**Openshift 4.3 Serverless Demo**

**Abstract:**

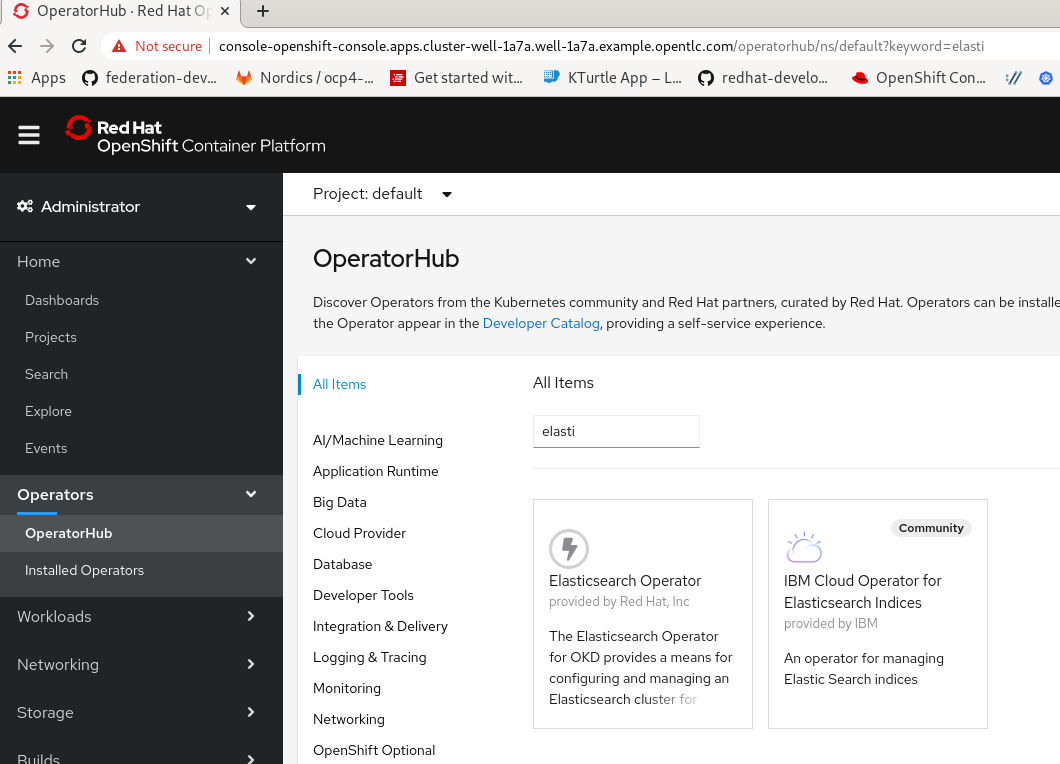
Red Hat serverless has three components, these are build, serving and events. Build is a plugable model for building artifacts like jar, files, zip or containers from source code. Serving is an event-driven model that serves the container with your application and can “scale to zero”. Events are the common infrastructure for consuming and producing events that will stimulate the application. For this lab, we will focus on building and serving only.

**Assumption:**

1. OCP 4.3 installed with 3 Masters and 3 worker nodes.
2. Machinesets are configured to allow node scaling.
3. A working Service Mesh operator deployed.

