Application deployment Steps

Now post creating the basic development structure & pushing code to Git Repositories for your frontend & backend code, you need to deploy your applications to test it & perform further development.

Now please follow the below steps to deploy your developed applications which would also help in creating pipelines for your applications.

After performing the below steps for one time, you will be able to develop your application further on Red Hat Openshift Dev Spaces & use pipelines to trigger the deployment automatically to deploy your code.

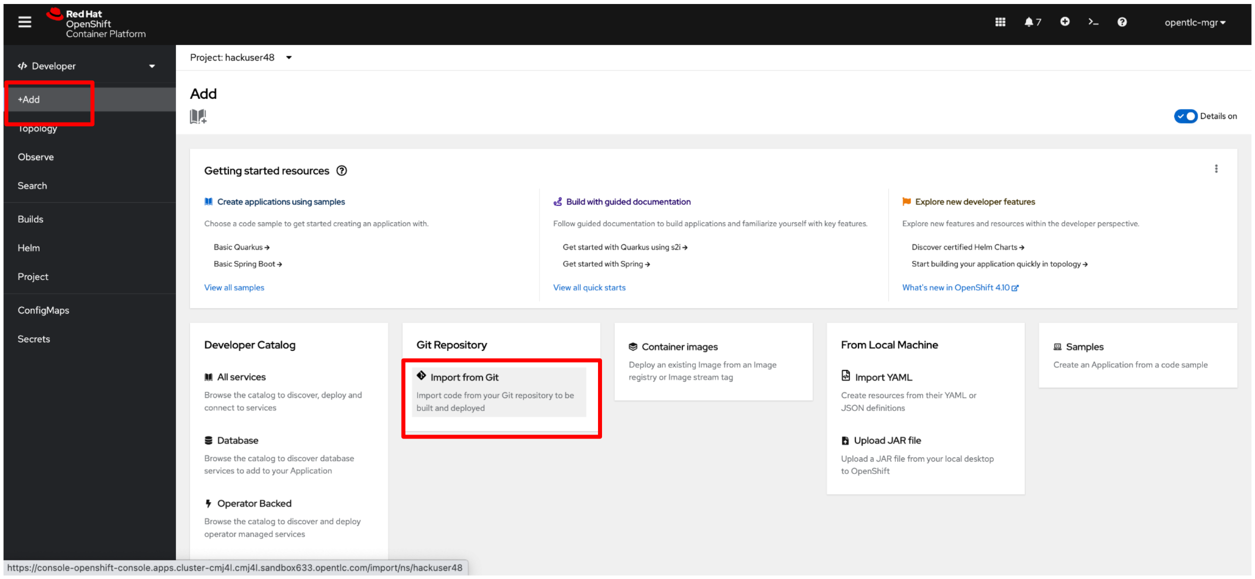
Note:

- In the below example , we are deploying Quarkus (Java) based backend application but the process will be same for your code language
- The difference will only be selecting the correct builder image as per your code language. Refer to screenshots below

Deploying applications on Openshift

In the below example, we are showcasing java based application deployment as an example. But the process would be the same for other applications as well other than selecting the right builder image.

Access the Openshift Console & from the Developer perspective on the OCP Web console, click on “+Add” and then select “Import from git”.



Choose a code sample to get started creating an application with.

Follow guided documentation to build applications and familiarize yourself with key features.

Explore new features and resources within the developer perspective.

