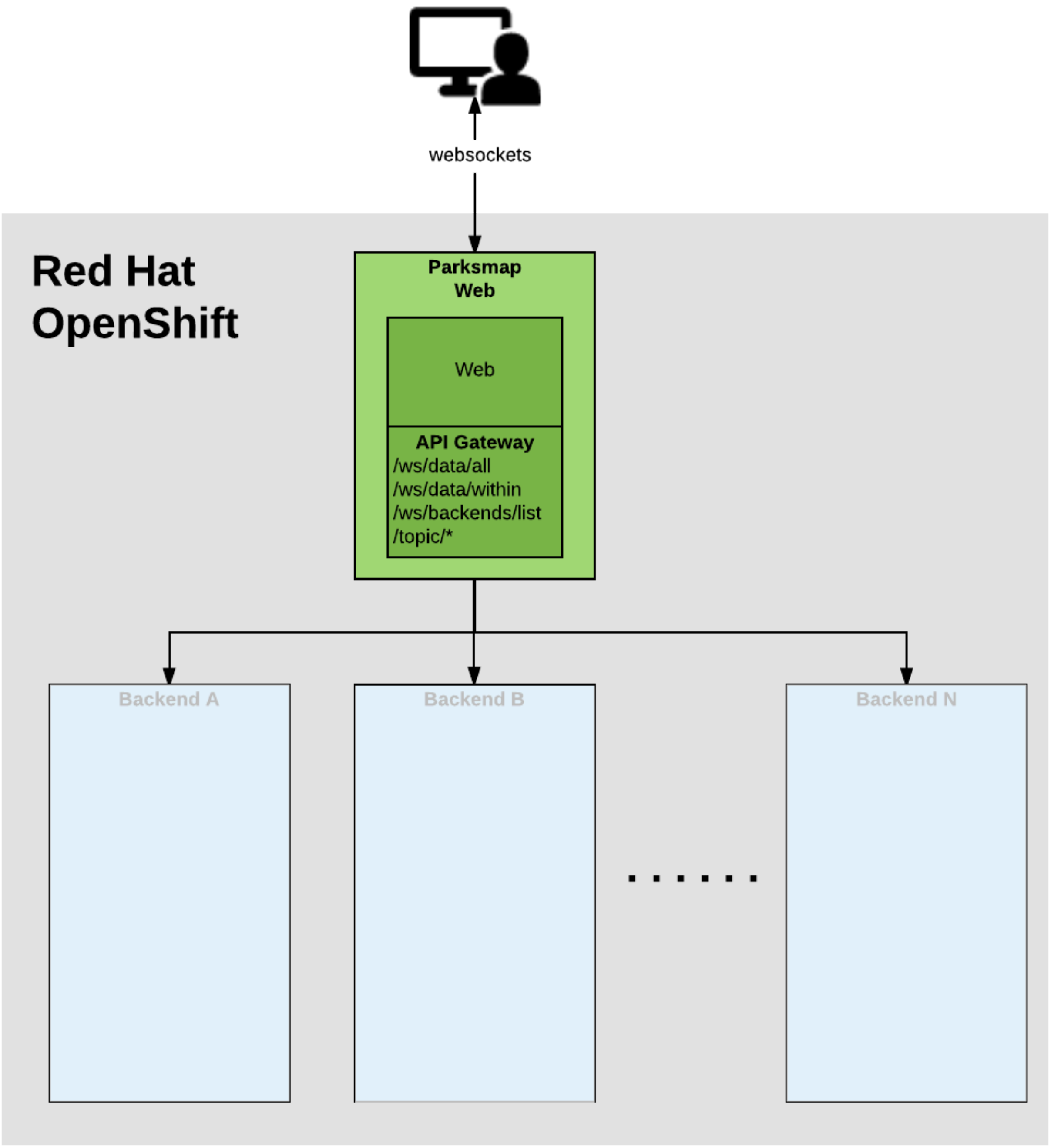
Starter Labs (Python)

WORKSHOP MODULES

Exposing Your Application to the Outside World

In this lab, we’re going to make our application visible to the end users, so they can access it.



websockets
Red Hat OpenShift
Parksmap Web
Web
API Gateway
/ws/data/all
/ws/data/within
/ws/backends/list
/topic/*
Backend A
Backend B
......
Backend N

Background: Routes

While **Services** provide internal abstraction and load balancing within an OpenShift environment, sometimes clients (users, systems, devices, etc.) **outside** of OpenShift need to access an application. The way that external clients are able to access applications running in OpenShift is through the OpenShift routing layer. And the data object behind that is a **Route**.

