

# **Learning IBM Business Automation Open Edition**

---

**None**

*IBM*

*None*

## Table of contents

---

1. IBM Business Automation Open Edition 9.0 Enablement	3
1.1 Labs at IBM TechXChange 2023	3
1.2 Your Journey Begins Here	3
1.3 IBM Business Automation Open Edition 9.0 Decision Manager Exercises	3
1.4 IBM Business Automation Open Edition 8.0 Process Automation Exercises	3
1.5 More topics	3
1.6 About the guides	3
2. Learn more	5
2.1 Videos	5
2.2 Red Hat Scholars	5
2.3 Guides	5
3. Guided exercises	6
3.1 Get the tools for IBAMOE	6
3.2 Getting Started with Kogito	11
3.3 TechXchange	34
3.4 DMN Exercises	35
3.5 16. IBM Business Automation Open Edition 9.0 CI/CD Overview	138
3.6 Order Management	155
3.7 Working with event driven processes in IBAMOE	235
3.8 25. Getting started bc	253
3.9 Deploying PAM on OpenShift with Operators	260
3.10 Tools	282

# 1. IBM Business Automation Open Edition 9.0 Enablement

---

## 1.1 Labs at IBM TechXChange 2023

 **Guided exercises for TechXchange Session 1717**

## 1.2 Your Journey Begins Here

 **Get the tools** 

 **First Kogito Project Setup** 

## 1.3 IBM Business Automation Open Edition 9.0 Decision Manager Exercises

**Learn DMN** 

**Implement a On-push CI/CD** 

## 1.4 IBM Business Automation Open Edition 8.0 Process Automation Exercises

**Getting started with Proces** 

**Event-driven processes** 

## 1.5 More topics

**Extra Learning** 

**IBAMOE on OpenShift** 

## 1.6 About the guides

### 1.6.1 Getting Started with IBAMOE: Order Management

A great guide for users who are trying IBAMOE for the first time. Recommended getting started guide.

### 1.6.2 Learn DMN

In this hands-on workshop, users can learn about the DMN specification, author decisions, and deploy it using IBM Decision Manager with basic, intermediate and advanced exercises.

### 1.6.3 Event-driven processes

This hands-on workshop allows the user to validate the experience of creating loan approval workflow with decision automation. The user will also have an introduction to Fuse, develop an endpoint and consume it using the workflow.

### 1.6.4 Learn the tools

In this hands-on workshop, create new decision services using Kogito tooling and the VSCode extension , create unit tests and deploy them on KIE Server.

## 1.6.5 Learn how to deploy IBAMOE on OpenShift using Operators

Try out the business automation operator and learn how to manage your KIE Apps on OpenShift.

Check the [Learn more section](#) for more guides and references about business automation with the projects under the [KIE](#) umbrella.

---

Last update: 2023-12-06

## 2. Learn more

---

### 2.1 Videos

---

KIE Live events are live streams designed to facilitate knowledge sharing about the Business Automation topic, including business rules, decisions, processes, resource planning, tooling, and AI. They're community events and anyone is welcome to attend.

Check out all the KIE Lives at: <https://red.ht/kielives>

### 2.2 Red Hat Scholars

---

You can refer to Red Hat Scholars to find several up-to-date guides about business automation topics, including Kogito.

Check out all the great guides available at: <https://redhat-scholars.github.io/cloud-native-business-automation/>

### 2.3 Guides

---

There are several guided exercises and workshops that are not yet part of this guide. You can find them below:

#### **Loan Approval Workshop with DMN**

This workshop is aimed at providing hands on experience creating DMN assets. This lab will implement a Loan Approval workflow. [Check out this guide here.](#)

#### **Loan Approval Workshop**

This workshop is aimed at providing hands-on experience creating Decision and Process Assets. This lab will implement a Loan Approval workflow. [Check out this guide here.](#)

#### **KIE Learning: IBAMOE and RHDM**

[This repository](#) is a set of explanation and hands-on labs which you can try on your environment and follow at your own pace. The step-by-step guides covers different topics, since basics to more advanced and specific features. The content is not sequential and it covers Decision Manager, Business Optimizer, Process Automation Manager and Kogito.

#### **DMN Handbook**

[This handbook](#) is a vademecum for the FEEL expression language from the DMN specification, as implemented by the Drools DMN open source engine.

---

Last update: 2023-12-06

## 3. Guided exercises

---

### 3.1 Get the tools for IBAMOE

---

#### 3.1.1 1. Introduction

Before getting started with **IBM Business Automation Open Edition 9.0**, we must first explore and prepare the different tools we'll need in order to go beyond the basics on this Business Automation journey.

We'll be using different tools as we move through the guided exercises: - During development, we'll use IBM Business Automation Open Editions **Canvas**, which is known in the open-source as KIE Sandbox. - To set up the local workstation, we'll use VSCode Plugins, Maven, GraalVM and more. - Since our goal with this enablement is to move towards Kogito adoption and its cloud-native approach, we'll mostly use Canvas and VSCode. Although, you can also find extra exercises covering the usage of Business Central, supported on the 8.x version.

---

Last update: 2023-12-06

### 3.1.2 2. Environment Setup

This section will cover the deployment of some of the tools locally that you will use in these exercises. **IBM Business Automation Open Edition 9.0**

In order to follow these guided labs, you should have IBAMOE 8.0+ in your local environment. IBAMOE is flexible in how you design. The developer runtime tools are predominantly focused in VSCode (download [here](#)) with Maven builds targeted at a more cloud native strategy. There are also runtime options available through the utilization of the KIE Server (Knowledge is Everything) to execute. This enablement session will go through both, but the evolution of the jBPM/Drools projects are definitely more towards the Kogito runtimes versus the Java EE-based KIE Server.

If you need to setup IBAMOE locally, you can use this repository to help you get up and running quickly: [IBM Business Automation Open Edition 9.0 Environment Setup](#). This will provide the files as two different forms, a locally built environment that will persist, or the option to have an ephemeral container that will lose the data stored between sessions. Both forms will provide a Business Central and KIE Server environment for use.

#### 2.1 VS Code

For running through the exercises, it is highly recommended that you get **Visual Studio Code** as this is the IDE that IBM Business Automation Open Edition 9.0 most supports for activities regarding both decisions and processes. To get Visual Studio Code go to [the Visual Studio Code download page](#) and download for your platform.

#### 2.2 Git, Maven and Java

These labs are going to assume you are already running , a running version of git, Maven 3.6.2+ and OpenJDK 11+. To get these added to your environment, you can follow the steps in this section. IBM Business Automation Open Edition 9.0 is built around git source code control and Maven archetypes. You will need these tools to do most of the content in these quick walk throughs.

##### 2.2.1 GIT

Git is the open-source solution for source code management. To get git, go to [here](#) to install it if you don't have it already. All of the major source code management systems work with the git command line and is ultimately platform-agnostic to those.

##### 2.2.2 MAVEN

With IBM Business Automation Open Edition 9.0, the components are built around a Maven architecture predominantly. What this ultimately means is your workstation needs to be able to communicate with one to many different Maven Repositories. These labs will use two in particular, the [Red Hat General Availability repository](#) and [Maven Central](#). You could easily replace the two repositories with a local environment one hosting the Maven dependencies as a mirror or based in a disconnected installation, but this is the easiest developer workflow for acquiring new dependencies. The reason we are pointing to the Red Hat Maven repository, at least in the short term, is that the builds for IBM Business Automation Open Edition 9.0 are being deployed there as they are the same binaries used within both the IBM and Red Hat products during the transition of Red Hat Process Automation Manager (RHPAM)/Red Hat Decision Manager (RHDM) from Red Hat into IBM Automation under the name of IBM Business Automation Open Edition 9.0 ( IBAMOE). The settings.xml file included below will use the local Maven repository at your **USER\_HOME/.m2/repository**, which when configuring Maven would be the default M2\_HOME that's created.

### Sample settings.xml file

```

<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository>${user.home}/.m2/repository</localRepository>
  <interactiveMode>true</interactiveMode>
  <usePluginRegistry>false</usePluginRegistry>
  <offline>false</offline>
  <profiles>
    <!-- Profile with online repositories required by IBAMOE -->
    <profile>
      <id>brms-bpms-online-profile</id>
      <repositories>
        <repository>
          <!-- Red Hat Maven Repository-->
          <id>jboss-ga-repository</id>
          <url>https://maven.repository.redhat.com/ga/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>jboss-ga-plugin-repository</id>
          <url>https://maven.repository.redhat.com/ga/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
    <profile>
      <id>maven-https</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <repositories>
        <repository>
          <!-- Maven Central Repository-->
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
  <activeProfiles>
    <!-- Activation of the BRMS/BPMS profile -->
    <activeProfile>brms-bpms-online-profile</activeProfile>
    <activeProfile>maven-https</activeProfile>
  </activeProfiles>
</settings>
```

#### 2.2.2.1 Linux/Windows/Mac

Follow the instructions at [the Maven Community](#) to download and update your path variables to incorporate it into your builds.

### 2.2.2.1.1 Example Linux installation

1. Download Maven 3.8.6 from [here](#)
2. If using the Skytap image, change to the downloads folder.

```
cd /home/pamadmin/Downloads
```

3. Now you need to extract the tar

```
tar xvzf apache-maven-3.8.6-bin.tar.gz
```

4. Now if there's an existing Maven in your stack, you can move it out with this command.

```
sudo mv /usr/share/maven /usr/share/maven-old
```

5. Now move the new download into your /usr/share with the following command.

```
sudo mv apache-maven-3.8.6 /usr/share/maven
```

6. With this, Maven should be updated, validate with running a Maven version command.

```
mvn -v
```

Your console should return a log similar to below

```
[pamadmin@host-1 maven]$ mvn -version
Apache Maven 3.8.6 (84538c9988a25aec085021c365c560670ad80f63)
Maven home: /usr/share/maven
Java version: 11.0.16.1, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.16.1.1-1.el8_6.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.18.0-372.26.1.el8_6.x86_64", arch: "amd64", family: "unix"
```

### 2.2.2.2 Mac Alternative

The easiest way to acquire Maven is to use **homebrew** and run the command `brew install maven`, which at the time of writing will install Maven 3.8.6.

### 2.2.3 JAVA

The assumption in these exercises are that you will utilize a supported JDK. These were tested with `openjdk 11.0.11 2021-04-20`, but other ones are supported. You should be using at least Java 11 with IBM Business Automation Open Edition 9.0 to ensure your best experience with the tooling.

## 2.3 Kafka

The Kafka setup requires `docker-compose` to run locally, to get this capability, you can use either Docker or Podman for this capability. To see which is right for you, [read this article](#). Coming later in 2022 will be the labs shifted to using an OpenShift deployment of AMQ Streams to minimize the local installations required.

Event-driven processes can react to the events that happens in the ecosystem. Kafka is an open-source even streaming platform, and currently, one of the most popular tools. In this type of architecture, we have the Kafka `topics` used as the communication layer in between the services. Each service can now be considered a `consumer` or a `producer`, in other words, each service can publish or consume events to/from the `topics`.

IBM supports the integration between IBAMOE and AMQ Streams (Kafka). To follow the labs, you should have an accessible Kafka server. The KIE Server (process engine) will communicate with the topics that we will create in the Kafka server.

If you don't have an environment available you can get a Kafka (Strimzi) server quickly running by using Docker. Let's clone the project to the enablement folder.

1. Create a new folder named `enablement` and access it:

```
mkdir ~/enablement && cd ~/enablement
```

2. Clone this repository to your local machine.

```
git clone https://github.com/hguerrero/amq-examples
```

The docker-compose file available in this quickstart should bootstrap everything you need to have your Strimzi up and running: Zookeeper, Kafka server v2.5.0, Apicurio Registry and a Kafka Bridge.

3. Access the `amq-examples/strimzi-all-in-one/` folder:

```
cd amq-examples/strimzi-all-in-one/
```

4. Start the Kafka environment:

```
docker-compose up
```

Docker will download the images and start the services for you. You now have a Kafka server running on localhost port 9092.

## 2.4 Creating the Kafka topics

In the upcoming labs we will need three topics: `incoming-requests`, `requests-approved` and `requests-denied`. Next, you need to create these topics in your Kafka server. If you are using the containerized option mentioned in this lab, you can create the topics using the `kafka-topics.sh` available in the `kafka` container.

1. Open a new terminal and let's using the `kafka` container we have just started. Enter the Kafka folder:

```
cd ~/enablement/amq-examples/strimzi-all-in-one /
```

2. Create the following three topics:

```
docker-compose exec kafka bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic incoming-requests
docker-compose exec kafka bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic requests-approved
docker-compose exec kafka bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic requests-denied
```

## 2.5 Next steps

At this point you should have IBAMOE, and Kafka on your environment. Before proceeding to the next steps, make sure you have both IBAMOE and Kafka server up and running.

---

Last update: 2023-12-06

## 3.2 Getting Started with Kogito

---

### 3.2.1 3. Introduction

This section will focus on how to setup your first Kogito projects from scratch. The method we're going to use for this is through the use of Maven to create it from an empty workspace, produce a DMN model and make it available both locally and how to use deploy it to OpenShift.

 [Initial project setup and walk through](#)

 [Deploy locally](#)

 [Deploy to OpenShift](#)

---

Last update: 2023-12-06

### 3.2.2 4. Using Maven to create the project workspace

---

In the previous section, you setup Maven locally in your environment, so you should now have access to all of the `mvn` commands that are associated with running it. The first thing we're going to create is a project using the proceeding steps:

1. We're going to create the service in Quarkus with the Maven commands below, this will create a Quarkus project called `quick-kogito` that will be versioned `1.0.0-SNAPSHOT` including the extensions `kogito-quarkus`, `dmn`, `resteasy-reactive-jackson`, `quarkus-smallrye-openapi`, `quarkus-smallrye-health` which will create a Quarkus DMN project with the openapi components to get the OpenAPI end points easily with health checks when deploying to OpenShift.

```
mvn io.quarkus:quarkus-maven-plugin:2.16.7.Final:create \
-DprojectGroupId=com.ibm.sample \
-DprojectArtifactId=quick-kogito \
-DprojectVersion=1.0.0-SNAPSHOT \
-DplatformVersion=2.16.7.Final \
-Dextensions=kogito-quarkus,dmn,resteasy-reactive-jackson,quarkus-smallrye-openapi,quarkus-smallrye-health
```

2. When you create this project you should get a bunch of Maven artifacts start to stream in your console that are being pulled and ultimately are left with a console message like the below:

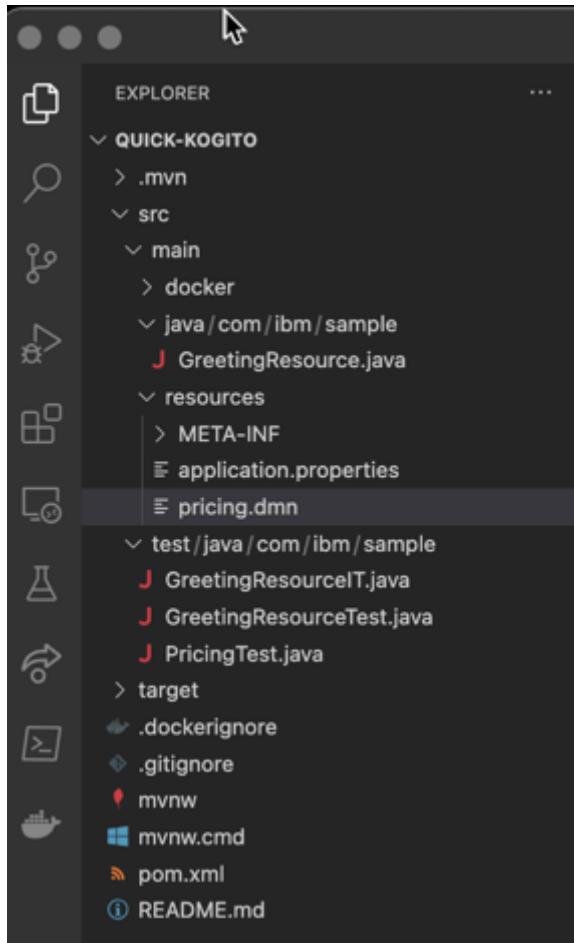
```
[INFO]
[INFO] =====
[INFO] Your new application has been created in /Users/timwutheNow/techxchange/quick-kogito
[INFO] Navigate into this directory and launch your application with mvn quarkus:dev
[INFO] Your application will be accessible on http://localhost:8080
[INFO] =====
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  8.555 s
[INFO] Finished at: 2023-07-17T13:35:03-04:00
[INFO] -----
```

NOTE: If you are using an image provided by IBM for labs, the 8080 port location that is labeled here will not work immediately because another service is utilizing this port. We will show how to change this in the next steps.

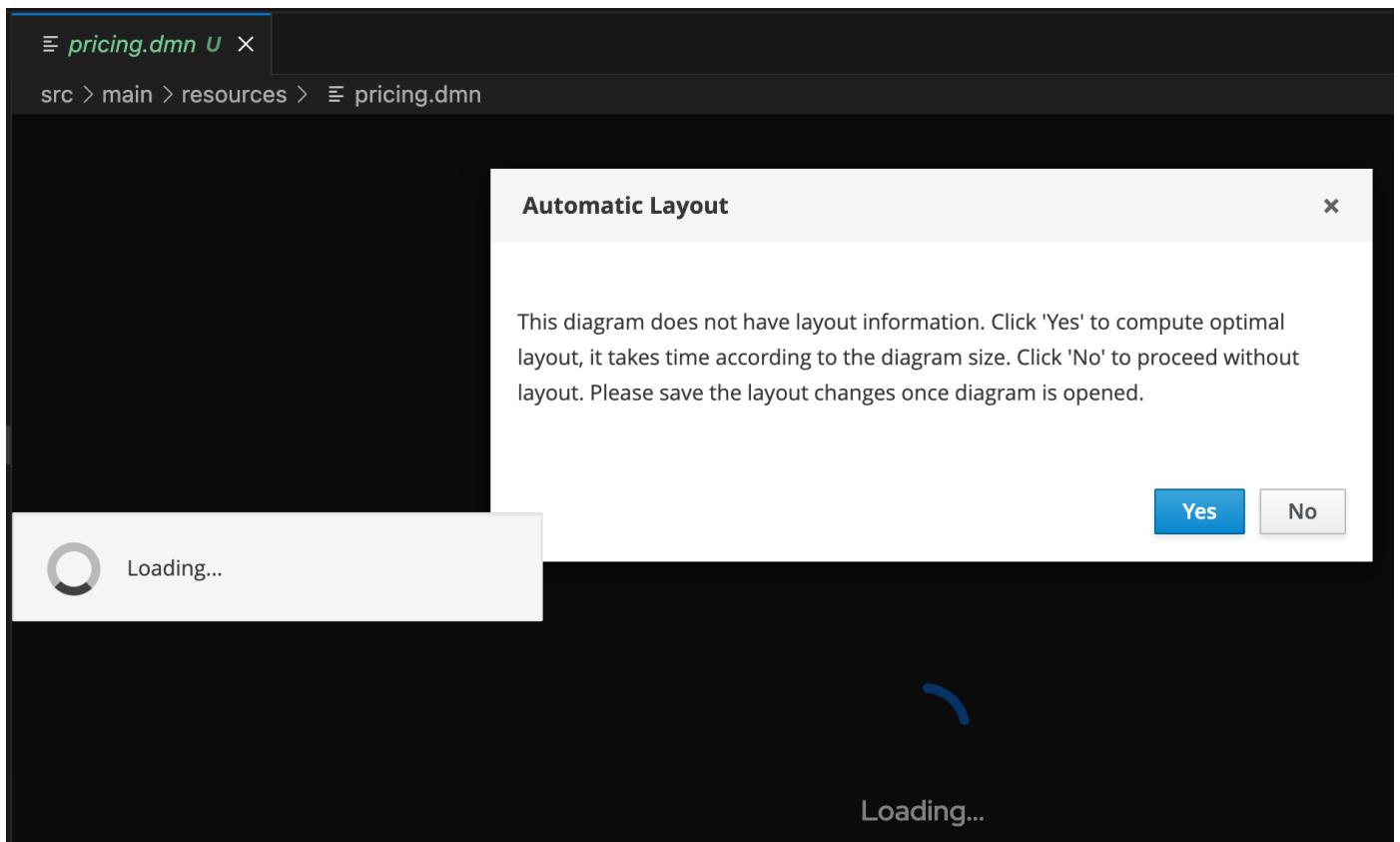
3. With VSCode installed to your PATH variables, you can open the workspace by doing the below command, otherwise open VSCode and navigate to where you ran the command for `quick-kogito` to be created at:

```
cd quick-kogito
code .
```

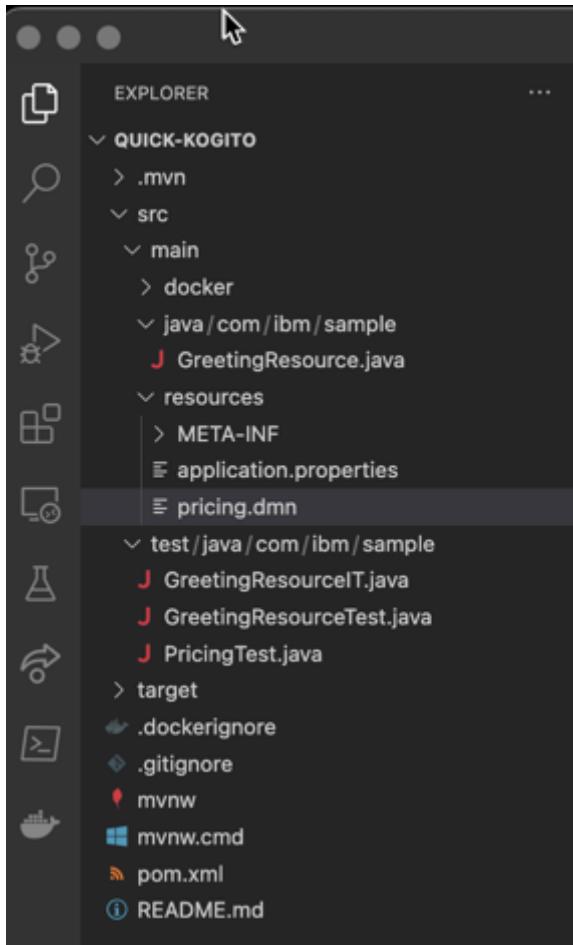
4. From here we can see the workspace's contents and if we expand the contents of `/src/main` you will see the creation of several artifacts. Within `java` you will have a `GreetingResource.java` and within `resources` you will have an `application.properties` and `pricing.dmn` file. These are sample files that can be later modified or deleted, but we will explore them first in this section, but will do more in later labs around the various end points.



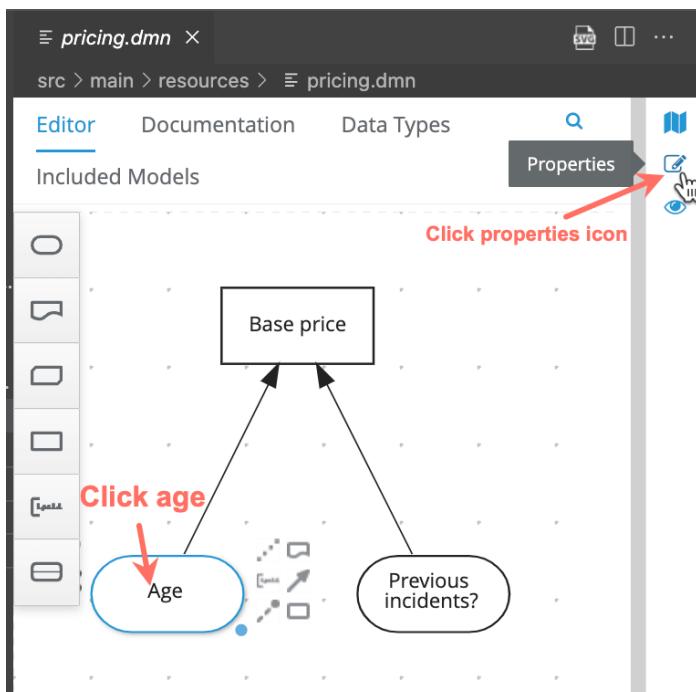
5. Let's first click the `pricing.dmn` file to open it. When you do so you may be greeted with a message similar to  
This diagram does not have layout information. Click 'Yes' to compute optimal layout, it takes time according to the diagram size.  
Click 'No' to proceed without layout. Please save the layout changes once diagram is opened. - if so click Yes to automap the DMN locations.



- When the diagram opens you will see something similar to below, so we will start exploring it. The DMN is made up of two inputs *Age* and *Previous incidents?*, which are used to make the decision, *Base price*.



7. If you click *Age* and then click the *Properties* icon on the right, you will open a pane for the input.

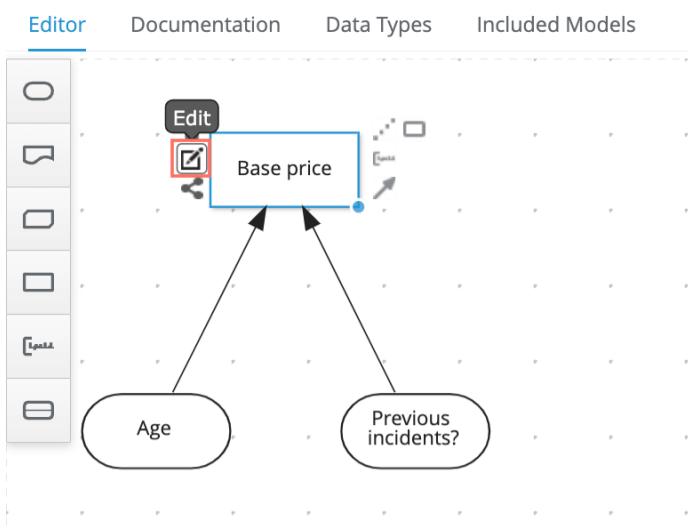


8. Within this pane, you can see information about the input *Age*, this includes that it is a number and what the input name is. More can be changed around this object, including changing the color of the node, font size, etc.

**Properties**

<b>Id</b>	<input type="text" value="3D76D863-9817-412A-ABC3-BEA66DCF03D1"/>			
<b>Description</b>	<input type="text"/>			
<b>Documentation Links</b>	<a href="#">+Add</a>			
<input type="text" value="None"/>				
<b>Name</b>	<input type="text" value="Age"/>			
<b>Information item</b> <b>Data type</b> <a href="#">Manage</a> <input type="text" value="number"/>				
<b>Styling details</b> <b>Background colour</b> <input type="color"/> <b>Border colour</b> <input type="color"/> <b>Font family</b> <input type="text"/> <b>Font colour</b> <input type="color"/> <b>Font size</b> <input type="text"/>				

9. To view the Decision, click the square decision node and select the `Edit` button to enter the decision for *Base Price*.



10. From here you will see the Decision Table that is associated with the Base Price decision. From here you will see two (2) input columns (`Age` and `Previous Incidents`), as well as one output column (`Base price`) all with their types below them. These types are controlled from the properties panel similarly to how they were opened when looking at `Age` a few steps ago. This decision has 4

different rows that could fire, with a Hit Policy of `UNIQUE` signified by the **U** in the top left corner of the table. A decision writer could make any comments they want to the table and have them saved towards the decision here

src > main > resources > `pricing.dmn`

[Editor](#) Documentation Data Types Included Models

« [Back to pricing](#) — **Stable** You're using the stable version of the Boxed Expression editor. Do you want to try the new editor? [Click here!](#)

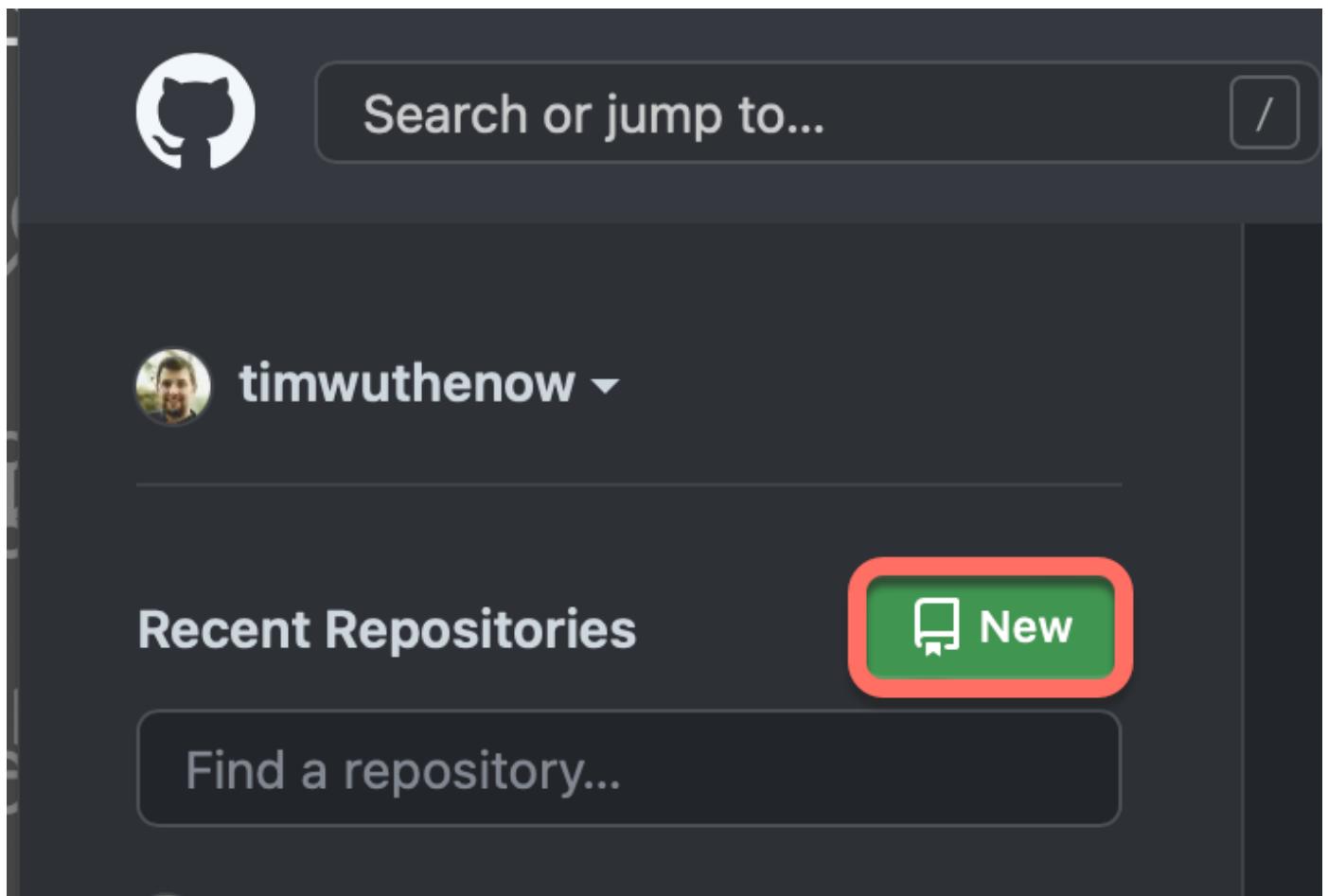
Base price (*Decision Table*)

U	Age (number)	Previous incidents? (boolean)	Base price (number)	comments
1	<21	false	800	
2	<21	true	1000	
3	>=21	false	500	
4	>=21	true	600	

#### 4.1 Create a GitHub repository for the project

One last thing we're going to do is to create a GitHub repository for this service, so you have somewhere to store our changes and also take advantage of building it into a cloud service running on OpenShift.

1. You can create a repository on [GitHub](#). To do this login to your GitHub username (or create one if you don't have one!) and from your home page and click the green `New` icon near the top left of the page (or you can navigate directly to [here](#))



2. Fill out the form with your values

- **Repository Name:** quick-openshift-kogito
- Select **Public** for now
- Add a **Description** if you want
- Click **Create Repository**

**Repository template**  
Start your repository with a template repository's contents.

No template ▾

---

**Owner \*** timwuthenow / **Repository name \*** quick-openshift-kogito ✓

Great repository names are short and memorable. Need inspiration? How about [verbose-adventure?](#)

**Description (optional)**  
A quick repo designed to show Kogito being deployed into OpenShift|

---

 **Public**  
Anyone on the internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

**Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

---

ⓘ You are creating a public repository in your personal account.

**Create repository**

3. Copy the command for **...or create a new repository on the command line** as we're going to take exactly what's in our repository add a mostly empty `README.md` and push those changes to GitHub. The below command is an example **and will not be the exact same as what you have on your repository**. Copy yours.

The screenshot shows a GitHub repository page for 'timwuthenow / quick-openshift-kogito'. The page has a dark theme. At the top, there are navigation links: Code (highlighted), Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below these are buttons for Pin, Unwatch (1), Fork (0), and Star (0). The main content area has a section titled 'Quick setup — if you've done this kind of thing before' with a link to 'Set up in Desktop' or 'HTTPS' or 'SSH' with the URL 'https://github.com/timwuthenow/quick-openshift-kogito.git'. A red arrow points from the text '...or create a new repository on the command line' to the copy icon (a square with a right-pointing arrow) at the end of the command-line code block. The command-line code is:

```
echo "# quick-openshift-kogito" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/timwuthenow/quick-openshift-kogito.git  
git push -u origin main
```

Below this, another red arrow points from the text '...or push an existing repository from the command line' to the copy icon at the end of the command-line code block. The command-line code is:

```
git remote add origin https://github.com/timwuthenow/quick-openshift-kogito.git  
git branch -M main  
git push -u origin main
```

At the bottom, there is a section titled '...or import code from another repository'.

4. When this is done, you are finished with this section. Proceed to either [deploying locally](#) or [deploying on OpenShift](#)

---

Last update: 2023-12-06

### 3.2.3 5. Deploying the Project locally as a Quarkus runtime

Now that we have seen a little of the DMN Decision, let's use the combined power of Quarkus ([Learn more here about the Kubernetes-native Java Stack that is the best place for Kogito to run!](#)) and Kogito to quickly build and deploy the service using Maven commands and also having the option to deploy it to OpenShift as well.

1. The first thing we're going to do is open a terminal (either in VSCode or whatever method you prefer) and validate that you have Maven and Java installed.

```
java -version
```

*Expected output should be similar to*

```
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment AdoptOpenJDK-11.0.11+9 (build 11.0.11+9)
OpenJDK 64-Bit Server VM AdoptOpenJDK-11.0.11+9 (build 11.0.11+9, mixed mode)
```

```
mvn -version
```

*Expected output should be similar to:*

```
Apache Maven 3.8.6 (84538c9988a25a0c085021c365c560670ad80f63)
Java version: 18.0.2, vendor: Homebrew, runtime: /opt/homebrew/Cellar/openjdk/18.0.2/libexec/openjdk.jdk/Contents/Home
Maven home: /opt/homebrew/Cellar/maven/3.8.6/libexec
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "12.6", arch: "aarch64", family: "mac"
```

2. Now that our environment has the appropriate resources, let's build and run the service locally. Quarkus does this very simply with the command `mvn quarkus:dev` which will operate the container in development mode. By default it will run on 2 ports when running this command, `8080` and `5005`. These ports can be maintained in the `application.properties` file in `src/main/resources/` or you can modify the `mvn quarkus:dev` command to load those properties from the command line. If you want to do the command line, you can do `mvn quarkus:dev -Dquarkus.http.port=8085 -Ddebug=7007` which would setup your application to run off of port `8085` and be able to attach a remote debugger at `7007`. To change the default host in `application.properties` add the following line.

```
quarkus.http.port=8085
```

3. After the project's `application.properties` has been updated, let's start the Quarkus service by running the following command. This command will run the remote debugger (which we are not using in these labs) on port `6006` so that if you already have another application running on `5005` (the default port) it won't give an error.

```
mvn quarkus:dev -Ddebug=6006
```

4. When the service is ready, it will produce a log similar to the below:

```
$ quick-kogito % mvn quarkus:dev -Ddebug=6006
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.ibm.sample:quick-kogito >-----
[INFO] Building quick-kogito 1.0.0-SNAPSHOT
[INFO] [ jar ] -----
...
2022-09-29 20:48:34,404 INFO  [org.kie.kog.cod.dec.DecisionValidation] (build-23) Initializing DMN DT Validator...
2022-09-29 20:48:34,405 INFO  [org.kie.kog.cod.dec.DecisionValidation] (build-23) DMN DT Validator initialized.
2022-09-29 20:48:34,405 INFO  [org.kie.kog.cod.dec.DecisionValidation] (build-23) Analysing decision tables in DMN Model 'pricing' ...
2022-09-29 20:48:34,410 INFO  [org.kie.kog.cod.dec.DecisionValidation] (build-23) analysis for decision table 'Base price':
2022-09-29 20:48:34,412 INFO  [org.kie.kog.cod.dec.DecisionValidation] (build-23)   Decision Table Analysis of table 'Base price' finished with no messages to be reported.
2022-09-29 20:48:34,494 INFO  [org.kie.kog.qua.com.dep.KogitoAssetsProcessor] (build-38) reflectiveEfestoGeneratedClassBuildItem
org.kie.kogito.common.deployment.KogitoGeneratedSourcesBuildItem@415815e1
--/ \ / / / _ | / \ // / / / /
-/ / / / / / _ | / , _/ , < / / / \
--\ \ \ \ \ / _ | _ / | _ / | \ _ / /
2022-09-29 20:48:35,728 INFO  quick-kogito 1.0.0-SNAPSHOT on JVM started in 2.432s.
2022-09-29 20:48:35,728 INFO  Listening on: http://localhost:8085
2022-09-29 20:48:35,728 INFO  Profile dev activated. Live Coding activated.
2022-09-29 20:48:35,728 INFO  Installed features: [cdi, kogito-decisions, kogito-predictions, kogito-processes, kogito-rules, resteasy-reactive, resteasy-reactive-jackson, smallrye-context-propagation, smallrye-openapi, swagger-ui, vertx]
```

### 5.1 Using the project that is created

Now that your application is created, you can start using it more for live development and change as you go.

- To access the Quarkus Development console, in the logs, you can Command/Ctrl+click the link in VSCode or go to ()[http://localhost:8085], directly.

The screenshot shows the Quarkus Dev UI interface. At the top, it displays the application name "quick-kogito 1.0.0-SNAPSHOT (powered by Quarkus 2.13.2.Final)". Below this, there are several cards:

- Build**: Shows "Build Steps".
- Configuration**: Shows "Config Editor" and "Dev Services".
- ArC**: Shows build time CDI dependency injection statistics: Beans 61, Observers 4, Interceptors 1, Fired Events, Invocation Trees, and Removed Components 93.
- Kogito - Process (jBPM)**: Shows "Add Kogito Processes capabilities - Includes Process (jBPM) Engine".
- RESTEasy Reactive**: Shows a warning message: "A JAX-RS implementation utilizing build time processing and Vert.x. This extension is not compatible with the quarkus-reactive extension, or any of the extensions that depend on it." It also lists "Endpoint scores", "List of endpoints", "Exception mappers", and "Parameter converter providers".
- SmallRye Health**: Shows "Monitor service health" with links to "Health" and "Health UI".
- SmallRye OpenAPI**: Shows "Document your REST APIs with OpenAPI - comes with Swagger UI" with links to "OpenAPI" and "Swagger UI".

- The development console will show various aspects of your deployment. A quick highlight will be:

- The list of end points that are available
- The produced Maven artifact name (in this case quick-kogito 1.0.0-SNAPSHOT)
- You can modify configurations of your running application from here
- The Swagger UI and OpenAPI pages to get more information and test out your end points.

We will be clicking the Swagger UI link to open the Swagger page to try out our sample DMN.

This screenshot is identical to the one above, showing the Quarkus Dev UI interface. The "SmallRye OpenAPI" card is specifically highlighted, showing the "Swagger UI" link which is enclosed in a red box.

- Here you can see your generated Quarkus endpoints that deal with the Decision model. When building a DMN Decision Service with IBM Business Automation Open Edition 9.0 in Kogito, it will be able to use the DMN Model, coupled with the data model created in it to generate the required end points and inputs. This creates a **Domain Specific Decision Service**, which is different than how KIE Server worked before in that you would interact with a generic API and insert the objects for execution. By going away from the classpath loading strategy in Kogito, Decision Services are ready to be deployed into a cloud native environment.

Your payloads are simpler and easier to reuse across any services as required without having to know exactly how the service is deployed, the objects associated with the deployment and more.

4. The service that was created for the Decision Service is called **Pricing Resource** and if you click any of the end points you will get more information on them. The first end point to look at is the `Post /pricing`. This endpoint can be used to execute a decision and return just the data associated with the decision (inputs and outputs).

5. After clicking, you can edit the Domain Specific payload in input that matches your decision input nodes (`Age` and `Previous incidents`) with values you want to test. Notice that the endpoints use faces in the object names, this matches the expected values that were tied to the inputs in the model as opposed to camelCase or snake\_case or PascalCase. This helps with a consistent view across the Kogito stack - from design to runtime.

6. Your result should look similar to the one below. As you can see, the Swagger-UI page produces a few things of note, the first being a Curl command to reproduce locally, the URL you called, and then ultimately the response from the service.

**Responses**

**Curl**

```
curl -X 'POST' \
  'http://localhost:8085/pricing' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "Age": 35,
    "Previous incidents?": true
}'
```

**Request URL**

```
http://localhost:8085/pricing
```

**Server response**

Code	Details
200	<p><b>Response body</b></p> <pre>{   "Previous incidents?": true,   "Age": 35,   "Base price": 600 }</pre> <p><b>Download</b></p> <p><b>Response headers</b></p> <pre>content-length: 54 content-type: application/json; charset=UTF-8</pre>

**Responses**

Code	Description	Links
default	DMN output	No links

**Media type**

application/json

Controls Accept header.

**Example Value | Schema**

```
{
  "Base price": 0,
  "Age": 0,
  "Previous incidents?": true
}
```

7. The next endpoint to evaluate is the `/pricing/dmnresult` endpoint which will tell you more about the decision that executed. You can use the same payload from the previous steps to show the differences of what is presented. That payload can be similar to the code below provided in JSON. Click `Try it out` and repeat the steps from above.

8. The input for the `Try it out` can be used from below.

**POST /pricing/dmnresult**

**Parameters**

No parameters

**Request body**

application/json

`Try it out`

```
{
  "Age": 35,
  "Previous incidents?": true
}
```

9. Click `Execute` with the updated payload.

**POST** /pricing/dmnresult

**Parameters**

No parameters

**Request body**

application/json

**DMN input**

```
{
  "Age": 35,
  "Previous incidents?": true
}
```

**Execute**

10. The result will go into more detail about which Decision Nodes were executed against and more. When there is more than one Decision Node available, you can see the results from each node.

**Responses**

**Curl**

```
curl -X 'POST' \
'http://localhost:8085/pricing/dmnresult' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "Age": 35,
  "Previous incidents?": true
}'
```

**Request URL**

http://localhost:8085/pricing/dmnresult

**Server response**

Code	Details
200	<p><b>Response body</b></p> <pre>{   "namespace": "ns1",   "modelName": "pricing",   "dmnContext": {     "Previous incidents?": true,     "Age": 35,     "Base price": 600   },   "messages": [],   "decisionResults": [     {       "decisionId": ".72DFBC92-82CB-4544-A25F-E2CED29497B6",       "decisionName": "Base price",       "result": 600,       "messages": [],       "evaluationStatus": "SUCCEEDED"     }   ] }</pre> <p><b>Download</b></p> <p><b>Response headers</b></p> <pre>content-length: 284 content-type: application/json; charset=UTF-8</pre>

**Responses**

Code	Description	Links
200	OK	No links

Now you can play with the DMN model more and see how those changes impact your Quarkus service. Since you're in Dev mode, as you make changes, those changes should be available to you each time you save. Try it out!

### 3.2.4 6. Under construction

This content will be completed soon!

#### 6.1 Deploying your service to OpenShift as a Kogito Quarkus Service

Now that we have seen a little of the DMN Decision, let's use the combined power of Quarkus ([Learn more here about the Kubernetes-native Java Stack that is the best place for Kogito to run!](#)) and Kogito to quickly build and deploy the service to OpenShift. If you have already done the deploy locally, you will be able to skip some of these steps and start at [Getting Ready for OpenShift Deployment](#).

When deploying to OpenShift you will need to take the following things into account:

- The application with your Kogito microservices is in a Git repository that is reachable from your OpenShift environment.
- You have access to the OpenShift web console with the necessary permissions to create and edit KogitoBuild and KogitoRuntime.
- If implementing Quarkus, update the pom.xml file of your project contains the following dependency for the Quarkus smallrye-health extension. This extension enables the liveness and readiness probes that are required for Quarkus-based projects on OpenShift. If you created the project like in [01\\_walk\\_through.md](#), you will already have this extension added to your deployment.

#### 6.2 Deploying the Kogito Operator

1. First create a namespace for this to deployed into from the OpenShift console. For example `tim-kogito-on-openshift`.

# Create Project

An OpenShift project is an alternative representation of a Kubernetes namespace.

[Learn more about working with projects ↗](#)

**Name \*** ?

tim-deploys-kogito-on-openshift

**Display name**

Kogito deploy to OpenShift

**Description**

Namespace used to deploy Kogito Decision Services to OpenShift.

[Cancel](#)

[Create](#)

2. Next, we need to install the Kogito Operator from the OperatorHub. To do this, login to the OpenShift Cluster console and login as an admin. From here, go to `Operator ==> OperatorHub` to open the OperatorHub.

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, there is a sidebar with the following navigation options:

- Administrator
- Home
  - Overview
- Projects
  - Projects (selected, highlighted with a blue bar)
  - Search
  - API Explorer
  - Events
- Operators
  - OperatorHub (button highlighted with a red box)

The main content area displays the "Project details" for "PR tim-kogito-on-openshift". The top navigation bar includes "Projects > Project details". Below the title, there are tabs for "Overview" (selected), "Details", "YAML", and "Workl".  
The "Overview" section contains the following details:

- Details** (with a "View all" link)
- Name**: tim-kogito-on-openshift
- Requester**: IAM#tim.wuthenow@ibm.com
- Labels**:
  - kubernetes.io/app=tim-kogito
  - openshift-pipelines.teams=tim-kogito
- Description**

On the right side, there is a vertical status column with icons and letters:

- A (green checkmark)
- B (yellow exclamation mark)
- C (orange circle)
- D (blue square)
- E (green triangle)
- F (yellow square)
- G (blue circle)
- H (green square)
- I (blue triangle)
- J (green circle)
- K (yellow square)
- L (blue circle)
- M (green square)
- N (blue triangle)
- O (yellow circle)
- P (blue square)
- Q (green triangle)
- R (blue circle)
- S (green square)
- T (blue triangle)
- U (green circle)
- V (yellow square)
- W (blue circle)
- X (green square)
- Y (blue triangle)
- Z (green circle)

3. Search for Kogito and click the IBM BAMOE Kogito Operator tile.

Project: tim-kogito-on-openshift ▾

## OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat Marketplace. You can install Operators on your clusters to provide optional add-on functionality. Operator capabilities will appear in the Developer Catalog providing a self-service experience.

All Items

- AI/Machine Learning
- Application Runtime
- Big Data
- Cloud Provider
- Database
- Developer Tools
- Development Tools
- Drivers and plugins
- Integration & Delivery
- Logging & Tracing
- Modernization & Migration

All Items

kogi



Red Hat

IBM BAMOE Kogito Operator  
provided by IBM

IBM BAMOE Kogito Operator for  
deployment and management of  
Kogito services.

4. From the Operator Form, click `Install` at the top left to install the Operator.

Operator install

5. Install it to your namespace you created earlier. This way if there's a version already installed, you're not colliding with it and it is contained to just your namespace. The drawback is that it is limited to this namespace.

## Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

### Update channel \* ⓘ

8.x

### Installation mode \*

All namespaces on the cluster (default)

Operator will be available in all Namespaces.

A specific namespace on the cluster

Operator will be available in a single Namespace only.

### Installed Namespace \*

PR tim-kogito-on-openshift



### Update approval \* ⓘ

Automatic

Manual

### IBM IBM BAMOE Kogito Operator

provided by IBM

#### Provided APIs

##### KB Kogito Build

KogitoBuild handles how to build a custom Kogito service in a Kubernetes/OpenShift cluster.

##### KI Kogito Infra

KogitoInfra is the resource to bind a Custom Resource (CR) not managed by Kogito Operator to a given deployed Kogito service. It holds the reference of a CR managed by another operator such as Strimzi. For example: one can create a Kafka CR via Strimzi...

##### KR Kogito Runtime

KogitoRuntime is a custom Kogito service.

6. The installation may take a few minutes, but once it is completed, you will be ready to deploy your service. Click **View Operator**

### Success

#### 6.2.1 DEPLOYING YOUR FIRST KOGITO APPLICATION ON OPENSHIFT

Now that we've got the Operator installed, we're going to see how it works. This lab will walk through the deployment of a Quarkus deployment of the Kogito Service. There will be a collapsed section to refer to if the service was a Spring Boot service instead so there's a reference to it to show how easy it both types of deployments are utilizing the operator.

1. Once the operator is installed, you will click the Kebab icon and `Create KogitoBuild` to create the build.

`Create KogitoBuild`

2. Since we're developing in Quarkus, our implementation details handled in yaml are fairly minimal. We will point to a git repository where the service is deployed. In the previous section you created a repository, e.g. `github.com/timwuthenow/quick-openshift-kogito`.

