Building Resilient Microservices with Istio and Red Hat OpenShift Service Mesh

Tell us what you think!

Congratulations on completing at least 50% of the course. We would appreciate feedback on your learning experience. Click below to fill out a brief survey.

TAKE ME TO THE SURVEY (HTTPS://SURVEY.OLE.REDHAT.COM/SURVEY?COURSE=DO328&COURSE_VER=2.0&MODALITY=COURSE&OFFERING=328&UTOKEN=CHJHYNUUBXBVBM51C2FTEQ%3D%3D)

NOT RIGHT NOW

Don't ask again

(/rol/app/courses/do328-2.0/pages/ch03s05) Previous (/rol/app/cou Next

VIDEO - Building Resilient Microservices with Istio and Red Hat OpenShift Ser ... | Guided Exercise: Observing Service Interactions with Kia ...

1

Guided Exercise: Observing Service Interactions with Kiali

- Deploy an application consisting of four microservices and visualize the service interaction and traffic flow using Kiali.

The application consists of four microservices:

- The first three microservices are `czech`, `english` and `spanish`, which are simple microservices that greet the user in Czech, English and Spanish respectively.
- `greet-api`: An API gateway, which acts as the entry point for the application. The API gateway calls the individual language services in different ways depending on the request.

Outcomes

You can visualize traffic flow and inter-service communication using Kiali.

To perform this exercise, ensure you have:

- A configured and running OpenShift cluster.
- An installed and running OpenShift Service Mesh in the OpenShift cluster.
- The OpenShift CLI (`/usr/local/bin/oc`).

On the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab observe-kiali start
```

The `lab` command deploys the `czech`, `english`, `spanish`, and `greet-api` services into your Red Hat OpenShift cluster. The source code is in the Git repository at https://github.com/RedHatTraining/DO328-apps (https://github.com/RedHatTraining/DO328-apps) in the `kiali-ge` folder.

You can examine the full deployment file in the `~/DO328/labs/observe-kiali/app-deployment.yaml` file. In the `app-deployment.yaml` file, note that a gateway and a virtual service is created which exposes the following endpoints:

- `/greet`: The API gateway calls each of the individual language services in alphabetical order:

  czech → english → spanish.

- `/chained`: The API gateway calls only the `english` service. The `english` service in turn calls another service to form a chain as follows:

  english → spanish → czech.

Procedure 3.3. Instructions

1. Log in to OpenShift and verify that the four microservices are deployed.

   1.1. Run the following command to load the environment variables created in the first guided exercise:

   ```
   [student@workstation ~]$ source /usr/local/etc/ocp4.config
   ```

   1.2. Log in to OpenShift:

   ```
   [student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
    -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_API}
   Login successful.
   ...output omitted...
   ```

   1.3. Set the current project to `observe-kiali`:

   ```
   [student@workstation ~]$ oc project observe-kiali                1
   Now using project "observe-kiali" on server ...output omitted...
   ```

   1.4. Verify that there are four pods in `Running` state:

```
[student@workstation ~]$ oc get pods
AME                         READY   STATUS    RESTARTS   AGE
czech-84c5754796-cpqfq       2/2     Running   0          49s
english-v1-684884c897-qk7j2  2/2     Running   0          49s
greet-api-7fb89fdc45-f7gs6   2/2     Running   0          49s
spanish-f8848fc89-c9slw      2/2     Running   0          49s
```

2. Log in to Kiali and verify that the four microservices are in a healthy state.

   2.1. Run the `oc get route` command to gather the Kiali web console URL. You can also copy the commands from the `get-kiali-url.sh` file in the `/home/student/DO328/labs/observe-kiali` folder..