Browse the catalog to discover, deploy and connect to services

Browse the catalog to discover database services to add to your Application

Browse the catalog to discover and deploy operator managed services

Import code from your Git repository to be built and deployed

Deploy an existing Image from an Image registry or Image stream tag

Create resources from their YAML or JSON definitions

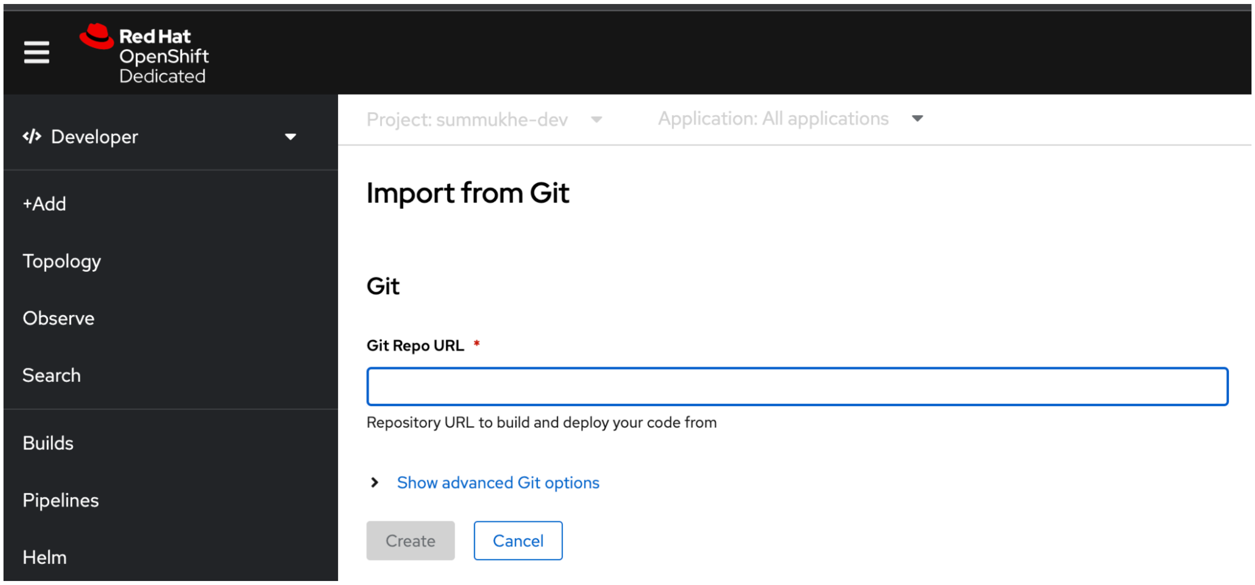
Upload a JAR file from your local desktop to OpenShift

Create an Application from a code sample

https://console-openshift-console.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com/import/ns/hackuser48

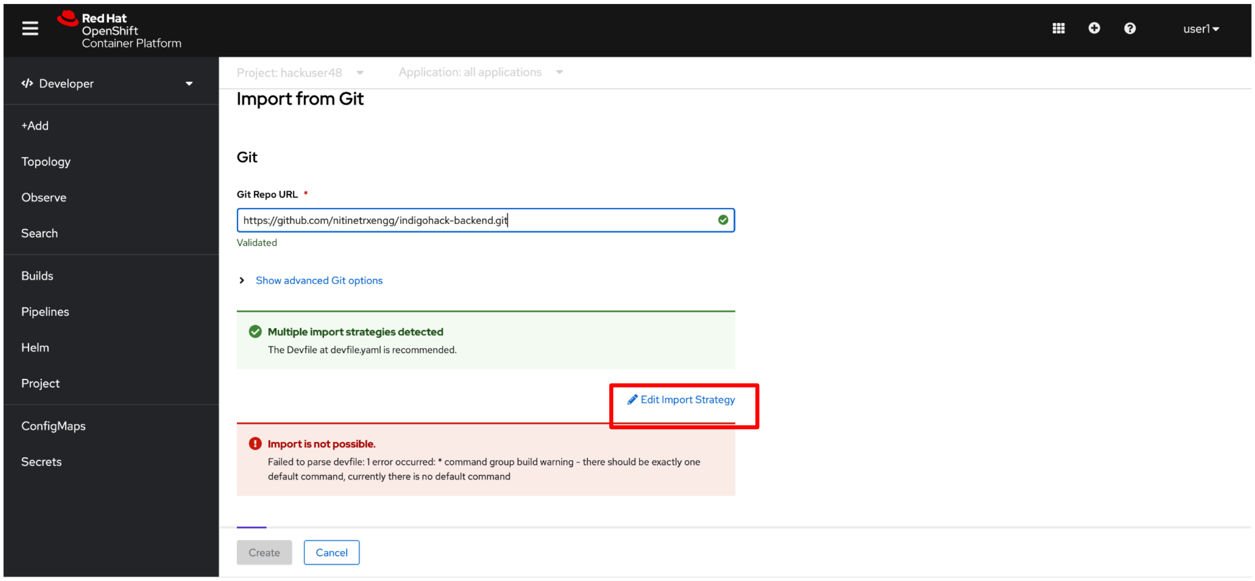
On the form that opens, fill the following details :

1. **Git repo URL** : Please enter your GitHub repository created for backend development (example - https://github.com/<yourGitHubUserName>/indigohack-backend.git)



Repository URL to build and deploy your code from

2. After providing the GitURL, it will give the below screen & click on "Edit Import Strategy" Link. Even if the screen does not come, we need to select the right builder image as per our code language mentioned in below point.



