Lab Center – Hands-On Lab

Session 4505

Cloud Pak for Applications

IBM

DISCLAIMER

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."
Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

We Value Your Feedback

Don’t forget to submit your Think 2020 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.

Access the Think 2020 agenda tool to quickly submit surveys from your smartphone or laptop.

Table of Contents

1 Introduction

This Lab explores Cloud Pak for Applications (CP4A).

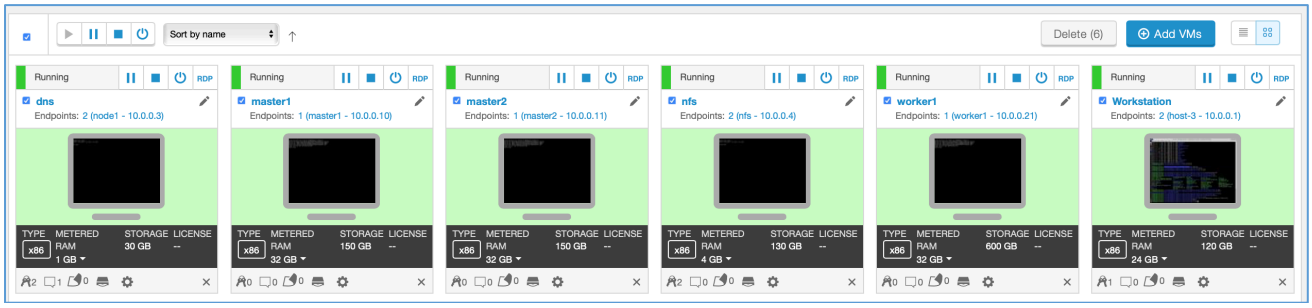
The IBM Cloud Pak™ for Applications provides a complete and consistent experience to speed development of applications built for Kubernetes using agile DevOps processes. You can easily modernize your existing applications with IBM integrated tools and develop new cloud-native applications faster for deployment on any cloud.

Running on Red Hat® OpenShift®, IBM Cloud™ Pak for Applications provides a hybrid, multicloud foundation built on open standards, enabling workloads and data to run anywhere. A self-service environment combines open source tools with your existing middleware for continuous compliance and visibility across secure, hybrid, multicloud environments.

Accelerate innovation while leveraging existing investments by transitioning traditional applications to the cloud in a way that makes sense for your business and is cost-effective.
This lab is based on CP4A Version 4.1.0.

1.1 Lab Environment

This Lab uses Skytap Environment.
As part of this lab, you would get set of Virtual Machines and required software is already installed. You will find 6 Nodes (VMs) in this lab as shown below.



All lab activities should be done on ‘Workstation” VM, which is the last-one in the list above.
Click and login to the VM.

Standard Password for Workstation VM login, as well as performing sudo activities is *passw0rd*
User: ibmdemo
Password: passw0rd

1.2 Command Help inside workstation VM

If you need to copy commands from this Lab document to the target workstation, you can do so by copying content via ‘clipboard’ provided by skytap.

Please note: For the smooth execution of lab, we have provided file named “commands.help”.
You can open this text file via File Manager. (/home/ibmdemo/Lab4505/commands.help) and open in text editor. Keep the file open, so its easy to copy commands in same VM.

2 Objective of the Lab

This lab is designed to introduce the key use-cases of Cloud Pak for Applications.
There are two key use cases:

1. Build Cloud Native Applications faster with the CP4A tooling (Accelerator for Teams – previously known as Kabanero Enterprise) and integrate with Pipelines (CI/CD).
    - Using the insurance quote sample application, which has 2 microservices (one Node.JS Frontend and one Spring Boot Backend), this lab walks through various tools of Accelerator for Teams.

2. Migrate a traditional WebSphere Application Server (WAS) Application to CP4A using Open Liberty Runtime.
    - Using the Customer Order reference application, this Lab documents the journey of bringing existing middleware applications to an OpenShift Container Platform with Analyze, Build & Deploy and Run & Manage phases

This lab also showcases some of the monitoring capabilities in both the scenarios.

At the end of this lab, you would be aware of various capabilities of CP4A and tools provided by this Cloud Pak. You will also understand how CP4A helps organisations to build new cloud native applications faster with agile methodology as well as how to modernize existing applications to deploy on a container platform.

3 Use case : Cloud Native Development

This section covers the Cloud Pak for Applications capabilities for building new Cloud Native Applications.

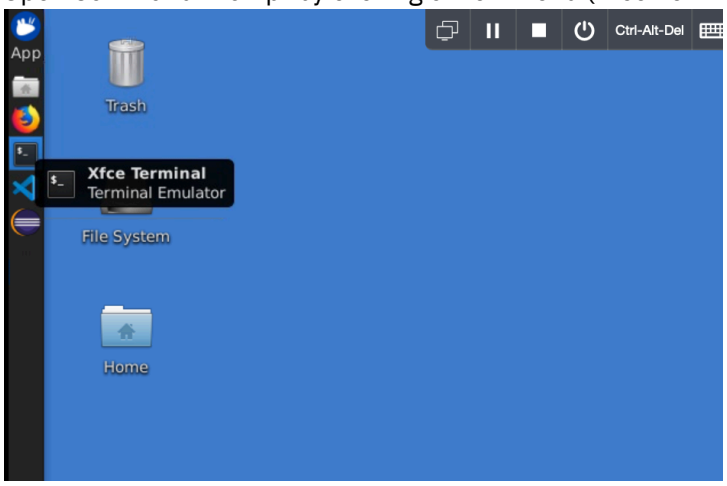
The following command line (CLI) tools are configured:

- oc -> OpenShift Command line utility
- git -> Standard GIT CLI
- appsody -> Accelerator For teams provide this CLI to manage various stacks

Step by Step Instructions:

1. Login to workstation, you would be logged in as
   user: ibmdemo
   password: passw0rd

2. Open Command Prompt by clicking on left menu (Xfce Terminal)



3. Review the command line tool called Appsody
   appsody version (It would show version 0.5.9)

For detailed command list please visit following link
https://appsody.dev/docs/cli-commands/

In this exercise, we will build a new cloud native application which is for getting insurance quote for health insurance provider. This sample-quote application has two microservices

1. Node.JS based front end microservice
2. Spring Boot based backend microservice

In the next section we would see how developer can quickly build this application and run it on container platform, during development phase as well as post development.

3.1 Build/Develop Frontend Node.JS Microservice

3.1.1 Setup Frontend Project using Stack defined as part of 'Accelerator for Teams'

- ***cd /home/ibmdemo/Lab4505***
- ***mkdir front-end***
- ***cd front-end***
- ***appsody init nodejs-express***
- ***ls -lart*** (Look at the files which are generated)

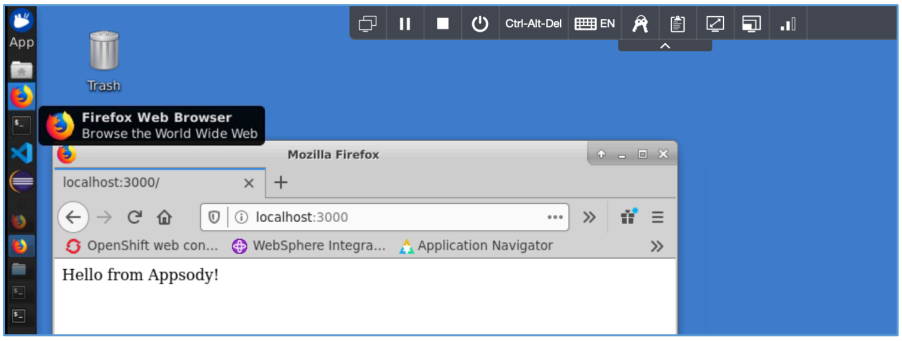
```
ibmdemo@workstation:~/Lab4505/front-end$ ls -lrt
total 80
-rw-rw-r-- 1 ibmdemo ibmdemo  2779 Apr 12 11:26 quote.js
-rw-rw-r-- 1 ibmdemo ibmdemo   290 Apr 12 11:26 app.js
-rw-rw-r-- 1 ibmdemo ibmdemo   616 Jun  9 09:37 package.json
-rw-rw-r-- 1 ibmdemo ibmdemo 51421 Jun  9 11:09 package-lock.json
drwxr-xr-x 2 root    root     4096 Jun  9 11:09 node_modules
drwxrwxr-x 2 ibmdemo ibmdemo  4096 Jun  9 11:11 config
drwxrwxr-x 2 ibmdemo ibmdemo  4096 Jun  9 11:11 views
drwxr-xr-x 2 ibmdemo ibmdemo  4096 Jun  9 11:11 test
ibmdemo@workstation:~/Lab4505/front-end$
```

With these commands, sample Node.JS microservice is created, which has some basic code. This can be used by developer to code Node.JS based front-end application. We are leveraging the 'nodejs-express' stack, which makes developer's life easier and he/she doesn't need to worry about docker, build, deploy etc. Stack would be useful here.

3.1.2 Build Node.JS Microservice Template

Run the sample project by running following commands while you are in in ***/home/ibmdemo/Lab4505*** folder.