The default OpenShift router (HAProxy) uses the HTTP header of the incoming request to determine where to proxy the connection. You can optionally define security, such as TLS, for the **Route**. If you want your **Services**, and, by extension, your **Pods**, to be accessible from the outside world, you need to create a **Route**.

Exercise: Creating a Route

You may remember that when we deployed the `parksmap` application, we un-checked the checkbox to create a **Route**. Normally it would have been created for us automatically. Fortunately, creating a **Route** is a pretty straight-forward process. You simply `expose` the **Service** via the command line or via the **Administrator Perspective**.

Creating a route using the Web Console

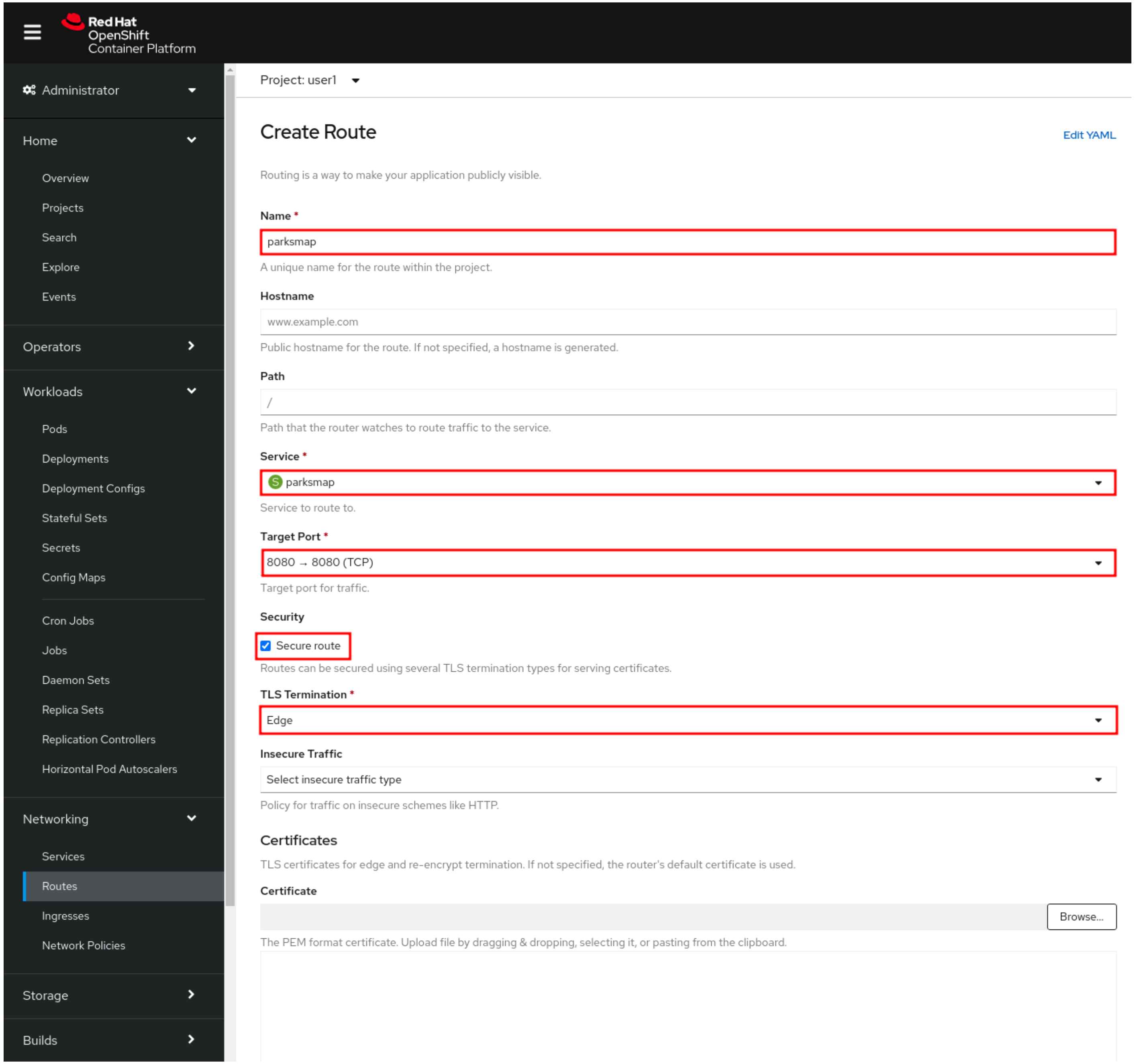
In the **Administrator Perspective** click **Networking** → **Routes** and then the **Create Route** button.