```
[student@workstation ~]$ KIALI_URL=$(oc get route kiali \
-n istio-system -o jsonpath='{.spec.host}')
```

   2.2. Access the Kiali web console using the `firefox` browser on your workstation.

WARNING

The lab start script updates the `ServiceMeshMemberRoll` resource of the service mesh control plane. This will cause the Kiali pod to be redeployed after some time. Check the status of the Kiali pod by running `oc get pods -n istio-system`, and proceed after you see it in `Running` state.

If the Kiali pod is restarted after you have logged into Kiali, or if you see errors in the Kiali console, verify the Kiali pod status and log in again.

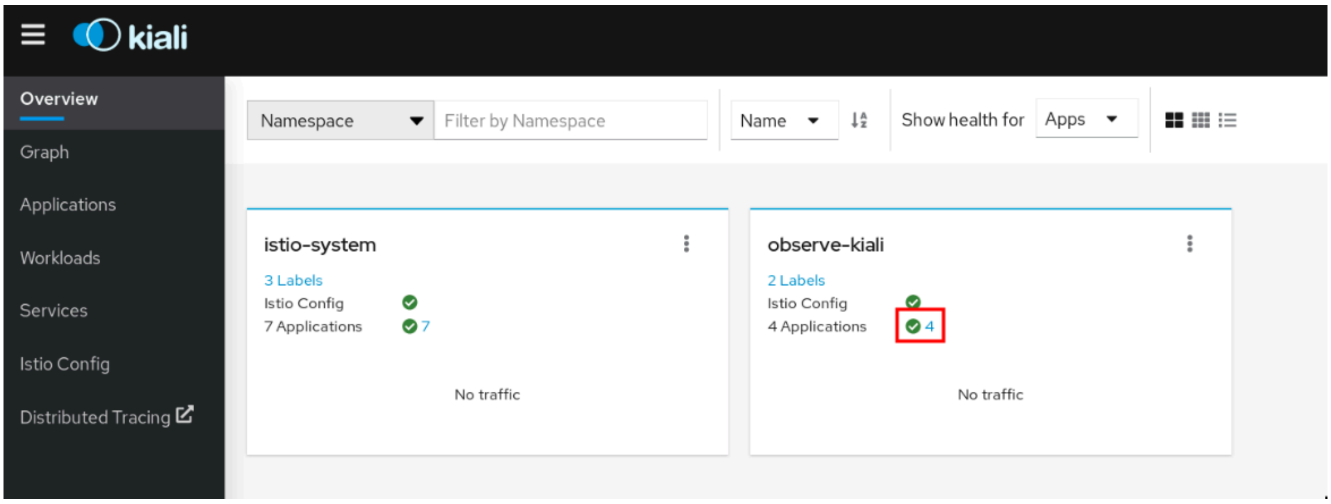
```
[student@workstation observe-metrics]$ firefox ${KIALI_URL} &
```

   2.3. Click **Log in with OpenShift**.

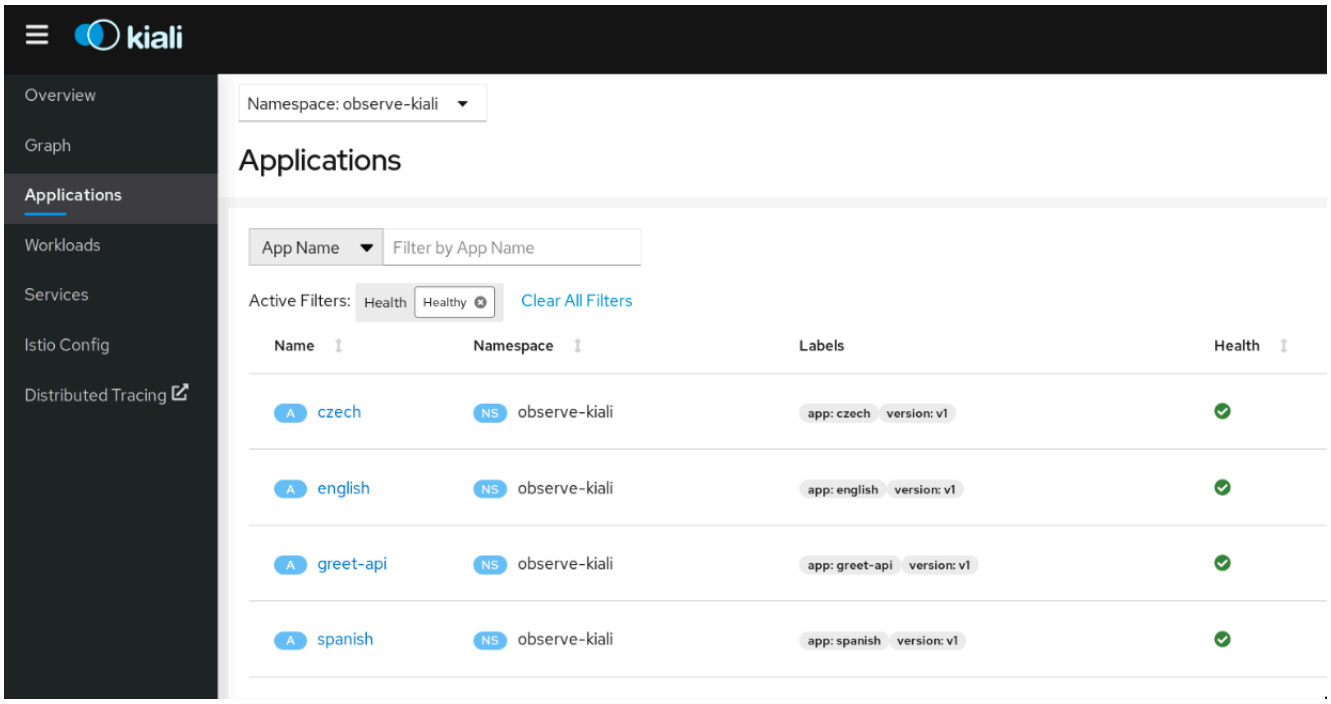
   Log in using your developer user account. Your user name is the `RHT_OCP4_DEV_USER` variable in the `/usr/local/etc/ocp4.config` classroom configuration file. Your password is the `RHT_OCP4_DEV_PASSWORD` variable in the same file.

   If you are prompted with a page asking you to authorize service account access to your account, then click **Allow selected permissions** to bring up the **Overview** page of the Kiali web console.

   2.4. Click the green tick icon in the **observe-kiali** namespace to view the **Applications** page.



Verify that all four microservices are healthy.



Name	Namespace	Labels	Health
A czech	NS observe-kiali	app: czech version: v1	✓
A english	NS observe-kiali	app: english version: v1	✓
A greet-api	NS observe-kiali	app: greet-api version: v1	✓