Last update: 2023-12-06

## 3.3 TechXchange

---

### 3.3.1 7. IBM Business Automation Open Edition 9.0 labs at IBM TechXchange 2023

Welcome to the Session 1717, at IBM TechXchange 2023.

In this session you'll have a hands-on experience with cloud-native decision automation using IBM Business Automation Open Edition 9.0. As part of our learning journey, we'll explore many capabilities of IBAMOE, along with having a better understanding of its architecture, how its components work together and ultimately, allows to create cloud-native decision services that can be deployed on cloud platforms such as Red Hat OpenShift Container Platform (OCP).

To create our decision models, we'll use a new tool made available in 9.0, IBM Business Automation Manager Canvas, and we'll combine it with the developer joy brought by Quarkus, and how we can quickly deploy our automation solutions to a cloud environment.

We will focus predominantly on the IBAMOE 9.0 release, exploring how to automate a decision using Open Standards such as DMN, and running it with a Kogito-based Decision Service.

#### Note

If you want to go a step further, you'll also have available to you, labs focused on the 8.0.3 version of Open Editions, specially, on the Process Automation capabilities. Aligned with the evolution of the technology landscape and the future of this solution, the process automation capabilities are currently shifting towards a cloud-native architecture as we'll learn today. We hope to see you around in TechXchange 2024, to experience together all the exciting innovation we're building for this IBM's open source business automation solution!

Now, let's get started!

#### 7.1 Create your first Decision Service using IBM Decision Manager Open Edition

[Intro to DMN](#) 

[Explore DMN](#) 

[Deeper Dive into DMN](#) 

##### 7.1.1 GO ONE STEP FURTHER (OPTIONAL)

7.1.1.1 Explore process automation with IBM Business Automation Open Edition 8.0

[Getting started with Process](#) 

[Event-driven processes](#) 

7.1.1.2 More Exercises

[First Kogito Project Setup from Scratch](#) 

[Implement a On-push CI/CD](#) 

---

Last update: 2023-12-06

## 3.4 DMN Exercises

---

### 3.4.1 8. Learn Decision Automation with DMN

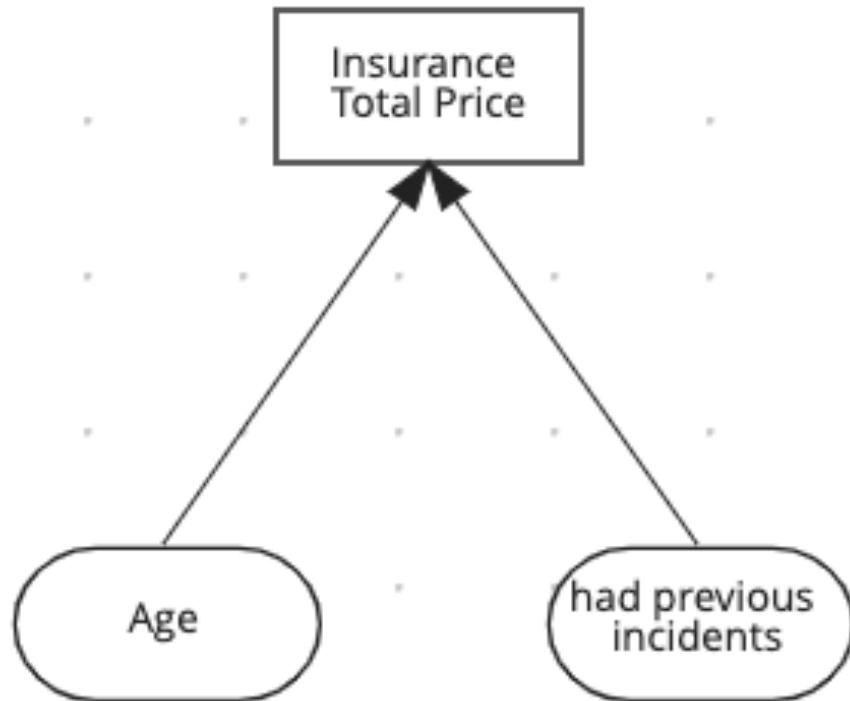
It's time to learn how Decision Automation can be made simple with Open Standards And Open Editions. Throughout these exercises, you'll experiment the development of decisions using Decision Model and Notation - DMN, and learn how to run, deploy and consume your decisions.

Before getting started, let's get familiar with the exercises you'll be working on and get introduced to the concepts of the DMN Standard, an increasingly adopted strategy for developing rules in the decision automation market.

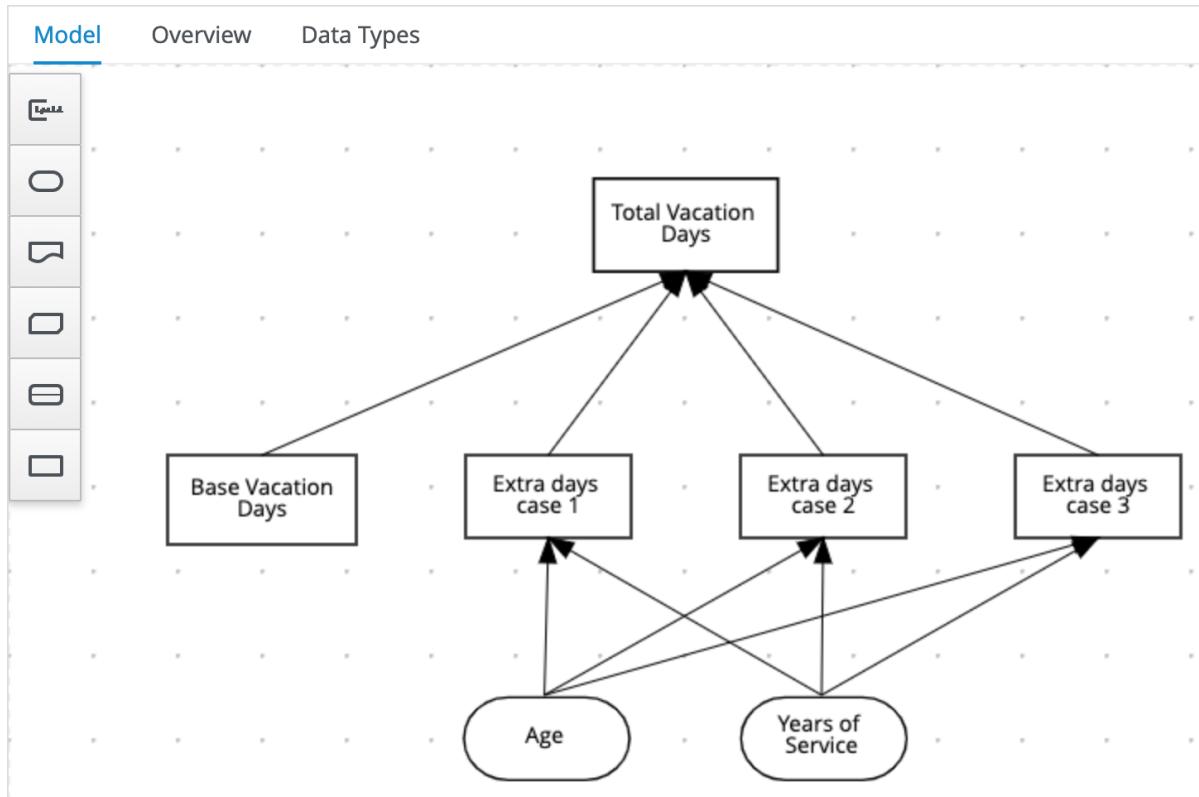
### 8.1 Goals of the Guided Exercises

These are the labs you have available in this workshop:

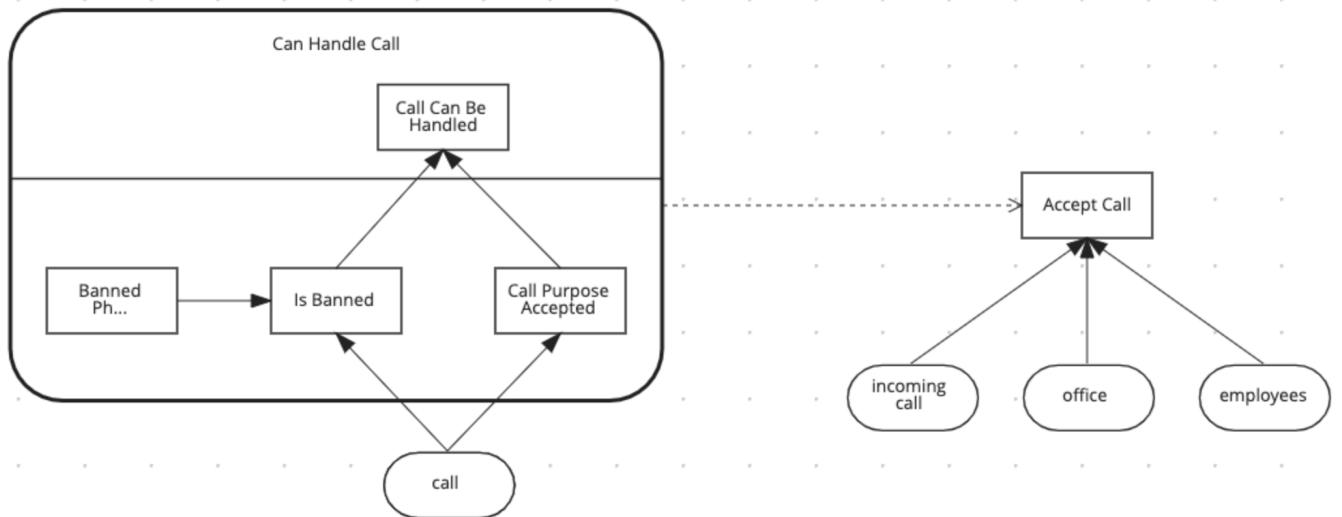
- Getting Started, Insurance Price calculation: Start by importing an existing module. Explore, deploy and test it using local development capabilities of Open Editions.



- Intermediate, Vacation Days: Author a model from scratch, use decision tables, work with different hit policies, different FEEL constructs and expressions. Finally, you will deploy it and test it, optionally running it on OpenShift.



- Advanced, Call Center: You will author a model from scratch, create data types, consume DMN decision services from within decision nodes, and more FEEL constructs and expressions. Finally, you will deploy and test it locally and optionally on OpenShift.



These are independent guided exercises and you don't need to implement the previous use case to implement the next one. If you are more comfortable with DMN, you can go to the more advanced labs immediately.

Before moving forward, get up to speed with what is DMN and explore the editor capabilities you'll be using.

## 8.2 Introduction to DMN

### 8.2.1 WHAT IS DMN

DMN, along with BPMN, is managed by the Object Management Group (OMG). Let's have a look at what OMG describes about the [DMN standard](#), in their website:

### OMG explains what is DMN

*"DMN is a modeling language and notation for the precise specification of business decisions and business rules. DMN is easily readable by the different types of people involved in decision management. These include: business people who specify the rules and monitor their application; business analysts."*

DMN is designed to work alongside BPMN and/or CMMN, providing a mechanism to model the decision-making associated with processes and cases. While BPMN, CMMN and DMN can be used independently, they were carefully designed to be complementary. Indeed, many organizations require a combination of process models for their prescriptive workflows, case models for their reactive activities, and decision models for their more complex, multi-criteria business rules. Those organizations will benefit from using the three standards in combination, selecting which one is most appropriate to each type of activity modeling. This is why BPMN, CMMN and DMN really constitute the “triple crown” of process improvement standards.”

IBM Business Automation Open Edition 9.0 brings a set of graphical tooling that allow you to author decisions using DMN and a lightweight engine that can execute these decisions. The engine and the authoring tooling set are decoupled and you can scale it independently.

#### 8.2.2 TOOLING SET

With IBM Business Automation Open Edition 9.0 you can author decisions in multiple ways - and using the exact same editor across different environments:

- [IBM Business Automation Manager Canvas](#) brings a new experience for easily working with decisions and processes in IBM Business Automation Open Edition 9.0. This is a highly scalable and lightweight method for maintaining diagrams that the community has moved towards and which IBM has most recently embraced and been working hard on!
- [Business Automation VSCode Extension](#)
- A developer IDE ([Visual Studio Code](#)) extension that allows the visualization and editing of BPMN, DMN and Test Scenarios inside VSCode.
- If you are currently using IBM Business Automation Open Edition 8.0 and are utilizing things like KIE Server and Business Central, you can also build the DMN model within there. The preference for this lab is to focus on IBM Business Automation Open Edition 9.0, but this option is capable of being done. A core component of IBM Business Automation Open Edition 9.0 is intercompatibility between versions and IBM's heavy emphasis on DMN is that the models produced for varying levels of the product are still compatible across releases. The walkthroughs are going to focus on IBM Business Automation Manager Canvas for the DMN labs.

Adding to the tools above, there are also other tools and knowledge sources you can leverage from the open source community. Listed below are you'll find resources developed in collaboration between IBM and Red Hat, for the community (not supported):

- [DMN FEEL Handbook](#)
- A handbook for the FEEL expression language from the DMN specification, as implemented by the Drools DMN open source engine.
- [Learn DMN in 15 minutes](#)
- A guided tour in a website through the elements of DMN
- [Open Source Online Editors](#)
- [BPMN.new](#) - A free online editor for business processes;
- [DMN.new](#) - A free online editor for decision models;
- [PMML.new](#) - A free online editor for scorecards;

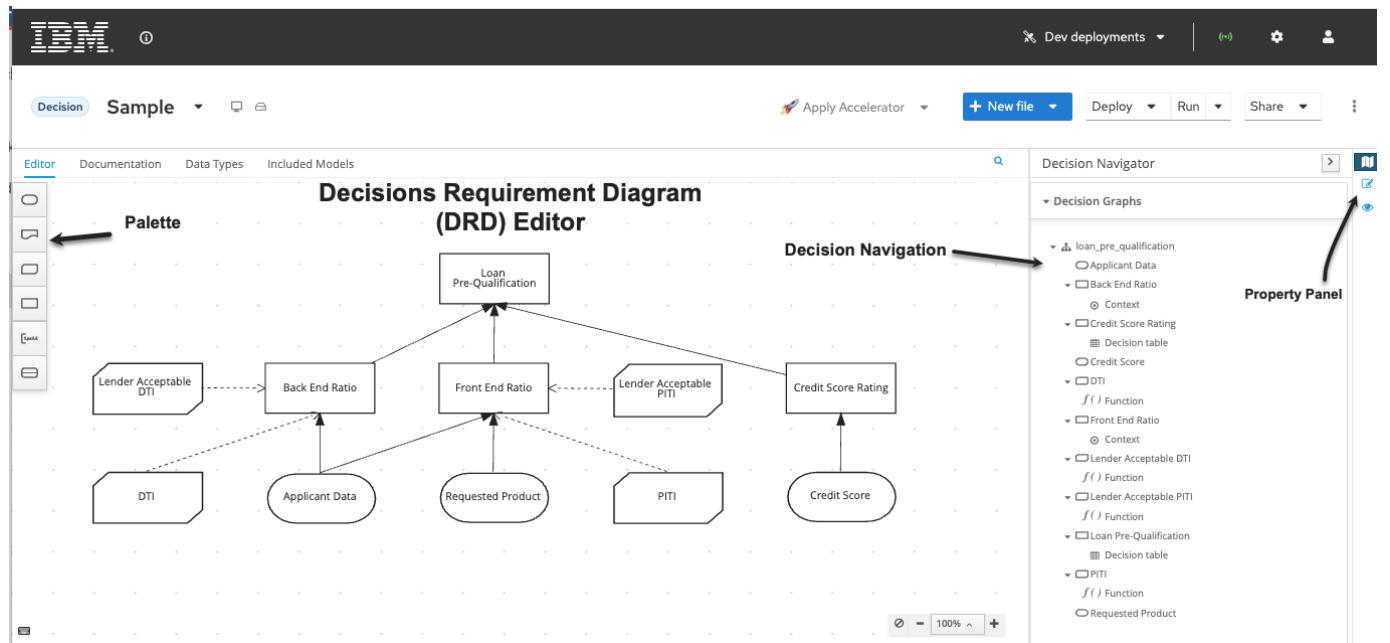
#### 8.2.3 EXPLORING THE DECISIONS EDITOR

Both in Canvas and VSCode, you'll have the same components within the the DMN Editor. They consist of:

## 8.2.3.1 Editor Pane

From the Editor pane you can edit and view multiple areas including:

- **Decision Navigator:** shows the nodes used in the Decision Requirements Diagram (DRD, the diagram), and the decisions behind the nodes. Allows for quick navigation through the model.
- **Decision Requirements Diagram Editor:** the canvas in which the model can be created.
- **Palette:** Contains all the DMN constructs that can be used in a DRD, e.g. Input Node, Decision Node, etc.
- **Expression Editor:** Editor in which DMN boxed expressions, like decision tables and literal expressions, can be created.
- **Property Panel:** provides access to the properties of the model (name, namespace, etc), nodes, etc.



[Editor](#)   [Documentation](#)   [Data Types](#)   [Included Models](#)

[Back to loan\\_pre\\_qualification](#)

### Credit Score Rating (*Decision table*)

<b>Decision table</b>			
U	<b>Credit Score.FICO (number)</b>	<b>Credit Score Rating (Credit_Score_Rating)</b>	<i>Enter Text</i>
1	<b><math>\geq 750</math></b>	"Excellent"	
2	<b>[700..750)</b>	"Good"	
3	<b>[650..700)</b>	"Fair"	
4	<b>[600..650)</b>	"Poor"	
5	<b><math>&lt; 600</math></b>	"Bad"	

#### 8.2.3.2 Data Types Pane

Within the Data Types pane you can define data types that are used for your decision, including complex types.

The screenshot shows the Data Types pane with the following structure:

- Custom Data Types** (Search bar: Search data types, Expand all, Collapse all)
- New Data Type** (highlighted)
- Requested\_Product** (Structure)
  - Marital\_Status** (string, values: "M", "D", "S")
- Applicant\_Data** (Structure)
- Post-Bureau\_Risk\_Category** (Structure)
  - Risk Category** (Risk\_Category)
  - Credit Contingency Factor** (number)
- Pre-Bureau\_Risk\_Category** (Structure)
  - Eligibility** (string, values: "Ineligible", "Eligible")

Last update: 2023-12-06

### 3.4.2 9. Getting Started with Decision Model and Notation

This lab introduces you to the deployment of an existing Decision Model and Notation (DMN) and validation of its decisions.

- Explore an existing DMN file created using IBM Business Automation Manager Canvas
- Test the DMN using IBM Business Automation Manager Canvas's Extended Services
- Deploy the existing DMN project to OpenShift
- Test the deployed DMN

#### 9.1 Examine Existing DMN Diagram

The following example describes an insurance price calculator based on an applicant's age and accident history. This is a simple decision process based on the following decision table:

**Insurance Total Price (Decision Table)**

U	Age (number)	had previous incidents (boolean)	Insurance Total Price (number)	Enter Text
1	>25	false	1000	
2	>25	true	1250	
3	[18..25]	false	2000	
4	[18..25]	true	3000	

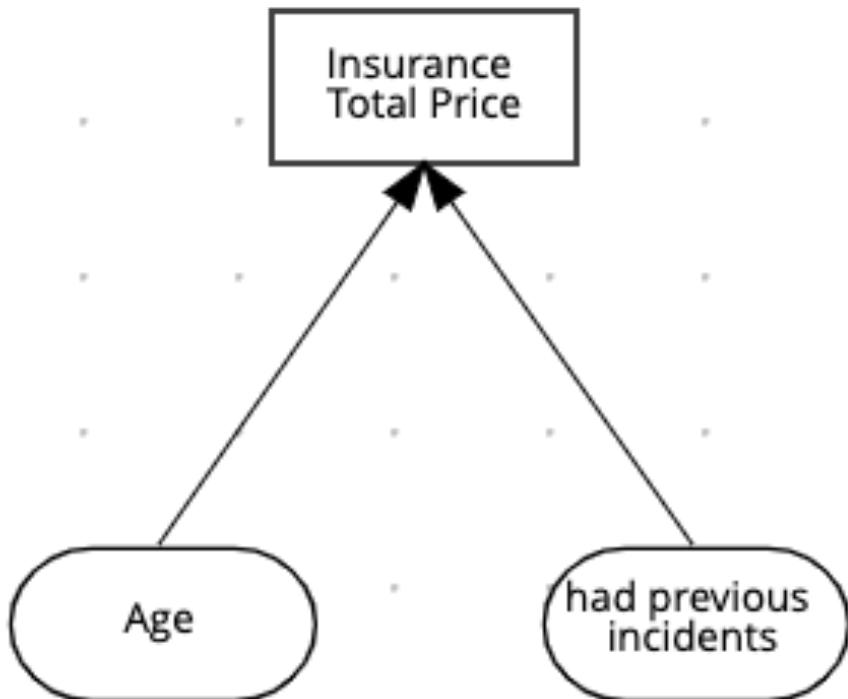
**DMN Decision Table**

The decision table was designed with using a DMN editor, this could be done in IBM Business Automation Manager Canvas, VSCode, Business Central or realistically any DMN compatible design tool. With how IBM Business Automation Open Edition 9.0 utilizes the DMN specification, we're easily able to import this diagram seamlessly in the tool that makes the most sense for our end user.

Notice that:

- The DMN decision table includes a *hit policy*, *inputs*, and *outputs*.
- Unlike the classic Drools Rules Language (DRL) based decision tables that can be created in Business Central, input and output names in DMN decision tables accept spaces. This allows for more clarity in your decisions aligning more to a business acumen.
- The conditions for the `Age` input is defined using the Friendly Enough Expression Language (FEEL).

The decision can also be represented by the following decision requirements diagram:



In this decision requirements diagram, you may note that the applicant's age and accident history are found in squared ovals, these indicate that these are the required inputs for the decision table "Insurance Total Price".

You can find the DMN component in the [DMN GitHub repository](#).

## 9.2 Import the DMN File into IBM Business Automation Manager Canvas

In this section, you can import the GitHub repository to IBM Business Automation Manager Canvas directly from the file or clone the repository, if you do the direct file.

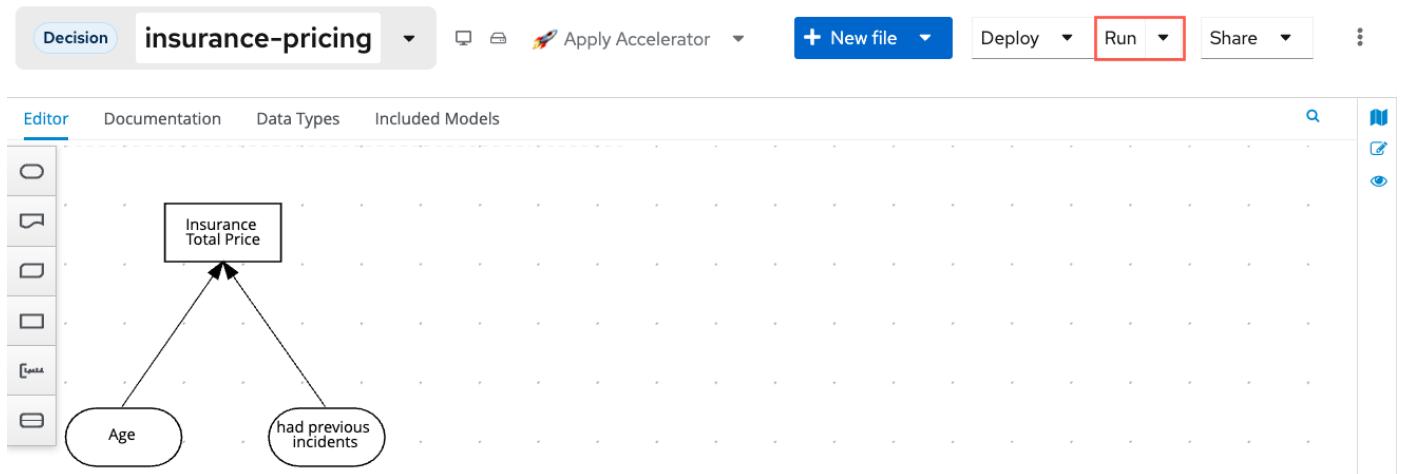
1. Open IBM Business Automation Manager Canvas by navigating to your instance of it (within the lab environments these are at <http://localhost:9090/#>). From here, you will copy the URL to the gist of the *Insurance Pricing* DMN located at [this repository location](#).

<https://raw.githubusercontent.com/timwutthenow/dmn-workshop-labs/master/policy-price/insurance-pricing.dmn>

2. With this link navigate to the [IBM Business Automation Manager Canvas](#) and under **Import From URL** paste the link from the previous step and click **Import**.

3. When the project is imported, you will see the DMN Editor with the **insurance-pricing** DMN model displayed. If you instead of pointing to a particular DMN, pointed to an entire project, any DMN/BPMN models associated with it would be able to viewed/edited within IBM Business Automation Manager Canvas.

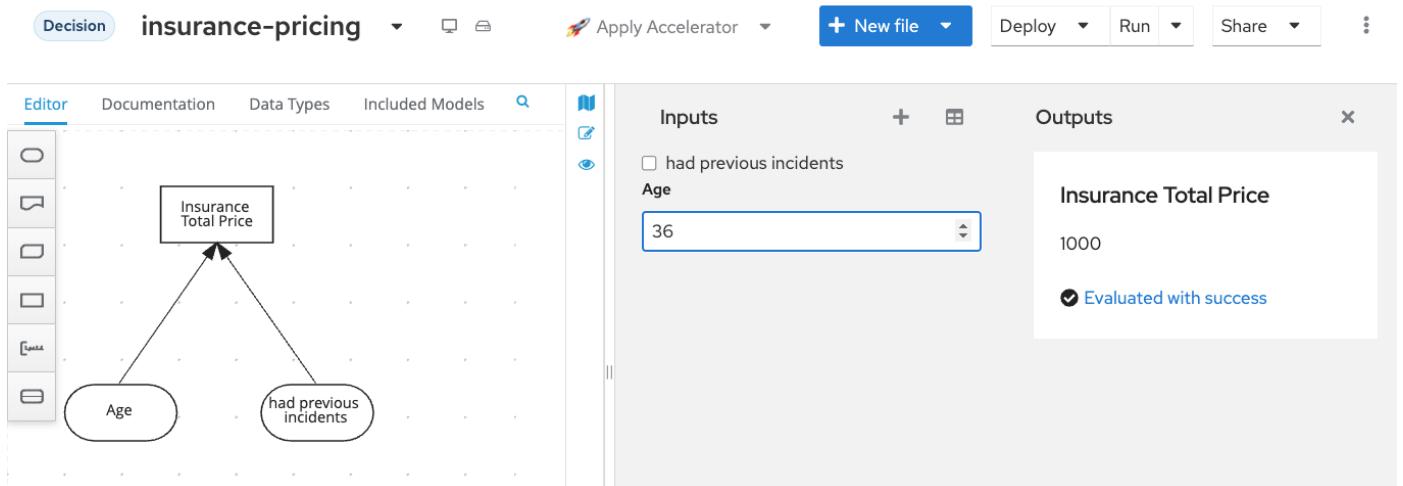
4. You can then click the **Run** button to get a local copy of this DMN running within the browser session itself. The best part about the DMN specification is that it is both a modeling specification, but also has an execution specification which allows for it to be as portable as it is.



5. This will have a section of the browser turn into a form and you can run the model right there. Modifying the checkbox based on the boolean of `had previous incidents` and set an `Age` based on the data type being a number.



Within this form you may see that if you just put an age in that the result returns (`null`). This is because the way that a typical form works when generated is that a non-checked box assumes `null`, which the rules by default do not check for null. To quickly get around this from the form you can check and uncheck the checkbox and move forward with your testing. This initializes the value, this is because the form when it sends it for initial execution has the boolean value as a null. How would you address this in the rules? *HINT:* since we're checking if the number of incidents has a checkbox or not, you can write it as `? = false` or `? = null`. Why the `?`? This is a way that DMN through the Friendly Enough Expression Language uses the column header object that you're testing to simplify checking a variable against a condition!



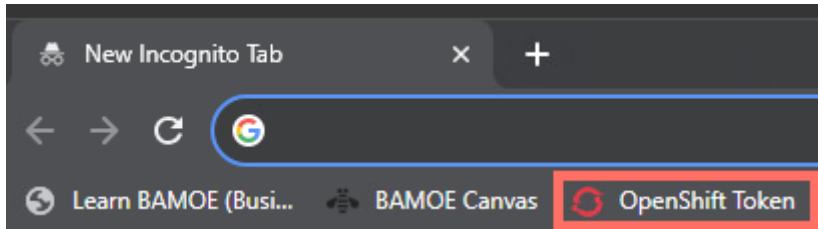
6. This service can be deployed as a sample Quarkus service to OpenShift right away as a sample service that you can invoke immediately. In the next section we will show how this is done from a development point of view.

### 9.3 Deploying this sample service to OpenShift

If you are using the provided lab environment for TechXchange, the environment itself provides access to a shared cluster for all attendees to login to OpenShift there is a bookmark in Chrome called `OpenShift Token` that will take you to a login page to create an OpenShift login token. Every user for the TechXchange environment has a created namespace to the student id that was given

to your workstation at the start of the lab (student01 through student20 with the namespace to that username with `-namespace`). The password for all of these instances is `Passw0rd`.

1. To create the OpenShift token to connect to the console at the bookmarked link **OpenShift Token** from Chrome.



2. From here you will login to OpenShift using your username and password created for the lab which is of the format `student01` through `student20` with password `Passw0rd`. If you get security issues with the link, accept them using the *Advanced* option in Chrome to proceed to the login screen.

#### 9.4 Conclusion

Congratulations, you've finished the first part of the DMN exercise. Next, you will have an intermediate level exercise that will guide you through the implementation, deployment and testing of the Vacation Days use case.

---

Last update: 2023-12-06

### 3.4.3 10. Vacation Days - Use case and project creation

In this lab you'll try out the combination of DMN decision tables with literal expressions. You will also explore a number of different FEEL constructs and expressions like, for example, ranges. Finally, you'll learn how to deploy the solution as a sample to OpenShift.

#### 10.1 Goal

- Implement a DMN model using the IBM Business Automation Manager Canvas DMN editor
- Deploy the existing DMN project to OpenShift
- Consume the DMN project using the REST API

#### 10.2 Problem Statement

In this lab we will create a decision service that determines the number of vacation days assigned to a given employee. The number of vacation days depends on age and years of service.

- Every employee receives at least 22 days.
- Additional days are provided according to the following criteria which will be broken into separate decision nodes:
  - a. Only employees younger than 18 or at least 60 years, will receive 5 days of vacation or employees with at least 30 years of service will receive an 7 extra days;
  - b. Employees with at least 30 years of service get 3 more days and also employees of age 60 or more, receive 4 extra days, on top of possible additional days already given;
  - c. If an employee has at least 15 but less than 30 years of service, 3 extra days are given. These 3 days are also provided for employees of age 45 or more. These 2 extra days can not be combined with the 5 extra days.

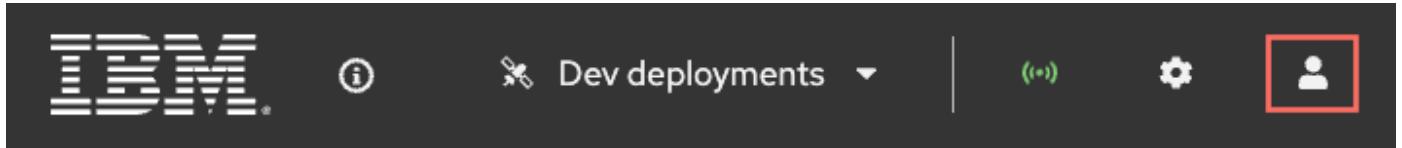
#### 10.3 Link IBM Business Automation Manager Canvas to your GitHub account and create a project in GitHub using Extended Services

When using IBM Business Automation Manager Canvas you can start from the decision model creation and work top-down to deliver your decision model. This is a feature that will enable you to create the decision model and publish a sample project to your Git provider. From here the project can be deployed in your traditional means or you can also utilize a sample service that IBM Business Automation Manager Canvas can create if you desire for *testing* purposes. To do this, you need to make sure your connection to GitHub is active with the token and from there you can jump right in!

#### 10.3.1 LINK GITHUB TO YOUR IBM BUSINESS AUTOMATION MANAGER CANVAS

In this section we will link your GitHub account to the IBM Business Automation Manager Canvas so we can easily synchronize changes in DMN with GitHub and our tooling, in this case IBM Business Automation Manager Canvas.

1. First click the User icon to connect your public GitHub account to the IBM Business Automation Manager Canvas.



2. Click the **Connect to an account** button to add a new Git provider

A screenshot of the "Connected accounts" page. It features a header "Connected accounts" with a close button "x". Below the header is a large "No accounts" icon (three people). A message reads: "Looks like you don't have any accounts connected yet". Below this, explanatory text says: "Connecting to external accounts enables Git and Cloud integrations." and "The connected accounts credentials are stored locally in this browser and are not shared with anyone." A prominent blue button at the bottom is labeled "Connect to an account".

Connected accounts x

 Looks like you don't have any accounts connected yet

Connecting to external accounts enables Git and Cloud integrations.

The connected accounts credentials are stored locally in this browser and are not shared with anyone.

[Connect to an account](#)

3. Select the card for GitHub

[◀ Back](#)[X](#)

## Select a provider

**Git** Allows integration with private repositories and provider-specific features, like GitHub Gists.

**GitHub**  
github.com

**GitHub Enterprise at IBM**  
github.ibm.com

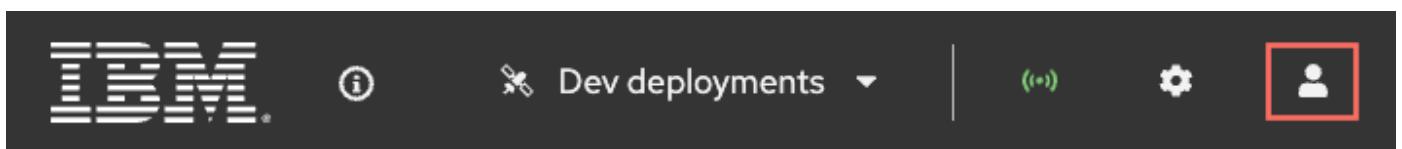
**GitLab**  
Available soon!

**Bitbucket**  
Available soon!

**Cloud** Allows Dev deployments to be created in your Cloud infrastructure.

**OpenShift**

**Kubernetes**  
Available soon!



4. Click *Generate new token* to create a new token that will be used by IBM Business Automation Manager Canvas

Settings / Developer settings

Personal access tokens

Generate new token

Revoke all

5. You can use similar properties to the token created below in the screenshot, but the main 2 to have right now are *repo* and *gist* - the others can be beneficial if you reuse this token for other purposes too, but are not required nor needed. You can change the date to never expiring or be as short as you want. Once the token is generated though, that is the only time you will see the actual token value.

- *Name*: Name your token a unique name from any previously created
- *Expiration*: This can either be a set time period, up to 1 year or never expiring
- The checkboxes you need are **repo** and **gist** to get the full benefit of IBM Business Automation Manager Canvas

**New personal access token**

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

**Note**  
Enablement

**Expiration \***  
Custom... 10/28/2022

**Select scopes**  
Scopes define the access for personal tokens. Read more about OAuth scopes.

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repos:status	Access commit status
<input type="checkbox"/> repos:deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repos:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events

<input checked="" type="checkbox"/> gist	Create gists
--	--------------

**Generate token** Cancel

6. Use the copy button that's created with the Token to use in IBM Business Automation Manager Canvas.

**Personal access tokens**

Generate new token Revoke all

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_...	Copy	Delete
-----------	------	--------

7. Return to IBM Business Automation Manager Canvas and insert the Token into the wizard.

[◀ Back](#)[✖](#)**Connect to GitHub** [github.com](https://github.com)**Personal Access Token (classic) \*** Paste your GitHub token here

Your token must include the 'repo' and 'gist' scopes.

[Generate new token](#)

The token you provide is locally stored in this browser and is never shared with anyone. [Learn more about GitHub tokens](#)

- When your token is pasted, the IBM Business Automation Manager Canvas will return a similar screen to below towards your GitHub account signifying you've connected, your GitHub user ID and some extra details.

[◀ Back](#)[✖](#)**Connect to GitHub** [github.com](https://github.com)

Successfully connected

Login timwuthenow

Name Tim Wuthenow

Email *(Empty)*

Token ghp\_\*\*\*\*\*H2lg

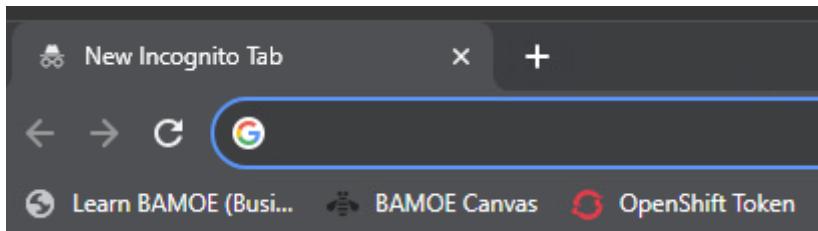
Created at 2023-08-09T01:43:40.493Z

[See connected accounts](#)**10.3.2 CREATE A DMN MODEL AND CREATE A PROJECT IN IBM BUSINESS AUTOMATION MANAGER CANVAS**

Now that our account is linked, let's go ahead and create a new DMN model and then later produce a project for it that will reside in our GitHub repositories. IBM Business Automation Manager Canvas provides an excellent way to both *push* and *pull* from your GitHub repositories. This methodology provides a true canvas over your process and decision models that sits on top of your artifact repositories as a means of seeing/creating/editing some of your most important assets for process and decision services. From IBM Business Automation Manager Canvas, you can create and edit various types of open-standards models in BPMN, DMN and PMML using the editors here. The editors are based on the standards, so if your model follows the standards, it will make a best effort representation of them to be able to edit. In theory a different DMN model written for another DMN

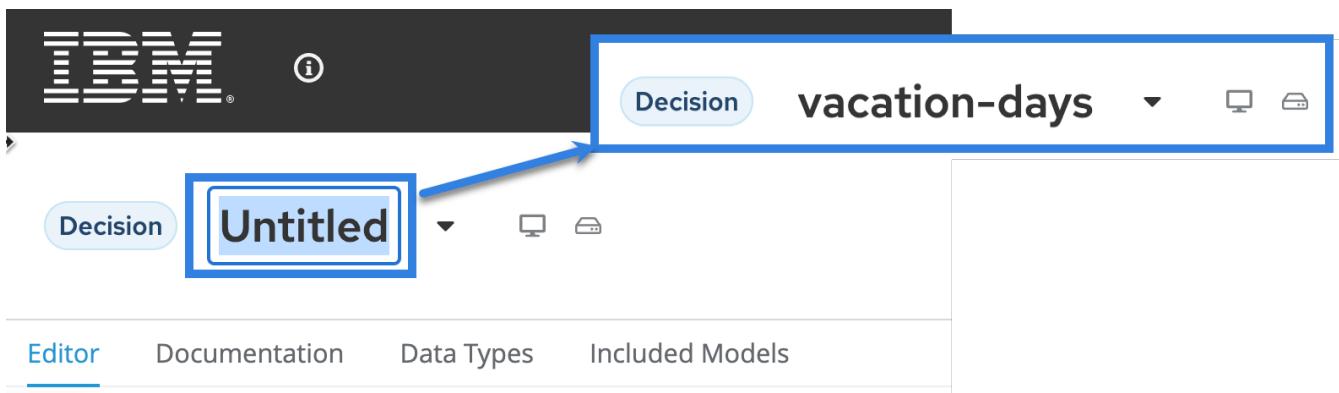
platform would be editable if it follows the specification. In this lab you will see how we can use these editors to produce a model and a project that's ready to be deployable in OpenShift.

1. Navigate to <http://localhost:9090/#> and we will start creating a DMN model for a Vacation Days Decision. This will provide some of the basics of a DMN decision and start getting you acquainted to IBM Business Automation Manager Canvas and DMN in general. This can either be done from the bookmark from Chrome on the desktop or by navigating to <https://localhost:9090/> on the provided image.



2. From the IBM Business Automation Manager Canvas landing page, you can create and edit various types of open-standards models in BPMN, DMN and PMML using the editors here. We will be creating a DMN model for this lab, so to do this, you can click **New Decision** to create a new DMN model.

3. You will now have an empty canvas and can start working on your DMN model for designing the number of vacation days decision. Change the name at the top of the model from `Untitled` to `vacation-days` or whatever you want to call it.



4. In the next section we will start to create the decision.

### 3.4.4 11. Vacation Days - Authoring Decisions

---

Let's get working on the decision model! When writing decisions with IBM Business Automation Manager Canvas, with DMN, we are able to define, model and create our decision all in one editor.

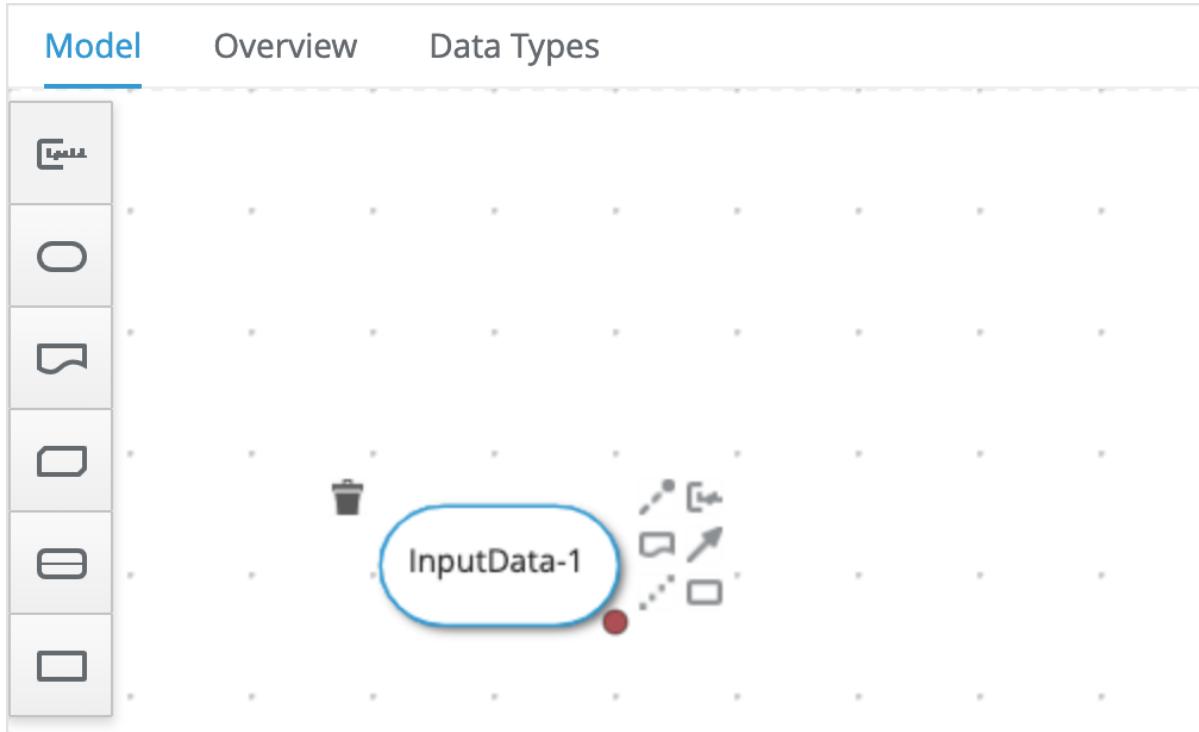
#### 11.1 Input Nodes

The problem statement describes a number of different inputs to our decision:

- **Age** of the employee
- **Years of Service** of the employee

Therefore, we should create two input nodes, one for each input:

1. Add an **Input** node to the diagram by clicking on the **Input** node icon and placing it in the DRD.



2. Double-click on the node to set the name. We will name this node `Age`.
3. With the `Age` node selected, open the property panel. Set the **data type** to `number`.

The screenshot shows the 'vacation-days' model selected in the top left. The workspace contains an oval node labeled 'Age', which is highlighted with a blue selection border. The right side of the screen features a property panel for the selected node:

- Properties** section:
  - Id**: `_2DE50728-317B-442F-A6F9-9490E695D749`
  - Description**: (empty text area)
  - Documentation Links**: (empty text area) with an **Add** button
  - Name**: `Age`
- Information item** section:
  - Data type**: `number` (with a **Manage** button)
- Styling details** section: (empty)

4. In the same way, create an **Input** node for `Years of Service`. This node should also have its **data type** set to `number`.

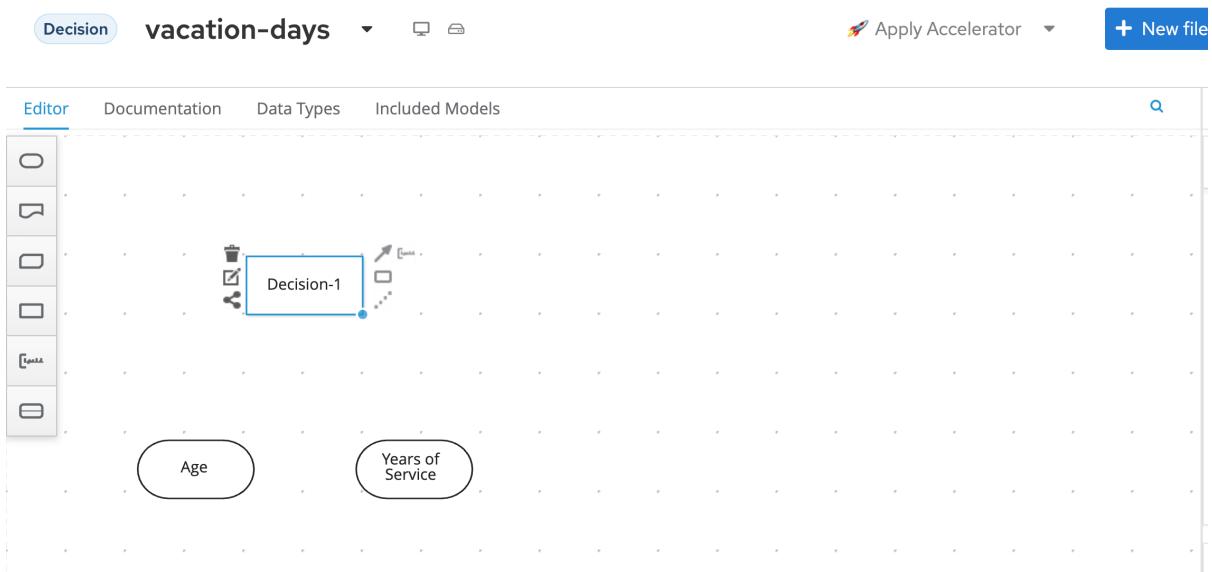
The screenshot shows the Business Central application interface. At the top, there's a navigation bar with 'Business Central' and a user icon for 'adminUser'. Below the navigation bar, the URL path is visible: 'Spaces > MySpace > vacation-days-decisions > master > vacation-days'. The main area is titled 'Decision Navigator' and contains a 'Decision Graphs' section. Under 'Decision Graphs', there's a tree view with 'vacation-days' expanded, showing 'Age' and 'Years of Service' as children. To the right of the tree view is a large, mostly empty grid area labeled 'Model'. At the bottom of the grid area, there are two circular icons: one labeled 'Age' and another labeled 'Years of Service'. The top menu bar includes standard options like 'Save', 'Delete', 'Rename', 'Copy', etc., and a 'Model' tab is currently selected.

