Fault Injection:

In this lab we will learn how to test the resiliency of an application by injecting systematic faults.

Before we start let us reset the route rules:

```
kubectl apply -f virtual-service-all-v1.yaml
```

Update the USER_NAME placeholder with your user name in `virtual-service-reviews-test-v2.yaml`:

```
vi virtual-service-reviews-test-v2.yaml
```

Then apply changes:

```
kubectl apply -f virtual-service-reviews-test-v2.yaml
```

Inject a route rule to create a fault using HTTP delay:

To start, we will inject a 7s delay between the `reviews v2` and `ratings` service for your user. reviews v2 service has a 10s hard-coded connection timeout for its calls to the ratings service.

```
vi virtual-service-ratings-test-delay.yaml
```

Then apply changes:

```
kubectl apply -f virtual-service-ratings-test-delay.yaml
```

To confirm the rule is in place:

```
kubectl get virtualservice ratings -o yaml
```

Output:

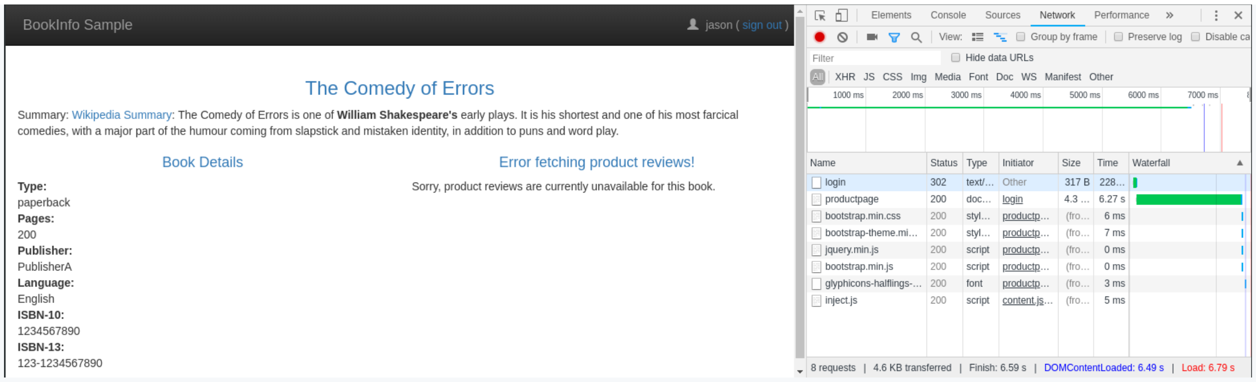
```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
{"apiVersion":"networking.istio.io/v1alpha3","kind":"VirtualService","metadata":{"ann
otations":{},"name":"ratings","namespace":"default"},"spec":{"hosts":["ratings"],"htt
p":[{"fault":{"delay":{"fixedDelay":"7s","percent":100}},"match":[{"headers":{"end-
user":{"exact":"USER_NAME"}}}],"route":[{"destination":{"host":"ratings","subset":"v1
"}}]},{"route":[{"destination":{"host":"ratings","subset":"v1"}}]}]}}
  creationTimestamp: 2018-10-26T15:21:42Z
  generation: 1
  name: ratings
  namespace: default
  resourceVersion: "14790"
  selfLink:
/apis/networking.istio.io/v1alpha3/namespaces/default/virtualservices/ratings
  uid: d7d7347f-d932-11e8-88c5-0242f983c5dd
spec:
  hosts:
  - ratings
  http:
  - fault:
      delay:
        fixedDelay: 7s
        percent: 100
    match:
    - headers:
        end-user:
          exact: USER_NAME
    route:
    - destination:
        host: ratings
        subset: v1
  - route:
    - destination:
        host: ratings
        subset: v1
```

Now we login to /productpage as your user and observe that the page loads but because of the induced delay between services the reviews section will show:

```
Error fetching product reviews!

Sorry, product reviews are currently unavailable for this book.
```

Open the Developer Tools menu (F12) -> Network tab - webpage actually loads in about 6 seconds.


BookInfo Sample

jason (sign out)

The Comedy of Errors

Summary: Wikipedia Summary: The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details

Type:
paperback
Pages:
200
Publisher:
PublisherA
Language:
English
ISBN-10:
1234567890
ISBN-13:
123-1234567890

Error fetching product reviews!

Sorry, product reviews are currently unavailable for this book.

Elements
Console
Sources
Network
Performance
View:
Group by frame
Preserve log
Disable ca
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms

Name	Status	Type	Initiator	Size	Time	Waterfall
login	302	text/…	Other	317 B	228…	
productpage	200	doc…	login	4.3 …	6.27 s	
bootstrap.min.css	200	styl…	productp…	(fro…	6 ms	
bootstrap-theme.mi…	200	styl…	productp…	(fro…	7 ms	
jquery.min.js	200	script	productp…	(fro…	0 ms	
bootstrap.min.js	200	script	productp…	(fro…	0 ms	
glyphicons-halflings-…	200	font	productp…	(fro…	3 ms	
inject.js	200	script	content.js…	(fro…	5 ms	

8 requests | 4.6 KB transferred | Finish: 6.59 s | DOMContentLoaded: 6.49 s | Load: 6.79 s

If you logout or login as a different user, the page should load normally without any errors.