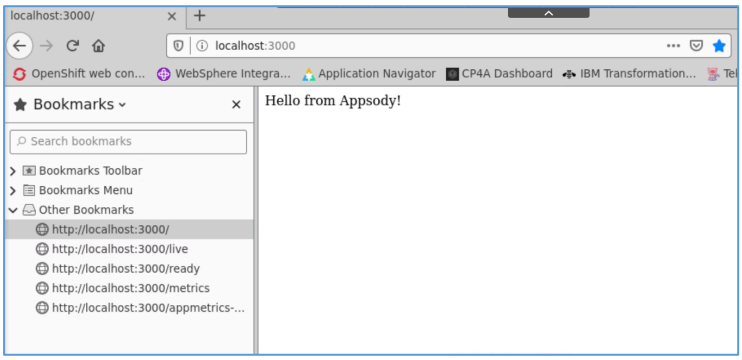
- ***cd /home/ibmdemo/Lab4505/front-end***
- appsody run (Project would run as docker container on developer's machine)
- Open the Firefox Browser and Access the application now at http://localhost:3000



- Access following additional URLs for quick monitoring (You may want to open browser bookmarks by pressing control+b)

  - Health endpoint: http://localhost:3000/health
  - Liveness endpoint: http://localhost:3000/live
  - Readiness endpoint: http://localhost:3000/ready
  - Metrics endpoint: http://localhost:3000/metrics
  - Dashboard endpoint: http://localhost:3000/appmetrics-dash (development only)

You may want to open browser bookmark (Press ctr + b)



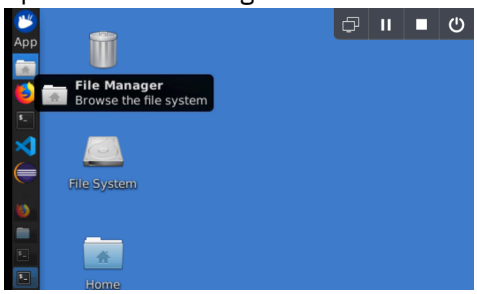
- Stop the application using control + C

Here we have seen how the sample application is running on local docker environment, and how the capabilities provided by stack around monitoring and metrics dashboard, helps developers to get early view of application performance.

So far, we were looking into sample code, lets add application specific code into this microservice.

3.1.3 Add required code for Insurance Quote for Frontend

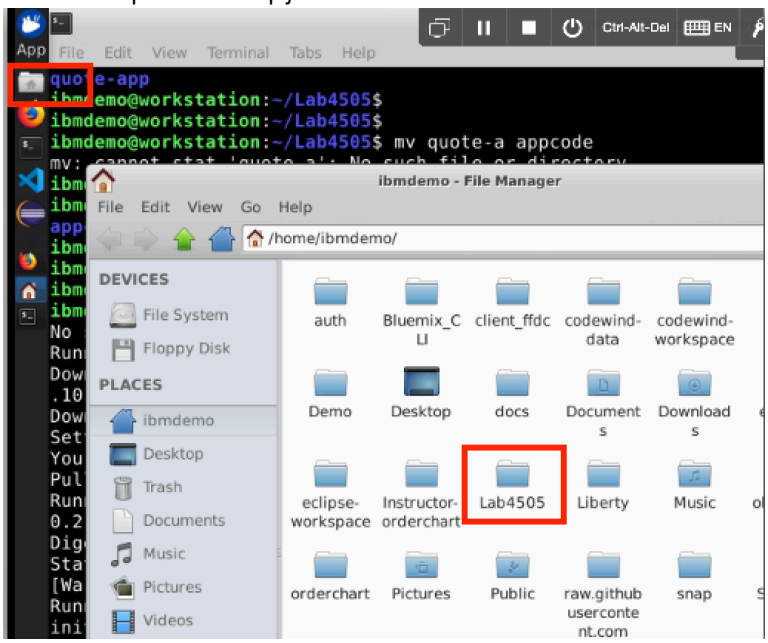
- Copy the application code into ***/home/ibmdemo/Lab4505/appcode/front-end***
- Open the 'File Manager' Window



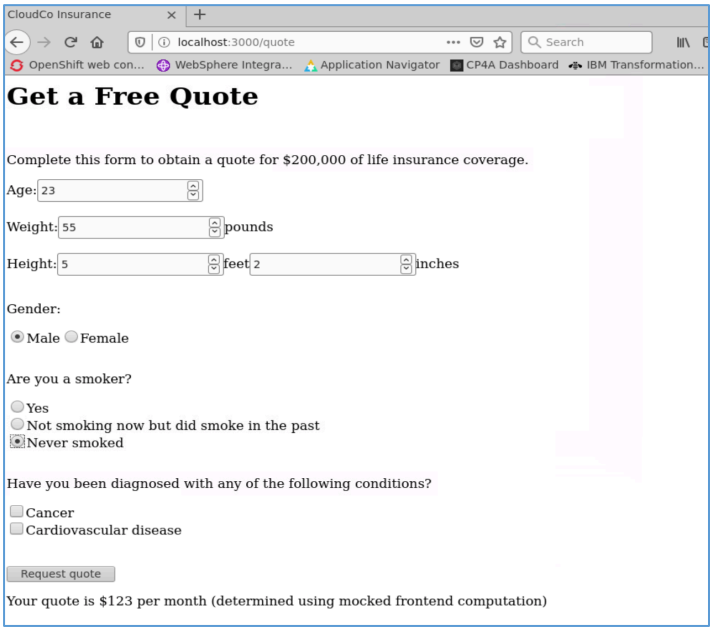
- Change directory to ***/home/ibmdemo/Lab4505/appcode/front-end*** and copy all 6 files to ***/home/ibmdemo/Lab4505/front-end***

  app.js
  config/
  quote.js
  test/
  views/
  package.json

Use File explorer to copy files



- There will get multiple prompts to overwrite the files. Use “Replace All’ to overwrite.
- Rebuild and test application using the following command
  appsody run
- Open the Firefox Browser and Access the application now at http://localhost:3000/quote



- stop the application
- Run tests with command: ***appsody test*** (Optional)

Please note: Currently the backend URL is “test-url” one. Go to config folder and you will find the file called development.json, which has local URL for backend.

So far, we have seen how Node.JS application can be built faster and run the development environment on docker on local.

Let’s create Java based backend service now.

3.2 Build / Develop Backend Spring Boot Microservice

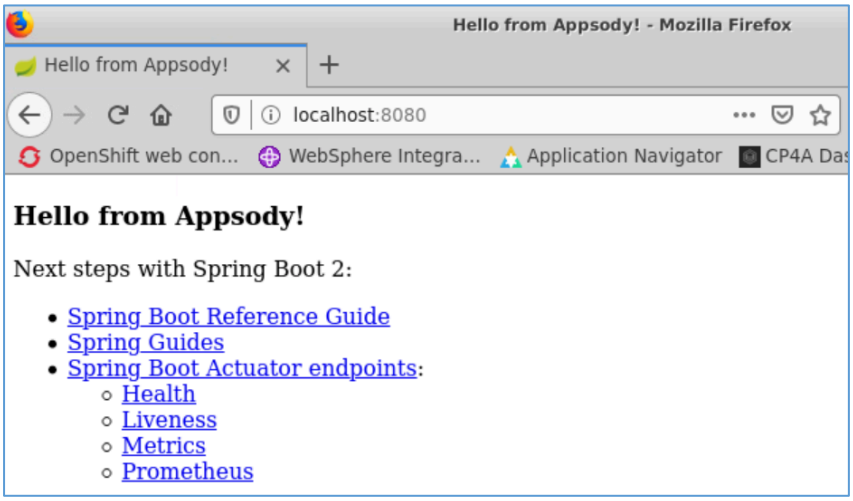
3.2.1 Setup Backend Project using Stack defined as part of 'Accelerator for Teams'

- ***cd /home/ibmdemo/Lab4505***
- ***mkdir back-end***
- ***cd back-end***
- ***appsody init java-spring-boot2***
- ***ls -l*** (Look at the files which are generated)
- You will find Spring Boot application located in 'src/main/java/application' folder

3.2.2 Build Microservice Template using stack

Run the sample project

- ***appsody run*** (Project would run as docker container on developer's machine)
- Access http://localhost:8080



- Access following additional URLs for quick monitoring
  - Health endpoint: http://localhost:8080/actuator/health
  - Liveness endpoint: http://localhost:8080/actuator/liveness
  - Metrics endpoint: http://localhost:8080/actuator/metrics
  - Prometheus endpoint: http://localhost:8080/actuator/prometheus
- Stop the application using control + C

3.2.3 Add required code for Insurance Quote for Backend

- Look at existing source code and copy
  From: ***/home/ibmdemo/Lab4505/appcode/back-end/***
  To: ***/home/ibmdemo/Lab4505/back-end/***

  Copy following files
  - src
  - target
  - mvnw
  - mvnw.cmd
  - pom.xml
- Run: ***appsody run***

Please Note: You can test the backend API using curl. The file ***backend-input.json*** contains sample input for the API.