A spanish	NS observe-kiali	app: spanish version: v1	✓

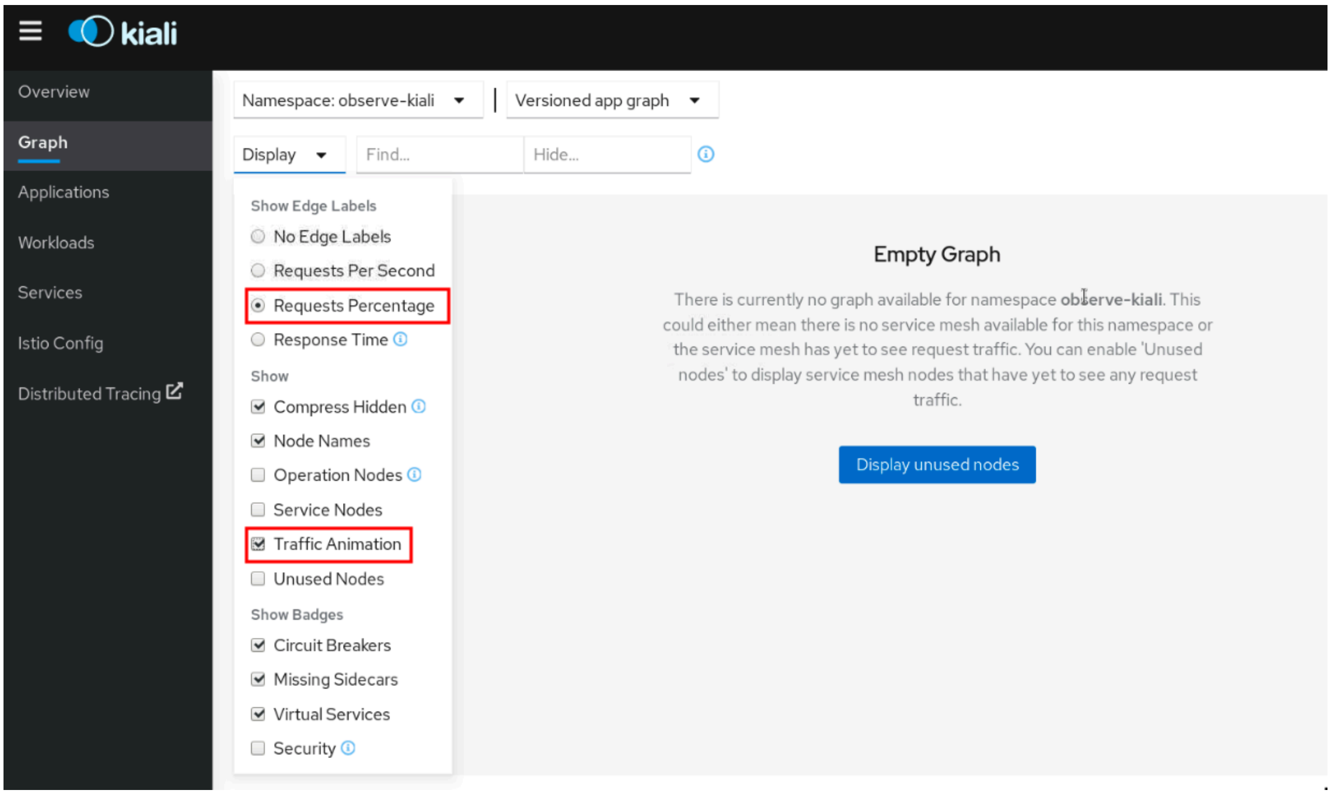
3. Invoke the `/greet` endpoint, and visualize the traffic flow in Kiali.

   3.1. Set up Kiali for traffic visualization.

   Click **Graph** in the left navigation panel. Because there is no traffic being sent to the microservices, the `Graph` page will be empty.

   Click **Display**, and select the **Requests Percentage** option to enable Kiali to show you the percentage of requests sent to different versions of a microservice.

   Select the **Traffic Animation** option to enable Kiali to show you an animated version of the traffic flow as requests come in to the service mesh.



Show Edge Labels: ( ) No Edge Labels, ( ) Requests Per Second, (●) Requests Percentage, ( ) Response Time

Show: [✓] Compress Hidden, [✓] Node Names, [ ] Operation Nodes, [ ] Service Nodes, [✓] Traffic Animation, [ ] Unused Nodes

Show Badges: [✓] Circuit Breakers, [✓] Missing Sidecars, [✓] Virtual Services, [ ] Security

There is currently no graph available for namespace **observe-kiali**. This could either mean there is no service mesh available for this namespace or the service mesh has yet to see request traffic. You can enable 'Unused nodes' to display service mesh nodes that have yet to see any request traffic.

   3.2. Run the `oc get route` command to get the URL of the istio gateway.

   You can also cut and paste the full command from the `get-ingress-gateway-url.sh` file in the `/home/student/DO328/labs/observe-kiali` folder.

   Export the ingress gateway URL to an environment variable called `GATEWAY_URL`.

1

