Cloud Enabled Use Case: App Modernization Journey Part 1 - Evaluation

This is Part 1 of the App Modernization Journey tutorial series. It will walk you through the process to evaluate and modernize a legacy application using IBM Cloud Transformation Advisor. This tutorial duration is about 30 minutes.

Before you start

This tutorial is meant for the user with a VPI designed to showcase Cloud Pak for Multicloud Management. If you have a running instance, please sign up for one.

Please log in to be able to reserve an instance.

1 Tutorial

On the journey to cloud, enterprise customers are facing challenges moving their **IBM WebSphere Applications** to cloud quickly and cost-effectively. The **IBM Cloud Pak for Applications** provides a cloud native application development experience and solution to modernize enterprise applications for cloud native deployments. Customers can migrate, modernize their existing applications with DB2's integrated data management capabilities and WebSphere middleware runtimes on Red Hat cloud. One of the tools included in the Cloud Pak is the **IBM Cloud Transformation Advisor**. **Transformation Advisor**, a developer tool that is installed on a local machine, helps developers and IT Architects analyze applications for deployment to the Cloud. **Transformation Advisor** helps you:

- identify the Java EE programming models in the app;
- determine the complexity of apps by listing a high-level inventory of the content and structure of each app;
- find Java EE application programming model and WebSphere APIs differences between the WebSphere profile types;
- learn any Java EE specification implementation differences that might affect the app.

Additionally, the tool provides a recommendation for the right IBM WebSphere runtime and Red Hat OpenShift platform. The tool also provides a complexity assessment to assess the ease of moving apps to Liberty or newer versions of WebSphere traditional. It will accelerate application migration to the cloud process, minimize application downtime and reduce cost.

This tutorial exercise is a part of the Application Modernization tutorial series which focus on the evaluation, re-platforming, and re-hosting modernization approaches. In this tutorial, you will learn how to use **Transformation Advisor** to show the value of using **Transformation Advisor** to evaluate on-premise Java applications and identify a migration candidate for moving to the cloud. When you complete this tutorial, you will know how to use the tool to identify and analyze on-premise Java applications without accessing their source code and to estimate the move to Cloud efforts.

Business Scenario

As shown in the image below, your company has a web application called **Red Rewards**, a WebSphere application showing the weather in various cities. Red Rewards

Figure 1. Red Rewards

Your company wants to move the application to a lightweight WebSphere Liberty server container on the cloud, but you don't have a lot of knowledge of the application, its dependencies and therefore are not sure how much effort the migration will take. You decide to use the **IBM Cloud Transformation Advisor** tool to quickly analyze the app and to identify if it is the right candidate application to move to cloud based on the analysis result.

In this tutorial, you will explore the following key capabilities:

- Learn how to collect Java application and configuration data using the **Transformation Advisor** command line tool.
- Learn how to use the Transformation Advisor to evaluate the move to cloud efforts and to identify the good candidate for migration

Log in to the tutorial VM and get started

This is the first task of the tutorial: starting the tutorial VPI.

Sign in to the **Workstation** VPI and get started

a. Launch the tutorial VPIs by clicking the **Play** button.

b. After the VPIs are started, click the **Workstation** VPI icon to access it.

c. Sign in to the **Workstation** VPI and get started

The **Workstation (Linux Desktop)** is displayed. You will execute all the tutorial tasks on this VPI.

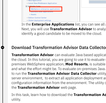
Review the on-prem WebSphere app

In this tutorial, we will use the Transformation Advisor applications deployed to the local WebSphere Application Server (WAS) environment. You are going to identify one of them to be the good candidate to move the cloud later.

a. Open WebSphere Application Server
In the **Workstation** VPI, you have a local WebSphere Application Server environment installed and configured.
To start the WAS server:

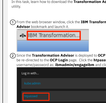
b. Open a terminal window by clicking its icon on the **Workstation** VPI desktop

c. In the terminal window, issue the command/script to start the WAS server.
/home/tbondemo/ipa-labs/scripts/startWAS.sh
When prompted, enter the **sudo** root password **passwordroot**

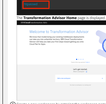


Wait for a few minutes the WAS server will be ready.

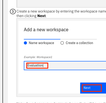
d. Review the WAS Admin Console to view the application deployed by clicking the **Firefox** icon on the Linux desktop toolbar to open a browser window.



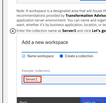
e. From the web browser window, click the **WebSphere Integrated Solution Console** bookmark to launch the WAS console.



f. In the WAS Admin Console login page, enter the User ID and Password as **wsadmin** **password** and click **Login**.



g. In the WAS console page, click **Applications > Application Types > WebSphere enterprise applications** to view all applications deployed.



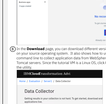
In the **Enterprise Applications** list, you can see all applications deployed. Next, you will use **Transformation Advisor** to analyze these applications to identify a good candidate to be moved to the cloud.

Download Transformation Advisor Data Collector utility

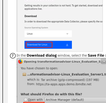
Transformation Advisor can evaluate Java based applications for migrating to the cloud. In this tutorial, you are going to use it to evaluate whether the on-premise Java app, Red Rewards, is a good candidate to move to the cloud. You should just what the effort might be. To evaluate on-premises Java applications, you need to run the **Transformation Advisor Data Collector** utility against the application server to collect application data. The application data includes application configuration information from the environment. The utility can be downloaded from the **Transformation Advisor** web page.

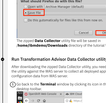
In this task, learn how to download the **Transformation Advisor Data Collector** utility.

a. From the web browser window, click the **IBM Transformation Advisor** bookmark to launch the Transformation Advisor.

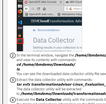


b. Since the **Transformation Advisor** is deployed in **OCP** on an operator, you will be directed to the **OCP Login** page. Click the **htpasswd** login option and enter the credentials/password: **ibmadmin/engagelab** and click **Log In.**





c. **Transformation Advisor Home** page is displayed.



d. Create a new workspace by entering the workspace name as **Evaluation** and click **Next**.

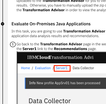
Note: A workspace is a designated area that will house the migration information for a specific application. Typically, a workspace is for a single application server environment. You can have one or more workspaces. However you want, whether it's by business application, location, or teams.

e. Create the collection name as **Server1** and click **Let's go.**

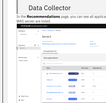
Note: Each workspace can be divided into collections for more focused assessment and planning. Like workspaces, collections can be named and organized in a way that is meaningful to you.

f. Once the **Workspace** and **Collection** are created, you will have options to either download the **Data Collector** utility or upload an existing data file. In this tutorial, you are going to use the **Data Collector** utility. Click **Data Collector** to download the utility.

g. In the **Download** page, you can download different versions of the utility based on your environment. For this tutorial, we are using WAS traditional, so we will download one to collect application data from WebSphere, WebLogic and Tomcat servers. Select the tutorial VPI is a Linux OS, click **Download Linux** to get the utility.



h. In the **Download dialog** window, select the **Save File** option and click **OK**.



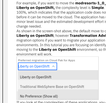
The zipped **Data Collector** utility file will be saved in **/home/tbondemo/Downloads** directory of the tutorial VPI desktop.

Run Transformation Advisor Data Collector utility

After downloading the Transformation Advisor data collector utility, you need to unpack it and run the utility against the WAS server where the application server is installed and configuration data from WAS server.

a. Go back to the **Terminal** window by clicking its icon in the **Workstation** VPI desktop toolbar

b. Unpack the data collector utility with command **tar -xvzf transformationadvisor-linux_Evaluation_Server1.tgz**
The data collector utility will be extracted.
Go to the extracted directory **/home/tbondemo/transformationadvisor-2.0.3** directory.
Run the data collector utility against the WAS server. Use the option to collect collecting the deployed applications information on the WAS server.
cd /home/tbondemo/Downloads/transformationadvisor-2.0.3
./bin/collectors/was/collect.sh -profiles=AppSrv01 -w /opt/IBM/WebSphere/AppServer -d AppSrv01 -wsadmin passwordpwd
When prompted, enter the **sudo** password as **passwordroot**



c. Unpack the data collector utility

d. Run the data collector utility with command **./collect.sh -profiles=AppSrv01 -w /opt/IBM/WebSphere/AppServer -d AppSrv01 -wsadmin passwordpwd**
The data collector utility will be started.

e. Enter the **sudo** password as root password **passwordroot**

f. Wait for the data collector to complete. Expect to see the following output:

In general, if your application server and the **Transformation Advisor** are installed on the same machine, you can use the **-profiles** option to run the data collector against multiple **Transformation Advisor** for data collection. In this example, WAS. Otherwise, you have to manually upload the data file to the **Transformation Advisor** in order to view the analysis results.

Evaluate On-Premises Java Applications

In this task, you are going to use **Transformation Advisor UI** to view the application data analysis results and recommendations.

a. Go back to the **Transformation Advisor** page in the web browser, click **Server1** in the **Collections** and **Recommendations**.