Insert **parksmap** in **Name** field.

From **Service** field, select **parksmap**. For **Target Port**, select **8080**.

In **Security** section, check **Secure route**. Select **Edge** from **TLS Termination** list.

Leave all other fields blank and click **Create**:



Create Route

Routing is a way to make your application publicly visible.

Name *
parksmap
A unique name for the route within the project.

Hostname
www.example.com
Public hostname for the route. If not specified, a hostname is generated.

Path
/
Path that the router watches to route traffic to the service.

Service *
S parksmap
Service to route to.

Target Port *
8080 → 8080 (TCP)
Target port for traffic.

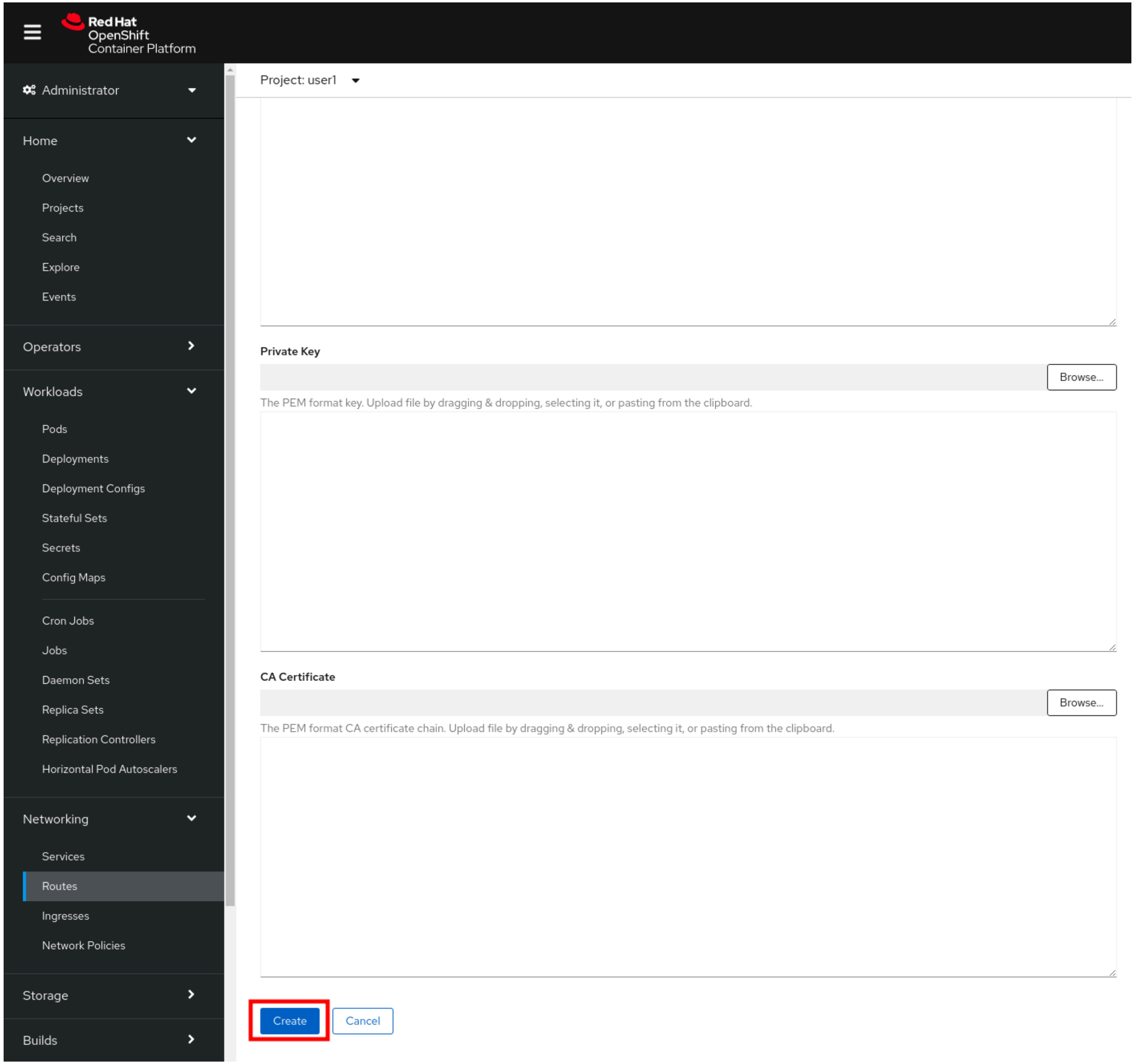
Security
☑ Secure route
Routes can be secured using several TLS termination types for serving certificates.