5. The model is constantly saved in browser storage, so you don't have to save it constantly.

## 11.2 Constants

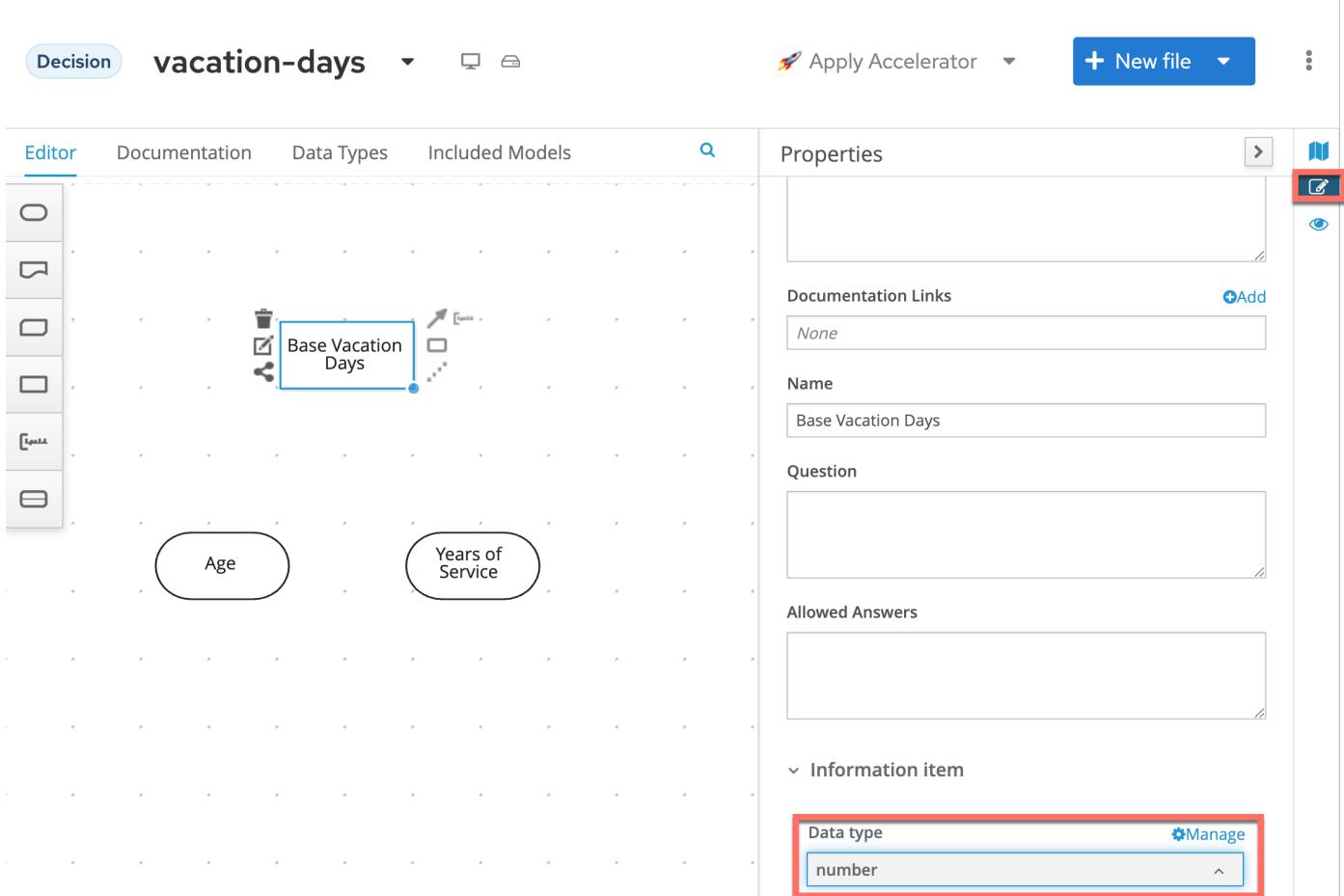
The problem statement describes that every employee receives at least 22 days. So, if no other decisions apply, an employee receives 22 days. This can be seen as a constant input value into our decision model. In DMN we can model such constant inputs with a **Decision** node with a **Literal** boxed expression that defines the constant value:

1. Add a **Decision** node to the DRD

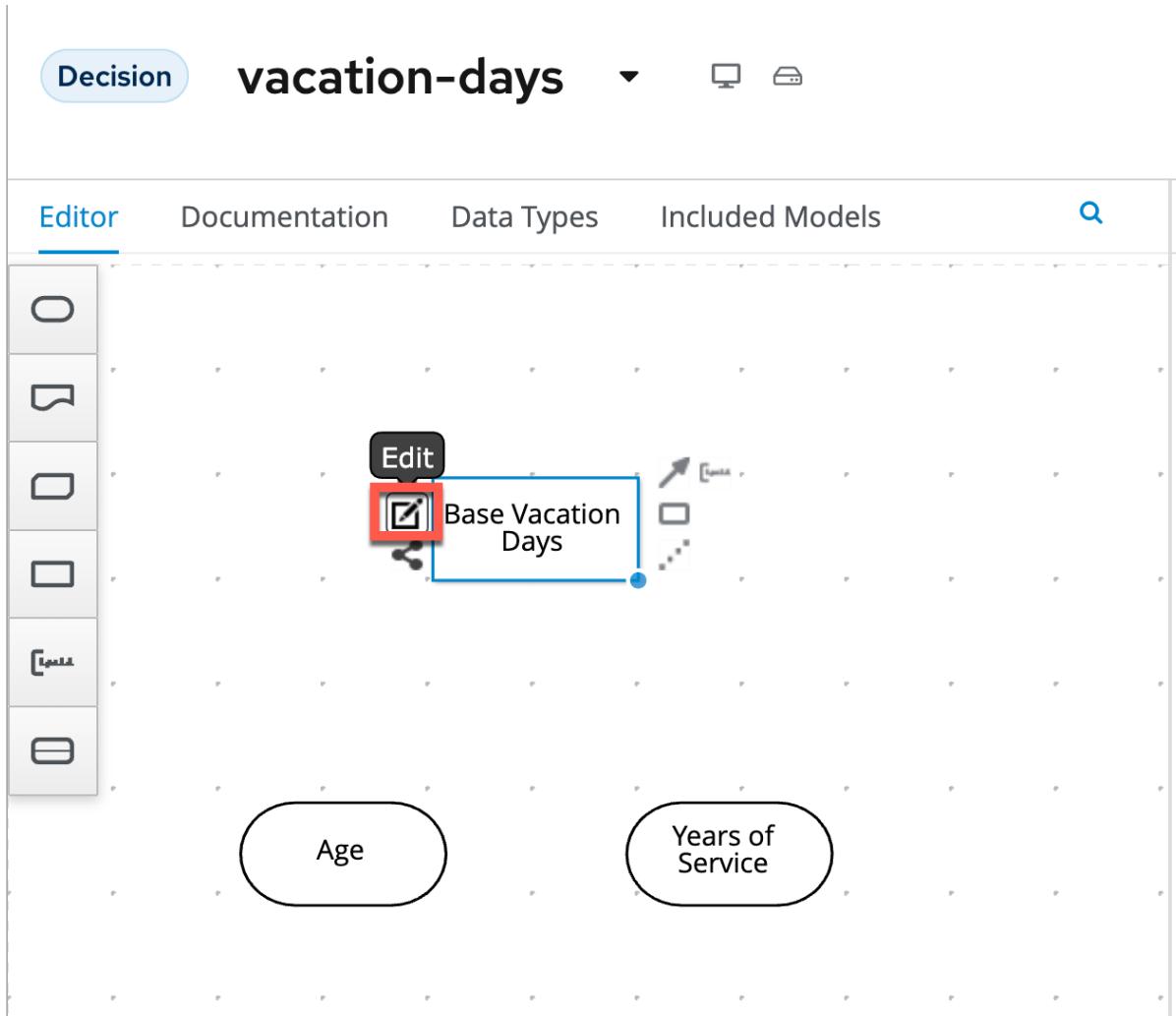


2. Give the node the name `Base Vacation Days`.

3. Click on the node to select it and open the property panel. Set the node's **data type** to `number`.



4. Click on the node and click on the **Edit** icon to open the expression editor.



5. In the expression editor, click on the box that says **Select expression** and select **Literal expression**.

Decision **vacation-days** ▾

Editor Documentation Data Types Included Models

[Back to vacation-days](#)

Base Vacation Days (<Undefined>)

Select expression

**FEEL Literal**

- Context
- Decision table
- Relation
- Invocation
- List

---

Paste

6. Simply set the **Literal Expression** to `22`, the number of base vacation days defined in the problem statement. When completed press the link to go **Back to vacation-days** to return to the main DMN diagram.

Decision **vacation-days** ▾  

---

**Editor** Documentation Data Types Included Models 

[← Back to vacation-days](#)

Base Vacation Days (*Literal*)

FEEL Literal ▾

```
= Base Vacation Days  
      (number)  
      22
```

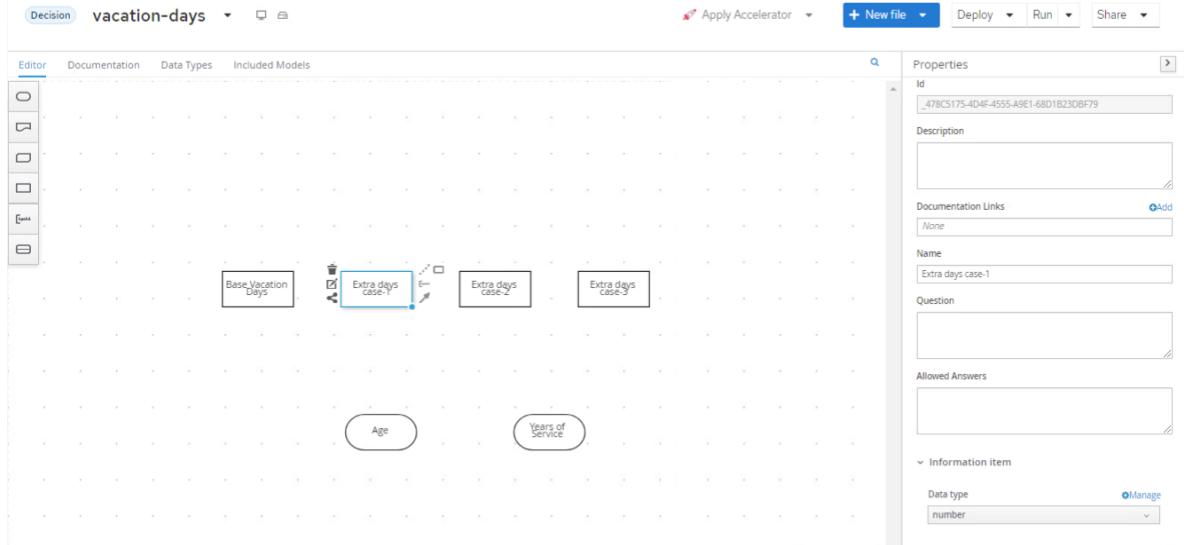
### 11.3 Decisions

The problem statement defines 3 decisions which can cause extra days to be given to employees based on various criteria. Let's simply call these decision:

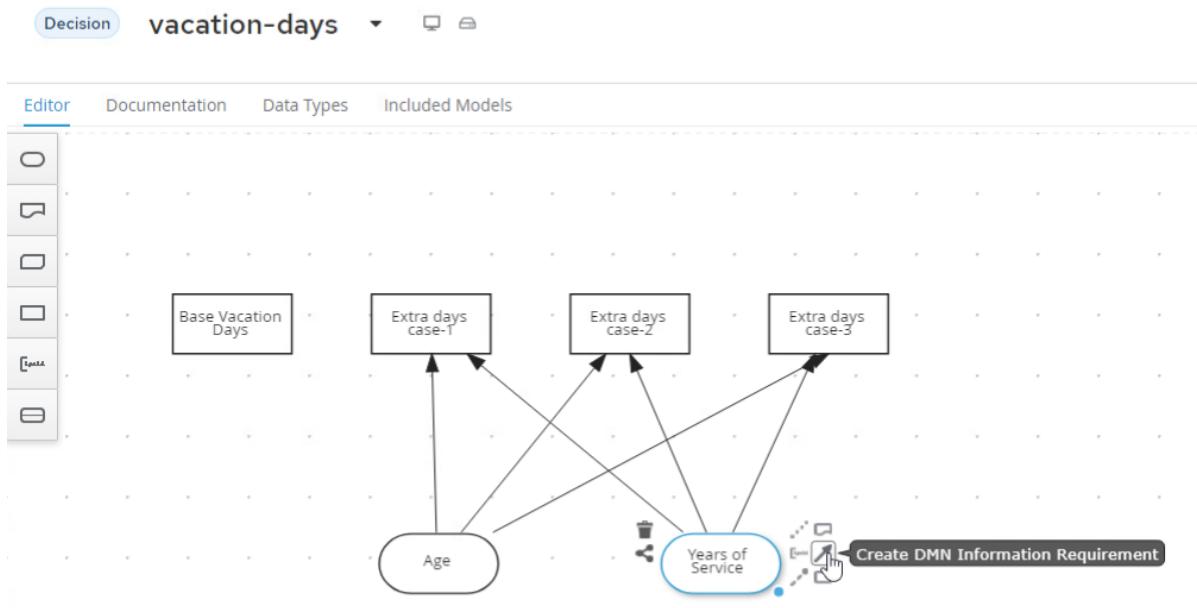
- Extra days case 1
- Extra days case 2
- Extra days case 3

Although these decisions could be implemented in a single decision node, we've decided, in order to improve maintainability of the solution and visibility of the solution, to define these decisions in 3 separate decision nodes.

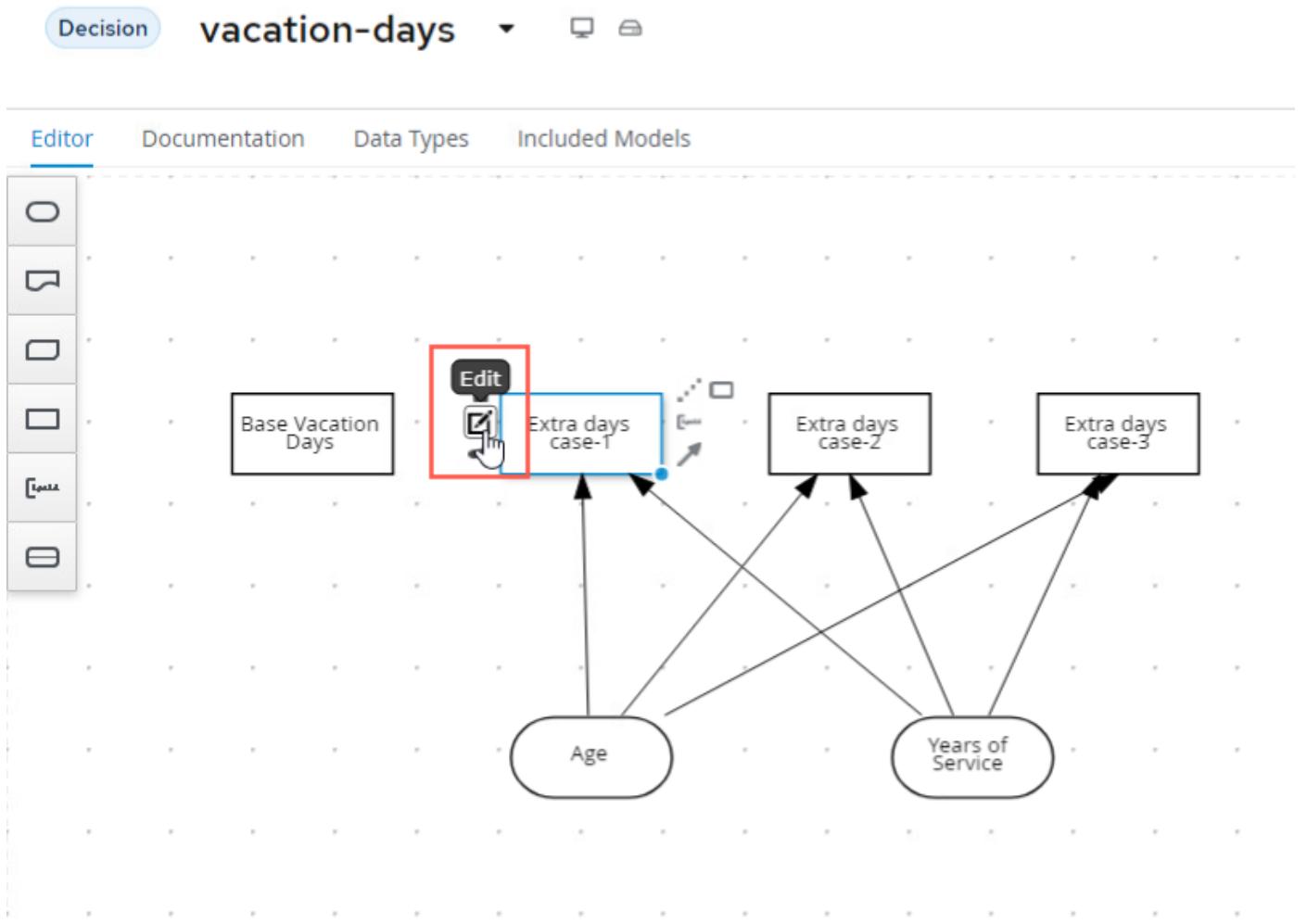
1. In your DRD, create 3 decision nodes with these given names. Set their **data types** to `number`. You can start setting one of the Decision nodes if you like as **Extra days case** with the `number` type and use the keyboard short cut with the node highlighted to copy using *Control + C* or *Command + C* then using *Control + V* or *Command + V* to copy and paste the same node two times. This will take the underlying data from the node and use it to craft your extra nodes. In my case this would produce **Extra days case**, **Extra days case-1** and **Extra days case-2** complete with the `number` data type already selected. You can do it however you feel most comfortable.



2. We need to attach both input nodes, **Age** and **Years of Service** to all 3 decision nodes. We can do this by clicking on an Input node, clicking on its arrow icon, and attaching the arrow to the Decision node. This tells the Decision node that it has a requirement of both **Age** and **Years of Service** for each respective node it connects. With DMN this explicit linkage is how decisions requirements diagrams (DRD) are crafted.



3. Select the **Extra days case 1** node and open its expression editor by clicking on the **Edit** button.



4. Select the expression **Decision Table** to create a boxed expression implemented as a decision table.

Decision **vacation-days** ▾

---

[Editor](#) [Documentation](#) [Data Types](#) [Included Models](#)

[Back to vacation-days](#)

Extra days case-1 (<Undefined>)

Select expression

**Click here**

*FEEL Literal*

{ } Context

**Decision table** **Select Decision Table**

**Relation**

f() Invocation

List

Paste

5. The first case defines 2 decisions which can be modelled with 2 rows in our decision table as such:

- employees younger than 18 or at least 60 years will receive 5 extra days, or ...
- employees with at least 30 years of service will receive 7 extra days

Decision **vacation-days** ▾

---

Editor Documentation Data Types Included Models

[Back to vacation-days](#)

Extra days case-1 (*Decision table*)

Decision table				
C>	Age (number)	Years of Service (number)	Extra days case-1 (number)	annotation-1
1	<18, >=60	-	5	
2	-	>=30	7	

6. To add new lines to your table, right click the first column and select "Insert below" under *Decision Rule* or you can click the **Plus Sign** below the 1 under the *U*.

Decision **vacation-days** ▾

---

Editor Documentation Data Types Included Models

[← Back to vacation-days](#)

Extra days case-1 (*Decision table*)

U	Age (number)	Years of Service (number)	Extra days case-1 (number)	annotation-1
1	<18, >=60	-	5	

**INPUT CLAUSE**

- + Insert left**
- + Insert right**
- + Insert**
- Delete**

---

**DECISION RULE**

- + Insert above**
- + Insert below**
- + Insert**
- Delete**

**Duplicate**

Decision **vacation-days** ▾

---

[Editor](#) [Documentation](#) [Data Types](#) [Included Models](#)

[Back to vacation-days](#)

Extra days case-1 (*Decision table*)

Decision table				
U	Age (number)	Years of Service (number)	Extra days case-1 (number)	annotation-1
1	<18, >=60	-	5	



7. Note that the **hit-policy** of the decision table is by default set to `u`, which means `Unique`. This implies that only one rule is expected to fire for a given input. In this case however, we would like to set it to `Collect Max`, as, for a given input, multiple decisions might match, but we would like to collect the output from the rule with the highest number of additional vacation days. To do this, click on the `u` in the upper-left corner of the decision table. Now, set the **Hit Policy** to `Collect` and when you click this, it will ask the **Aggregator function** to use, select `MAX` so you pull the largest value from the table.

Decision **vacation-days** ▾

---

[Editor](#) [Documentation](#) [Data Types](#) [Included Models](#)

[← Back to vacation-days](#)

Extra days case-1 (Decision table)

C	Age (number)	Years of Service (number)	Extra days case-1 (number)	annotation-1
Hit policy	Aggregator function			
UNIQUE	<None>			
FIRST	SUM			
PRIORITY	COUNT			
ANY	MIN			
<b>COLLECT</b>	<b>MAX</b>			
RULE ORDER				
OUTPUT ORDER				

8. Finally, we need to set the default result of the decision. This is the result that will be returned when none of the rules match the given input. This is done as follows:

- Select the output/result column of the decision table (*Extra days case-1*).
- Open the properties panel on the right-side of the editor. Expand the **Default output** section.
- Set the `Default output value` to `0`.

The screenshot shows the IBM Business Automation Manager Canvas interface. On the left, there is a 'vacation-days' decision table with two rows. Row 1 has inputs: Age (<18, >=60) and Years of Service (-). The output 'Extra days case-1' is 5. Row 2 has inputs: Age (-) and Years of Service (>=30). The output 'Extra days case-1' is 7. On the right, the 'Properties' panel shows the 'Id' field with a value of A357860D-CA2E-46C7-8C8C-806E1BAB4A27. The 'Default output' section is highlighted with a red box and a red arrow pointing to the 'Default output value' field, which contains the value 0.

#### 11.4 Decision Runner

Your model will be saved throughout this process, if you want to see how the decision is starting to execute. You can start to use the DMN Runner (which we could have started from the moment we added input values, but more fun with decisions in place!). To do this click **Run** just below the Black IBM banner.

This screenshot shows the same canvas environment as the previous one, but the 'Run' button in the top toolbar is highlighted with a red box. The decision table and properties panel remain the same, displaying the same data and configuration as in the first screenshot.

Now that the DMN Runner is open, you can see that the IBM Business Automation Manager Canvas created a form based on the inputs you have and is also displaying the results of the various decision nodes that we have. You will see that **Base Vacation Days** is currently displaying 22 and the rest of the decisions are (*null*) because we have no input data. If you start adding data, e.g. an Age of 60 with 5 years of service, you will see the result also update to have Extra days case-1 having 5 as the final result. Since DMN is also an execution specification with how it is written, all of the updates we make to our decisions, as we make them will be taken once the cell that was being actively changed is no longer selected. You can try this now or just leave it open while you're creating your other decisions so that you can see how your model is actively behaving based on your inputs.

**11.5 Continuining with Decisions for Use Case 2 and Use Case 3**

1. The other two decisions can be implemented in the same way. Now, implement the following two decision tables:

- Case 2:

**Decision vacation-days**

**Properties**

**Id**: \_96C1DAA2-1FEF-4846-9EEE-121C2E75420B

**Description**

**Output values**

**Default output**

**Default output value**: 0

Decision table				
C>	Age (number)	Years of Service (number)	Extra days case... (number)	annotation-1
1	-	>=30	4	
2	>=60	-	3	

- Case 3:

**Decision vacation-days**

**Properties**

**Id**: \_0D7B7C03-B837-401A-A047-9BE0A76E064F

**Description**

**Output values**

**Default output**

**Default output value**: 0

Decision table				
C>	Age (number)	Years of Service (number)	Extra days case... (number)	annotation-1
1	-	[15..30]	3	
2	>=45	-	2	

NOTE: If you want to resize the columns, you can double click the lines between them to auto-resize. More auto-resizing to column headers features are coming soon!

1. If you check the DMN Runner if you are using the same Age: 60 and Years of Service: 30 will now render a decision for each one in the table and you can adjust to execute other decisions.

The screenshot shows the DMN Runner interface with the following details:

- Inputs:** Age: 60, Years of Service: 30.
- Outputs:** Four separate cards representing decision results:
  - Extra days case-2:** Value: 4, Status: Evaluated with success.
  - Extra days case-1:** Value: 7, Status: Evaluated with success.
  - Base Vacation Days:** Value: 22, Status: Evaluated with success.
  - Extra days case-3:** Value: 3, Status: Evaluated with success.

### 11.6 Total Vacation Days

The total vacation days needs to be determined from the base vacation days and the decisions taken by our 3 decision nodes. As such, we need to create a new Decision node, which takes the output of our 4 Decision nodes (3 decision tables and a literal expression) as input and determines the final output. To do this, we need to:

1. Create a new Decision node in the model. Give the node the name `Total Vacation Days` and set its **data type** to `number`.

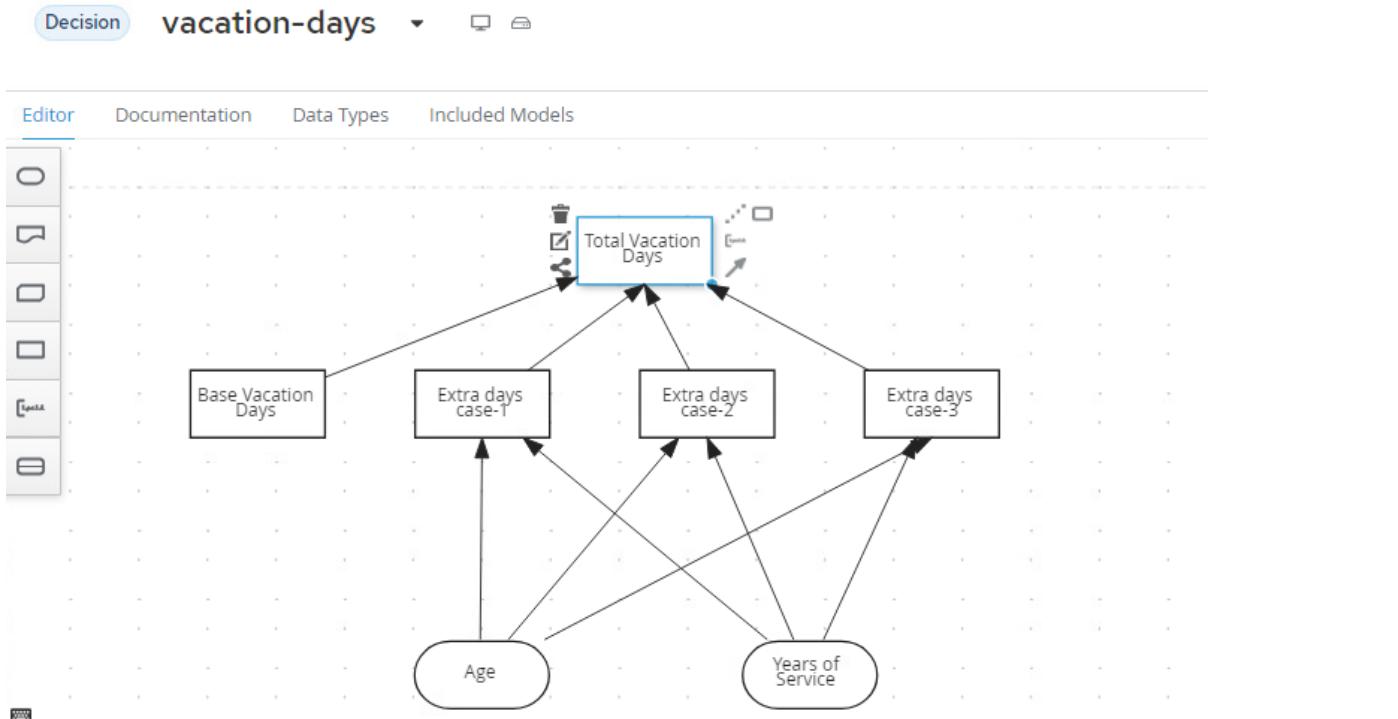
**Decision vacation-days**

**Total Vacation Days (Literal)**

```
FEEL Literal
=
Total Vacation Days
(number)

Base Vacation Days +
max( Extra days case-1, Extra days case-3 ) +
Extra days case-2
```

2. Connect the 4 existing Decision nodes to the node. This defines that the output of these nodes will be the input of the next node. Without putting a line connecting `Age` or `Years of Service` connected to `Total Vacation Days`, this data will not be available to this decision, not that they are needed here, but just an overall note of how DMN works!



3. Click on the `Total Vacation Days` node and click on **Edit** to open the expression editor. Configure the expression as a literal expression.

4. We need to configure the following logic:

- Everyone gets the `Base Vacation Days`.
- If both case 1 and case 3 add extra days, only the extra days of one of this decision is added. So, in that case we take the maximum.
- If case 2 adds extra days, add them to the total.

5. The above logic can be implemented with the following FEEL expression - note that these are using the values that are produced by the Decision Nodes names and match the case *exactly*:

Decision vacation-days

[Editor](#) Documentation Data Types Included Models

[Back to vacation-days](#)

Total Vacation Days (*Literal*)

FEEL Literal ▾

```
= Total Vacation Days
  (number)

Base Vacation Days +
max( Extra days case-1, Extra days case-3) +
Extra days case-2
```

```
Base Vacation Days +
max( Extra days case-1, Extra days case-3) +
Extra days case-2
```

6. Using your DMN Runner you can test your decision exactly as it sits today and return a Total Days based on the sample employee from before or any other changes.

IBM

Decision vacation-days

Apply Accelerator ▾ + New file ▾ Deploy ▾ Run ▾ Share ▾

Editor Documentation Data Types Included Models

Inputs

- Age: 60
- Years of Service: 30

Outputs

- Extra days case-2: 4, Evaluated with success
- Extra days case-1: 7, Evaluated with success
- Total Vacation Days: 33, Evaluated with success
- Base Vacation Days: 22, Evaluated with success
- Extra days case-3: 3, Evaluated with success

7. Your model is now ready for consumption. There are a few ways to explore how to use it which we will cover in the next section!

Last update: 2023-12-06

### 3.4.5 12. Vacation Days - Consuming Decisions

---

There are several ways to consume decisions, but the first way we will explore is through the utilization of the Canvas provided methods, but there will also be methods showing how to do it with IBM Business Automation Open Edition 8.0 provided as informational and not required for these exercises.

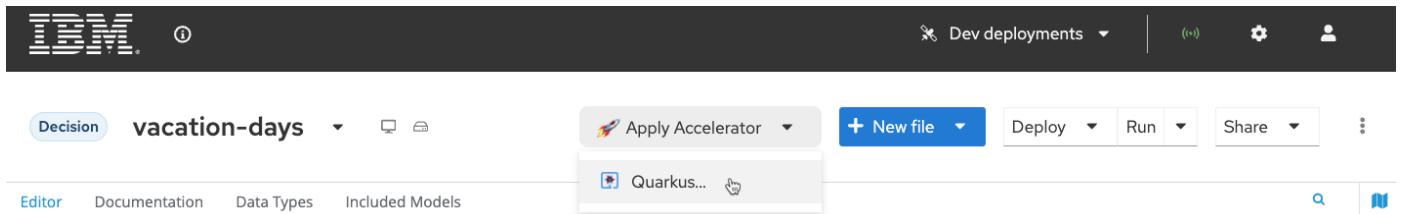
#### **12.1 Deploying and testing the Decision Service**

With our decision model completed, we can now package our DMN model in a Kogito service. This can be done using the Accelerators included with IBM Business Automation Manager Canvas. These accelerators will quickly take the DMN file and start building the basis of the project as a IBM Business Automation Open Edition 9.0 compatible decision service.

## 12.2 Applying the Accelerator

To take this decision service from a standalone DMN model to a full Kogito architected decision, you will use the accelerator by clicking the button at the top of the screen. To do so follow the instructions and then we will synchronize the project to your GitHub account and also be capable being deployed as an OpenShift service.

1. At the top of the DMN model page for `vacation-days`, click the button **Apply Accelerator** and select **Quarkus....**



2. If you want to learn more about the accelerator, you can click the GitHub link and change the branch to the `9.0.0-quarkus-full` to learn more about it. Afterwards just click **Apply** to restructure your project into a Kogito service.

## Quarkus Accelerator

An Accelerator is a template. Applying it will move your current files according to the Accelerator specifications and create a new commit for it.

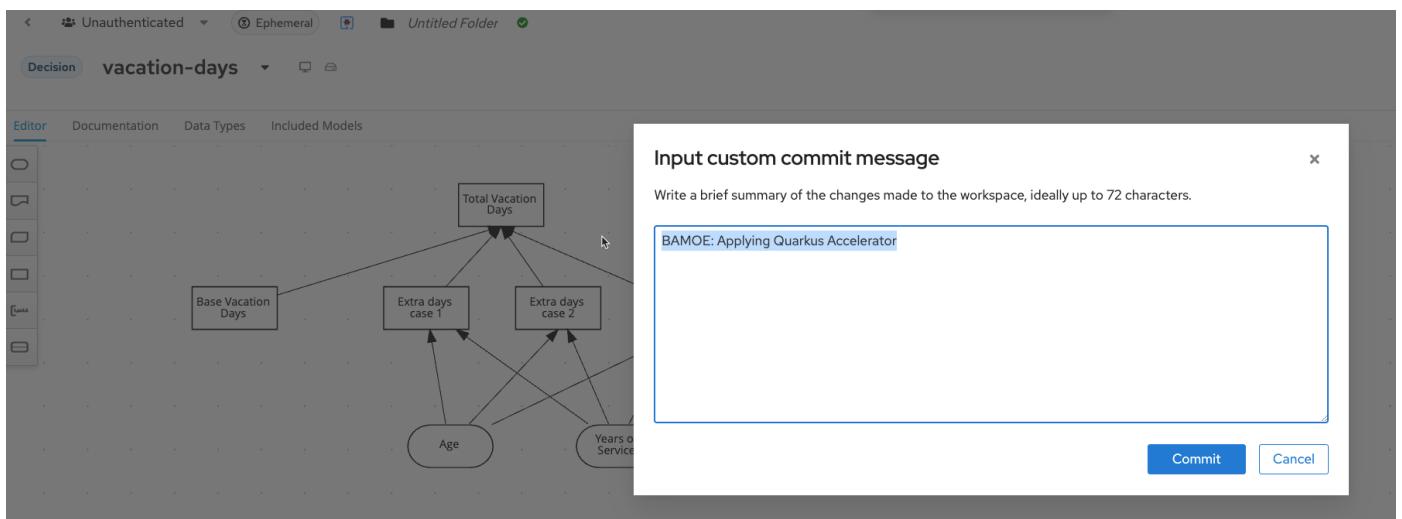
This Accelerator is hosted at <https://github.com/ibm/bamoe-canvas-quarkus-accelerator>  
Git ref: @9.0.0-quarkus-full

Decisions (.dmn) will be moved to:	<code>src/main/resources/dmn</code>
Score cards (.pmml) will moved to:	<code>src/main/resources/dmn</code>
Workflows (.bpmn, .bpmn2) will be moved to:	<code>src/main/resources/bpmn</code>
Other files will be moved to:	<code>src/main/resources/others</code>

*This action is permanent. Any changes made after applying an Accelerator may result in your files being in different directories.*

**Apply**   **Cancel**

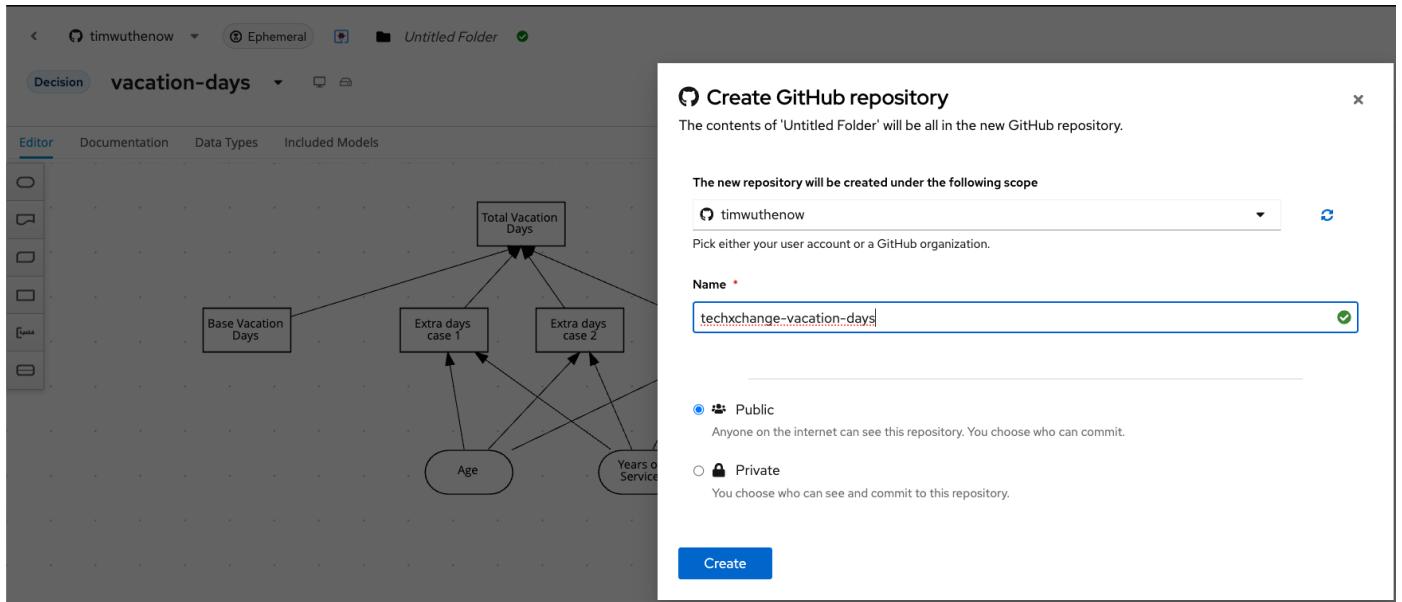
3. The wizard will come with a pop-up asking for a commit message for the change since this will create a Git project. You can use the default message or put whatever you would like, to do so press **Commit**.



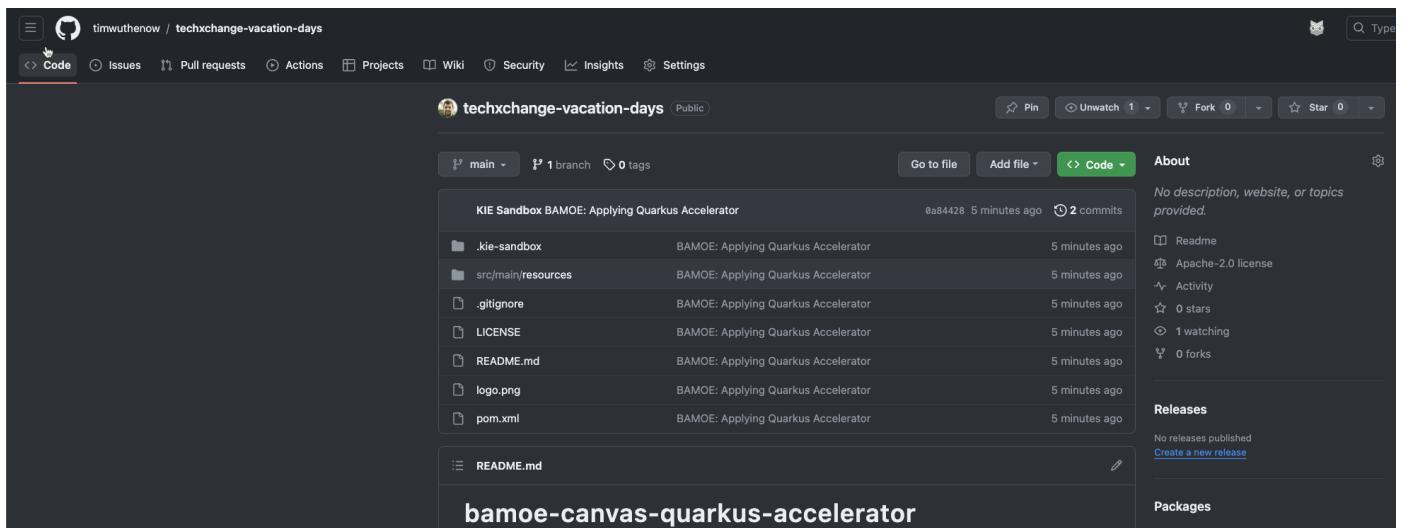
4. To create the project in Git, click **Share**, use your GitHub tokened ID.

5. Once you select your ID, a new option **Create GitHub repository...** is now available, select this.

6. The repository you create, can be anything you want, for the purposes of this lab, it will be called **techxchange-vacation-days**.



7. When you create the repository, a message will open at the top with a link to the Git repository that was created. In the case of the example it is at [this example repository](#).



### 12.3 Deploying to OpenShift

Now that this repository is created, you can work on it locally or deploy it to OpenShift. For this lab, we will choose OpenShift since we have a cluster ready to use for TechXchange or you can use the [Red Hat OpenShift Sandbox](#) to provision a small environment to try it out on your own.

1. Login to get an OpenShift Token using the bookmark on Chrome. This will be used to connect IBM Business Automation Manager Canvas to OpenShift and deploy a sample development environment of the decision service.

The screenshot shows a browser window with the URL `localhost:9090/#/a411e146-8c6f-4ac7-892a-3a5d10887f25/file/src/main/resources/dmn/call-center-decisions-final.dmn`. The tab bar has several items: 'Learning IBM Busin...', 'BAMOE Canvas', 'OpenShift Token' (highlighted with a red box), 'Business Central', 'Weather App', 'EmployeeOnboardi...', 'payment test', 'KIE Server Swagger', 'Pricing App', and 'OpenShift Console'. A notification message in the top right corner says 'GitHub repository created.' with a link to <https://github.com/timwutheNow/techxchange>.

2. In the login screen use your username and password - this will be some combination of `student01` to `student20` with password `Passw0rd`. The username is all lowercase. This will

The screenshot shows a 'Log in to your account' form. It has two input fields: 'Username \*' containing 'student01' and 'Password \*' containing redacted text. Below the fields is a large blue 'Log in' button. To the right, there is a dark panel with the Red Hat OpenShift logo and the text 'Welcome to Red Hat OpenShift'.

3. Click the **Display Token** link and switch back to the tab with IBM Business Automation Manager Canvas open on it. You will use the token contents from the OpenShift token page shortly by clicking on **Dev deployments** and then selecting the pull down **Select authentication** and then **Connect to an account**.

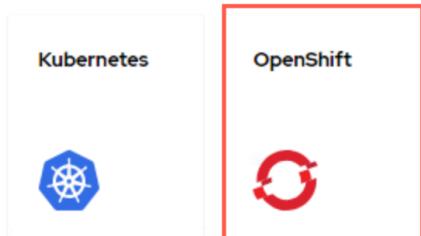
The screenshot shows the 'Create' page of the IBM Business Automation Manager Canvas. At the top, there is a dropdown menu labeled 'Dev deployments' with a red number '1' next to it. Below the menu, a modal window titled 'Select authentication' has a red number '2' next to it. Inside the modal, there is a link '+ Connect to an account...' which is also highlighted with a red box. A red number '3' is placed next to this link.

4. On the next wizard, select the **OpenShift Tile** to connect to an OpenShift instance.

## Select a provider



**Cloud** Allows Dev deployments to be created in your Cloud infrastructure.



5. From here you will use the information from the OpenShift token page. Your namespace will be your `student##-namespace`, so if your username is student07, then you would connect to `student07-namespace`. The token and the console locations are both a part of the token as well. You will see a checkbox to Insecurely disable TLS validation, check it as this is a self contained environment with certificates not present. Press **Connect** when the form is completed.

Your API token is  
sha256~[REDACTED]

Log in with this token

Connect to OpenSh

oc login -token=sha256~ -server=https://api.ocp.ibm.edu:6443

Use this token directly against the API

curl -H "Authorization: Bearer sha256~" "https://api.ocp.ibm.edu:6443/apis/user.openshift.io/v1/users/~"

Configure a new Developer

Namespace (project) \* student20-namespace

Host \* https://api.ocp.ibm.edu:6443

Token \* sha256~

Insecurely disable TLS certificate validation

Checking this option will insecurely disable TLS certificate verification for this account. This is an alternative to not having to deal with the browser's restrictions when your cluster is behind an HTTPS endpoint with a self-signed certificate. Please be advised that the use of self-signed certificates is a weaker form of security, so consider contacting your cluster admins to use a trusted certificate. For more information, refer to <https://cwe.mitre.org/data/definitions/295.html>

Connect

6. As long as it was successful to connect, you will be greeted with a Connect to OpenShift successful modal, click **Continue**.

**Connect to OpenShift**

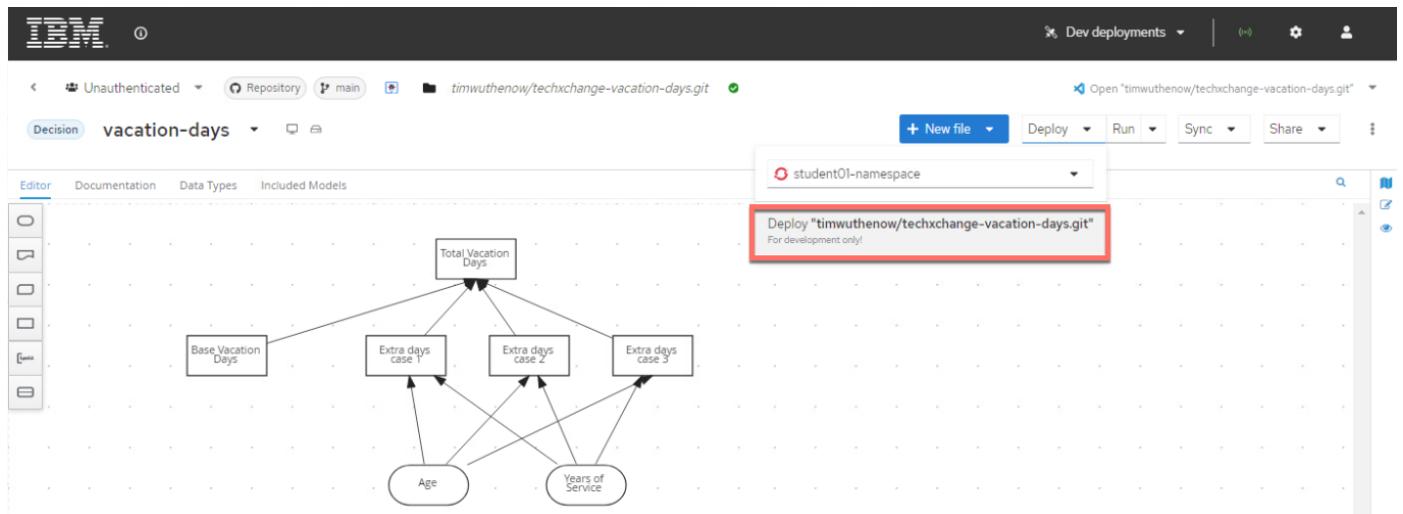
---

Successfully connected

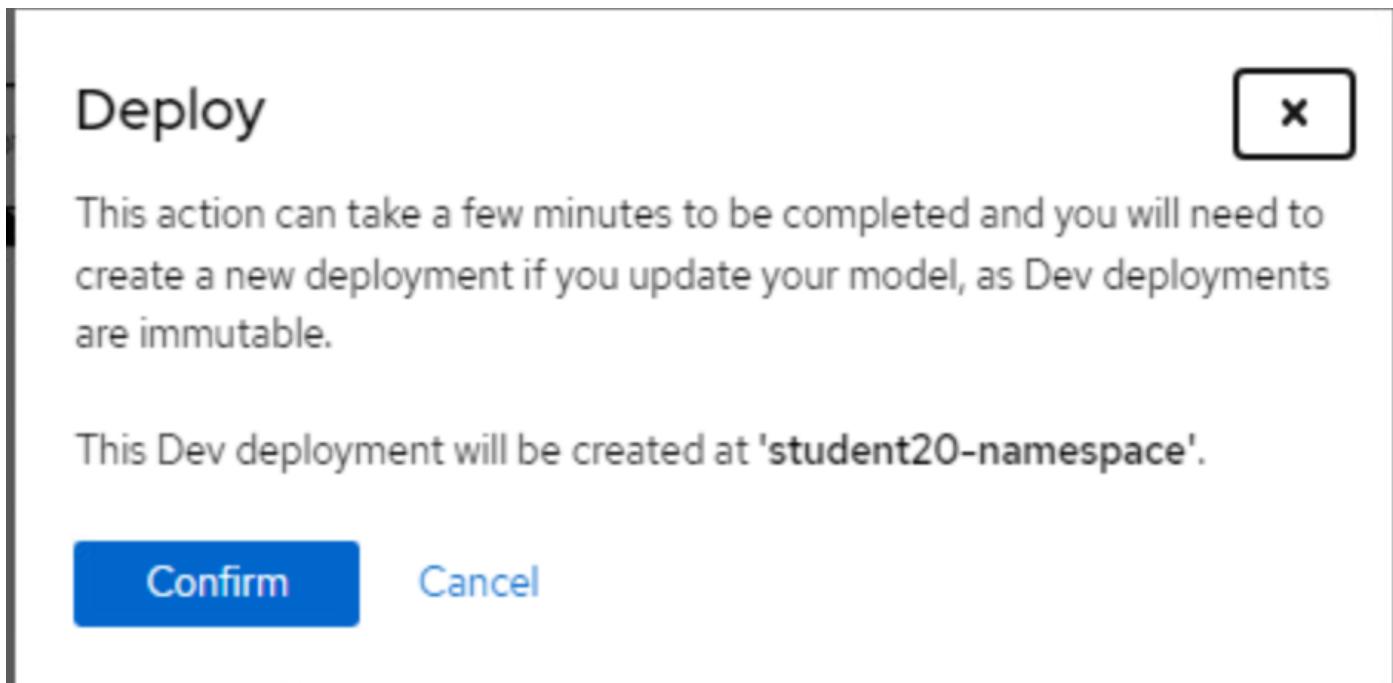
Namespace student20-namespace  
Host https://api.ocp.ibm.edu:6443  
Token sha2\*\*\*\*\*vNZw  
Created at 2023-08-30T19:39:57.425Z  
TLS Certificate Verification Disabled

**Continue**

7. Now you can press the **Deploy** button and have the capability to select the namespace that you connected to and deploy a sample of your service to it.



8. From here a modal will pop up explaining that the deployment can take a few minutes and it is only intended for development, etc. Click **Confirm** on this modal and your service will be deployed.



9. This process will take several minutes to complete.

10. The pulldown with deploy will refresh every 30 seconds, when the service is available, it will reflect as such. The way that the deployments from these services work are is that they are completely immutable and unaware of one another, so you could have all versions deployed as you make changes to validate how you want to eventually promote your DMN model to later environments. So if you were to deploy it again, there would be multiple different versions of it deployed. Canvas can manage these deployment samples, which can later be removed when you're done with them if you choose. If you click the green check marked box, it will take you to the service.

