## Table of Contents

# Red Hat OpenShift Service Mesh Installation Lab

**Goals**

- Set up the Red Hat® OpenShift® Service Mesh *operator*
- Use the Red Hat OpenShift Service Mesh operator to deploy a *multi-tenant* Service Mesh

# 1. Set Up Red Hat OpenShift Service Mesh on Cluster

Installing the Service Mesh involves :

1. Installing Elasticsearch, Jaeger, Kiali
2. Installing the Service Mesh Operator
3. Creating and managing a *ServiceMeshControlPlane* resource to deploy the Service Mesh control plane
4. Creating a *ServiceMeshMemberRoll* resource to specify the namespaces associated with the Service Mesh.

**NOTE** The latest supported product installation instructions are located here (https://docs.openshift.com/container-platform/4.1/service_mesh/service_mesh_install/installing-ossm.html).

## 1.1. Install Service Mesh Operator Dependencies

The Red Hat OpenShift Service Mesh Operator has dependencies *Elasticsearch*, *Jaeger* and *Kiali* operators.

In this section of the lab, you will install these operator dependencies from the *Catalog* of your OCP web console (which pulls operators from: OperatorHub (https://operatorhub.io/)).

### 1.1.1. OCP Web Console *Catalog*

1. From the previous lab, you should already have a tab in your browser opened to the OCP Web Console.

If you do not still have the OCP Web Console open, its URL can be identified by executing the following in the remote *bastion* node of your lab environment:

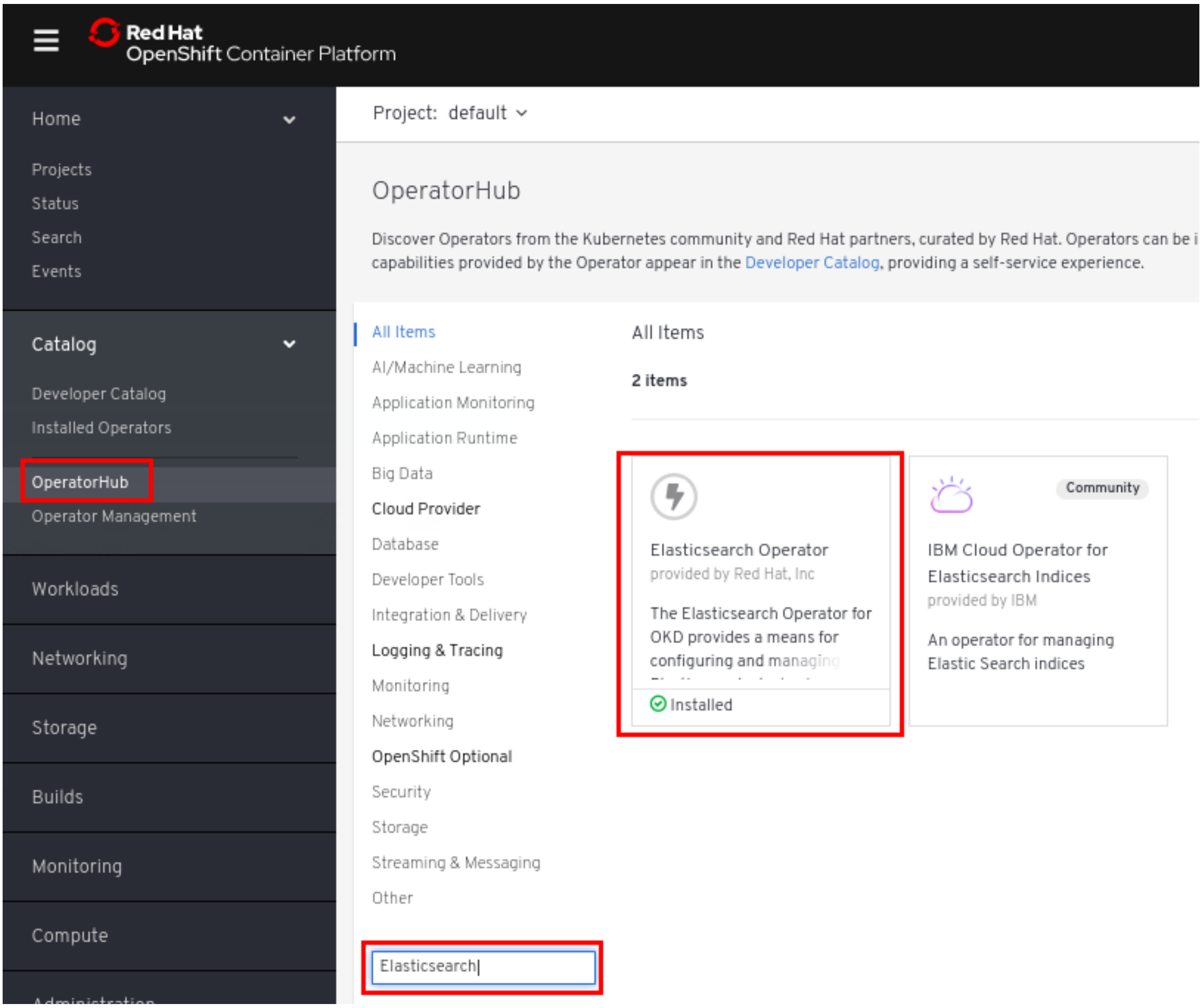
```
$ echo -en "\n\nhttps://`oc get route console -o template
--template {{.spec.host}} -n openshift-console`\n"
```