Import from Git

Git

Git Repo URL *

https://github.com/nitinetrxengg/indigohack-backend.git

Validated

Multiple import strategies detected

The Devfile at devfile.yaml is recommended.

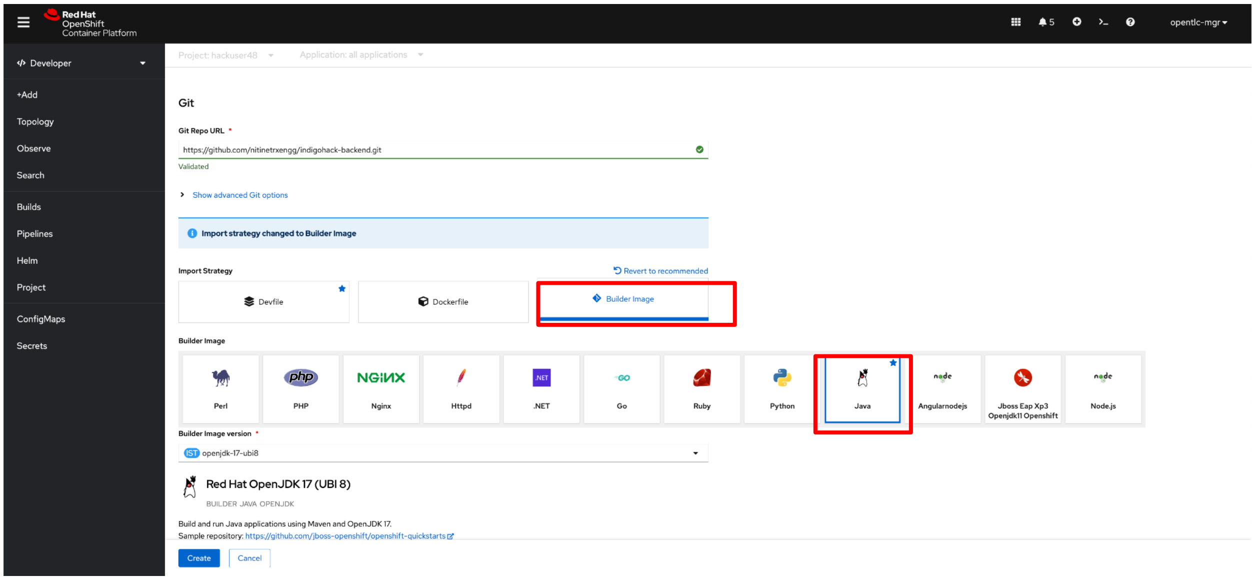
Import is not possible.

Failed to parse devfile: 1 error occurred: * command group build warning - there should be exactly one default command, currently there is no default command

3. Select the "Builder Image" option. It will remove the above error & populate other options. By default, it is showing Java as a builder image in this example as git repository have java code

Note: Ideally builder image will be auto selected depending on the the code language in GitHub repository but please make sure then you have to select the right builder image as per the following options

- .Net based backend code - Select .Net builder image
- Java based backend code - Select Java builder image
- Python based backend code - Select Python builder image
- Nodejs based backend code - Select Node.js builder image
- React based frontend code - Select Node.js builder image