**Pre-requisites (Service Mesh deployment):**

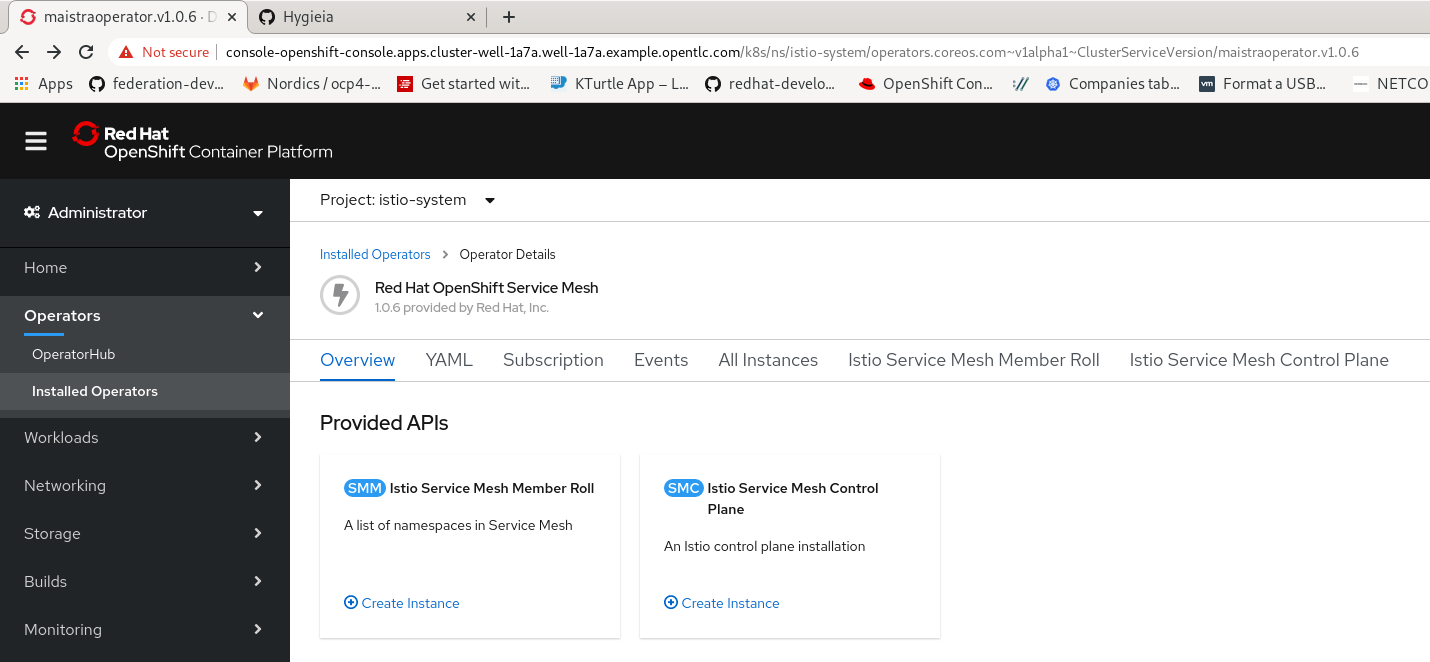
1. Deploy service mesh and the required components in your OCP 4.3 cluster (Make sure you are selecting the Red Hat supported operator).
   1. Jaeger
   2. Elastic Search
   3. Kiali
   4. Openshift Service Mesh



1. Once Service mesh and the required components are installed, you now need to deploy the service mesh control plane. The service control plane needs to be deployed in istio-system namespace.

$ oc new-project istio-system

1. Deploy Service Mesh control plane to the newly created namespace istio-system.



* 1. Wait until the control plane pods are completely created before going to the next steps. Below are the pods that you need expect once service control plane deployment is completed.

$ oc get pods

NAME READY STATUS RESTARTS AGE

grafana-9d64c5f55-qt44t 2/2 Running 0 2m40s

istio-citadel-cc769b7cd-9lr58 1/1 Running 0 7m33s

istio-egressgateway-7c976b4cb6-p4hwg 1/1 Running 0 3m36s

istio-galley-6b77d9979c-cl4sq 1/1 Running 0 5m52s

istio-ingressgateway-54947d94fd-5b9kg 1/1 Running 0 3m36s

istio-pilot-946bf9b59-2zkjj 2/2 Running 0 4m28s

istio-policy-d88cf66cb-5kgbt 2/2 Running 0 5m15s

istio-sidecar-injector-554c66fc5f-kbrpl 1/1 Running 0 3m18s

istio-telemetry-7dd5b6cfb4-sct25 2/2 Running 0 5m14s

jaeger-5c784f744b-cd4pm 2/2 Running 0 5m55s

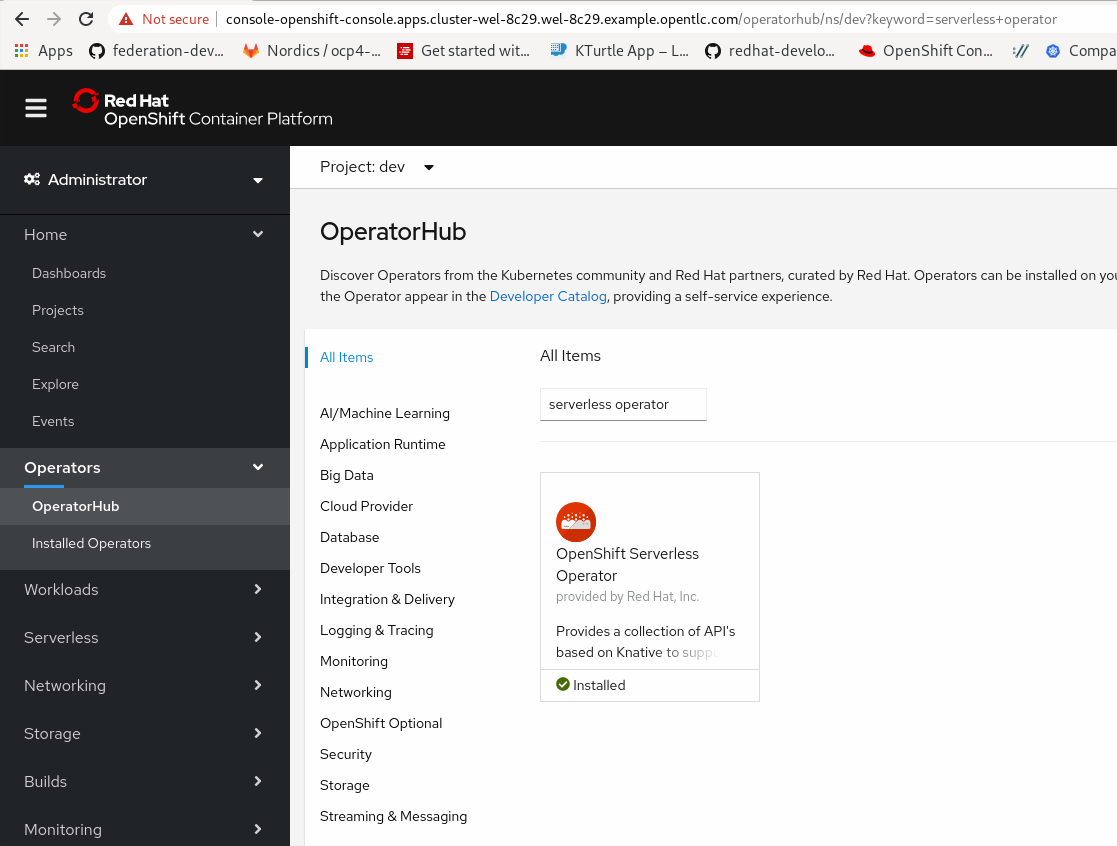
kiali-7d8776f4f7-ll8xh 1/1 Running 0 107s