11. When you access the service with a form similar to the one you had in IBM Business Automation Manager Canvas and from that page you can explore it more using the kebab icon in the top right to access the Swagger-UI page to get access to the auto-generated DMN service's API page. The first post can be used to execute the decision itself.

12. This completes this lab as you can see how the deployment to OpenShift works with IBM Business Automation Manager Canvas!

#### 12.4 Deploying the Decision Service in KIE Server (Not for TechXchange)

This section is not needed for TechXchange, but is just to provide some extra ways to showcase now you can deploy your decision. With our decision model completed, we can now package our DMN model in a Deployment Unit (KJAR) and deploy it on the Execution Server. To do this:

1. In the bread-crumb navigation in the upper-left corner, click on `vacation-days-decisions` to go back to the project's Library View.
2. Click on the **Deploy** button in the upper-right corner of the screen. This will package our DMN mode in a Deployment Unit (KJAR) and deploy it onto the Execution Server (KIE-Server).
3. Go to the **Execution Servers** perspective by clicking on "Menu → Deploy → Execution Servers". You will see the **Deployment Unit** deployed on the Execution Server.

#### 12.5 Testing DMN Solution

In this section, you will test the DMN solution with Execution Server's Swagger interface and via Java KIE Client API.

#### 12.6 Testing the solution via REST API

In this section, you will test the DMN solution with KIE Server's Swagger interface.

The Swagger interface provides the description and documentation of the Execution Server's RESTful API. At the same time, it allows the APIs to be called from the UI. This enables developers and users to quickly test, in this case, a deployed DMN Service.

1. Navigate to [KIE Server](#)
2. Locate the **DMN Models** section. The DMN API provides the DMN model as a RESTful resources, which accepts 2 operations:
  - a. `GET` : Retrieves the DMN model.
  - b. `POST` : Evaluates the decisions for a given input.
3. Expand the `GET` operation by clicking on it.
4. Click on the **Try it out** button.
5. Set the **containerId** field to `vacation-days-decisions` and set the **Response content type** to `application/json` and click on **Execute**

The screenshot shows the KIE Server Swagger UI for the `/server/containers/{containerId}/dmn` endpoint. The `containerId` parameter is set to `call-centre-decisions_1.0.0`. The `Execute` button is highlighted in blue. The `Responses` section shows a dropdown for `application/json`.

6. If requested, provide the username and password of your **Business Central** and **KIE-Server** user.
7. The response will be the model-description of your DMN model.

Next, we will evaluate our model with some input data. We need to provide our model with the **age** of an employee and the number of **years of service**. Let's try a number of different values to test our decisions.

1. Expand the `POST` operation and click on the **Try it out** button
2. Set the **containerId** field to `vacation-days-decisions`. Set the **Parameter content type** and **Response content type** fields to `application/json`.
3. Pass the following request to lookup the number of vacation days for an employee of 16 years old with 1 year of service (note that the namespace of your model is probably different as it is generated. You can lookup the namespace of your model in the response/result of the `GET` operation you executed earlier, which returned the model description).

```
{ "dmn-context":{ "Age":16, "Years of Service":1 } }
```

1. Click on **Execute**. The result value of the `Total Vacation Days` should be 27.
2. Test the service with a number of other values. See the following table for some sample values and expected output.

Age	Years of Service	Total Vacation Days
16	1	27
25	5	22
44	20	24
44	30	30
50	20	24
50	30	30
60	20	30

## 12.7 Using the KIE Java Client (if you were using IBM Business Automation Open Edition 8.0)

IBM Business Automation Open Edition 8.0 provides a KIE Java Client API that allows the user to interact with the KIE-Server from a Java client using a higher level API. It abstracts the data marshalling and unmarshalling and the creation and execution of the RESTful commands from the developer, allowing him/her to focus on developing business logic.

In this section we will create a simple Java client for our DMN model.

**IMPORTANT:** If your KIE Server is exposed via https you need to configure the `javax.net.ssl.trustStore` and `javax.net.ssl.trustStorePassword` in the Java client code using the Remote Java API. If not, you may get a `rest.NoEndpointFoundException`. For more information check [this solution](#) Red Hat's knowledge base.

1. Create a new Maven Java JAR project in your favourite IDE (e.g. IntelliJ, Eclipse, Visual Studio Code).
2. Add the following dependency to your project:

```
<dependency>
  <groupId>org.kie.server</groupId>
  <artifactId>kie-server-client</artifactId>
  <version>7.67.2.Final-redhat-00006</version>
```

```
<scope>compile</scope>
</dependency>
```

1. Create a Java package in your `src/main/java` folder with the name `org.kie.dmn.lab`.
2. In the package you've just created, create a Java class called `Main.java`.
3. Add a `public static void main(String[] args)` method to your main class.
4. Before we implement our method, we first define a number of constants that we will need when implementing our method (note that the values of your constants can be different depending on your environment, model namespace, etc.):

```
private static final String KIE_SERVER_URL = "http://localhost:8080/kie-server/services/rest/server";
private static final String CONTAINER_ID = "vacation-days-decisions";
private static final String USERNAME = "bamAdmin";
private static final String PASSWORD = "ibmpam1!";
```

!!!  INFO: If you're using the Linux environment on Skytap use the following.

```
~~~java
private static final String KIE_SERVER_URL = "http://localhost:8080/kie-server/services/rest/server";
private static final String CONTAINER_ID = "vacation-days-decisions";
private static final String USERNAME = "pamadmin";
private static final String PASSWORD = "pamadmin";
~~~
```

5. KIE-Server client API classes can mostly be retrieved from the `KieServicesFactory` class. We first need to create a `KieServicesConfiguration` instance that will hold our credentials and defines how we want our client to communicate with the server:

```
KieServicesConfiguration kieServicesConfig = KieServicesFactory.newRestConfiguration(KIE_SERVER_URL, new EnteredCredentialsProvider(USERNAME, PASSWORD));
```

6. Next, we create the `KieServicesClient`:

```
KieServicesClient kieServicesClient = KieServicesFactory.newKieServicesClient(kieServicesConfig);
```

7. From this client we retrieve our `DMNServicesClient`:

```
DMNServicesClient dmnServicesClient = kieServicesClient.getServicesClient(DMNServicesClient.class);
```

8. To pass the input values to our model to the Execution Server, we need to create a `DMNContext`:

```
DMNContext dmnContext = dmnServicesClient.newContext();
dmnContext.set("Age", 16); dmnContext.set("Years of Service", 1);
```

9. We now have defined all the required instances needed to send a DMN evaluation request to the server:

```
ServiceResponse<DMNResult> dmnResultResponse = dmnServicesClient.evaluateAll(CONTAINER_ID, dmnContext);
```

10. Finally we can retrieve the DMN evaluation result and print it in the console:

```
DMNDecisionResult decisionResult = dmnResultResponse.getResult().getDecisionResultByName("Total Vacation Days"); System.out.println("Total vacation days: " +
decisionResult.getResult());
```

11. Compile your project and run it. Observe the output in the console, which should say: **Total vacation days: 27**

The complete project can be found [here](#)

```
package org.kie.dmn.lab;

import org.kie.api.builder.KieScannerFactoryService;
import org.kie.api.internal.weaver.KieWeaverService;
import org.kie.dmn.api.core.DMNContext;
import org.kie.dmn.api.core.DMNDecisionResult;
import org.kie.dmn.api.core.DMNRResult;
import org.kie.server.api.model.ServiceResponse;
import org.kie.server.client.CredentialsProvider;
import org.kie.server.client.DMNServicesClient;
import org.kie.server.client.KieServicesClient;
import org.kie.server.client.KieServicesConfiguration;
import org.kie.server.client.KieServicesFactory;
import org.kie.server.client.credentials.EnteredCredentialsProvider;

/**
 * Vacation Days DMN Client
 */
public class Main {

    private static final String KIE_SERVER_URL = "http://localhost:8080/kie-server/services/rest/server";
    private static final String CONTAINER_ID = "vacation-days-decisions";
```

```
private static final String USERNAME = "bamAdmin";

private static final String PASSWORD = "ibmpam1!";
// Comment out the above 2 lines if using the Skytap image and uncomment the two below to use those logins
// private static final String USERNAME = "pamadmin";
// private static final String PASSWORD = "pamadmin";

public static void main(String[] args) {
    CredentialsProvider credentialsProvider = new EnteredCredentialsProvider(USERNAME, PASSWORD);
    KieServicesConfiguration kieServicesConfig = KieServicesFactory.newRestConfiguration(KIE_SERVER_URL, credentialsProvider);
    KieServicesClient kieServicesClient = KieServicesFactory.newKieServicesClient(kieServicesConfig);

    DMNServicesClient dmnServicesClient = kieServicesClient.getServicesClient(DMNServicesClient.class);

    DMNContext dmnContext = dmnServicesClient.newContext();
    dmnContext.set("Age", 16);
    dmnContext.set("Years of Service", 1);

    ServiceResponse<DMNResult> dmnResultResponse = dmnServicesClient.evaluateAll(CONTAINER_ID, dmnContext);

    DMNDecisionResult decisionResult = dmnResultResponse.getResult().getDecisionResultByName("Total Vacation Days");
    System.out.println("Total vacation days: " + decisionResult.getResult());
}
}
```

---

Last update: 2023-12-06

### 3.4.6 13. Call Centre - Intro and Use Case

This is an advanced Decision Model & Notation lab that introduces DMN Decision Services, Relations, nested boxed expressions, etc. It also explores a number of different FEEL constructs and expressions like, for example, `list contains`.

#### 13.1 Goals

- Implement a DMN model using the IBM Business Automation Manager Canvas DMN editor
- Deploy the existing DMN project to OpenShift

#### 13.2 Problem Statement

In this lab we will create a decision that determines if a call-center can take an incoming call. Whether a call will be accepted by a certain office depends on:

- The office accepts the call.
- There are employees currently available at the office.

Whether the office can accept a call depends on:

- whether the phone number has been banned
- the purpose of the phone call ("help" or "objection").

### 13.3 Create a new DMN Decision

Similar to the first lab, we're going to use IBM Business Automation Manager Canvas to define and deploy a DMN decision model, we first need to create a new model:

#### 1. Navigate to [IBM Business Automation Manager Canvas](#)

- From the IBM Business Automation Manager Canvas landing page, you can create and edit various types of open-standards models in BPMN, DMN and PMML using the editors here. We will be creating a DMN model for this lab, so to do this, you can click **New Decision** to create a new DMN model.

**Create**

**Decision**

New Decision (highlighted)

**Import**

From URL More options... Upload

Import a Git repository, a GitHub Gist, Bitbucket Snippet, or any other file URL.

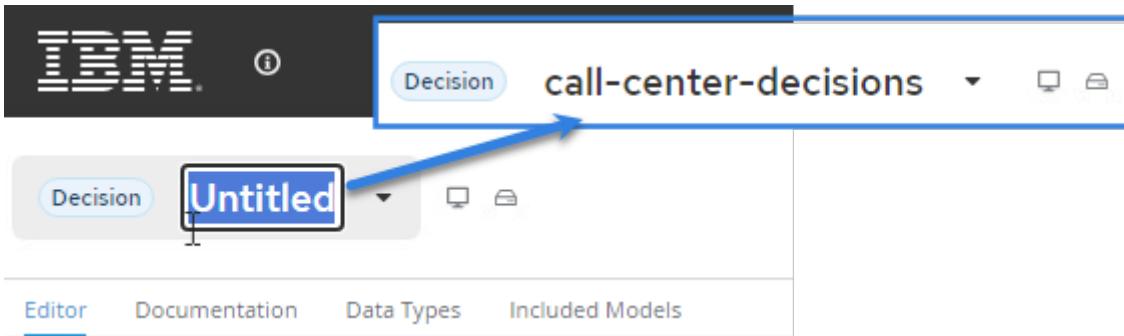
URL

Drag & drop files and folders here... or

Select files... Select folder...

**Recent models**

- You will now have an empty canvas and can start working on your DMN model for designing the number of vacation days decision. Change the name at the top of the model from `Untitled` to `call-center-decisions` or whatever you want to call it.



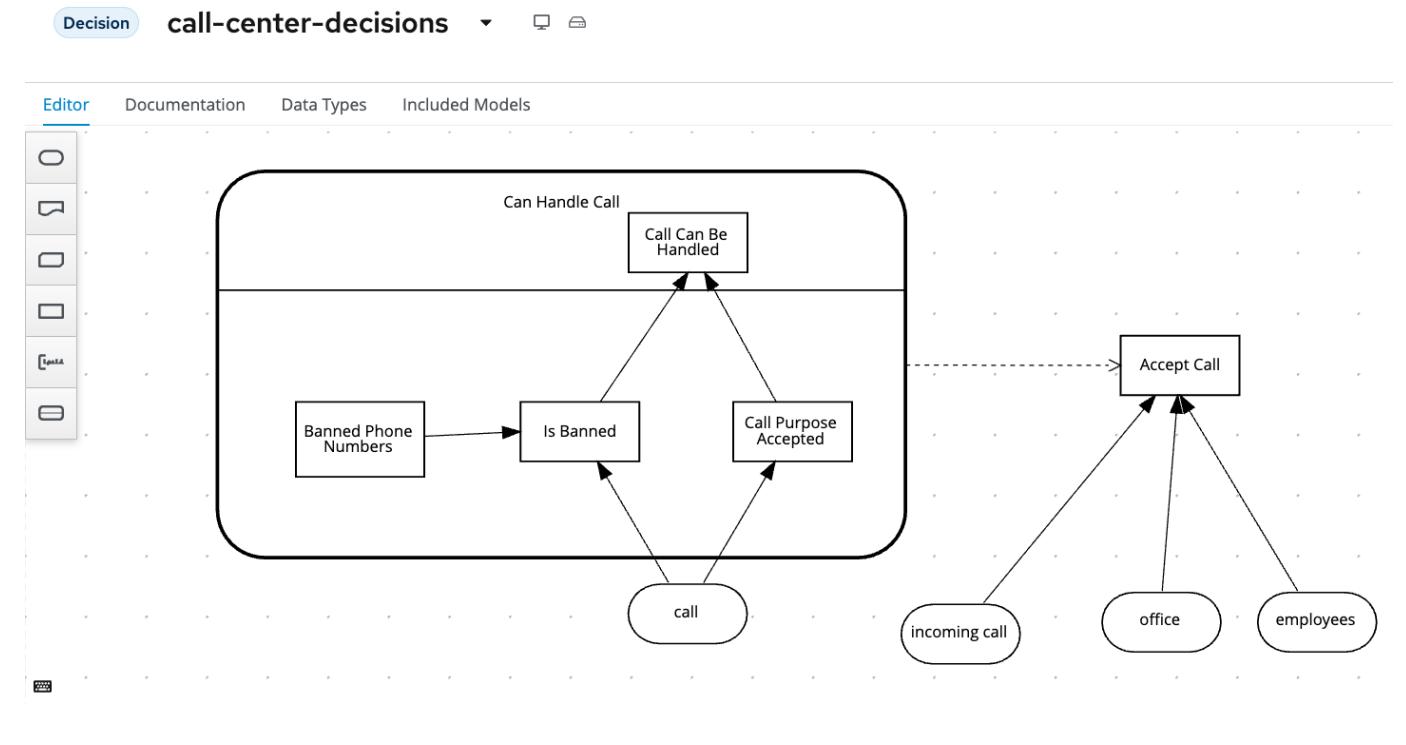
- In the next section we will start to create the decision.

### 13.4 Next Steps

You can do this lab in 2 ways:

- If you already have (some) DMN knowledge, we would like to challenge you to build the solution by yourself. After you've built solution, you can verify your answer by going to the next module in which we will explain the solution and will deploy it onto the runtime.
- Follow this step-by-step guide which will guide you through the implementation.

To do this on your own without the walk-through your model will ultimately end up looking like the below model with 5 Decision Nodes (*Banned Phone Numbers*, *Is Banned*, *Call Purpose Accepted*, *Call Can Be Handled*, *Accept Call*). Four of these are built in a Decision Service to control what parts of the decision are exposed when calling the service and ultimately a final decision to *Accept Call*.



Last update: 2023-12-06

### 3.4.7 14. Authoring the decisioning for the Call Center

---

With this lab, it is highly recommended to use DMN Runner and while you're building your model and the various custom data types to have the example right next to you. Taking advantage of the tools at hand will make your decision design much smoother and faster!

The problem statement describes a number of different inputs to our decision:

- **Call:** the incoming call into the call-center
- **Employees:** the employees of certain office.
- **Office:** an office to which the call could potentially be routed.

Furthermore, the problem statement describes that phone numbers could be banned. So, also **banned numbers** can be regarded as an input to our model (although we will not implement it as an input in this lab).

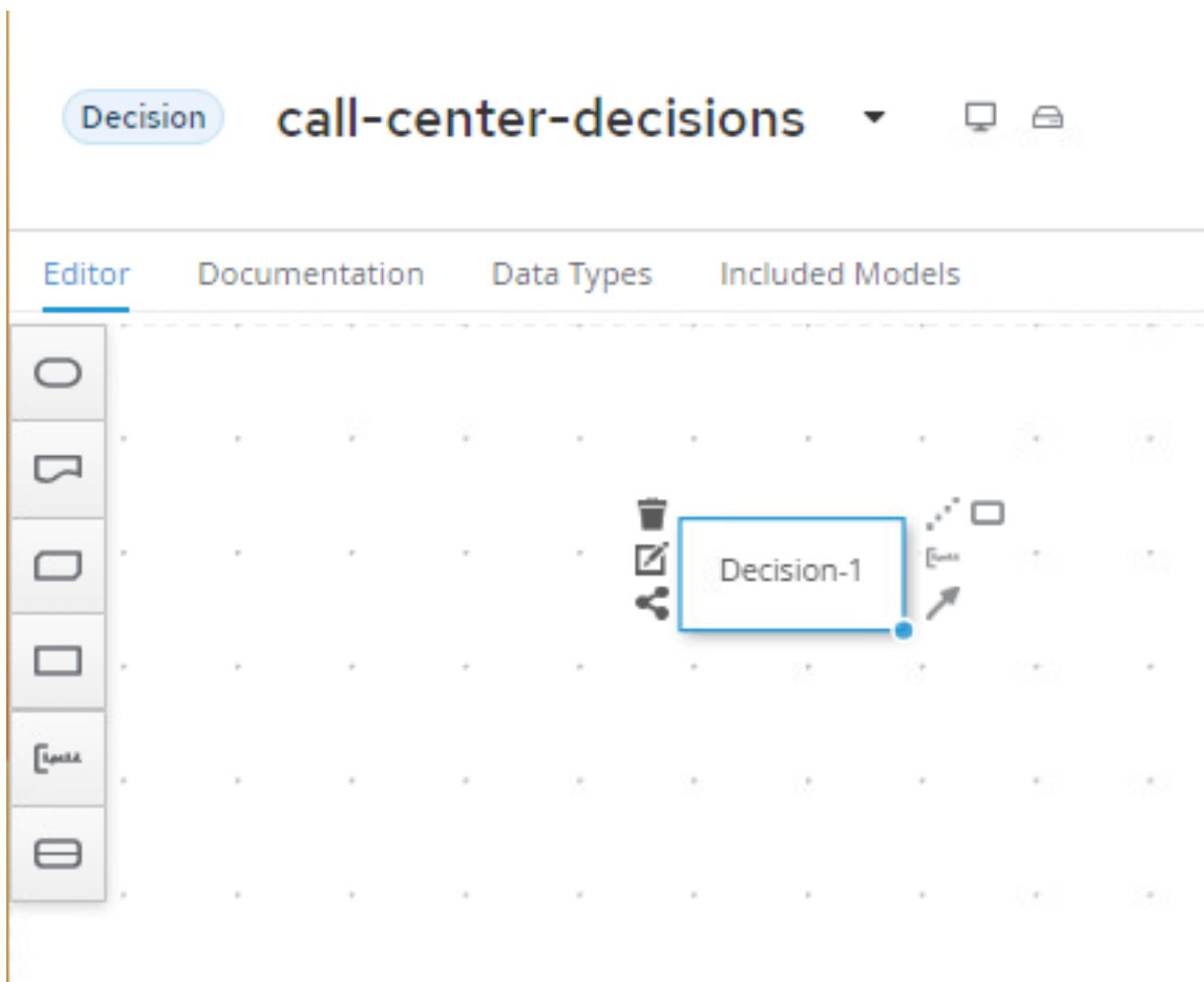
With the given input, we need to make the following decisions:

- **Accept Call:** the final decision we need to make is whether the given office will accept the call.
- **Can Handle Call:** whether the office can actually accept the call. As defined in the problem statement, this depends on:
  - whether the phone number has been banned;
  - the purpose of the phone call ("help" or "objection").

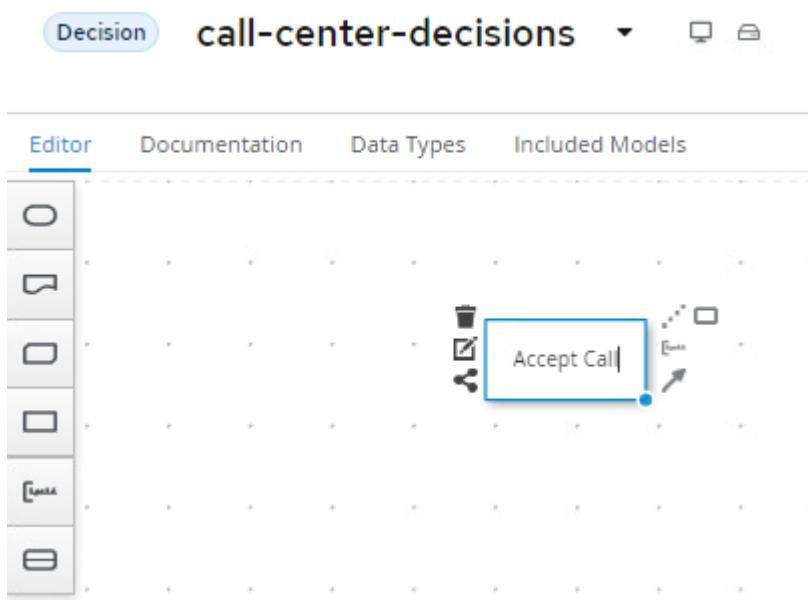
**14.1 . Accept Call Decision Structure**

- **Accept Call:** the final decision we need to make is whether the given office will accept the call.

a. Add a **Decision** node to the diagram by clicking on the **Decision** node icon and placing it in the DRD.



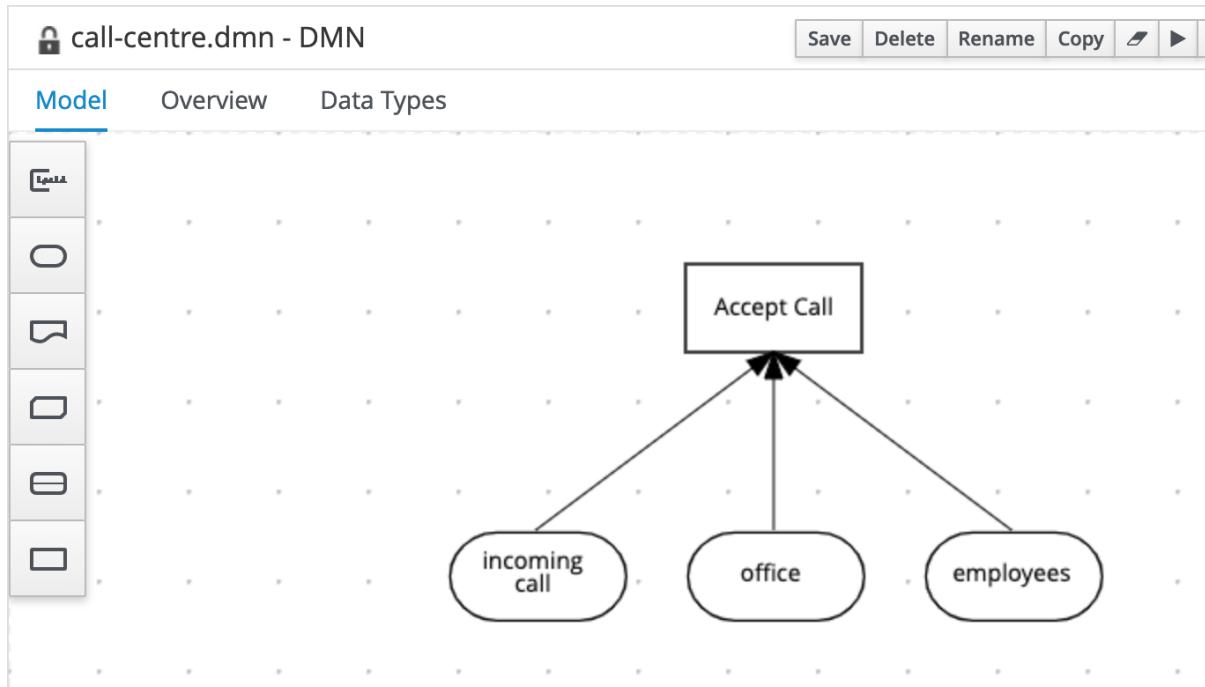
b. Double-click on the node to set the name. We will name this node `Accept Call`.



- c. With the Accept Call node selected, open the property panel. Set the **Output data type** to boolean.

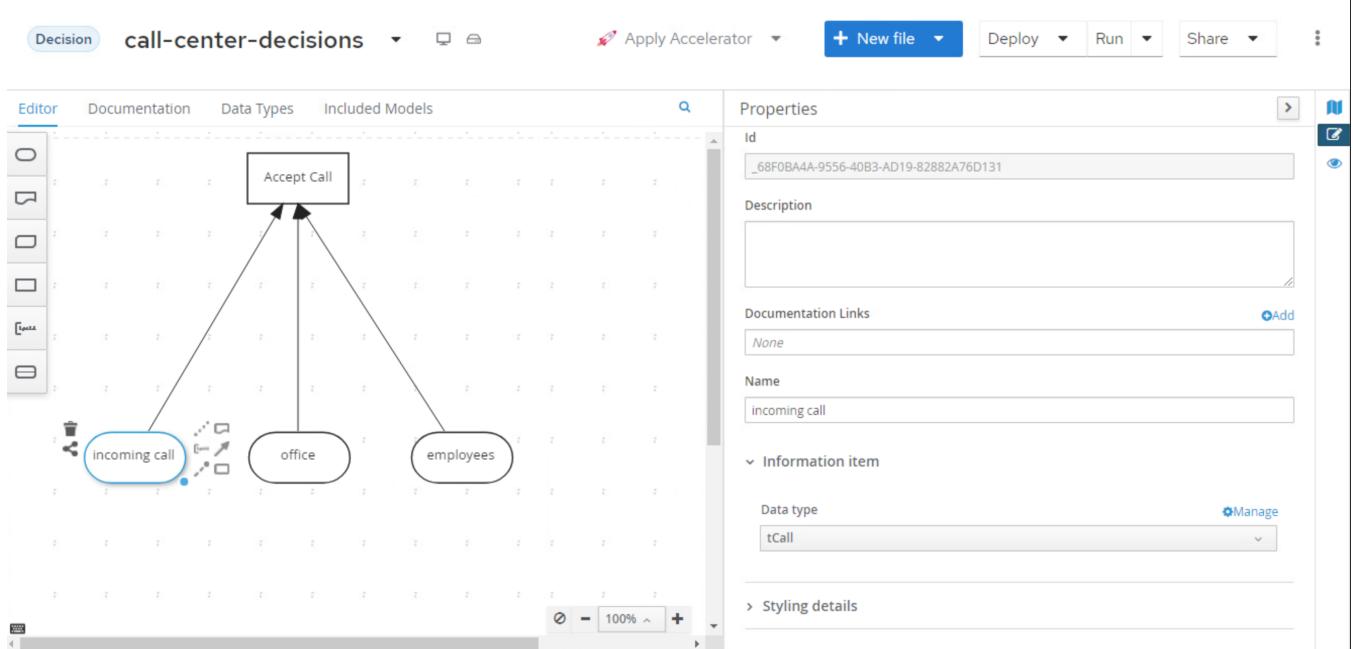
Property	Value
Id	_D46F8A76-873D-4FE2-8180-C5B2303762F4
Description	(empty)
Documentation Links	None
Name	Accept Call
Question	(empty)
Allowed Answers	(empty)
Data type	boolean

- d. The input of this decision is the **incoming call**, **office** and **employee**. Create these 3 input nodes and connect them to the **Accept Call** decision.



e. We can now set data types of our input nodes.

- i. Click on the `incoming call` node, open the property panel and in the **Output data type** section and click on the **Manage** button. This will open the **Custom Data Types** window.



- ii. In the **Custom Data Types** window, click on the **+ Add** button.

No custom data types have been defined.

Data types are referenced in the input and output values for decision tables. Custom data types allow you to reference more complex data types, beyond the simple "default" types.

Currently, there are no custom data types available for you to view or edit. To get started, add a custom data type.

**+ Add a custom Data Type**

- iii. Define the data type `Phone Number` as a structure and be sure to press the blue checkmark.



- iv. When you click the blue checkmark, a new row will open for the next type, for the first element, call it `phone number` with type `string` and then press the blue checkmark.

\* Name      \* Type  
phone number string

Add Constraints ⓘ

v. Now to create one more item, press the plus sign in the circle that appears where the checkmark was.

New Data Type

Search data types

Expand all | Collapse all

\* Name      \* Type  
phone number string

vi. For this data type call it `country prefix` with a type `string`. You can add a constraint to the limited values if you want, but not required, the example will use "+421" and "+420". These are more used for the forms rendering than anything else at this time. Make sure you check the blue checkmark when you're finished.

Custom Data Types

New Data Type

Search data types

Expand all | Collapse all

\* Name      \* Type  
country prefix string

+ "421", "420" ⓘ

vii. Define another data type `tcall` as a **Structure** which will contain both the `tPhoneNumber` structure and an additional field `purpose` of type `string`:

**tCall** Structure

**tPhoneNumber** Structure

"tPhoneNumber" is a structured data type with 2 fields.

phone tPhoneNumber

purpose string

View this data type

viii. When you've created the `tcall` type, go back to the DRD by clicking on the **Editor** tab.

New Data Type

- tPhoneNumber** Structure
  - phone number string
  - country prefix string ✖ "420", "421"
- tCall** Structure
  - phone tPhoneNumber
  - purpose string
- Custom Data Types

ix. Select the `incoming call` node, and in the property panel, under **Information item**, set the node's **Output data type** to `tcall`

f. Next, define the following data type similar to `tcall` called `tOffice` with a field `location` and set it as the **Output data type** of the `office` input as such:

**tOffice** Structure

location	string
----------	--------

g. Define the data type for `employees` as follows. Note that we've first defined the type `tEmployee`, and afterwards we've defined `tEmployees` as a List of `tEmployee`.

Custom Data Types

New Data Type		Search data types	Expand all	Collapse all
phone	tPhoneNumber			
purpose	string			
▼ tOffice	Structure			
location	string			
▼ tEmployee	Structure			
name	string			
office location	string			
tEmployees	tEmployee	<input checked="" type="checkbox"/> List <input checked="" type="checkbox"/> Yes		

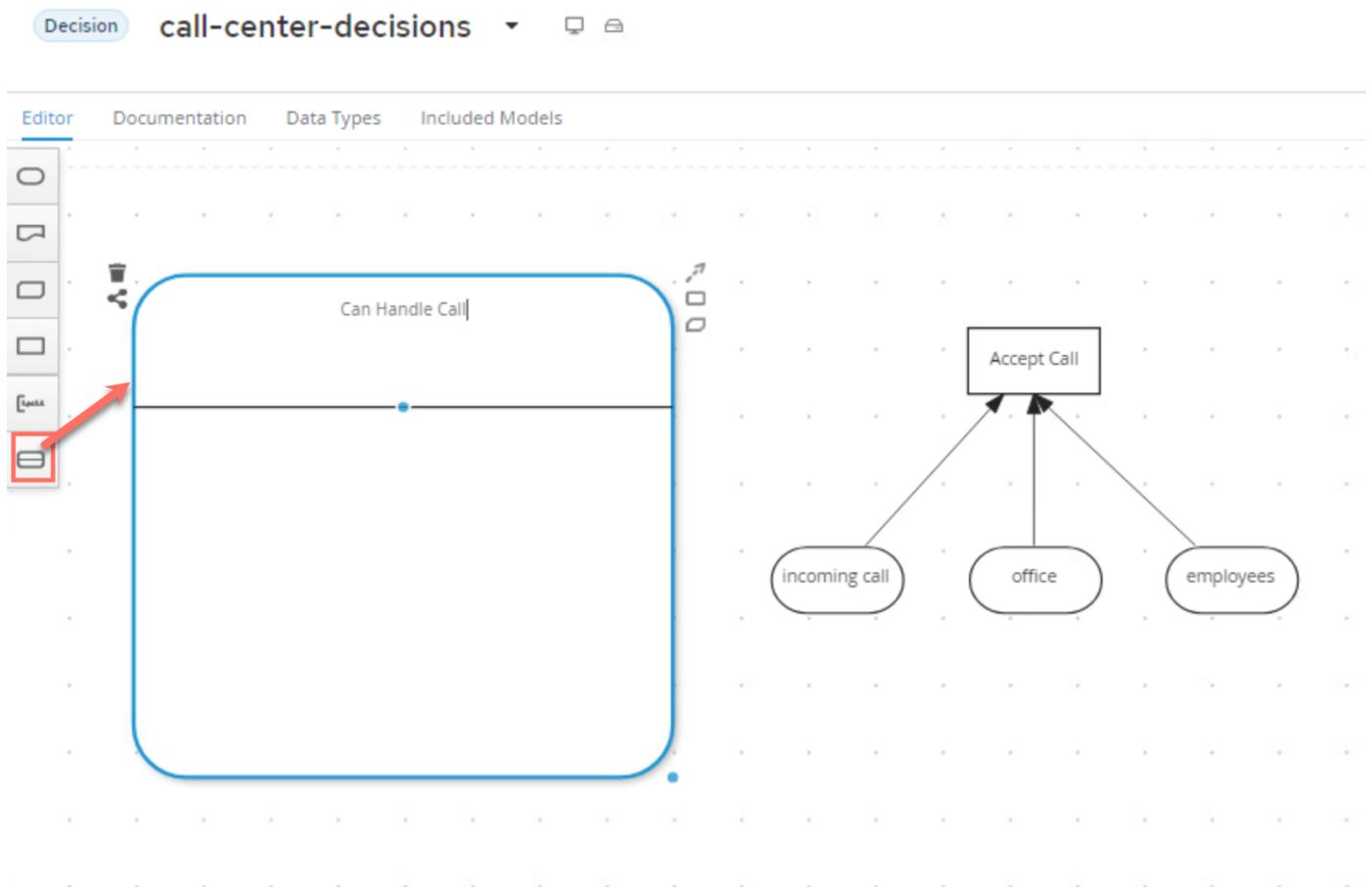
## 14.2 Decision Service

With the main structure defined, we can now look at the requirements of the decision whether the office can actually accept the call. As defined in the problem statement, this depends on:

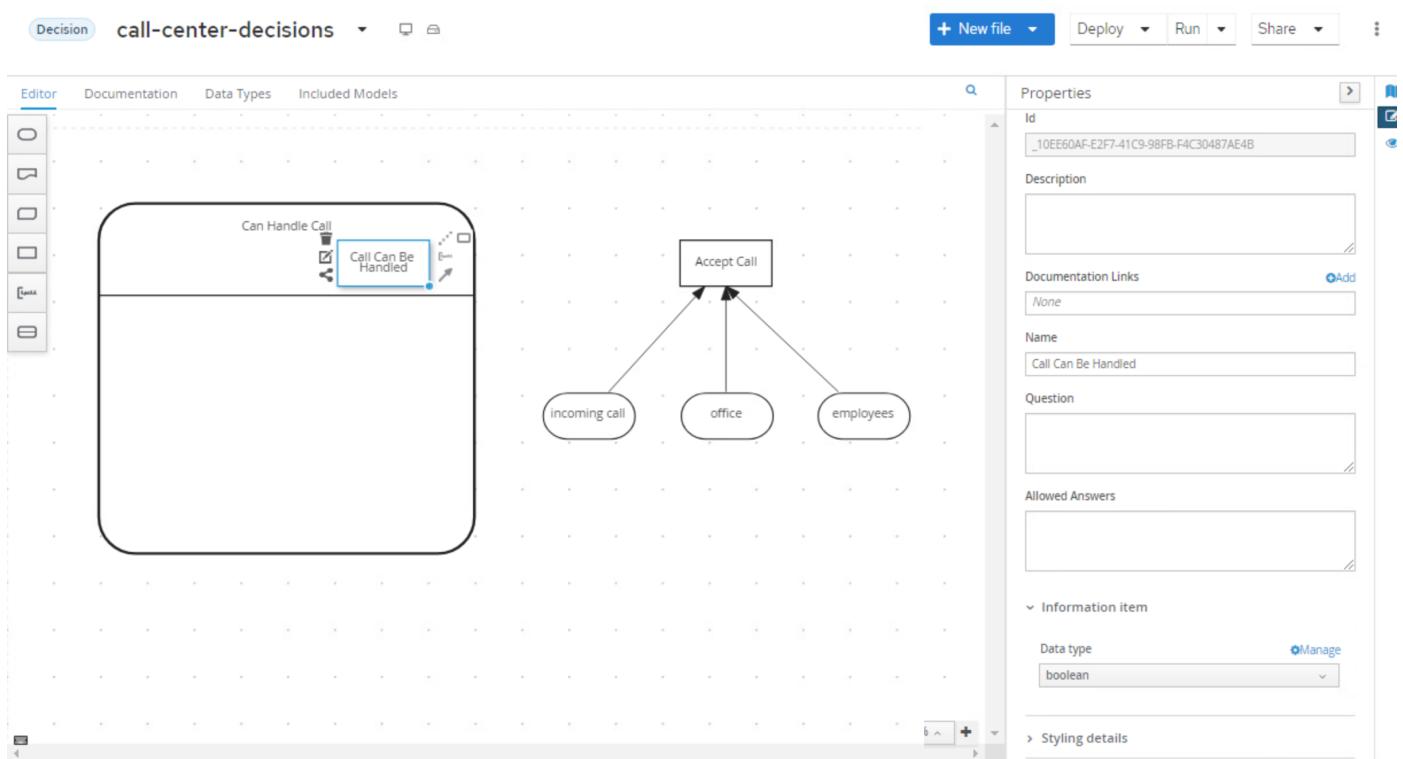
- whether the phone number has been banned.
- the purpose of the phone call ("help" or "objection").

We will model this decision as a DMN **Decision Service** that can be called by our main decision `Accept Call`.

1. First, model the **Decision Service** in the DRD and give it the name `Can Handle Call`. Set its **Output data type** to `boolean`.

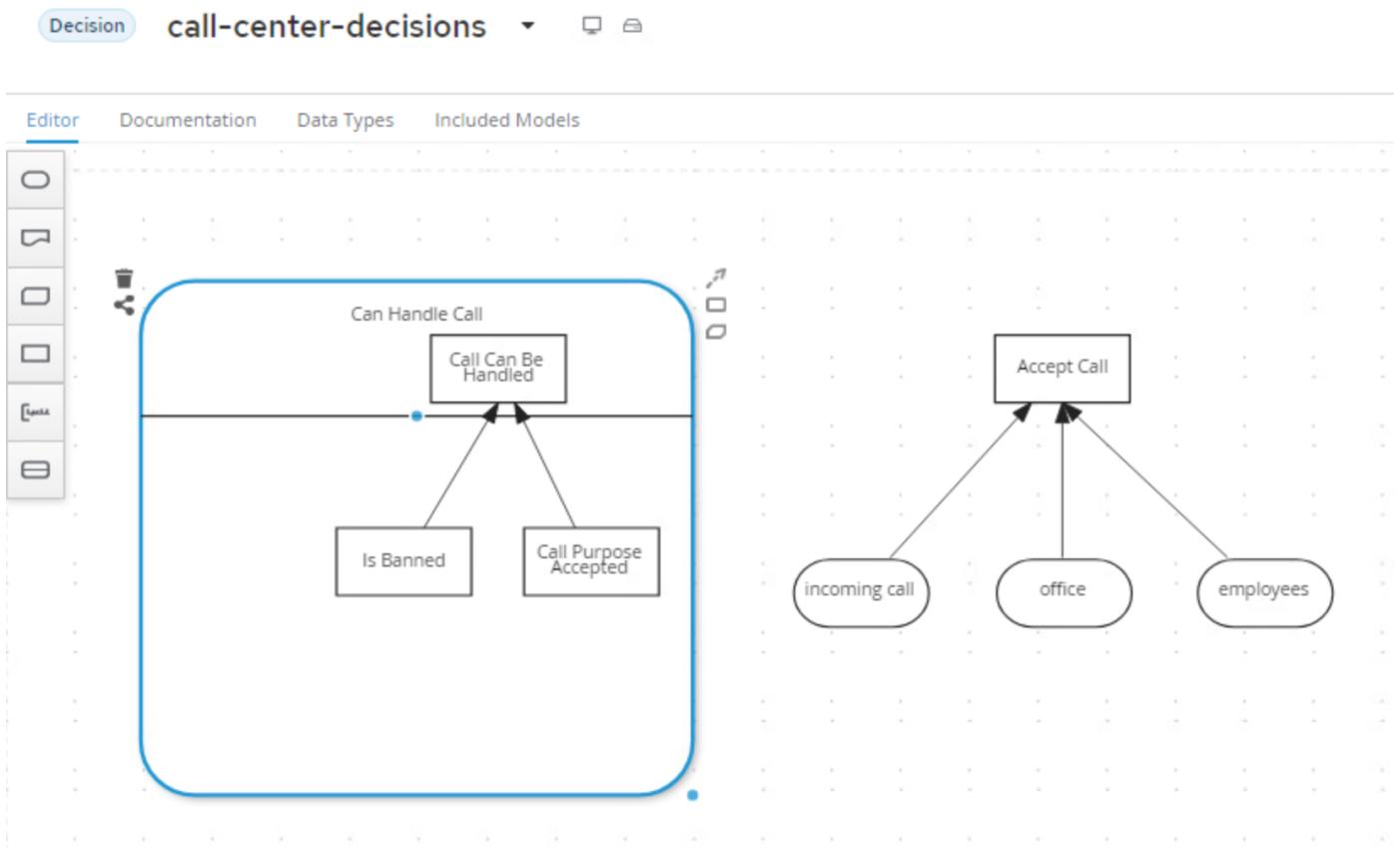


2. Add a **Decision Node** to the **Decision Service**. Name it `Call Can Be Handled` and set its **Output data type** to `boolean`.

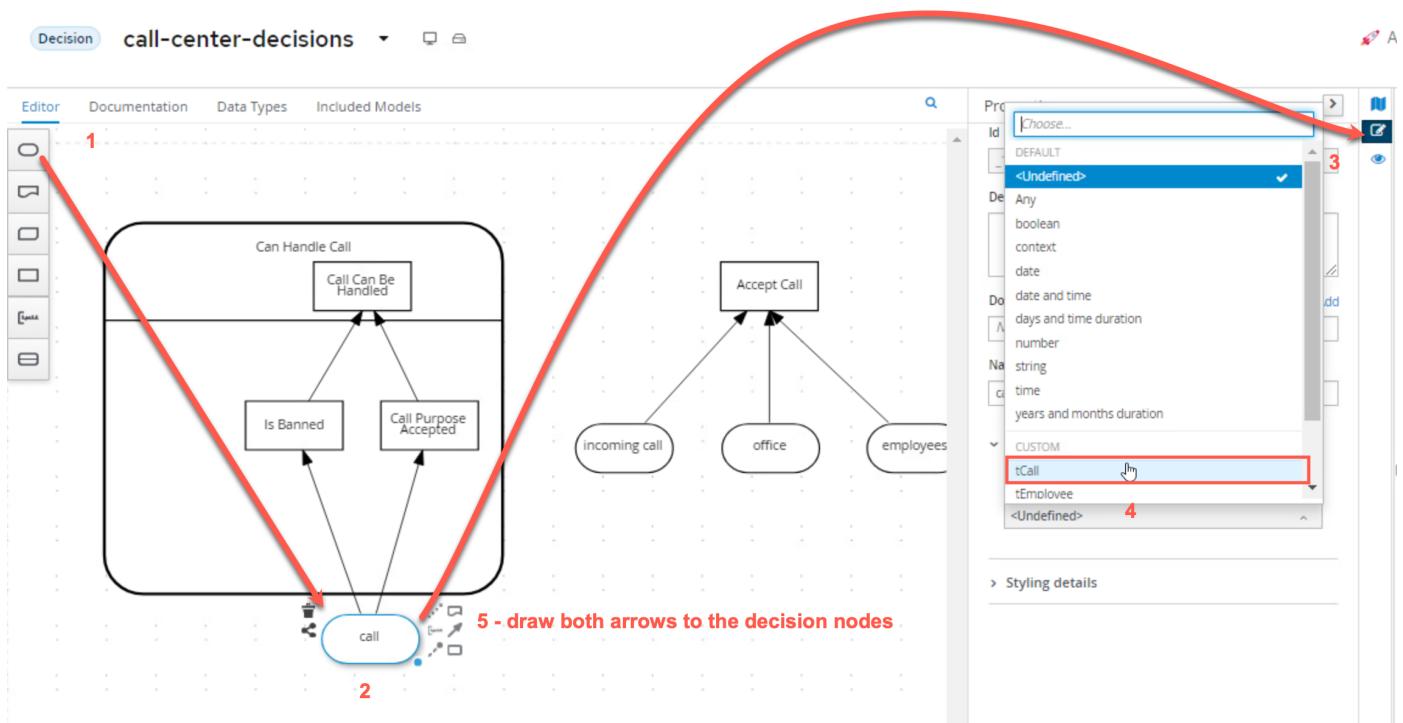


3. Add 2 additional **Decision Nodes** and name them `Is Banned` and `Call Purpose Accepted`. Both should have an **Output data type** of type `boolean`. These will be put "below the bar" on the Decision Service node, so the decisions will be evaluated, but the results will not be passed back from final service by default.

4. Connect the 2 **Decision Nodes** to the `Call Can Be Handled` node.

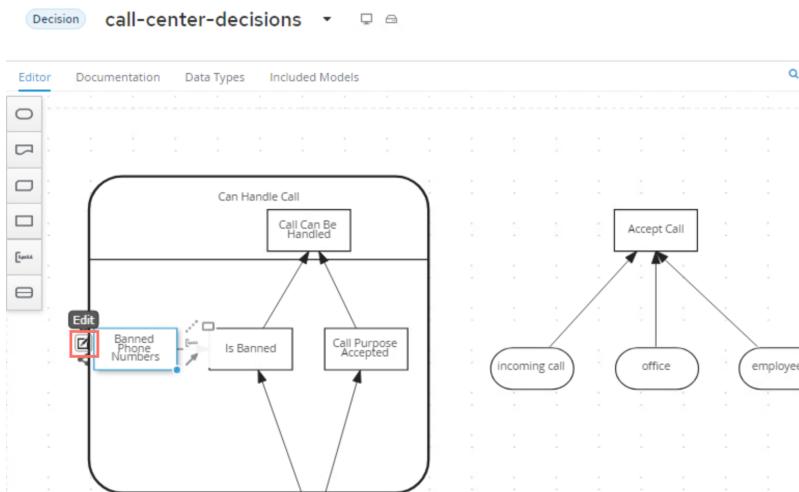


5. The input to both the `Is Banned` and `Call Purpose Accepted` decisions is going to be a separate item called `call`. You will create the input node called `call` with data type `tCall`. While it will look like you're creating a separate call item here (and that is definitely how it looks), this decision service could be called independently of the whole service, which is why this call object can be used. Later in the lab, you will see how this ultimately works!



