Liberty OpenAPI Lab 2: OpenAPI on the Cloud

Introduction

This lab demonstrates how to lift-and-shift a Liberty package with RESTful APIs into Bluemix to enable the same "API Discovery" on-prem environment we saw in the OpenAPI on Liberty Server lab, but on the cloud! The objective of this lab is to learn how APIs are deployed to and can be invoked in Liberty and Bluemix.

Business Scenario

Your team has created applications which other development teams in the company want to explore and integrate with. To facilitate a smooth integration, you are asked to run your application in the Bluemix cloud.

Prerequisites

- An IBM Bluemix account available and ready to use
    - If you do not have an account, go to https://console.ng.bluemix.net/ to create an account for free.
- Familiarity with Linux commands

Lab Overview

In this lab, you will

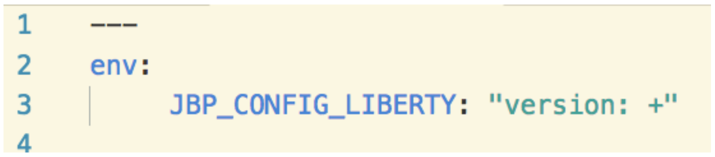
- Explore the Liberty Server Package
- Push the application to Bluemix
- Inspect the APIs in Bluemix

2.1 Explore the Liberty Server package that will be deployed to Bluemix

In this section, you will explore the application that will be pushed to Bluemix. We have already provided the application and server configurations for you.

1. Locate the Liberty Package server used in this lab
   a) Cd into the **cascon** lab folder where the **defaultServer** package is located

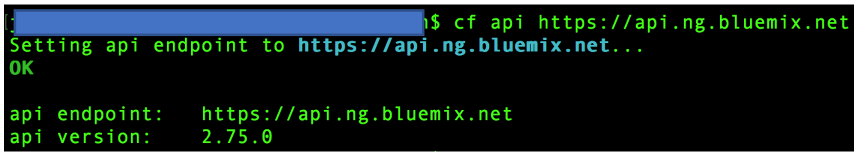
2. Understand the lab artifacts.
   a) The **manifest.yml** file lets us set up some environment variables that allow us to use a custom buildpack.



```
---
env:
    JBP_CONFIG_LIBERTY: "version: +"

```

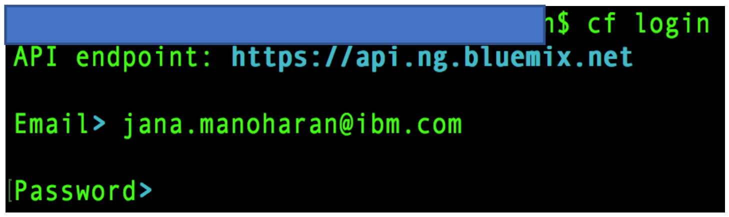
   b) The **defaultServer** folder is the package that will be pushed in to Bluemix later in the lab. It contains the server configurations for the application, and the application itself in the form of a **.war** file

3. Set an API Endpoint to the with the endpoint for your region and login to Bluemix
   a) In the command line type in **`cf api https://api.ng.bluemix.net`**



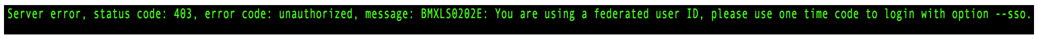
```
[redacted]$ cf api https://api.ng.bluemix.net
Setting api endpoint to https://api.ng.bluemix.net...
OK

api endpoint:   https://api.ng.bluemix.net
api version:    2.75.0
```

   b) Login to Bluemix using the command **`cf login`** and enter your Bluemix credentials

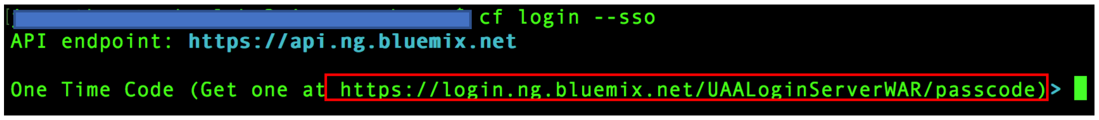


```
[redacted]$ cf login
API endpoint: https://api.ng.bluemix.net

Email> jana.manoharan@ibm.com

Password>
```

      i. If you get a 403 error saying you are a federated user ID, use this command to login: **`cf login --sso`**, then go to the link provided and login with your user id to get your password.

