

Starter Labs (Python)

WORKSHOP MODULES

Workshop Summary

Environment Overview

Using Homeroom

Architecture Overview of the
ParksMap Application

Exploring the CLI and Web Console

Deploying Your First Container Image

Scaling and Self Healing

Exposing Your Application to the
Outside World

Exploring OpenShift's Logging
Capabilities

Role-Based Access Control

Remote Access to Your Application

Deploying Python Code

Adding a Database (MongoDB)

Application Health

Automate Build and Deployment with
Pipelines

Automation for Your Application on
Code Changes

Further Resources

Workshop Links

Role-Based Access Control

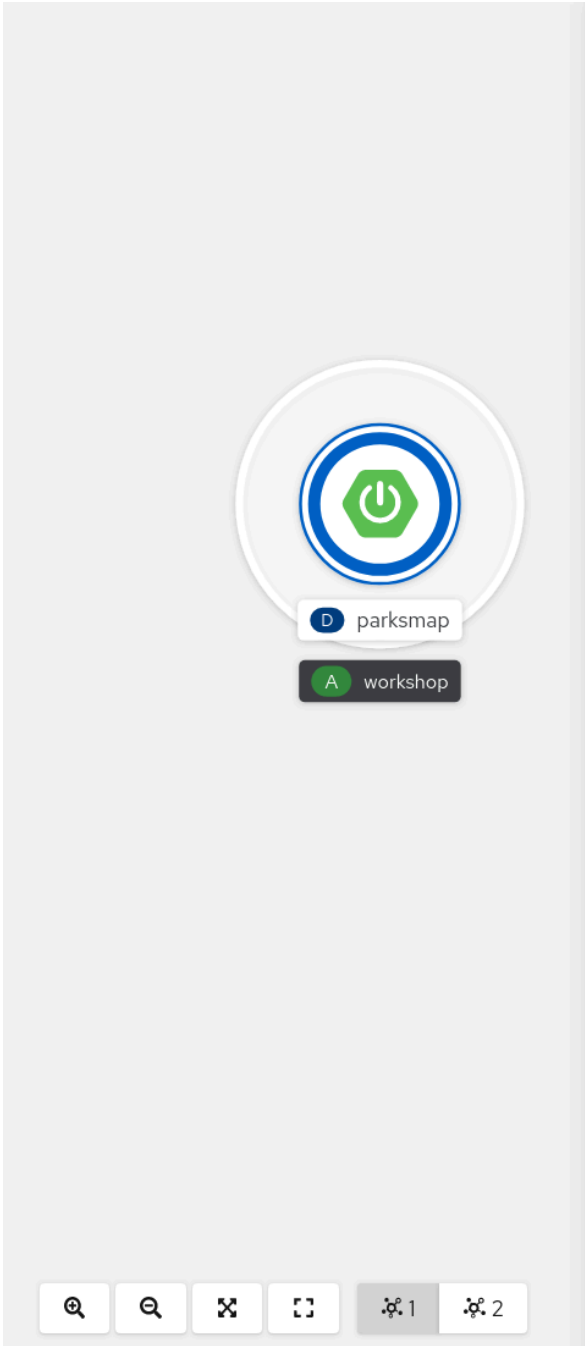
Almost every interaction with an OpenShift environment that you can think of requires going through the OpenShift's control plane API. All API interactions are both authenticated (AuthN - who are you?) and authorized (AuthZ - are you allowed to do what you are asking?).

In the log aggregation lab we saw that there was an error in reference to a **Service Account**.

As OpenShift is a declarative platform, some actions will be performed by the platform and not by the end user (when he or she issues a command). These actions are performed using a **Service Account** which is a special type of `user` that the platform will use internally.

OpenShift automatically creates a few special service accounts in every project. The **default** service account is the one taking the responsibility of running the pods, and OpenShift uses and injects this service account into every pod that is launched. By changing the permissions for that service account, we can do interesting things.

You can view current permissions in the web console, go to the Topology view in the Developer Perspective, click the `parksmap` entry, go to the **Details** tab, and then click the **Namespace**.



parksmap

workshop

1 pod

app=workshop

app.kubernetes.io/com...=parks...

app.kubernetes.io/inst...=parks...

app.kubernetes.io/par...=worksh...


app.openshift.io/run...=spring-b...

app.openshift.io/runtime-n...=us...


component=parksmap

role=frontend

1 2

 parksmap

Actions ▾


 **Health Checks** ✕

Container parksmap does not have health checks to ensure your application is running correctly. [Add Health Checks](#)

Details

Resources


Monitoring

 1 pod

Name

parksmap

Namespace

 user1

Labels

app=workshop

app.kubernetes.io/com...=parks...

app.kubernetes.io/inst...=parks...

app.kubernetes.io/par...=worksh...


app.openshift.io/run...=spring-b...

app.openshift.io/runtime-n...=us...

component=parksmap

role=frontend

Pod Selector

 app=parksmap

Node Selector

Update Strategy

RollingUpdate

Max Unavailable

25% of 1 pod

Max Surge

25% greater than 1 pod

Progress Deadline Seconds

600 seconds

Min Ready Seconds

Not Configured

https://lab-getting-started-python-labs.apps.rosa-7s42b.rfax.p1.openshiftapps.com/user/user4/dashboard/

2/9

Then, click **Role Bindings**.

[Namespaces](#) > [Namespace Details](#)

NS user1 ✓ Active

[Details](#) [YAML](#) [Role Bindings](#)

Create Binding

Filter

Name

Search by name...