prometheus-6967f6f77d-hbn44 2/2 Running 0 7m1s

1. This steps is optional - but you can test your service mesh deployment using this sample app from Istio website - <https://istio.io/docs/examples/bookinfo/>

**Serverless (Knative) Preparation:**

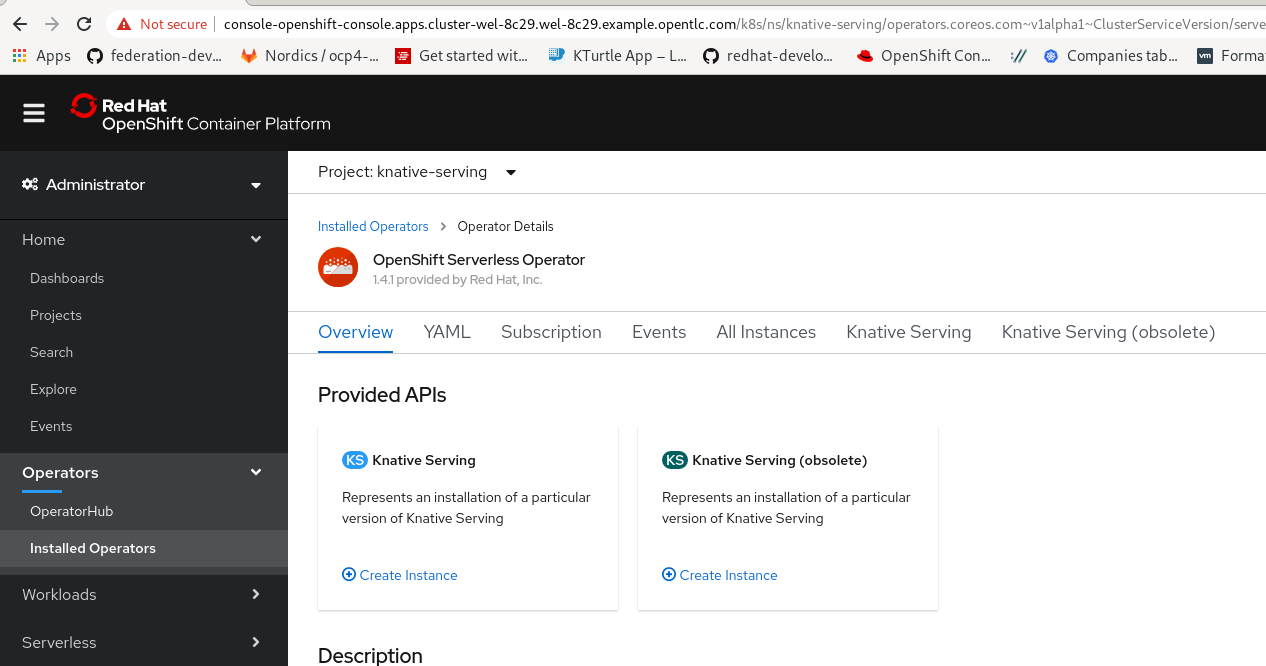
1. Login as Admin to your OCP 4.3 cluster
2. Navigate to Operator Hub and select Serverless Operator (by Red Hat) to install the operator.



4. Create a namespace and name it knative-serving. This is the name-space that we will use to deploy knativeserving instance using the serverless operator

$ oc new-project knative-serving

5. Deploy knative-native serving on the knative-serving namespace



Note to wait for the pods to be created before proceeding to the next steps. You should see something like below.