6. The `Is Banned` decision also needs a collection of banned phone numbers. Instead of implementing this as an **Input** node, we will implement this as a DMN **Relation Decision**.

a. Create a new **Decision Node** and name it `Banned Phone Numbers`. Connect it to the `Is Banned` decision node.



b. The **Output data type** of this nodes is a new custom data type, which is a list of `tPhoneNumber`. We'll name this type `tPhoneNumbers`, similar to how `tEmployees` was setup before:

Custom Data Type	Properties
New Data Type	purpose: string Documentation Links: None Name: Banned Phone Numbers Question: Allowed Answers: Information item: Data type: tPhoneNumbers
purpose: string tOffice: Structure location: string tEmployee: Structure name: string office location: string tEmployees: #Employee List Yes	
tPhoneNumbers: tPhoneNumber List Yes	

c. Click on the **Edit** button of the `Banned Phone Numbers` node. Set the **logic type** of the decision to `Relation`. Create the following table, you will need to add one extra column and use the headers as shown of `country prefix` with type `string` and `phone number` with type `string`:

Decision **call-center-decisions** ▾

[Editor](#) [Documentation](#) [Data Types](#) [Included Models](#)

[Back to call-center-decisions](#)

### Banned Phone Numbers (*Relation*)

Banned Phone Numbers (tPhoneNumbers)		
#	country prefix (string)	phone number (string)
1	"+420"	"602000002"
2	"+421"	"902000001"

7. We can now implement the logic of the `Is Banned` decision. Click on the **Edit** button of the decision node. We will implement the logic as a **Literal Expression**. Define the following FEEL expression as below showing that we're going to take the Relational List from `Banned Phone Numbers` and comparing it to `call.phone` which is using the tCall's phone number input:

```
list contains(Banned Phone Numbers, call.phone)
```

Decision

# call-center-decisions



Editor

Documentation

Data Types

Included Models

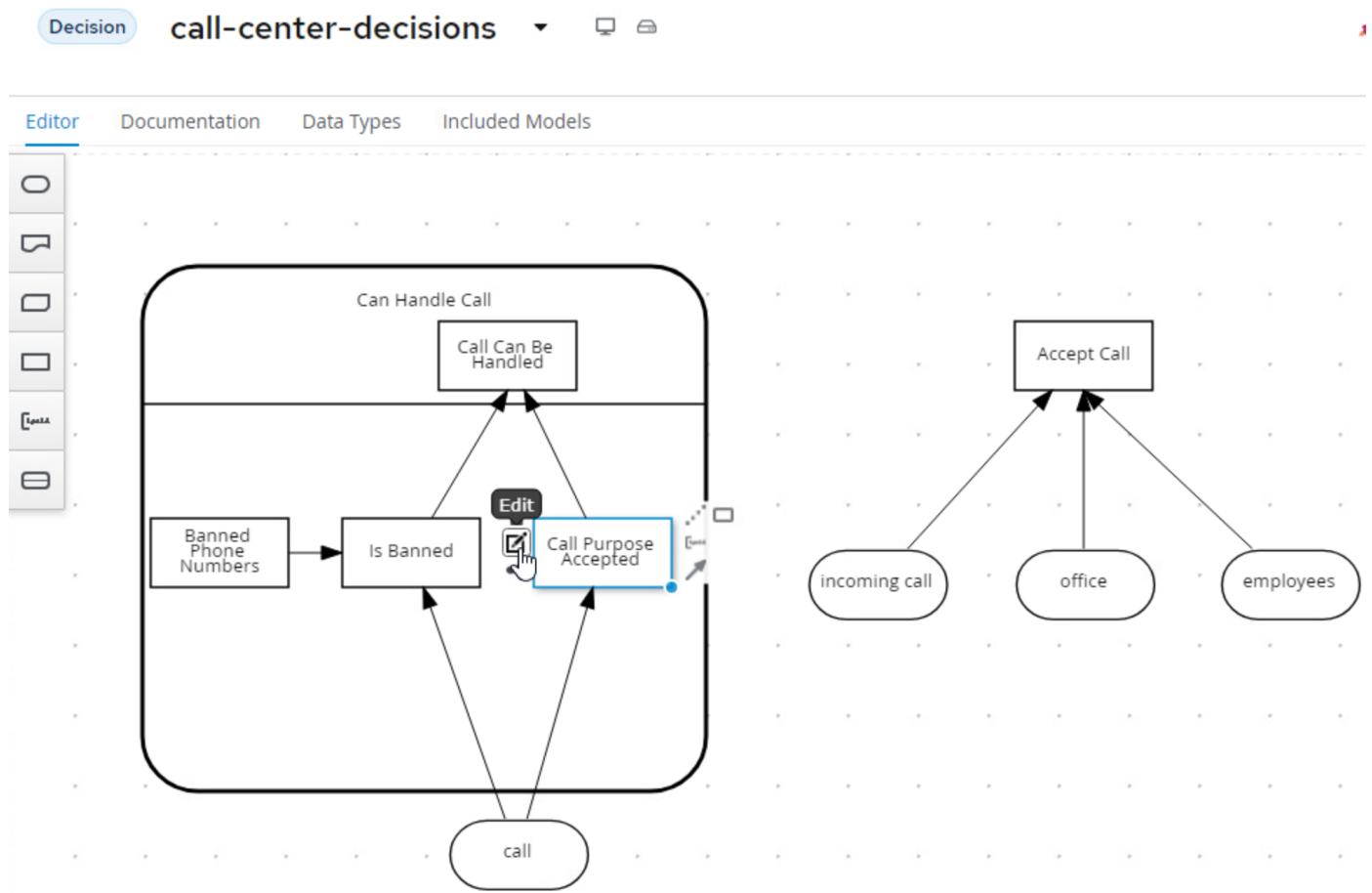
[← Back to call-center-decisions](#)

## Is Banned (*Literal*)

The screenshot shows the FEEL Editor interface. At the top left, it says "FEEL Literal". Below that is a code editor area containing the following code:

```
= Is Banned  
(boolean)  
  
list contains(Banned Phone Numbers, call.phone )
```

8. The next node for which we want to implement the decision logic is `Call Purpose Accepted`. Click on the node, and click on the **Edit** button.



9. Implement the following logic as a **Decision Table**:

Decision call-center-decisions ▾

---

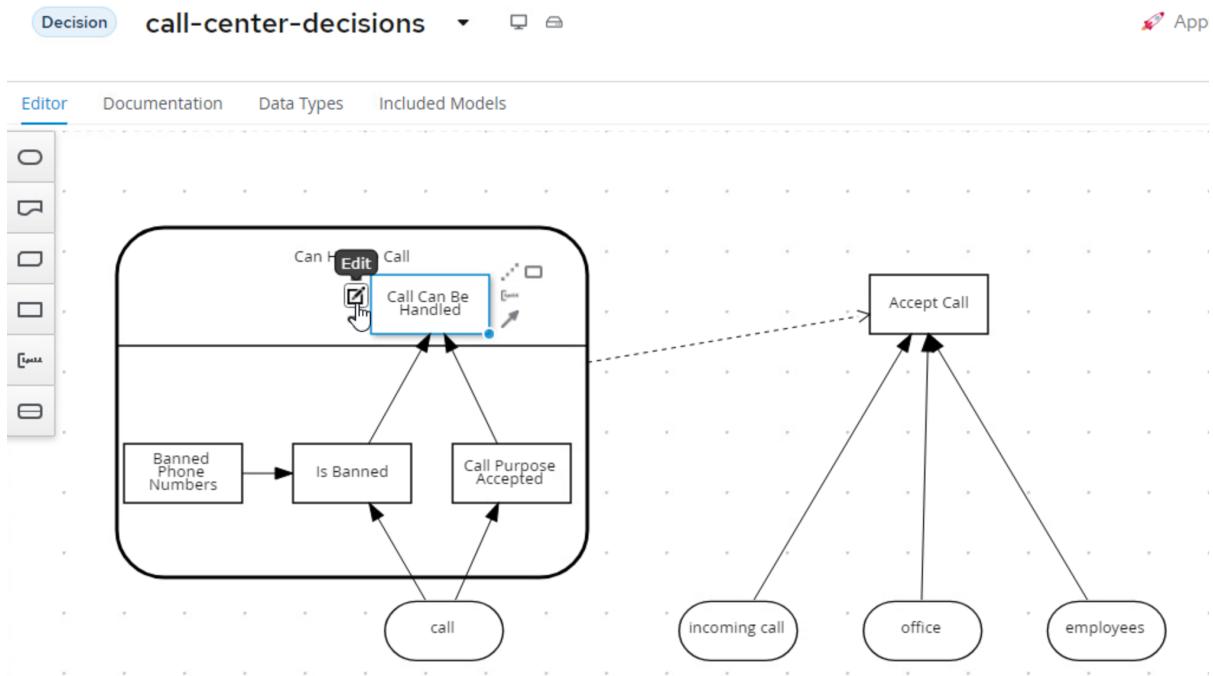
Editor Documentation Data Types Included Models

[← Back to call-center-decisions](#)

Call Purpose Accepted (*Decision table*)

Decision table			
	call.purpose (string)	Call Purpose Accepted (boolean)	annotation-1
U			
1	"help"	true	
2	"objection"	false	

10. Click on the Call Can Be Handled node and click on the node's **Edit** button.



11. In the decision editor, click the **Select expression** to **Decision Table** and implement the following table:

Decision **call-center-decisions** ▾  

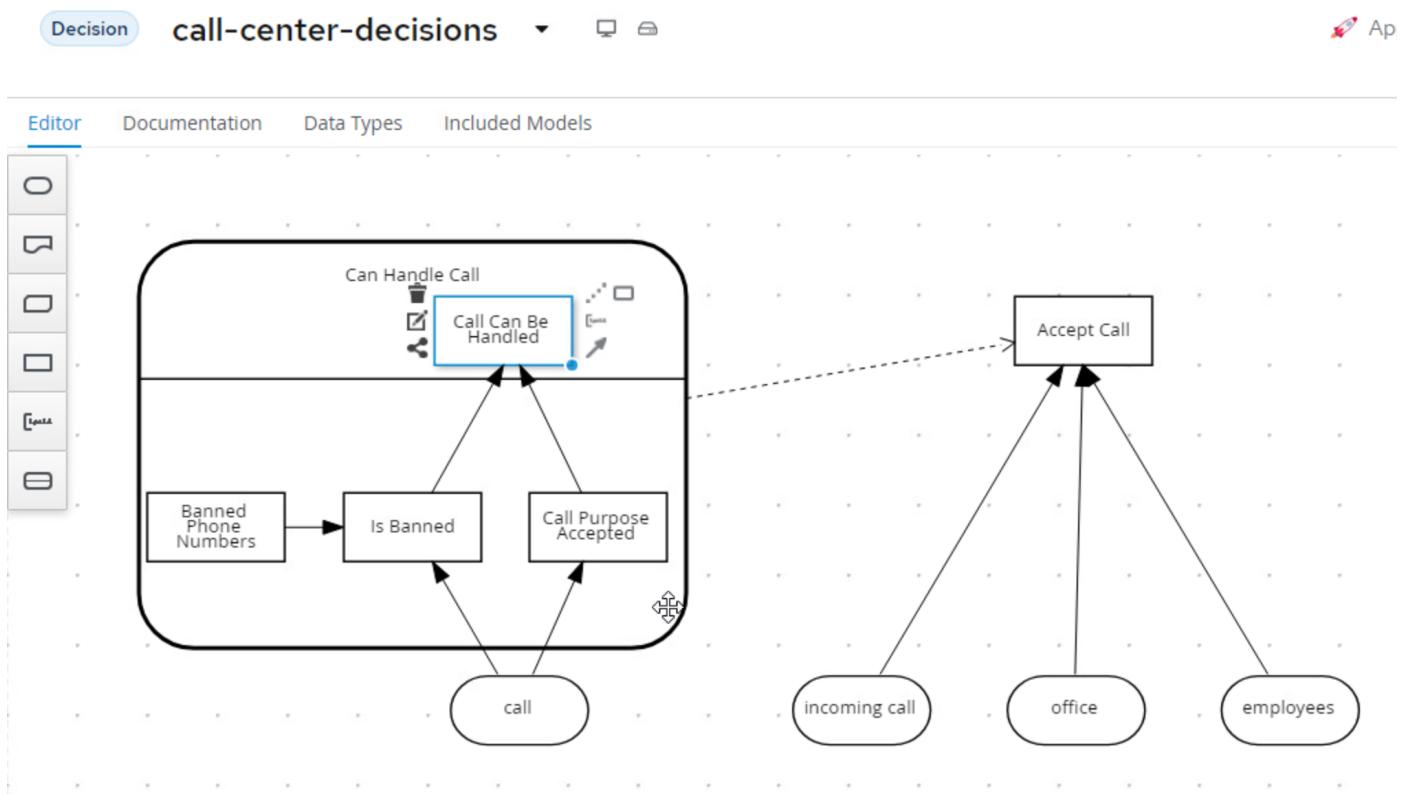
[Editor](#) [Documentation](#) [Data Types](#) [Included Models](#)

[← Back to call-center-decisions](#)

Call Can Be Handled (*Decision table*)

Decision table				
U	Call Purpose Accepted (boolean)	Is Banned (boolean)	Call Can Be Han... (boolean)	annotation-1
1	false	true	true	
2	true	-	false	
3	-	false	false	

12. We can now implement the decision of `Accept Call Decision`. To do this we need to link the Decision Service to the `Accept Call`, to do this, you will click the outside of the Decision Service node and use the `Create DMN Knowledge Requirement` to connect the Decision Service to the `Accept Call`.



#### 14.3 "Accept Call" Decision Logic

- Finally we need to Implement the `Accept Call` decision. This is going to be built as a `Context` since this can be built as a boxed expression that will have multiple steps yielding the final decision to either Accept or Deny the call!

Decision call-center-decisions ▾

---

Editor Documentation Data Types Included Models

[Back to call-center-decisions](#)

Accept Call (<Undefined>)

Select expression

FEEL Literal

{}

Context

- Decision table
- Relation
- f() Invocation
- List

---

Paste

New

---

2. With the Context now the Decision Type. The first row is going to be `call` can be handled with type `boolean`.

[Decision](#)

# call-center-decisions

[Editor](#)[Documentation](#)[Data Types](#)[Included Models](#) [Back to call-center-decisions](#)

## Accept Call (*Context*)

The screenshot shows the configuration of a context named 'Accept Call'. The context has one row with the expression 'Call can be handled' and type 'boolean'. The row is highlighted with a blue border. The 'Name' field contains 'Call can be handled' and the 'Data Type' field is set to 'boolean'. A 'Select expression' button is visible next to the expression field.

Name	Expression	Type
Call can be handled	Select expression	boolean

**Name:** Call can be handled

**Data Type:** boolean

3. Now we need to set the expression for the first row, this is going to be of type `Invocation`.

Decision

## call-center-decisions



Editor

Documentation

Data Types

Included Models

[← Back to call-center-decisions](#)

## Accept Call (Context)

{} Context ▾

Accept Call (boolean)	
Call can be handled (boolean)	Select expression
<result>	Select expression

**FEEL Literal**

- { } Context
- [ ] Decision table
- [ ] Relation
- f Function

**f()** Invocation  

4. For the invocation, we need to use it to call the Decision Service that was created.

5. [call-centre.dmn - DMN](#) 

Model Overview Data Types

[« Back to "Accept Call"](#)

### Context

#	Accept Call (boolean)	
1	Call can be handled (boolean)	# Can Handle Call
		1 call (tCall) incoming call
2	Employee at the office (boolean)	employees[office location = office.location]
	<result>	if Call can be handled then count(Employee at the office) > 0 else false

Notice that the line 1 is the invocation of the decision service "Can Handle Call". This is an **Invocation** of the `Can Handle Call` service, passing the `incoming call` input as the variable `call`. The output of this invocation will be the `boolean` variable `Call can be handled`. This will be done using a Literal FEEL function to do this.

1. Next create a new line with a data type `Employee at the Office` with data type `boolean`. Set the expression as another Literal expression. In this function, you will be mapping the Office locations of the employees who are working to the location of the office location. Since this is a list, you're essentially getting a collection of the employees office location and the location of the office that the call is coming from and trying to map them.

```
employees[office location = office.location]
```

2. After this is set, the last row, `<result>` can be set with a Literal expression again with the following:

```
if Call can be handled then count(Employee at the Office) > 0 else false
```

The `call can be handled` variable as then used to validate the decision result in the last line.

The screenshot shows the Camunda Modeler interface with the project name "call-center-decisions". The "Editor" tab is selected. The decision model "Accept Call" is displayed, showing its context and structure. The context includes variables like "Call can be handled" (boolean), "Employee at the Office" (boolean), and "<result>". The decision logic involves invoking the "Can Handle Call" service with the input "call" (tCall) and receiving the output "incoming call". It then maps "Employee at the Office" to a list of employees where "office location" equals "office.location". Finally, the result is determined by checking if the count of employees is greater than 0, returning true or false.

#### 14.4 Next steps

Next, we should deploy the project in OpenShift as a sample service.

Last update: 2023-12-06

### 3.4.8 15. Deploying and testing the Decision Service

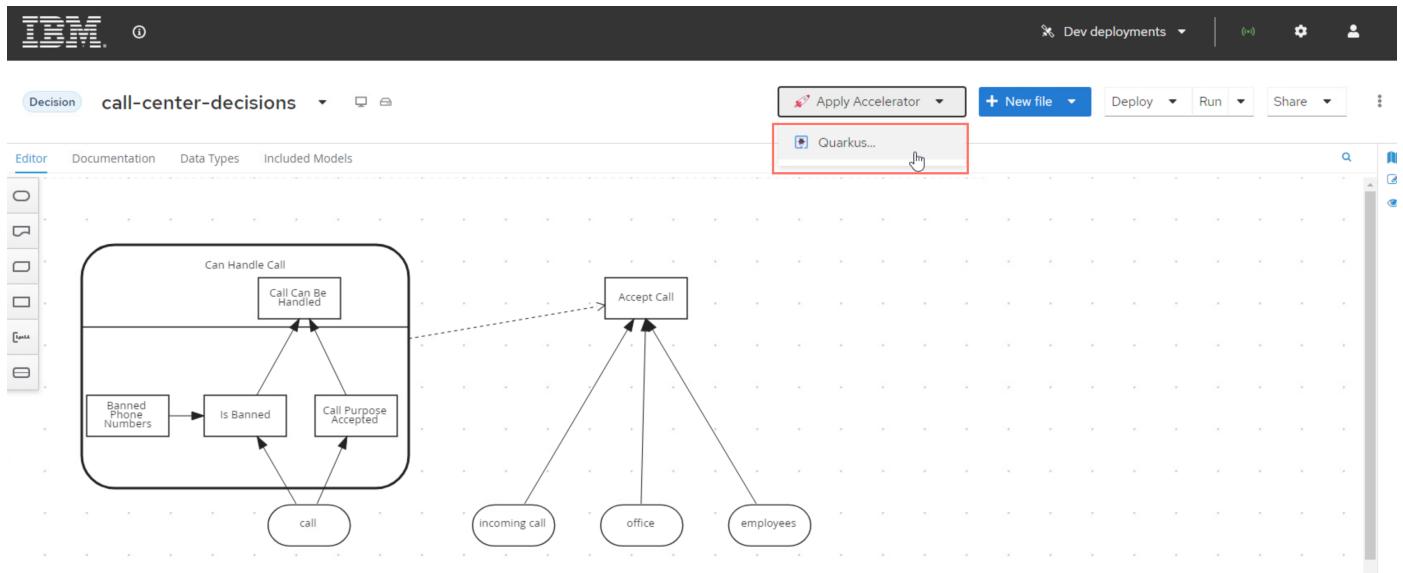
---

With our decision model completed, we can now package our DMN model in a Kogito service. This can be done using the Accelerators included with IBM Business Automation Manager Canvas. These accelerators will quickly take the DMN file and start building the basis of the project as a IBM Business Automation Open Edition 9.0 compatible decision service.

### 15.1 Applying the Accelerator

To take this decision service from a standalone DMN model to a full Kogito architected decision, you will use the accelerator by clicking the button at the top of the screen. To do so follow the instructions and then we will synchronize the project to your GitHub account and also be capable being deployed as an OpenShift service.

- At the top of the DMN model page for `call-center-decisions`, click the button **Apply Accelerator** and select **Quarkus....**



- If you want to learn more about the accelerator, you can click the GitHub link and change the branch to the `9.0.0-quarkus-full` to learn more about it. Afterwards just click **Apply** to restructure your project into a Kogito service.

## Quarkus Accelerator

An Accelerator is a template. Applying it will move your current files according to the Accelerator specifications and create a new commit for it.

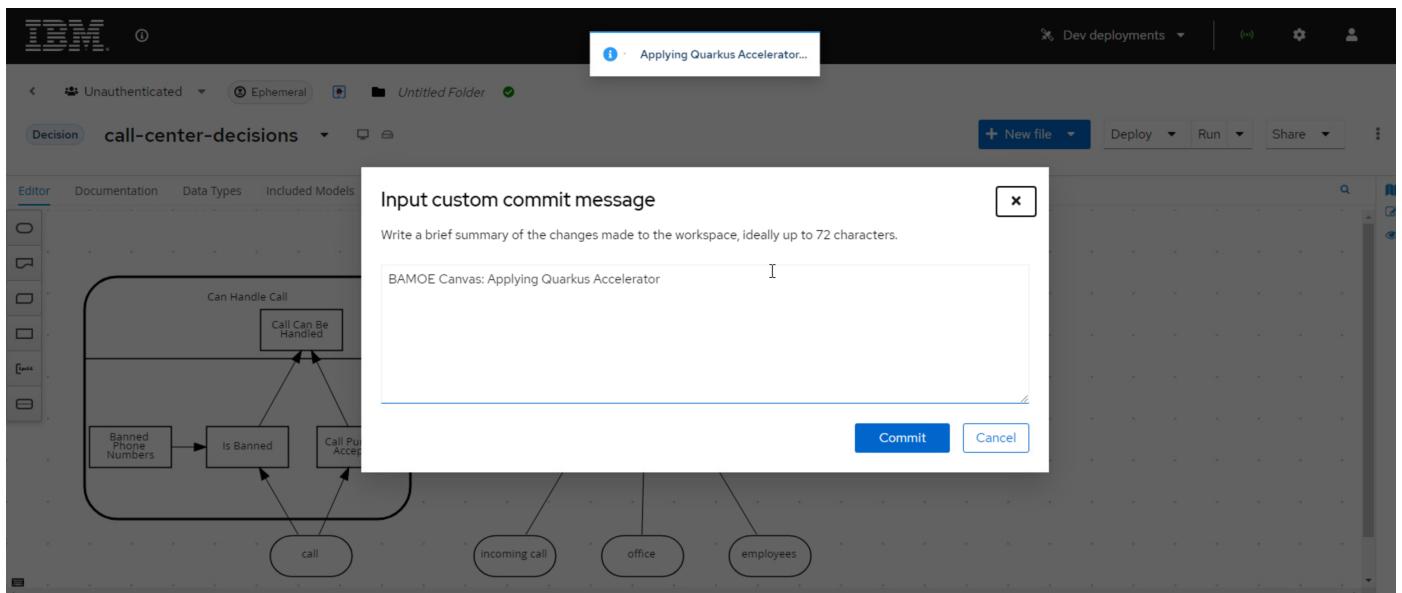
This Accelerator is hosted at <https://github.com/ibm/bamoe-canvas-quarkus-accelerator>  
 Git ref: @9.0.0-quarkus-full

Decisions (.dmn) will be moved to:	<code>src/main/resources/dmn</code>
Score cards (.pmml) will moved to:	<code>src/main/resources/dmn</code>
Workflows (.bpmn, .bpmn2) will be moved to:	<code>src/main/resources/bpmn</code>
Other files will be moved to:	<code>src/main/resources/others</code>

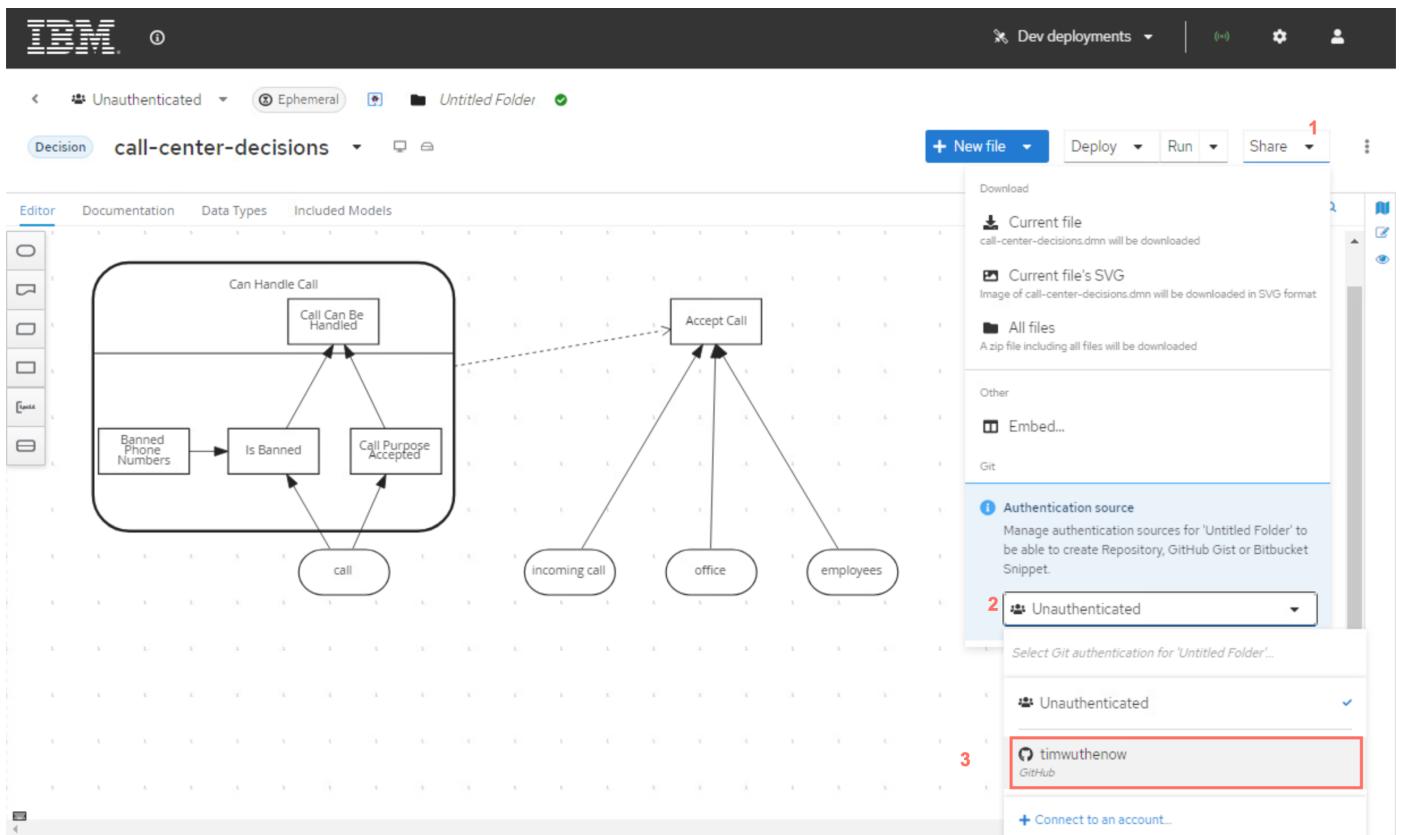
*This action is permanent. Any changes made after applying an Accelerator may result in your files being in different directories.*

[Apply](#)
Cancel

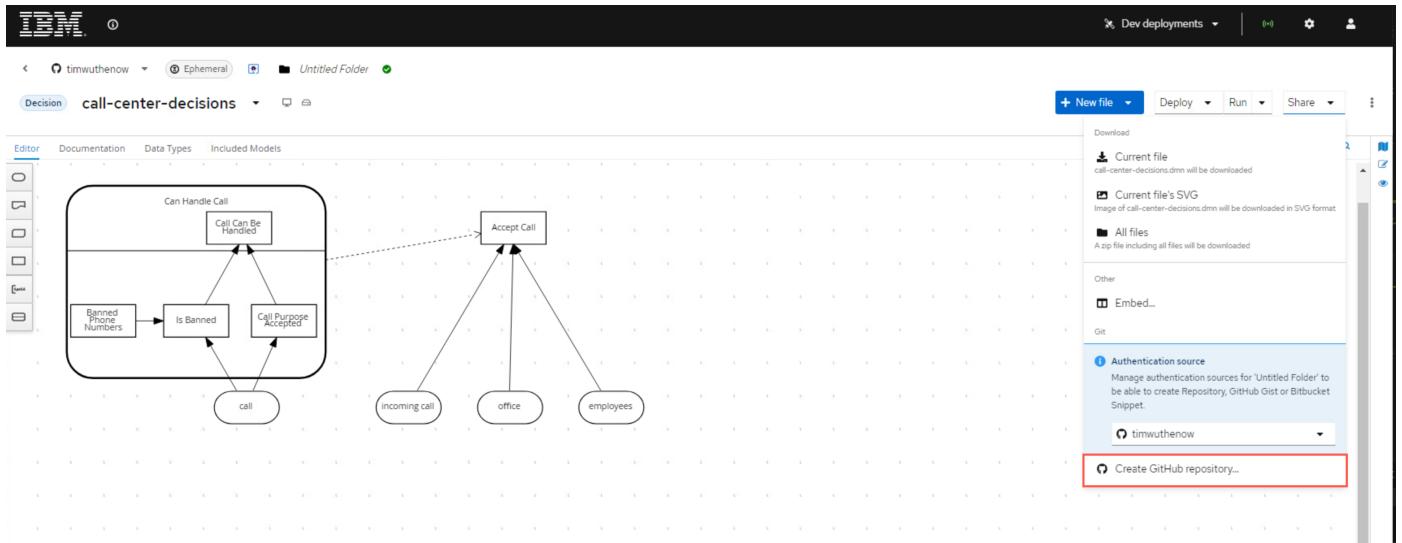
- The wizard will come with a pop-up asking for a commit message for the change since this will create a Git project. You can use the default message or put whatever you would like, to do so press **Commit**.



4. To create the project in Git, click **Share**, use your GitHub tokened ID.



5. Once you select your ID, a new option **Create GitHub repository...** is now available, select this.



6. The repository you create, can be anything you want, for the purposes of this lab, it will be called **techxchange-call-center**.

## Create GitHub repository

The contents of 'Untitled Folder' will be all in the new GitHub repository.

The new repository will be created under the following scope

timwuthenow



Pick either your user account or a GitHub organization.

Name \*

techxchange-call-center



Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

**Create**

7. When you create the repository, a message will open at the top with a link to the Git repository that was created. In the case of the example it is at [this example repository](#).

The screenshot shows a GitHub repository page for 'techxchange-call-center'. The repository is public and has 1 branch and 0 tags. The main file listed is 'README.md'. The repository has 2 commits, all made 17 minutes ago by the same user. The commit messages are: 'BAMOE Canvas: Applying Quarkus Accelerator' for each commit. The repository has 0 stars, 1 watching, and 0 forks. There are sections for 'About', 'Releases', and 'Packages', all of which are currently empty.

**BAMOE Canvas** BAMOE Canvas: Applying Quarkus Accelerator

9a823fe 17 minutes ago 2 commits

.kie-sandbox  
src/main/resources  
.gitignore  
LICENSE  
README.md  
logo.png  
pom.xml

Go to file Add file <> Code

**About**

No description, website, or topics provided.

Readme Apache-2.0 license Activity 0 stars 1 watching 0 forks

**Releases**

No releases published Create a new release

**Packages**

No packages published Publish your first package

**README.md**

## bamoe-canvas-quarkus-accelerator

This is the BAMOE Canvas Quarkus Accelerator for Decisions and Workflows

Apply it to your repository inside BAMOE Canvas to build and run your decisions.

*NOTE: Docker is required to run this image with all its dependencies.*

### Running

Start the Quarkus server with:

- mvn quarkus:dev

## 15.2 Deploying to OpenShift

Now that this repository is created, you can work on it locally or deploy it to OpenShift. For this lab, we will choose OpenShift since we have a cluster ready to use for TechXchange or you can use the [Red Hat OpenShift Sandbox](#) to provision a small environment to try it out on your own.

1. Login to get an OpenShift Token using the bookmark on Chrome. This will be used to connect IBM Business Automation Manager Canvas to OpenShift and deploy a sample development environment of the decision service.

The screenshot shows a browser window with the URL `localhost:9090/#/a411e146-8c6f-4ac7-892a-3a5d10887f25/file/src/main/resources/dmn/call-center-decisions-final.dmn`. The tab bar has several items: 'Learning IBM Busin...', 'BAMOE Canvas', 'OpenShift Token' (highlighted with a red box), 'Business Central', 'Weather App', 'EmployeeOnboardi...', 'payment test', 'KIE Server Swagger', 'Pricing App', and 'OpenShift Console'. A message box in the center of the screen says 'GitHub repository created.' with a link to <https://github.com/timwutheNow/techxchange>.

2. In the login screen use your username and password - this will be some combination of `student01` to `student20` with password `Passw0rd`. The username is all lowercase. This will

The screenshot shows a 'Log in to your account' form. It has fields for 'Username \*' containing 'student01' and 'Password \*' containing 'Passw0rd'. Below the fields is a blue 'Log in' button. To the right is a dark sidebar with the Red Hat OpenShift logo and the text 'Welcome to Red Hat OpenShift'.

3. Click the **Display Token** link and switch back to the tab with IBM Business Automation Manager Canvas open on it. You will use the token contents from the OpenShift token page shortly by clicking on **Dev deployments** and then selecting the pull down **Select authentication** and then **Connect to an account**.

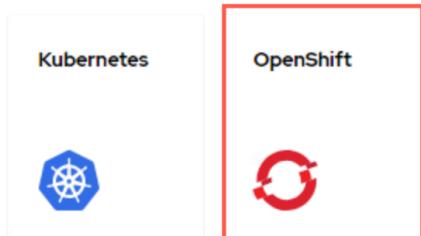
The screenshot shows the 'Create' page of IBM Business Automation Manager Canvas. At the top, there's a dropdown menu labeled 'Dev deployments' with a red box around the number '1' next to it. Below the menu, a dropdown labeled 'Select authentication' has a red box around the number '2' next to it. At the bottom of the dropdown, there's a button labeled '+ Connect to an account...' with a red box around it and the number '3' next to it.

4. On the next wizard, select the **OpenShift Tile** to connect to an OpenShift instance.

## Select a provider



**Cloud** Allows Dev deployments to be created in your Cloud infrastructure.



5. From here you will use the information from the OpenShift token page. Your namespace will be your `student##-namespace`, so if your username is student07, then you would connect to `student07-namespace`. The token and the console locations are both a part of the token as well. You will see a checkbox to Insecurely disable TLS validation, check it as this is a self contained environment with certificates not present. Press **Connect** when the form is completed.

Your API token is  
sha256~[REDACTED]

Log in with this token

Connect to OpenSh

oc login -token=sha256~ -server=https://api.ocp.ibm.edu:6443

Use this token directly against the API

curl -H "Authorization: Bearer sha256~" "https://api.ocp.ibm.edu:6443/apis/user.openshift.io/v1/users/~"

Configure a new Developer

Namespace (project) \* student20-namespace

Host \* https://api.ocp.ibm.edu:6443

Token \* sha256~

Insecurely disable TLS certificate validation

Checking this option will insecurely disable TLS certificate verification for this account. This is an alternative to not having to deal with the browser's restrictions when your cluster is behind an HTTPS endpoint with a self-signed certificate. Please be advised that the use of self-signed certificates is a weaker form of security, so consider contacting your cluster admins to use a trusted certificate. For more information, refer to <https://cwe.mitre.org/data/definitions/295.html>.

Connect

6. As long as it was successful to connect, you will be greeted with a Connect to OpenShift successful modal, click **Continue**.

**Connect to OpenShift**

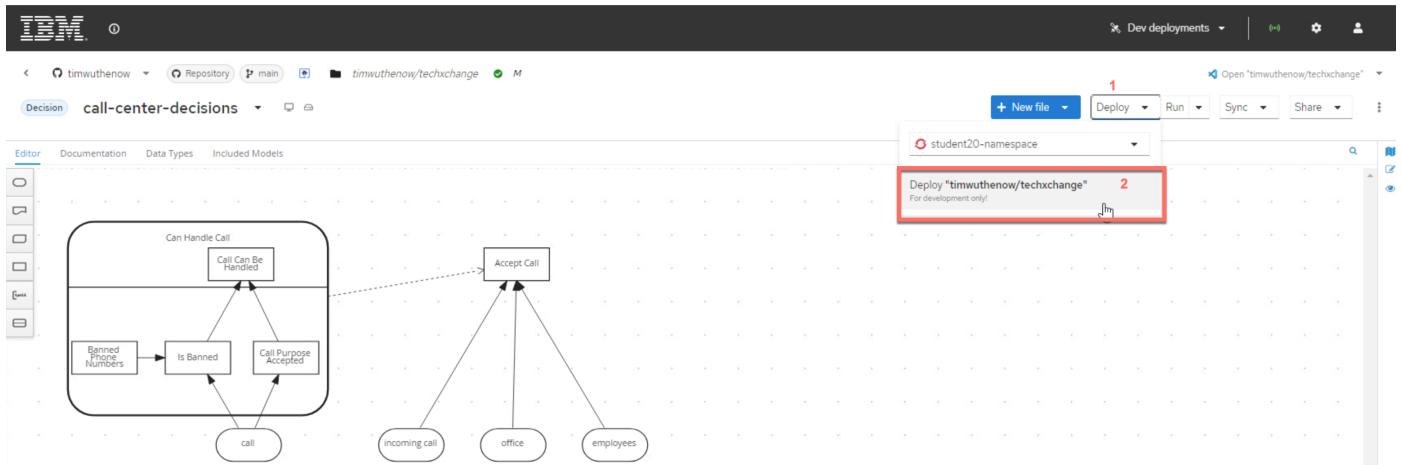
---

Successfully connected

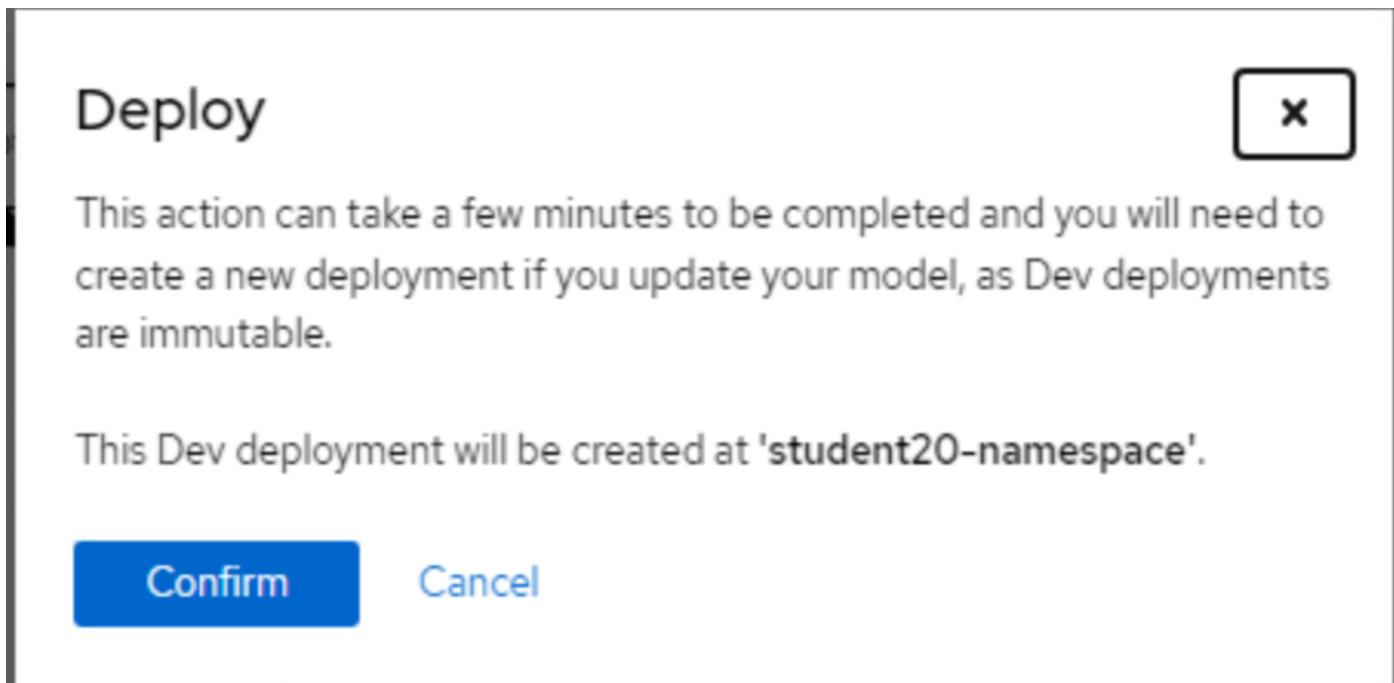
Namespace student20-namespace  
Host https://api.ocp.ibm.edu:6443  
Token sha2\*\*\*\*\*vNZw  
Created at 2023-08-30T19:39:57.425Z  
TLS Certificate Verification Disabled

**Continue**

7. Now you can press the **Deploy** button and have the capability to select the namespace that you connected to and deploy a sample of your service to it.



8. From here a modal will pop up explaining that the deployment can take a few minutes and it is only intended for development, etc. Click **Confirm** on this modal and your service will be deployed.



9. This process will take several minutes to complete.

Canvas deploy in progress

10. The pulldown with deploy will refresh every 30 seconds, when the service is available, it will reflect as such. The way that the deployments from these services work are is that they are completely immutable and unaware of one another, so you could have all versions deployed as you make changes to validate how you want to eventually promote your DMN model to later environments. So if you were to deploy it again, there would be multiple different versions of it deployed. Canvas can manage these deployment samples, which can later be removed when you're done with them if you choose. If you click the green check marked box, it will take you to the service.



11. When you access the service with a form similar to the one you had in IBM Business Automation Manager Canvas and from that page you can explore it more using the kebab icon in the top right to access the Swagger-UI page to get access to the auto-generated DMN service's API page. The first post can be used to execute the decision itself.

**dmc-dev-deployment-quarkus-app API** 0.0.0 (OAS3)

/q/openapi

### Call \_45center \_45decisions Resource

**GET** /call-center-decisions

**POST** /call-center-decisions

**Parameters**

No parameters

Request body application/json

```
DMN input
{
  "purpose": "string",
  "office": {
    "location": "string"
  },
  "employees": [
    {
      "name": "string",
      "office location": "string"
    }
  ],
  "call": {
    "phone": {
      "phone number": "string",
      "country prefix": "+420"
    },
    "purpose": "string"
  }
}
```

**Execute** **Clear**

12. This completes this lab as you can see how the deployment to OpenShift works with IBM Business Automation Manager Canvas!

Last update: 2023-12-06

## 3.5 16. IBM Business Automation Open Edition 9.0 CI/CD Overview

This lab was forked from the KIE Live session delived by Rafael Soares on YouTube found [here](#). This lab has been updated to more current releases and also includes a provisioning of Nexus3 to deploy a Maven Repository in your environment. You do not need to do this if you have a Maven repository you can deploy your artifacts to. This lab will be using the OpenShift provided Pipelines built upon Tekton. With IBM Business Automation Open Edition 9.0, the projects are based on Maven artifact architectures, which provides an easy to build and deploy mechanism for all of your product needs. These can be done in Tekton as seen here, or with your favorite/enterprise provided CI/CD tools.

The project in this lab is a Spring Boot hosted Decision Service that is utilizing the embedded KIE Server that will have a build and deploy triggered through the changes in your code base that are pushed to the specified build branch. With Decision Service projects, it is highly recommended that on all push events you consider a build of the project. This ensures that all the resources that are required are constantly available, breaking changes are identified immediately and if you want to release immediately, it's very easy to do so.

### 3.5.1 16.1 Lab Overview

Within this lab you will see how you can use OpenShift Pipelines (a.k.a Tekton) to automate the delivery of decision services implemented with IBAMOE. In this lab you will see:

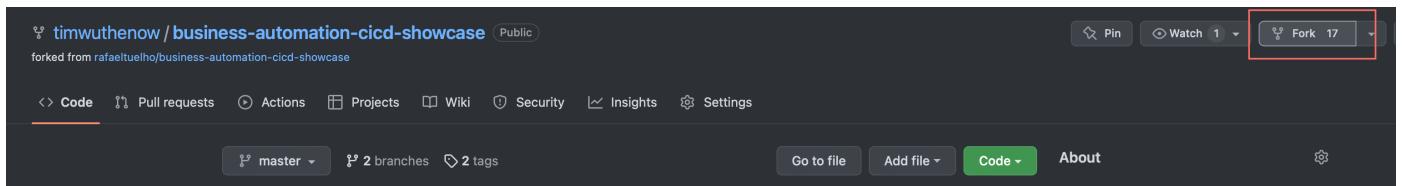
- The automation of repeatable decisions using the [DMN](#) specification;
- Decision tables implementation using XLS.
- Usage of the rules engine based on KIE Server and running on top of SpringBoot
- CI/CD Pipeline implemented using Tekton
- How to configure webhooks in your pipeline to deploy based on changes on a git repository
- Automated tests for decisions (with Test Scenarios ) that are considered during the pipeline execution
- Deployment with zero downtime with OpenShift rolling deployment strategy

### 3.5.2 16.2 Pre-requisites

- Java 8
- OpenShift 4.10+
- OpenShift Command Line client ( oc )
- [VSCode](#)
- [VSCode Business Automation Extension](#)

### 3.5.3 16.3 Installing on OpenShift

1. Fork this repository, [to get started](#).



2. Clone your fork to your local machine.

```
git clone https://github.com/${yourgithubuser}/business-automation-showcase.git
cd business-automation-showcase
```

### 1. Run the provisioning script (Linux/MacOS):

```
sh provision.sh
```

### 2. When the script runs, you will be presented with a prompt to enter a namespace name, which is the namespace that will be created for your environment to be deployed in.

```
business-automation-cicd-showcase % ./provision.sh
Input a namespace root - first letter lowercase
namespace: tim-demo
```

### 3. When this completes you will get a console with a few links, as well as an admin password to the Nexus environment that was created.

```
*****
Use this URL in your GitHub Webhook configuration for automatic deployment
http://el-ba-cicd-event-listener-tim-demo-rhdm-kieserver-cicd.openshift.io

Use this URL to access the front-end application:
http://decision-service-webclient-tim-demo-rhdm-kieserver-cicd.openshift.io

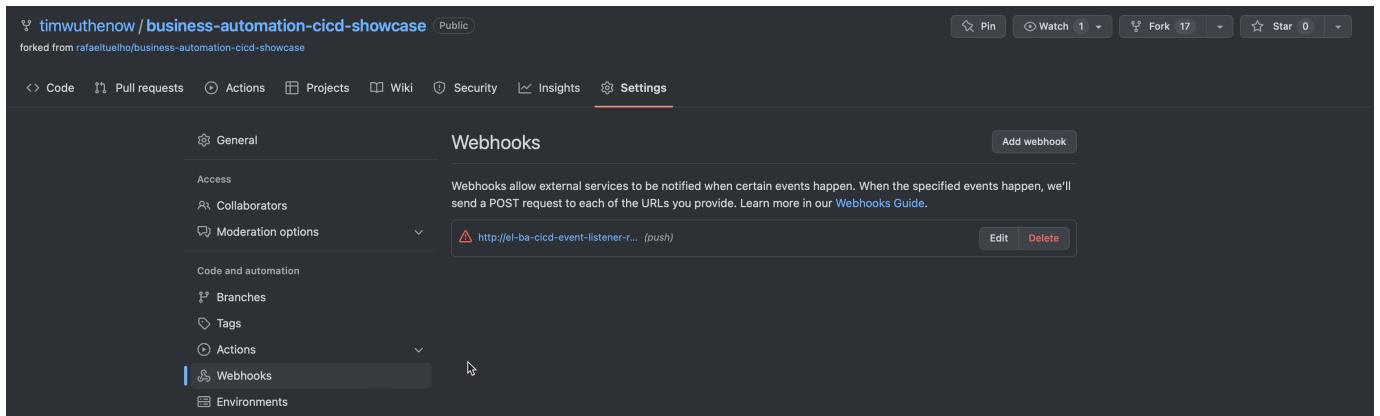
Use this URL to access the Nexus Repository:
http://nexus3-tim-demo-rhdm-kieserver-cicd.openshift.io

Use this password for admin access to Nexus 3:
long-strung-out-password-used-for-nexus-repo-admin

*****
```

## 3.5.4 16.4 Configuring the automatic deployment on GitHub

### 1. To configure the webhook for automated deployment, open your fork in your GitHub. Next, add a new webhook by opening "Settings -> Webhook -> Add webhook button".



### 1. Fill the form with the information below:

### 2. **Payload URL:** provided after the provisioning. You can also get it using the command:

```
echo $(oc get route el-ba-cicd-event-listener --template='http://{{.spec.host}}')
```

### 3. **Content type:** application/json

### 4. **Secret:** empty

### 5. **Which events would you like to trigger this webhook?**: Just the push event .

At this point, you should already have a fully automated integration and deployment lifecycle for the business application. Any changes pushed to your repository will trigger the pipeline in your OpenShift cluster.

### 3.5.5 16.5 Maven Repository Setup

Typically an environment would already have a Maven repository included, but if you wanted to set one up and use your own, this is a quick run through of what's required to get the Maven Repository to connect to the current location of the IBM Business Automation Open Edition 9.0 Maven artifacts which reside in the Red Hat Maven General Access Repository.

From the console log you got earlier, you would have received two messages with routes and password information for the Nexus repository location (the 3rd item) and the admin password (the commands to get these are found within the `provision.sh` file). If you hold Control (Linux/Windows) or Command (Mac) and click the link in VSCode, you will go to the location of your newly provisioned Nexus3 repository.

```
*****  
Use this URL in your GitHub Webhook configuration for automatic deployment  
http://el-ba-cicd-event-listener-tim-demo-rhdm-kieserver-cicd.  
south.containers.appdomain.cloud  
  
Use this URL to access the front-end application:  
http://decision-service-webclient-tim-demo-rhdm-kieserver-cicd.i  
south.containers.appdomain.cloud  
  
Use this URL to access the Nexus Repository:  
http://nexus3-tim-demo-rhdm-kieserver-cicd.  
us-south.containers.appdomain.cloud  
  
Use this password for admin access to Nexus 3:  
!1631767  
*****
```

By default, the Nexus repository includes a Mirror of [Maven Central](#) which anyone can publish to following [this documentation](#) and is the typical environment for Community releases, but enterprise releases of Open Source Software can be found here as well. The practice within Red Hat, has been to publish artifacts to its own repository found [here](#) as these are the enterprise releases and outside of community. No access is required to pull from this repository. We will configure our newly minted Nexus repository to have a Mirror of the Red Hat Maven General Access Repository so that for any product releases that are used, it will pull them from Red Hat Maven General Access Repository and publish them in the Nexus repository as a proxy. This is common in most enterprise environments instead of a direct connection to the repositories to offset downtime, have security checks, etc.