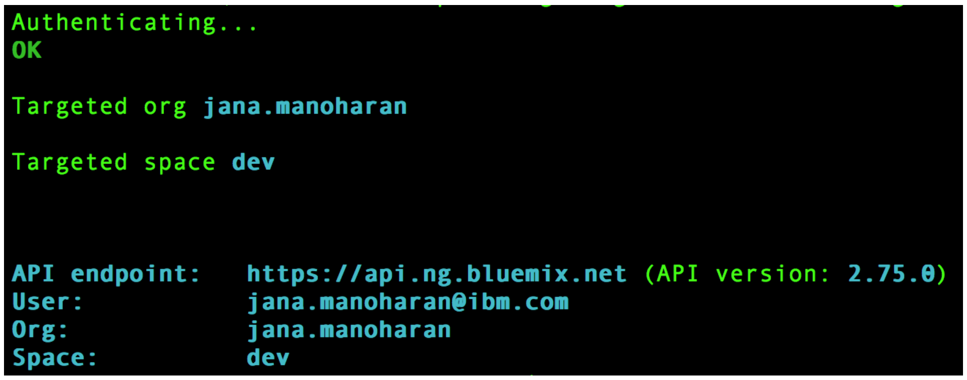


```
Server error, status code: 403, error code: unauthorized, message: BMXLS0202E: You are using a federated user ID, please use one time code to login with option --sso.
```



```
[redacted] cf login --sso
API endpoint: https://api.ng.bluemix.net

One Time Code (Get one at https://login.ng.bluemix.net/UAALoginServerWAR/passcode)>
```

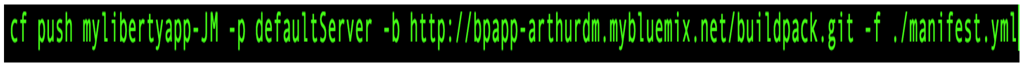
c) At this point, you are logged in to Bluemix and you are ready to push the liberty package into Bluemix.



```
Authenticating...
OK

Targeted org jana.manoharan

Targeted space dev


API endpoint:   https://api.ng.bluemix.net (API version: 2.75.0)
User:           jana.manoharan@ibm.com
Org:            jana.manoharan
Space:          dev
```

d) The final step is to push the Liberty server to Bluemix.

   i. Use this command to push, where AAA is your initials. This is to ensure that the application name is unique
   `cf push mylibertyapp-AAA -p defaultServer -b http://bpapp-arthurdm.mybluemix.net/buildpack.git -f ./manifest.yml`



```
cf push mylibertyapp-JM -p defaultServer -b http://bpapp-arthurdm.mybluemix.net/buildpack.git -f ./manifest.yml
```

   ii. It may take about 1 – 2 minutes for the server to be pushed to Bluemix. While this is happening, take this time to understand the command you just executed.

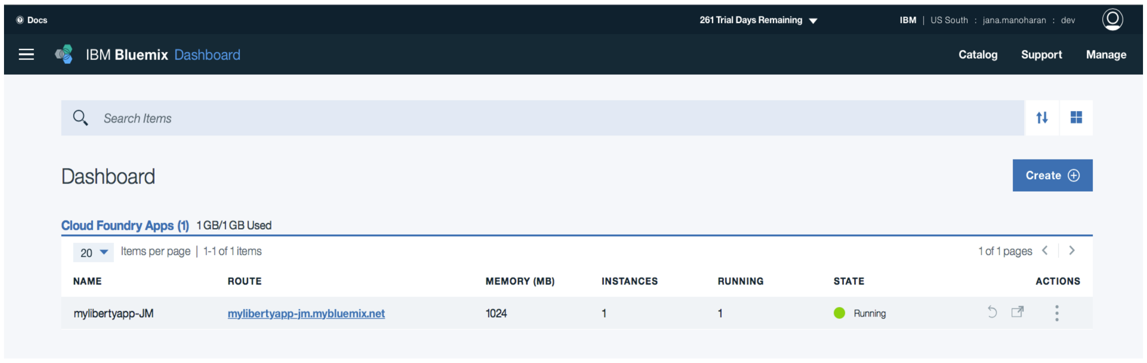
      1. **`cf push mylibertyapp-AAA`** – this will push a Cloud Foundry app to Bluemix
      2. **`-p defaultServer`** – this argument lets us specify the package we want to push. In the **defaultServer** package, the server configurations (**server.xml**) and the application to push (**apps/airlines.war**) is included
      3. **`-b http://bpapp-arthurdm.mybluemix.net/buildpack.git`** - this argument allows us to specify which buildpack to use for the app. In this case, we are using a custom Liberty buildpack which includes the OpenAPI feature.
      4. **`-f ./manifest.yml`** – in the manifest.yml, environment variables can be set for the application. In the file, we have specified to use the custom Liberty buildpack for all the configurations.