- Open new command prompt
- ***cd /home/ibmdemo/Lab4505*** (Make sure there exists backend-input.json)
- ***curl -X POST -d @backend-input.json -H "Content-Type: application/json" http://localhost:8080/quote***

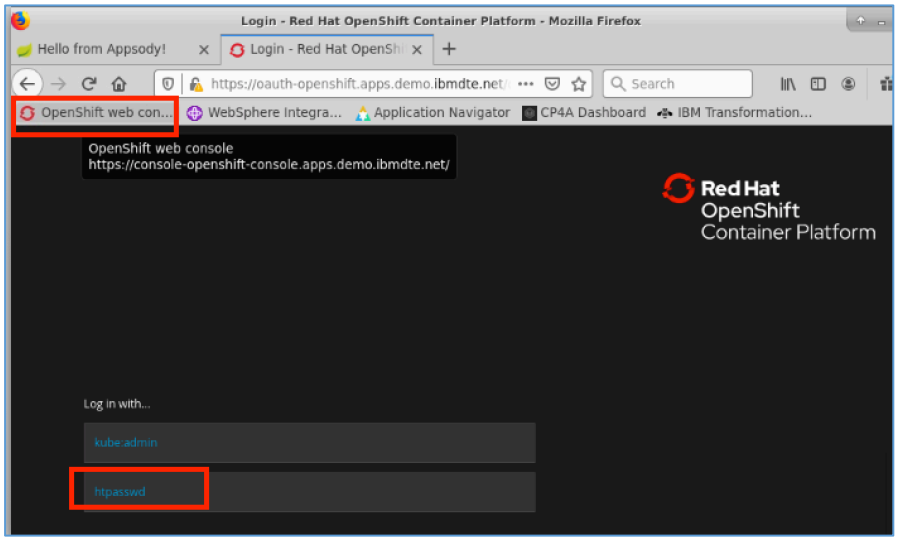
  Expect response: {"quotedAmount":30,"basis":"mocked backend computation"}

- Stop the application using control + C

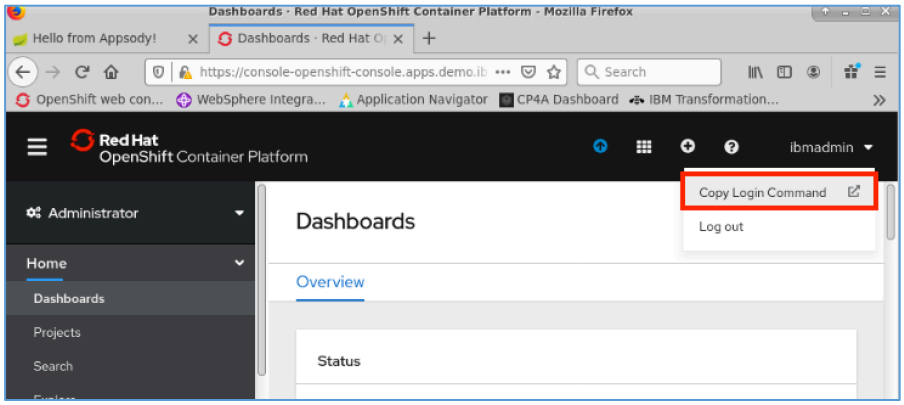
4 Deploy the newly developed microservices on Cloud Pak for Application/OCP4.3 Instance

4.1 Deploy Backend Microservice built on Spring Boot

- ***cd /home/ibmdemo/Lab4505/back-end***
- Login to OCP API Server, using following command
- oc login --token=<token> --server=https://<server>:6443
    - Follow the steps below to get the login command
    - Login to console using Firefox bookmark, click on 'htpasswd' to login



    - Accept the username and password and click Login



- Then Click on 'Display Token' and copy the login URL as shown below

- Paste the oc login command on command prompt

Next few commands will deploy the new Spring Boot application, which is tested locally, on to OCP 4.3 environment

- Switch to project 'kabanero'
- cd /home/ibmdemo/Lab4505/back-end
- ***oc project kabanero***
- ***docker login default-route-openshift-image-registry.apps.demo.ibmdte.net -u $(oc whoami) -p $(oc whoami -t)***
- ***appsody deploy --tag kabanero/quote-backend:v1 --push-url default-route-openshift-image-registry.apps.demo.ibmdte.net --push --pull-url image-registry.openshift-image-registry.svc:5000 --namespace kabanero***

- Next few commands will show you how to access the application.
- ***oc get route -n kabanero***
- Note down the backend URL. It should be http://back-end-kabanero.apps.demo.ibmdte.net/

Verify the service by calling following command

- ***cd /home/ibmdemo/Lab4505/***
- ***curl -X POST -d @backend-input.json -H "Content-Type: application/json" back-end-kabanero.apps.demo.ibmdte.net/quote***
-
```
ibmdemo@workstation:~/Lab4505/backend$ curl -X POST  -d @backend-input.json  -H "Content-Type: application/json"  backend-kabanero.apps
.demo.ibmdte.net/quote
{"quotedAmount":30,"basis":"mocked backend computation"}ibmdemo@workstation:~/Lab4505/backend$
ibmdemo@workstation:~/Lab4505/backend$
```
- Also look at '***oc get svc***' and make sure backend is running at **back-end:8080**

4.2 Deploy Frontend Microservice built on Node.JS

- ***cd /home/ibmdemo/Lab4505/front-end***
- ***appsody deploy --generate-only***
- New file is generated app-deploy.yaml

- Customize the app-deploy.yaml file and add backend_URL ENV Variable, as shown below
- Open the file via text editor and add following entry

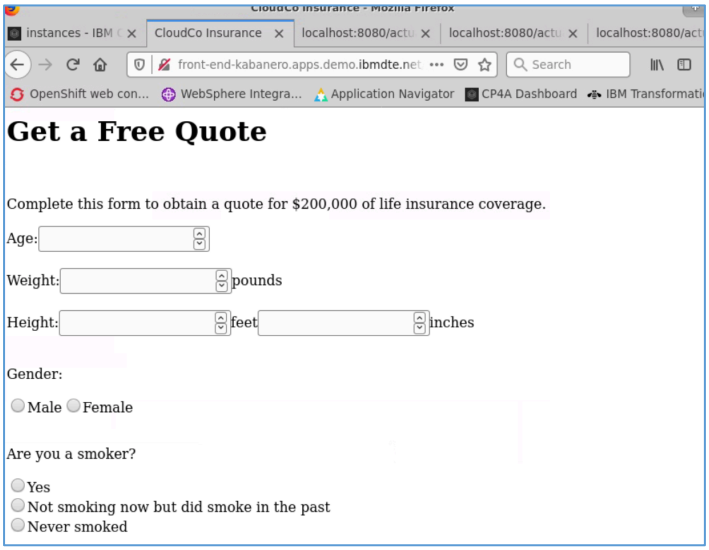
```
env:
  - name: BACKEND_URL
    value: http://back-end:8080/quote
```

-

```
stack: nodejs-express
version: 1.0.0
env:
  - name: BACKEND_URL
    value: http://back-end:8080/quote
```

- ***appsody deploy --tag kabanero/quote-frontend:v1 --push-url default-route-openshift-image-registry.apps.demo.ibmdte.net --push --pull-url image-registry.openshift-image-registry.svc:5000 --namespace kabanero --force***

- ***oc get route -n kabanero***
- Select the route for name ‘frontend’ and copy the same to browser
- Verify the service is working fine



5 Cloud Native Application – DevOps Integration

Now that both of the applications are running on CP4A/OCP4.3 environment, lets add pipeline so that build and deploy can happen automatically, when developer updates the code or triggered manually. In this lab, we will configure the manual trigger of Pipelines, Webhook can be configured, but for this lab, we will skip Webhook and follow user-initiated build and deploy process. These Pipelines are based on opensource project, Tekton.

Please Note: We would delete previous instance of backend microservice, so that we can focus on pipeline and let pipeline deploy the new instance for us. Please execute following two commands.

Clean up existing backend
cd /home/ibmdemo/Lab4505/back-end
appsody deploy delete

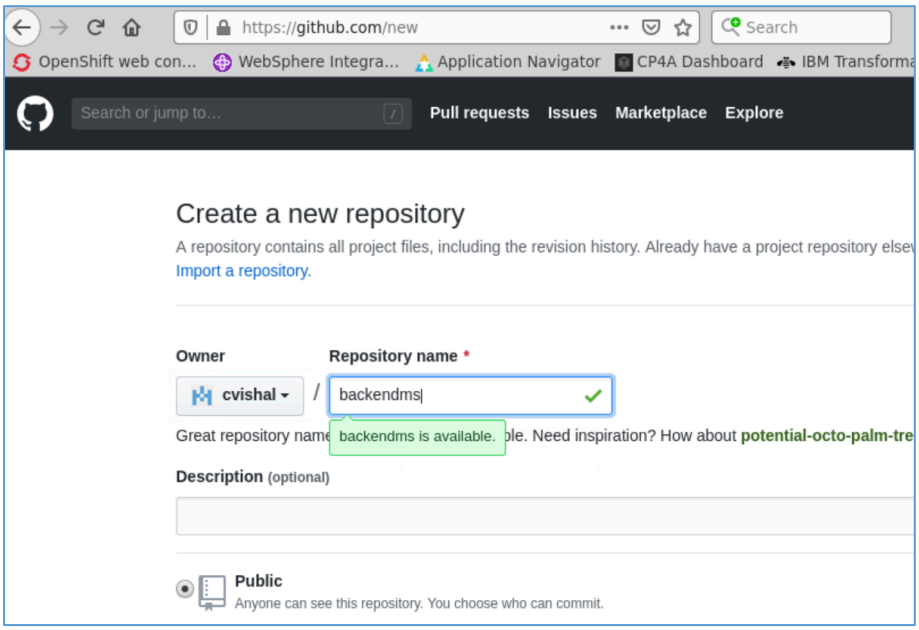
5.1 Configure Pipelines / CI-CD for Backend Microservice

Check in Backend microservices code and build CI-CD using CP4A, Pipeline Dashboard. (If there is time constraint, skip 5.1.1 and 5.1.2. Jump to 5.1.3.)