Git

Git Repo URL *

https://github.com/nitinetrxengg/indigohack-backend.git

Validated

Show advanced Git options

Import strategy changed to Builder Image

Import Strategy

Revert to recommended

Devfile | Dockerfile | Builder Image

Builder Image

Builder Image version *

IST openjdk-17-ubi8

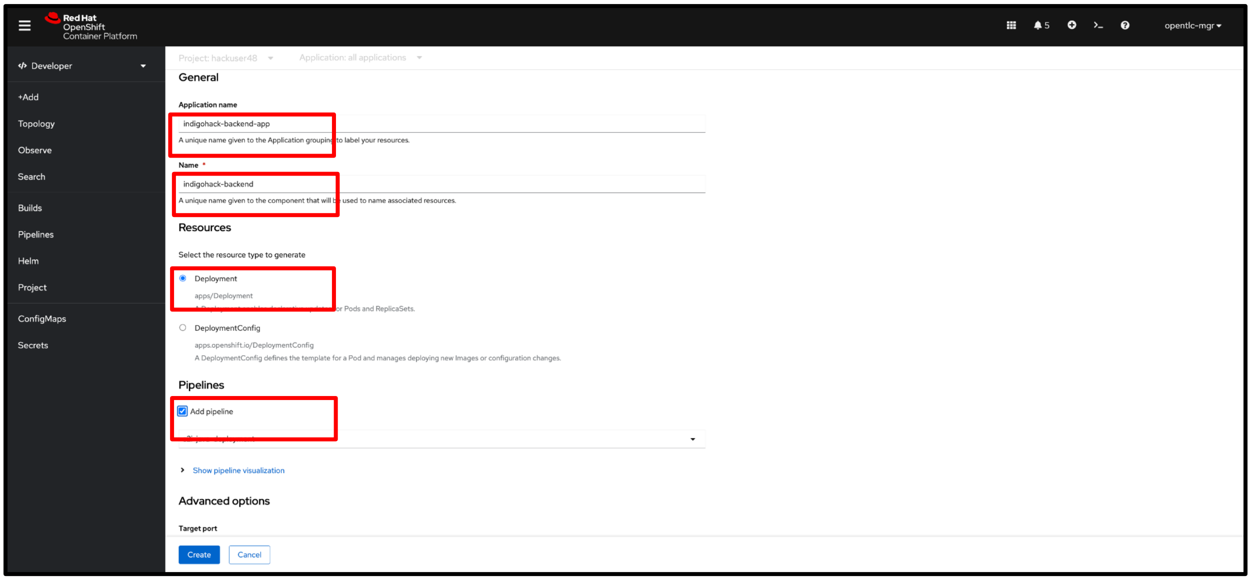
Red Hat OpenJDK 17 (UBI 8)

BUILDER JAVA OPENJDK

Build and run Java applications using Maven and OpenJDK 17.
Sample repository: https://github.com/jboss-openshift/openshift-quickstarts

Now scroll down & other fields of the form are visible. Fill them as below or use your values

4. **Application name** : indigohack-backend-app
5. **Name**: indigohack-backend (Note: This is the connection name that you need to code in the frontend code to talk to backend service.. Append this connection name with indigohack-backend & put this in your frontend code configuration)
6. **Resource Type** : Deployment
7. **Add Pipeline** : Check the select box
8. **Target Port :** 8080 (Note: We have used 8080 port in our example application as a port. Change this value as per the port that you have defined in your code to expose the application - This is the connection port that you need to code in the frontend code to talk to backend service)



General

Application name

indigohack-backend-app

A unique name given to the Application grouping to label your resources.

Name *

indigohack-backend

A unique name given to the component that will be used to name associated resources.

Resources

Select the resource type to generate

● Deployment

apps/Deployment

or Pods and ReplicaSets.

○ DeploymentConfig

apps.openshift.io/DeploymentConfig

A DeploymentConfig defines the template for a Pod and manages deploying new Images or configuration changes.