4. Investigate the application in Bluemix and play with the APIs

   a) Open a web browser and enter this URL: https://console.ng.bluemix.net/

   b) Login in to Bluemix with your credentials

c) You will be directed to the Bluemix Dashboard in which your apps are displayed.



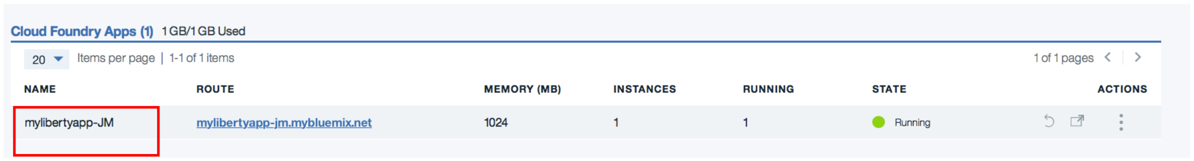
Dashboard

Cloud Foundry Apps (1) 1 GB/1 GB Used

20 Items per page | 1-1 of 1 items   1 of 1 pages

NAME	ROUTE	MEMORY (MB)	INSTANCES	RUNNING	STATE	ACTIONS
mylibertyapp-JM	mylibertyapp-jm.mybluemix.net	1024	1	1	Running	

d) Click on the app that you just deployed to investigate it

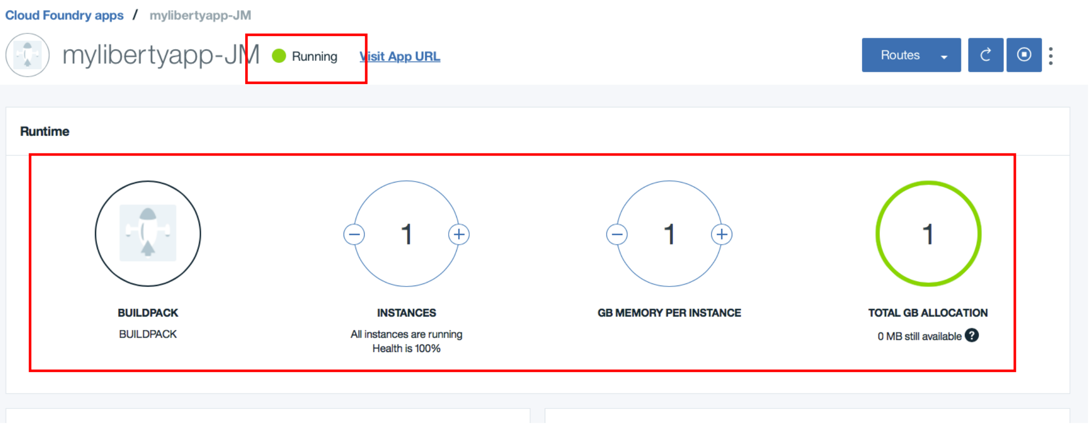


Cloud Foundry Apps (1) 1 GB/1 GB Used

20 Items per page | 1-1 of 1 items   1 of 1 pages

NAME	ROUTE	MEMORY (MB)	INSTANCES	RUNNING	STATE	ACTIONS
mylibertyapp-JM	mylibertyapp-jm.mybluemix.net	1024	1	1	Running	

e) You will be presented with some general information about the application. From this UI, you can see that the application is running. Moreover, the application is running on the custom buildpack we specified, with one instance using 1 gigabyte of memory.



