**Lab Guide**

IBM Business Automation Manager Open Edition

Nigel Crowther – ncrowther@uk.ibm.com

Hands-on Lab

Watsonx Orchestrate Integrating with Open-Source Solutions

V1.0

01-Jan-2024



NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

**TRADEMARKS**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

© Copyright International Business Machines Corporation 2020.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

**Table of Contents**

[0. Introduction 4](#_Toc155189514)

[0.1 Pre-requisites 4](#_Toc155189515)

[1. Build a decision service to calculate speeding fines 5](#_Toc155189516)

[1.1 Before you start 6](#_Toc155189517)

[1.2 Run the Decision Service 9](#_Toc155189518)

[1.3 Expose the decision service 10](#_Toc155189519)

[1.4 Test the Decision Service 11](#_Toc155189520)

[2. Generate an OpenAPI specification 13](#_Toc155189521)

[3. Import the OpenAPI into watsonx Orchestrate 17](#_Toc155189522)

[4. Persist Decisions 20](#_Toc155189523)

[5. Create a Composite Flow using Decisions, Cloudant and React 24](#_Toc155189524)

[6. Advanced Lab – Create your own Cloudant instance for Watsonx Orchestrate 31](#_Toc155189525)

[6.1 Local host Prerequisites 31](#_Toc155189526)

[6.2 Run on local host 31](#_Toc155189527)

[6.3 Code Engine Hosting Prerequisites 32](#_Toc155189528)

[6.4 Deployment to code engine on IBM Cloud 33](#_Toc155189529)

[7. Conclusion 35](#_Toc155189530)

# Introduction

In this Lab you connect watsonX Orchestrate using three open-source solutions:

* **BAMOE** – to run a decision service.

See <https://www.ibm.com/products/business-automation-manager-open-editions>

* **Cloudant** – to store data.

See https://www.ibm.com/products/cloudant

* **React** –to create reports.

See <https://react.dev/>

The Lab is structured into the following sections.

* Build a speeding ticket decision service.
* Build a skill to save the ticket to a database.
* Build a skill flow to combine the above and then invoke a Ticketing user interface.

## Pre-requisites

To perform this Lab, you need:

* A GitHub Account. See https://github.com/
* An IBM watsonx Orchestrate Standard account with Builder role.
* Techzone BAMOE VM : <https://techzone.ibm.com/my/reservations/create/64a7121eb9776b0016a10cb0>

Alternatively, you can build your own environment, in which case you need:

* VSC and the IBM BAMOE plugin
* Git Bash
* Latest Java JDK

# Build a decision service to calculate speeding fines

In this task, you ...

* Import a speeding ticket decision service.
* Run the service locally.
* Expose the decision service to the outside world

## Before you start

In this exercise, you will build and test the decision service using Visual Studio Code (VSC).

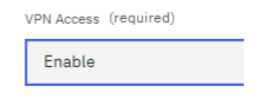
Find the following BAMOE Techzone image (or later version if available):

<https://techzone.ibm.com/collection/ibm-business-automation-manager-open-editions-environment/environments>

A screenshot of a computer

Description automatically generated

**IMPORTANT!** When reserving, ensure IP address is enabled:



Once the VM is reserved, login with Remote Desktop (RDP):

**User:** .\techzone

**Password:** IBMDem0s!

For further details on setting up the VM, see the getting started guide here: <https://ibm.ent.box.com/folder/170819029423>

1. Open Git Bash by clicking the Git Bash  icon on the Windows **Taskbar** at the bottom of the screen.
2. Within the Git Bash shell, clone the rule project using the command:

git clone <https://github.com/ncrowther/bamoe2wxo>

1. Hit return. You should see the git repository cloned into the local drive:

Text

Description automatically generated

Now open the decision project with VSC:

cd bamoe2wxo/

code SpeedingDecisionService/

1. VSC will open:

A screenshot of a computer

Description automatically generated

1. On the left-hand side of Visual Studio Code in file explorer locate *src\main\resources\TrafficViolation.dmn*A screenshot of a computer

   Description automatically generated
2. Double-click the DMN file to view it:

A screenshot of a diagram

Description automatically generated

1. On the far right, select the edit icon . In the *DMN Model Name,* Replace [YOUR\_INITIALS] with your initials. For example, if your initials are XXX, the *DMN Model name* would be *XXXTrafficViolation*:

A screenshot of a computer

Description automatically generated

1. Select the *Fine* decision and edit by selecting . You should see the decision table below:

A screenshot of a computer

Description automatically generated

This decision table will be used later in the lab to calculate speeding fines depending on severity.