Pipelines

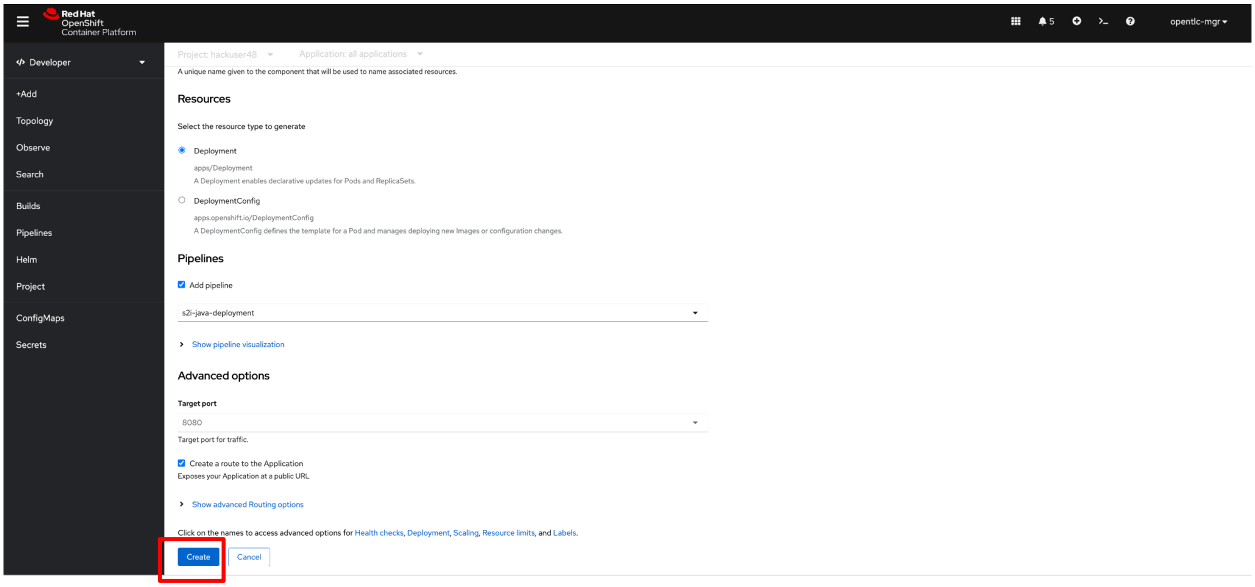
☑ Add pipeline

Show pipeline visualization

Advanced options

Target port

9. Leave other values to default. Once the form is filled, scroll to the bottom, click "Create" to deploy your application



A unique name given to the component that will be used to name associated resources.

Resources

Select the resource type to generate

● Deployment
apps/Deployment
A Deployment enables declarative updates for Pods and ReplicaSets.

○ DeploymentConfig
apps.openshift.io/DeploymentConfig
A DeploymentConfig defines the template for a Pod and manages deploying new Images or configuration changes.

Pipelines

☑ Add pipeline

s2i-java-deployment

› Show pipeline visualization

Advanced options

Target port

8080

Target port for traffic.

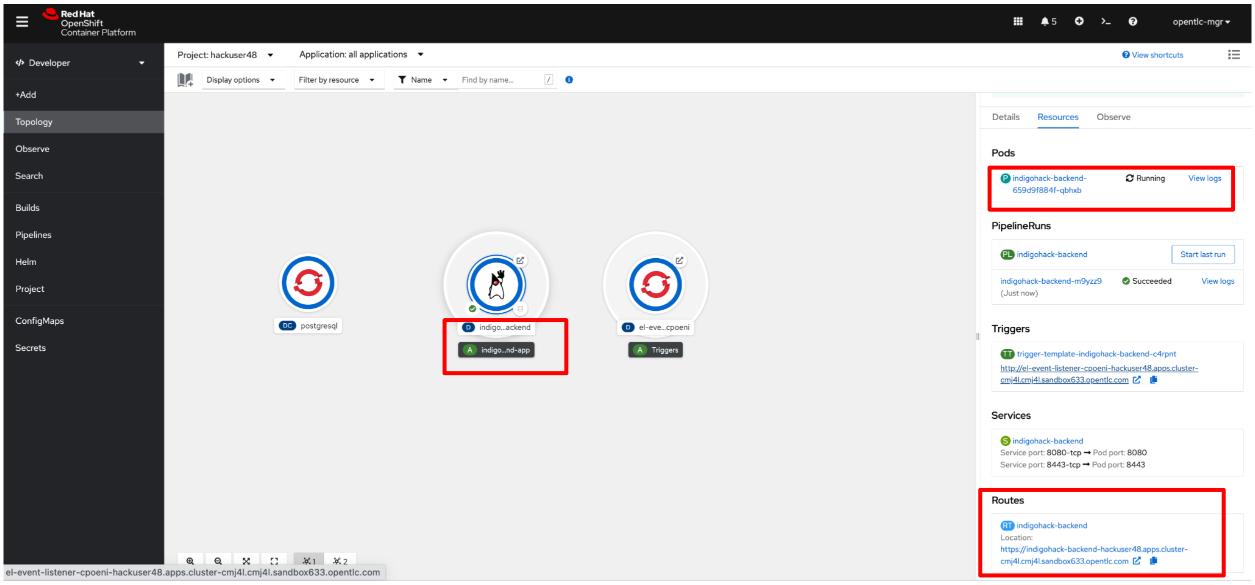
☑ Create a route to the Application
Exposes your Application at a public URL

› Show advanced Routing options

Click on the names to access advanced options for Health checks, Deployment, Scaling, Resource limits, and Labels.