```
[student@workstation ~]$ GATEWAY_URL=$(oc get route istio-ingressgateway -n istio-system \
 -o jsonpath='{.spec.host}')
```

3.3. Use the `curl` command to keep sending continuous requests to the `/greet` endpoint. This command will not return to the prompt, and will continue to run unless you explicitly stop it with **Ctrl+C**.

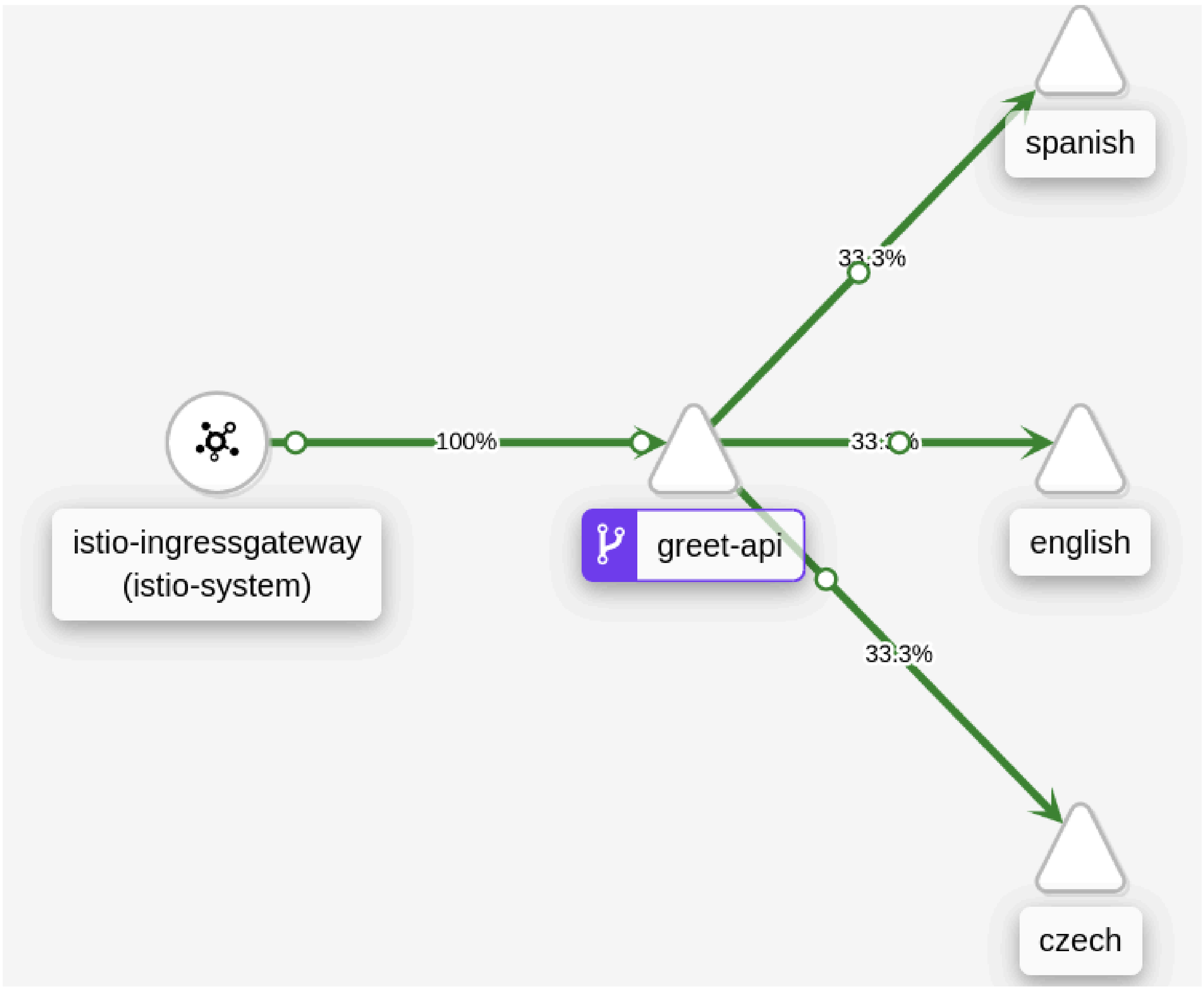
You can also run the `invoke-greet.sh` script in the `/home/student/DO328/labs/observe-kiali` folder