5.1.1 Check-in code in github account

Create github repo and push the project to git

- Assume that you have access to github.com. If not, please create new ID and login to github.com.
- If needed github.com might send verification code to your email, please complete sign-in process.
- Create a new public repository called 'backendms'



- Note down the git URL from next screen (Example: https://github.com/cvishal/backendms.git)
- ***cd /home/ibmdemo/Lab4505/back-end***
- Push all the application code to this repo

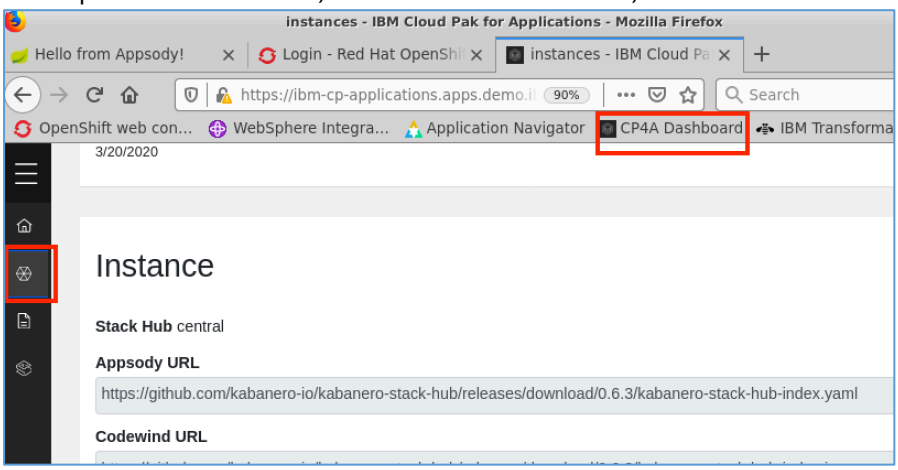
- Run following git commands

```
git init
git add .
git commit -m "first commit"
git remote add origin https://github.com/<your-git-account>/backendms.git
git push -u origin master
```

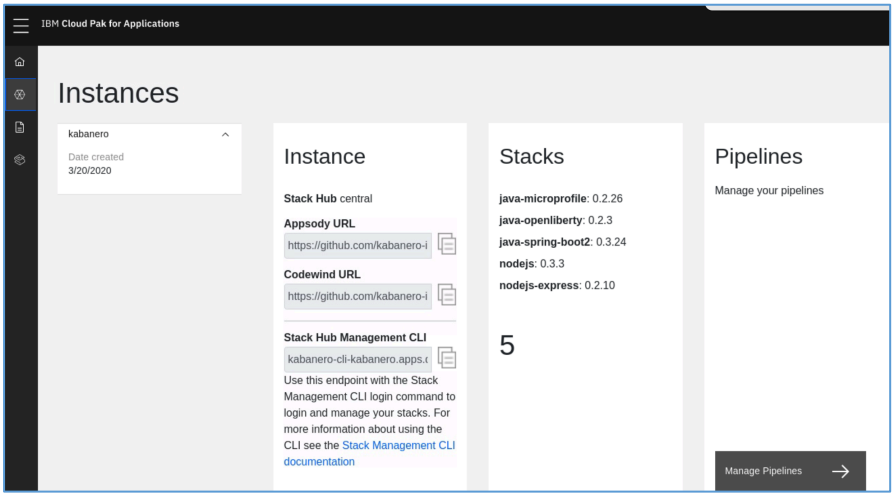
(Enter your git credentials)

5.1.2 Login to Pipeline Dashboard and create pipeline resources

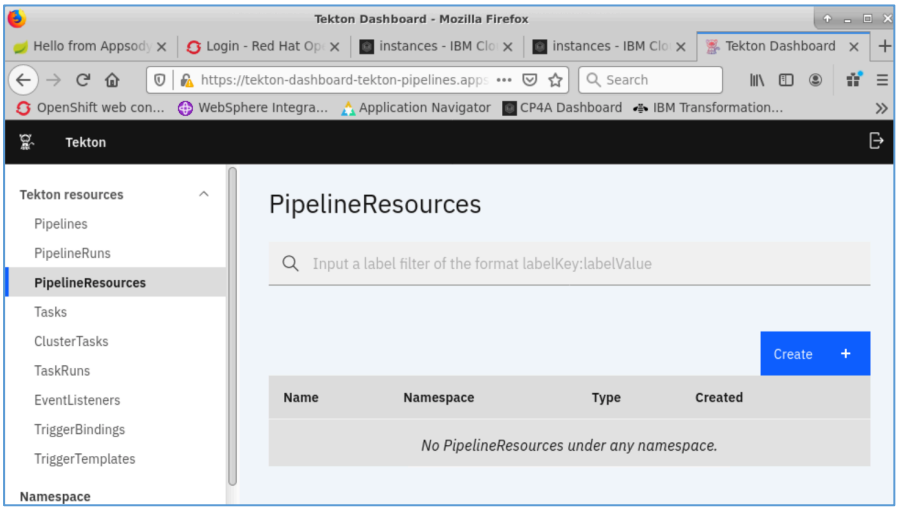
- Login to CP4A Console and open Pipeline Dashboard
- First Open Firefox browser, click on CP4A Dashboard, Select Instances from Left Navigation.



- Look at Pipelines



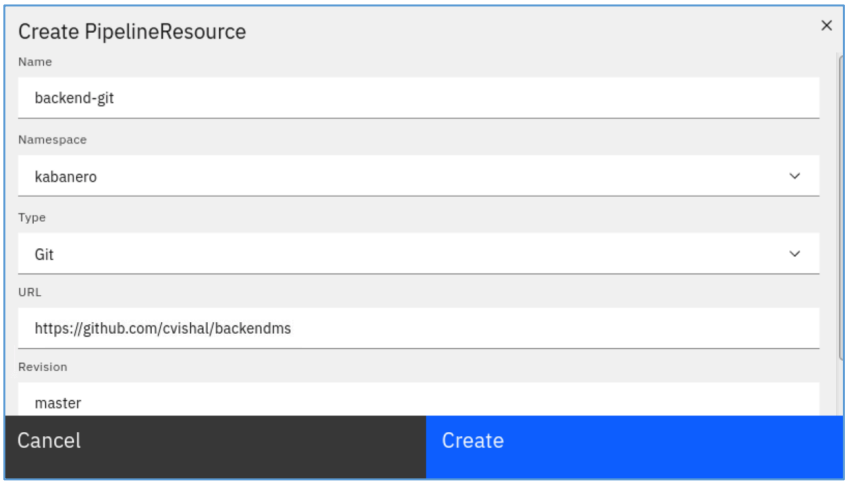
- Click on Manage Pipelines
- Login with htpasswd like and accept the userid/passwprd (ibmadmin/passw0rd)
- Then Click on “PipelineResouces” and create new resource



- Click on “Create” button from right side of the page.

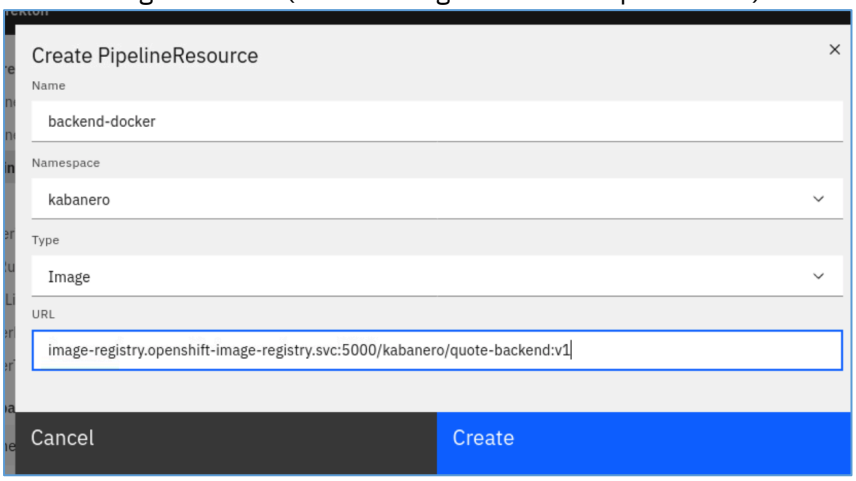
Basically, we need to create two pipeline resources. There are two resources already created but we recommend you create your own resources.

1. Git Resource (Mention the GIT details by creating git resource)



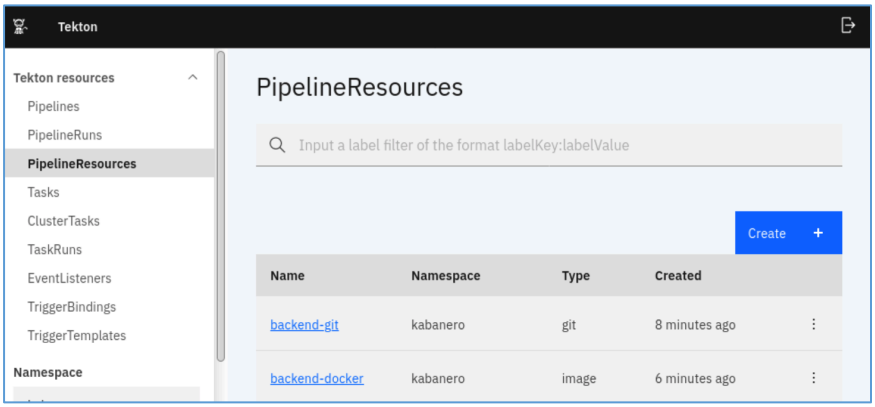
You can name as backend-git-<yourname>.
You can give your GIT Location URL which you have created in the previous step.
Use Revision as ‘master’

2. Docker Image resource (Provide image name and repo location)



Name: backend-docker-<yourname>
URL Should be “image-registry.openshift-image-registry.svc:5000/kabanero/quote-backend:v1”