$ oc get pods -n knative-serving

NAME READY STATUS RESTARTS AGE

activator-84b5f4f4d6-nsdjz 1/1 Running 0 3h17m

autoscaler-84d84b65db-g9wps 1/1 Running 0 3h17m

autoscaler-hpa-66f7f47c4c-rlz4z 1/1 Running 0 3h17m

controller-cb7dff454-2vw92 1/1 Running 0 3h17m

networking-istio-c9c98f574-8dc4p 1/1 Running 0 3h17m

webhook-65b6db457c-9ktth 1/1 Running 0 3h17m

$

6. Once all knative-serving pods are running, you can deploy an application. There are many ways of doing this and we will go through those samples in the next section.

**Serverless App deployment**

**Using Command Line**

1. Create a namespace for the sample app that you will deploy - In this example I will use serverless-demo

$ oc new-project knative-serving

2. Deploy sample go hello world application. Below is the example code. Let’s call this service.yaml

apiVersion: serving.knative.dev/v1alpha1

kind: Service

metadata:

name: helloworld-go

namespace: serverless-demo

spec:

template:

spec:

containers:

- image: gcr.io/knative-samples/helloworld-go

env:

- name: TARGET

value: "Go Sample v1"

3. Instantiate the application using the below command.

$ oc apply -f service.yaml -n serverless-demo

service.serving.knative.dev/helloworld-go created

4. Wait until the container is created.

$ oc get pods

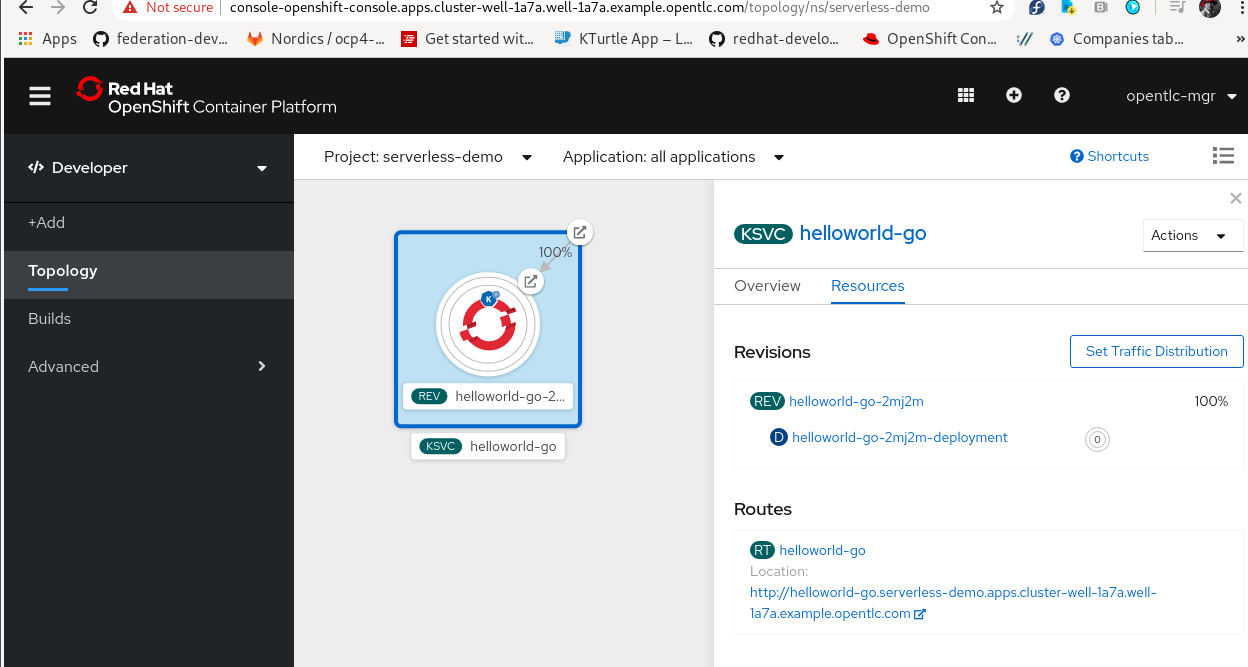
NAME READY STATUS RESTARTS AGE

helloworld-go-2mj2m-deployment-5b5999d699-2dwwv 0/2 ContainerCreating 0 8s

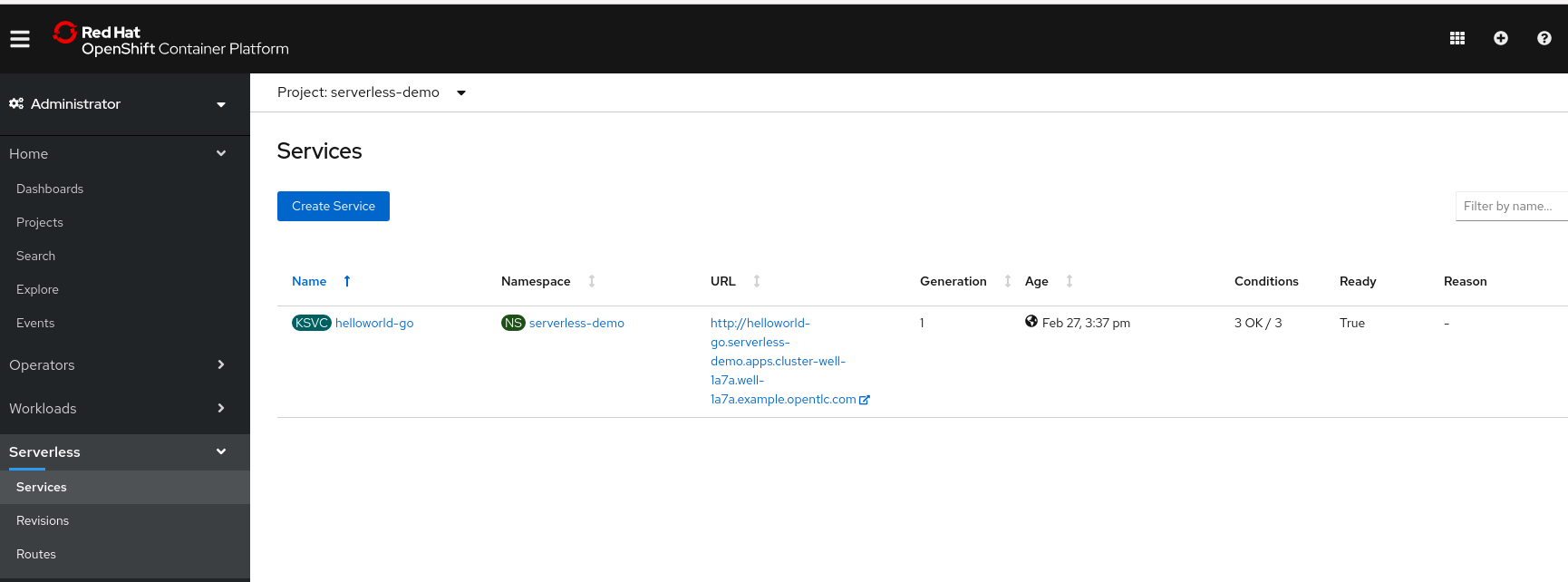
5. Once the container is created, go to the developer console and select topology and check the status of your application. If it is successfully deployed using knative resource type, then you should see something like below:

In this view, you can see the following:

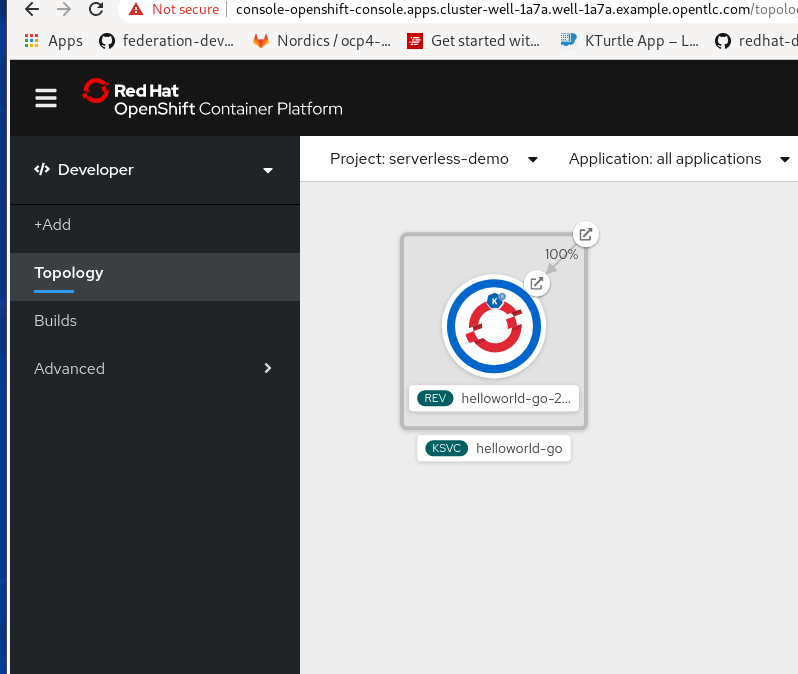
1. Resources of the application that you just deployed.
2. Routes of the application
3. The pods that was created by the deployment
4. The revisions where you can set traffic distribution.
5. Overall information that includes, labels, anotations, name of the application and the owner of the applications.



6. To verify further, go to Serverless Tab on your admin console and check the services, revisions and routes. In the routes provided, you can access your sample application.