Log in using credentials of: `admin / r3dh4t1!`

2. In the OCP Web Console, navigate to: `Catalog -> Operator Hub`

### 1.1.2. Install Elasticsearch Operator

1. In the *OperatorHub* catalog of your OCP Web Console, type **Elasticsearch** into the filter box to locate the Elasticsearch Operator.



Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. Operators can be i capabilities provided by the Operator appear in the Developer Catalog, providing a self-service experience.

Elasticsearch Operator
provided by Red Hat, Inc
The Elasticsearch Operator for OKD provides a means for configuring and managing
✓ Installed

IBM Cloud Operator for Elasticsearch Indices
provided by IBM
An operator for managing Elastic Search indices

2. Click the Elasticsearch Operator to display information about the Operator

3. Click Install

4. On the *Create Operator Subscription* page, specify the following:

   a. Select `All namespaces on the cluster (default)`.

This installs the Operator in the default *openshift-operators* project and makes the Operator available to all projects in the cluster.

   b. Select the preview Update Channel.

   c. Select the Automatic Approval Strategy.

   d. Click Subscribe

5. The *Subscription Overview* page displays the Elasticsearch Operator's installation progress.
6. After about a minute, at the command line, view the new resource that represents the Elasticsearch Operator:

```
$ oc get ClusterServiceVersion

NAME                                          DISPLAY
VERSION                REPLACES   PHASE
elasticsearch-operator.4.1.14-201908291507   Elasticsearch
Operator   4.1.14-201908291507              Succeeded
```