Once the application is deployed, it should open up the "Topology" view from the Developer perspective as shown below. Click on Indigo backend Deployment. Initially the Pods will not be running in the Pods section. The deployment happens via the pipeline, so you need to wait till the PipelineRuns show complete and the Pod shows 1 entry as running.

After the pod shows as running, click on Route link in the screenshot to access the backend application endpoint



el-event-listener-cpoeni-hackuser48.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com

Pods

P indigohack-backend-659d9f884f-qbhxb Running View logs

PipelineRuns

PL indigohack-backend

indigohack-backend-m9yzz9 (Just now) Succeeded View logs

Triggers

TT trigger-template-indigohack-backend-c4rpnt

http://el-event-listener-cpoeni-hackuser48.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com

Services

S indigohack-backend

Service port: 8080-tcp ➞ Pod port: 8080

Service port: 8443-tcp ➞ Pod port: 8443

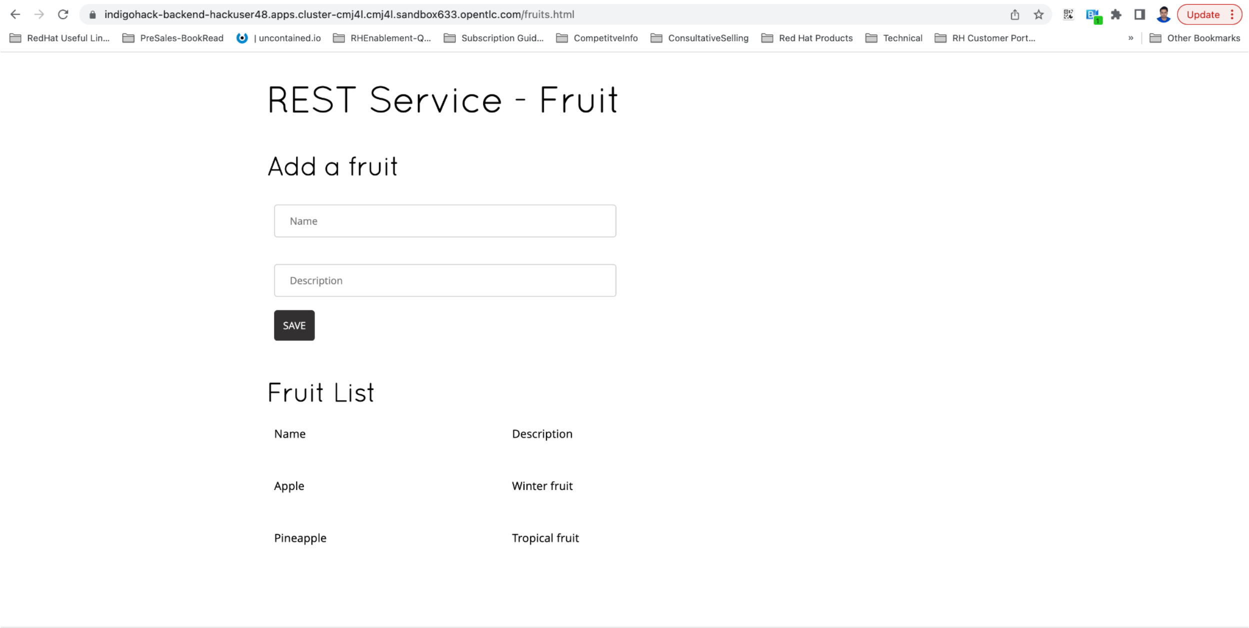
Routes

RT indigohack-backend

Location:

https://indigohack-backend-hackuser48.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com

It will open the Openshift Route to show the web page (or the base URL) of the application in your default browser. If your application has any extra context paths, you need to add it in the address bar in the browser. This concludes the deployment of your backend application.



indigohack-backend-hackuser48.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com/fruits.html

REST Service - Fruit

Add a fruit

Name

Description

SAVE

Fruit List

Name	Description
Apple	Winter fruit
Pineapple	Tropical fruit

Important Note: Follow the same procedure for deploying the frontend application. Click on Add+ in Openshift web console at developer perspective & follow the "Deploying Applications on Openshift" section but this time with frontend code GitHub Repository & provide the different application name & name for frontend code.