Name ↑	Role Ref ↓	Subject Kind ↓	Subject Name ↓	Namespace ↓	
RB admin	CR admin	User	user1	NS user1	⋮
RB edit	CR edit	ServiceAccount	pipeline	NS user1	⋮
RB system:deployers	CR system:deployer	ServiceAccount	deployer	NS user1	⋮
RB system:image-builders	CR system:image-builder	ServiceAccount	builder	NS user1	⋮
RB system:image-puller	CR system:image-puller	Group	system:serviceaccount:s:user1	NS user1	⋮
RB system:image-pullers	CR system:image-puller	Group	system:serviceaccount:s:user1	NS user1	⋮

Exercise: Grant Service Account View Permissions

The parksmap application wants to talk to the OpenShift API to learn about other **Pods**, **Services**, and resources within the **Project**. You'll soon learn why!

```
oc project user4
```

Then:

```
oc policy add-role-to-user view -z default
```

The `oc policy` command above is giving a defined *role* (`view`) to a user. But we are using a special flag, `-z`. What does this flag do? From the `-h` output:

```
-z, --serviceaccount=[]: service account in the current namespace to use as a user
```

The `-z` syntax is a special one that saves us from having to type out the entire string, which, in this case, is `system:serviceaccount:user4:default`. It's a nifty shortcut.

The `-z` flag will only work for service accounts in the **current** project. If you're referring to a service account in a different project, use the ``-n <project>`` switch.

Now that the `default` **Service Account** now has **view** access, so now it can query the API to see what resources are within the **Project**. This also has the added benefit of suppressing the error message! Although, in reality, we fixed the application.

Another way you could have done the same is by using the OpenShift console. Once you're on the **Workloads** → **Deployments** page, click on the **Namespace**, then **Role Bindings** and then the **Create Binding** button.

[Namespaces](#) > Namespace Details

NS user1

Active

Details

YAML

Role Bindings

Create Binding

Filter

Name

Search by name...

Name ↑	Role Ref ↓	Subject Kind ↓	Subject Name ↓	Namespace ↓	
RB admin	CR admin	User	user1	NS user1	⋮
RB edit	CR edit	ServiceAccount	pipeline	NS user1	⋮

Select **view** for the Role Binding Name **user4** for the Namespace, **view** for the Role Name, **Service Account** for the Subject, **user4** for the Subject Namespace, and **default** for the Subject Name.

Create Role Binding

Associate a user/group to the selected role to define the type of access and resources that are allowed.

Role Binding

Name *

view

Namespace *

PR user1

Role

Role Name *

CR view

Subject

- ☐ User
- ☐ Group
- ☒ Service Account

Subject Namespace *

PR user1

Subject Name *

default

Create

Cancel

Once you're finished editing permissions, click on the **Create** button.

[Namespaces](#) > Namespace Details

NS user1 ✓ Active

[Details](#) [YAML](#) [Role Bindings](#)

Create Binding

Filter

Name

Search by name...

Name ↑	Role Ref ↑	Subject Kind ↑	Subject Name ↑	Namespace ↑	
RB admin	CR admin	User	user1	NS user1	⋮
RB edit	CR edit	ServiceAccount	pipeline	NS user1	⋮
RB system:deployers	CR system:deployer	ServiceAccount	deployer	NS user1	⋮
RB system:image-builders	CR system:image-builder	ServiceAccount	builder	NS user1	⋮
RB system:image-puller	CR system:image-puller	Group	system:serviceaccount s:user1	NS user1	⋮
RB system:image-pullers	CR system:image-puller	Group	system:serviceaccount s:user1	NS user1	⋮
RB view	CR view	ServiceAccount	default	NS user1	⋮

Exercise: Redeploy Application

One more step is required. We need to re-deploy the `parksmap` application because it's given up trying to query the API.

```
oc rollout restart deployment/parksmap
```

A new deployment is immediately started. Return to Topology view and click the `parksmap` entry again to watch it happen. You might not be fast enough! But it will be reflected in the **ReplicaSet** number.

The screenshot shows the OpenShift dashboard interface. At the top, there are filters for 'Project: user1' and 'Application: all applications', along with a 'View shortcuts' link. Below these are 'Display Options', 'Filter by Resource', and a search bar 'Find by name...'. The main content area is divided into two parts. On the left is a topology view showing a circular icon with a green power button symbol, labeled 'D parksmap' and 'A workshop'. On the right is a detailed view for the 'parksmap' deployment. It includes a 'Health Checks' section with a warning icon and text: 'Container parksmap does not have health checks to ensure your application is running correctly. [Add Health Checks](#)'. Below this are tabs for 'Details', 'Resources', and 'Monitoring'. The 'Details' tab is active, showing a circular progress indicator with '0 scaling to 1'. Below the indicator are several key-value pairs: 'Name: parksmap', 'Update Strategy: RollingUpdate', 'Namespace: NS user1', 'Max Unavailable: 25% of 1 pod', 'Labels: app=workshop, app.kubernetes.io/com...=parks...', and 'Max Surge: 25% greater than 1 pod'. There is also an 'Edit' button next to the labels.

If you look at the logs for the application now, you should see no errors. That's great.

(Optional) Exercise: Grant User View Permissions

If you create a project, you are that project's administrator. This means that you can grant access to other users, too. If you like, give your neighbor view access to your project using the following command:

In the following command(s), replace `user4` with the user name of the person to whom you want to grant access.

```
oc policy add-role-to-user view user4
```

Have them go to the project view by clicking the **Projects** button and verify that they can see your project and its resources. This type of arrangement (view but not edit) might be ideal for a developer getting visibility into a production application's project.

[Continue](#)