1. Press Save (Ctl S).

## Run the Decision Service

In this step we are going to run our decision service in development mode using Quarkus. For details on Quarkus, visit <https://quarkus.io/>

**Important**: if you are not using VSC 1.85 or later, you must update. Select *Help->Restart to update.*

1. Within the VSC editor, click on the **Terminal** menu at the top of the screen:



1. In the drop down click on **New Terminal**.

A screenshot of a computer

Description automatically generated with medium confidence

1. In this terminal window, verify you are in the *SpeedingDecisionService* folder and then run Mavern build:

mvn quarkus:dev

1. Once built, our decision model is deployed to Quarkus and ready to run in development mode:

A computer screen with text

Description automatically generated

## Expose the decision service

1. Within the VSC editor, click the PORTS tab:

A screen shot of a computer

Description automatically generated

1. Select *Forward a Port*
2. Enter kogito service port *8123* as the Port and hit return. You will see the following:

A close up of a message

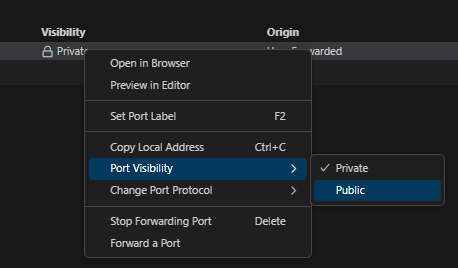
Description automatically generated

1. Select *Allow* and enter your GitHub credentials.
2. You can find the port in the Quarkus terminal output. After a few seconds, you should see this:

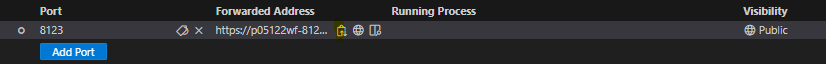
A black background with white text

Description automatically generated

1. Right-click *Private* in the *Visibility* column and change from *Private* to *Public*:



1. Press *Continue* to accept the warning message. Select the *copy local address* button  to store the exposed URL into your clipboard.



## Test the Decision Service

Let’s look at the API generated by Quarkus.

1. Open the Chrome browser within the VM.
2. Paste the following URL into the browser address and press **Enter**.

[[REMOTE\_URL]/q/swagger-ui](localhost:8123/q/swagger-ui)

Where [REMOTE\_URL] is the url you copied in the last step of the previous section. Accept the warning by pressing *Continue*. You should see this:  
A screenshot of a computer

Description automatically generated

You are looking at the decision service endpoints exposed to the internet. If you followed the instructions in the previous section, your endpoints will have your initials prepended. We can see three endpoints; The only endpoint we are interested in is the first **POST /TrafficViolation** endpoint. The second one includes additional trace information which we won’t be using.

1. Click on the **POST /TrafficViolation** endpoint to expand it, then click the **Try it out** button on the right-side.  
   A white rectangular object with a black border

   Description automatically generated
2. Enter the following data into the request body:

{

"Violation": {

"Date": "2023-12-21",

"SpeedLimit": 20,

"ActualSpeed": 40

},

"DriverId": "D45454"

}

1. Press *Execute*. You should see the following in the server response:

**Response Body**

**{**

**"Violation": {**

**"SpeedLimit": 20,**

**"ActualSpeed": 40,**

**"Date": "2023-12-21"**

**},**

**"DriverId": "D45454",**

**"Fine": {**

**"Points": 3,**

**"Amount": 500**

**}**

1. Verify the points are 3 and the fine is 500.
2. Congratulations, you have built and deployed the Speeding Ticket decision service on your local machine and exposed it to the outside world. The next step is to generate an OpenAPI specification to execute the service from watsonx Orchestrate.

# Generate an OpenAPI specification

In this section you generate an OpenAPI specification for watsonx Orchestrate.