TLS Termination *
Edge

Insecure Traffic
Select insecure traffic type
Policy for traffic on insecure schemes like HTTP.

Certificates
TLS certificates for edge and re-encrypt termination. If not specified, the router's default certificate is used.

Certificate
The PEM format certificate. Upload file by dragging & dropping, selecting it, or pasting from the clipboard.



Project: user1

Private Key

Browse...

The PEM format key. Upload file by dragging & dropping, selecting it, or pasting from the clipboard.

CA Certificate

Browse...

The PEM format CA certificate chain. Upload file by dragging & dropping, selecting it, or pasting from the clipboard.

When creating a **Route**, some other options can be provided, like the hostname and path for the **Route** or the other TLS configurations.

Creating a route using the command line

When using the command line, we can first verify that we don’t already have any existing **Routes**:

```
oc get routes
```

```
No resources found.
```

Now we need to get the **Service** name to expose:

```
oc get services
```

```
NAME       CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
parksmap   172.30.169.213   <none>        8080/TCP   5h
```

Once we know the **Service** name, creating a **Route** is a simple one-command task:

```
oc create route edge parksmap --service=parksmap
```

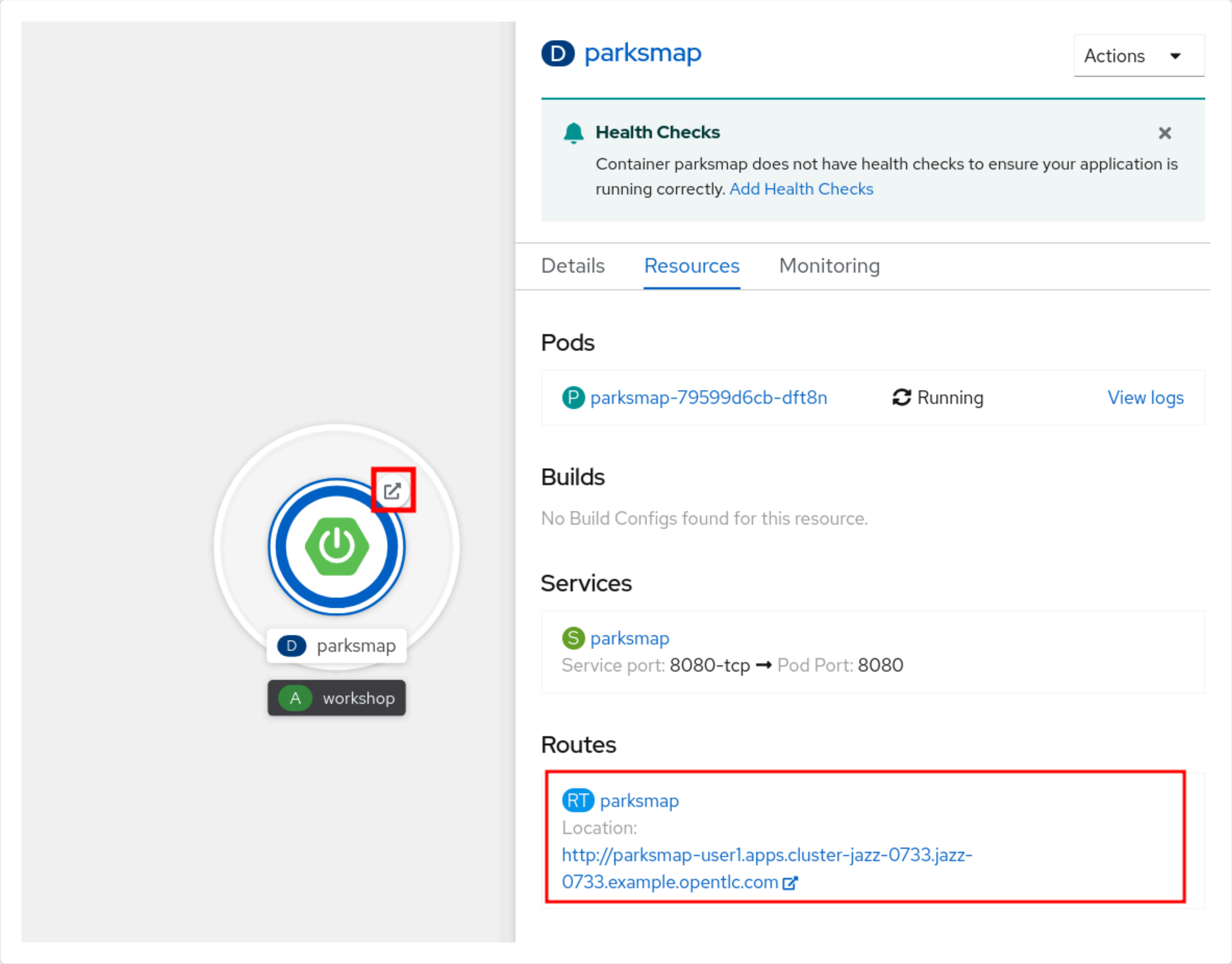
```
route.route.openshift.io/parksmap exposed
```