IBM Cloud Transformation Advisor
Home
Workspaces
Data Collections
Data has arrived! Application has been processed!
Data Collector
Recommendations
Server1
Data Collections
Recommendations
Data Collector
Upload
Data Collections
Recommendations
Data Collector
Upload
Data Collections
Recommendations
Data Collector
Upload

The first section in the detail analysis summary page is the **Complexity** section. The overall complexity for the **Mod Resorts** application is simple, indicating that the application can be directly moved to the cloud without any code change.

a. Scroll down to the **Application Details** section. You can see there is no code change required and no development cost, however, the application migration over all development cost is 5 days. This estimate is based on data from IBM Services engagements, which includes migrating management, server configuration, and testing.

b. Continue to scroll down to the Issues section. You can see that the only minor potential issue listed is on configuring the application in Docker container.

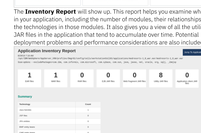
c. Next, scroll down to the bottom of the page and click the **Technology Report** link, this will open a new browser window to show the application **Evaluation Report**.

The report lists all java technologies that the application used and whether these technologies are supported by a specific WebSphere platform from Liberty for Java on IBM Cloud to WebSphere traditional for z/OS. It is used to determine whether a particular WebSphere product is suitable for an application.

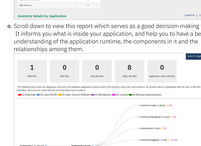
You can see from the report that since the **Mod Resorts** application only uses Java Servlet, the application is supported by all WebSphere platforms.

d. Go back to the **Transformation Advisor** page and click the **Inventory Report** link.

The **Inventory Report** will show up. This report helps you examine what is in your application, including the number of modules, their relationships and the technologies in those modules. It also gives you a view of all the utility JAR files in the application that tend to accumulate over time. Potential deployment problems and performance considerations are also included.



e. Scroll down to view this report which serves as a good decision-making tool. It informs you what is inside your application, and help you to have a better understanding of the application runtime, the components in it and the relationships among them.

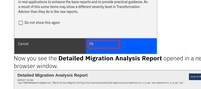


From the analysis reports you looked at above, you know that the **Mod Resorts** application is supported by **Liberty on OpenShift** as the target environment, and the issue that the tool identified would not affect the application migration. You can confidently select the application as a good candidate for moving to the cloud with minimum effort.

f. Go back to the **Transformation Advisor** page and click the **Analysis Report** link.

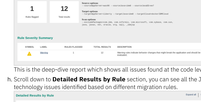
g. Click **OK** to continue.

Now you see the **Detailed Migration Analysis Report** opened in a new browser window.



This is the deep-dive report which shows all issues found at the code level.

h. Scroll down to **Detailed Results by Rule** section, you can see all the Java technology issues identified based on different migration rules.



For the **Mod Resorts** application, there is one warning rule regarding the application configuration in Docker containers.

i. Click the **Show results** link.
You can see the detailed analysis of the issue at the code level, in a specific class file and specific line. This will help developers to pinpoint where the issue is or potential issue may be.

j. Click the **Show rule help** link.

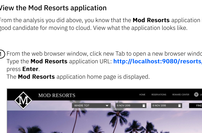
This will expand the **Rule Help** section which provides recommended solutions on how to fix the issue.

For the Docker container configuration issue, the utility provides best practice suggestions to externalize the configuration for the container.

View the Mod Resorts application

From the analysis you did above, you know that the **Mod Resorts** application is a good candidate for moving to cloud. View what the application looks like.

1. From the web browser window, click new Tab to open a new browser window.
   Type the **Mod Resorts** application URL: **http://localhost:9080/resorts/** and press **Enter**.
   The **Mod Resorts** application home page is displayed.



2. Click **WHERE TO?** dropdown menu to see the city list.

3. Click **PARIS, FRANCE** from the list to view the weather of the city.

Now that you reviewed the application, our next step is to move this application to cloud. Please go to the Part 2 of the tutorial series (Part 2) to continue.

Summary

In this tutorial, you learned how to evaluate the existing Java application by using IBM Cloud Transformation Advisor. As a part of IBM App Modernization solutions in **IBM Cloud Pak for Applications**, the Transformation Advisor tool provides a recommendation for the right-fit IBM WebSphere Application Server edition. It offers advice, best practices and potential solutions to assess the ease of moving apps to Liberty or to newer versions of WebSphere traditional. It accelerates the migration of application cloud, minimizes errors and risks, and reduces time to market. To learn more about IBM App Modernization solutions, please visit Cloud Pak for Applications.

How likely are you to recommend this tutorial to a friend or colleague?

Not at all likely — Extremely likely: 1 2 3 4 5 6 7 8 9 10

How can we improve? Or any other comments...

Submit

Cloud Enabled Use Case: App Modernization Journey Part 2 - Replatform

Step by Migration to Partial Java Application to OpenShift using Transformation Advisor

Before you start

The tutorial is meant to be used with a Linux environment, preferably a VM for Replatform. If you do not already have a running instance, please sign up for one.

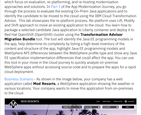
Please log in to be able to reserve an instance.

Tutorial

On the journey to cloud, enterprise customers are facing challenges moving their applications to a hybrid cloud environment. Red Hat OpenShift and the **IBM Cloud Pak for Applications** provides a consistent and consistent experience and solution to modernize enterprise applications for cloud-native deployments. Customers can take their existing Java EE applications, modernize the application tools and develop new cloud-native applications and deploy the deployment on Cloud. One of the tools included in the Cloud Pak is the **IBM Cloud Transformation Advisor**. Cloud Transformation Advisor is a cloud-based tool that is available at no charge to help you quickly analyze your enterprise Java EE applications for deployment to the cloud.

This tutorial exercise is a part of the Application Modernization tutorial series which focus on evaluation, re-platforming, and re-hosting modernization-centric Java EE applications. The IBM Cloud Transformation Advisor will guide you through the process to evaluate the existing On-Prem Java applications and to identify the candidate to be moved to the cloud using the IBM Cloud Transformation Advisor. In this tutorial, we will be using the Transformation Advisor to analyze the and DAR approach to move an existing application to the cloud. You learn how to package a selected candidate Java application to Liberty container and deploy it to **Red Hat OpenShift Container Platform (RHOCP) using Transformation Advisor**. **Migration Bundle** data. The tool automatically generates the Dockerfile, the app, help determine its complexity by running a high-level inventory of the application, and provide a report of the results. The report provides details on WebSphere API differences between the WebSphere profile and Liberty and Java SE 10 specification implementation differences that could affect the app. You can use the report to fix the issues and then rerun the analysis and packaging of the WebSphere apps without accessing source code and to prepare Java SE to a rapid cloud deployment.

In this tutorial, the primary in the image below, your company has a web application called **Mod Resorts**, a WebSphere application moving the weather in various locations. Your company wants to move this application from on-premises to the cloud.



As a quick start, you have already analyzed the application using the **Transformation Advisor** tool in Part 1 of the tutorial series. Based on the analysis you know that you can move this application from the traditional WebSphere environment to a Linux environment on OpenShift Container Platform without any code change. Now you are planning to use the **Transformation Advisor** migration plan to create the migration bundle, to containerize the application and to deploy the Docker container to an OpenShift Kubernetes cluster environment.

In this tutorial, you will explore the following key capabilities:

- Learn how Transformation Advisor can accelerate application migrating to cloud process, minimize errors and risks and reduce time to market
- Learn how to deploy the applications to a Linux Docker container environment using the migration bundle created by Transformation Advisor

Getting Started

In this task, you will started by accessing the tutorial VPN.

So Linux VPN Users have provided for this tutorial, The Red Hat OpenShift Container Platform **RHOCP v4.3**, is installed in a VPN, the **master1** VPN, the **master2** VPN, the **master3** VPN, and the **bastion** VPN. The **bastion** and 3 compute nodes the master nodes are running as compute nodes as well.

The CLI commands used in this tutorial are listed in the **Commands.txt** file located at the **/home/ibmdemo/cpd-labs/shared/utilities** directory. The **Workstation** VPN for you to copy and paste these commands to the Terminal window during the tutorial.

1. Log in to the **Workstation** VPN and get started.
   a. Launch the tutorial VPN by clicking the **Start** button.

   b. When the VPNs are started, click the **Workstation** VM icon to access it.

   The **Workstation Linux Desktop** is displayed. You will execute all the tutorial labs on this VM.

Review the On-Prem App

In this task, you will take a look at **Mod Resorts** application deployed to the local WebSphere Application Server. The application is ready for you to start migrating to cloud application to the cloud using Open Liberty Operator later.