**Important!** To generate this specification, the decision service in the previous section must still be running.

1. Reopen Git Bash shell
2. Within the Git Bash shell, open the *OpenApiGenerator* folder in VSC:

code OpenApiGenerator/

1. The VSC editor appears.

A screenshot of a computer program

Description automatically generated

1. Expand the *data* folder and edit *config.json*. You should see this:

A computer screen shot of a code

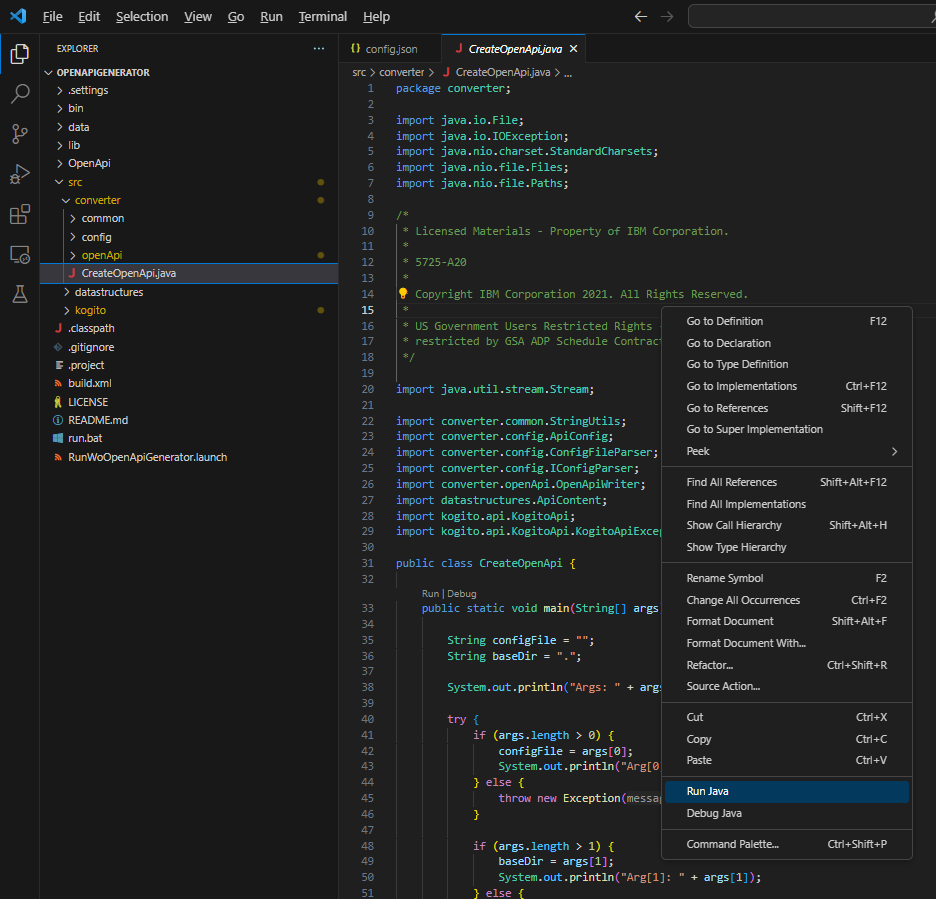
Description automatically generated

1. Change the name of *decisionId* by replacing [YOUR\_INITIALS] with your initials. This should be the same name as in section 7. For example, if your initials are XXX, the DMN Model name would be XXXTrafficViolation.
2. Change the internal URL to the local URL of your running Kogito service. If you are using the VM this should be correctly set.
3. Change the external URL to the one defined on page 10.
4. Save *config.json*
5. Open *CreateOpenApi.java* in the *src/converter* folder:

A screenshot of a computer

Description automatically generated

1. Right-click anywhere in the code and select *Run Java*:



1. An open API file is generated:

A screen shot of a computer

Description automatically generated

1. Within VSC, open

.\generated\[YOUR\_INITIALS]TrafficViolation.json

and paste its contents into <https://editor.swagger.io>

1. You should see the same decision service we inspected in section 1.4

A screenshot of a computer

Description automatically generated

1. There are differences compared with the original:

* Your initials have been appended.
* *x-ibm* annotations have been added.
* The data model has been pulled into the specification.
* Only one endpoint is generated.

The OpenAPI is now ready to be imported into watsonx Orchestrate!

# Import the OpenAPI into watsonx Orchestrate

1. Login to **IBM watsonx Orchestrate**
2. Under build skills, select *add skills*
3. Select *From files*
4. Drag the OpenAPi file into the drop area:

*[GIT]\bamoe2wxo\OpenApiGenerator\generated\[YOUR\_INITIALS]TrafficViolation.json*