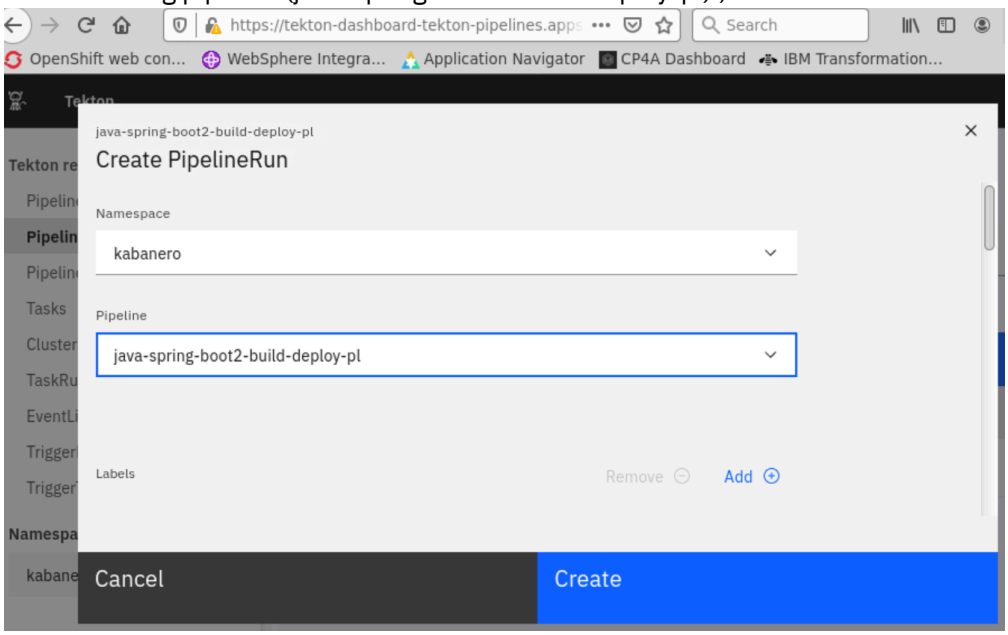
3. Make sure you see both resources.



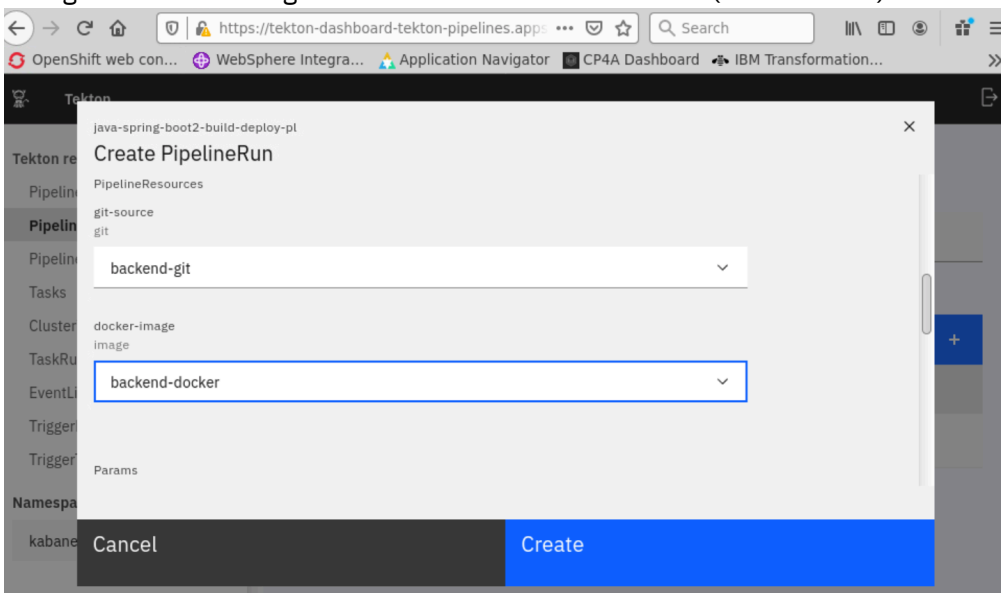
5.1.3 Configure Pipeline Runs using existing pipeline

- Create a new pipeline run.
- On Pipeline Dashboard, click on Pipeline Runs and create a new Pipeline Run

- Select existing pipeline (java-spring-boot2-build-deploy-pl) , scroll further.

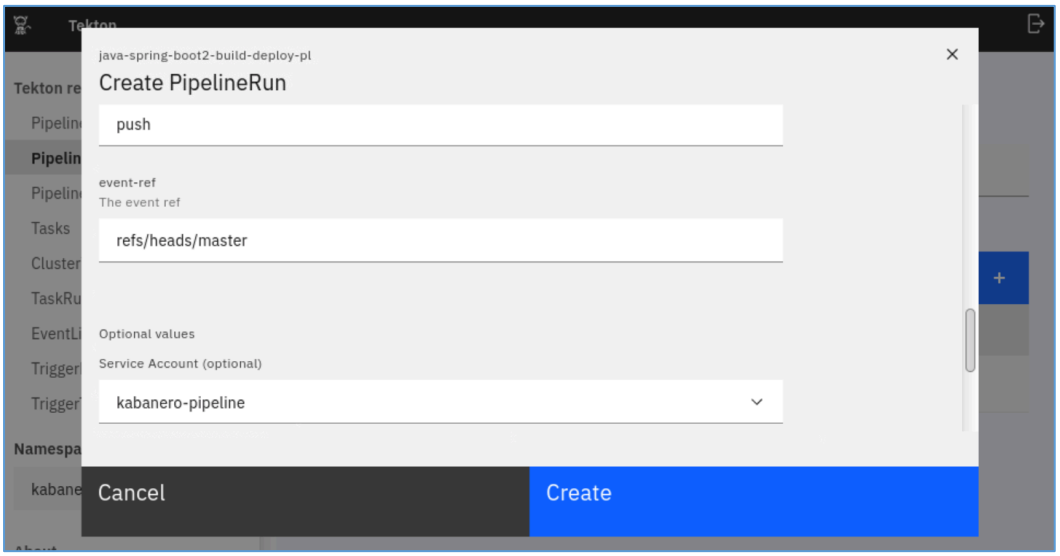


- Add git and docker image information as mentioned below (Scroll further)

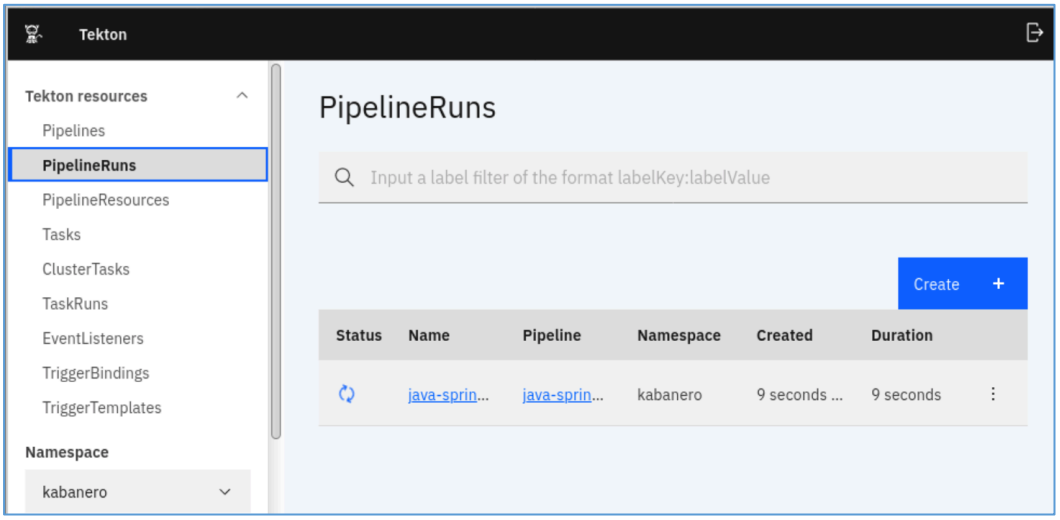


- Add following for “The Event Type”
  - Push
- Adding following for “The Event Ref”
  - refs/heads/master
- Select ServiceAccount as
  - Kabanero-pipeline

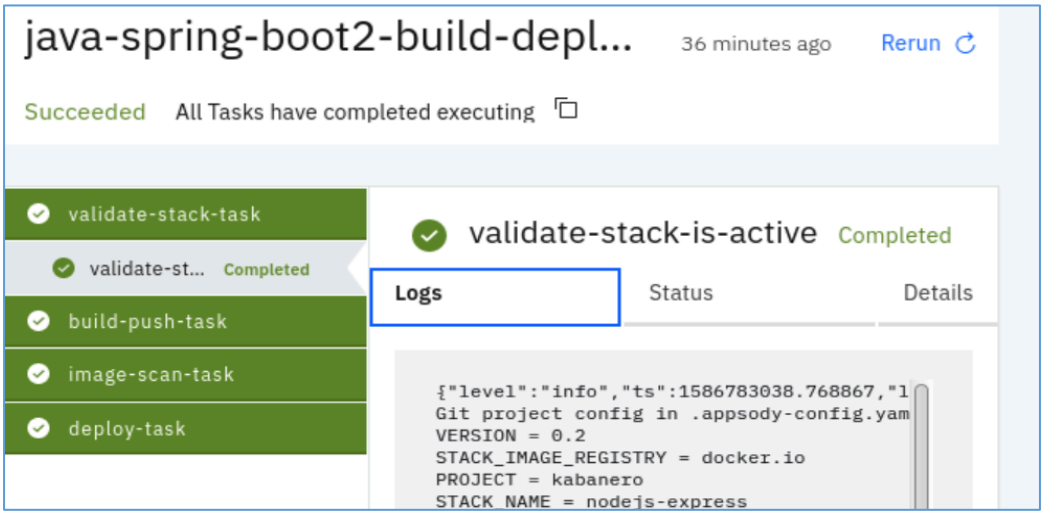
- As mentioned below:



- Click Create.
- You will find pipeline run has started.



