Istio Timeout & Retry

Istio VirtualService 객체를 통해 대상 서비스에 Timeout과 Retry를 설정하고, 정상 동작하는지 확인한다.

Timeout - 카프카 설치

```
helm repo add incubator https://charts.helm.sh/incubator
helm repo update
kubectl create namespace kafka
helm install my-kafka --namespace kafka incubator/kafka
```

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry# helm repo add incubator https://charts.helm.sh/incubator
"incubator" has been added to your repositories
root@labs-910775232:/home/project/Istio-Timeout-and-Retry# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "incubator" chart repository
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈Happy Helming!⎈
root@labs-910775232:/home/project/Istio-Timeout-and-Retry# kubectl create namespace kafka
namespace/kafka created
root@labs-910775232:/home/project/Istio-Timeout-and-Retry# helm install my-kafka --namespace kafka incubator/kafka
WARNING: This chart is deprecated
W0318 06:35:09.220523  198069 warnings.go:70] policy/v1beta1 PodDisruptionBudget is deprecated in v1.21+, unavailable in v1.25+; use policy/v1 PodDisruptionBudget
W0318 06:35:10.231433  198069 warnings.go:70] policy/v1beta1 PodDisruptionBudget is deprecated in v1.21+, unavailable in v1.25+; use policy/v1 PodDisruptionBudget
NAME: my-kafka
LAST DEPLOYED: Fri Mar 18 06:35:08 2022
NAMESPACE: kafka
STATUS: deployed
REVISION: 1
NOTES:
### Connecting to Kafka from inside Kubernetes

You can connect to Kafka by running a simple pod in the K8s cluster like this with a configuration like this:

  apiVersion: v1
  kind: Pod
  metadata:
    name: testclient
    namespace: kafka
  spec:
    containers:
    - name: kafka
      image: confluentinc/cp-kafka:5.0.1
      command:
        - sh
        - -c
        - "exec tail -f /dev/null"

Once you have the testclient pod above running, you can list all kafka
topics with:

  kubectl -n kafka exec testclient -- ./bin/kafka-topics.sh --zookeeper my-kafka-zookeeper:2181 --list
```

kubectl get svc my-kafka -n kafka

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry# kubectl get svc my-kafka -n kafka
NAME       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
my-kafka   ClusterIP   10.100.103.70   <none>        9092/TCP   52s
```

tutorial 네임스페이스에 Istio Activation

네임스페이스가 없을 시, 생성 후 실행

kubectl label namespace tutorial istio-injection=enabled --overwrite

```
Every 1.0s: kubectl get all -n kafka

NAME                         READY   STATUS    RESTARTS   AGE
pod/my-kafka-0               0/1     Pending   0          2m57s
pod/my-kafka-zookeeper-0     1/1     Running   0          2m57s
pod/my-kafka-zookeeper-1     1/1     Running   0          117s
pod/my-kafka-zookeeper-2     1/1     Running   0          87s

NAME                                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/my-kafka                    ClusterIP   10.100.103.70    <none>        9092/TCP                     2m57s
service/my-kafka-headless           ClusterIP   None             <none>        9092/TCP                     2m57s
service/my-kafka-zookeeper          ClusterIP   10.100.187.145   <none>        2181/TCP                     2m57s
service/my-kafka-zookeeper-headless ClusterIP   None             <none>        2181/TCP,3888/TCP,2888/TCP   2m57s

NAME                                READY   AGE
statefulset.apps/my-kafka           0/3     2m57s
statefulset.apps/my-kafka-zookeeper 3/3     2m57s
```

Timeout : Fail-Fast를 통한 Caller 자원 보호(장애전파 차단)

- Order Aggregate(Order.java)에 저장전 Thread.sleep 삽입

```
@PrePersist
public void onPrePersist(){
        try {
                Thread.currentThread().sleep((long) (800 + Math.random() * 220));
        } catch (InterruptedException e) {
                e.printStackTrace();
        }
}
```

Dockerizing (Image Build, and Push)

ECR 레파지토리 URI
979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order

- Order 프로젝트 루트로 콘솔 이동

```
mvn package
docker build -t [IMAGE_NAME] .
docker push [IMAGE_NAME]
```

docker build -t 979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order:latest .