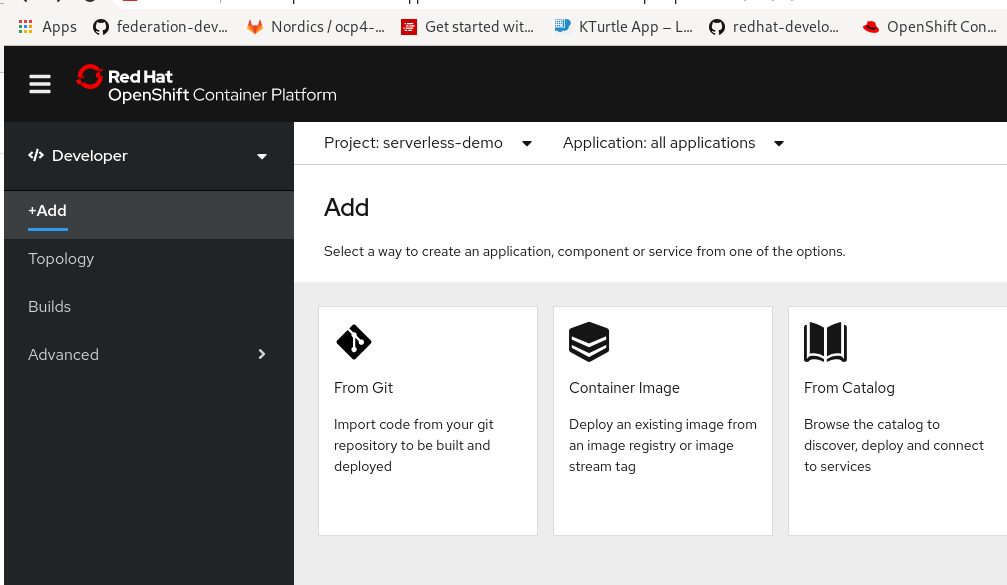


7. To test the serverless application, you can hit the routes provided and verify the status of the pods going to the developer console to see the status of the pod being re-created. This is now the point where you have successfully deploy an application that can scale down to 0 and bring it back up again once the expose url is hit by a request.