- Once complete, you would see following status.

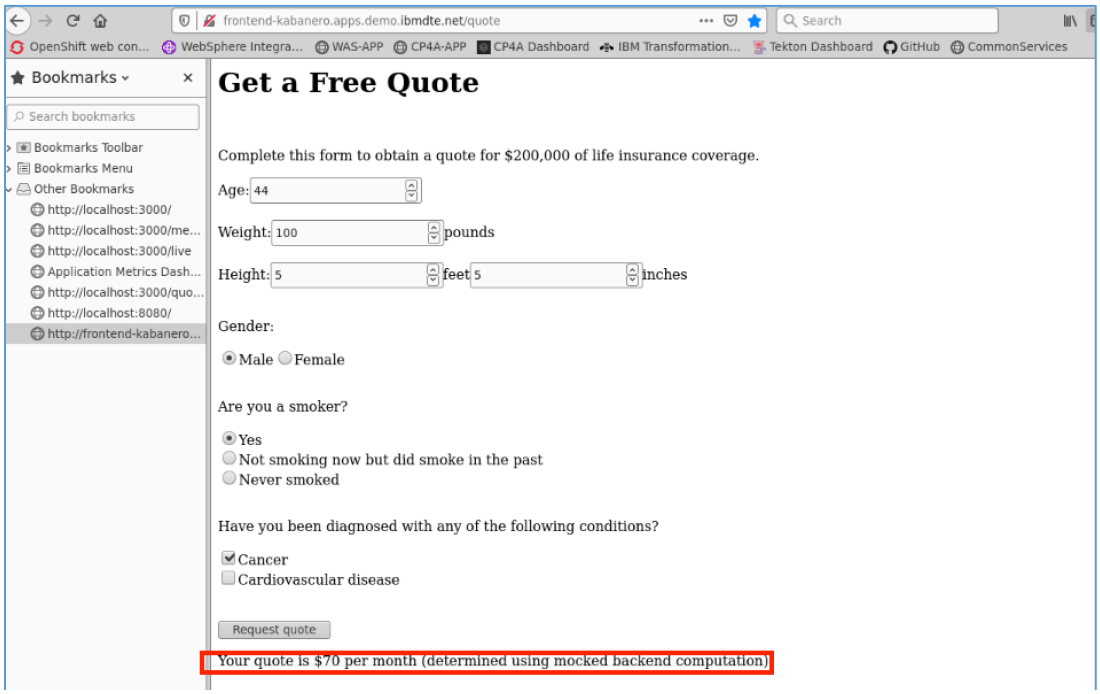


Validate the application is running fine:
You should see back-end-* pod.

```
ibmdemo@workstation:~/Lab4505/backend$ oc get pods
NAME                                                  READY   STATUS      RESTARTS   AGE
backend-5f5ffc4cb5-ghpkd                              1/1     Running     0          13m
codeready-operator-6c959f64d7-8d8sz                   1/1     Running     0          44h
frontend-5d848c555c-65k8v                             1/1     Running     0          6h41m
icpa-landing-7bdf866765-m6j6b                         1/1     Running     15         33d
java-spring-boot2-build-deploy-pl-run-1587669721261-build-d8rtm   0/9     Completed   0          52m
java-spring-boot2-build-deploy-pl-run-1587669721261-deplo-kk26z   0/3     Completed   0          41m
```

This task takes about 10 minutes. Once complete, you will find the new application is updated with new code changes. Here we have started pipeline manually, but once we configure the webhook with git, the pipeline will get triggered when there is new push into git.

Access the application and make sure it works.



6 Use case : Existing Application Modernization.

6.1 Introduction

Runtime modernization moves a traditional application to a "built for the cloud" runtime with the least amount of effort. IBM® WebSphere® Liberty is a fast and dynamic Java™ application server that is built on the open source Open Liberty project. Ideal for the cloud, Liberty is a combination of IBM technology and open source software. It has fast startup times (less than 2 seconds), no server restarts to pick up changes, and a simple XML configuration.

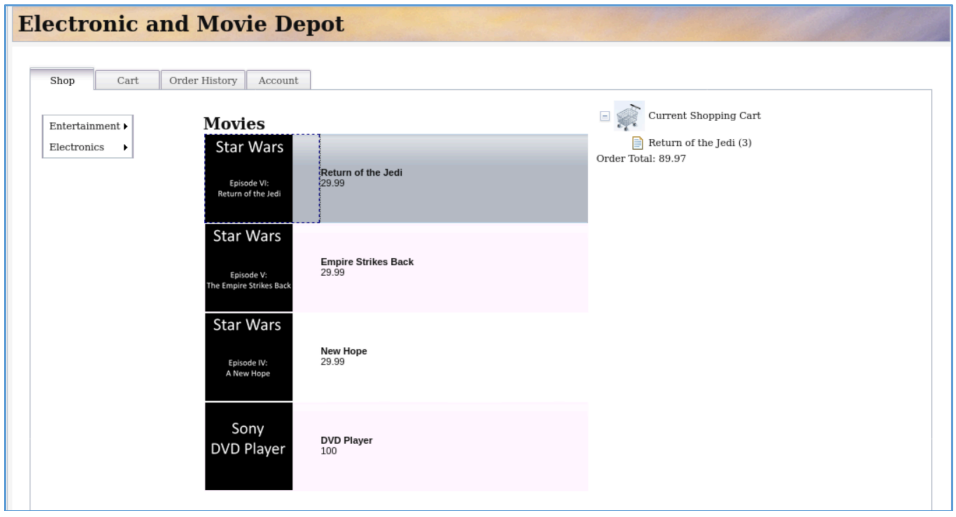
However, WebSphere Liberty doesn't support all the traditional Java EE and WebSphere proprietary functions. You might need to change some code to move an application to the new runtime. You also need to move the application configuration from traditional WebSphere to WebSphere Liberty's XML configuration files.

This path gets the application onto a cloud-ready runtime container that is easy to use and portable. However, the application is mostly unchanged and is not modernized to a newer architecture such as microservices. This is first step towards modernization, and you will see immediate benefits.

In this exercise, we will look at existing application which is running on WebSphere Application Server ND (Version 8.5.5.14) and how IBM Modernization Tool like Transformation Advisor helps.

The objective of this lab is to showcase how Transformation Advisor helps move the existing application to container platform, with mandatory changes which are pointed by this tool.

Application is typical retail store application, called '**Customer Order Services Application'**. Where user can login, browse various items related to Electronics products and Movies then add into cart.



The lab has following flow:

1. Look at existing application which is deployed on WAS 8.5.5.14, access the application
2. Run the Transformation Advisor, which is part of Cloud Pak for Applications.
3. Download the new artifacts and make necessary changes
4. Run the Liberty Operator to deploy same application to Cloud Pak for Application on Liberty
5. Access the new application

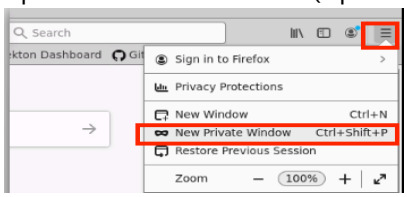
6.2 Existing Application on WAS ND

To start this exercise, you need to bring up the existing WebSphere Application Server instance.

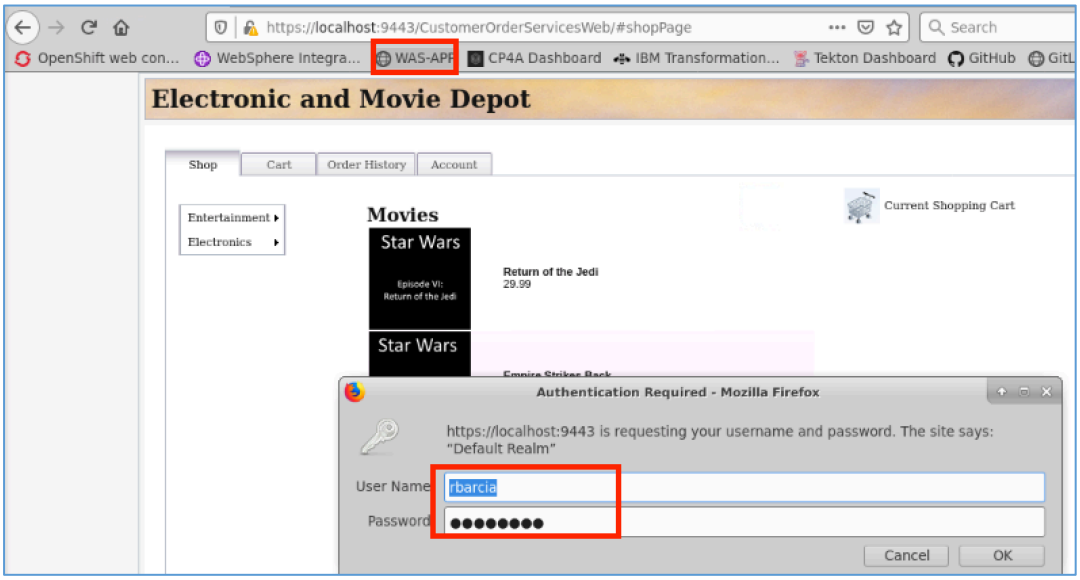
- Open the terminal window
- ***cd /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin***
- ***sudo ./startServer.sh server1***
- (Password for sudo: passw0rd)
- Wait for server to come up