To setup the mirror follow these steps:

- From the route that was produced from the console output (or you can run `oc get route | grep nexus` where you will need to paste the output) and use that link to get to the Nexus admin screen.
- From here, using the admin password that was a part of your console output. Click `Sign In`. Alternatively the command to run is below.

```
echo "$(oc exec $(oc get pod -o template --template '{{range .items}}{{.metadata.name}}{{$n}}{{end}}' | grep nexus) -- cat /nexus-data/admin.password)"
```

The screenshot shows the Sonatype Nexus Repository Manager interface. At the top, there's a header with the logo and "Sonatype Nexus Repository Manager OSS 3.42.0-01". To the right are links for "Search components", "Sign in" (which is highlighted with a red border), and other navigation icons. Below the header, a sidebar on the left has "Browse" selected, with options for "Welcome", "Search", and "Browse". The main content area displays a "Welcome" message: "Welcome Learn about Sonatype Nexus Repository Manager" and a section titled "What's new in Nexus Repository 3.40 Pro?".

- Copying the password, login as `Admin` with the password from the previous command.

The screenshot shows a "Sign In" dialog box. It contains a message: "Your **admin** user password is located in **/nexus-data/admin.password** on the server." Below this, there are two input fields: one for the username "admin" and one for the password, which is represented by a series of dots. To the right of the password field is a key icon with a dropdown arrow. At the bottom are two buttons: a blue "Sign in" button and a white "Cancel" button.

- You will now have a Wizard popup to walk you through some setup, we're going to do very minimal changes: you can change the password (e.g. `password`). Typically in this development/throwaway environment, enabling anonymous access is the best way to do this, so select that option `Enable anonymous access`. Then click finish.

Please choose a password for the admin user 2 of 4

New password:  
.....

Confirm password:  
.....

[Back](#) [Next](#)

Configure Anonymous Access 3 of 4

Enable anonymous access means that by default, users can search, browse and download components from repositories without credentials. Please **consider the security implications for your organization**.

Disable anonymous access should be chosen with care, as it **will require credentials for all** users and/or build tools.

[More information](#)

Enable anonymous access  
 Disable anonymous access

[Back](#) [Next](#)

5. Now that the repository is ready to be configured, we are going to create a mirror of the Red Hat Maven General Access Repository so that we can pull resources through the proxy instead of direct. To do this, click the Gear icon (⚙) to open the settings screen.

6. From here click **Repositories** to view the current repositories and also to be able to create another.

7. Here you will see a list of default repositories. By default, Nexus will include the connection to *Maven\_Central*, but we'd also like to add the Red Hat Maven General Access Repository connection to pull the resources from this repository. To do so, click **Create Repository** to build a new repository.

Name	Type	Format	Status	URL	Health check	IQ Policy Vi...
maven-central	proxy	maven2	Online - Ready to Connect	<a href="#">copy</a>	Analyze	<a href="#">Edit</a>
maven-public	group	maven2	Online	<a href="#">copy</a>	<a href="#">Edit</a>	<a href="#">Edit</a>
maven-releases	hosted	maven2	Online	<a href="#">copy</a>	<a href="#">Edit</a>	<a href="#">Edit</a>
maven-snapshots	hosted	maven2	Online	<a href="#">copy</a>	<a href="#">Edit</a>	<a href="#">Edit</a>
nuget-group	group	nuget	Online	<a href="#">copy</a>	<a href="#">Edit</a>	<a href="#">Edit</a>
nuget-hosted	hosted	nuget	Online	<a href="#">copy</a>	<a href="#">Edit</a>	<a href="#">Edit</a>
nuget.org-proxy	proxy	nuget	Online - Remote Available	<a href="#">copy</a>	Analyze	<a href="#">Edit</a>

8. Scroll down and we're going to create a `maven2 (proxy)`. The reason we're doing a proxy is that we're going to allow our repository to reach out to the Red Hat Maven General Access Repository to retrieve artifacts it does not already have, but if it has previously pulled them, will be stored in the Nexus repository's cache. This way if there are new releases, you can pull as required, but are not taking the entire repository as required.

 **Repositories** /  **Select Recipe**

Recipe ↑
 apt (hosted) >
 apt (proxy) >
 bower (group) >
 bower (hosted) >
 bower (proxy) >
 cocoapods (proxy) >
 conan (proxy) >
 conda (proxy) >
 docker (group) >
 docker (hosted) >
 docker (proxy) >
 gitlfs (hosted) >
 go (group) >
 go (proxy) >
 helm (hosted) >
 helm (proxy) >
 maven2 (group) >
 maven2 (hosted) >
 maven2 (proxy) >

9. On the form that opens, you will add the following information (the name red\_hat\_ga needs to be exact to match the lab expectations):

10. **Name:** red\_hat\_ga

11. **What type of artifacts does this repository store:** Release

12. **Remote Storage:** <https://maven.repository.redhat.com/ga>

The screenshot shows the 'Create Repository' configuration page for a 'maven2 (proxy)' repository. The 'Name' field is set to 'red\_hat\_ga'. The 'Online' checkbox is checked. In the 'Maven 2' section, the 'Version policy' dropdown is set to 'Release'. Under 'Layout policy', 'Strict' is selected. The 'Content Disposition' dropdown is set to 'Inline'. In the 'Proxy' section, the 'Remote storage' field contains 'https://maven.repository.redhat.com/ga'. There are sections for 'Use the Nexus Repository truststore' (checkbox unchecked), 'Blocked' (checkbox unchecked), and 'Auto blocking enabled'.

13. Click **Create repository** at the bottom to deploy the proxy instance of the Red Hat Maven General Access Repository

The screenshot shows a confirmation dialog box with the title 'NOTE'. It contains sections for 'Negative Cache' (disabled) and 'HTTP' (disabled). At the bottom are two buttons: 'Create repository' (highlighted with a red border) and 'Cancel'.

14. Your repositories should now be setup and look similar to the below.

The screenshot shows a table titled 'Repositories' with the subtitle 'Manage repositories'. At the top left is a 'Create repository' button. To the right is a 'Filter' icon. The table has columns: 'Name ↑', 'Type', 'Format', 'Status', 'URL', 'Health check', and 'IQ Policy Vi...'. There are 8 rows of data:

Name ↑	Type	Format	Status	URL	Health check	IQ Policy Vi...
maven-central	proxy	maven2	Online - Ready to Connect	copy	Analyze	
maven-public	group	maven2	Online	copy		
maven-releases	hosted	maven2	Online	copy		
maven-snapshots	hosted	maven2	Online	copy		
nuget-group	group	nuget	Online	copy		
nuget-hosted	hosted	nuget	Online	copy		
nuget.org-proxy	proxy	nuget	Online - Remote Available	copy	Analyze	
red_hat_ga	proxy	maven2	Online - Ready to Connect	copy	Analyze	

## 3.5.6 16.6 First run of the Pipeline to configure Persistent Volume Claims for the workspaces

With the first run of the pipeline, you may have to run the pipeline manually. To do so, login to the OpenShift console for your project. You can see the console URL by running `oc whoami --show-console` and that will return the console link, Command/Control click it to open from VS Code.

Once logged in, make sure to go to our project that you created in the `provision.sh` script. Follow these steps to run the pipeline for the first time.

1. Make sure you change your project to the one you created in `provision.sh`. From the cluster home page, you can click `Pipelines` and then click `Pipelines`

The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes the Red Hat logo and the text "Red Hat OpenShift Container Platform". On the left, a sidebar menu lists various sections: "Administrator", "Home" (selected), "Overview" (highlighted with a blue bar), "Projects", "Search", "API Explorer", "Events", "Operators", "Workloads", "Networking", "Storage", "Builds", "Pipelines" (selected, highlighted with a red border), "Tasks", and "Triggers". A red box highlights the "Pipelines" section, which has a red number "2" and a hand cursor icon over it. The main content area is titled "Overview" and "Cluster". It features a "Getting started resources" section with links to "Set up your cluster" (checked), "Add identity providers", "Configure alert receivers", and "View all steps in documentation". Below this is a "Details" section with "View settings" and the following information:

<b>Cluster API address</b>	<input type="text"/>
<b>Cluster ID</b>	<input type="text"/>
<b>Provider</b>	IBMCLOUD
<b>OpenShift version</b>	4.10.32
<b>Update channel</b>	Not available

2. From the Pipelines screen, confirm your namespace matches what you created in the first part of the lab and after that, click the *kebab* icon to open the menu to Start Pipeline.

## Project: cicd-rhdm-kieserver-cicd ▾ 1 - confirm your namespace!

### Pipelines

[Create ▾](#)
[Pipelines](#) [PipelineRuns](#) [PipelineResources](#) [Conditions](#)

Name	Last run	Task status	
ba-cicd-pipeline	-	-	<span style="color: #0070C0;">PL</span> <span style="font-size: small;">More options</span> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <span style="color: #0070C0;">Start</span> <span style="color: red; font-size: 2em;">3</span>  <span>Add Trigger</span>  <span>Remove Trigger</span>  <span>Edit labels</span>  <span>Edit annotations</span>  <span>Edit Pipeline</span>  <span>Delete Pipeline</span> </div>

3. Once you click start pipeline, a form will come up to set some settings for the pipeline. These should only have to be the first time you run this pipeline. You will modify the ones below if you followed the previous section for the workspaces to align to the ConfigMap of `settings.xml` used by Maven to build and the Persistent Volumes used for workspace data (git clones and the Maven artifacts).

4. `maven-local-repo`: click the dropdown menu and select `PersistentVolumeClaim` and in the *Select a PVC* dropdown that appears, select `maven-repo-pvc`

**maven-local-repo \***

PersistentVolumeClaim

Select a PVC

Select a PVC

**PVC maven-repo-pvc** PVC source-workspace-pvc

5. `maven-settings`: click the drop down and select `Config Map` and select from the new dropdown `custom-maven-settings`

**maven-settings \***

Config Map

**Config Map \***

Select a ConfigMap

Select a ConfigMap

CM config-logging-triggers

CM config-observability-triggers

CM config-service-cabundle

CM config-trusted-cabundle

CM custom-maven-settings 

6. *shared-workspace*: click the dropdown menu and select `PersistentVolumeClaim` and in the *Select a PVC* dropdown, select `source-workspace-pvc`

**Start Pipeline****maven-local-repo \***

PersistentVolumeClaim

PVC maven-repo-pvc

**maven-settings \***

Config Map

**Config Map \***

CM custom-maven-settings

**Items** Add item**shared-workspace \***

PersistentVolumeClaim

PVC source-workspace-pvc

**Advanced options** 

Show credential options

Cancel

Start

7. With this complete you can click the `start` button to begin the pipeline

## Start Pipeline

maven-local-repo \*

shared-workspace \*

PersistentVolumeClaim

PVC source-workspace-pvc

Advanced options

Show credential options

Cancel Start

### 3.5.7 16.7 Testing GitHub and Pipeline integration

If you run this test, a new deployment should be triggered. The pipeline will deploy the decision service for the first time.

1. In your terminal, access your project folder.
2. Commit and push. You can use this empty commit sample if you need:

```
git commit -m "an empty commit to test the pipeline" --allow-empty
git push origin master
```

1. In OpenShift, access: "**Pipelines -> ba-cicd-pipeline -> Pipeline Runs**" and check the progress of your application deployment.

Pipelines > Pipeline details

### ba-cicd-pipeline

Actions ▾

- Details**
- Metrics
- YAML
- PipelineRuns
- Parameters
- Resources

**Pipeline details**

```

graph LR
    A[clean] --- B[git-clone]
    B --- C[prepare]
    C --- D[run-tests]
    D --- E[build-kjar]
    E --- F[build-service]
    F --- G[build-contain...]
    G --- H[deploy]
    
```

**Name**: ba-cicd-pipeline

**TriggerTemplates**: [ba-cicd-trigger-template](#)

**Namespace**: NS

**Labels**: No labels

**Annotations**: 0 annotations

**Created at**: Oct 10, 2022, 4:43 PM

**Owner**: No owner

**Tasks**:

- fix-repo-permission (clean)
- git-clone
- fix-repo-permission (prepare)
- mvn (run-tests)
- mvn (build-kjar)
- mvn (build-service)
- mvn-jkube (build-container-img)
- mvn-jkube (deploy)

**Workspaces**:

- maven-local-repo
- maven-settings
- shared-workspace

### 3.5.8 16.8 Using the web application

The web application allows you to interact with the deployed rules and decisions in a specific Decision Server (KieServer or Kogito runtime). To use the deployed web app to interact with the deployed decisions, first you need to set the KIE Server URL in the web app settings.

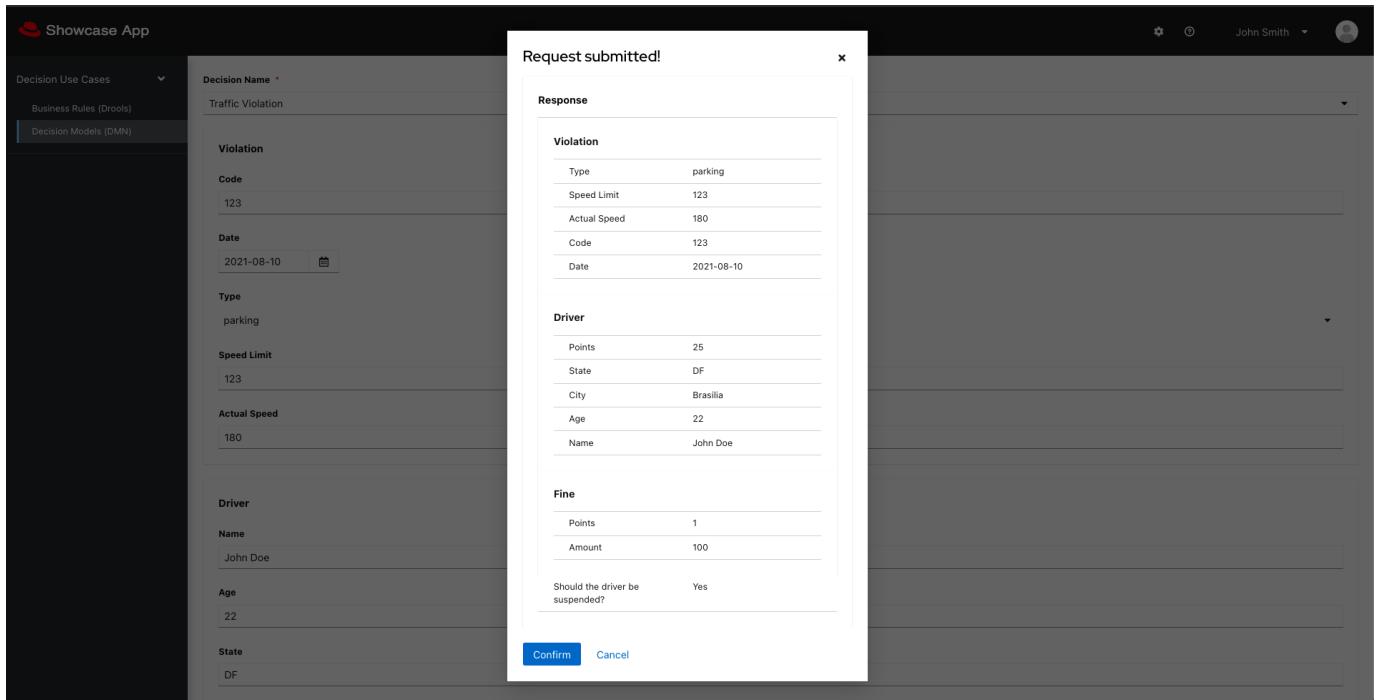
1. The deployed decision service is now deployed and accessible. Get your deployed KIE Server route. You can use the command:

```
echo "http://$(oc get route business-application-service-route -n rhdm-kieserver-cicd | awk 'FNR > 1 {print $2}')"/rest/server"
```

2. Open your web application. The URL was provided in the installation step. If you lost it, use the command

```
oc get route decision-service-webclient --template='http://{{.spec.host}}' -n rhdm-kieserver-cicd
```

1. In the web application, click on the settings icon on the top right corner. In the field **Kie Server Base URL**, insert KIE Server URL.
2. You can use the "Test Connection" button to validate the communication between the two services, then Save.
3. You should be able to test the available decisions and rules.



With this, the whole demo is now set up and ready to use.

NOTE: If you get interested in see how this webapp was developed the src code is available [here](#)

### 3.5.9 16.9 Extra information

The provisioning script `provision.sh` will:

- Create a new namespace called `rhdm-kieserver-cicd`
- Install OpenShift Pipelines
- Create the pipeline resources
- Deploy a front-end application that you can use to interact with the decision service once you deploy it.

At the moment there are 4 projects in this repository:

- `decisions-showcase`: Decision use cases using Business Rules (Drools) and Decision Logic (DMN)
- `business-application-service`: Spring Boot runtime based Kie Server exposing the API for Decisions provided with this Showcase demo
- `cicd`: Tekton Pipeline resources to implement a fully automated CI/CD pipeline for your Business Application Services
- `monitoring`: working in progress...

To see a detailed instruction on each service and each deployment processes (with images), check:

- [Provisioning and testing the CI/CD Pipeline](#)
- [Provisioning and testing the webclient application](#)

#### 16.9.1 Interested in Kogito?

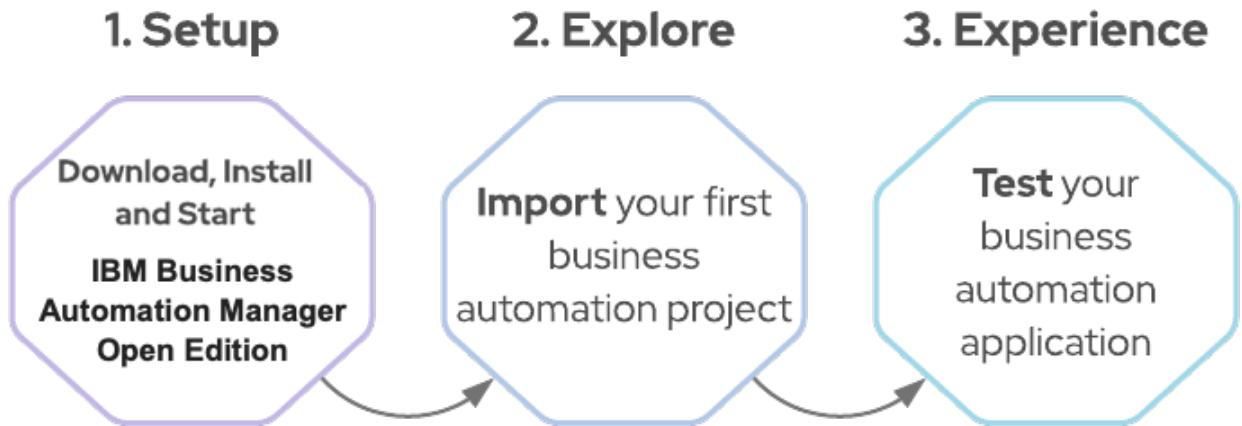
Check out the [kogito-quarkus](#) branch to see this same demo but using Kogito based Decision Services instead of KieServer.

Last update: 2023-12-06

## 3.6 Order Management

### 3.6.1 17. Getting started with IBAMOE

This guide shows you the experience of using IBM Business Automation Open Edition 8.0 to author, deploy, and execute your business automation applications. With three steps, this guide will get you from installation to deployment and testing of a business application:



We will install IBAMOE locally, and it will run on top of Red Hat JBoss EAP (a.k.a. WildFly). Once we have it up and running, we will import an existing application, so that we have an overview of some capabilities by exploring the tool and the project itself. Finally, we'll wrap up by deploying the project and testing it.

#### 17.1 Pre-requisites

We expect you to have installed in your machine:

- Java JDK 11 ( if you don't have it yet, you can download OpenJDK built by Red Hat <https://developers.redhat.com/openjdk-install> )
- GIT client (<https://git-scm.com/>)
- *IBM Business Automation Open Edition 8.0 Installation Demo*: **NOTE:** You should use this installer to quickly install EAP, PAM and pre-configure the environment and user access you'll need. `$ git clone https://github.com/timwuthenow/ibamoe-setup.git`

You should now have successfully installed IBM Business Automation Open Edition 8.0.

You have two key components deployed in your Red Hat EAP right now: **Business Central** and **KIE Server**.

**Business Central** is the component that allows you to develop business assets like processes and decisions, to manage projects, build and package them. Finally, you can deploy it in KIE Server.

**KIE Server** is a lightweight engine capable of executing business assets like processes, cases and decisions. It can be easily integrated with your services, for example via REST or JMS.

Luckily, IBM Business Automation Open Edition 8.0 comes with a number of out-of-the-box template and example applications that can be used to quickly build and deploy a process microservice.

## 17.2 Explore the Asset

Let's start by accessing Business Central.

1. In your browser, access Business Central by navigating to <http://localhost:8080/business-central>

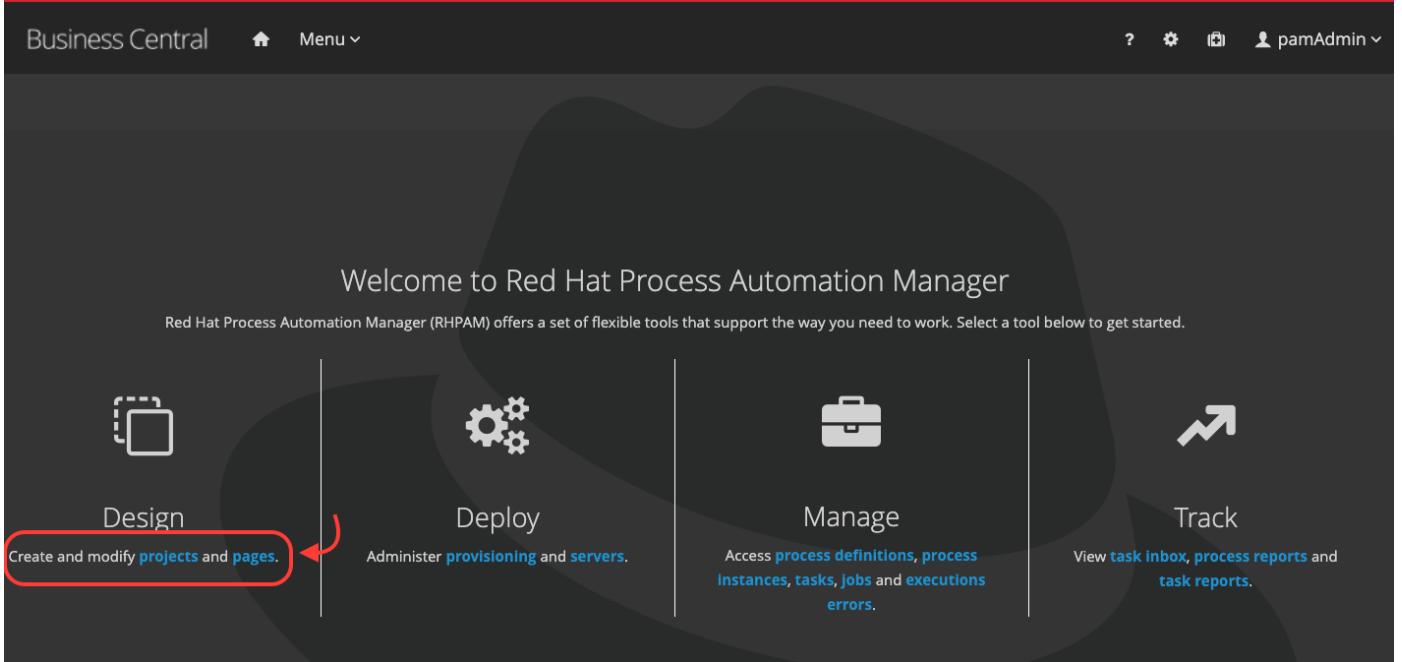
2. Log in with the credentials:

3. User: **bamAdmin**

4. Password: **ibmpam1!**

 INFO: If you're using the Linux environment on Skytap use *pamadmin:pamadm1n* as the username password

5. Click on "Design, create and modify projects and pages"

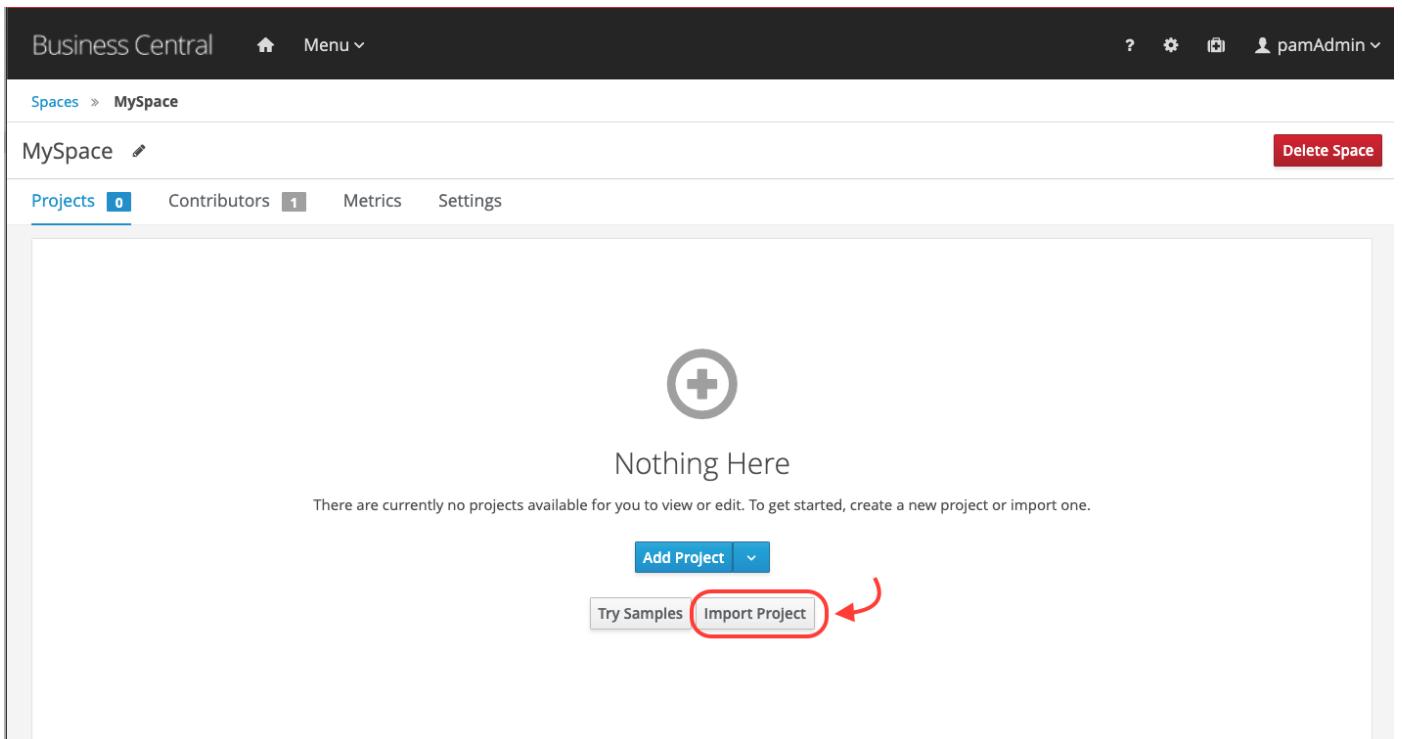


Welcome to Red Hat Process Automation Manager

Red Hat Process Automation Manager (RH-PAM) offers a set of flexible tools that support the way you need to work. Select a tool below to get started.

- Design**   
Create and modify [projects](#) and [pages](#).
- Deploy**   
Administer [provisioning](#) and [servers](#).
- Manage**   
Access [process definitions](#), [process instances](#), [tasks](#), [jobs](#) and [executions](#) [errors](#).
- Track**   
View [task inbox](#), [process reports](#) and [task reports](#).

6. Select "MySpace", and next, click on "Import Project":



Business Central  Menu    pamAdmin 

Spaces  MySpace 

MySpace 

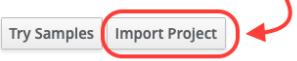
Projects  Contributors  Metrics Settings



Nothing Here

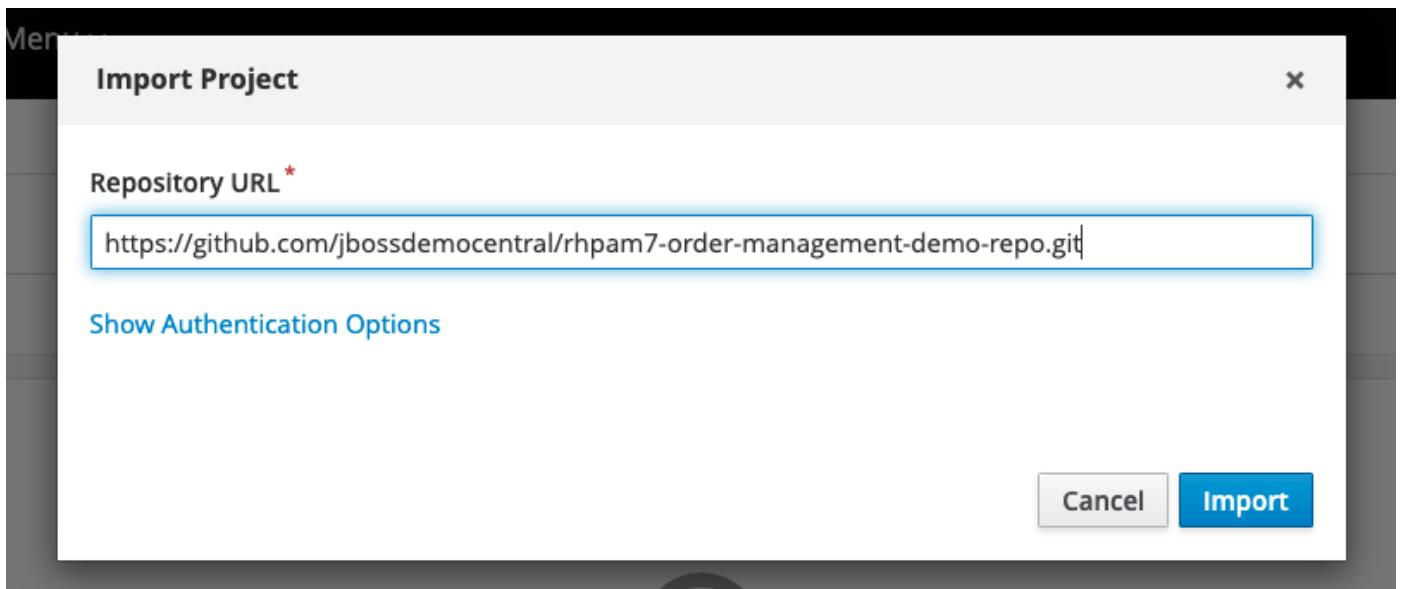
There are currently no projects available for you to view or edit. To get started, create a new project or import one.

[Add Project](#) 

[Try Samples](#) [Import Project](#) 

7. Insert the following repository URL, and click on Import.

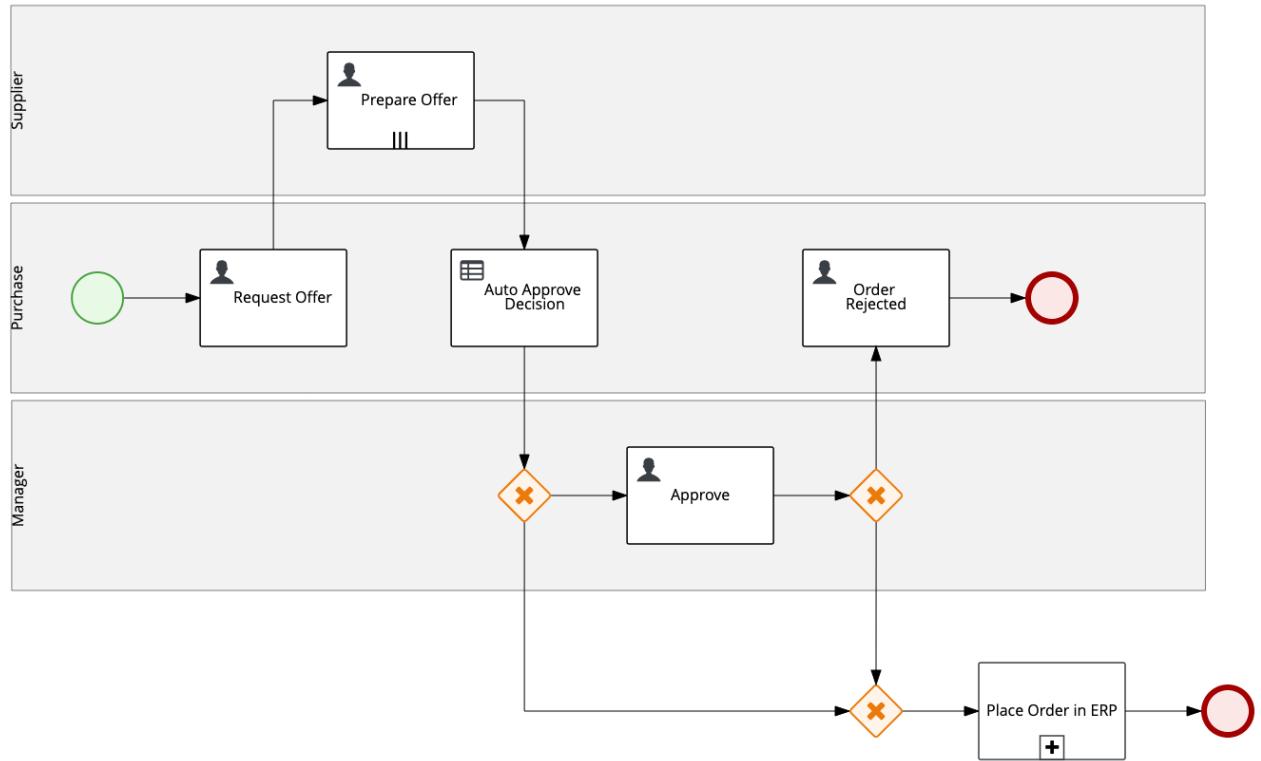
```
https://github.com/jbossdemocentral/rhpam7-order-management-demo-repo.git
```



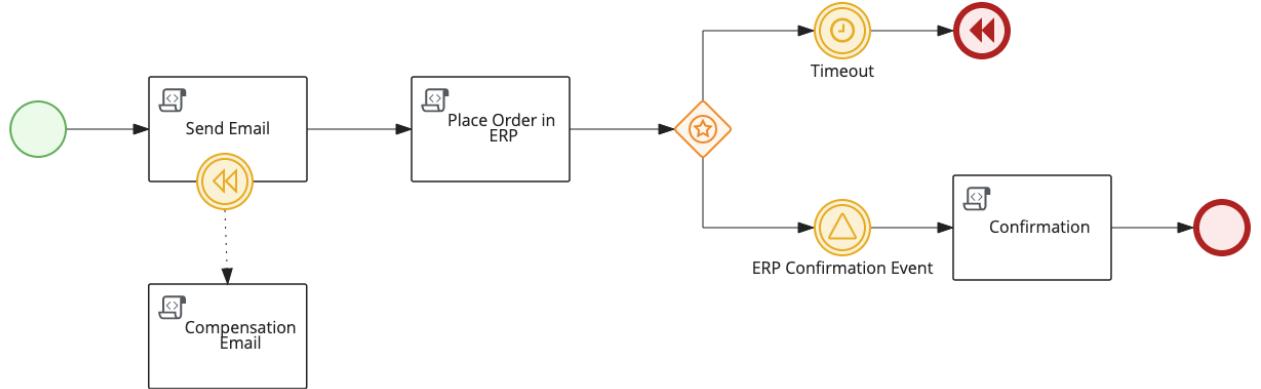
8. Select the Order-Management project and click on OK.

9. Once the project has been imported, notice it has 27 assets. Click on the filter button "All" and select Process.

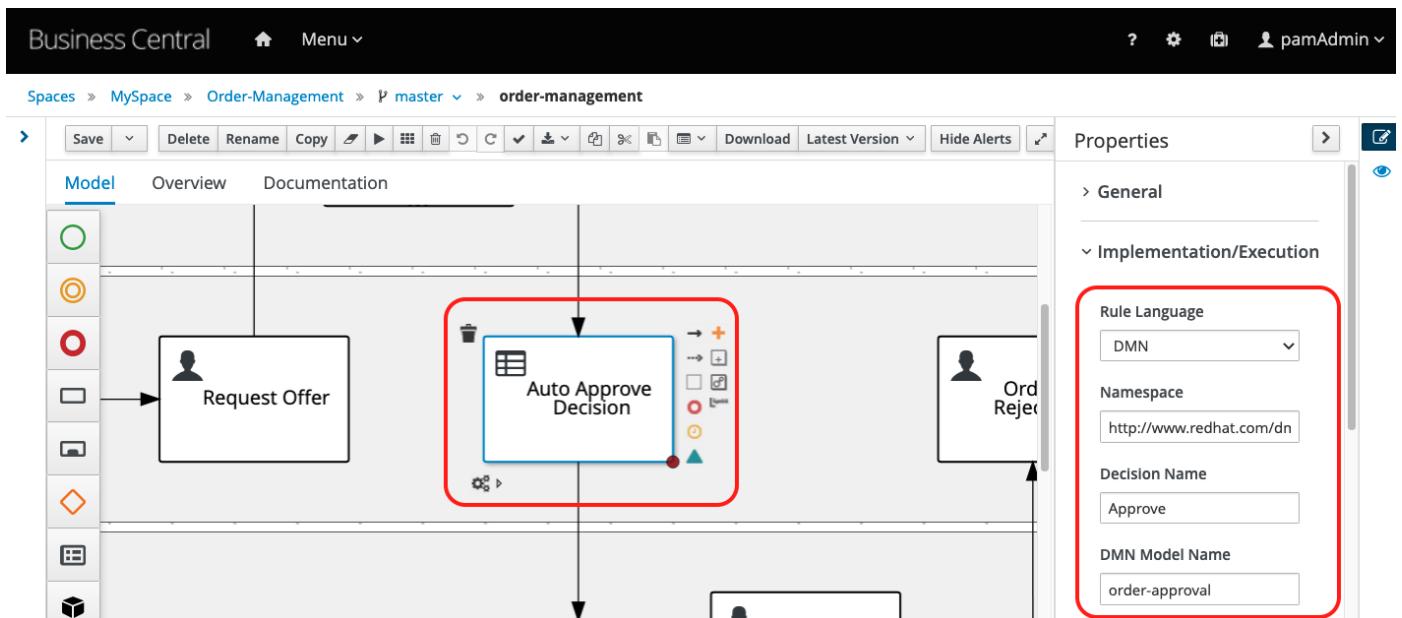
10. Open the order-management process. This is the automated process that determines the approval or denial of an order request. As you see below, it is implemented with the BPMN2 standard.



11. The final element of this process, is a sub-process "Place Order in ERP". This subprocess includes advanced bpmn2 modeling concepts like compensation and event based gateways. Have in mind that PAM supports the modeling of advanced flows using the bpmn2 specification, but don't worry if you don't fully get what is happening in this subprocess.



12. Notice this process tasks are aggregated in three lanes: Manager, Purchase and Supplier. The approval decision will be made based on multiple authors, but, in this process we even have the support of automated decision. The automated decision is made on the node "Auto Approve Decision", that references a DMN Model that is also part of this business project.

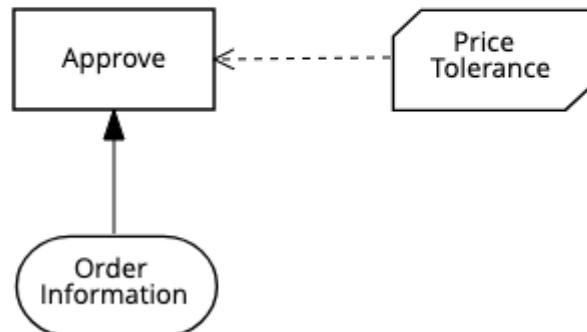


13. Close the process modeler. Now, filter the assets by Decision. You should see a Test Scenario and a DMN model.

The screenshot shows the Business Central asset management interface. At the top, there's a navigation bar with "Business Central" and "Menu". Below the navigation bar, the path is "Spaces > MySpace > Order-Management > master". There are buttons for "Test", "Build", "Deploy", and "Hide Alerts". The main area is titled "Order-Management" and shows a list of assets. The "Assets" tab is selected, showing 27 items. A search bar and a filter dropdown are present. The filter dropdown has "Decision" selected, indicated by a red arrow. The list contains two entries:

	Name	Type	Last modified	Created
	val-test	Test Scenarios	27 weeks ago	27 weeks ago
	val	DMN	37 weeks ago	37 weeks ago

14. Open the order-approval. It is a simple decision model that can define the "Approve" decision based on the data input "Order Information" and on the "Price Tolerance" business rules.



15. Now, close the decision asset. In your project page, click on the Deploy button. Business Central will trigger the build of this maven project, that will be packaged in a KJAR (the deployment unit which contains the assets) and will be deployed on the KIE server.

Business Central    Home    Menu ? ⚙️ 🖨️ 👤 pamAdmin

Spaces > MySpace > Order-Management > master

Order-Management

Assets 27 Change Requests 0 Contributors 1 Metrics Settings

All < 1 > of 2 Import Asset Add Asset

Approve-OrderInfo	Forms	Last modified 30 weeks ago	Created 134 weeks ago
Approve-taskform	Forms	Last modified 30 weeks ago	Created 134 weeks ago
DecisionTask	Work Item Definitions	Last modified 71 weeks ago	Created 71 weeks ago

16. Once the build and deployment has finished, you'll see a successful deployment message. Click on the "View deployment details" link.

The screenshot shows the Business Central interface for the Order Management project. At the top, there's a navigation bar with 'Business Central', a home icon, 'Menu', and a user 'pamAdmin'. Below the navigation is a breadcrumb trail: 'Spaces > MySpace > Order-Management > master'. A green banner at the top right says 'Deploy to server configuration successful and container successfully started.' with a 'View deployment details' button, which is highlighted with a red rectangle. Below the banner, the page title is 'Order-Man' and the main navigation tabs are 'Assets' (27), 'Change Requests' (0), 'Contributors' (1), 'Metrics', and 'Settings'. There are filters ('All'), a search bar, and pagination ('1-15 of 27', '1 of 2'). At the bottom right are 'Import Asset' and 'Add Asset' buttons.

17. The page will show a running “default-kieserver” with the “order-management\_1.1-SNAPSHOT” container deployed.

Our business project is now available to be consumed by client applications! Let's have a look at how we can consume this business application.

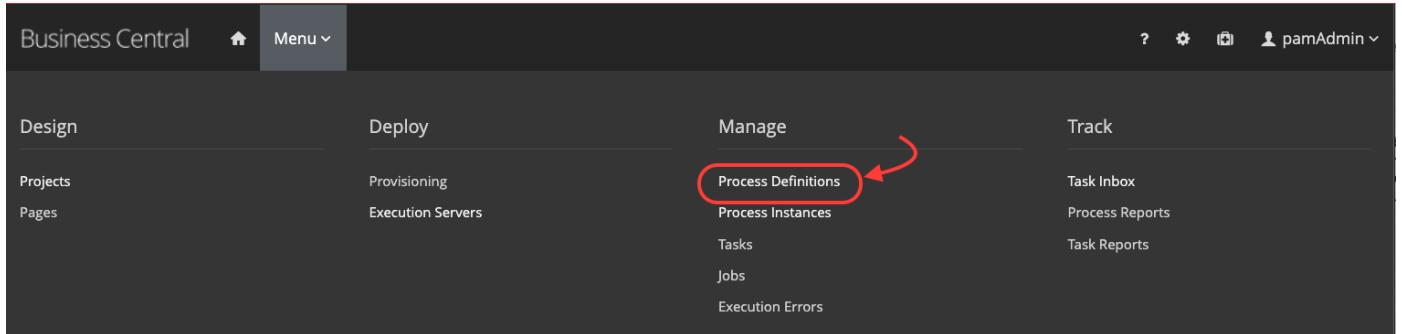
### 17.3 Experience

The engine, KIE Server, is the service which exposes the business project and also the one we use when integrating with client applications. It comes with a Swagger UI that allows us to test the RESTful endpoints of the engine and consume rules deployed on it.

Another way to consume our business project is to use Business Central UI to interact with the engine and test our business assets.

For this hello world, let's use Business Central process and task management capabilities.

1. In Business Central, let's open the Menu in the top bar and navigate to "Process Definitions"



2. We can see three different process definitions. We'll start a new process instance based on the "order-management" process. Click on the actions kebab, and select "Start"

Name	Version	Deployment	Actions
order-management	2.0	order-management_1...	<span>⋮ Start View Process Instances ⋮</span>
OM-demo-init	1.1	order-management_1...	<span>⋮</span>
erp-integration	1.0	order-management_1....	<span>⋮</span>

3. The form that opened is also part of our business process and we can customize it if needed. For now, let's just fill in the data required to start our process instance, and click the "Submit" button.

4. Item Name: Laptop Dell XPS 15

5. Urgency: Medium

**order-management**

- ✓ Correlation key
- ✓ Form

### Order Information

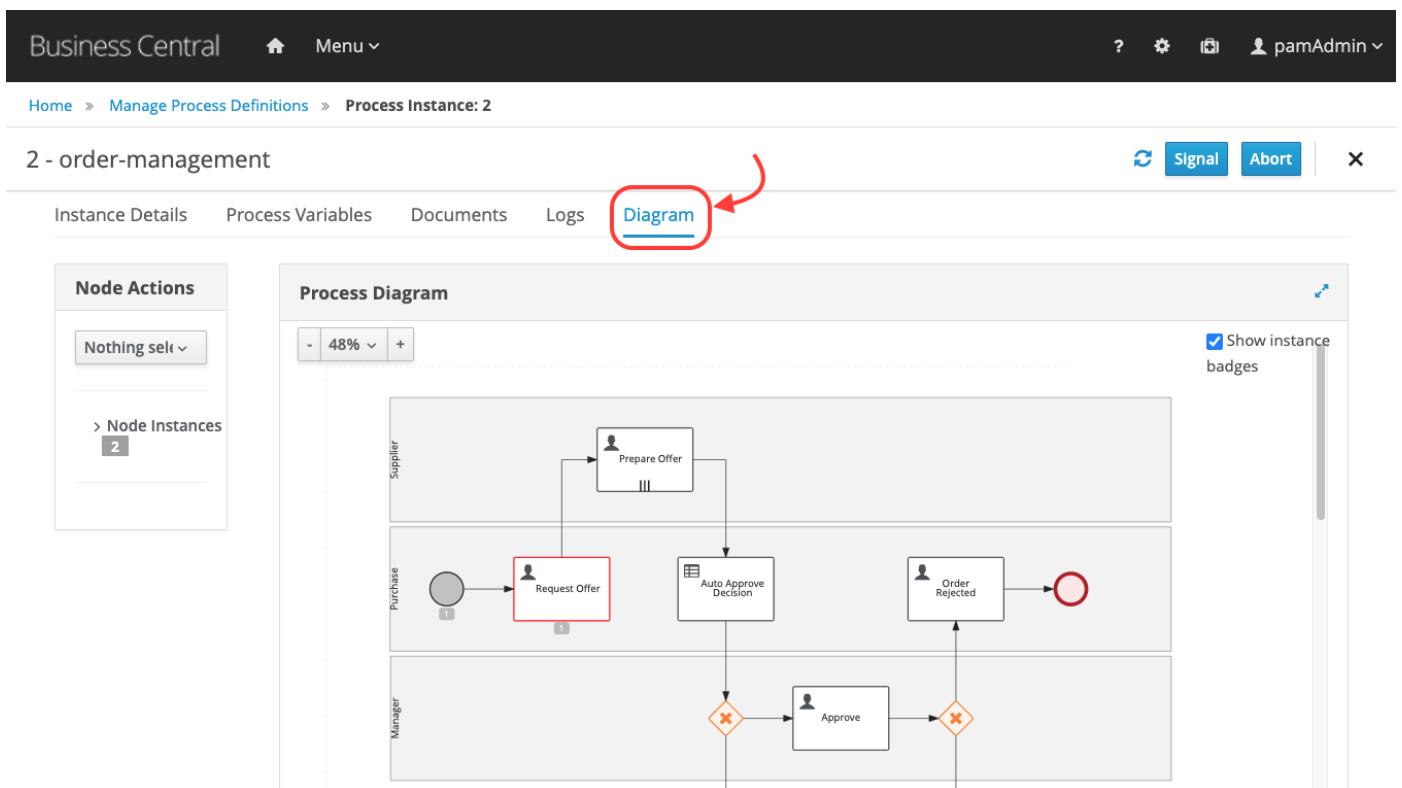
**Item name\***

**urgency\***

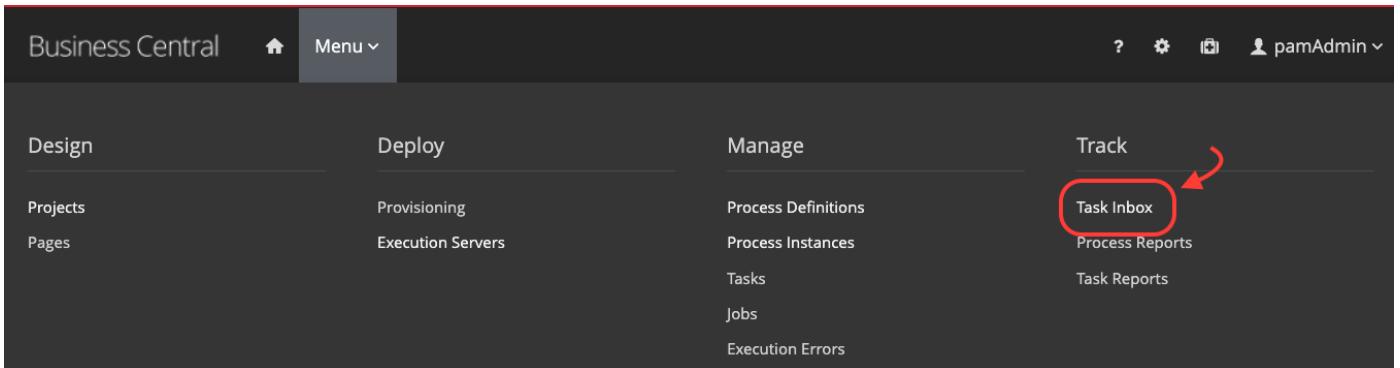
Low  Medium  High

**Submit**

6. A new process instance will start in the engine. In order to visualize the current status, click on "Diagram".



7. Notice we currently have a Human Task named "Request Offer" waiting for human intervention. Now, let's work on this task. In the Menu, access the "Task Inbox":



8. In the list you should see a list of tasks you have permission to see and work on. Let's claim the Request Offer task to our user, and start working on it. Click on the kebab and select the "Claim and Work" option:

The screenshot shows the 'Task Inbox' page. On the left, there is a 'Filters' sidebar with a 'Status' section containing several filter options. The 'InProgress' and 'Ready' checkboxes are checked. The main area shows a table of tasks. One task, 'Request Offer', is selected. A context menu is open over this task, with the 'Claim and Work' option highlighted by a red circle and arrow. Other options visible in the menu include 'Claim', 'Start', 'View process', and 'Edit'.