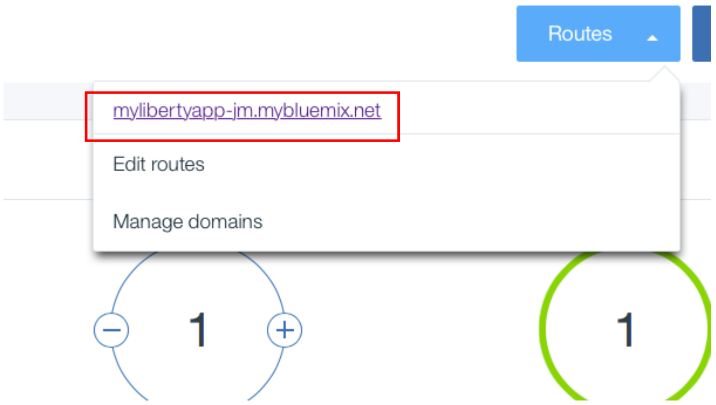
BUILDPACK: BUILDPACK

INSTANCES: All instances are running, Health is 100%

GB MEMORY PER INSTANCE

TOTAL GB ALLOCATION: 0 MB still available

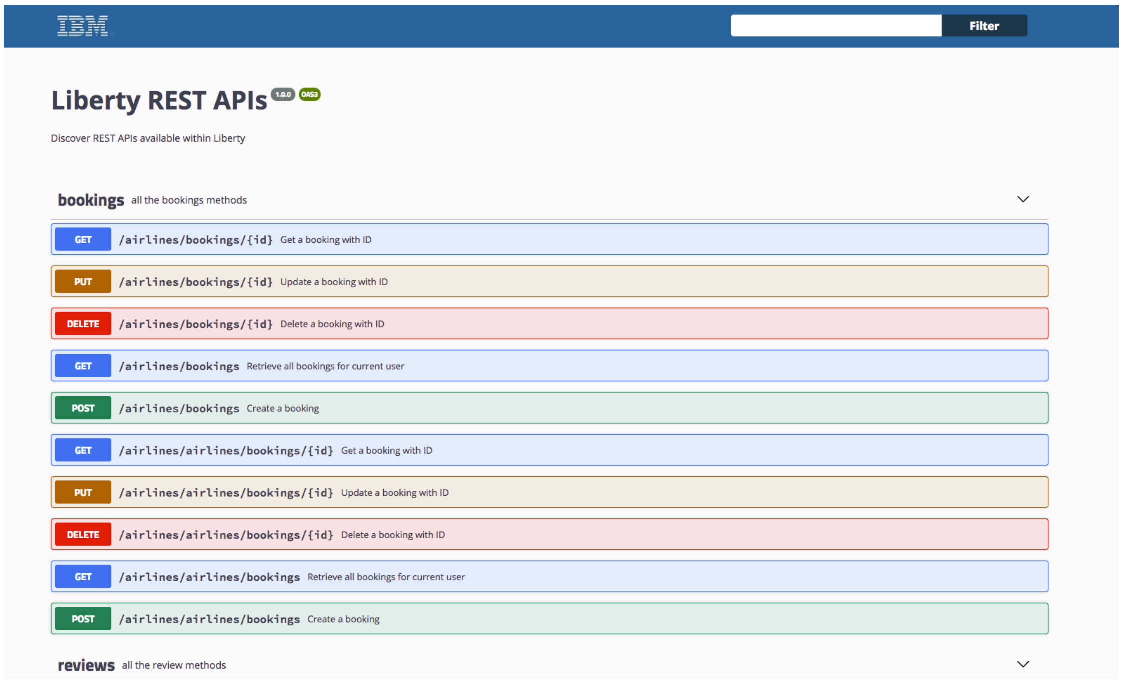
f) Expand the **Routes** button. Bluemix will automatically generate the route for you. It will use the lower case of the application name provided at the push command (**mylibertyapp-AAA**) and append "**.bluemix.net**". Click on it to visit the app.



g) This application is only built with an API, for this reason, it does not have anything in the context root of the application. To view the API Explorer, go to the following URL, replacing your initials for the **AAA** in the URL:

https://mylibertyapp-AAA.mybluemix.net/api/explorer/

h) As you have seen before in **Eclipse**, the application contains the APIs for the **Airlines App** from **Lab 1**.



Liberty REST APIs

Discover REST APIs available within Liberty

bookings all the bookings methods

GET	`/airlines/bookings/{id}`	Get a booking with ID
PUT	`/airlines/bookings/{id}`	Update a booking with ID
DELETE	`/airlines/bookings/{id}`	Delete a booking with ID
GET	`/airlines/bookings`	Retrieve all bookings for current user
POST	`/airlines/bookings`	Create a booking
GET	`/airlines/airlines/bookings/{id}`	Get a booking with ID
PUT	`/airlines/airlines/bookings/{id}`	Update a booking with ID
DELETE	`/airlines/airlines/bookings/{id}`	Delete a booking with ID
GET	`/airlines/airlines/bookings`	Retrieve all bookings for current user
POST	`/airlines/airlines/bookings`	Create a booking

reviews all the review methods

i) Take some time to explore the REST APIs by expanding the operations and understanding the content of each operation.

You have just experienced how easy it is to push a Liberty package with RESTful APIs into Bluemix to enable the same environment to view REST APIs as we did in the previous lab.

Congratulations! You have successfully completed this lab. In the next lab, you will learn how to document REST APIs for your app using OpenAPI programming models.