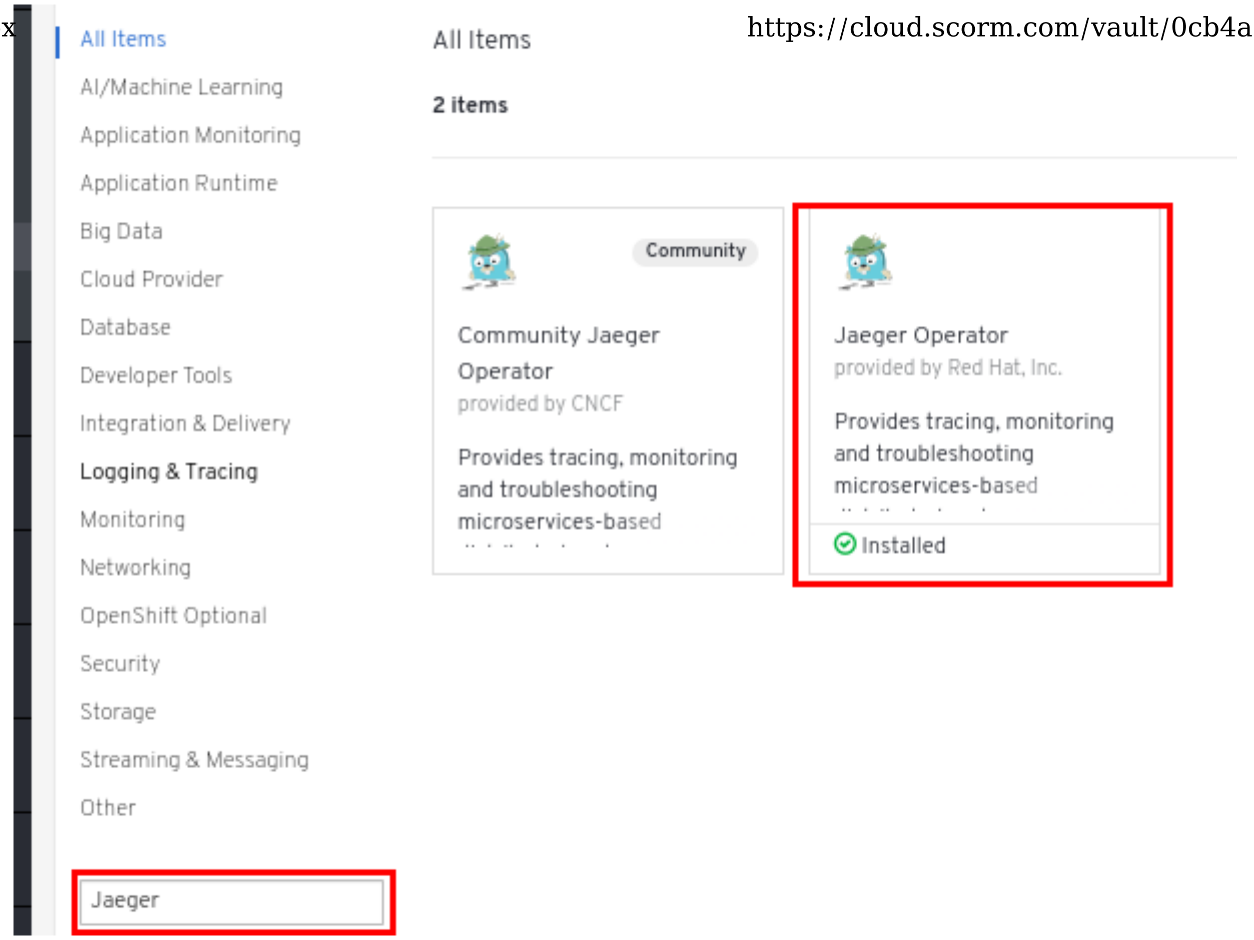
7. After about a minute, view the status of the Elasticsearch operator pod in the *openshift-operators* namespace:

```
$ oc get pod  -n openshift-operators | grep "^elasticsearch"

elasticsearch-operator-6c4fdc5975-pcx88   1/1     Running   0
1d2h
```

### 1.1.3. Install Jaeger Operator

1. In the *OperatorHub* catalog of your OCP Web Console, type **Jaeger** into the filter box to locate the Elasticsearch Operator.



2. Click the *Jaeger Operator provided by Red Hat* to display information about the Operator.
3. Click Install.
4. On the *Create Operator Subscription* page, select :
   a. All namespaces on the cluster (default).
   b. Select the stable Update Channel.
   c. Select the Automatic Approval Strategy.
   d. Click Subscribe
5. The *Subscription Overview* page displays the Jaeger Operator's installation progress.
6. After about a minute, at the command line, view the new resource that represents the Jaeger Operator:

```
$ oc get ClusterServiceVersion | grep jaeger


jaeger-operator.v1.13.1                      Jaeger Operator
1.13.1                           Succeeded
```

