Department of Computing

Bachelor of Science (Hons) in Software Development

Cloud Data Centres - Y4

Lab 6 – all four parts

Student name: Sebastian Firsaev

Student Number: C00263348

Lecturer: Dr Lei Shi

Table of Contents

Traffic Shifting

Introduction:

This section outlines my experience with weight-based routing in Istio, specifically focusing on migrating traffic between different versions of the reviews service in the Bookinfo application. Initially, I routed all traffic to reviews:v1 and observed the absence of rating stars, as expected. I then transferred 50% of the traffic to reviews:v3, noticing red star ratings approximately half the time. Finally, I routed 100% of the traffic to reviews:v3, resulting in consistent red star ratings. This exercise showcased Istio's flexible traffic management capabilities, enabling seamless traffic migration between service versions. After completing the evaluation, I removed the application routing rules as instructed.

Getting Started with Traffic Shifting

Routing all traffic to v1 version:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/bookinfo/networking/virtual-service-all-v1.yaml
virtualservice.networking.istio.io/productpage created
virtualservice.networking.istio.io/reviews created
virtualservice.networking.istio.io/ratings created
virtualservice.networking.istio.io/details created
```

Routing 50% of traffic from v1 to v3:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/bookinfo/networking/virtual-service-reviews-50-v3.yaml
virtualservice.networking.istio.io/reviews configured
```

Confirming change:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl get virtualservice reviews -o yaml
apiVersion: networking.istio.io/v1
kind: VirtualService
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"networking.istio.io/v1alpha3","kind":"VirtualService","metadata":{"annotations":{},"name":"reviews"
,"namespace":"default"},"spec":{"hosts":["reviews"],"http":[{"route":[{"destination":{"host":"reviews","subset":"v1"},"w
eight":50},{"destination":{"host":"reviews","subset":"v3"},"weight":50}]}]}}
  creationTimestamp: "2024-05-17T12:54:23Z"
  generation: 2
  name: reviews
  namespace: default
  resourceVersion: "31133"
  uid: 0b7c2776-7096-4594-978d-2ad92d5d7c6a
spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
      weight: 50
    - destination:
        host: reviews
        subset: v3
      weight: 50
```

Checking Stars:

edy of Errors

e of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and

Book Reviews

> An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!
>
> — Reviewer1
>
> ★★★★★

> Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.
>
> — Reviewer2
>
> ★★★★☆

Book Reviews

> An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!
>
> — Reviewer1

> Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.
>
> — Reviewer2

Reviews served by:
reviews-v1-5fd6d4f8f8-b5jgr

Routing all traffic to v3:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/bookinfo/networking/virtual-service-reviews-v3.yaml
virtualservice.networking.istio.io/reviews configured
```

Removing routing rules:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl delete -f samples/bookinfo/networking/virtual-service-all-v1.yaml
virtualservice.networking.istio.io "productpage" deleted
virtualservice.networking.istio.io "reviews" deleted
virtualservice.networking.istio.io "ratings" deleted
virtualservice.networking.istio.io "details" deleted
```

Request Routing

Introduction:

This section outlines my experience with request routing in Istio for the Bookinfo application. Initially, I configured Istio to direct all traffic to version 1 of each microservice. This ensured consistent behaviour, with the reviews section displaying without rating stars, as expected.

Next, I implemented user-based routing, directing traffic from user "Jason" to reviews:v2, enabling star ratings display. Other users did not see stars, illustrating successful user-based routing.

This task demonstrated Istio's dynamic traffic management capabilities, laying the groundwork for further exploration in traffic shifting.

Test the new routing configuration

Applying virtual services to route v1:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/bookinfo/networking/virtual-service-all-v1.yaml
virtualservice.networking.istio.io/productpage created
virtualservice.networking.istio.io/reviews created
virtualservice.networking.istio.io/ratings created
virtualservice.networking.istio.io/details created
```

