# Introduction to Cloud Computing – Exercise 7

## **Scope**: Istio service mesh, request routing

Admission:

The aim of the laboratory is to get acquainted with the Istio technology and its use in the architecture of microservices.

To perform the following exercises, you should have a virtual machine with minikube installed and running.

## 1. About Istio

Familiarize yourself with Istio technology, you can use the official Istio -> https://istio.io/latest/about/service-mesh/ website or public information on the Internet.

Explain what Istio technology is and in what key aspects it is used (what are its advantages)

## 2. Bookinfo Application

At this point, we will implement an application presenting information about books (description, information such as ISBN, page numbers, etc.), we will need it to continue experimenting with Istio.

The application is divided into four microservices:

- *productpage* – This microservice extracts information from *details* and *reviews* microservices

- *details* – Provides informationabout the book

- *reviews* – Provides information about reviews about books and obtains information about ratings from the ratings microservice

- *ratings* – Provides book ratings

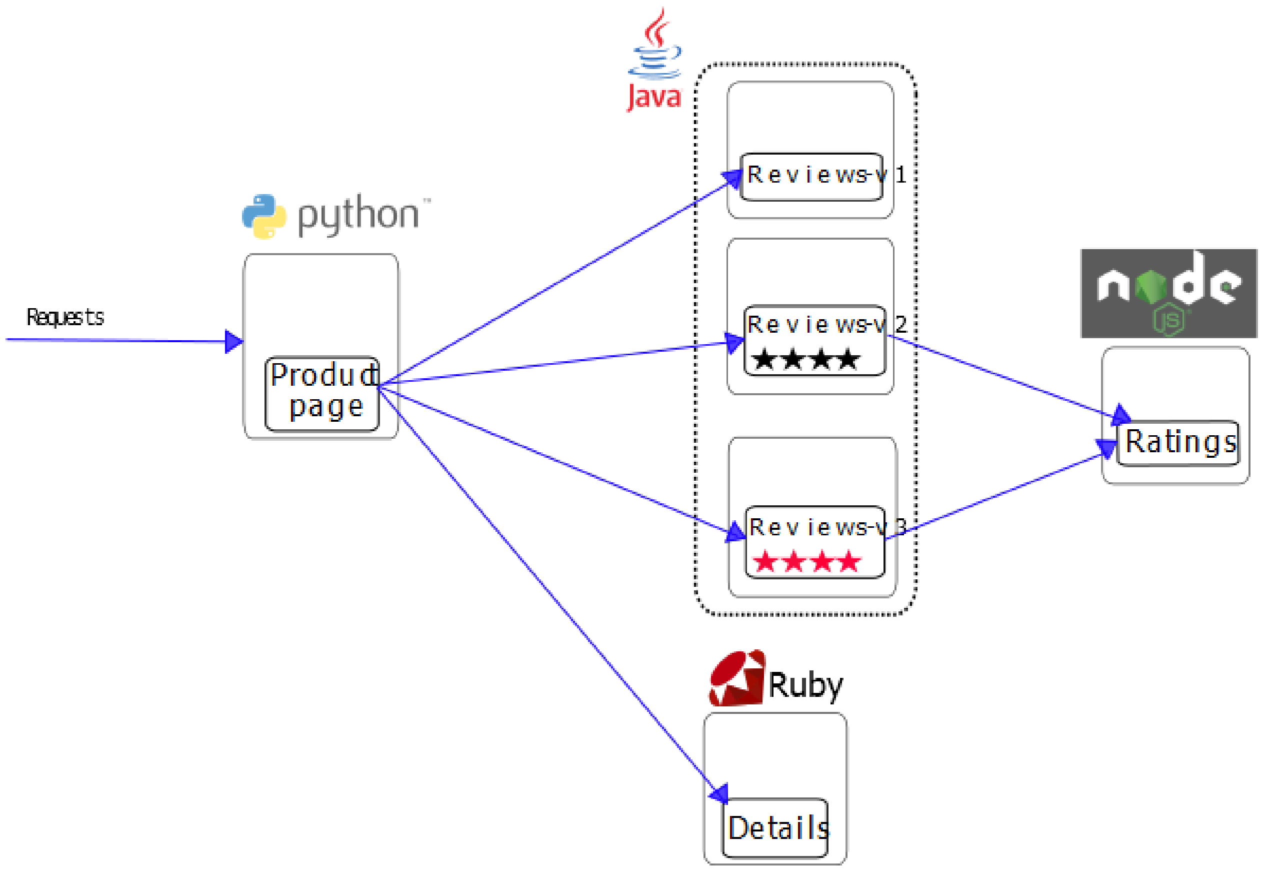
Also, the application comes in three versions of the *microservice reviews:*

- Version v1 – Does not retrieve information from *the ratings* microservice

- Version v2 – Retrieves information from the ratings microservice and displays black stars

- Version v3 – Retrieves information from the ratings microservice and displays red stars

The end-to-end architecture is shown below:



This application is multilingual, i.e. microservices are written in different languages. It is worth noting that these services are not dependent on Istio, but are an interesting example of a service mesh, especially due to the multitude of services, languages and versions of the review service.