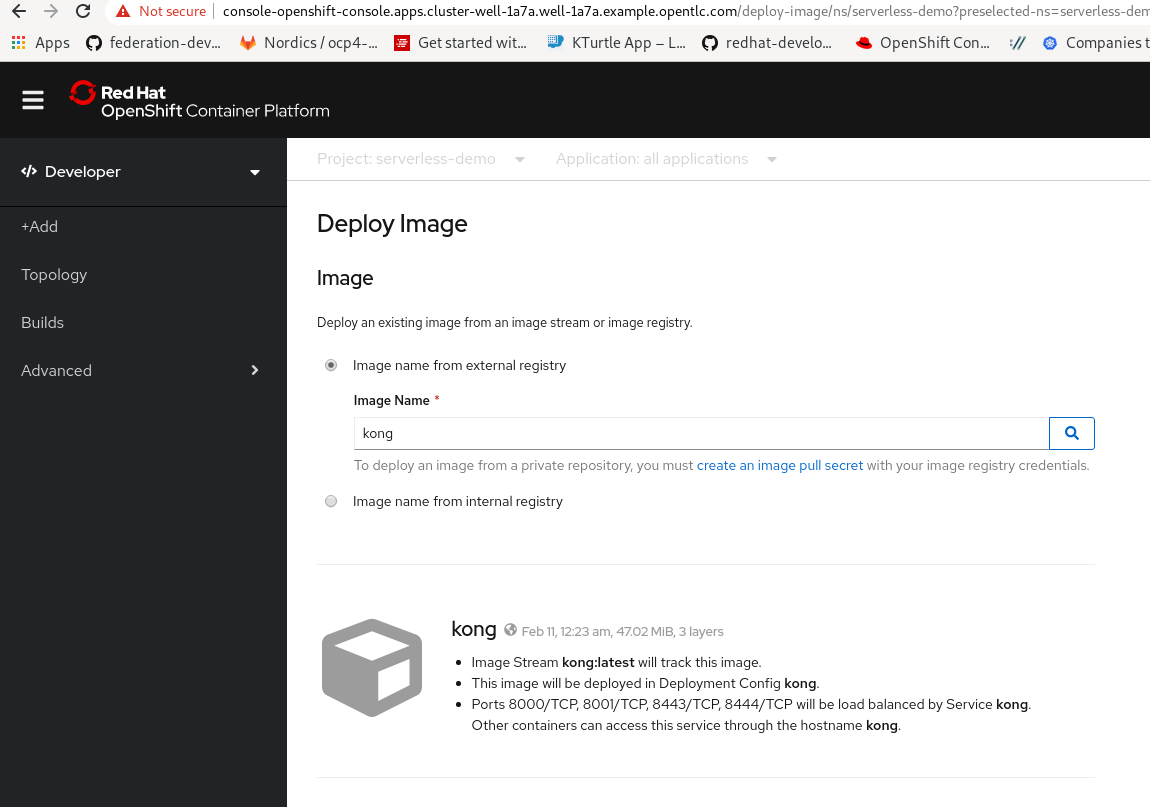


**Using Developer Console.**

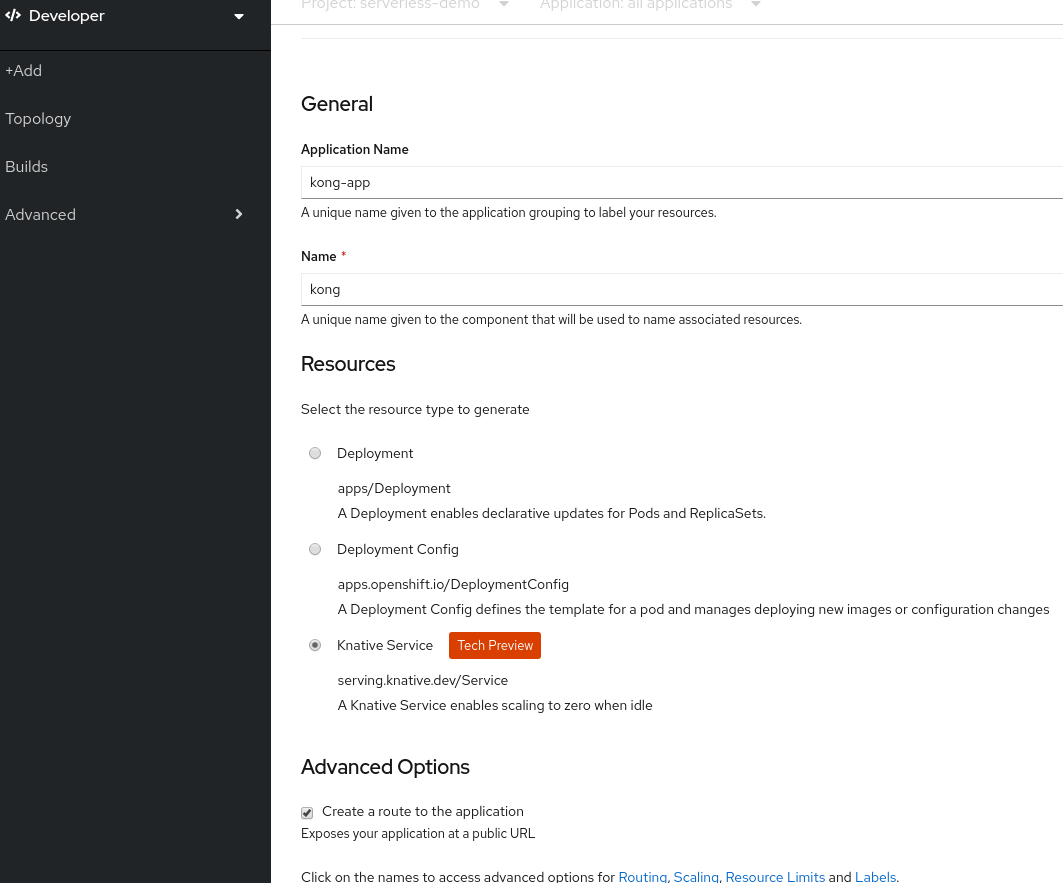
This is to show that you can create a serverless