1. Start WebSphere Application Server
   a. Open **workstation** VPN, you have a local WebSphere Application Server VPN which hosts the **Mod Resorts** application.
   b. Start the WebSphere server.
   c. Open a terminal window by clicking its icon on the **Workstation** VPN desktop and run

   d. Wait for terminal prompting, once the server started to run the WAS VPN server, **/home/ibmdemo/cpd-labs/shared/utilities/d.sh** shell script and enter the **wsadmin** user password of **passwd4d**

   e. Wait for terminal stating WSVR0001I: Server dmgr open for e-business

2. Open WebSphere Admin Console
   a. Open **workstation** VPN, you have a local WebSphere Application Server VPN which hosts the **Mod Resorts** application.
   b. Start the WebSphere server.
   c. Open a terminal window by clicking its icon on the **Workstation** VPN desktop and run

   d. Wait for terminal stating, once the server started to run the WAS VPN server, **/home/ibmdemo/cpd-labs/shared/utilities/d.sh** shell script and enter the **wsadmin** user password of **passwd4d**

   e. Wait for terminal stating WSVR0001I: Server dmgr open for e-business

3. Open the WebSphere Admin Console login page, enter the User ID and Password
   a. **wsadmin** password and click **Login**.
   b. On the WAS Console page, click **Applications** → **Application Types** → **WebSphere enterprise applications** Choose the dropdown menu

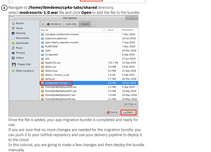
   c. In the **Enterprise Applications** list, you can see all applications deployed. Click **Mod Resorts** application name to open the properties page.

   d. Scroll down to the **Modules** section, in the page application, you can deployed on to a Web server.

4. Open the **Mod Resorts** application
   a. From the web browser window, click new Tab to open a new browser window. Type the **Mod Resorts** application URL **https://localhost:9443/modresorts/** and press **Enter**
   b. The **Mod Resorts** application home page is displayed.

MOD RESORTS
Home
Hotels
Flights
Cars
Cruises
Local Attractions
Travel Deals
Weather
About Mod Resorts
Contact Us
Sign Out
Welcome, Guest
Hotels in Paris
Hotels in London
Hotels in Rome
Hotels in Barcelona
Hotels in Prague
Hotels in Athens
Hotels in Amsterdam
Hotels in Vienna
Hotels in Budapest
Hotels in Lisbon
Hotels in Madrid
Hotels in Berlin
Hotels in Venice
Hotels in Florence
Hotels in Munich
Hotels in Milan
Hotels in Istanbul
Hotels in Athens
Hotels in Zurich
Hotels in Geneva
Hotels in Brussels
Hotels in Copenhagen
Hotels in Stockholm
Hotels in Oslo
Hotels in Helsinki
Hotels in Warsaw
Hotels in Krakow
Hotels in Budapest
Hotels in Prague
Hotels in Vienna
Hotels in Munich
Hotels in Berlin
Hotels in Hamburg
Hotels in Frankfurt
Hotels in Cologne
Hotels in Dusseldorf
Hotels in Stuttgart
Hotels in Leipzig
Hotels in Dresden
Hotels in Nuremberg
Hotels in Hannover
Hotels in Bremen
Hotels in Essen
Hotels in Dortmund
Hotels in Bonn
Hotels in Mannheim
Hotels in Karlsruhe
Hotels in Wiesbaden
Hotels in Augsburg
Hotels in Bielefeld
Hotels in Bochum
Hotels in Wuppertal
Hotels in Bremerhaven
Hotels in Kiel
Hotels in Lubeck
Hotels in Rostock
Hotels in Potsdam
Hotels in Erfurt
Hotels in Jena
Hotels in Weimar
Hotels in Eisenach
Hotels in Gotha
Hotels in Ilmenau
Hotels in Suhl
Hotels in Meiningen
Hotels in Schmalkalden
Hotels in Arnstadt

③ Navigate to **/home/demoness/ryle-labs/shared** directory, select **modresorts-2.0.war** file and click **Open** to add the file to the bundle.



Once the file is added, your app migration bundle is completed and ready for export.

If you are same that no more changes are needed for the migration bundle, you can push it to your GitHub repository and use your delivery pipeline to deploy it to the cloud.

In this lab, we are going to make a few changes and then deploy the bundle manually.

Update the Migration Bundle

After the migration bundle is created, you need to do one additional update.

① Click **Download** to download it to the local machine.

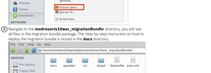
② Navigate to **/home/demoness/Downloads** directory.

③ Rename the downloaded bundle zip file and select **Extract Here** to unzip the bundle zip file in **/home/demoness/Downloads** directory.

④ Navigate to **/home/demoness/Downloads/modresorts-2.0war_migrationBundle** directory, you will see all files in the migration bundle package. The step-by-step instruction on how to replay the migration bundle is stored in the **docs** directory.

⑤ **/modresorts-2.0war_migrationBundle/src/main/liberty/config** directory contains all configuration files. Edit the **server.env** file by selecting the file, right clicking and opening it with **Text Editor**.

⑥ Change the server feature from **2.0 → 2.1**, because the application requires servlet 3.1 on Liberty server and the default highest supported version in WAS Liberty v855 is servlet 3.0. Click **Save** to save the changes.



Testing the Migration Bundle Locally (Optional)

You have the **Mod Resorts** application migration package. In general, before deploying any application to the OpenShift cluster, you need to deploy it to a local Liberty server for testing first. To save time, you can skip this task, but the steps for testing are documented here for your reference.

① Go back to the terminal window and issue the command below to stop the WAS server.

```
cd /home/demoness/IBM/WebSphereAppServer/profiles/AppSrv01/bin/stopServer.sh
server1
```

If prompted, enter the was.policy password as **passw0rd**, and WAS admin username and password as **wasadmin**.

② Create a Liberty server named **server1** in the Liberty installation directory to test the application.

```
/home/demoness/ryle-labs/shared/wlp/bin/server create server1
```

③ Copy the application binary file in the migration bundle **target** directory to **/usr/servers/server1/apps** directory.

```
cp -rpm /home/demoness/Downloads/modresorts-2.0war_migrationBundle/target/*.war
/home/demoness/ryle-labs/shared/wlp/usr/servers/server1/apps
```

④ Change the ownership to Liberty runtime user.

```
/home/demoness/Downloads/modresorts-2.0war_migrationBundle/src/main/liberty/config/server.env
/home/demoness/ryle-labs/shared/wlp/usr/servers/server1/server.env
```

⑤ Start the Liberty server.

```
/home/demoness/ryle-labs/shared/wlp/bin/server start server1
```

⑥ Check the Liberty logs to confirm that your application has started correctly and listening on the HTTP port 9080.

```
cat /home/demoness/ryle-labs/shared/wlp/usr/servers/server1/logs/messages.log
```

You will see from the log message the application is available with a URL `http://localhost:9080/modresorts/`.

⑦ Open the web browser window, type the application as **localhost:9080/modresorts**, and press **Enter**. You will be able to access the application from the local Liberty server.

Containerize Liberty Application

In this task, you will containerize the application. You will create a Liberty image that has your migrated application installed and working, and then test the image to ensure it is operational.

① From the Terminal window issue, the command below to stop the Liberty server if it is running to ensure that the necessary ports are available.

```
/home/demoness/ryle-labs/shared/wlp/bin/server stop server1
```

② Go to where your migration artifacts are located and build your image from the Dockerfile.

```
cd /home/demoness/Downloads/modresorts-2.0war_migrationBundle
docker build --no-cache -t default-route-openshift-image-
registry.apps.demo.demofie.net/modresorts-2.0war/modresorts-2.0war:latest .
```

where:

- **default-route-openshift-image-registry.apps.demo.demofie.net** is the OpenShift Image Registry URL, and
- **modresorts-2.0war** is the name of namespace in OpenShift that you are using to host the image
- **modresorts-2.0war:latest** is the application container image.

The base Liberty image will be pulled down and used to create an image that contains your application.

Once the Docker image is built, create a container instance from the image and monitor that it is starting correctly.

```
docker run -p 9080:9080 default-route-openshift-image-
registry.apps.demo.demofie.net/modresorts-2.0war/modresorts-2.0war:latest
```

If everything looks good, the image has been started and mapped to the port 9080. You can access it from your browser with this:

http://localhost:9080/modresorts/

③ After testing, press **Ctrl+C** to stop the container.

Deploy the Application to OpenShift

In this step, you will deploy the docker image you have created to Red Hat OpenShift Container Platform (RHOCP). Before you begin, you will need to push the image to the OpenShift image registry.

Note: The migration artifacts generated by **Transformation Advisor** (specifically the operator/application/application-cr.yaml file) assume that the default Docker Registry is being used. If you choose to use a different registry, remember to update the image name in the YAML file appropriately.

① From the Terminal window, issue the command below to login to the **RHOCP**.

```
docker login -u kubadmin -p xrgyg-mrfbm https://api.demo.demofie.net:6443
```

② Create a new namespace called **modresorts2war** where you are going to host the application.

```
oc new-project modresorts2war
```

You will see the message to confirm that you are in the demo namespace.

```
Now using project "modresorts2war" on server "https://api.demo.demofie.net:6443".
You can add applications to this project with the 'new-app' command. For example, try:
    oc new-app rails-postgresql-example
to build a new example application in Perl. Or use kubectl to deploy a simple Kubernetes application:
    kubectl create deployment hello-node --image=gcr.io/hello-minikube-zero-install/hello-node
```

③ Log in to the OpenShift Docker registry with the command.

```
docker login -u demadmin -p Bjsc mhcwmr -t default-route-openshift-image-registry.apps.demo.demofie.net
```

④ Change the image name to push your docker image to the OpenShift image registry.