## 3. Installing Istio

Before you start the installation, make sure that the minicube is running and call

```
kubectl cluster-info
```

Download the latest version of Istio

```
curl-L https://istio.io/downloadIstio | .sh-
```

Go to the directory with the downloaded version

```
cd istio-VERSION_NUMBER
```

Inside you will find two locations:

- Examples in *samples/* folder

- Source code of the istioctl client in the *bin* / folder

Add *the istioctl* client to the system paths

```
export PATH=$PWD/bin:$PATH
```

*A demo* configuration profile will be used for installation, it is a special profile created for testing purposes.

Call

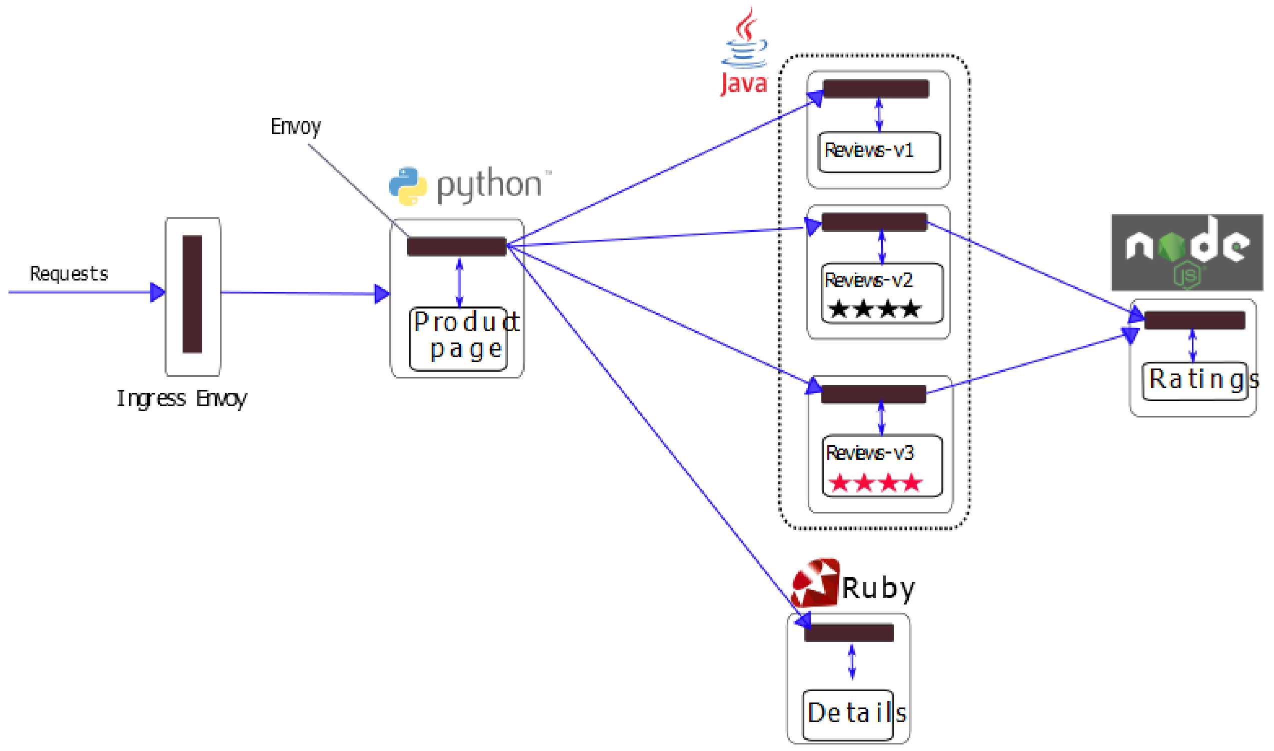
```
istioctl install --set profile=demo
```

If everything went well, addubernetes to K and a new namespace to inject *Envoy* proxy by default during implementation.

```
kubectl label namespace default istio-injection=enabled
```

## 4. Launch the application

When we have to install Istio we canrun our application. Running a test application using Istio does not require any changes to the application itself. Instead, simply configure and run the application in an Istio-enabled environment, with Envoy helpers injected into each service. The resulting implementation will look like this:



All microservices will be packaged with an Envoy helper that intercepts incoming and outgoing service calls, providing the hooks needed for external control via Istio, routing, telemetry data collection, and application-wide policy enforcement.

Bygiving in the root folder Istio run the application using the yaml file

```
kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
```

The command starts all four services shown in the bookinfo application architecture diagram. All 3 versions of the *review* service, v1, v2 and v3 have been launched. In a real environment, new versions would be deployed at intervals.