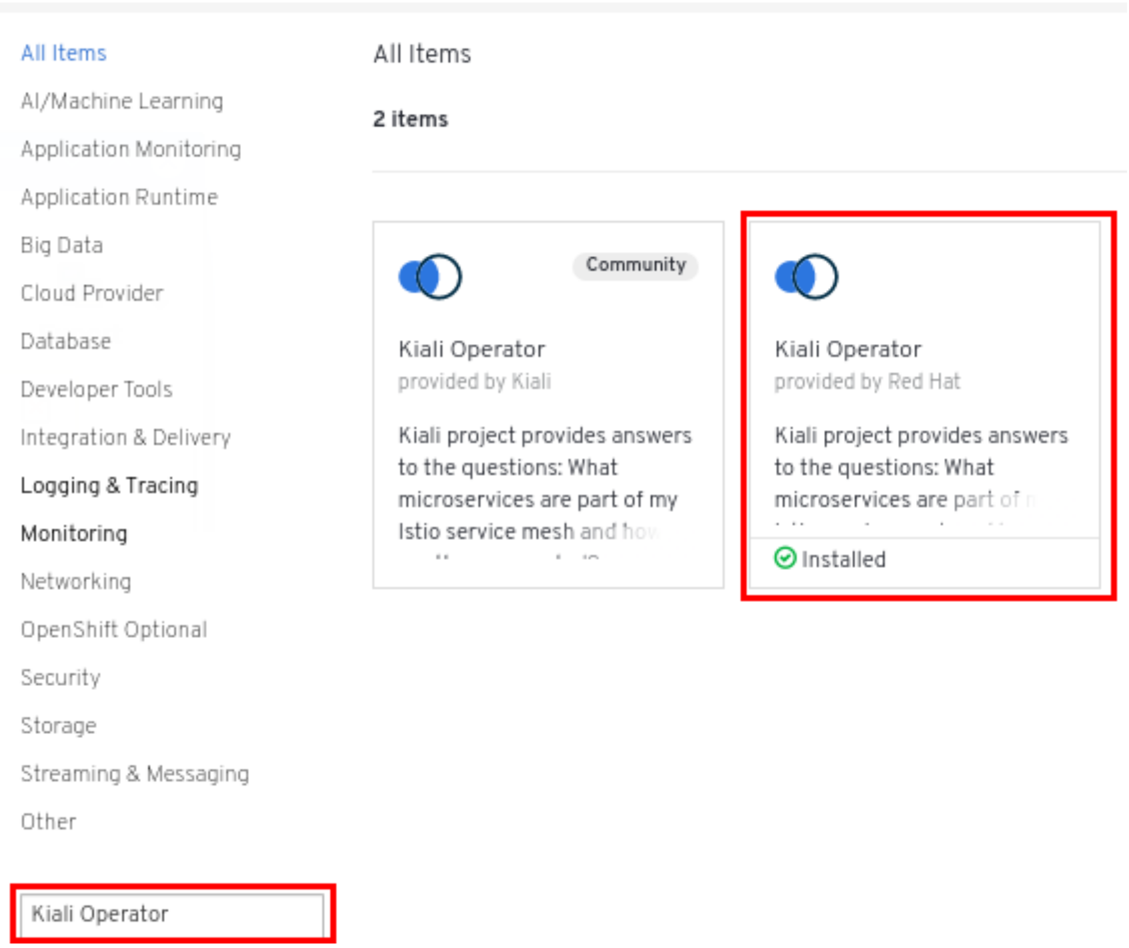
7. View the status of the Jaeger operator pod in the *openshift-operators* namespace:

```
$ oc get pod  -n openshift-operators | grep "^jaeger"


jaeger-operator-98cc68944-rgmcc   1/1     Running   0
1d2h
```

## 1.1.4. Install Kiali Operator

1. In the *OperatorHub* catalog of your OCP Web Console, type **Kiali Operator** into the filter box to locate the Elasticsearch Operator.



Kiali project provides answers to the questions: What microservices are part of my Istio service mesh and how

Kiali project provides answers to the questions: What microservices are part of

2. Click the *Kiali Operator provided by Red Hat* to display information about the Operator.

3. Click Install.

4. Populate the entries of the *Create Operator Subscription* page as follows :

   a. Select *All namespaces* on the cluster (default)

   b. Select the *stable Update Channel*

   c. Select the Automatic Approval Strategy

   d. Click Subscribe

5. The *Subscription Overview* page displays the Kiali Operator’s installation progress

6. After about a minute, at the command line, view the new resource that represents the Jaeger Operator:

```
$ oc get ClusterServiceVersion | grep kiali



kiali-operator.v1.0.5                       Kiali Operator
1.0.5                           Installing
```

7. View the status of the Kiali operator pod in the *openshift-operators* namespace:

```
$ oc get pod  -n openshift-operators | grep "^kiali"

kiali-operator-64c8487b6f-pp4k9   1/1     Running   0
1d2h
```

## 1.2. Set Up Service Mesh Operator

Now that pre-req operators have been installed, the next step in installing the service mesh is to install the service mesh operator.