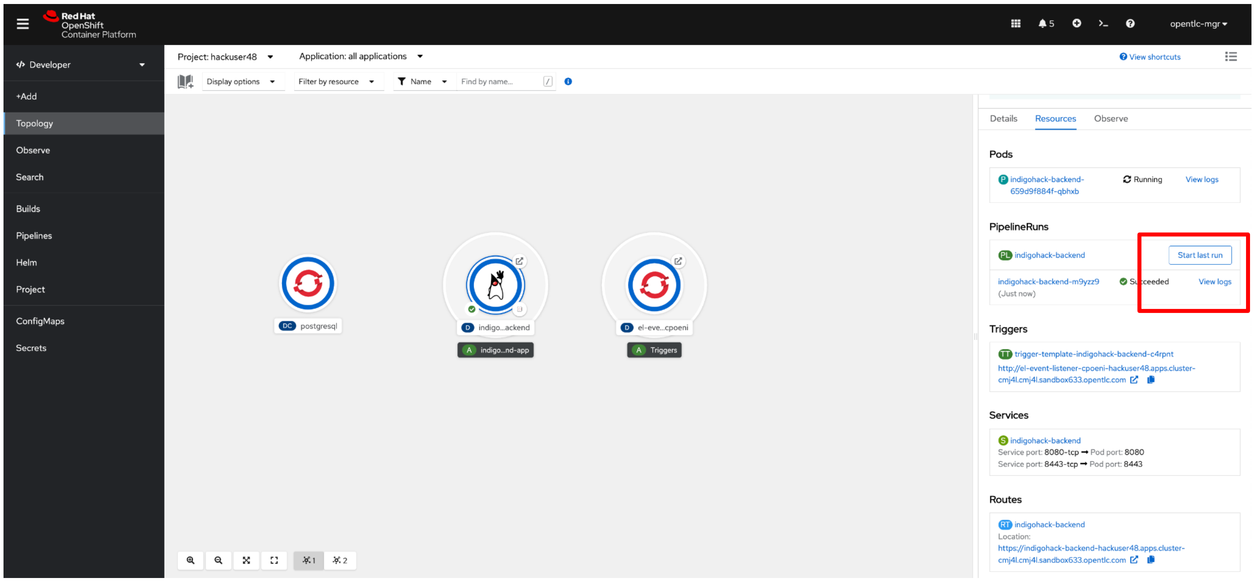
Triggering rebuild of applications via Pipeline

Now after performing the above steps, you have your initial first version of applications deployed from GitHub repos and Pipeline configurations created.

Now go back to your Red Hat Openshift Dev Spaces workspaces for frontend & backend & do further development of your code.

To redeploy the changes of your code, commit the code to GitHub & trigger the build from OCP web console.

To re-trigger a build of an application via the pipelines, in the Developer perspective, click on the Topology panel and then click on the application deployment (backend or frontend) which you want to trigger. Verify that on the right hand side, you can see the desired name of the application. In the below screenshot, the indigo-backend application is selected. Then click on the button labeled “Start last run”, as shown in the screenshot below. You can see that a new pipeline instance is running. You need to wait till the status is Succeeded.



Pods

P indigohack-backend-659d9f884f-qbhxb Running

PipelineRuns

PL indigohack-backend

indigohack-backend-m9yzz9 (Just now) Succeeded

Triggers

TT trigger-template-indigohack-backend-c4rpnt
http://el-event-listener-cpoeni-hackuser48.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com

Services

S indigohack-backend
Service port: 8080-tcp → Pod port: 8080
Service port: 8443-tcp → Pod port: 8443

Routes

RT indigohack-backend
Location:
https://indigohack-backend-hackuser48.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com