# Final Project: Dealer Evaluation

**Estimated Time Needed:** 45 minutes

## Learning objectives

After completing this project, you will be able to:

- Clone a repository from Github
- Run Maven build
- Host a Spring Boot Java web application
- Write User Stories
- Containerize your web application
- Deploy your container on IBM Cloud
- Enhance your code
- Deploy changes on IBM Cloud

## About the course project

Welcome to the final project for your first **Cloud Native, Microservices, Containers, DevOps, and Agile** course. You will apply the knowledge and skills learned in this course to a simulated scenario.

In this project, you are given a **Dealer Evaluation** application written in Java. You need to plan the modernization of this application and implement those Stories. You are not being tested on Java programming skills in this project but will be required to complete each task, with hints and guidance provided.

The 6 tasks in this hands-on project correspond to the activities performed by someone working in a Cloud Native environment, working with Microservices and CI/CD, and practicing Scrum methodology.

You are currently viewing this lab in a Cloud-based Integrated Development Environment (Cloud IDE). It is a fully online integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

# Working with files in Cloud IDE

If you are new to Cloud IDE, this section will show you how to create and edit files which are part of your project in Cloud IDE.

To view your files and directories inside Cloud IDE, click this files icon to reveal it.

If you have cloned (using `git clone` command) boilerplate/starting code, then it will look like the following:

Otherwise, a blank project looks like this:

## Create a new file

You can right-click and select the **New File** option to create a file in your project.

You can also choose `File -> New File` to do the same.

It will then prompt you to enter the name of this new file. In the example below, we are creating `sample.html`.

Clicking the file name `sample.html` in the directory structure will open the file on the right pane. You can create all different types of files; for example, `FILE_NAME.js` for JavaScript files.

In the example, we just pasted some basic html code and then saved the file.

Save it by:

- Using the menu
- Press `command + S` on Mac or `CTRL + S` on Windows
- Or it can Autosave it for you too

# Verify the environment and command-line tools

1. Open a terminal window by using the menu in the editor: `Terminal > New Terminal`.

   **Note: If the terminal is already opened, please skip this step.**

2. Verify that `docker` CLI is installed.

   ```
   docker --version
   ```

You should see the following output, although the version may be different:

3. Verify that `ibmcloud` CLI is installed.

```
ibmcloud version
```

You should see the following output, although the version may be different:

4. Verify that `java` is installed.

```
java --version
```

You should see the following output, although the version may be different:

5. Verify that Maven is installed.

```
mvn --version
```

You should see the following output, although the version may be different:

# Start Code Engine

1. On the menu in your lab environment, click the `Cloud` dropdown menu and select `Code Engine`. The code engine setup panel appears. Click `Create Project` to begin.

Another way to get to it is by clicking the following action:

Create Code Engine Project in IDE

2. The code engine environment takes a while to prepare. You will see the progress status indicated in the setup panel.

3. Once the code engine setup is complete, you can see that it is active. Click `Code Engine CLI` to begin the pre-configured CLI in the terminal, as shown below.

4. You will observe that the pre-configured CLI startup and the home directory are set to the current directory. As a part of the pre-configuration, the project and Kubeconfig have been set up. The details are shown on the terminal as follows.

# Task 1: Create application

1. Open a terminal window by using the menu in the editor: **Terminal > New Terminal**.

2. If you are not currently in the project folder, copy and paste the following code to change to your project folder.

   ```
   cd /home/project
   ```

3. Run the following command to clone the Git repository that contains the starter code needed for this project if the Git repository doesn't already exist.

   ```
   git clone https://github.com/ibm-developer-skills-network/lmmoq-dealerEvaluation.git dealerEvaluation
   ```

4. Change to the directory **dealerEvaluation** to start working on the lab.

   ```
   cd dealerEvaluation
   ```

5. List the contents of this directory to see the artifacts for this lab.

```
ls
```

6. Run the following command on the terminal to host your web page.

```
mvn spring-boot:run
```

7. To test your application in the browser, run the application first.

Launch Application

8. It will look like this:

*Assessment*: Save the URL from Cloud IDE browser. You will need to submit this for evaluation.

*Assessment*: Note down the price of Headphones from Binglee. You will need this for evaluation.

9. In your terminal, press `CTRL+C` to stop your web server.

# Task 2: Plan application modernization

One of the key improvements needed for your Dealer Evaluation application is to modernize it and use a cloud environment to host it.

This is a major piece of work, so it should be planned as an Epic and several stories.

### Epic

An Epic is a large and high-level item of work that provides a way to capture and communicate the broader goal of a project. They are too complex to be estimated accurately at the outset, so they are often broken down into smaller, more manageable user stories or features.

### Story

A story (or user story) is a short, simple description of a feature or requirement from the perspective of the user. It helps the team understand what needs to be built and why.

## Create Stories

Let's first look at an example of a complete user story for "Containerizing the application."

**Example: Containerizing the application**

**Task description:** Have the ability to package the application and be deployable in Docker

**User story:**
**As a** Developer, I want to containerize the application using Docker so that it can be easily deployed, scaled, and managed in different environments.

**So that** the application runs consistently across different environments (development, testing, production). Dependencies are managed efficiently without conflicts. And deployment and scaling become easier.

## User stories: Assessment

As part of the assessment, create two user stories for the following tasks, based on the format provided in the example above.

**User story 1: Deploying on the IBM Cloud**

**Description:** Instead of hosting on self-managed or in-house virtual machines, deploy the application on the IBM Cloud using the IBM Cloud Code Engine.