Displaying defined routes:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl get virtualservices -o yaml
apiVersion: v1
items:
- apiVersion: networking.istio.io/v1
  kind: VirtualService
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |
        {"apiVersion":"networking.istio.io/v1alpha3","kind":"VirtualService","metadata":{"annotations":{},"name":"bookin
fo","namespace":"default"},"spec":{"gateways":["bookinfo-gateway"],"hosts":["*"],"http":[{"match":[{"uri":{"exact":"/pro
ductpage"}},{"uri":{"prefix":"/static"}},{"uri":{"exact":"/login"}},{"uri":{"exact":"/logout"}},{"uri":{"prefix":"/api/v
1/products"}}],"route":[{"destination":{"host":"productpage","port":{"number":9080}}}]}]}
    creationTimestamp: "2024-05-16T21:40:27Z"
    generation: 1
    name: bookinfo
    namespace: default
    resourceVersion: "10687"
    uid: 7798997f-ee2c-4b4c-a84b-b3e05069d290
  spec:
    gateways:
    - bookinfo-gateway
    hosts:
    - '*'
    http:
    - match:
      - uri:
          exact: /productpage
      - uri:
```

Test the new routing configuration

Testing new routing configuration, no stars are displayed no matter the refreshes

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2

Route based on user identity

Route to v2 is the user is jason:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/bookinfo/networking/virtual-service-reviews-test-v2.yaml
virtualservice.networking.istio.io/reviews configured
```

```
spec:
  hosts:
  - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: reviews
        subset: v2
  - route:
    - destination:
        host: reviews
        subset: v1
```

Stars are visible due to login as Jason:

dy of Errors

his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1

★★★★★

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2

★★★★☆

Reviews served by:
reviews-v2-6f9b55c5db-k4vbw

No Stars visible due to different route (user Sebastian):

y of Errors

is most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns an

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2

Reviews served by:
reviews-v1-5fd6d4f8f8-b5jgr

Cleanup

```
PS C:\Users\s3bas\istio-1.22.0> kubectl delete -f samples/bookinfo/networking/virtual-service-all-v1.yaml
virtualservice.networking.istio.io "productpage" deleted
virtualservice.networking.istio.io "reviews" deleted
virtualservice.networking.istio.io "ratings" deleted
virtualservice.networking.istio.io "details" deleted
```

Fault Injection

Introduction:

This section details my exploration of fault injection in Istio, focusing on testing the resiliency of the Bookinfo application. The task involved injecting HTTP delays and aborts to assess the application's behavior under adverse conditions.

Initially, I injected a 7-second delay between the reviews:v2 and ratings microservices for user "jason." This exercise highlighted a bug in the application's timeouts, common in microservices architectures. The fault injection rules facilitated the identification of this anomaly without impacting end users. Additionally, I introduced an HTTP abort fault for user "jason" in the ratings microservice. As expected, this immediately displayed the "Ratings service is currently unavailable" message on the Bookinfo web application for user "jason."

These fault injection tests underscore the importance of testing microservice resiliency and demonstrate Istio's capability in identifying and diagnosing potential issues in distributed systems.

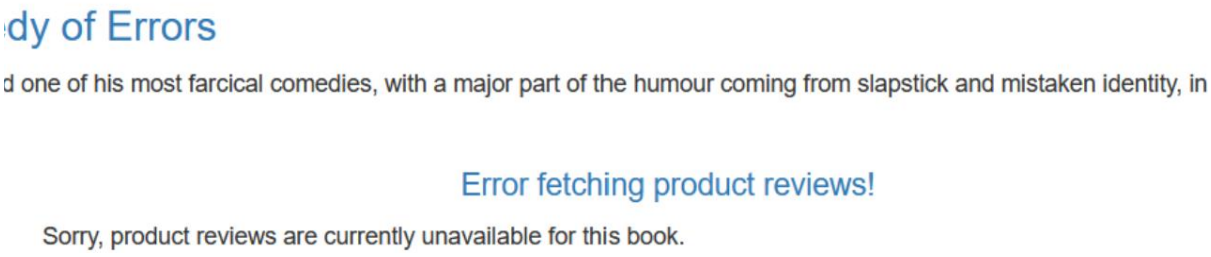
Creating a fault injection rule to delay traffic coming from the test user:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/bookinfo/networking/virtual-service-ratings-test-delay.yaml
virtualservice.networking.istio.io/ratings created
```

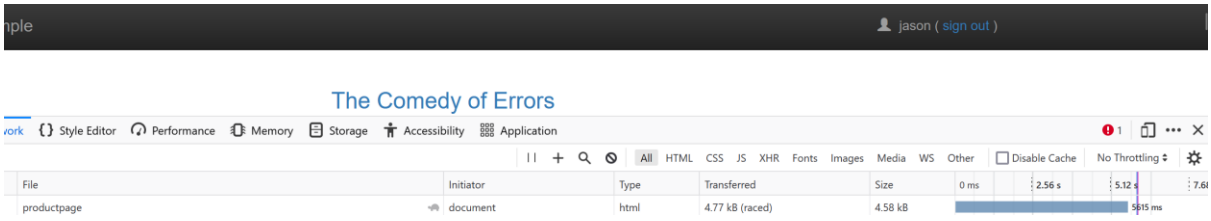
Confirming rule creation:

```
  generation: 1
  name: ratings
  namespace: default
  resourceVersion: "33719"
  uid: d00105ef-c382-4c6a-9cc6-79af8bd0baab
spec:
  hosts:
  - ratings
  http:
  - fault:
      delay:
        fixedDelay: 7s
        percentage:
          value: 100
    match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: ratings
        subset: v1
  - route:
    - destination:
        host: ratings
        subset: v1
PS C:\Users\s3bas\istio-1.22.0>
```

Log in as Jason, slow login and missing reviews:



After reload the page was slow to reload as shown in dev tools:



Fixing the bug

introducing a HTTP abort fault:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/bookinfo/networking/virtual-service-ratings-test-abort.yaml
virtualservice.networking.istio.io/ratings configured
```

Confirming:

```
  uid: d00105ef-c382-4c6a-9cc6-79af8bd0baab
spec:
  hosts:
  - ratings
  http:
  - fault:
      abort:
        httpStatus: 500
        percentage:
          value: 100
    match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: ratings
        subset: v1
  - route:
    - destination:
        host: ratings
        subset: v1
PS C:\Users\s3bas\istio-1.22.0>
```

ratings unavailable are displayed for Jason:

dy of Errors

of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition t

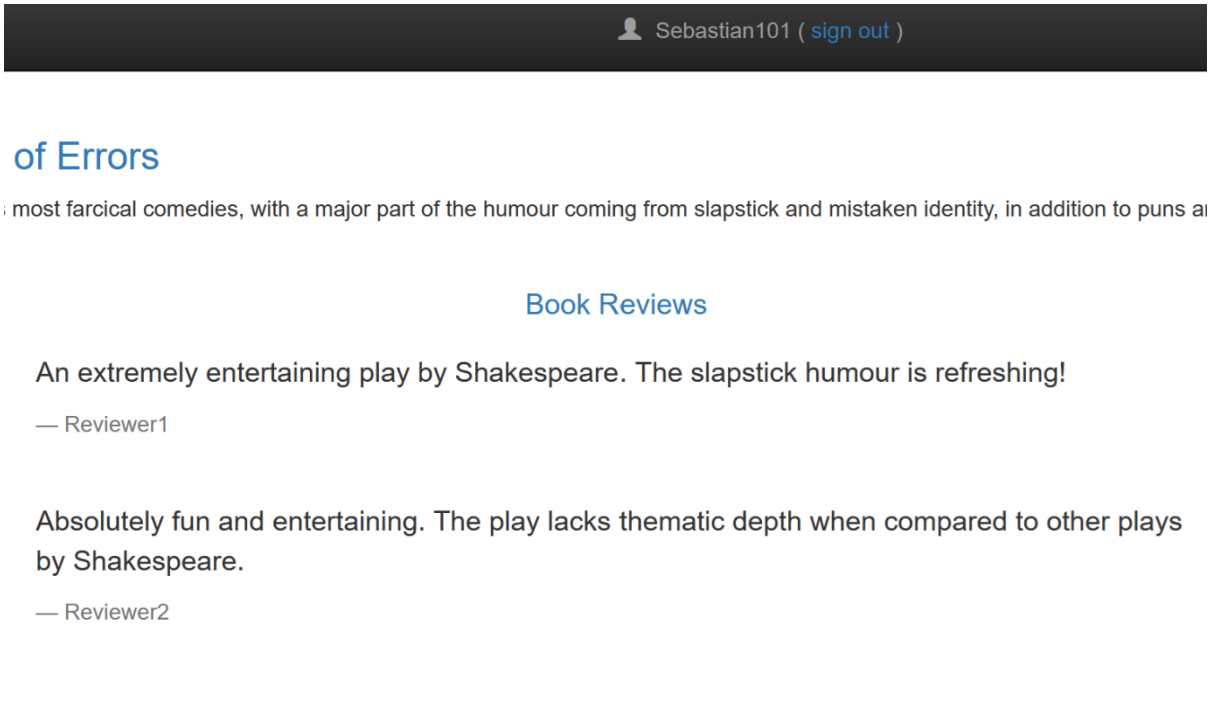
Book Reviews

> An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!
>
> — Reviewer1
>
> *Ratings service is currently unavailable*