Note: The migration artifacts generated by **Transformation Advisor** (specifically the operator/application/application-cr.yaml file) assume that the default Docker Registry is being used. If you choose to use a different registry, remember to update the image name in the YAML file appropriately.

⑤ From the Terminal window, issue the command below to login to the **RHOCP** Web Console at **https://api.demo.demofie.net:8443**

⑥ Create a new namespace called **modresorts2war** where you are going to host the application.

```
oc tag --source=docker default-route-openshift-image-registry.apps.demo.demofie.net/modresorts2war/modresorts2war:latest
modresorts2war/modresorts2war:latest --insecure=true
```

⑦ Everything looks good, the image has been stored and mapped to the port 9080. You can access it from your browser with this:

http://localhost:9080/modresorts/

⑧ After testing, press **Ctrl+C** to stop the container.

Deploy the Application to OpenShift

In this step, you will deploy the docker image you have created to Red Hat OpenShift Container Platform (RHOCP). Before you begin, you will need to push the image to the OpenShift image registry.

Note: The migration artifacts generated by **Transformation Advisor** (specifically the operator/application/application-cr.yaml file) assume that the default Docker Registry is being used. If you choose to use a different registry, remember to update the image name in the YAML file appropriately.

① From the Terminal window, issue the command below to login to the **RHOCP**.

```
docker login -u demadmin -p Bjsc mhcwmr -t default-route-openshift-image-registry.apps.demo.demofie.net
```

② Change the image name to push your docker image to the OpenShift image registry.

```
docker push default-route-openshift-image-
registry.apps.demo.demofie.net/modresorts2war/modresorts2war:latest
docker tag docker.io/application-docker-image:v1 default-route-openshift-image-
registry.apps.demo.demofie.net/modresorts2war/modresorts2war:latest
```

③ Log in to the **RHOCP** image registry.

④ Verify the pushed Docker image in **RHOCP**.

a. Click the web browser icon on the desktop toolbar to go back to the browser window.

b. Click **OpenShift web console** bookmark to access the **RHOCP Web Console**.

c. If the **RHOCP Web Console** Login page is displayed. Click the **Htpasswd** field, accept the default username/password and click **Log In**.

d. From the **RHOCP** web console home page, click **Builds>Image Streams**.

e. Change project to namespace/ns/ns **default** → **modresorts2war**.

f. You can see the image you just pushed is listed. Click its link to view its details.

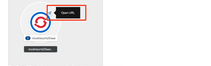
In the **Image Stream Details** page **Overview** section, you will see the image repository and its tag information. In a production environment, a public image repository is mapped to an internal image repository which will be used to deploy the application. The internal image repository is **image-registry.openshift-image-registry.svc:5000/modresorts2war/modresorts2war:latest**.

⑤ Change the image type from **Administrator** → **Developer**.

⑥ Navigate to **Topology** page. Click **Container Image** to add new content (deployment) to the project using an container image.

⑦ In the **Deploy Image** page, select the **Image name from internal registry** option, and set the **Project**, **ImageStream** and **Tag** values **modresorts2war**, **modresorts2war** and **latest**.

⑧ Scroll down to the **General** section, accept all default settings and click **Create**.



⑨ Your application is now deployed. To view it, click the **Open URL** link.

The **Open Liberty** home page is displayed in a new browser window.

⑩ Open the application context root **/resorts** to the end of the URL and navigate through the **Mod Resorts** application home page to displayed.

⑪ Navigate through the application web pages as you did in the WAS version to verify the application deployment.

Summary

In this tutorial, you have learned how to use Transformation Advisor to prepare a migration bundle for your application and deploy it to cloud. As a part of IBM App Modernization solutions in IBM **Cloud Pak for Applications**, the Transformation Advisor will accelerate application migrating to the cloud process, minimize errors and reduce time to market. To learn more about IBM App Modernization solutions, please visit Cloud Pak for Applications.

⑫ Open the application context root **/resorts** to the end of the URL and navigate through the **Mod Resorts** application home page to displayed.

⑬ Navigate through the application web pages as you did in the WAS version to verify the application deployment.

Cloud Enabled Use Case: App Modernization Journey Part 3 - Rehost

This tutorial provides instructions on how to rehost a sample application on WebSphere Application Server 9000 environment by a BMX container without any code change and to deploy it to the Kubernetes based Red Hat OpenShift Container Platform 4.6 on IBM Cloud.

1 Tutorial

On the journey to cloud, enterprise customers are facing challenges moving their existing on-premises applications to cloud quickly and effortless.

The **IBM Cloud Pak for Application** provides a complete and consistent hybrid platform for application modernization, integration and automation deployments. Customers can quickly modernize their existing applications with IBM's integrated tools and develop new cloud-native applications faster for deployment on any cloud.

One of the tools included in the Pak is the **IBM Cloud Transformation Advisor**. **Transformation Advisor**, a tool that is available at no charge to help customers quickly identify and analyze their existing WebSphere applications for a rapid cloud deployment.

Transformation Advisor provides recommendations for the right-fit IBM WebSphere Application Server editions, and offers advice, best practices, and potential solutions to assess the ease of moving apps to Liberty or newer versions of WebSphere Application Server, and to identify the best-fit application environments to cloud while minimizing cost and reducing time to market.

This tutorial exercise is a part of the Application Modernization tutorial series, Part 3 of the Application Modernization Journey series, focused on rehosting, modernization approaches and solutions.

In Part 3 of the App Modernization Journey (Evaluation), you go through the process to evaluate the existing On-Prem Java applications and to identify the candidates to be rehosted to the cloud using the Transformation Advisor.

In Part 1 of the App Modernization Journey (Rehost), you learn how to move a traditional WebSphere application into a **Liberty container** without any code change and deploy it to **Red Hat OpenShift Container Platform**. **ROKS**K environment using the **Transformation Advisor.**

This tutorial showcases the re-hosting process. The host uses Lift and Shift approach to move the application binaries and configurations to a container image. The host now runs in a container. The host now runs in a container. See application from a traditional **WebSphere Application Server (WAS)** environment to a **WAS** container environment. The host runs in a container. The host runs in a **ROKS**K environment.

Transformation Advisor provides the following tasks:

Assuming that as illustrated below, your company has a web application called **Mod Resorts**, a WebSphere application showing the weather in various locations. Your company wants to move this application from on-premises to OPen to the cloud (in this case, IBM Cloud). The **Mod Resorts** application is a Java application improved upon **Coffee Cafe** for the application by leveraging scalable cloud capabilities with **IBM Cloud Pak for Applications** that the enterprise currently deploys with its cloud native application.

As a tech lead, you have already analyzed the application using **Transformation Advisor** and found that it is a good candidate. Based on the analysis you know that you can move this application from the traditional WebSphere Server ND cell environment to a containerized WAS Base server environment without any code change.

Now you are planning to package the application in WAS Base container and to deploy it to Red Hat OpenShift Container Platform cluster.

By moving an application from a traditional WAS ND cell to a WAS Base container in cloud, you can take advantage of cloud to support application high availability and scalability, to reduce operational costs, to improve time to market, to simplify infrastructure and operation cost while leveraging your team's existing WebSphere admin skill.

In this tutorial, you will explore the following key capabilities:

- Understand how to move a WebSphere app from an on-premises environment to the OpenShift cluster without any changes to the app code, while utilizing existing WebSphere administration skills.
- Learn the process to create a WAS Docker container.
- Get familiar with the WAS Base container on OpenShift deployment process.

Start the tutorial environment

In this task, log in the **Workstation** VM and get started.

Sra Linux VPs have been provided for this:

- The Red Hat OpenShift Container Platform, **BOSCP v4.2**, is installed in 4 VPs, the **master1** VPs, the **master2**VPs, the **dns** VPs, and the **worker2** VPs, with 2 worker nodes and 3 control planes nodes (the master nodes are among the worker nodes as well).
- The **Workstation** VPs is the one you use to access and work with **BOSCP**. **BOSCP** is installed in the **Workstation** VP and
  User ID: **ibmdemo**
  Password: **passw0rd**
  The **Workstation** VP is shared in the **Workstation** VM Terminal for **sudo** in the lab.
- The CLI commands used in this lab are listed in the **Commands.txt** file located at the **/home/ibmdemo/pak-labs/was900base** directory of **Workstation** VP. You can copy and paste these commands to the **Terminal** window during the tutorial.

1. Launch the tutorial VPs by clicking the **Play** button.

2. Once the VPs are started, click the **Workstation** VM icon to access it.

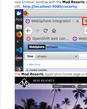
3. The **Workstation Linux Desktop** is displayed. You will exercise all the tutorial tasks on the VM.