	Task	Process Definition Id	Status	Create	View process
<input type="checkbox"/>	Request Offer	Order-Manage...	Ready	07-Nov-2020 22...	

9. You'll see the task data available for your analysis, as a knowledge worker - someone responsible for executing the task.

10. Click on the blue "Start" button to start working on the task. Based on this offer, we'll define our reply. Inform the following data and click on the blue "Complete" button:

11. Category : optional

12. Target Price: 250

## 13. Suplier list: supplier 1

Business Central Home > Task Inbox > Task: 1

### 1 - Request Offer

**Work** Details Assignments Comments Admin Logs

**Order Information**

order id	item name	urgency
1	Printer HP LaserJet Pro M402n	low

category\*  
 basic  optional

target price

suppliers list

**Save** **Release** **Complete**

14. According to our process, a new task will be created for the suppliers. The supplier should provide an offer - so let's do it. Still on the task list, claim and work the task "Prepare Offer":

Business Central Home > Task Inbox

### Task Inbox

Active filters: Status: Ready, Reserved, InProgress X

Save Filters | Clear All

Task	Process Definition Id	Status	Created On
Prepare Offer	Order-Management.order-manag...	Ready	07-Nov-2020 22:20:29

10 Items ▾ 1 of 1

Claim and Work

15. Click "Start" blue button, inform any date, and the best offer as **1000**. Click on complete.

Business Central Home > Task Inbox > Task: 3

### 3 - Prepare Offer

**Work** Details Assignments Comments Admin Logs

supplier1

**Order Information**

item name*	urgency*
Printer HP LaserJet Pro M402n	<input checked="" type="radio"/> Low <input type="radio"/> Medium <input type="radio"/> High

**Your Best Offer**

delivery date\* best offer\*

11/09/2020 08:20  1000

**Save** **Release** **Complete**

16. At this point, the automatic approval was already taken, and our request was not automatically approved. You can confirm this by visualizing the process instance. On the kebab, select "View Process"

The screenshot shows the Business Central Task Inbox interface. On the left, there are filters for tasks based on status: Completed, Created, Error, Exited, Failed, InProgress (selected), Obsolete, Ready (selected), Reserved, and Suspended. The main area displays a table with columns: Task, Process Definition Id, Status, and Create. One row is selected, showing the task 'Approve' from the process 'Order-Manage...', status 'Ready', and creation date '07-Nov-2020 22...'. To the right of the table, a context menu is open with options: Claim, Claim and Work, and View process. The 'View process' option is circled in red.

17. You'll be redirected to the list of process instances. Select the process instance with id 1, and then, choose the "Diagram" option:

The screenshot shows the Business Central Manage Process Instances interface. The top navigation bar includes Home, Manage Process Instances, and Process Instance: 1. Below, tabs for Instance Details, Process Variables, Documents, Logs, and Diagram are present, with Diagram selected. On the left, a sidebar shows Node Actions (Nothing selected) and Node Instances (7). The main area displays a Process Diagram titled 'Supplier' with three participants: Purchase and Manager. The flow starts with a 'Request Offer' task in Purchase, which sends a message to 'Prepare Offer' in Supplier. 'Prepare Offer' then sends a message to 'Auto Approve Decision' in Purchase. 'Auto Approve Decision' has two outgoing paths: one to 'Approve' in Manager, and another to 'Order Rejected' in Manager. 'Approve' has two outgoing paths: one to 'Order Rejected' and another back to 'Auto Approve Decision'. A checkbox 'Show instance badges' is checked in the top right corner of the diagram area.

At this point, you have learned how you manage processes and tasks using Business Central. You know how to start new process instances, how to interact with the process tasks and how to complete them.

What about finishing this process by your own? Following the same idea, In Business Central, you can reprove the request, reject the order and reach the end of this process instance.

#### 17.4 Conclusion

Congratulations, you successfully concluded a Hello World in IBM Business Automation Open Edition 8.0.

In this guide, we installed Red Hat PAM, imported a project directly from GitHub, checked out the a process definition modeled and an automation decision.

We wrapped up our tutorial by deploying and testing our services using Business Central UI. If you want to know more about the Order Management demo, we recommend you take a look at the project's instructions at the github repository located [here](#).

---

Last update: 2023-12-06

### 3.6.2 18. Overview of Order Management Process

This is a Process Management lab in which you will implement an Order Management process. The process will use BPMN2 constructs like **Swimlanes**, **User Tasks**, **Gateways**, combined with decision-based routing based on a **DMN Model** (Decision Model & Notation). It also introduces more dynamic concepts of the IBM Business Automation Open Edition 8.0 process engine, like dynamic assignments of tasks based on process instance data.

#### 18.1 Goals

- Create an Order Management project in IBM Business Automation Open Edition 8.0
- Define and create the process' domain model using the platform's Data Modeller.
- Implement an order management process in the process designer
- Implement decision logic in a DMN model.
- Create forms with the platform's Form Modeller.
- Deploy the project to the platform's Execution Server.
- Execute the end-to-end process.

#### 18.2 Pre-reqs

- Successful completion of the *Environment Setup Lab* or
- An existing, accessible, DM/PAM 7.3+ environment.

#### 18.3 Problem Statement

In this lab we will create an Order Management process that manages the process of ordering a new phone or laptop.

- Start the process by providing the order information.
- The supplier sends an offer stating the expected delivery date and its best offer.
- Depending on the urgency of the order and the price, the order can be auto-approved by a DMN decision.
- If the order is not auto-approved, the manager needs to complete an approval step.

---

Last update: 2023-12-06

### 3.6.3 19. Lab Walk Through

---

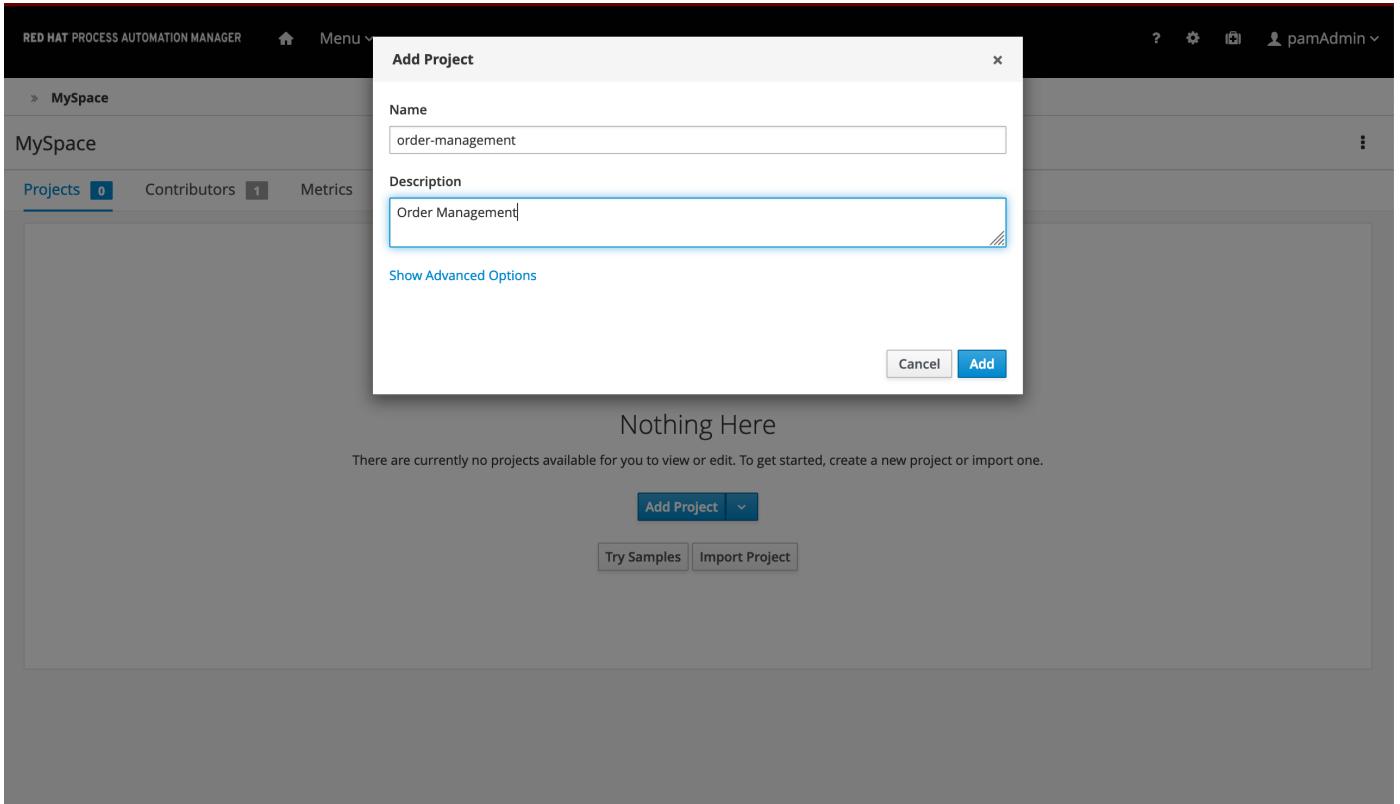
#### **19.1 Create a Project**

To define and deploy a business process, we first need to create a new project in which we can store the BPMN2 model, our domain model and the forms required for user interaction. To create a new project:

1. Navigate to **Business Central**
2. Login to the platform with the provided username and password.
3. Click on **Design** to navigate to the Design perspective.

The screenshot shows the Red Hat Process Automation Manager interface. At the top, there is a dark header bar with the text "RED HAT PROCESS AUTOMATION MANAGER" on the left, a home icon, and a "Menu" dropdown on the right. The right side of the header also shows a user profile for "pamAdmin". Below the header is a navigation bar with "MySpace" selected. Underneath the navigation bar, there are three tabs: "Projects" (0), "Contributors" (1), and "Metrics". The main content area is titled "Nothing Here" and features a large circular icon with a plus sign in the center. Below this icon, the text "Nothing Here" is displayed, followed by the message "There are currently no projects available for you to view or edit. To get started, create a new project or import one." At the bottom of the content area are two buttons: "Add Project" (blue) and "Try Samples" (grey). To the right of "Add Project" is a dropdown arrow. Below these buttons is another "Import Project" button.

4. In the Design perspective, create a new project. If your space is empty, this can be done by clicking on the blue **Add Project** button in the center of the page. If you already have projects in your space, you can click on the blue **Add Project** icon at the top right of the page.
5. Give the project the name `order-management`, and the description "Order Management".



With the project created, we can now start building our solution.

## 19.2 Lab Walk through

In this section we will first create the Domain Model within Business Central and then walk through the creation of the assets associated with the Process.

### 19.2.1 THE DOMAIN MODEL

The business process will collect and carry data through the execution of the process. This data is stored in a data model or domain model. In this lab, we collect two types of data:

- **OrderInfo** : contains information about the order, like the item and the price.
- **SupplierInfo** : contains information about the supplier, like the name and the expected delivery date.



1. In your project, click on the *Add Asset* button in the middle of the screen.

The screenshot shows the Business Central interface with the title 'Business Central' and a menu bar with 'Menu'. Below the menu, the path 'Spaces > OM-Lab-Space > order-management > master' is visible. The main content area is titled 'order-management' and shows a large button with a plus sign containing the text 'Nothing Here'. Below this button, a message states 'There are currently no assets available for you to view or edit. To get started, create a new asset or import one.' At the bottom of the screen, there are two buttons: 'Import Asset' and 'Add Asset'.

2. In the drop-down menu in the upper-left corner, select *Model*. Click on the *Data Object* tile.

The screenshot shows the 'Add Asset' screen with a dropdown menu set to 'Model'. Below the dropdown, there are two tiles: 'Data Object Model' and 'Enumeration Model'. Each tile has a circular icon and a descriptive label.

3. Give the *Data Object* the name `orderInfo`. Leave the package set to default.

**Create new Data Object**

**Data Object\***  
OrderInfo

**Package**  
com.om\_lab\_space.order\_management

**JPA**  
 Persistable i

**+ Ok** **Cancel**

4. Add the following fields to the `OrderInfo` data object:

Identifier	Label	Type
item	item name	String
urgency	urgency	String
targetPrice	target price	double
managerApproval	approved	Boolean

5. When you've added the fields, save the data object by clicking on the *Save* button in the top menu.

6. Use the `_breadcrumb`` navigator at the top-left of the screen to navigate back to our `order-management` project.

7. Click on the blue *Add Asset* button in the top-right corner and create a new *Data Object*

8. Give it the name `SupplierInfo`

**Create new Data Object**

**Data Object\***  
SupplierInfo

**Package**  
com.om\_lab\_space.order\_management

**JPA**  
 Persistable i

**+ Ok** **Cancel**

9. Give the `SupplierInfo` object the following fields:

Identifier	Label	Type
offer	best offer	double
deliveryDate	delivery date	Date
user	user	String

10. We're done creating our data model.

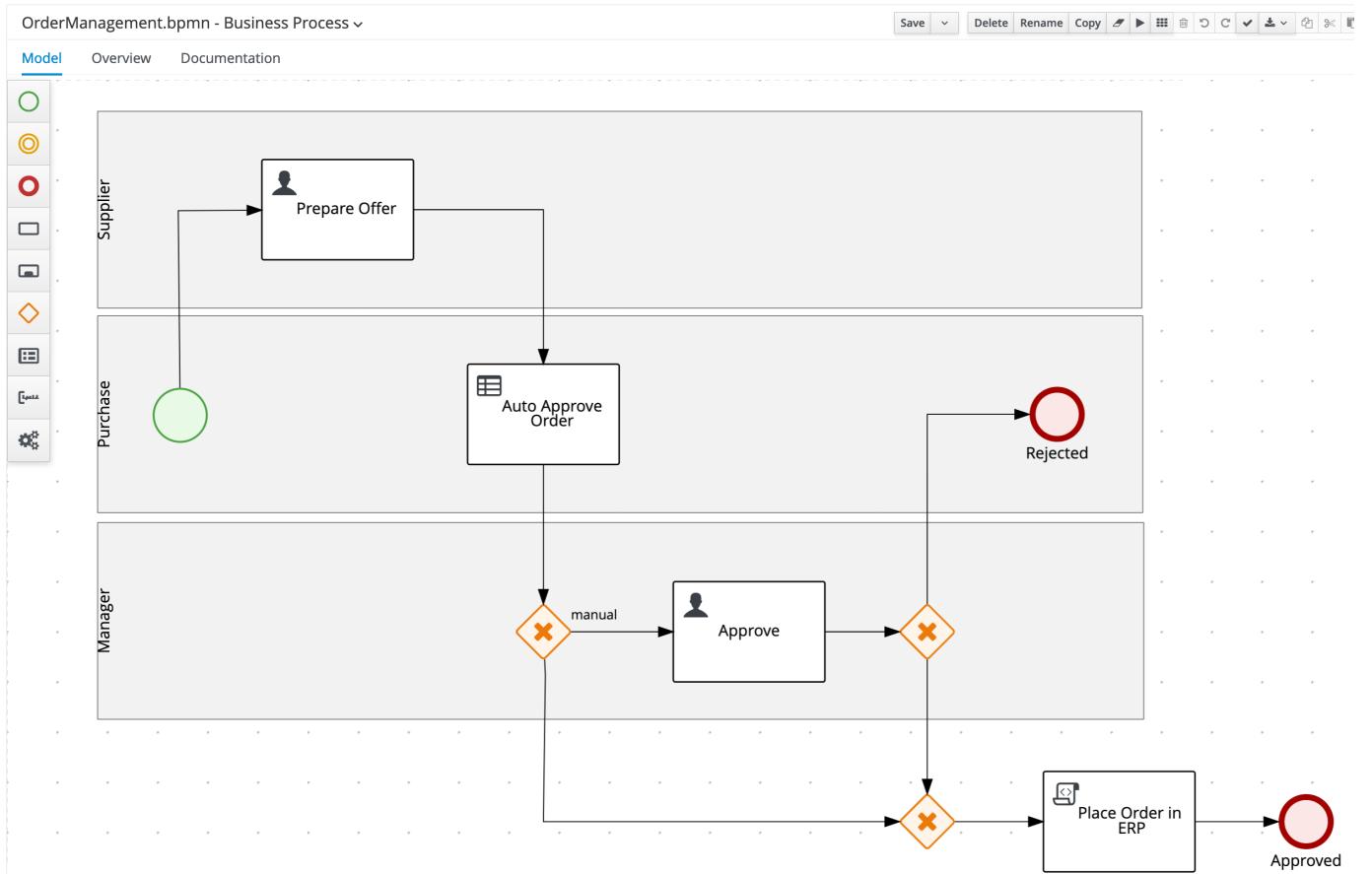
The screenshot shows the Business Central interface with the following details:

- Header:** Business Central, Menu, Help, Settings, User: parmAdmin.
- Breadcrumb:** Spaces > OM-Lab-Space > order-management > master.
- Page Title:** order-management
- Toolbar:** Test, Build, Deploy, Hide Alerts, More.
- Filter Bar:** Assets (2), Change Requests (0), Contributors (1), Metrics, Settings.
- Search Bar:** All (dropdown), Search icon.
- Table:**
  - Orderinfo:** Data Objects, Last modified today, Created today.
  - Supplierinfo:** Data Objects, Last modified today, Created today.
- Pagination:** 1-2 of 2, Page 1 of 1, Import Asset, Add Asset.

We can now start with our process design.

### 19.3 Process Design

With the domain model defined, we can now sketch out the main flow of the process, the actors, the user task nodes and the required automation decisions.



1. Create a new Business Process asset. Name it `OrderManagement`. You can do this by clicking `Add an Asset` and then selecting `Business Process` and then setting the name as `OrderManagement`.
2. When the process designer opens, scroll down in the property panel on the right side of the screen, until you see the section `Process Data`.
3. Expand the `Process Data` section and add the following 3 `Process Variables` by clicking on the `+` sign.

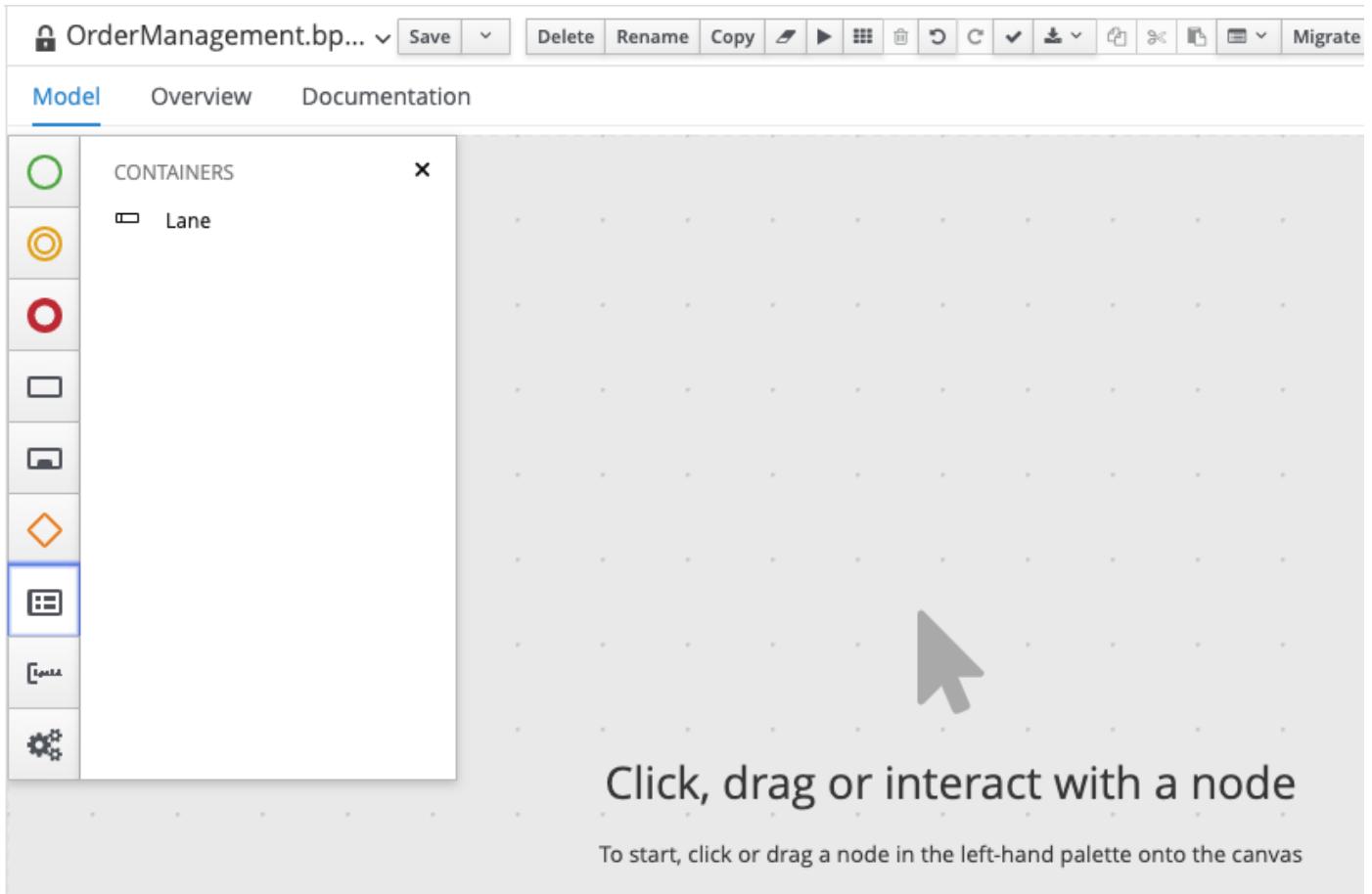
Name	Data Type
orderInfo	OrderInfo
supplierInfo	SupplierInfo
approved	Boolean

The screenshot shows the Camunda BPM process designer interface. On the left is a vertical palette with various icons. In the center is a workspace with a single horizontal boundary box. To the right is a property panel. The 'Process Data' section is expanded, showing a table of process variables:

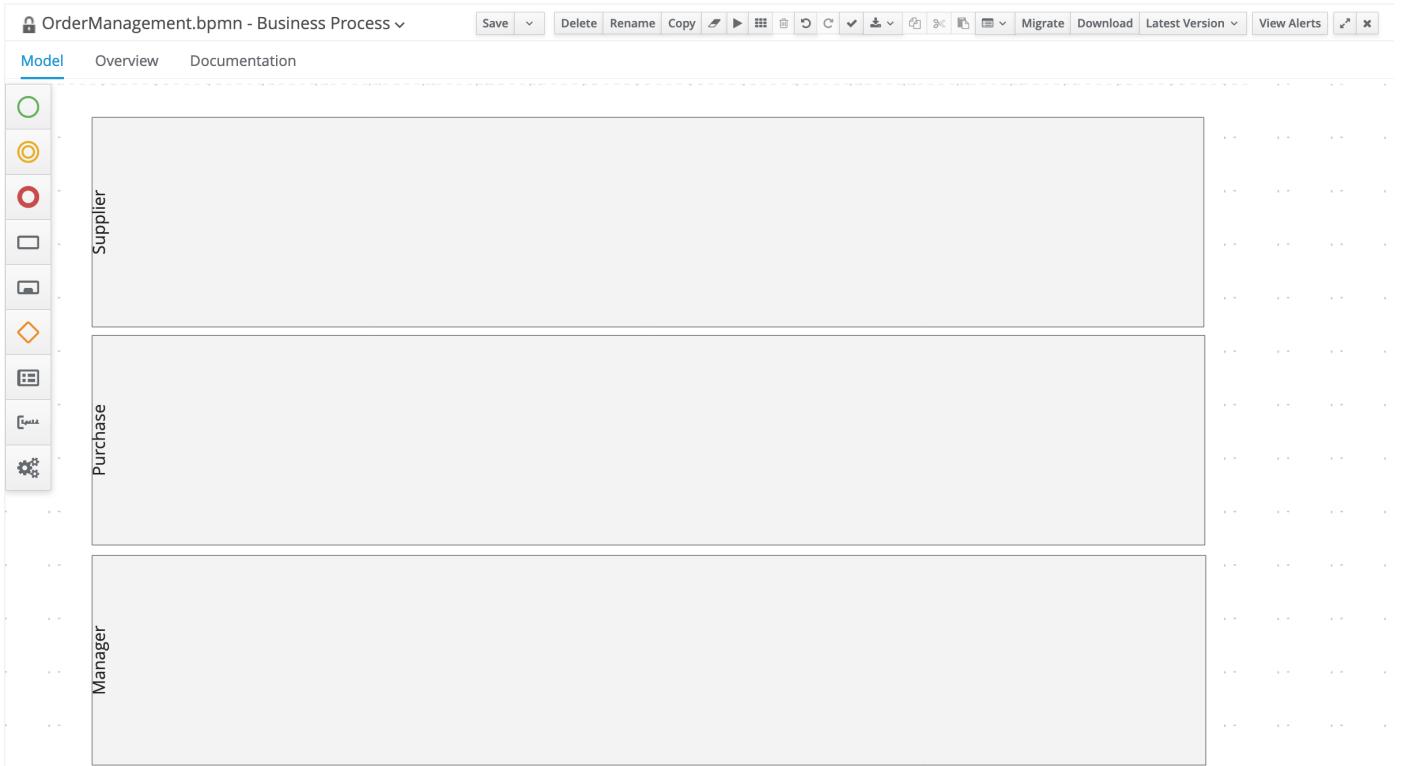
Name	Data Type	KPI	+
orderInfo	OrderInfo [com.or]	<input type="checkbox"/>	
supplierInfo	SupplierInfo [com.]	<input type="checkbox"/>	
approved	Boolean	<input type="checkbox"/>	

#### 19.3.1 PREPARE OFFER

1. In the palette on the left-side of the editor, select the `Lane` component:



2. Create the following 3 swimlanes: **Supplier , Purchase , Manager**



3. Create the **Start Event** node in the **Purchase** swimlane.

4. Create the `Prepare Offer` **User Task** node in the `Supplier` swimlane and connect it to the **Start Event** node. Set the following properties on the node via the properties panel on the right side of the screen:

a. Task Name: `PrepareOffer`

b. Subject: `Prepare Offer for #{orderInfo.item}`

c. Actors: `#{supplierInfo.user}`

d. Input:

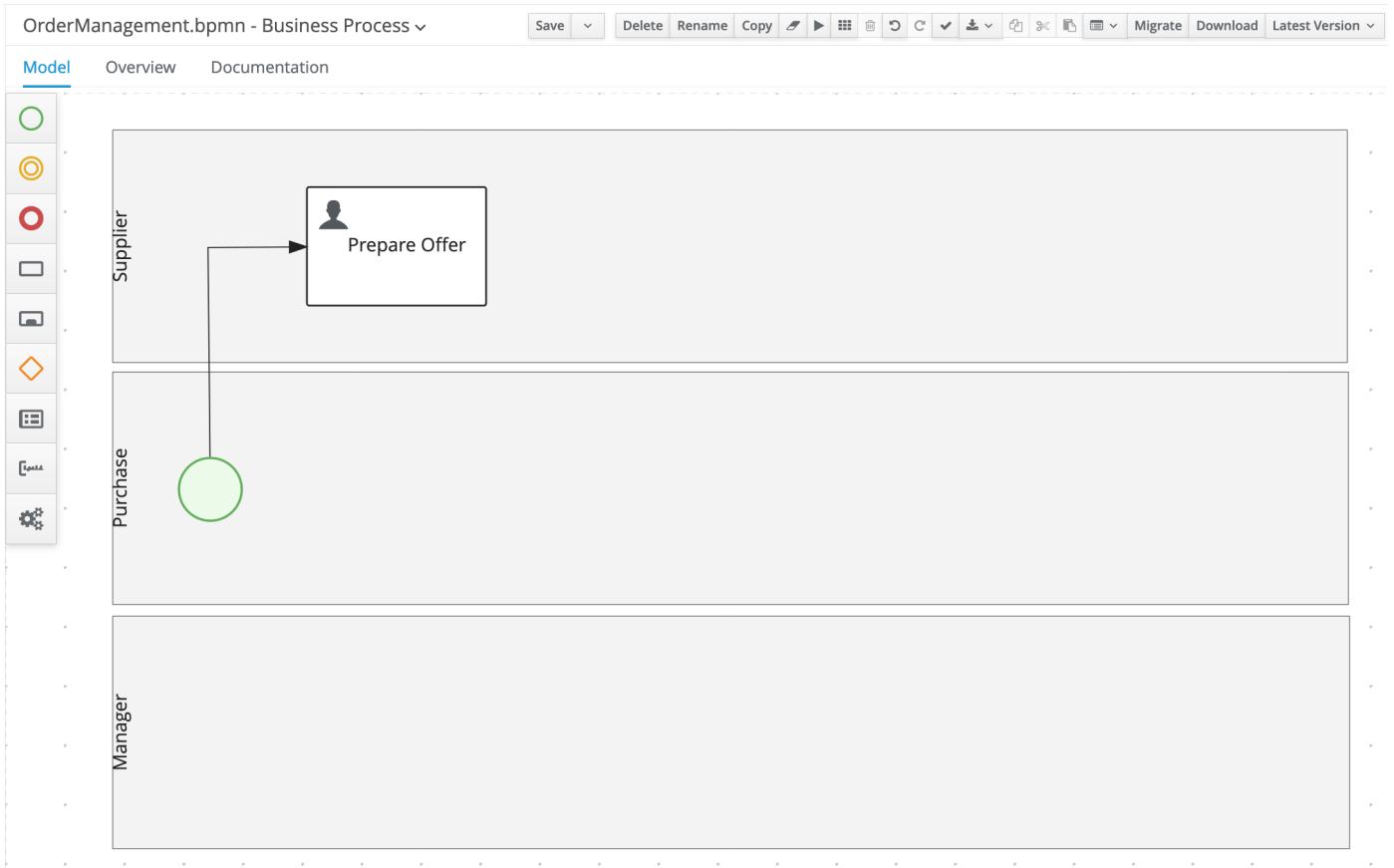
#### Data Inputs and Assignments

Name	Data Type	Source
orderInfo	OrderInfo	orderInfo
supplierInfo	SupplierInfo	supplierInfo

a. Output

#### Data Outputs and Assignments

Name	Data Type	Source
supplierInfo	SupplierInfo	supplierInfo



5. Create the `Auto Approve Order` **Business Rule** node in the `Purchase` swimlane and connect it to the `Prepare Offer` node. Set the following properties:

6. Rule language: DMN

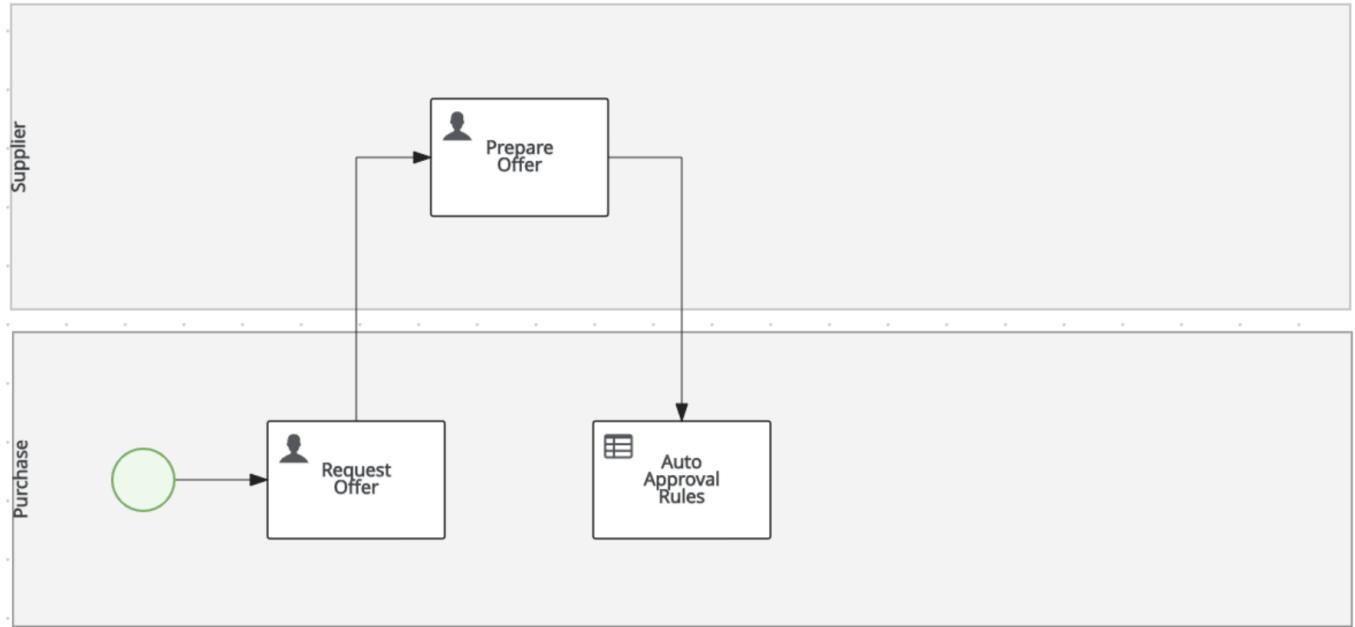
## 7. Assignments:

### Data Inputs and Assignments

Name	Data Type	Source
Order Information	OrderInfo	orderInfo
Supplier Information	SupplierInfo	supplierInfo

### Data Outputs and Assignments

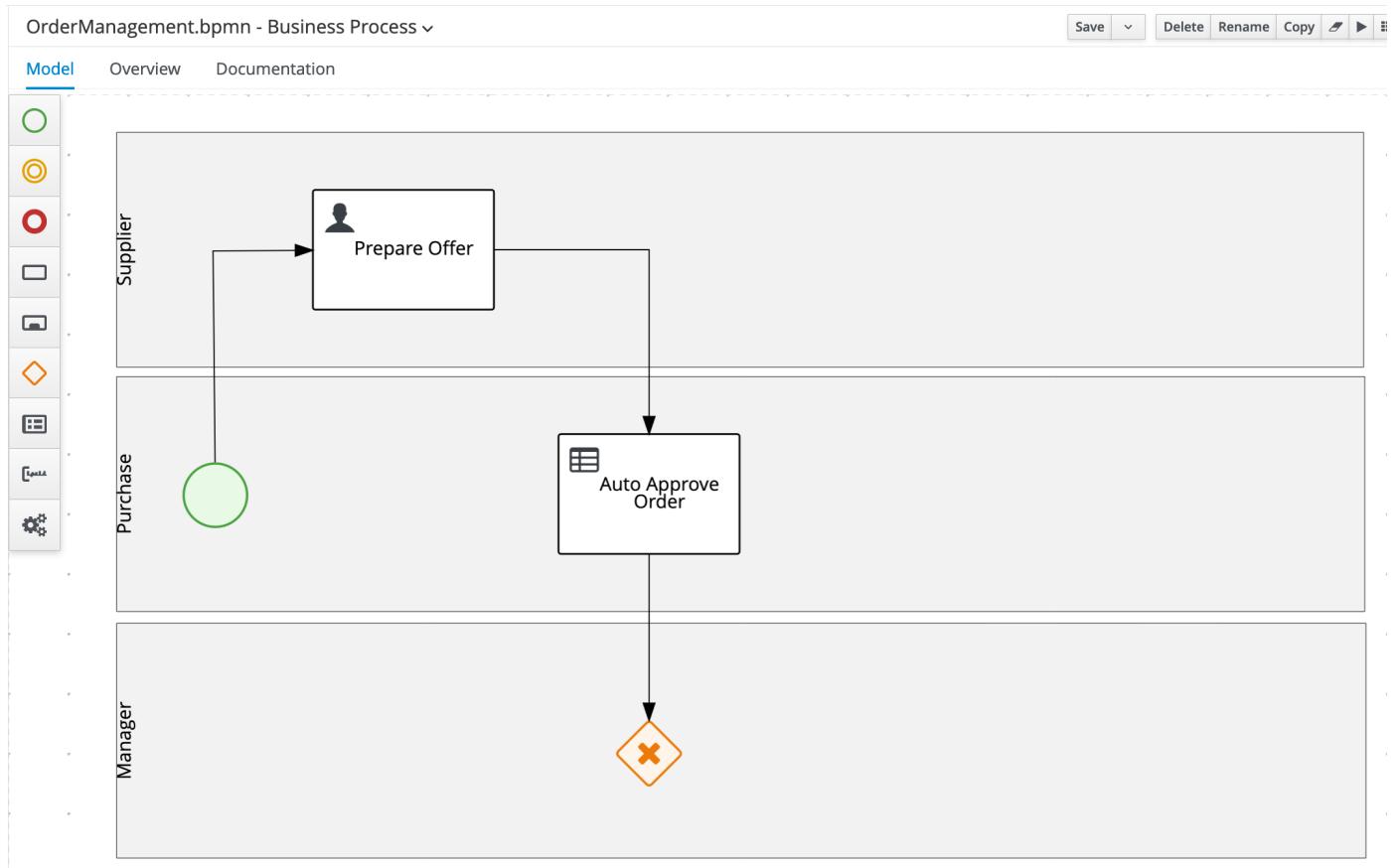
Name	Data Type	Target
Approve	Boolean	approved



INFO: After we've created our DMN Decision Model, we will revisit the configuration of this node to reference this DMN model via its `name` and `namespace` properties.

### 19.3.2 EXCLUSIVE GATEWAY

1. Create an **X-OR Gateway/Exclusive Gateway** in the `Manager` swimlane, below the `Auto Approve Order` node and connect it to that node.



2. Create the **Approve User Task** in the **Manager** swimlane and connect it to the **X-OR gateway**. Set the following properties:

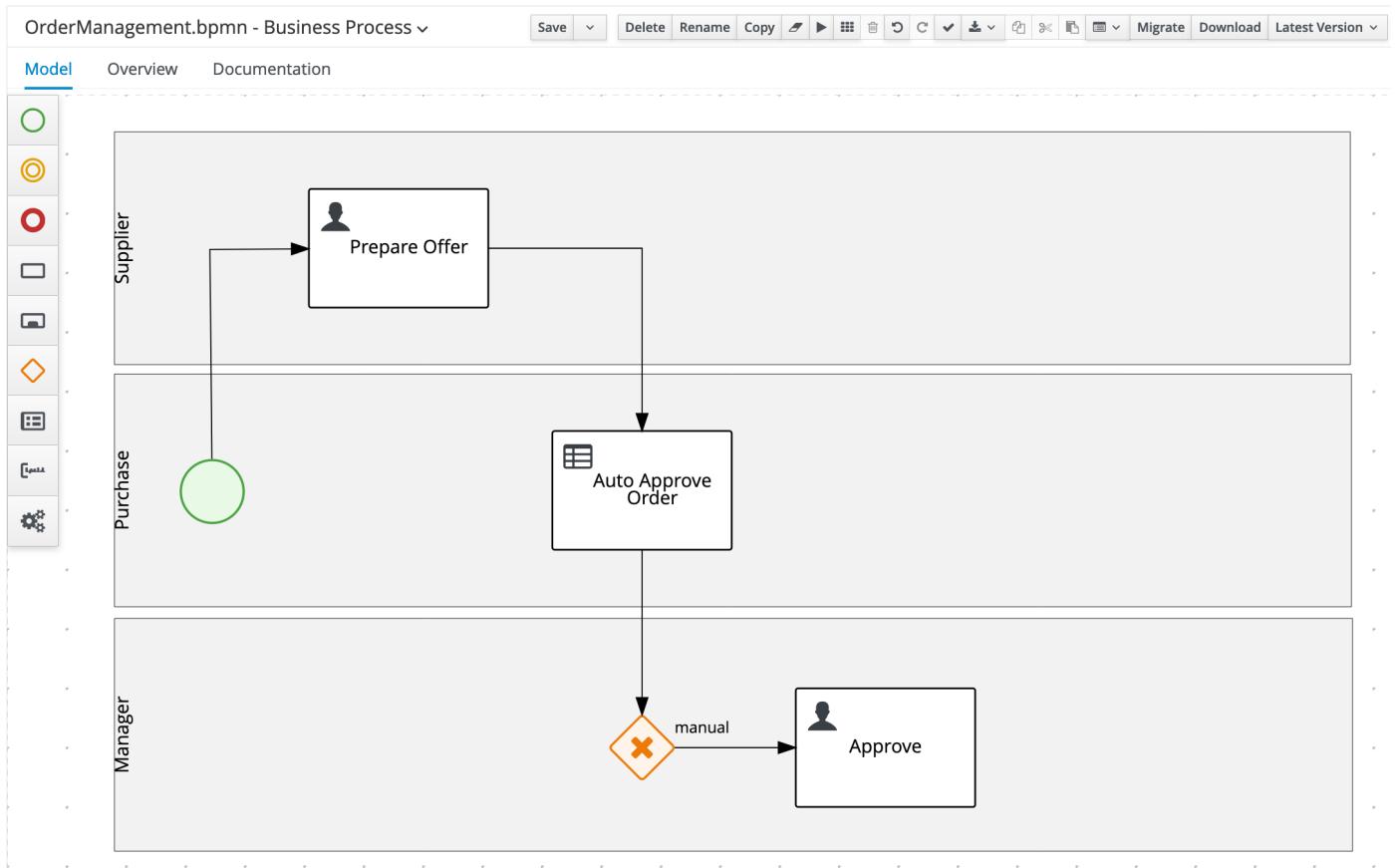
- Task Name: Approve
- Subject: Approve Order of #{orderInfo.item}
- group: rest-all
- Assignments:

#### Data Inputs and Assignments

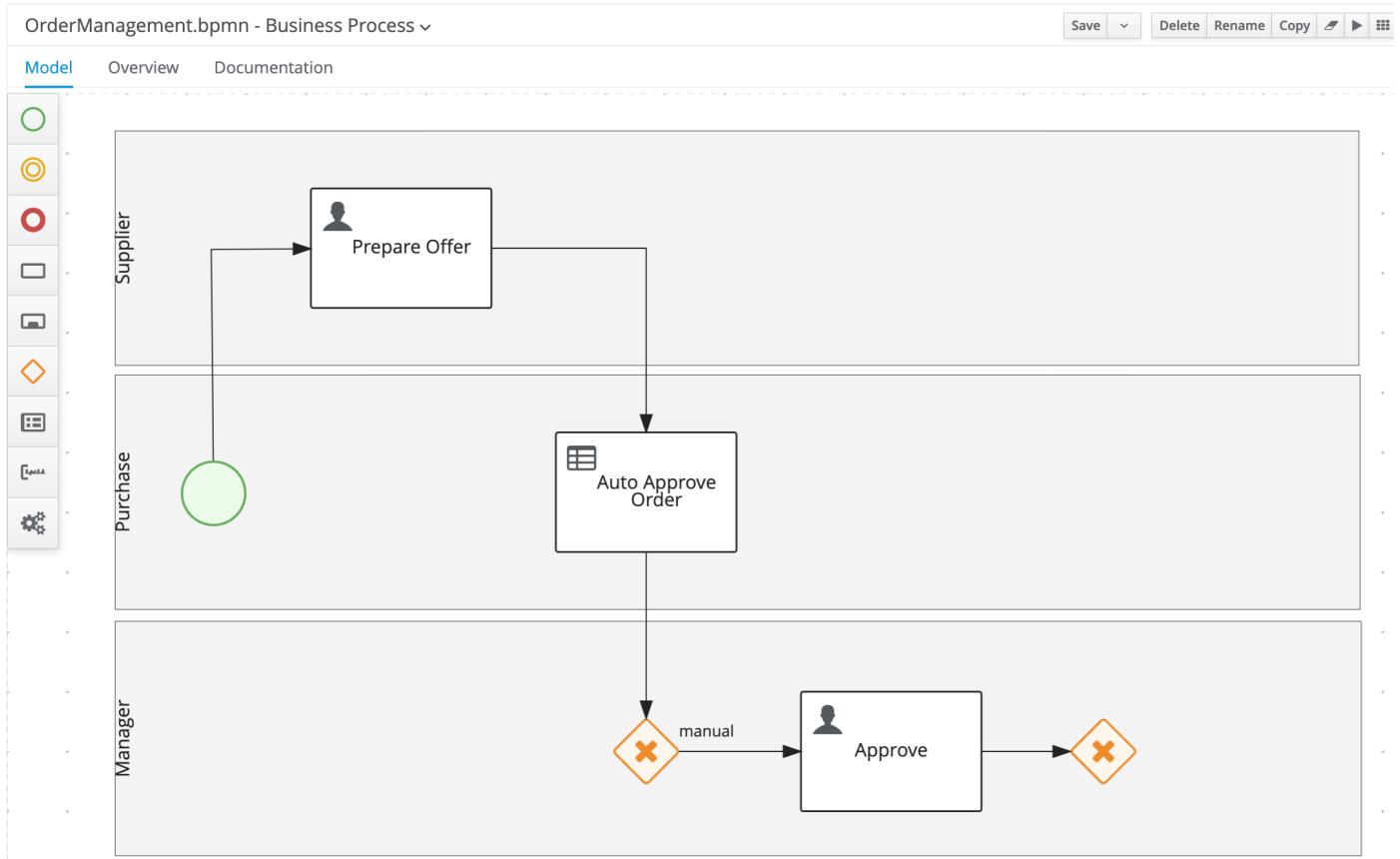
Name	Data Type	Source
orderInfo	OrderInfo	orderInfo
supplierInfo	SupplierInfo	supplierInfo

#### Data Outputs and Assignments

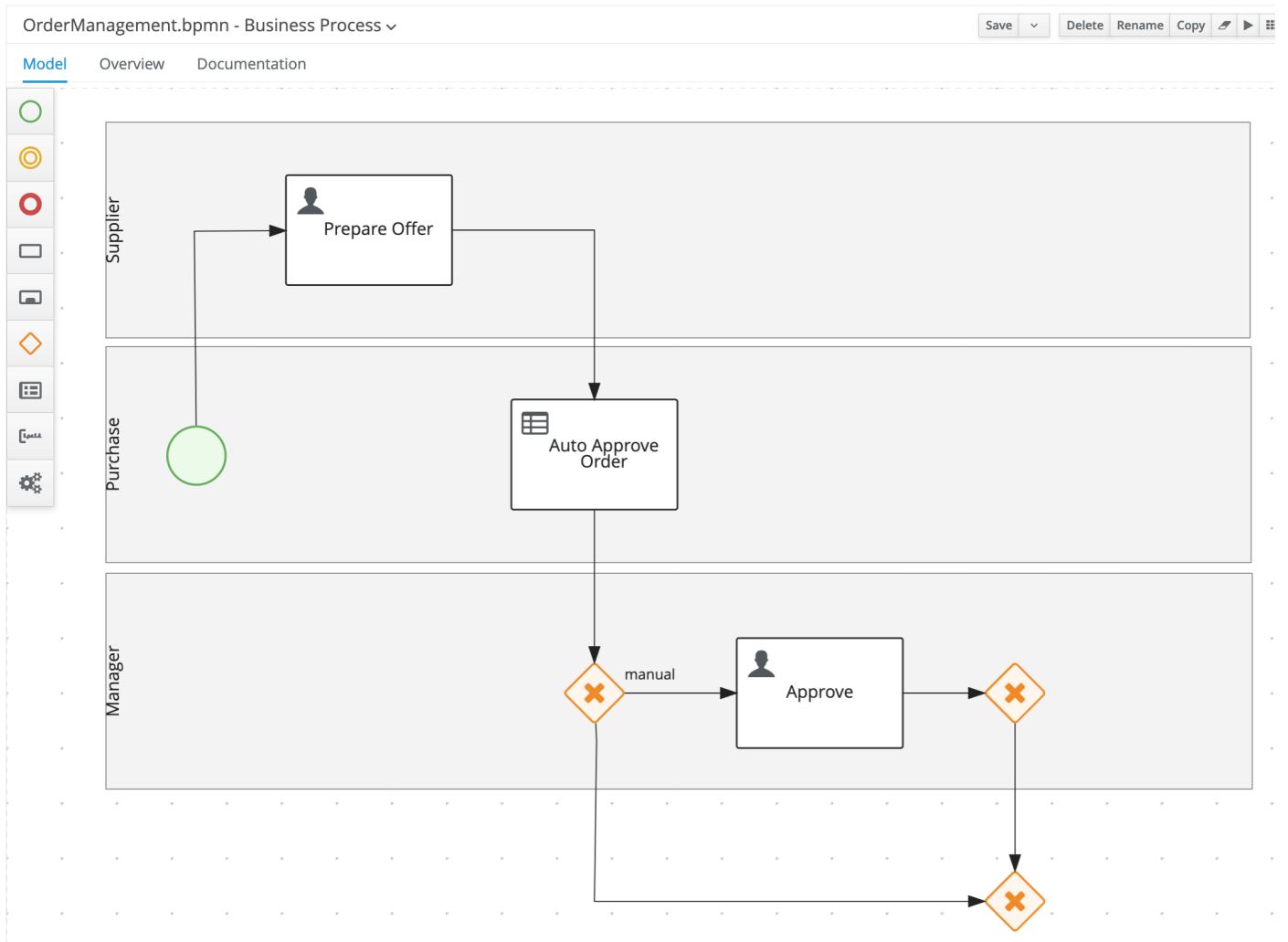
Name	Data Type	Target
orderInfo	OrderInfo	orderInfo



