit exceeded’

2. Run manually:

```
curl -H "X-Forwarded-For: 192.168.1.101" -i http://localhost:30110/health
```

Simulate App Recovery

- Change the EXPIRE_TIME in environment-config.yml to 60.
- Apply changes and restart the application deployment.
- Since the limit is 3, try to curl one time to see if the redis retain it memory:

```
curl -H "X-Forwarded-For: 192.168.1.101" -i http://localhost:30110/health
```

- Delete the pod, it will restart by kubernetes

```
kubectl delete pod -n rate -l app=rate-limiter-app
```

- Retest using the same ip, if it ‘rate limited’ on the 3rd attempt, mean the app recover successfully

```
curl -H "X-Forwarded-For: 192.168.1.101" -i http://localhost:30110/health
```

Inspect Redis State

```
kubectl exec -it  -n rate redis-0 -- redis-cli

> keys *
```