Review the on-prem WebSphere application

In this task, review the **Mod Resorts** application deployed to the local WAS cell environment. This application will be ported to a WAS container later.

1. Click the **Terminal** icon to open a terminal.
   The **Workstation** VM also have a local WebSphere Application Server which hosts the **Mod Resorts** application. To start the server:
   a. Open a terminal window by clicking its icon on the **Workstation** VM desktop

   b. In the terminal window, issue the command below to start the **WAS** server instance. Enter the **passw0rd** when prompted for the password:
   /home/ibmdemo/pak-labs/was900base directory
   /home/ibmdemo/pak-labs/shared/ear0814.sh
   alternatively, type the **sudo /home/ibmdemo/pak-labs/shared/ear0814.sh**



   Within a few minutes, the WAS server will be ready.
2. Now the on-prem application deployed on the WAS server.
   a. Access the WAS Admin console to see the applications deployed by clicking the **WAS Admin console** bookmark on the Firefox web browser window.

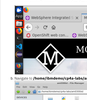
   b. Log in to the WAS Admin Console login page, enter the User ID and Password: User ID: **ibmadmin** Password: **passw0rd**
   c. In the **WebSphere Enterprise Applications** > **Applications** > **Application Types** > **WebSphere enterprise applications** to view the applications deployed.

   d. The **Enterprise Applications** list, you can see all applications deployed. The **Mod Resorts** application is the list. It is the application you are going to move it to WAS container and to deploy it to the OpenShift cluster.

3. Review the **Mod Resorts** application.
   a. From the web browser window, click the **Open a new tab** icon to open a new browser window with the **Mod Resorts** application
   http://localhost:9080/modresr/

   Mod Resorts application home page is displayed.

4. Click the **MOUNT TOP** button menu to see the cities list.



5. Click **PARIS, FRANCE** to see the local time and the weather of the city.

   You have reviewed the application. Next you have to port/move the application in WAS container without any code change and to deploy it to an OpenShift cluster.
6. Stop the WAS server by issuing the following command in the Terminal window:
   stop
   /opt/IBM/WebSphere/AppServer/profiles/Dmgr01/bin/stopServer.sh server1
   When prompted enter the WAS password as **passw0rd**, then enter the WAS server nodestate as **wasadmin/passw0rd**

Build a WAS Base server container image

In this task, build a WAS Base server Docker container image with the **Mod Resorts** application deployed.

Note: According to Docker's best practices you should create a base WAS Base image, which adds a single application and the corresponding configuration. You should then use this base image as the starting point for building the images for all application servers. It is debugging process, because any changes you make to the present if you spawn a new container from the image.

There are few key files you needed to build your WAS Base server container image:

- Dockerfile – the file defines how the Docker image that has your application and configuration pre-loaded is built.
- App runtime – the ear or war file of your application.
- Automation scripts – the file defines how to install application in the WAS server.
- wpsConfig.py – the WAS admin scripts in Jython format configures the WAS server for your application.
- FIXCMDSH2 – the file contains the WAS server runtime password use in the WAS container.

1. Review **Dockerfile**, **app-install.props** file, and **appConfig.py** file.
   a. From the Desktop tool bar, click the **File Manager** icon to open it.

   b. Navigate to **/home/ibmdemo/pak-labs/was900base** directory.

   c. Double click the **Dockerfile** to open it in **Text editor** for reviewing.

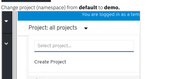
```
# Base image
FROM ibm.com/websphere-traditional:8.5.0.3

#Workloads powered for admin console

COPY FIXCMDSH2 /var/FIXCMDSH2

COPY wpsConfig.py /work/config/

COPY app install.props /work/config/app-install.props

COPY mod-resorts-1.0.war /work/config/mod-resorts-1.0.war

RUN /work/configure.sh

# Define commonly used ports.
EXPOSE 9060 9043 9080 9443

# Start server
CMD /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin/startServer.sh server1 -trace -traceSpec com.ibm.websphere.*=all:com.ibm.ejs.*=all -jvmDebugPort 7777 -jvmDebugSuspend false -disableJettyTransport true -disableJettyAdminConsole true -disableJettyAdminApp true -disableJettyDefaultApp true -disableJettyDocs true -disableJettyTest true -disableJettyWebProtocol true -disableJettyWebservices true -disableJettyWebsocket true -disableJettyWebtop true -disableJettyWelcome true -disableJettyWsdl true -disableJettyWsdm true -disableJettyWsfw true -disableJettyWsrr true -disableJettyWsrp true -disableJettyWsrr10 true -disableJettyWsrp10 true -disableJettyWsrp20 true -disableJettyWsrp21 true -disableJettyWsrp22 true -disableJettyWsrp23 true -disableJettyWsrp24 true -disableJettyWsrp25 true -disableJettyWsrp26 true -disableJettyWsrp27 true -disableJettyWsrp28 true -disableJettyWsrp29 true -disableJettyWsrp30 true -disableJettyWsrp31 true -disableJettyWsrp32 true -disableJettyWsrp33 true -disableJettyWsrp34 true -disableJettyWsrp35 true -disableJettyWsrp36 true -disableJettyWsrp37 true -disableJettyWsrp38 true -disableJettyWsrp39 true
```

accept the default username/password and click **Log In.**

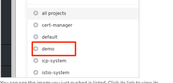
The **RHOCP Web Console Home** page is displayed.

d. From the **RHOCP Web Console Home** page, click **Builds>Images Streams.**

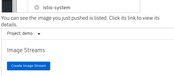
e. Change project (namespace) from **default** to **demo.**



f. You can see the image you just pushed is listed. Click its link to view its details.



In the **Image Stream Details** Page **Overview** section, you will see the public image repository you used to push the image. Notice that the public image repository is mapped to the internal image repository which will be used to deploy the application. The internal image repository is: **image-registry.openshift-image-registry.svc:5000/demo/modresorts-twas**



4. Deploy the WAS container to RHOCP cluster

Once the WAS container image is pushed to the **RHOCP** image registry, you can deploy the WAS container to the **RHOCP** cluster with OpenShift CLI.

1. View the deployment YAML files. In this tutorial, three YAML files are used to deploy the WAS container, these files are:
   - deploy.yaml – create a WAS container in the OpenShift cluster
   - service.yaml – create a service for the WAS container
   - route.yaml – create a route of the WAS container service

   a. From the **File Manager** window, navigate to **/home/ibmdemo/cp4a-labs/am030ivt/deploy** directory where these three YAML files are located.

   b. Double-click the **deploy.yaml** file to open it in the text editor. The file contains the basic requirements of the WAS container deployment. The container name is set as **modresorts-app-twas** and it is deployed to the **demo** namespace.

   c. Double-click the **service.yaml** file to view its contents. The file exposes the **Pod** Resources application ports **9080** and **9443**. The port type in the OpenShift cluster is set as **ClusterIP**.

   d. Double-click the **route.yaml** file to view its contents. The file defined the OpenShift route used for the WAS container as **modresorts-app-twas.apps.demo.ibmdte.net** which enables users to access the **Mod Resorts** application on a public address.

2. Now you can deploy the WAS container to **RHOCP** cluster with these YAML files. From the Terminal window, change to the **/home/ibmdemo/cp4a-labs/am030ivt** directory.
   cd /home/ibmdemo/cp4a-labs/am030ivt
3. Issue OpenShift CLI command to deploy the WAS container to the OpenShift cluster:
   oc apply -f deploy
   This command will deploy the three YAML files from the deploy directory and create WAS container pod, service and route in the **demo** namespace.

5. Verify the deployment

In this task, access the OpenShift Web Console to verify the WAS container deployment.

1. Go back to the **OpenShift Web Console**, click **Workloads>Pods**, then select **demo** project from the project list.

2. You will see that the WAS container is deployed and is in running status. You can click its name link to go to its overview page where you can see its memory and CPU usage and other detailed information.

3. Now from the **OpenShift Wen Console** navigation panel, click **Networking>Services**. You will see the WAS container service. Click the service name to view its details.

In the Service Details page, you can see the service ports as you defined in the **service.yaml** file.

4. Next, navigate to **Networking>Route** to view the WAS container service route. Click the Location URL to launch the **Mod Resort** application.

5. The **Mod Resort** application URL will be launched in a new web browser window. Type the **Mod Resorts** application context root **resorts** in the end of the URL and press **Enter**.

6. Click **Advanced>Accept Risk and Continue**, the **Mod Resorts** application home page is displayed.

7. Click **WHERE TO?** to view city list.
8. Click **LAS VEGAS, USA** from the list, the weather of the city is displayed

Summary

In this tutorial, you learned the App Modernization re-platform process. You learned how to move an existing traditional WAS application to WAS Base container and how to deploy it to an OpenShift cluster without any code change. IBM Cloud Pak for Applications, built on the Red Hat OpenShift Container Platform, provides a long-term solution to help you transition between public, private, and hybrid clouds, and to create new business applications. As a key component of **IBM Cloud Pak for Applications**, traditional WebSphere application Server can still run in containers, and reap the benefits of consistency and reliability that containers provide.