The following example introduces a 5 second delay in 10% of the requests to the ratings:v1 microservice:

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - fault:
      delay:
        percent: 10
        fixedDelay: 5s
    route:
    - destination:
        host: ratings
        subset: v1
```

Inject a route rule to create a fault using HTTP abort:

Let's introduce an HTTP abort to the ratings microservices for the your user.

Update the USER_NAME placeholder with your user name in `virtual-service-ratings-test-abort.yaml`:

```
vi virtual-service-ratings-test-abort.yaml
```

Then apply changes:

```
kubectl apply -f virtual-service-ratings-test-abort.yaml
```

To confirm the rule is in place:

```
kubectl get virtualservice ratings -o yaml
```

Output:

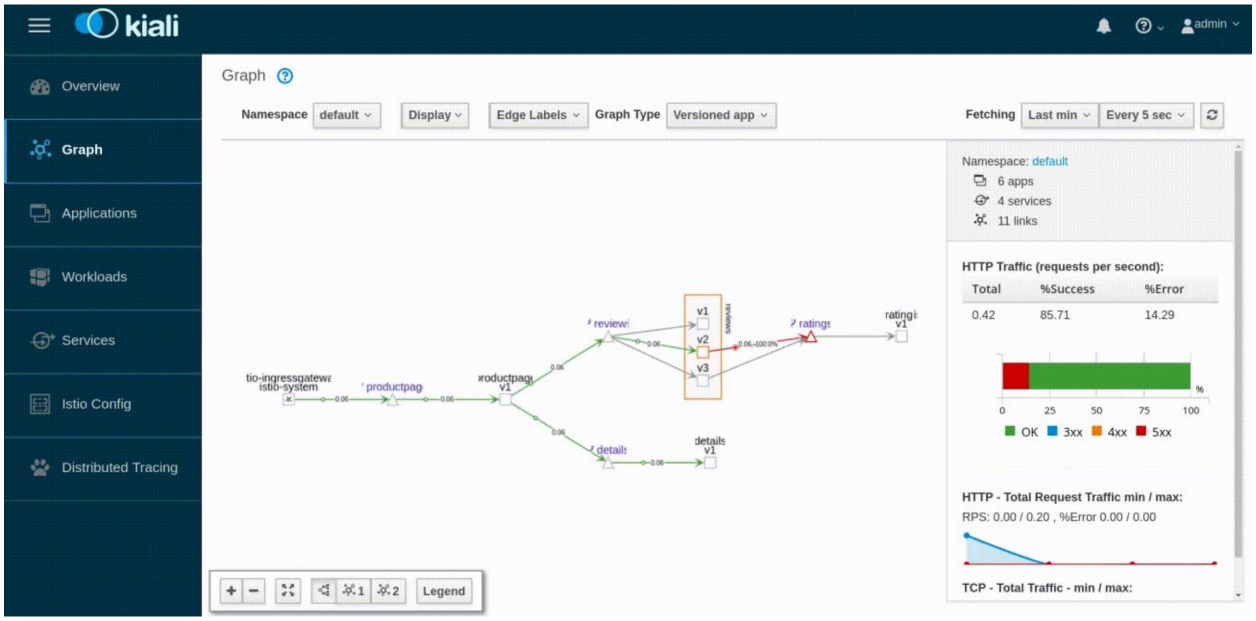
```
kind: VirtualService
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"networking.istio.io/v1alpha3","kind":"VirtualService","metadata":{"ann
otations":{},"name":"ratings","namespace":"default"},"spec":{"hosts":["ratings"],"htt
p":[{"fault":{"abort":{"httpStatus":500,"percent":100}},"match":[{"headers":{"end-
user":{"exact":"USER_NAME"}}}],"route":[{"destination":{"host":"ratings","subset":"v1
"}}]},{"route":[{"destination":{"host":"ratings","subset":"v1"}}]}]}}
  creationTimestamp: 2018-10-26T15:21:42Z
  generation: 1
  name: ratings
  namespace: default
  resourceVersion: "22405"
  selfLink:
/apis/networking.istio.io/v1alpha3/namespaces/default/virtualservices/ratings
  uid: d7d7347f-d932-11e8-88c5-0242f983c5dd
spec:
  hosts:
  - ratings
  http:
  - fault:
      abort:
        httpStatus: 500
        percent: 100
    match:
    - headers:
        end-user:
          exact: USER_NAME
    route:
    - destination:
        host: ratings
        subset: v1
  - route:
    - destination:
        host: ratings
        subset: v1
```

Now we login to /productpage as your user and observe that the page loads without any new delays but because of the induced fault between services the reviews section will show:

```
Ratings service is currently unavailable.
```

Check the flows in Kiali graph, where you should see the red communication between `reviews:v2` and `ratings`.



Namespace: default | Graph Type: Versioned app | Fetching: Last min | Every 5 sec

Namespace: default
6 apps
4 services
11 links

HTTP Traffic (requests per second):

Total	%Success	%Error
0.42	85.71	14.29

🟩 OK 🟦 3xx 🟧 4xx 🟥 5xx

HTTP - Total Request Traffic min / max:
RPS: 0.00 / 0.20 , %Error 0.00 / 0.00

TCP - Total Traffic - min / max:

Now, Verify the fault injection rule by logging out (or logging in as a different user), the page should load normally without any errors.

The following example returns an **HTTP 400** error code for **10%** of the requests to the ratings:v1service:

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - fault:
      abort:
        percent: 10
        httpStatus: 400
    route:
    - destination:
        host: ratings
        subset: v1
```