application using Developer Console.

1.Select container image. 

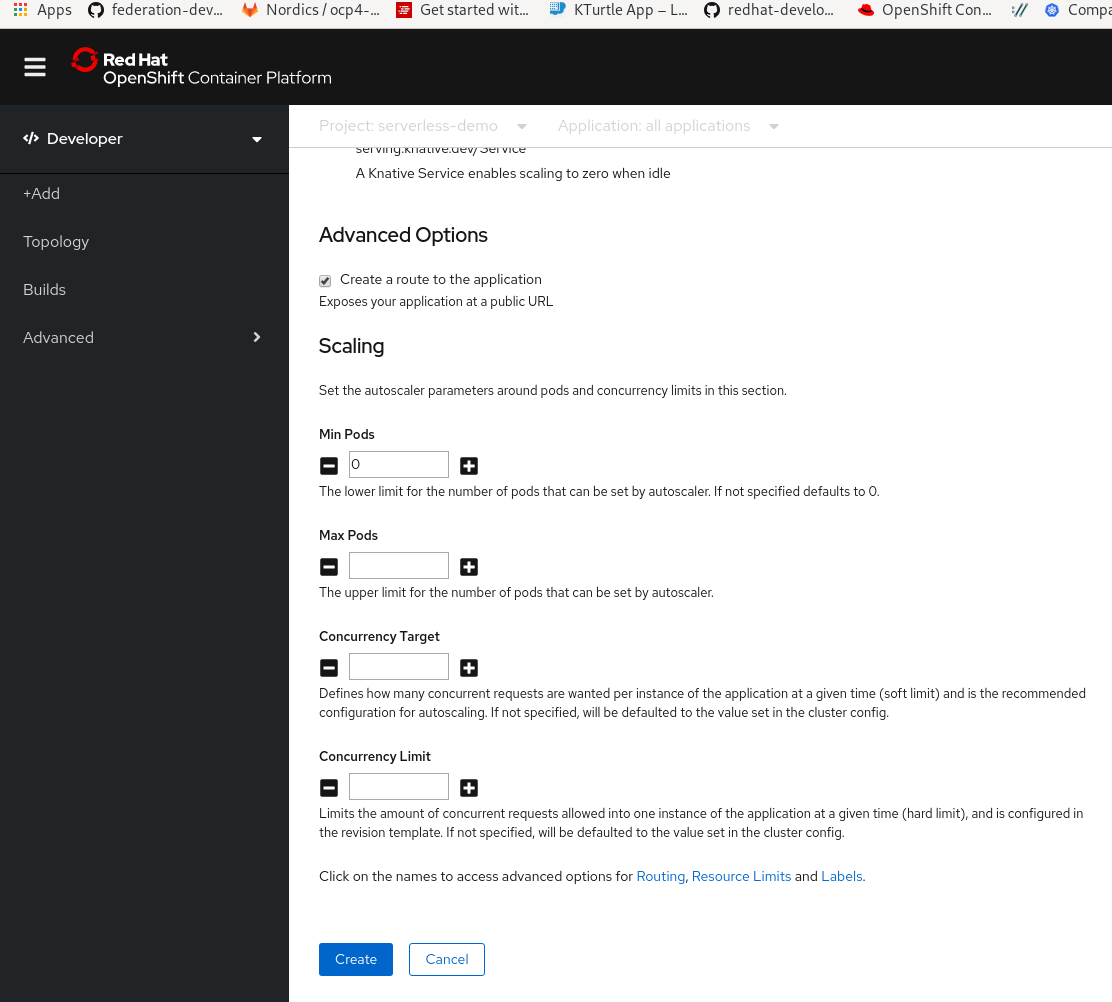
2. Make sure you verify the image by clicking the button.