Make sure that all services and pods are correctly defined and running.

```
$ kubectl get services
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
details ClusterIP 10.0.0.31 <none> 9080/TCP 6m
Kubernetes ClusterIP 10.0.0.1 <none> 443/TCP 7d
productpage ClusterIP 10.0.0.120 <none> 9080/TCP 6m
ClusterIP ratings 10.0.0.15 <none> 9080/TCP 6m
reviews ClusterIP 10.0.0.170 <none> 9080/TCP 6m
```

and

```
$ kubectl get pods
NAME READY STATUS RESTARTS AGE
details-v1-1520924117-48z17 2/2 Running 0 6m
productpage-v1-560495357-jk1lz 2/2 Running 0 6m
ratings-v1-734492171-rnr5l 2/2 running 0 6m
reviews-v1-874083890-f0qf0 2/2 Running 0 6m
reviews-v2-1343845940-b34q5 2/2 running 0 6m
reviews-v3-1813607990-8ch52 2/2 Running 0 6m
```

To make sure that our application works,we can send a simple query using the curl method, from one of our applications (e.g. *ratings*)

```
$ kubectl exec RATINGS_POD_NAME -c ratings -- curl -sS productpage:9080/productpage |
grep -o "<title>.*</title>"
```

Making our service available externally

Now that Bookinfo is running, we can make the application available outside our Kubernetes cluster for access, for example, from a browser. The Istio Gateway is used for this .

Launch Istio Gateway

```
kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
```

Then make sure that everything is done successfully

```
$ kubectl get gateway
NAME AGE
bookinfo-gateway 32s
```

When we have the gateway running, we can proceed to the configuration of ports and host.

Call 4 commands to

```
$ export INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o
jsonpath='{.spec.ports[?( @.name=="http2")].nodePort}')
$ export SECURE_INGRESS_PORT=$(kubectl -n istio-system get service istio-
ingressgateway -o jsonpath='{.spec.ports[?( @.name=="https")].nodePort}')
$ export TCP_INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway
-o jsonpath='{.spec.ports[?( @.name=="tcp")].nodePort}')
```

```
$ export INGRESS_HOST=$(minikube ip)
```

After configuring the ports and host, add the address to our gateway

```
$ export GATEWAY_URL=$INGRESS_HOST:$INGRESS_PORT
```

To confirm that the Bookinfo application is accessible from outside the cluster, run the following curl command:

```
$ curl -s "http://${GATEWAY_URL}/productpage" | grep -o "<title>.*</title>"
<title>Simple Bookstore App</title>
```

You can also point your browser to *http://$GATEWAY_URL/productpage* to view the Bookinfo web page. If you refresh the page several times, you should see the different versions of the reviews shown on the product page, presented in the style of round robin (red stars, black stars, no stars), because we have not yet used Istio to control the routing of the version.

Before we proceed with further tasks,familiarize yourself with the concepts of destination rule, virtual service, and subset. Explain in a few words what they mean.

To  use Istio to control the routing of the Bookinfo version, you must define the available versions, called subsets, in destination rules.

Read the yaml file and perform

```
$ kubectl apply -f samples/bookinfo/networking/destination-rule-all.yaml
```

You can preview the available rules with the command

```
$ kubectl get destinationrules -o yaml
```

## 5. Routing configuration – traffic to a specific version

At this point, you will add your own rulesto control traffic between sites. To route traffic to only one version, use virtual services that set the default version of microservices. In this case, virtual services will direct traffic to version v1.

First, add a new virtual service

```
$ kubectl apply -f samples/bookinfo/networking/virtual-service-all-v1.yaml
```

Wait a few seconds for the configuration to load.

Display the configuration with the command

```
$ kubectl get virtualservices -o yaml
```

Indicate the place where to determine that all traffic should be directed to *v1 ratings*.

You can also view related subsets with the

```
$ kubectl get destinationrules -o yaml
```

You can easily test the new configuration by refreshing the /productpage of the Bookinfo application again.

Open the Bookinfo page in your browser. The URL is http://$GATEWAY_URL/productpage, where $GATEWAY_URL is the external IP address of the incoming traffic.

Note that the part of the review page shows without rating stars, no matter how many times you refresh it. This is because Istio has been configured so that all traffic from the review service is routed to the reviews:v1 version, and that version of the service does not have access to the star rating service.

## 6. Routing configuration – traffic depending on query criteria

At this point, we will configure the traffic so that it has a specific purpose for a specific user. More specifically, we want traffic for Jason to be redirected to reviews:v2. This example is made possible by the *productpage* service adding a custom end-user header to all outbound HTTP requests to the review service.

Remember, reviews:v2 is the version that includes the star feature.

Call user-based routing

```
$ kubectl apply -f samples/bookinfo/networking/virtual-service-reviews-test-v2.yaml
```

Make sure the rule is created

```
$ kubectl get virtualservice reviews -o yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
...
Spec:
  Hosts:
  - reviews
  http:
```

```
  -Match:
    -Headers:
        end-user:
          Exact: Jason
    Route:
    - destination:
        Host: reviews
        subset: v2
  -Route:
    - destination:
        Host: reviews
        subset: v1
```

What version is for each other user?

On the /productpage of the Bookinfo application, useyourself as jason (password jason).

Refresh your browser. What do you see?

See how the page looks like for any other user (any name), has the page changed?

Edit samples/bookinfo/networking/virtual-service-reviews-test-v2.yaml so that Jason has a review version:v3. Show the result to the presenter and add a screenshot to the report.