A screenshot of a computer

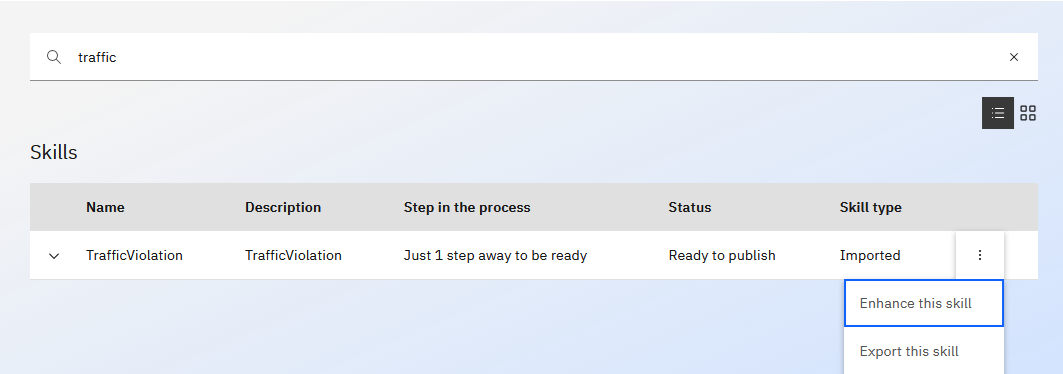
Description automatically generated

1. Click Next. Select the *[YOUR\_INITIALS]TrafficViolation* skill and press *Add*.

A screenshot of a computer

Description automatically generated

1. Find the [YOUR\_INITIALS]*TrafficViolation* skill.
2. Right click the button with three vertical dots, and select *Enhance this skill*:



1. Publish the skill:

A close up of a logo

Description automatically generated

1. Click on *Skills Catalog*
2. Find the [YOUR\_INITIALS]*TrafficViolation* skill (type in the full name to find it). Select it and press *Add skill*

A white background with black text

Description automatically generated

1. Click on Connect app  and enter any username and password (credentials are placebos).
2. Go to *Chat* and run the skill by typing *[YOUR\_INITIALS]trafficviolation*
3. The Traffic Violation data form appears. Enter the following:

A screenshot of a computer

Description automatically generated

1. Hit *Apply*. This invokes the decision service via your local PC using the exposed port.
2. The response should be:

**A screenshot of a computer

Description automatically generated**

Congratulations! You have run your first BAMOE Decision Service from watsonx Orchestrate.

# Persist Decisions

In this section we call a database to store speeding tickets. We use a public instance. If you are interested in creating your own private instance, follow section 5.

1. Reopen Git Bash shell
2. Within the Git Bash shell, open the *cloudantFacade* in VSC

code cloudantFacade/dataservice

1. The VSC editor should appear with the *cloudantFacade/Dataservice* folder.

A screenshot of a computer

Description automatically generated

1. Expand the *openapi* folder and edit *dataApi.yaml*
2. Perform a global search and replace of the text [YOUR\_INITIALS], replacing with your initials. To do this, select Ctrl-R:

A screenshot of a computer

Description automatically generated

1. Save *dataApi.yaml*
2. Paste the contents of *dataApi.yaml* into <https://editor.swagger.io>
3. You should see the following, with your initials instead of XXX:

A screenshot of a computer

Description automatically generated

1. Now we will import this API into watsonx Orchestrate
2. Login to IBM watsonx Orchestrate
3. Under *Build* *skills*, select *add skills* from *Files* then drag and drop the *dataApi.yaml* API spec:

A screenshot of a computer

Description automatically generated

1. Click Next. Select the first twoskills and press *Add*.

A screenshot of a computer

Description automatically generated

1. Publish both skills and add them to your catalog, as described in section 3
2. You should now have both skills published under the *cloudantWxoFacade* and added to your catalog.
3. Run the skill *write a doc to cloudant.* Enter a *driverId* starting with D and then six random numbers:

A screenshot of a computer

Description automatically generated

1. Hit Apply.
2. Now run the *get docs from cloudant* skill. You should see all speeding fines retrieved from the database.
3. Find your entry by using the search  button:

A screenshot of a computer

Description automatically generated

Congratulations! You have run your first Cloudant skill by entering data and retrieving it from watsonx Orchestrate.

# Create a Composite Flow using Decisions, Cloudant and React

We will now combine the Decision skill and the Cloudant skill with a React UI.

1. Under *Build* *Skills*, select the down arrow on the *Add Skills* button and choose *Create a skill flow*.

A blue and yellow box with text

Description automatically generated

1. You should see the following:

A screenshot of a computer

Description automatically generated

1. Select the pen  and name the flow *[YOUR\_INITIALS]* *IssueSpeedingTicket*, where [YOUR\_INITIALS] are your initials.
2. Select the  symbol and add [YOUR\_INITIALS]TrafficViolation. Your flow should look like this (where NTC should be your initials):

A screenshot of a message

Description automatically generated

1. Select the rightmost  symbol and add *[YOUR\_INITIALS]write a doc to Cloudant*. Your flow should look like this:

A screenshot of a computer

Description automatically generated

1. Select *write a doc to Cloudant*.
2. Select *generate mapping suggestions*.
3. Slide the toggle *Hide this form from the user*. You should see this:

A screenshot of a computer

Description automatically generated

1. At the very bottom of the form, enter *Speeding* as the violation:

A screenshot of a computer

Description automatically generated

1. Select the rightmost  symbol and add *[YOUR\_INITIALS]get docs from Cloudant*. Your flow should now look like this:

A screenshot of a computer

Description automatically generated

1. Select the rightmost  symbol and add *Custom Forms->Input Form*
2. Slide the toggle *Hide this form from the user*. You should see this:

A screenshot of a computer

Description automatically generated

1. Press *the Add input field* button and assign it a type of *Single line text*:

A screenshot of a computer

Description automatically generated

1. Press *Next* and give it a name of *TicketURL.* Press Apply. You should see this:

A screenshot of a computer

Description automatically generated

1. Select *TicketURL* and map it to *driverId* from *get docs from Cloudant.*

**Important!** This must be *driverId* from *get docs from Cloudant* and **not** *write a doc to cloudant*

A screenshot of a computer

Description automatically generated

1. In the *Define Transformations* section, enter the following data exactly as specified:

Operator:

Replace a string

Search for:

D

Replace with:

<https://fineui.1apbmbk49s5e.eu-gb.codeengine.appdomain.cloud/?driverId=D>

You should now see this:

A screenshot of a computer

Description automatically generated

1. Select the rightmost  symbol and add *Custom Forms->Output Form.*
2. Enter # in the output form and select  *Input form*->*TicketURL* as the field:

A screenshot of a computer program

Description automatically generated

1. You should see this:

A screenshot of a chat

Description automatically generated

1. The completed skill flow should look as follows:

A screenshot of a computer

Description automatically generated

1. Press *Actions* and then save, publish, and add the skill to your catalog.
2. In Chat, run the *[YOUR\_INITIALS]IssueSpeedingTicket* skill.
3. Enter a speeding violation. Ensure your driver Id starts with a capital D, for example D123456:

A screenshot of a computer

Description automatically generated

1. Press Apply.
2. After a while you should all speeding violations. Search for your driver id specified above. If you find more than one entry for the same driver id, select the topmost one:

A screenshot of a computer

Description automatically generated

1. Press Apply
2. You should see a Ticket URL displayed. Click the URL. This URL invokes a reporting tool that creates the speeding ticket. You should see a speeding ticket like the one below:

A screenshot of a computer

Description automatically generated

Congratulations! You have completed the lab!

# Advanced Lab – Create your own Cloudant instance for Watsonx Orchestrate

In this section we implement the Cloudant skill used in the previous section.

## Local host Prerequisites

1. Create a Cloudant instance in your IBM Cloud account.
2. See: <https://cloud.ibm.com/docs/Cloudant?topic=Cloudant-getting-started-with-cloudant>
3. Create a Cloudant database with name *wxodb.*
4. Create credentials.

## Run on local host

1. Edit *setenv.bat* to your Cloudant credentials (see above).
2. Open Command prompt (NOT powershell).  Enter:

   setenv.bat

 Start a local nodejs server:

   npm start

## Code Engine Hosting Prerequisites

1. Follow Local Host Prerequisites above.
2. Create a *Code Engine* instance in IBM Cloud account. If using TechZone, reserve the following:

A screenshot of a computer

Description automatically generated

1. Login to the provisioned environment.
2. Create a code engine project called cloudant-facade, with the image referencing your docker image.

A screenshot of a computer

Description automatically generated

1. Within your CodeEngine project, select Secrets and configmaps. Press Create.
2. Create a configmap with name *dataserviceconfig* and press *Create.*
3. Create a code engine *configmap configuration* with your Cloudant credentials:

A screenshot of a computer

Description automatically generated

## Deployment to code engine on IBM Cloud

1. Open Git Bash shell from VSC
2. Login to IBM Cloud.

ibmcloud login --sso

1. In the IBM Cloud console, go to **Manage > Account > Account resources > Resource groups.** Select the resource group for Code Engine. E.g. *default*

ibmcloud target -g default

1. Select the code engine project:

ibmcloud ce project select -n [PROJECT\_NAME]

1. Start Docker Desktop
2. In Bash Shell, go to the dataservice folder e.g: /bamoe2wxo/cloudantFacade/dataservice
3. Within this folder, edit *build.sh* and *run.sh* and change the REGISTRY to your Docker registry.
4. Run *build.sh* to build the sample application in your docker repo:

./build

1. Run run.sh to deploy the application to Code Engine on IBM Cloud:

./run

1. Open your IBM Cloud app route in the browser

# Conclusion

In this lab were created a decision service and called as a skill from wastonx Orchestrate. We then created a composite flow to store the results of the decision into a database.

Thank you for taking this lab and I hope you create new applications using this technology.