Access the application

- Open the Firefox browser (Open Private Window)

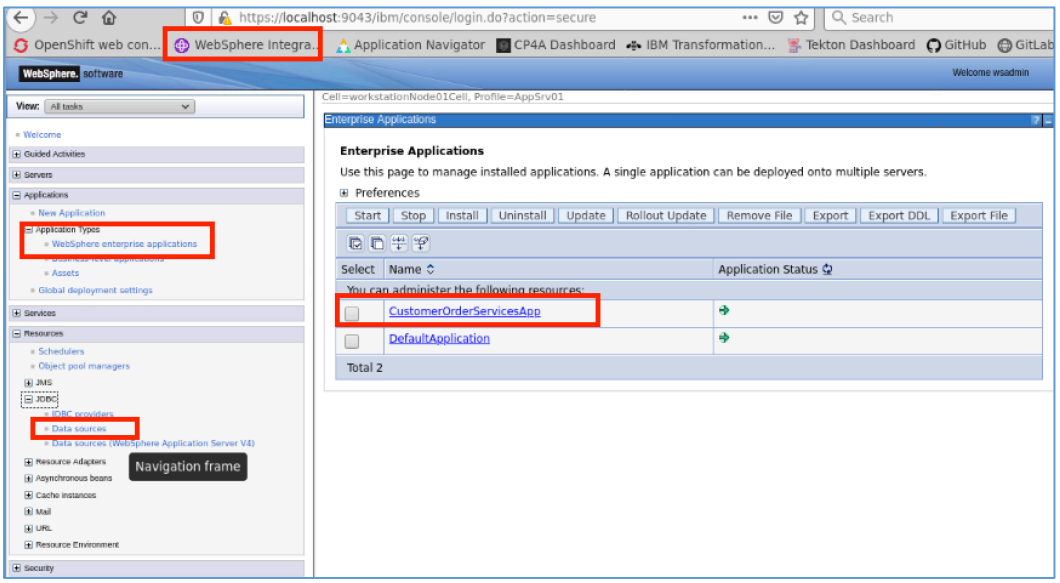


- https://localhost:9443/CustomerOrderServices/Web (Or Click on WAS-APP bookmark)
- Login with 'rbarcia' and password as 'bl0wfish'



Access the Admin Console (Use New Private Browsing Window)

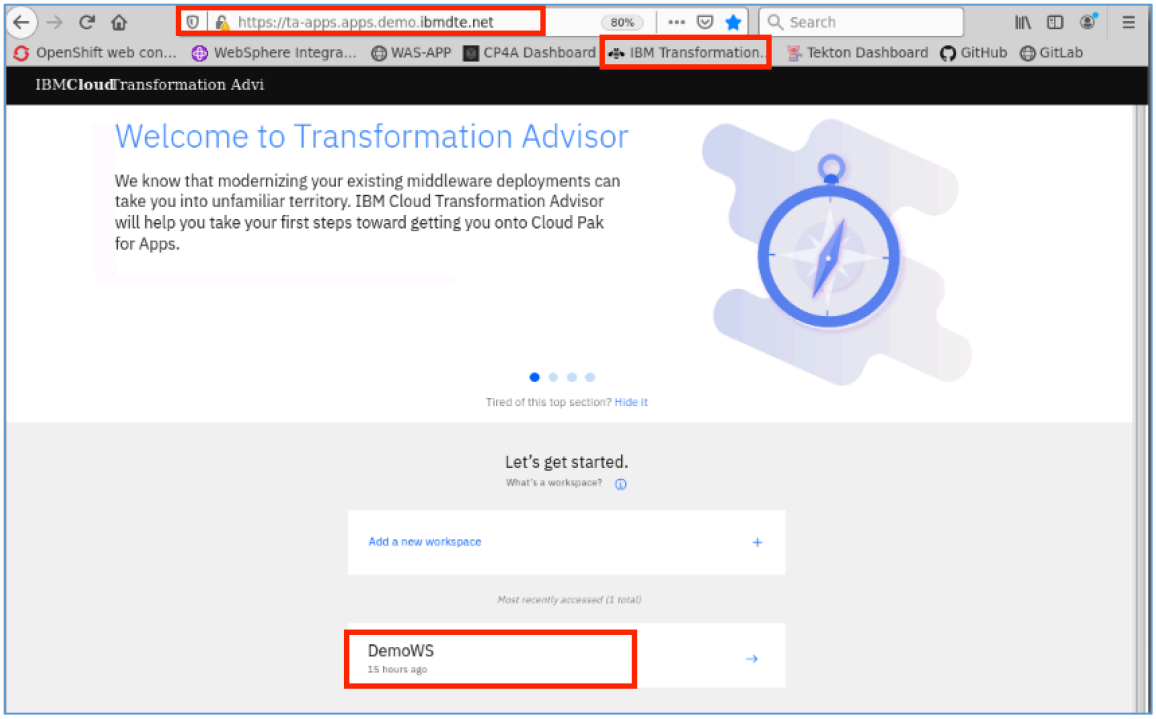
- http://localhost:9060/ibm/console (Login with wsadmin/passw0rd)
- Browse various resources like application and data source



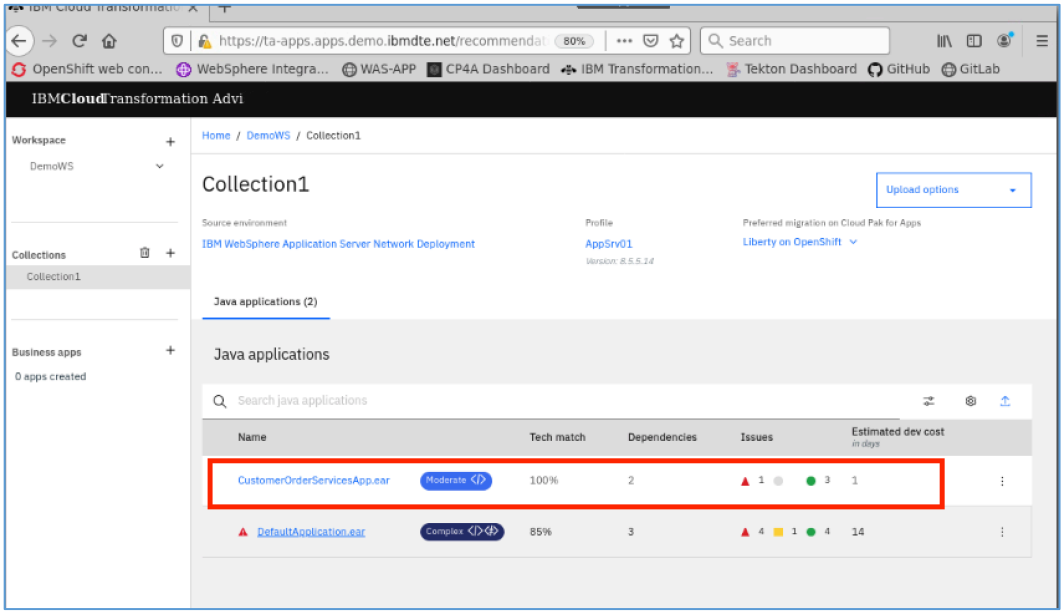
6.3 Transformation Advisor

Transformation Advisor is already available with Cloud Pak for Applications. It's possible to download the data collector and run it on the WAS ND instance. In the interest of time, let's look at the data that is already populated here in this tool. *(Data Collector tool has already run. But if you want to re-run, you can download and run it again)*

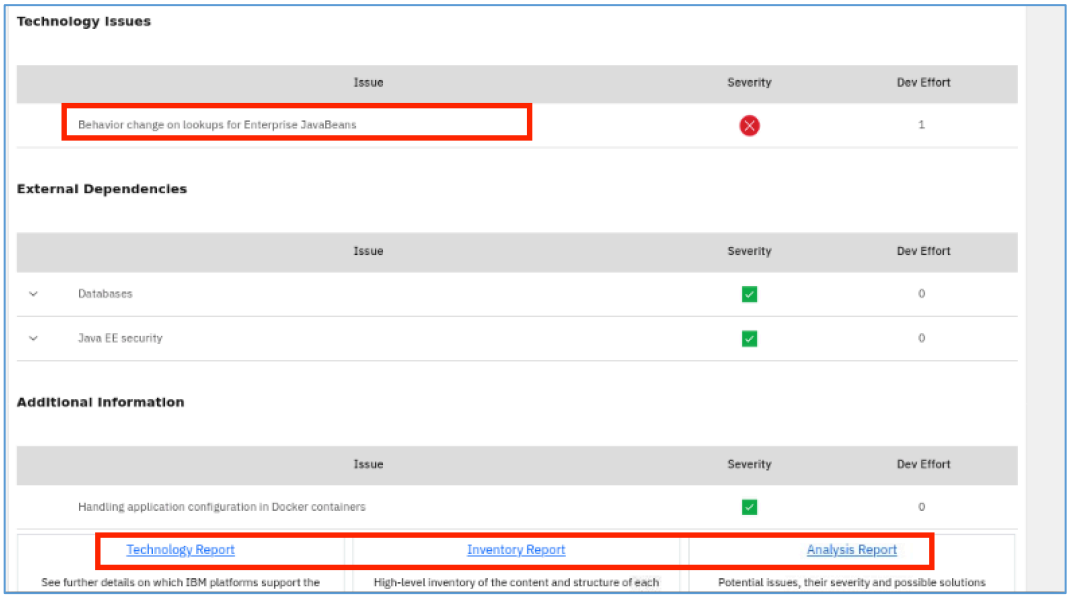
- Login to Transformation Advisor, using htpasswd and then click on pre-populated DemoWS.