> Absolutely fun and entertaining. The play lacks thematic depth when compared to other by Shakespeare.
>
> — Reviewer2
>
> *Ratings service is currently unavailable*

Reviews served by:
reviews-v3-7d99fd7978-fxlvq

Rating not called for other users:



Circuit Breaking

Introduction:

This section details how circuit breaking operates within an Istio-enabled setup. Initial steps involved setting up Istio and deploying the httpbin sample application. Configuring circuit breaking included defining destination rules specifying parameters like connection limits, maximum pending requests, and outlier detection thresholds. To validate the setup, I employed fortio, a client tool, to simulate varying traffic levels to the httpbin service. Through successive load tests with escalating concurrency levels, we observed the circuit breaker's behavior. Analysis of response statuses and istio-proxy statistics provided insights into how circuit breaking mechanisms manage network challenges. Key takeaways include the critical role of circuit breaking in bolstering application resilience and the practical insights gained for future deployments. This exploration provided valuable hands-on experience for building robust microservice architectures, ensuring optimal performance and reliability in real-world scenarios.

Enabling httbin sample:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/httpbin/httpbin.yaml
serviceaccount/httpbin created
service/httpbin created
deployment.apps/httpbin created
```

Creating destination rule:

```
PS C:\Users\s3bas\istio-1.22.0> @"
>> apiVersion: networking.istio.io/v1alpha3
>> kind: DestinationRule
>> metadata:
>>   name: httpbin
>> spec:
>>   host: httpbin
>>   trafficPolicy:
>>     connectionPool:
>>       tcp:
>>         maxConnections: 1
>>       http:
>>         http1MaxPendingRequests: 1
>>         maxRequestsPerConnection: 1
>>     outlierDetection:
>>       consecutive5xxErrors: 1
>>       interval: 1s
>>       baseEjectionTime: 3m
>>       maxEjectionPercent: 100
>> "@ | kubectl apply -f -
destinationrule.networking.istio.io/httpbin created
PS C:\Users\s3bas\istio-1.22.0>
```

Confirming rule:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl get destinationrule httpbin -o yaml
apiVersion: networking.istio.io/v1
kind: DestinationRule
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"networking.istio.io/v1alpha3","kind":"DestinationRule","metadata":{"annotations":{},"name":"httpbin
","namespace":"default"},"spec":{"host":"httpbin","trafficPolicy":{"connectionPool":{"http":{"http1MaxPendingRequests":1
,"maxRequestsPerConnection":1},"tcp":{"maxConnections":1}},"outlierDetection":{"baseEjectionTime":"3m","consecutive5xxEr
rors":1,"interval":"1s","maxEjectionPercent":100}}}}
  creationTimestamp: "2024-05-17T14:10:16Z"
  generation: 1
  name: httpbin
  namespace: default
  resourceVersion: "36595"
  uid: 3ad1190d-e391-439d-bcd5-17141b08e8ed
spec:
  host: httpbin
  trafficPolicy:
    connectionPool:
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1
      tcp:
        maxConnections: 1
    outlierDetection:
      baseEjectionTime: 3m
      consecutive5xxErrors: 1
      interval: 1s
      maxEjectionPercent: 100
PS C:\Users\s3bas\istio-1.22.0>
```

Adding a client

Creating a client to send traffic to the httpbin service.

```
PS C:\Users\s3bas\istio-1.22.0> kubectl apply -f samples/httpbin/sample-client/fortio-deploy.yaml
service/fortio created
deployment.apps/fortio-deploy created
```

Making curl call from fortio:

```
PS C:\Users\s3bas\istio-1.22.0> $FORTIO_POD = kubectl get pods -l app=fortio -o 'jsonpath={.items[0].metadata.name}'
PS C:\Users\s3bas\istio-1.22.0> kubectl exec "$FORTIO_POD" -c fortio -- /usr/bin/fortio curl -quiet http://httpbin:8000/
get
{
  "args": {},
  "headers": {
    "Host": "httpbin:8000",
    "User-Agent": "fortio.org/fortio-1.60.3",
    "X-Envoy-Attempt-Count": "1",
    "X-Forwarded-Client-Cert": "By=spiffe://cluster.local/ns/default/sa/httpbin;Hash=ff90e4656beec493e11ea5e3dcb3c8f561b
27ca1bf5b664c8eb9ab733cb5c623;Subject=\"\";URI=spiffe://cluster.local/ns/default/sa/default"
  },
  "origin": "127.0.0.6",
  "url": "http://httpbin:8000/get"
}
```