Verify the **Route** was created with the following command:

```
oc get route
```

```
NAME       HOST/PORT                            PATH      SERVICES   PORT       TERMINATION   WILDCARD
parksmap   parksmap-user4.apps.rosa-7s42b.rfax.p1.openshiftapps.com             parksmap   8080-tcp
edge          None
```

You can also verify the **Route** in the **Developer Perspective** under the **Resources** tab for your `parksmap` deployment configuration. Also note that there is a decorator icon on the `parksmap` visualization now. If you click that, it will open the URL for your **Route** in a browser.



D parksmap

Health Checks

Container parksmap does not have health checks to ensure your application is running correctly. Add Health Checks

Pods

P parksmap-79599d6cb-dft8n Running View logs

Builds

No Build Configs found for this resource.

Services

S parksmap

Service port: 8080-tcp → Pod Port: 8080

Routes

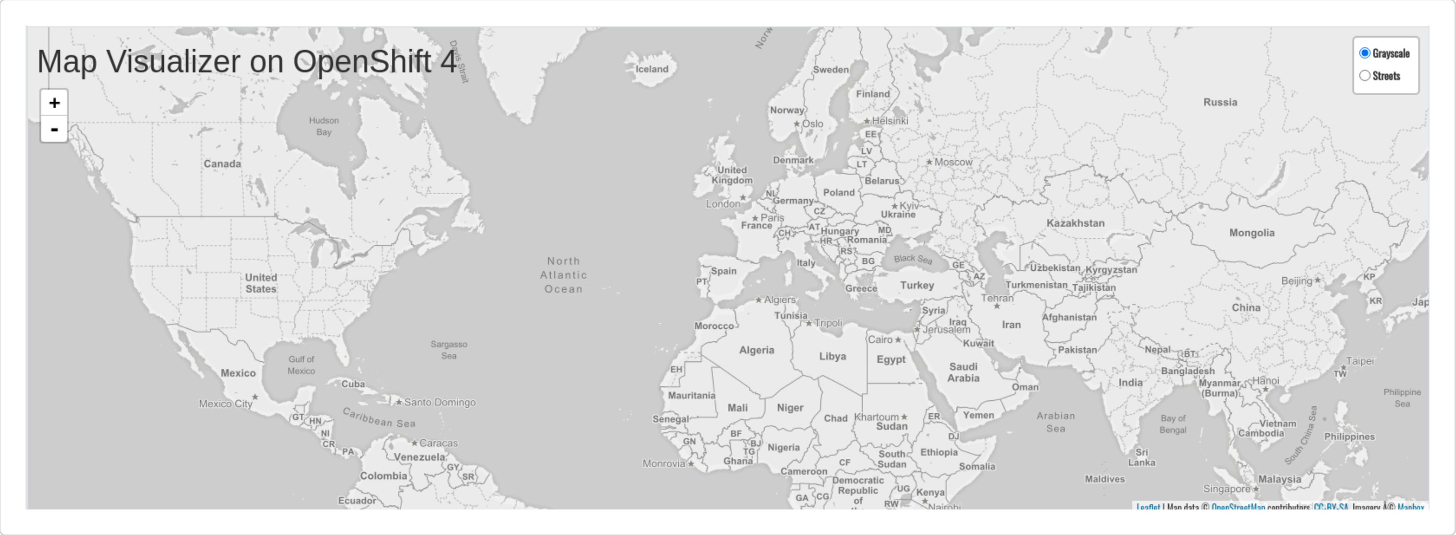
RT parksmap

Location:

http://parksmap-user1.apps.cluster-jazz-0733.jazz-0733.example.opentlc.com

This application is now available at the URL shown in the Developer Perspective. Click the link and you will see it.

> At first time, the Browser will ask permission to get your position. This is needed by the Frontend app to center the world map to your location, if you don’t allow it, it will just use a default location.



Map Visualizer on OpenShift 4

Continue