- Click on ‘DemoWS’ and then ‘Collection1’



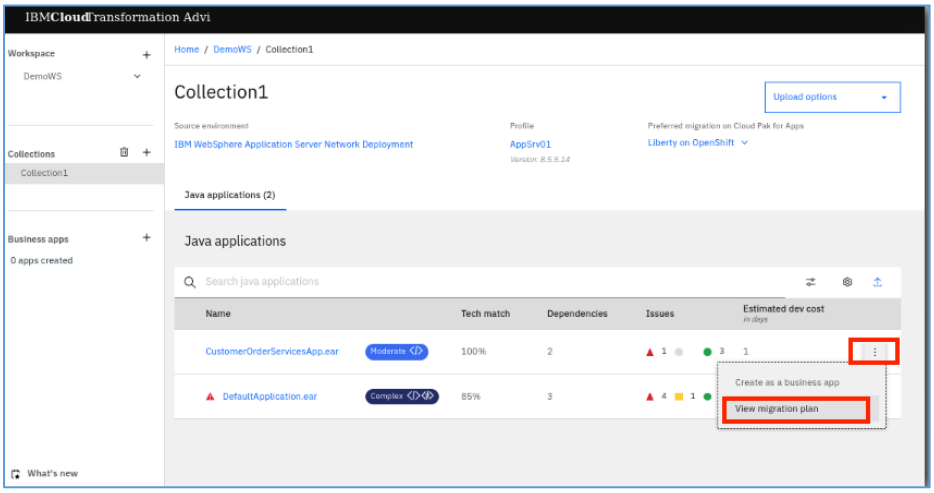
- Click on name ‘CustomerOrderServcies.ear’ and look at the various reports, which shows the Issues for this migration.



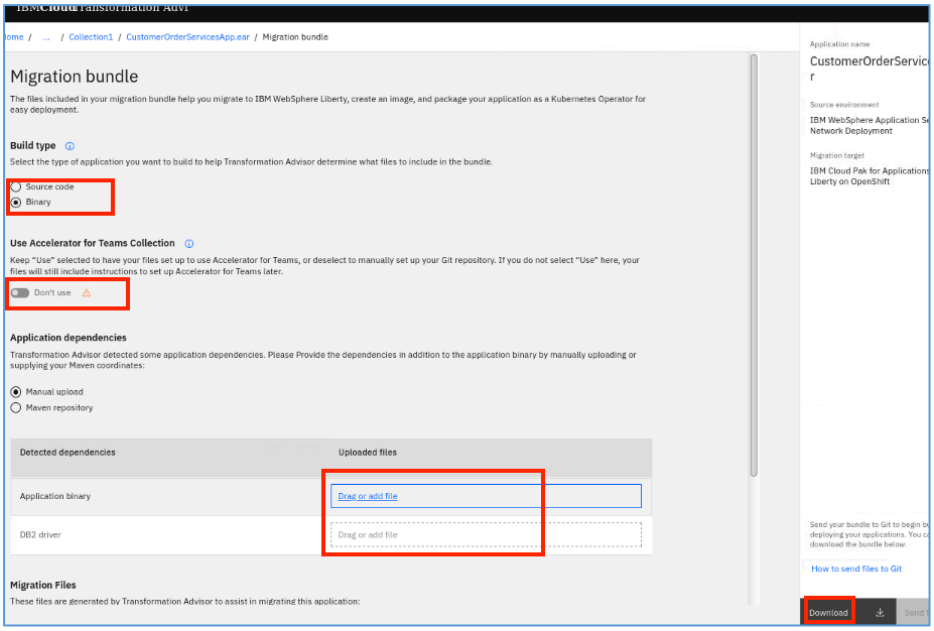
- Look at the detailed 3 reports
    1. Analysis Report: Detailed Analysis Report
    2. Inventory Report: All inventory of application
    3. Technology Report: JEE technologies used in application

Please note: When you look at the detailed Analysis Report, you will find that there is one mandatory change needed for application. Ideally developer need to work on that code change, and build new .ear file. Also there need some changes in server.xml such as datasource password etc. We have made application change already and new .ear file is available in the folder called /home/ibmdemo/Lab4505/migration-help/. Please use these files when working with containers.

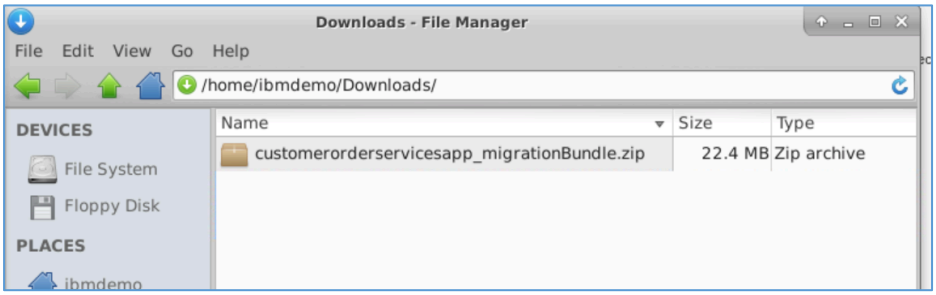
- Look at the artifacts generated by ‘Transformation Advisor’, by going to ‘View Migration Plan’



- Make sure ‘Use Accelerator for Team Collection is disabled’, ‘binary’ as build type, upload application file as well as db2jcc4.jar from /home/Lab4505/migration-help/ folder.



- You can either download this file or you can use one, which is already downloaded and unzipped.
- You will find following file in Downloads folder (which is already present and unzipped)



- File size is about 22 MB

6.4 Prepare Application deployment on CP4A/OCP4.3

- Open the terminal
- ***cd /home/ibmdemo/Downloads***
- You will find following files (its already unzipped to save time)

```
ibmdemo@workstation:~/Downloads$ ls -lrt
total 17912
-rw-rw-r-- 1 ibmdemo ibmdemo 18315685 Apr 18 21:10 customerorderservicesapp_migrationBundle.zip
drwxrwxr-x 2 ibmdemo ibmdemo     4096 Apr 18 21:10 target
drwxrwxr-x 3 ibmdemo ibmdemo     4096 Apr 18 21:10 src
drwxrwxr-x 4 ibmdemo ibmdemo     4096 Apr 18 21:10 operator
-rw-rw-r-- 1 ibmdemo ibmdemo     3794 Apr 19  2020 READ_THIS_FIRST.md
-rw-rw-r-- 1 ibmdemo ibmdemo      733 Apr 19  2020 pom.xml
-rw-rw-r-- 1 ibmdemo ibmdemo     1512 Apr 19  2020 Dockerfile
```

- Dockerfile for building new image
- operator directory with all required yamls to deploy openliberty operator and application
- src (Liberty configuration and related files)
- target (updated application)
- run following command
  cp /home/ibmdemo/Lab4505/migration-help/server.xml /home/ibmdemo/Downloads/src/main/liberty/config/server.xml

6.5 Deploy the operator and application

Following set of commands will deploy the application on Cloud Pak for Application / OCP4.3. First, we deploy operator CRDs, followed by necessary Service Account related commands. Then we deploy operator and then the application.

If you haven't logged to OCP instance yet, login to OCP instance as mentioned in "Section 4.1"

In this case, name of new project is ***customerorder1***
If you plan to change this to something else, please change further occurrences of ***customerorder1.***

Create a new project

oc new-project customerorder1

Build docker image and push into local docker repo

docker login default-route-openshift-image-registry.apps.demo.ibmdte.net -u $(oc whoami) -p $(oc whoami -t)

docker build -t customerorder1/customerorderservicesapp:latest .

docker tag customerorder1/customerorderservicesapp:latest default-route-openshift-image-registry.apps.demo.ibmdte.net/ customerorder1/customerorderservicesapp:latest

docker push default-route-openshift-image-registry.apps.demo.ibmdte.net/ customerorder1/customerorderservicesapp:latest

Make necessary changes in Application Deployment YAML, with the correct Image Information.

cd /home/ibmdemo/Downloads/operator/application

vi application-cr.yaml

Change image repository:
From: image-registry.openshift-image-registry.svc:5000/**customerorderservicesapp**/customerorderservicesapp
To: image-registry.openshift-image-registry.svc:5000/***customerorder1***/customerorderservicesapp

Install the open-liberty operator (Following files are pre-generated for you by TA)

cd /home/ibmdemo/Downloads/operator
oc create -f application/application-crd.yaml
oc apply -f deploy/service_account.yaml
oc apply -f deploy/role.yaml
oc apply -f deploy/role_binding.yaml
oc apply -f deploy/operator.yaml

Install Application

oc apply -f application/application-cr.yaml
oc get svc
oc get pods

You should see the application pod in Running state.

Create Route for the application
oc create -f /home/ibmdemo/Lab4505/migration-help/route.yaml

Access the application (Open the private browsing window)
https://cosapp.apps.demo.ibmdte.net/CustomerOrderServicesWeb/
Login with rbarcia/bl0wfish

7 Summary

In this lab, we covered two use cases

1. Build Cloud Native Application and how ‘Accelerator for Teams’ is useful for quickly building these applications. Integrate with CI-CD pipeline. Look at monitoring data.
2. Application modernization by moving monolithic application to containers as first step towards modernization. Look at monitoring data.