```
[student@workstation observe-metrics]$ while true; \
 do curl ${GATEWAY_URL}/greet; \
 sleep 3;done
Ahoj světe! | Hello World! | Hola Mundo!
Ahoj světe! | Hello World! | Hola Mundo!
...output omitted...
```

3.4. Switch to the Kiali **Graph** page, and observe the traffic animation. You might have to wait for a few seconds while Kiali captures data from the Envoy proxies and renders the animation.

By default, Kiali displays a graph with the services and their versions (**Versioned app graph**).

Click **Versioned app graph** and select the **Service graph** option to display a more compact graph with only the services in the application.

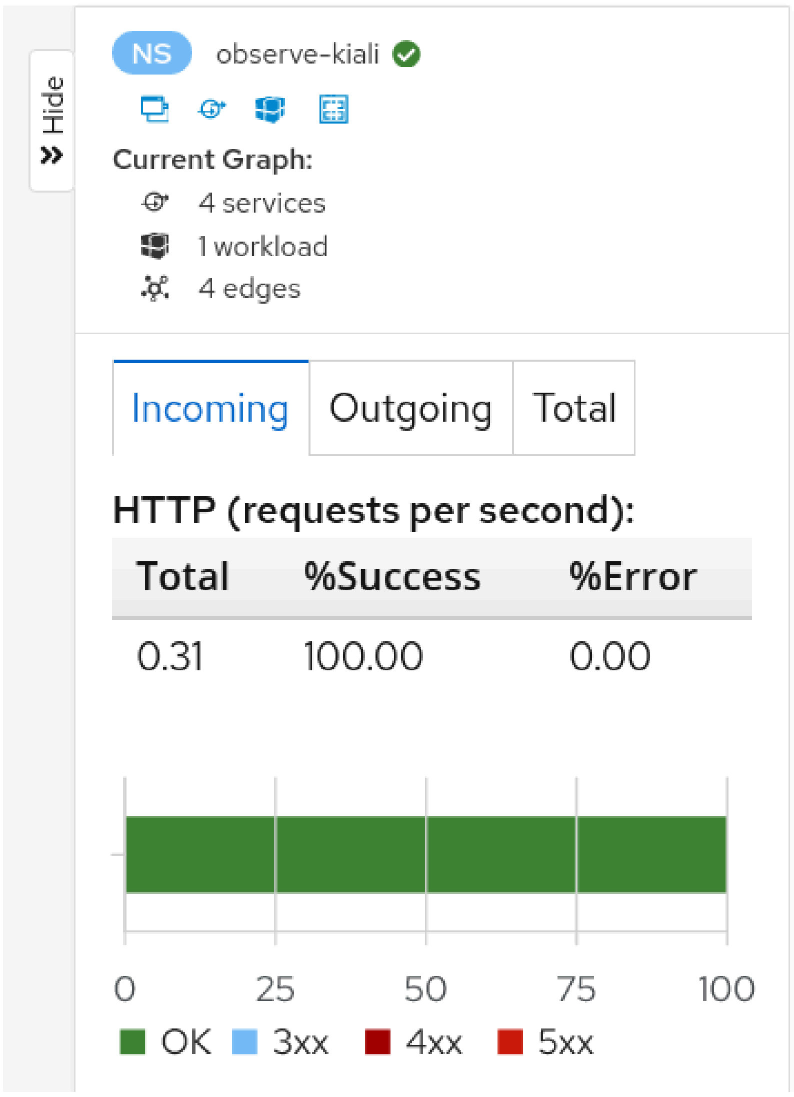


Note how the `greet-api` calls the individual services. You should see the percentage of responses being equally split between the three language services at this point.

A side panel to the right of the graph shows more details about the overall service mesh. You can click the services in the graph, and the side panel will show details of the selected service. Clicking anywhere other than the displayed services switches back to the overall service mesh view.

You can click the **Hide** or **Show** button on the side panel to hide or show the side panel respectively.

1



HTTP (requests per second):

Total	%Success	%Error
0.31	100.00	0.00

3.5. Press **Ctrl+C** to stop the curl command in the command line terminal window where you were invoking the `/greet` endpoint.

4. Invoke the `/chained` end point, and visualize the traffic flow in Kiali.

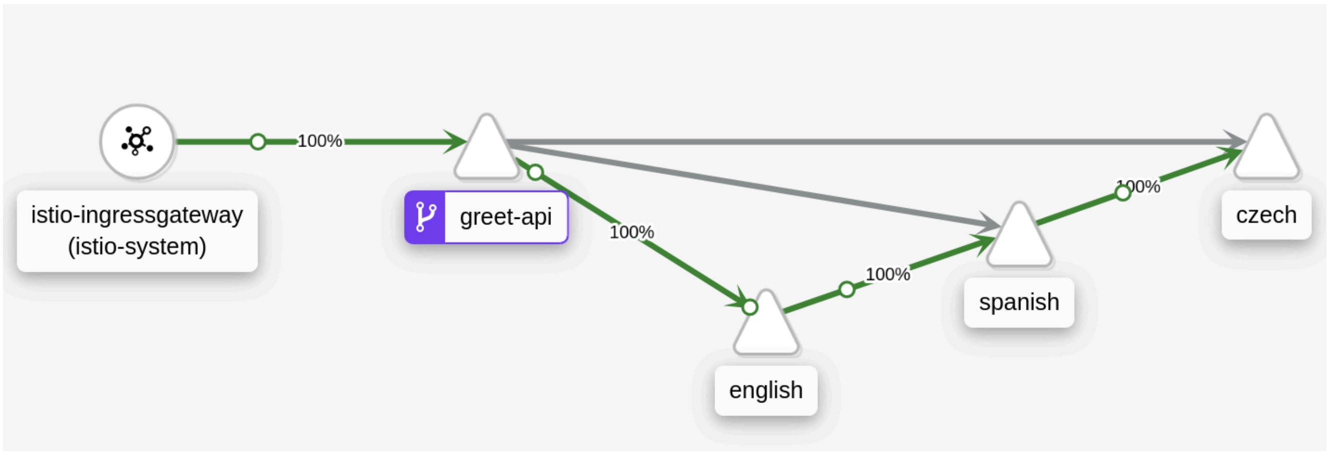
   4.1. Use the `curl` command to keep sending continuous requests to the `/chained` endpoint. Observe that the language services are now called in a different order.

   You can also run the `invoke-chained.sh` script in the `/home/student/DO328/labs/observe-kiali` folder