1. Create an Istio operator namespace, then switch into the istio-operator project:

```
oc adm new-project istio-operator --display-name="Service Mesh Operator"
oc project istio-operator
```

2. Create the Istio operator in the `istio-operator` project:

```
oc apply -n istio-operator -f https://raw.githubusercontent.com/Maistra/istio-operator
/maistra-1.0.0/deploy/servicemesh-operator.yaml
```

**Sample Output**

3. Verify that the operator is running:

```
oc get pod -n istio-operator
```

**Sample Output**

4. Verify that the operator launched successfully:

```
oc logs -n istio-operator $(oc -n istio-operator get pods -l name=istio-operator
--output=jsonpath={.items..metadata.name})
```

**Sample Output**

## 1.3. Deploy Control Plane

Now that the Service Mesh Operator has been installed, you can now install a Service Mesh *control plane*.

The Github repository at https://github.com/Maistra/istio-operator/tree/maistra-1.0.0/deploy/examples (https://github.com/Maistra/istio-operator/tree/maistra-1.0.0/deploy/examples) includes a few custom resource examples that you can use to deploy this control plane.

### 1.3.1. ServiceMeshControlPlane

The previously installed Service Mesh operator watches for a *ServiceMeshControlPlane* resource in all namespaces. Based on the configurations defined in that *ServiceMeshControlPlane*, the operator creates the Service Mesh *control plane*.

In this section of the lab, you define a *ServiceMeshControlPlane* and apply it to the *istio-system* namespace.

1. Create a namespace called *istio-system* where the Service Mesh *control plane* will be installed.

   ```
   $ oc adm new-project istio-system --display-name="Service Mesh
   System"
   ```

2. Create the custom resource file in your home directory:

```
echo "apiVersion: maistra.io/v1
kind: ServiceMeshControlPlane
metadata:
  name: service-mesh-installation
spec:
  threeScale:
    enabled: false

  istio:
    global:
      mtls: false
      disablePolicyChecks: false
      proxy:
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 500m
            memory: 128Mi

    gateways:
      istio-egressgateway:
        autoscaleEnabled: false
      istio-ingressgateway:
        autoscaleEnabled: false
        ior_enabled: false

    mixer:
      policy:
        autoscaleEnabled: false

      telemetry:
        autoscaleEnabled: false
        resources:
          requests:
            cpu: 100m
            memory: 1G
          limits:
            cpu: 500m
            memory: 4G

    pilot:
      autoscaleEnabled: false
      traceSampling: 100.0

    kiali:
      dashboard:
        user: admin
        passphrase: redhat
    tracing:
      enabled: true

" > $HOME/service-mesh.yaml
```

3. Note the following:
   - Mutual TLS is disbled by setting `mtls` to false.
   - Your Kiali username is set to `admin` and Kiali password to `redhat`.
   - You are setting the image prefix of the Istio images to `registry.redhat.io/openshift-istio-tech-preview`. This means that you are using the Red Hat provided images rather than upstream images.
4. Now create the service mesh *control plane* in the `istio-system` project:

   ```
   oc apply -f $HOME/service-mesh.yaml -n istio-system
   ```

   **Sample Output**

   ```
   servicemeshcontrolplane.maistra.io/service-mesh-installation created
   ```

5. Watch the progress of the deployment:

   ```
   watch oc get pods -n istio-system
   ```

   - It takes a minute or two before pods start appearing, and you may see some pods temporarily in `Error` and `CrashLoopBackoff` states that resolve themselves within a few seconds.

**NOTE** The entire installation process can take approximately 10-15 minutes.

6. Once the operator completes the installation successfully, confirm that you see the following pods all running successfully:

```
NAME                                   READY   STATUS    RESTARTS   AGE
grafana-86dc5978b8-m7wqf               1/1     Running   0          80s
istio-citadel-6656fc5b9b-dc8dr         1/1     Running   0          6m38s
istio-egressgateway-66c8cdd978-qgkmr   1/1     Running   0          2m42s
istio-galley-69d8bbb7c5-fx84w          1/1     Running   0          6m16s
istio-ingressgateway-844848f59f-gklxr  1/1     Running   0          2m42s
istio-pilot-798976867d-hc9mr           2/2     Running   0          3m44s
istio-policy-54556f8b9c-drn66          2/2     Running   3          4m52s
istio-sidecar-injector-694c49c4b7-8r28t 1/1    Running   0          111s
istio-telemetry-8949d7ffd-95kzt        2/2     Running   3          4m52s
jaeger-65f55f7bc6-7mcdx                1/1     Running   0          8m17s
kiali-d566b556c-l77lf                  1/1     Running   0          57s
prometheus-5cb5d7549b-nvxv5            1/1     Running   0          9m42s
```