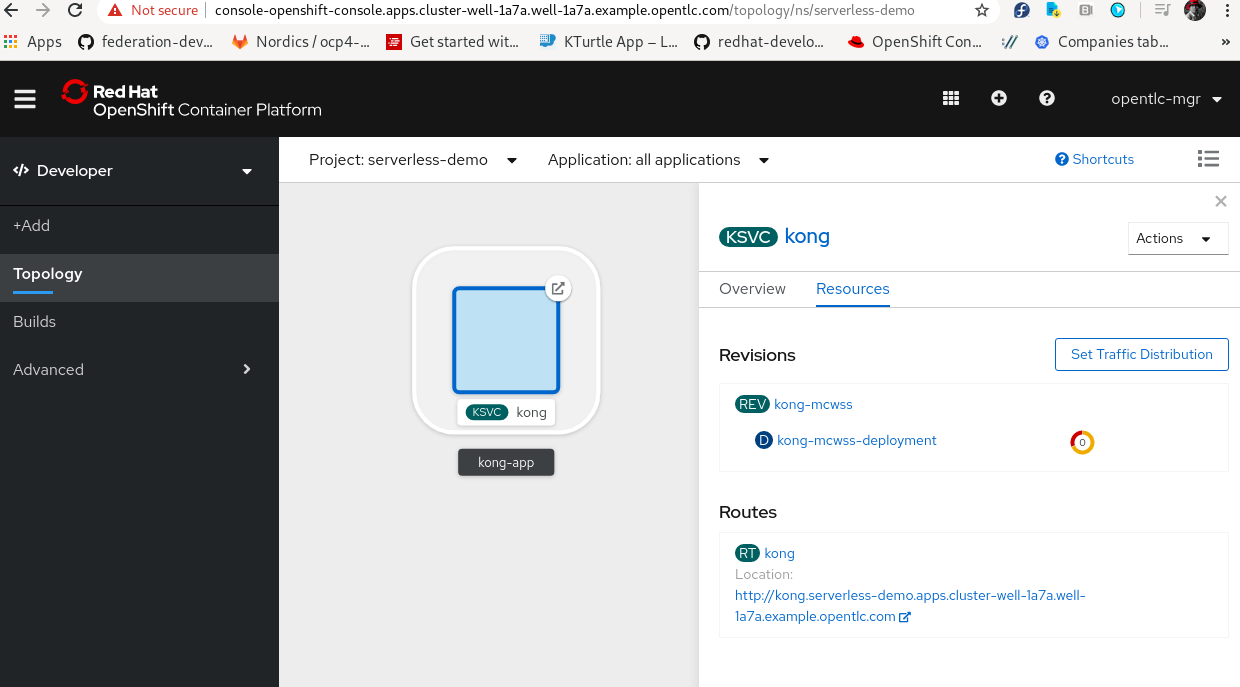
3. Select Knative Service as your resource type.



4. You can choose the number of pods as you Max number of pods when scaling up as part of you serverless option and click create.



4. You will automatically be directed to Topology view. You can then monitor the status of your deployment.



5. Once the container is running, you can test the application by selecting the routes and see if you can access exposed routes. If you are succesful, you can wait for a few minutes and see if the pods will scale to zero. Once the pods has scaled to zero, hit the routes again and monitor if the pods are re-created by the knative serving. Once you see it re-created, your demo is a success!!

**References:**

<https://blog.openshift.com/knative-serving-your-serverless-services/>

<https://github.com/tnscorcoran/openshift-servicemesh>