**User story 2: Read products and dealers from JSON**

**Description:** The current code has hard-coded data, which is not an ideal practice. To move away from this, you need to write a story.

# Task 3: Containerize the application

The first step toward modernizing your application is to containerize it using Docker.

### Docker

Docker is an open-source platform and toolset that allows you to automate the deployment, scaling, and management of applications using containerization. Containers are lightweight, isolated, and portable environments that package applications and their dependencies, enabling them to run consistently across different computing environments.

## Package application

You first need to create a package to be used in Docker.

```
mvn package
```

## Create Dockerfile

Open **Dockerfile** in IDE

Use the following code to put in it:

```
# Use OpenJDK 21 as the base image
```

```
FROM openjdk:21-jdk-slim
COPY target/comparison-0.0.1-SNAPSHOT.jar /usr/app/comparison.jar
CMD ["java", "-jar", "/usr/app/comparison.jar"]
```

## Build Docker image

Use the following commands to create Docker image:

```
cd /home/project/dealerEvaluation
docker build -t comparison .
```

## Run Docker image in Cloud IDE

```
docker run -it -d -p 8080:8080 comparison
```

## Verify Docker image

List Docker images and verify you see `comparison`.

```
docker images
```

**Assessment**: Note down the output of `docker images` command. You will need this for assessment and grading later.

## Verify Docker image is running

```
docker ps -a
```

**Assessment**: Note down the `NAMES` value from the output. You will need this for assessment and grading later.

## Verify in browser

Launch Application

### Stop and clean Docker container

Finally, clean up running containers.

```
docker ps -aq | xargs docker stop | xargs docker rm
```

**Assessment**: Note down the output from the command to stop and cleanup containers. You will need this for assessment and grading later.

# Task 4: Deploy on IBM Cloud

You have successfully run the Dealer Evaluation app from Cloud IDE.

As part of this learning platform, you are provided with basic infrastructure on IBM Cloud. You don't need an external sign up, and all setup has been done for you already.

### Build and tag Docker image

The Docker build command builds Docker images from a Dockerfile.

```
cd /home/project/dealerEvaluation
docker build . -t us.icr.io/${SN_ICR_NAMESPACE}/comparison
```

## Push the image to IBM Cloud

To upload an image to a registry, use `docker push`.

Let's push the image to IBM Cloud.

```
docker push us.icr.io/${SN_ICR_NAMESPACE}/comparison
```

## Deploy the image to IBM Cloud

Deploy the image on IBM Cloud Code Engine.

```
ibmcloud ce application create --name comparison --image us.icr.io/${SN_ICR_NAMESPACE}/comparison --registry-secret icr-secret --port 8080
```

Check the status

```
ibmcloud ce application get --name comparison
```

To just get the URL to your application hosted in IBM Cloud

```
ibmcloud ce application get --name comparison --output url
```

Which looks like: `https://comparison.somerandomalphanumeric.us-south.codeengine.appdomain.cloud`

Use the above URL and add `/price/index.html`

**Assessment**: Note down the url pointing to your Dealer Evaluation application hosted on IBM Cloud. You will need this for assessment and grading later.

# Task 5: Read products and dealers from JSON

In this task, you will work on the story that you specified earlier. The hard-coded data about products and dealers will be replaced by reading dynamically from JSON. This move will help you avoid mixing data with code and gives you greater flexibility in the future to decouple the two.

This move will also help you go toward reading data from the database. But this is out of scope for this project.

### Make changes in Dealer Service

The dealer products and prices are hard coded in the following file:
`/home/project/dealerEvaluation/src/main/java/com/comparison/price/service/DealerService.java`

Open **DealerService.java** in IDE

In the function `getData`, replace the code with following code:

```
try {
    String json = JsonReader.readJsonFromClasspath("json/dealers.json");
    return objectMapper.readValue(json, DealersDTO.class);
} catch (Exception e) {
    return null;
}
```

## Make changes in product service

The products are hard coded in the following file: `/home/project/dealerEvaluation/src/main/java/com/comparison/price/service/ProductService.java`

Open **ProductService.java** in IDE

In the function `getData`, replace the code with the following code:

```
try {
    String productsJson = JsonReader.readJsonFromClasspath("json/products.json");
    return objectMapper.readValue(productsJson, ProductsDTO.class);
} catch (Exception e) {
    return null;
}
```

# Task 6: Deploy changes on IBM Cloud

You have previously packaged and deployed the container image to IBM Cloud. Do this again for this step so your changes are reflected.

```
cd /home/project/dealerEvaluation
mvn package
docker build . -t us.icr.io/${SN_ICR_NAMESPACE}/comparison
```

```
docker push us.icr.io/${SN_ICR_NAMESPACE}/comparison
```

This time, we are not creating but updating the image. So, we will call the command with `update` instead of `create`.

```
ibmcloud ce application update --name comparison --image us.icr.io/${SN_ICR_NAMESPACE}/comparison --registry-secret icr-secret --port 8080
```

You should see the output for Headphones from All Dealers like below:

**Assessment**: Note down the price of Headphones from Binglee, after data is read from JSON. You will need this for assessment and grading later. The price will be different from the example above.

# Summary

## Congratulations

You have now completed all of the enhancements needed for Dealer Evaluation.

In this lab, you have gained an understanding of how a change is done in an application. First, start with breaking down the tasks into manageable stories, and then work on them and iterate to a better version of your software. You have also seen how a locally hosted Java web application can be converted to a Docker image, tested, and run locally. Finally, you hosted it in the cloud.

Many developers follow the same process to make their applications cloud native.

## Author(s)

Muhammad Yahya