7. Press **Ctrl+C** to exit the *watch*.

8. Examine the created routes in the `istio-system` project:

```
oc get routes -n istio-system
```

**Sample Output**

```
    NAME                    HOST/PORT
PATH   SERVICES             PORT    TERMINATION   WILDCARD
    grafana                 grafana-istio-system.apps.cluster-<GUID>>.<GUID>>.
<SANDBOX>.opentlc.com                 grafana                <all>
None
    istio-ingressgateway    istio-ingressgateway-istio-system.apps.cluster-<GUID>>.
<GUID>>.<SANDBOX>.opentlc.com     istio-ingressgateway   80                   None
    jaeger                  jaeger-query-istio-system.apps.cluster-<GUID>>.<GUID>>.
<SANDBOX>.opentlc.com            jaeger-query           <all>   edge          None
    kiali                   kiali-istio-system.apps.cluster-<GUID>>.<GUID>>.
<SANDBOX>.opentlc.com                   kiali                  20001   reencrypt
None
    prometheus              prometheus-istio-system.apps.cluster-<GUID>>.<GUID>>.
<SANDBOX>.opentlc.com              prometheus             <all>                 None
```

- Expect to see routes for `grafana`, `prometheus`, and `kiali` among others.

9. Enter this command to get the URL of the Kiali web console

```
oc get route kiali -n istio-system -o jsonpath='{"https://"}{.spec.host}{"\n"}'
```

**Sample Output**

+

```
https://kiali-istio-system.apps.cluster-<GUID>.<GUID>.<SANDBOX>.opentlc.com
```

1. Start a web browser on your computer and vist the Kiali URL.
2. At the login screen, enter the credentials:
   a. Username: admin
   b. Password: r3dh4t1!
      - Expect to see the Kiali user interface, which you use later in this course.

## 1.4. ServiceMeshMemberRoll

The Service Mesh operator has installed a control plane configured for multitenancy. This installation reduces the scope of the control plane to only those projects/namespaces listed in a **`ServiceMeshMemberRoll`**.

In this section of the lab, you create a ServiceMeshMemberRoll resource with the project/namespaces you wish to be part of the mesh. This *ServiceMeshMemberRoll* is required to be named *default* and exist in the same namespace where the *ServiceMeshControlPlane* resource resides (ie: *istio-system*).

1. Define a *ServiceMeshMemberRoll* called *default* that includes a single member project called: *user1-tutorial*.

```
echo "apiVersion: maistra.io/v1
kind: ServiceMeshMemberRoll
metadata:
  name: default
spec:
  members:
  # a list of projects joined into the service mesh
  - user1-tutorial
" > $HOME/service-mesh-roll.yaml
```

A project called *user1-tutorial* will be created in a later lab of this course. It will contain various backend business services that will be auto-injected with the Service Mesh *data plane* (aka: an *istio* side-car container) .

2. Now create the service mesh control plane *membership roll* in the `istio-system` project:

```
oc apply -f $HOME/service-mesh-roll.yaml -n istio-system
```

**Sample Output**

```
servicemeshmemberroll.maistra.io/default created
```

## 2. Update Security Context Contraints

1. Add edit access for user `user1` to istio-system namespace

```
oc adm policy add-role-to-user edit user1 -n istio-system
```

**Sample Output**

```
Warning: User 'user1' not found
clusterrole.rbac.authorization.k8s.io/edit added: "user1"
```

**NOTE** You will see an error indicating the user `user1` was not found because you have not logged in yet using that user id. OpenShift only creates user objects once a user logs into the cluster successfully. The command was successful despite this warning.

This concludes the *Red Hat OpenShift Service Mesh Installation* lab.

Last updated 2020-02-26 17:17:48 EST