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry/order# docker build -t 979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order:latest .
Sending build context to Docker daemon 59.83 MB
Step 1/4 : FROM openjdk:8u212-jdk-alpine
 ---> a3562aa0b991
Step 2/4 : COPY target/*SNAPSHOT.jar app.jar
 ---> Using cache
 ---> 48321b235347
Step 3/4 : EXPOSE 8080
 ---> Using cache
 ---> 964284c9b22d
Step 4/4 : ENTRYPOINT ["java","-Xmx400M","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar","--spring.profiles.active=docker"]
 ---> Using cache
 ---> b23480138a82
Successfully built b23480138a82
Successfully tagged 979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order:latest
```

docker push 979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order:latest

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry/order# docker push 979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order:latest
The push refers to repository [979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order]
58e6bdc89baa: Pushed
ceaf9e1ebef5: Pushed
9b9b7f3d56a0: Pushed
f1b5933fe4b5: Pushed
latest: digest: sha256:079b87ae2a46219caf08e381d63e76233bea36ec5552f428c7d2e8a12e8e9e8e size: 1159
```

tutorial 네임스페이스에 Order 배포 command

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry/order# kubectl apply -f - <<EOF
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: order
    namespace: tutorial
    labels:
      app: order
  spec:
    replicas: 1
    selector:
      matchLabels:
        app: order
    template:
      metadata:
        labels:
          app: order
      spec:
        containers:
          - name: order
            image: 979050235289.dkr.ecr.ap-northeast-2.amazonaws.com/user003003-order
            ports:
              - containerPort: 8080
            resources:
              limits:
                cpu: 500m
              requests:
                cpu: 200m
EOF

deployment.apps/order configured
```

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry/order# kubectl get pod -n tutorial
NAME                                  READY   STATUS    RESTARTS   AGE
customer-7894f979bb-rff9h             2/2     Running   0          101m
order-cc74866c-4v76q                  2/2     Running   0          20s
preference-v1-dfb576f75-xm8v2         2/2     Running   0          98m
recommendation-v1-7f65d67d4d-rrtw9    2/2     Running   0          98m
recommendation-v2-6f8c6975cc-9rcwr    2/2     Running   0          97m
recommendation-v2-6f8c6975cc-g8mbt    2/2     Running   0          84m
```

Order 서비스 생성

kubectl expose deploy order --port=8080 -n tutorial

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry/order# kubectl expose deploy order --port=8080 -n tutorial
service/order exposed
```

주문서비스 Timeout이 설정된 VritualService생성

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry/order# kubectl apply -f - <<EOF
>     apiVersion: networking.istio.io/v1alpha3
>     kind: VirtualService
>     metadata:
>       name: vs-order-network-rule
>       namespace: tutorial
>     spec:
>       hosts:
>       - order
>       http:
>       - route:
>         - destination:
>             host: order
>         timeout: 3s
> EOF
virtualservice.networking.istio.io/vs-order-network-rule created
```

Siege를 통한 Order 서비스 부하 주입

kubectl run siege --image=apexacme/siege-nginx -n tutorial

kubectl exec -it pod/siege-d484db9c-m8ktq -c siege -n tutorial -- /bin/bash

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry# kubectl exec -it pod/siege-d484db9c-m8ktq -c siege -n tutorial -- /bin/bash
root@siege-d484db9c-m8ktq:/#
```

siege -c30 -t20S -v --content-type "application/json" 'http://order:8080/orders POST {"productId": "1001", "qty":5}'

```
root@labs-910775232:/home/project/Istio-Timeout-and-Retry# kubectl exec -it pod/siege-d484db9c-m8ktq -c siege -n tutorial -- /bin/bash
root@siege-d484db9c-m8ktq:/# siege -c30 -t20S -v --content-type "application/json" 'http://order:8080/orders POST {"productId": "1001", "qty":5}'
** SIEGE 4.0.4
** Preparing 30 concurrent users for battle.
The server is now under siege...
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.04 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.05
```

```
HTTP/1.1 504     3.00 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.00 secs:      24 bytes ==> POST http://order:8080/orders
HTTP/1.1 201     1.06 secs:     214 bytes ==> POST http://order:8080/orders
HTTP/1.1 504     3.00 secs:      24 bytes ==> POST http://order:8080/orders

Lifting the server siege...
Transactions:                      8 hits
Availability:                   4.35 %
Elapsed time:                  19.21 secs
Data transferred:               0.01 MB
Response time:                 67.48 secs
Transaction rate:               0.42 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                   28.10
Successful transactions:           8
Failed transactions:             176
Longest transaction:            3.05
Shortest transaction:           0.91

root@siege-d484db9c-m8ktq:/#
```

•*Order 서비스에 설정된 Timeout 임계치를 초과하는 순간, Istio에서 연결을 차단하는 것을 확인*