Multicloud Management Use Case: App Centric Management with Application Navigator

Jay Thoden van Velzen, Jonathan Sloan, Lisa Romero
Jay Thoden van Velzen is a Senior Technical Staff Member (STSM) and Application Navigator lead on a large portion of IBM Cloud Paks and other Hybrid Cloud offerings, including Multicloud Management, Security, Liberty Collective and Automation Companion. The duration of the talk is about 45 minutes.

Tutorial

One of the challenges customers are facing in their Journey to Cloud is how to conduct "Day 2 operations" in a Multicloud environment. Its quickly determine the health of your apps, find the root cause of a problem, remediate problems, manage cases, track and incorporate changes. **IBM Cloud Paks for Applications** provides tools and capabilities for different such issues. Included in this pak is IBM Application Navigator, IBM Cloud Pak for Watson AIOps, and Automation Companion, which complements IBM Net Cool/Impact. Application Navigator, by allowing you to visualize, inspect, and interact with the deployed resources that are managed by Kubernetes, Red Hat OpenShift, IBM Cloud Pak for Multicloud Management applications that is designed to facilitate an app centric management experience for cloud native and hybrid applications, across both cloud and traditional platforms. By integrating with Multicloud Management, Application Navigator provides a single view across your business application, combining with both kind specific out of the box and customizable dashboards. **App Navigator** enhances operational productivity by providing a single pane of glass for managing your cloud native applications. It puts the tools you need most in context and at your fingertips and yields a superior user experience for applications that dramatically surpasses the traditional approach of using multiple consoles and platform consoles.

Here is the business scenario this lab covers:

Your company has started a project to break a big monolithic application into different components. The IBM Cloud Transformation Advisor, which analyzes the insight on how to modernize the application the best way. You replaced parts of the monolithic application with new cloud native microservices and deployed successfully to a Kubernetes cluster environment. In addition, you moved some application components to the right managed Liberty Collective environment. Unfortunately, not all parts of the application are ready to be modernized and remains in the traditional existing WebSphere ND cell infrastructure. So finally, you ended up with a hybrid application which runs across IMB Cost, Liberty Collectives and traditional WAS ND cells. One of the operations team is exactly for us if your application is running fine or not.

In order to quickly determine the root cause of an issue and get the information needed to make decisions for the application across IMB Cost, Liberty Collectives and Containers, you want to use the App Navigator tool to be a single access point to view/navigate/interact with the applications and their components.

In this tutorial, you will explore the following key capabilities:

- Learn how to deploy a cloud native application in a Kubernetes cluster
- Learn how to configure WAS ND Cell with App Navigator
- Learn how to configure Liberty Collective with App Navigator
- Learn how to use App Navigator as an inspection tool to facilitate monitoring and problem determination for WAS ND, Liberty Collective and Kubernetes Clusters

Log in to the Workstation VM and get started

In this task, you start the tutorial VM.

Six Linux VPs have been provided for this tutorial.

- The Red Hat OpenShift Container Platform **BINOCP v4.3**, is provisioned in 4 VPs, the **master1** VM, the **master2**VM, the **dev** VM, and the **worker2** VM, with 2 worker nodes and 3 Compute nodes (the master nodes are also provisioned as worker nodes).
- The **IBM Cloud Pak for Applications 3CPAK** v4.1 is installed on the **BINOCP**
- The **App Navigator** has been deployed in **BINOCP** cluster as a part of **IBM Cloud Pak for Applications**. Navigate to the Navigator UI and learn how to install **IBM Cloud Pak for Applications** on OpenShift, please refer to https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.0/install.html
- The **Workstation** VM is the one you will use to access and work with **BINOCP** in the tutorial.
  The login credentials for the **Workstation** VM are:
  User: **admin**
  Password: **passw0rd** (this is the same **sudo** password).
- The OLI commands used in this tutorial are stored in the **Commands.txt** file located at the **/home/demouser/ycha-labs/mwitt360/** directory of the **Workstation** VM for you to copy and paste these commands to the Terminal window during the tutorial.

Launch the tutorial VPs by clicking the **Play** button.

After the VPs are started, click the **Workstation** VM icon to access it.

The **Workstation** desktop is displayed. You will execute all the tutorial tasks on this VM.

Deploy a Cloud Native Application to a Kubernetes cluster

In this task, you will go to directory **stockholder**, a sample cloud native application, and use **OCP** to create its using **OCP** CLI.

Click the **Workstation** desktop, click the **Terminal** icon to open **Terminal** window.

In the **Terminal** window, navigate to **/home/demouser/Student/ycha-labs/mwitt360/stockholder** directory, type **cd /home/demouser/ycha-labs/mwitt360/**.

Log in **BINOCP Web Console** with command:
oc login https://api.ocp.demo.ibmdte.net:6443
Log into a demonstration **oc** environment **https://api.demo.ibmdte.net:6443**
Create the **stockholder** project:
OpenShift uses namespace to host cloud resources. In this tutorial you are going to create a namespace called **stockholder**, and deploy the cloud native application to the cloud native application **stockholder** namespace. Create the namespace with the following command:
oc new-project stockholder
or oc new-project stockholder
The **demo** namespace and **customer** namespace are now created.

Deploy the **stock-trader** cloud native application with the command below:
oc new-app -f stock-trader.yaml -n stockholder

```
[demouser@ycha-labs-mwitt360 stockholder]$ oc new-app -f stock-trader.yaml -n stockholder
--> Deploying template "stockholder/stock-trader" for "stock-trader.yaml"
    mongodb-persistent
    ------------------
    MONGODB_USER = user
    MONGODB_PASSWORD = mongopass
    MONGODB_DATABASE = sample
    MONGODB_ADMIN_PASSWORD = admin123
    MONGODB_SERVICE_NAME = mongodb
    MONGODB_REPLICA_NAME = rs0
    MONGODB_KEYFILE_VALUE = supersecretkeyfortheapp
    MONGODB_MEMORY_LIMIT = 512Mi
    MONGODB_VOLUME_CAPACITY = 512Mi
    MONGODB_SERVICE_PORT = 27017
    MONGODB_ADMIN_PORT = 27018
    MONGODB_REPLICA_PORT = 27019

    stock-trader
    ------------
    ENVIRONMENT = development
    MONGO_DB_HOSTS = mongodb:27017
    MONGO_DB_USERNAME = user
    MONGO_DB_PASSWORD = mongopass
    MONGO_DB_DATABASE = sample
    MONGO_DB_REPLICASET = rs0
    MONGO_DB_SSL = false
    MONGO_DB_SSL_VALIDATE = false
    MONGO_DB_SSL_CA = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    MONGO_DB_SSL_CERT = /var/run/secrets/kubernetes.io/serviceaccount/tls.crt
    MONGO_DB_SSL_KEY = /var/run/secrets/kubernetes.io/serviceaccount/tls.crt
    MONGO_DB_SSL_PASSPHRASE = supersecretkeyfortheapp
    MONGO_DB_AUTHSOURCE = admin
    MONGO_DB_AUTHMECHANISM = SCRAM-SHA-1
    MONGO_DB_REPLICA_NAME = rs0
    MONGO_DB_SERVICE_NAME = mongodb
    MONGO_DB_SERVICE_PORT = 27017
    MONGO_DB_ADMIN_PORT = 27018
    MONGO_DB_REPLICA_PORT = 27019
    MONGO_DB_ADMIN_PASSWORD = admin123
    MONGO_DB_ADMIN_USERNAME = admin
    MONGO_DB_ADMIN_AUTHSOURCE = admin
    MONGO_DB_ADMIN_AUTHMECHANISM = SCRAM-SHA-1
    MONGO_DB_ADMIN_SSL = false
    MONGO_DB_ADMIN_SSL_VALIDATE = false
    MONGO_DB_ADMIN_SSL_CA = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    MONGO_DB_ADMIN_SSL_CERT = /var/run/secrets/kubernetes.io/serviceaccount/tls.crt
    MONGO_DB_ADMIN_SSL_KEY = /var/run/secrets/kubernetes.io/serviceaccount/tls.crt
    MONGO_DB_ADMIN_SSL_PASSPHRASE = supersecretkeyfortheapp

--> Creating resources ...
    secret "mongodb" created
    secret "stock-trader" created
    service "mongodb" created
    service "mongodb-replica" created
    service "mongodb-admin" created
    service "stock-trader" created
    deploymentconfig.apps.ocp.io "mongodb" created
    deploymentconfig.apps.ocp.io "stock-trader" created
    persistentvolumeclaim "mongodb" created
    imagestream.image.openshift.io "stock-trader" created
    buildconfig.build.openshift.io "stock-trader" created
    deploymentconfig.apps.ocp.io "stock-trader" created
    route.route.openshift.io "stock-trader" created
--> Success
    Application is not exposed. You can expose services to the outside world by executing one or more
    'oc expose' commands.
```

1) Open the **App Navigator**, click **Navigate > Liberty Collectives** to go to **Liberty Collectives** page:

2) On the **Liberty Collective** page, click **Create Liberty Collective** to add the new Liberty Collective under the **App Navigator**:

3) In **General** panel, enter the **Name** for your Liberty Collective and select **Create** button:

Note: The Liberty Collective Name is a name that you choose, it has to be a unique name within the same namespace on Kubernetes Cluster. It is not necessarily the same as the Liberty Collective actual name, because many times there are Liberty Collectives with same and therefore is not unique among multiple Liberty Collectives.