```
[student@workstation observe-metrics]$ while true; \
 do curl ${GATEWAY_URL}/chained; \
 sleep 3;done
Hello World! -> Hola Mundo! -> Ahoj světe!
Hello World! -> Hola Mundo! -> Ahoj světe!
...output omitted...
```

   4.2. Switch to the Kiali **Graph** page, and observe the traffic animation. You might have to wait for a few seconds while Kiali captures data from the Envoy proxies and renders the animation.

1



Note how the `greet-api` only calls the `english` service, which calls the other languages in a chain.

Do not interrupt the command line terminal where you are sending traffic to the `/chained` endpoint. Leave it running. You will need this for subsequent steps in this exercise.

5. Deploy version 2 of the `english` microservice and view the updated traffic flow in Kiali.

5.1. From a new command line terminal, run the `oc create` command to deploy version 2 of the `english` microservice. This new version prints a more informal greeting. The deployment resource is provided in the `english-v2-deploy.yaml` file in the `/home/student/DO328/labs/observe-kiali` folder.

```
[student@workstation ~]$ cd ~/DO328/labs/observe-kiali
[student@workstation observe-kiali]$ oc create -f english-v2-deploy.yaml
deployment.apps/english-v2 created
```

5.2. Run the `oc get pods` command and verify that version 2 of the `english` microservice is deployed and in `Running` state.