Tripping the circuit breaker

Calling the service with two concurrent connections (-c 2) and send 20 requests:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl exec "$FORTIO_POD" -c fortio -- /usr/bin/fortio load -c 2 -qps 0 -n 20 -loglevel
 Warning http://httpbin:8000/get
{"ts":1715955488.767891,"level":"info","r":1,"file":"logger.go","line":254,"msg":"Log level is now 3 Warning (was 2 Info
)"}
Fortio 1.60.3 running at 0 queries per second, 8->8 procs, for 20 calls: http://httpbin:8000/get
Starting at max qps with 2 thread(s) [gomax 8] for exactly 20 calls (10 per thread + 0)
{"ts":1715955488.780800,"level":"warn","r":24,"file":"http_client.go","line":1104,"msg":"Non ok http code","code":503,"s
tatus":"HTTP/1.1 503","thread":1,"run":0}
Ended after 90.087363ms : 20 calls. qps=222.01
Aggregated Function Time : count 20 avg 0.0089120272 +/- 0.008365 min 0.004089162 max 0.033659787 sum 0.178240544
# range, mid point, percentile, count
>= 0.00408916 <= 0.005 , 0.00454458 , 40.00, 8
> 0.005 <= 0.006 , 0.0055 , 65.00, 5
> 0.006 <= 0.007 , 0.0065 , 70.00, 1
> 0.007 <= 0.008 , 0.0075 , 75.00, 1
> 0.01 <= 0.011 , 0.0105 , 80.00, 1
> 0.011 <= 0.012 , 0.0115 , 90.00, 2
```

Most still made it through:

```
10.108.184.10:8000: 3
Code 200 : 19 (95.0 %)
Code 503 : 1 (5.0 %)
Response Header Sizes : count 20 avg 218.65 +/- 50.16 min 0 max 231 sum 4373
Response Body/Total Sizes : count 20 avg 634.45 +/- 90.26 min 241 max 656 sum 12689
All done 20 calls (plus 0 warmup) 8.912 ms avg, 222.0 qps
```

Increasing concurrent connection to 3:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl exec "$FORTIO_POD" -c fortio -- /usr/bin/fortio load -c 3 -qps 0 -n 30 -loglevel
 Warning http://httpbin:8000/get
{"ts":1715955696.405780,"level":"info","r":1,"file":"logger.go","line":254,"msg":"Log level is now 3 Warning (was 2 Info
```

Much higher failure rate:

```
Code 200 : 13 (43.3 %)
Code 503 : 17 (56.7 %)
Response Header Sizes : count 30 avg 99.866667 +/- 114.2 min 0 max 231 sum 2996
Response Body/Total Sizes : count 30 avg 420.6 +/- 205.4 min 241 max 656 sum 12618
All done 30 calls (plus 0 warmup) 9.871 ms avg, 193.6 qps
```

Querying the istio-proxy stats:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl exec $FORTIO_POD -c istio-proxy -- pilot-agent request GET stats | Select-String
 "httpbin" | Select-String "pending"

cluster.outbound|8000||httpbin.default.svc.cluster.local.circuit_breakers.default.remaining_pending: 1
cluster.outbound|8000||httpbin.default.svc.cluster.local.circuit_breakers.default.rq_pending_open: 0
cluster.outbound|8000||httpbin.default.svc.cluster.local.circuit_breakers.high.rq_pending_open: 0
cluster.outbound|8000||httpbin.default.svc.cluster.local.upstream_rq_pending_active: 0
cluster.outbound|8000||httpbin.default.svc.cluster.local.upstream_rq_pending_failure_eject: 0
cluster.outbound|8000||httpbin.default.svc.cluster.local.upstream_rq_pending_overflow: 18
cluster.outbound|8000||httpbin.default.svc.cluster.local.upstream_rq_pending_total: 33
```

Clean up:

```
PS C:\Users\s3bas\istio-1.22.0> kubectl delete destinationrule httpbin
destinationrule.networking.istio.io "httpbin" deleted
```