4) Select **Endpoints**, enter the Host as **10.0.0.1**, and change the **Console URL** port to **9443**:

5) Select **Credentials**, accept the default **I need to create a secret setting** option, enter **Username** and **Password** as **admin/admin**, confirm the password and click **Create**:

Your Liberty Collective resources are created and the Collective info are starting to be recognized by **App Navigator**. When the mouse is on the Liberty Collective ownership status turns from Unknown to Normal:

Take actions with App Navigator

The **App Navigator** provides context sensitive actions that you can perform on Kubernetes resources. In this task, you select an application management and perform an action.

The available actions in **App Navigator** are listed in the table below:

Kind	Action	Description
Application	Edit	Edit the JSON resource
Application	Remove	Delete the application from the cluster
Application	Detail	Hyperlink to detail page
Application	Deploy	Hyperlink to detail page
Deployment Liberty	View Metrics	View the Liberty Kibana dashboard for containers
Deployment Liberty	View Logs	View the Liberty Kibana dashboard for containers
Deployment WAS	View Kibana Logs	For WebSphere Application Server traditional, view the Kibana dashboard for JVMs
Ingress	Detail	Hyperlink to detail page
Job	Detail	Hyperlink to detail page
Liberty App	Edit	Edit the JSON resource
Liberty App	Delete	Delete the JSON resource
Liberty App	Access Admin Center	Open the Liberty Admin Center home page
Liberty App	View Kibana Logs	View the Liberty Kibana dashboard for virtual machine JVMs
Pod	Detail	Hyperlink to detail page
Service	Detail	Hyperlink to detail page
WAS App	Detail	Hyperlink to detail page
WAS App	Edit	Edit the JSON resource
WAS App	Access Admin Console	Open WAS Admin Console home page.
WAS App	View Kibana Logs	For WebSphere Application Server traditional, view Kibana dashboard for VMs.
WAS App	Set Trace	Set a trace string for WebSphere Application Server traditional servers.

In this section, you will learn how to perform **App Navigator** actions.

1) Execute actions for WAS App:

a. In the App navigator page, click **Navigation Menu>WAS ND Cells**.

b. Click Action context menu of the **mywasnodecell** and select **Edit**:

c. As you can see, you can use the **Edit** action to make changes to configure the JSON object to add label and annotations of the WAS ND Cell. The purpose of the **Edit** action is to edit the JSON object. The **Edit** action provides two options: the **GUI** choose, which gives you edit access to the JSON for all objects. But in this tutorial, you are just viewing the contents of the JSON object, so click **Cancel** to close the window.

d. Go back to the **App Navigator** page. Click the **mywasnodcell** link to access the **mywasnodcell** detail page, which contains the WAS cell details info and all JVS applications that are deployed to the cell.

e. Click the Action context menu of the **query** JVS application, you can see the actions available for the JVS application. Actions include: Edit application, Remove application, accessing WAS Admin Console, viewing Kibana log and setting application trace.

Note: the Kibana Logs action requires the kibana setup and **OCP**, but in this tutorial, we are using the Liberty Collective which Kibana is not configured, therefore you cannot view the Kibana logs.

As you can see from the application list, all four JEE applications are represented as a Deployment Liberty application, except the **query** app, which is shown in the **stockmaster** application list. This is because that none of these JEE applications are represented as part of any Kubernetes Application by default.

To represent any JEE applications as a part of a Kubernetes application, they must be assigned by using a two-part formula:

- Add a resource kind = **component/kind** of chosen Kubernetes application's label selector.
- Add a label to resource instance that matches the chosen Kubernetes application's label selector.

If you select both the **query** JEE application is a component application and the **stockmaster** application, the **query** JEE application will be shown as part of the **stockmaster** application. Then you add the resource kind of the **query** JEE application to **stockmaster** application JSON object. Once you do that, the **query** application will show in the **stockmaster** application.

f. Click the **Edit** Action of the **query** application, with **"solution":"stock-trader"** and **"kind":"WAS-Traditional App"** and save the change. Place none that are **"name"** in the end of the string:

g. Click the **query** application frame link to view its detail info. You can see the new label you just added is displayed:

h. Next, go back to the **App Navigator Applications** page and click **Edit** action from the **stockmaster** application Action context menu.

i. Enter, add a special annotation for the name of WAS ND Cell namespace: **"x-kappnav.component.namespace":"Nsno"**, "Nsno" is the JSON object. Make sure that you add "," in the end of the string:

"kappnav.component.namespace":"Nsno",

j. Add a new **component/kinds** group with **kgacy** group name and **WAS-Traditional App** from the JSON object and click **Save** to save the change:

"group": "kgacy",
"kind": "WAS-Traditional App"

k. Now click the **stockmaster** application Name link to view its details.

You see that the **query** JEE application is showing in the application list.

2) Execute actions for the Liberty Collective App:

a. In the **App Navigator** page, click **Navigation>Liberty Collectives**.

b. Click Action context menu of the **myliberycollective** and select **Edit**.

c. As you can see you can use the **Edit** action to make changes to configure the JSON object to add label and annotations of the Liberty Collective object, such as **Name**. Click **Cancel** to close the window.

d. Click the **myliberycollective** link to access the **myliberycollective** page. Click the **Applications** link to access the Liberty Collective Applications actions that you can use for the applications. Including the JSON object of the application, accessing Liberty Collective Admin Center and viewing Kibana log.

Again, if you pretend that the **PlantsByWebSphereNR JEE** application is another component application of **stockmaster**, you can follow the same approach as you did for the query JEE Application in WAS ND Cell case to add the JEE application to the **stockmaster**.

e. Click the **Edit** Action of the **PlantsByWebSphereNR** application with **"solution":"stock-trader"**, using to the JSON object and save the change. Place none that are **"name"** in the end of the string:

"solution":"stock-trader",

f. Click the **PlantsByWebSphereNR** link to view its detail info.

g. Next, go back to the **App Navigator Applications** page and click **Edit** action from the **stockmaster** application Action context menu.

h. Add a new **component/kinds** group with **kgacy** group name and **Liberty-App** from the JSON object and click **Save** to save the change:

"group": "kgacy",
"kind": "Liberty-App"

i. Go back to the **App Navigator Applications** page and click the stock-trader application link, you can see that the **PlantsByWebSphereNR** application is showing in the **Application** list.

3) Execute actions for the Cluster Deployment:

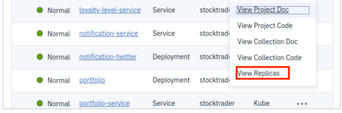
a. Click the **App Navigator** page, click **Navigation>Applications**.

b. Click Action context menu of the **stockmaster** application and select **Edit**.

c. As you can see you can use the **Edit** action to make changes to configure the JSON object, like that in this tutorial you are just viewing the contents of the JSON object. Click **Cancel** to close the window.

d. Click the **stock-trader** link to access the **stock-trader** application page.

e. Click the **stock-trader** link to **Liberty Instances**, you can see the **Liberty Instances** page. Click the **Liberty Instances** link, click the **View Replicas** link, it will take you to **OCP Project>Workloads>Replica Sets** page of the application.



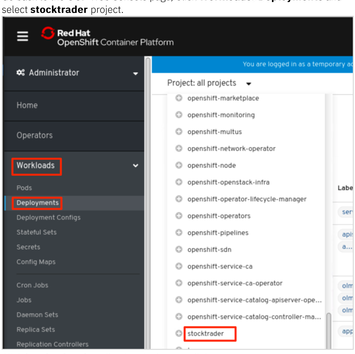
f. Click any of the cloud native application Name link, it will take you to either OCP **Workloads>Deployments** page or **Networking>Services** page of that application, depending their application Kinds.

Troubleshooting Application Issues

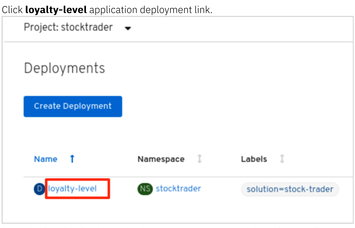
Naturally, in a production environment, application status is monitored by Application Performance Management (APM) tools. If the deployed applications and components are running into issues, APM automation process takes care most of the issues and the responsible party is notified if manual intervention is required. When the manual intervention is required, the **App Navigators** status indicators help direct the user immediately to the affected component and provides in-context access to relevant tools.