```
[student@workstation observe-jaeger]$ oc get pods
NAME                           READY   STATUS    RESTARTS   AGE
czech-84c5754796-cpqfq         2/2     Running   0          10m
english-v1-684884c897-qk7j2    2/2     Running   0          10m
english-v2-f696b69db-s285s     2/2     Running   0          27s
greet-api-7fb89fdc45-f7gs6     2/2     Running   0          10m
spanish-f8848fc89-c9slw        2/2     Running   0          10m
```

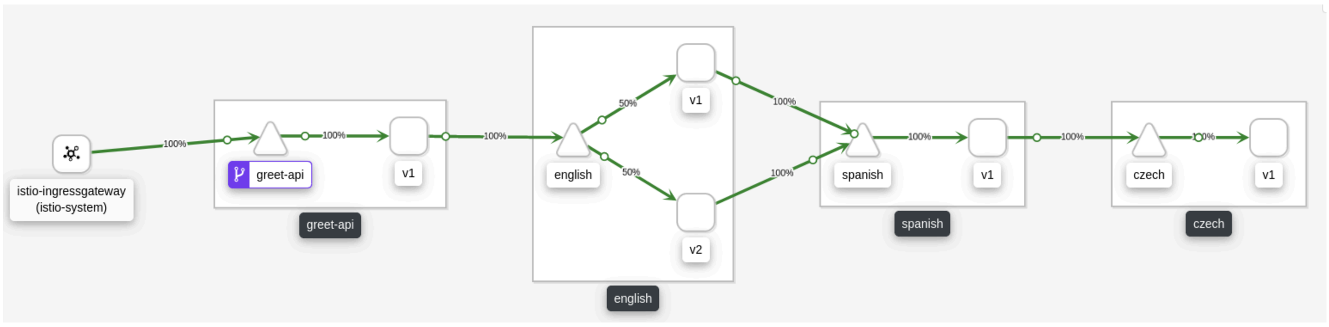
5.3. Switch to the command line terminal window running the curl command. After a while, you should see the output change to:

```
Hello World! -> Hola Mundo! -> Ahoj světe!
Hello World! -> Hola Mundo! -> Ahoj světe!
Hello World! -> Hola Mundo! -> Ahoj světe!
...output omitted...
Howdy! This mesh ain't big enough for both of us. -> Hola Mundo! -> Ahoj světe!
Hello World! -> Hola Mundo! -> Ahoj světe!
Howdy! This mesh ain't big enough for both of us. -> Hola Mundo! -> Ahoj světe!
Hello World! -> Hola Mundo! -> Ahoj světe!
...output omitted...
```

Traffic for the `english` service is split equally (load balanced) between both versions.

5.4. Switch to the Kiali **Graph** page, and observe the traffic animation. You might have to wait for a few seconds while Kiali captures data from the Envoy proxies and renders the animation.

Click **Service graph** and select **Versioned app graph**. This will change the graph to display versions of services.



> **NOTE**
>
> Your graph might not look exactly like the above. Graph nodes from previous scenarios might still be visible, but grayed out in the graph.

Note how Kiali shows the request percentage split equally between version 1 and version 2 of the `english` microservice.

6. Redirect all traffic bound for the `english` microservice to version 2 of the service. View the updated traffic flow in Kiali.

6.1. From a new command line terminal, run the `oc create` command to deploy version 2 of the `english` microservice. This new version prints a more informal greeting. The deployment resource is provided in the `english-v2-all.yaml` file in the `/home/student/DO328/labs/observe-kiali` folder.

> **NOTE**
>
> Do not worry about the details in the YAML resource file. You will learn more about traffic shaping and load balancing in subsequent chapters.

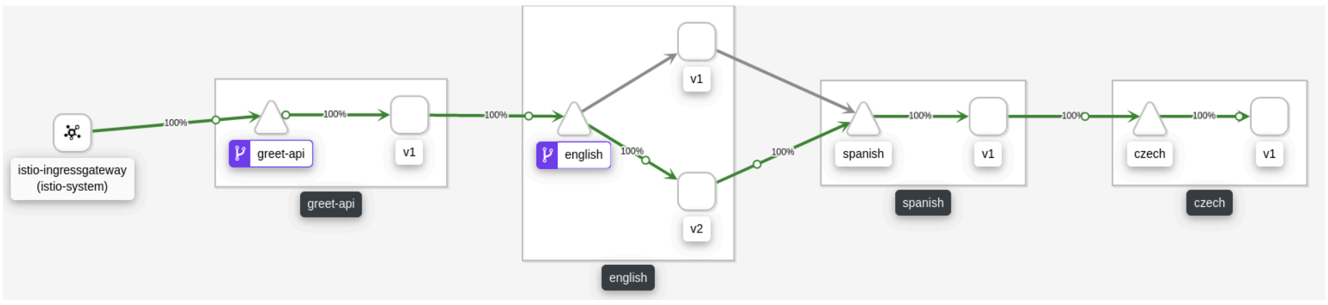
```
[student@workstation observe-kiali]$ oc create -f english-v2-all.yaml
destinationrule.networking.istio.io/english created
virtualservice.networking.istio.io/english-v2-all created
```

6.2. Switch to the command line terminal window running the curl command. After a while, you should see the output change to:

```
Hello World! -> Hola Mundo! -> Ahoj světe!
Howdy! This mesh ain't big enough for both of us. -> Hola Mundo! -> Ahoj světe!
Hello World! -> Hola Mundo! -> Ahoj světe!
...output omitted...
Howdy! This mesh ain't big enough for both of us. -> Hola Mundo! -> Ahoj světe!
Howdy! This mesh ain't big enough for both of us. -> Hola Mundo! -> Ahoj světe!
Howdy! This mesh ain't big enough for both of us. -> Hola Mundo! -> Ahoj světe!
Howdy! This mesh ain't big enough for both of us. -> Hola Mundo! -> Ahoj světe!
...output omitted...
```

All traffic for the `english` service is sent to version 2.

6.3. Switch to the Kiali **Graph** page, and observe the traffic animation. You might have to wait for a few seconds while Kiali captures data from the Envoy proxies and renders the animation.



Note how Kiali shows 100% of traffic being sent to version 2 of the `english` microservice.

6.4. Press **Ctrl**+**C** to stop the curl command in the command line terminal window where you were invoking the `/chained` endpoint.

7. Return to the home directory.

```
[student@workstation observe-kiali]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab observe-kiali finish
```

This concludes the section.

p/DO328BuildingResilientMicroserviceswithIstioandRedHatOpenShiftServiceMesh)

Welcome to the Building Resilient Microservices with Istio and OpenShift Service Mesh (DO328) group!
cschunke Aug 12, 2023

We are excited to launch a space dedicated to the Red Hat Training course, Building Resilient Microservices with Istio and Red Hat OpenShift Service Mesh! To gain the most value from this group - click the "Join Group" button in the upper right hand corner of the group home page.We encourage group members to...

3 1 432

Revision: do328-2.0-469a011

1