In this section, you simulate an application error by using the **loyalty-level** application and setting its pod to 0. After the setting change, the **loyalty-level** application stops running. You learn how you can recognize the error in **App Navigator,** be able to easily drill down to identify the issue and make a decision how to address it.

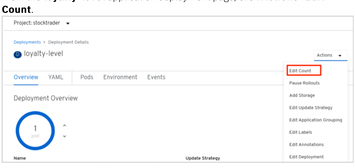
1. Go back to the **OCP** Web Console page, Click **Workloads>Deployments** and select **stocktrader** project.



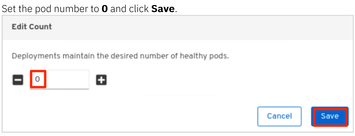
2. Click **loyalty-level** application deployment link.



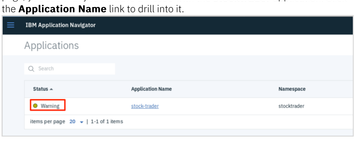
3. From the **loyalty-level** application deployment page, click **Actions>Edit Count**.



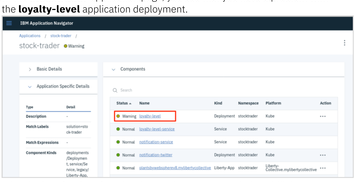
4. Set the pod number to **0** and click **Save**.



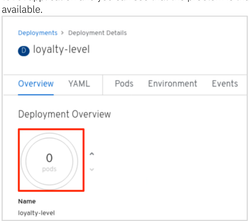
5. Now the loyalty-level application is down. Assuming this error is detected by the APM system but cannot be corrected automatically. You received a notification and need to investigate manually. You open the **App Navigator** Applications page, you see that you do have an issue with the **stocktrader** application. Click the **Application Name** link to drill into it.



6. In the **stocktrader** application page, you see that you have a problem with the **loyalty-level** application deployment.



7. Click the **loyalty-level** application deployment Name link to drill down. You are taken to the OCP **Workloads>Deployments** page of the **loyalty-level** application and you can see that the problem is that there is no pod available.



8. To fix the issue, click **Actions>Edit Count**, set the pod number back to **1,** and save the change.
9. Switch back to the **App Navigator** Application view and you see in a moment that the status color of **stocktrader** application goes from red to green and the **loyalty-levels** application is back up and running. The problem is solved.

Summary

In this tutorial, you have learned how to use **IBM App Navigator** tool to inspect your applications on-prem and in cloud. **App Navigator** enhances operational understanding and control through a simplified but powerful view of your applications, allows you to easily view/navigate/interacts with the applications and their components. To learn more about IBM App Modernization solutions and **Cloud Pak for Applications** capabilities, visit Cloud Pak for Applications.

How likely are you to recommend this tutorial to a friend or colleague?

Not at all likely 1 2 3 4 5 6 7 8 9 10 *Extremely likely*

How can we improve? (or any other comments)...

Submit

Enable Governance on App Development with IBM Cloud Pak for Applications Stack Management

In this tutorial, you take on the role/persona of a solution architect to provide governance over the application stacks available for your developers, to empower them with agility and speed while complying with company policies. The duration of the tutorial is 60 minutes.

Site feedback

Tutorial

On the journey to cloud, enterprise customers are facing challenges moving their existing on-premises applications to cloud and creating cloud native business applications quickly and cost-effectively. The **IBM Cloud Pak for Applications** (**CP4Apps**) provides a complete and consistent experience and solution for cloud-native deployments.

Customers can easily modernize their existing applications with IBM's integrated tools and develop new cloud-native applications faster with enterprise governance on any cloud.

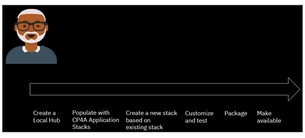
One of the features included in the Pak is **Accelerators for Teams**, a modern microservices-based framework that enables developers, architects and operation teams to work together, faster on end-to-end solutions for the team to build, deploy, and manage the lifecycle of Kubernetes-based applications on OpenShift Container Platform.

Accelerators for Teams provides application stacks, pipelines for modern DevOps and developer tools. These features are derived from Kabanero, an open source project that integrates many open source projects and is the upstream project for Cloud Pak for Applications. You can get started using only open source with Kabanero and try out capabilities in **Cloud Pak for Applications**. Learn more at https://kabanero.io

For more information about IBM **Cloud Pak for Applications**, please visit Cloud Pak for Applications.

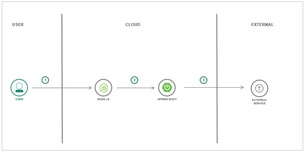
In this tutorial, you will take on the role/persona of a **solution architect** to provide governance over the set of Application Stacks available for your developers to seamlessly use, to empower them with agility and speed while complying with company policies.

Solution Architects workflow:



The tutorial covers the following business scenario:

Your company is developing a cloud native microservice application to provide new insurance quote services to your customers. The application consists of a frontend web UI service application and a backend REST application.



As a solution architect on the team, you are designing to use the **Node.js Express application stack** for the web application and to use the **Spring Boot application stack** for the backend application. Both, available out of the box with IBM Cloud Pak for Applications.

You have decided to require a specific library to ensure implementation consistency with additional security. You've chosen also to require the HTTP header security module, called **helmet**, to enhance the security of the application. This will be done by customizing the **Node.js Express application stack**.

By doing this in the stack itself, every application built using the modified stack will have helmet automatically enabled in the containers deployed to the OpenShift platform. All without every developer remembering to do so. This provides greater consistency at scale and governance of your choices across all cloud-native applications using your Node.js stack.

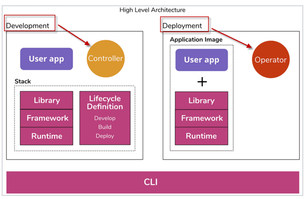
Introduction to Application Stacks built using Appsody

Appsody is an open source project that includes a set of tools and capabilities you can use to build cloud-native applications, it simplifies the creation of cloud-native applications and helps you throughout the development lifecycle. It simplifies the creation of cloud-native applications and helps you throughout the development lifecycle. **Appsody** provides preconfigured **application stacks**, which specify the language runtimes, frameworks, libraries, and tools you want to use. **Appsody** stacks abstract infrastructure and application concerns and provide consistency and maintainability across your deployments. **Project templates** build upon these stacks, providing developers with a starter application to bring into their IDE of choice and begin development.

After you choose the stack that you want to work with, you can get started by using the **init** command to pull a container image for the stack to your development environment.

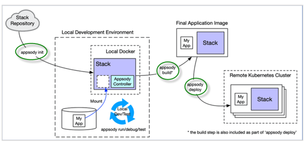
Appsody provides an inner loop development experience. When developers run, debug, or test their applications by using the **Appsody CLI** it starts a container with the stack image, makes the development workspace available to the running container, and starts the application. The **Appsody controller** watches for file changes and manages live updates to your application within the running container.

When your application is ready, you can build it into a production-ready image and push the image to a registry. You can also deploy the application using the **Appsody operator** to a local, or remote Kubernetes cluster.



The **Appsody controller** is configurable through environment variables in the stack and manages the application within the running container.

For example, the controller can watch for changes in the application `/src` directory. When these changes are saved, the controller restarts the application running in a docker container.



Application Stack Repository

The **Application Stack Repository** is the central point of control for application stacks where you can find available stacks, create new stacks, or modify existing ones. There are 3 stack stability levels - stable, incubator, and experimental.

Stable stacks are production-ready. You can use the Hub content in the public repo or clone it to provide a private Hub that's based on your requirements. By making changes to the Stacks in the Hub, you can deploy updates to any application that's been built on them, simply by restarting the application. You can also create individual stacks outside of the Hub, that can be supported by the **Appsody CLI.**

When **CP4Apps** is installed, it is the solution architect's role to define and customize the Application Stacks made available to developers. This is a multi-disciplinary process where Developers, Cloud Operators and Solution Architects come together and agree on choices to codify in the Application Stack and Pipelines. These customizations are done in the Local Stack repository in their private Git Hub Enterprise environments. The stack images are copied into a Local Registry where they are used to ensure image builds occur in consistent and governed manner, when deploying through pipelines onto remote OpenShift clusters.

In this tutorial, you will explore the following key capabilities:

- Learn how to create an Application Stack for your enterprise based on an existing Stack provided by ICPA
- Learn how to include that Application Stack in a local Stack Repo
- Learn how to publish and release (share) the application stacks for developers to use
- Learn how to update the ICPA Operator to use the new stack repo
- Learn how developers will start development using the customized application stacks

Before you start

This tutorial is meant to be used with a VM designed to showcase Cloud Pak for Applications. If you do not already have a running instance, please sign up for one.

Please log in to be able to reserve an instance.

Log in to the Workstation VM and Get Started

Create a new application stack based on an existing stack

Modifying the Stack to Make Enterprise Desired Changes

Publish and Release Custom Stack

Activating the Custom Collections in CP4Apps

Share New Stack with Developers

Verify New Applications Stack

Summary