3. Create an X-OR Gateway/Exclusive Gateway in the Manager swimlane, after the Approve node and connect it to that node.

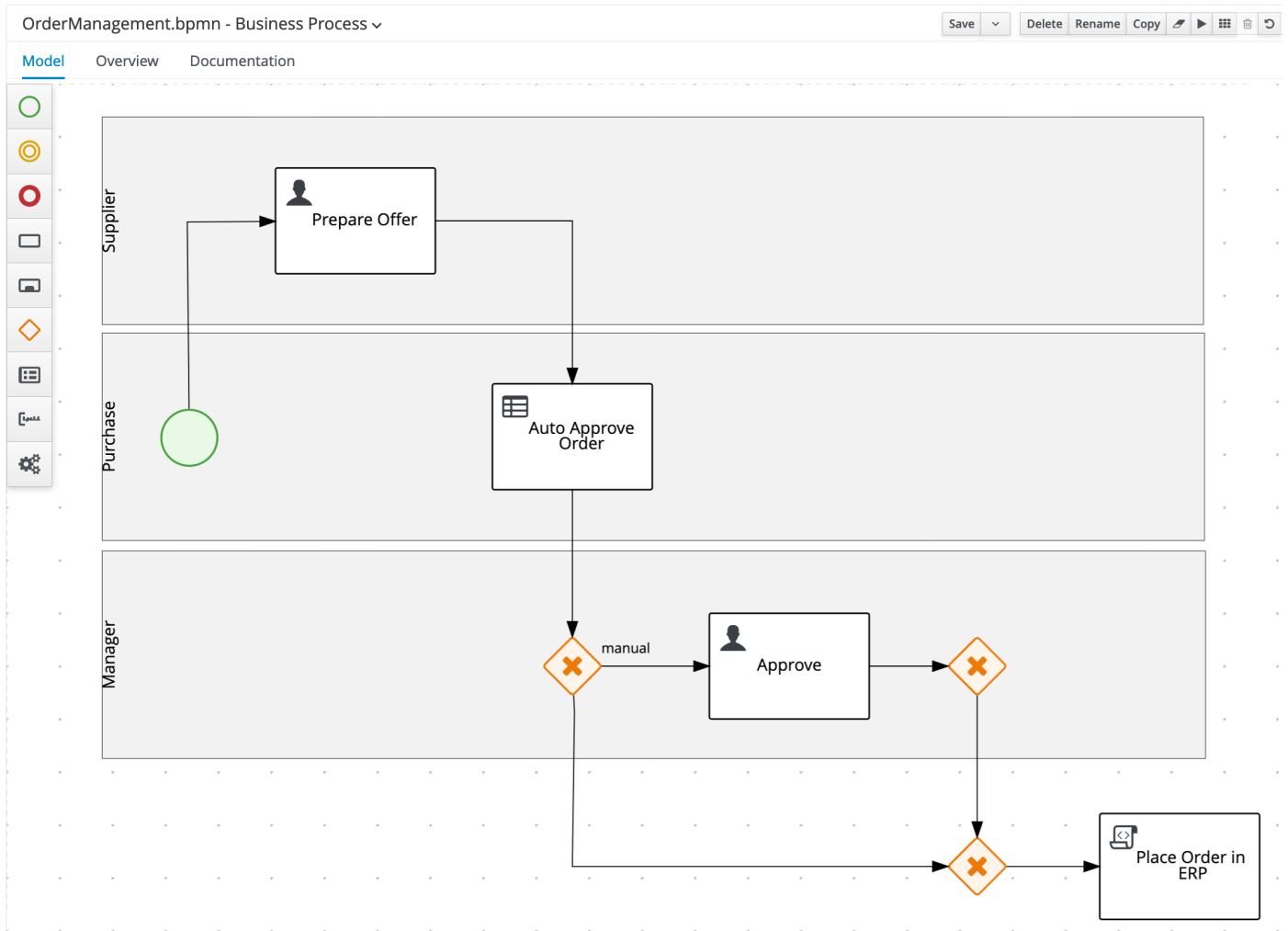


4. Create another **X-OR Gateway/Exclusive Gateway** under the `Manager` swimlane (so outside of the swimlane) and connect it to the two other **X-OR Gateways/Exclusive Gateways** as shown in image below:

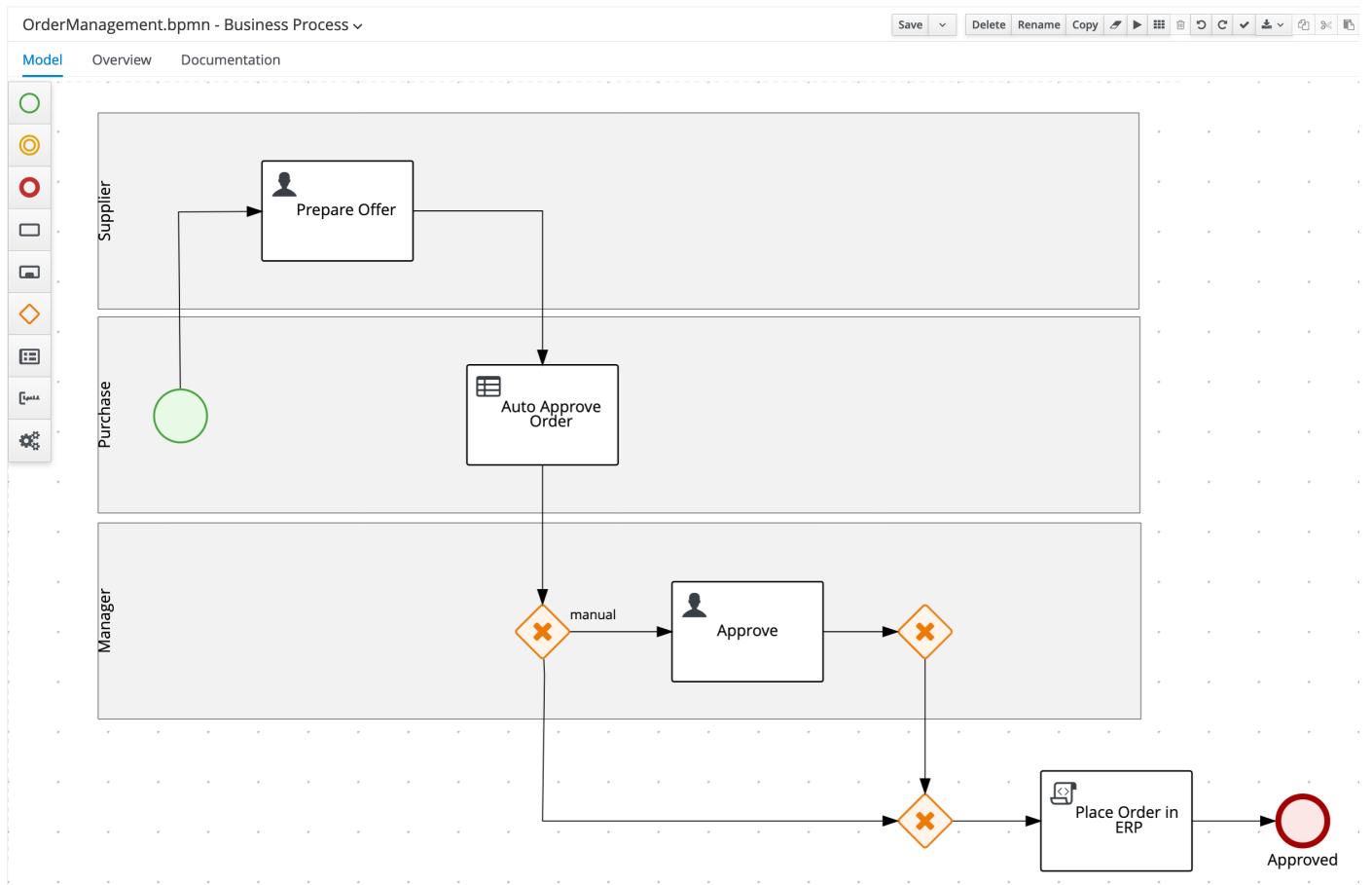


5. Create the `Place Order in ERP` **Script Task** under the `Manager` swimlane (so outside of the swimlanes) and connect it to the **X-OR Gateway** we created earlier. Set the following script in the node's properties properties:

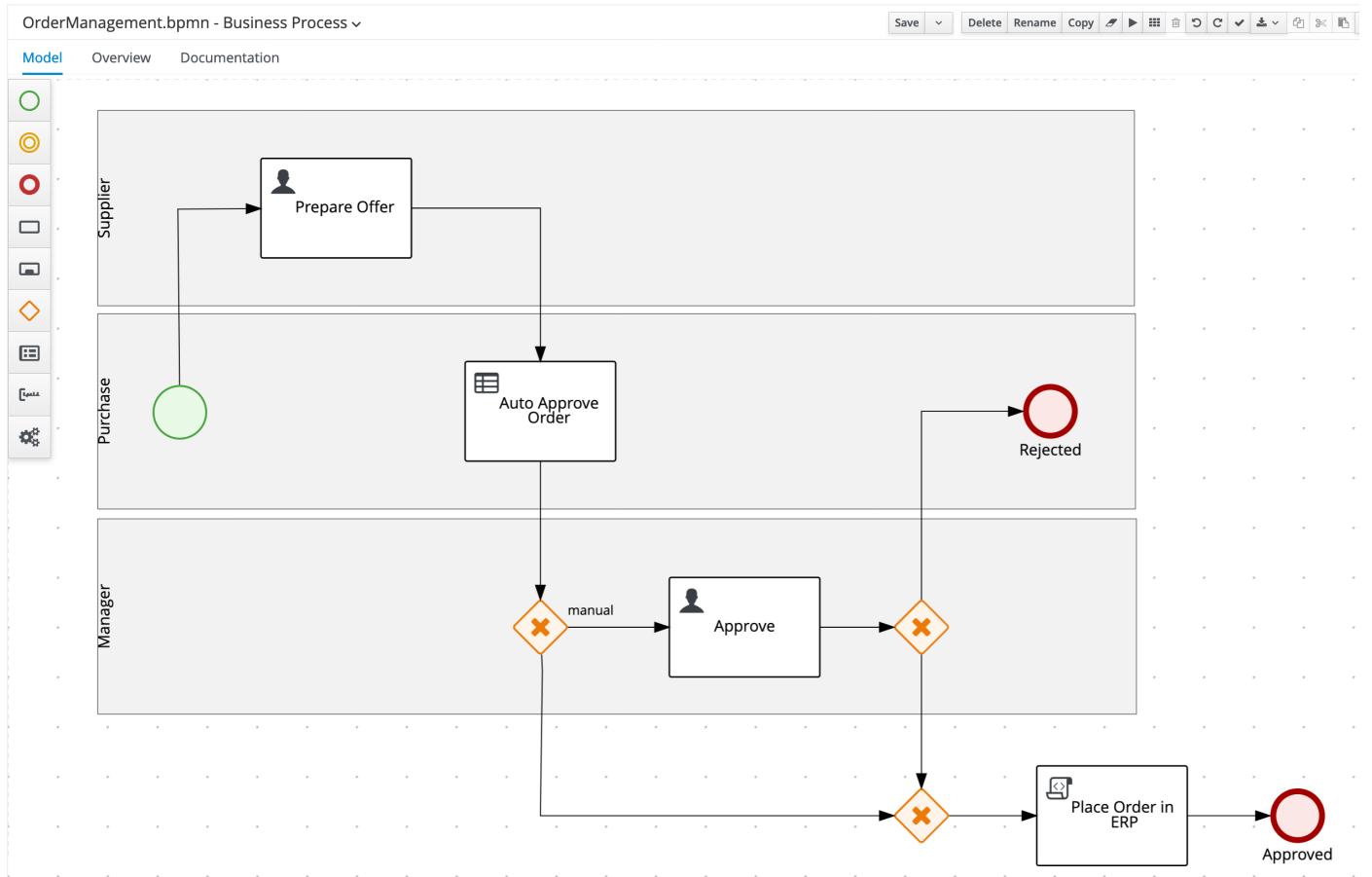
```
System.out.println("Place Order in ERP");
```



6. Create an **End Event** node under the `Manager` swimlane (so outside of the swimlanes) and connect it to the `Place Order in ERP` node. Name it `Approved`.

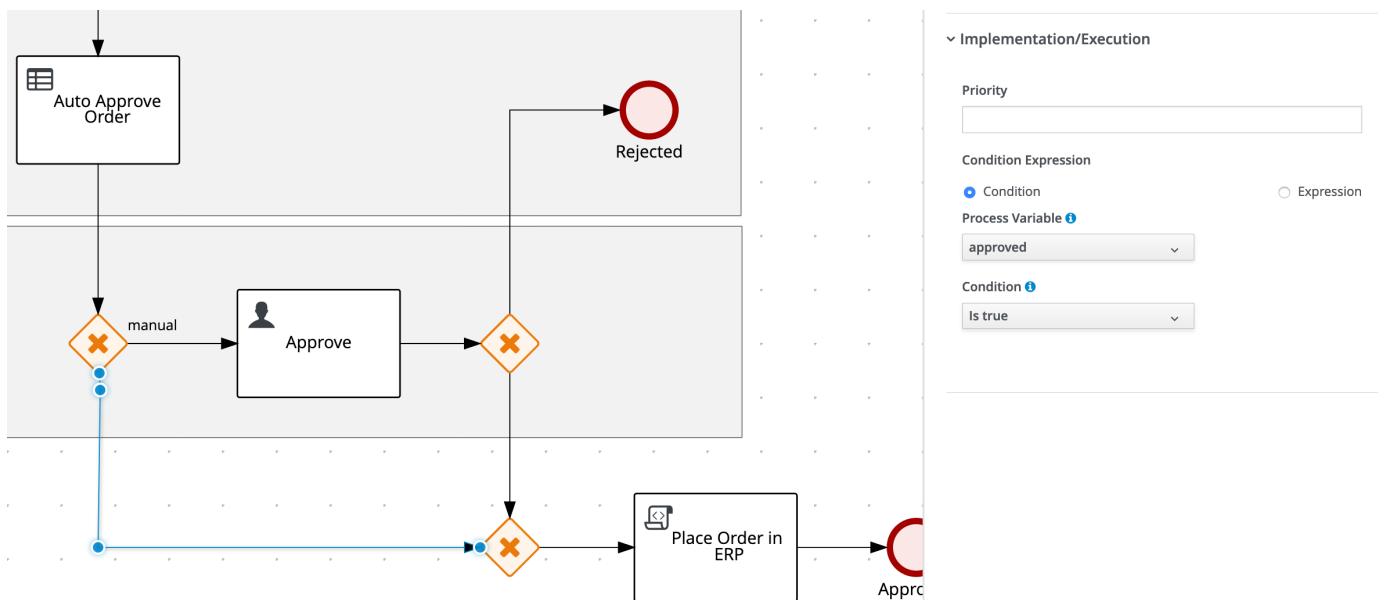


7. Create an **End Event** node in the **Purchase** swimlane and connect it to the **X-OR Gateway**. Name it **Rejected**.

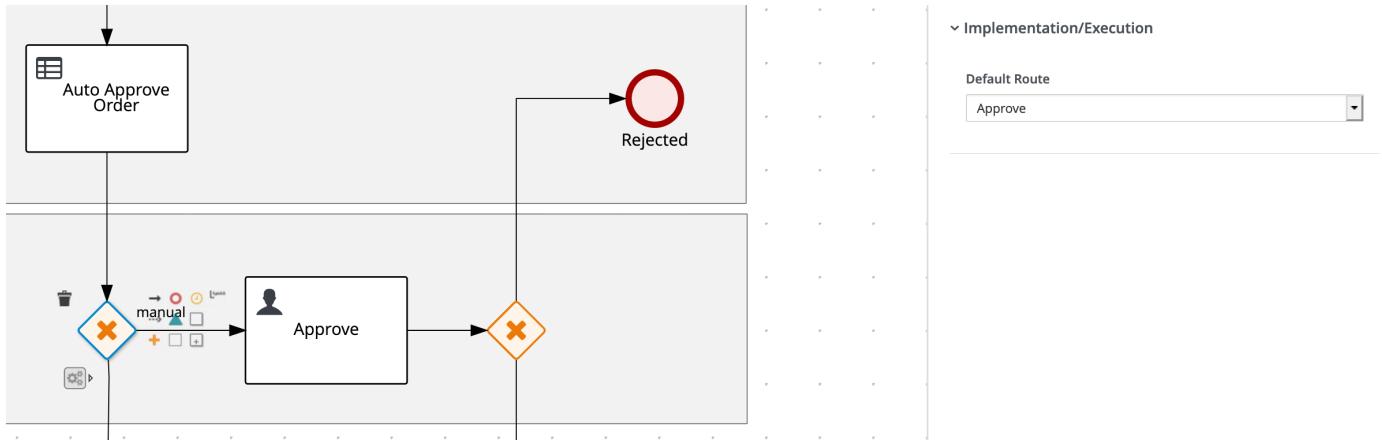


8. On the **Sequence Flow** from the **X/OR Gateway** before the **Approve** node that is connected of the other **X/OR Gateway**, set the following condition, which tells the process engine that this path should be taken when the order is not automatically approved:

- Process Variable: `approved`
- Condition: `Is true`

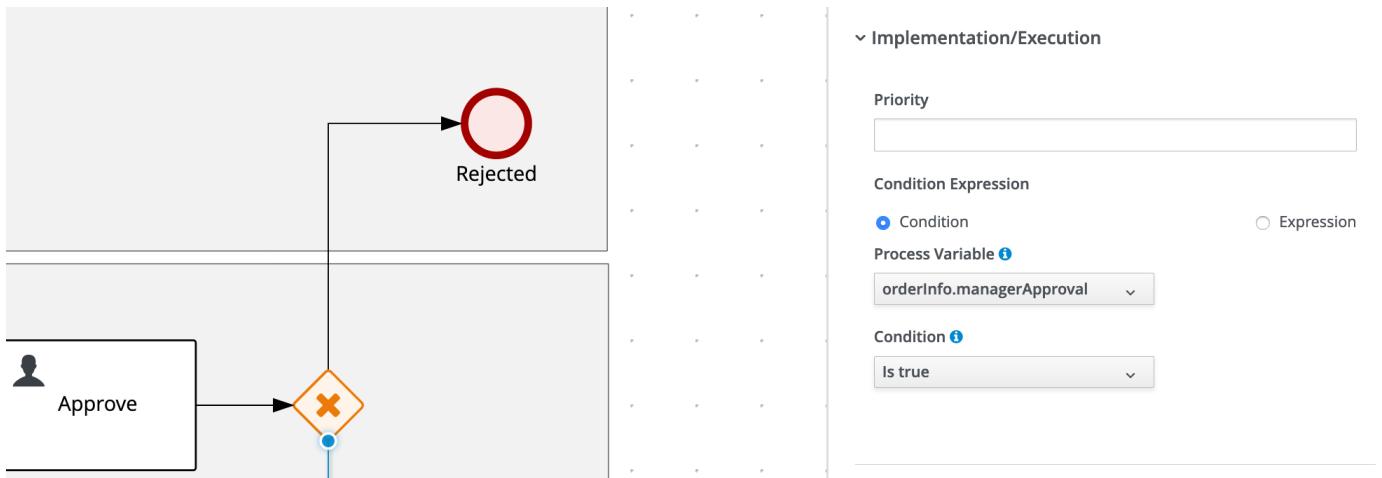


9. On the **Gateway** before the **Approve** node, set the **Default Route** property to `Approve`.

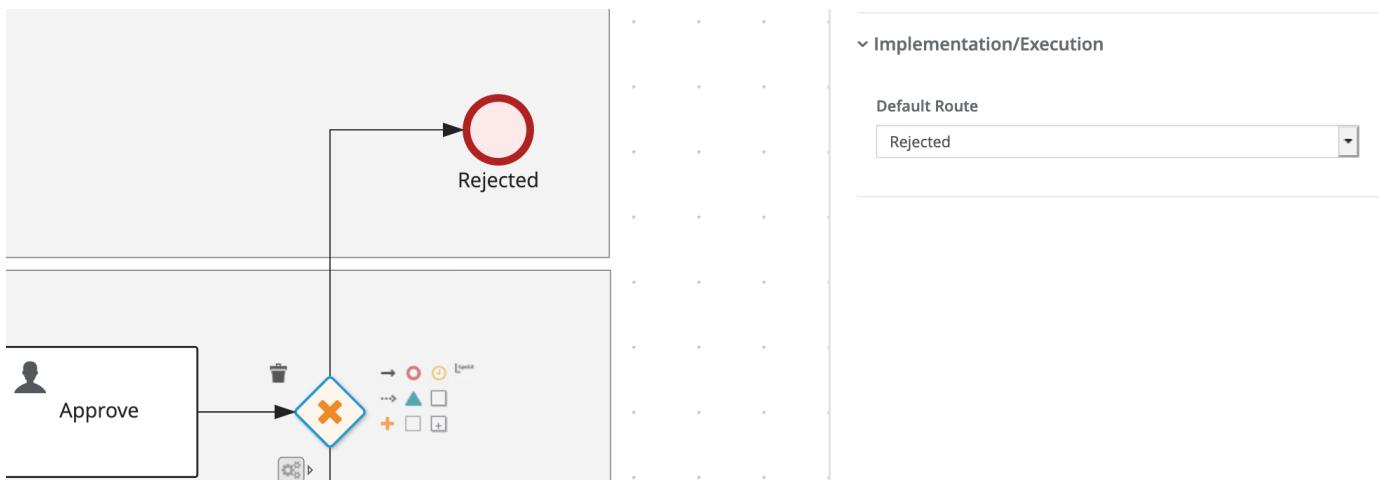


10. On the **Sequence Flow** from the **X/OR Gateway** after the **Approve** task, which is connected to the **X/OR Gateway** before the **Place Order in ERP** task, set the following condition:

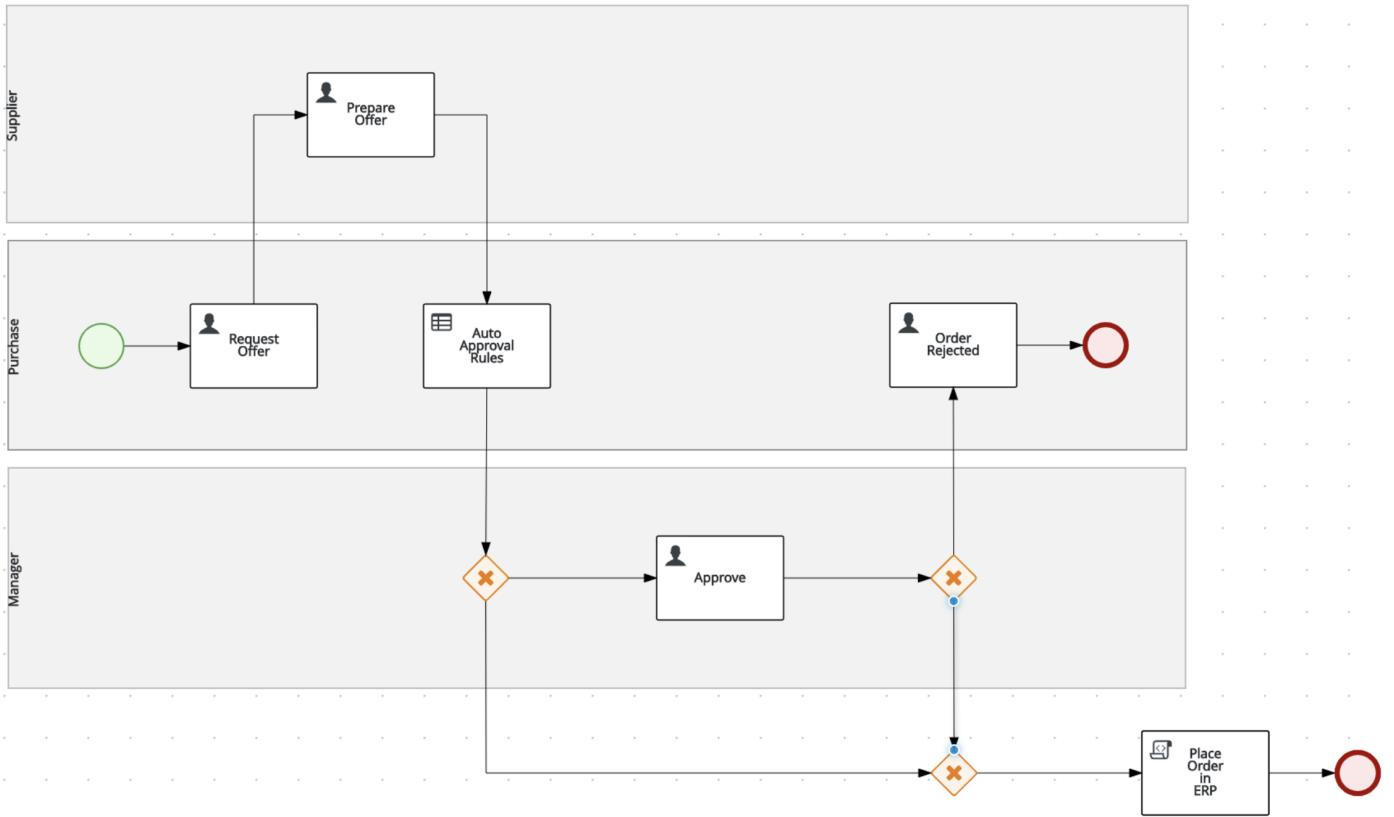
- Process Variable: `orderInfo.managerApproval`
- Condition: `Is true`



11. On the **X/OR Gateway** after the **Approval** node , set the **Default Route** to **Rejected** .



12. Save the process definition.



With the overall layout of the process definition complete, the routing logic implemented, and the I/O assignments defined, we can now implement the business rules of our automated approval decision.

#### 19.4 Business Rules and Decisions

Our **Order Management** process contains a **Business Rule Task**, but we have not yet defined the *Decision Model* that will be used in the task. In this paragraph we will implement the automatic approval rules in the form of a DMN model.

##### 19.4.1 CREATING THE DMN INPUTS AND BKM

1. In the main project page, the so called **library view**, click on the **Add Asset** button.
2. In the next screen, set the drop-down filter to **Decision**. Select the **DMN** asset. Give it the name `order-approval`.

**Create new DMN**

**DMN \***

**Package**

+ Ok
Cancel

3. In the DMN editor, open the property-panel on the right-side of the screen and set the *Namespace* property to: `http://www.redhat.com/dmn/demo/order-management-dmn`. First we need to import our data-model, so we can use it in our DMN decisions. In the DMN editor, click on the *Data Types* tab and click on the *Import Data Object* button at the right-hand side of the screen:

order-approval.dmn - DMN

Model Overview Documentation **Data Types** Included Models

**Custom Data Types**

Data types determine the structure of the data used in DMN boxed expressions. You can use basic data types (example, Boolean) or you can use this dialog to create custom data types.

[View more >](#)

Search...

Expand all / Collapse all  Import Data Object

No custom data types have been defined.

There are currently no custom data types available for you to view or edit. To get started, add a custom data type.

4. Select both the `OrderInfo` and `SupplierInfo` objects and click on the *Import* button:

**Import Data Object**

Project Data Objects

com.om\_lab\_space.order\_management.OrderInfo  
 com.om\_lab\_space.order\_management.SupplierInfo

- Note:** This action is a one time import. If the Data Object is updated you will need to reimport it to have the changes reflected in the respective data type.

With the 2 datatypes imported, we need to create a third type that will hold the possible values for the `urgency` field of our `Order Information`. Click on the blue *Add* button in the top-right corner.

1. In the entry that opens, give the data type the *Name* `Urgency` and the *Type* `string`:

The screenshot shows the 'order-approval.dmn - DMN' interface with the 'Data Types' tab selected. Under 'Custom Data Types', there are two entries: 'OrderInfo (Structure)' and 'SupplierInfo (Structure)'. For 'OrderInfo', a new data type 'Urgency' is being defined. The 'Name' field contains 'Urgency' and the 'Type' dropdown is set to 'string'. Below this, an 'Add Constraints' button is visible. The 'constraint type' dropdown is set to 'Enumeration', and the values 'low' and 'high' are listed.

2. Click on the *Add Constraints* button, select `Enumeration` as the *constraint type*, and set the values low and high`.

The screenshot shows the 'Data Type constraints' dialog box. It displays two enumeration values: 'low' and 'high'. At the bottom right, there is a blue 'Add' button with a plus sign. At the bottom left, there are 'OK' and 'Cancel' buttons.

3. Click on the blue checkmark button to save the type.

Custom Data Types

Data types determine the structure of the data used in DMN boxed expressions. You can use basic data types (example, Boolean) or you can use this dialog to create custom data types.

[View more >](#)

Search... Search icon

[Expand all](#) / [Collapse all](#) [+ Add](#) [Import Data Object](#)

\* Name      \* Type

Urgency	string
---------	--------

List  String

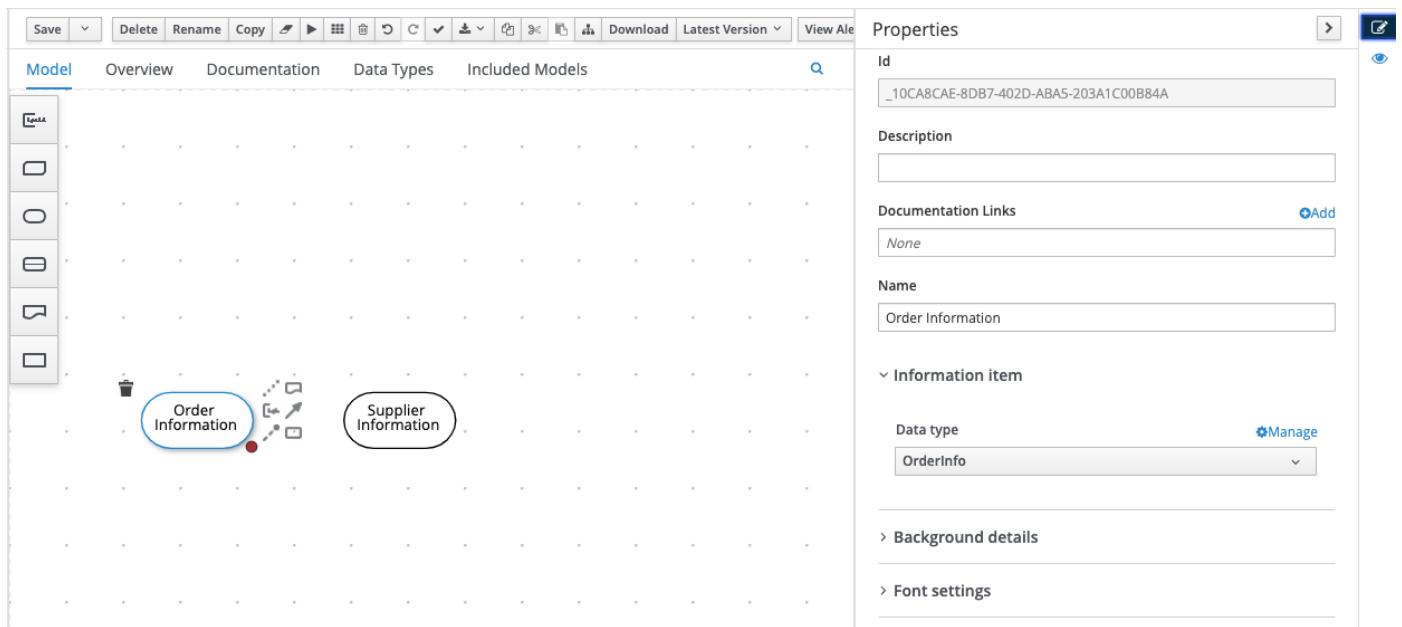
low, high

✓ ✗ ✖

Ctrl + S

4. Navigate back to the model via the *Model* tab. Add 2 `Input` nodes to the model and name them `Order Information` and `Supplier Information`.

5. Select the `Order Information` node. Open the properties panel on the right-hand side of the screen, and set the *Data type* to `OrderInfo`.



6. Do the same for the `Supplier Information` node. Set the *Data type* to `SupplierInfo`. Create a new Business Knowledge Model node, name it `Price Tolerance`.
7. Click on the node, and click on the *Edit* button to start editing the node:

The screenshot shows the DMN Modeler interface with the 'order-approval.dmn - DMN' model selected. The 'Model' tab is active. A node labeled 'Price Tolerance' is selected and highlighted with a blue border. A speech bubble icon with the word 'Edit' and a checkmark icon is positioned above the node. The 'Properties' panel on the right shows the following details for the 'Price Tolerance' node:

- Name:** Price Tolerance
- Data type:** OrderInfo

Below the node, there is a table with two rows and two columns. The first row contains a column labeled 'F'. The second row is empty.

8. Click in the *Edit parameters*. An editor will open. Click on *Add parameter*. Name the parameter `order information` and set the type to `OrderInfo`.

The screenshot shows the 'Edit Parameters' dialog for the 'Price Tolerance' function. The dialog has a header 'Edit Parameters' and a button 'Add parameter'. Below it, there is a list of parameters: 'order information' with type 'OrderInfo'. A red 'Delete' button is positioned next to the type. The main area shows the function definition: 'F' followed by a decision table with one row and two columns. The first column is labeled '(order information)'.

9. Right click in the empty white cell under the parameter definitions and select *Clear*. The text *Select expression* will appear in the cell. Click on the cell and select *Decision Table*.

The screenshot shows a context menu titled 'Select logic type' open over the empty white cell under the parameter definitions. The menu includes options: 'Literal expression', 'Context', 'Decision Table' (which is highlighted in blue), 'Relation', 'Function', and 'Invocation'. The main area shows the function definition: 'F' followed by a decision table with one row and two columns. The first column is labeled '(order information)'.

10. Add an *input clause* to the decision table. The name of the *input clause* is `order information.urgency`, which references the `urgency` attribute of the `order information` parameter. Set the type to `Urgency`, which references the `Urgency` enumeration we created earlier.

order-approval.dmn - DMN

Model Overview Documentation Data Types Included Models

« Back to order-approval

### Price Tolerance (Function)

F		Price Tolerance (<Undefined>)
U	informati (Urgenc	<b>Edit Input Clause</b> Name: order information.urgency Data Type: Urgency
1		Description

11. Set the *output clause* data type to `number`. Leave the name empty.

order-approval.dmn - DMN

Model Overview Documentation Data Types Included Models

« Back to order-approval

### Price Tolerance (Function)

F	Price Tolerance (<Undefined>)	
	(order information)	
U	information.urg (Urgency)	output (Urgency)
1		

Edit Output Clause

Name: Name

Data Type: number

Manage

12. Click on the `Price Tolerance` cell (top cell of the table), and set the data type to `number`.

order-approval.dmn - DMN

Model Overview Documentation Data Types Included Models

« Back to order-approval

Price Tolerance (*Function*)

F	Price Tolerance (number)		
	(order information)		
U	order information.urgency (Urgency)	(number)	Description
1			

**Edit Expression**

Name: Price Tolerance

Data Type: number

13. Implement the rest of the decision table as shown below. And save the DMN model.

order-approval.dmn - DMN

Model Overview Documentation Data Types Included Models

« Back to order-approval

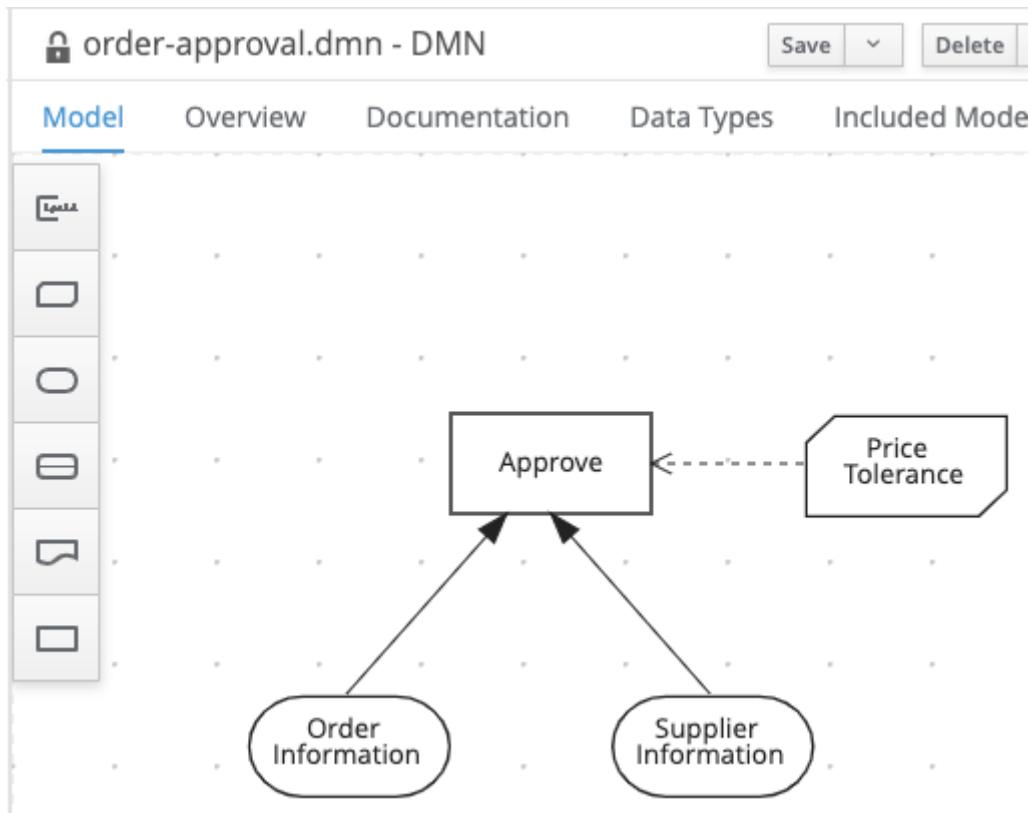
Price Tolerance (*Function*)

F	Price Tolerance (number)		
	(order information)		
U	order information.urgency (Urgency)	(number)	Description
1	"low"	1.15	
2	"high"	1.25	

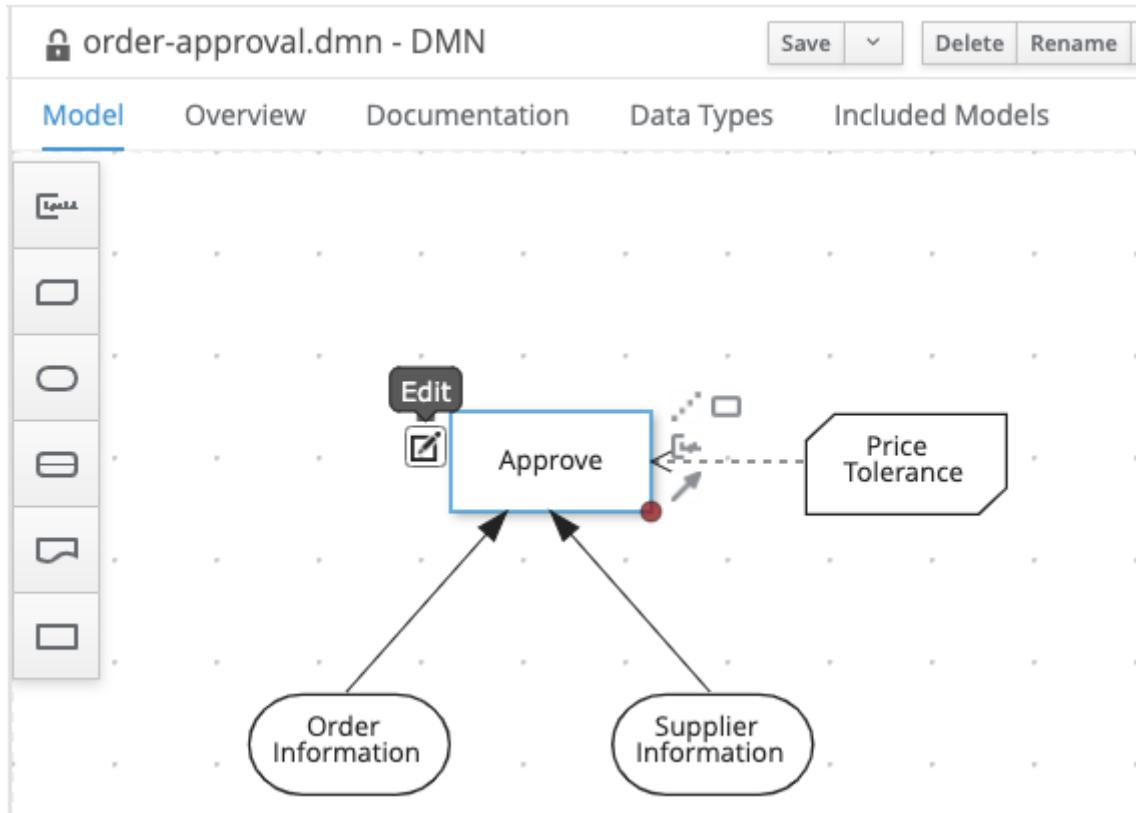
#### 19.4.2 WRITING THE DMN DECISION

In this section we will complete the writing of the DMN decision.

1. Navigate back to the model by clicking on the *Back to order-approval* link at the top-left of the editor. Create a new *Decision Node* and name it *Approve*. Connect the 2 input nodes and out *Price Tolerance* business knowledge model node to the new decision node.



2. Select the *Approve* decision node and click on the edit button.



3. Click on *Select Expression*, and set the logic type to *Literal Expression*.

order-approval.dmn - DMN

Model Overview Documentation Data Types Included Models

« Back to order-approval

Approve (<Undefined>)

Select expression

Select logic type

- Literal expression
- Context
- Decision Table
- Relation
- Function
- Invocation

4. Enter the following expression: `Supplier Information.offer < Price Tolerance(Order Information) * Order Information.targetPrice`

order-approval.dmn - DMN

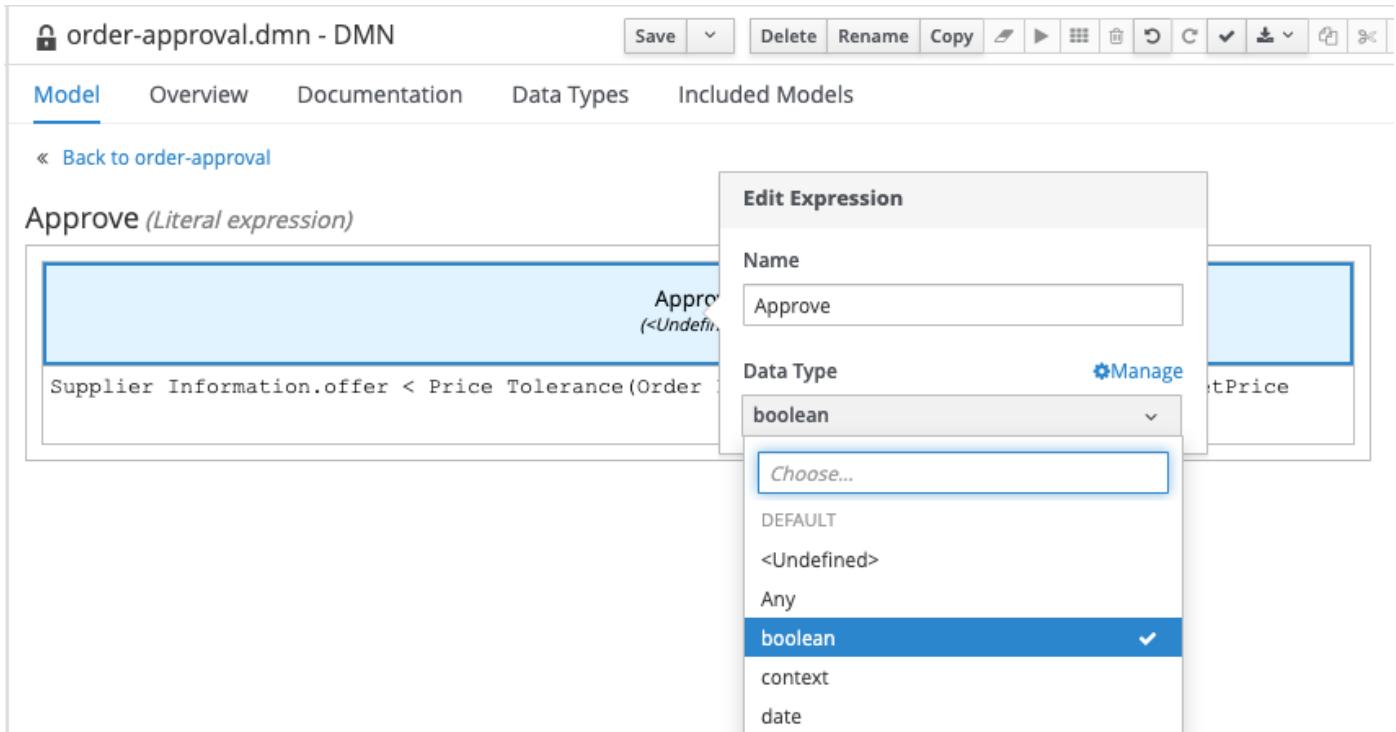
Model Overview Documentation Data Types Included Models

« Back to order-approval

Approve (Literal expression)

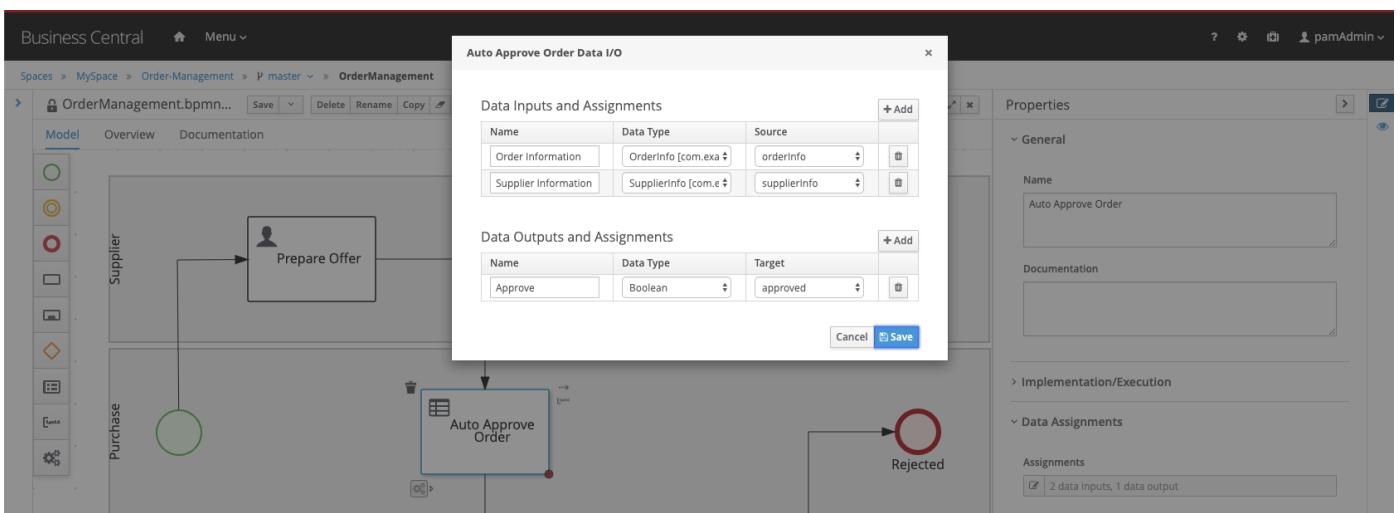
Approve (<Undefined>)
<code>Supplier Information.offer &lt; Price Tolerance(Order Information) * Order Information.targetPrice</code>

5. Click on the Approve cell (top cell of the table), and set the data type to boolean.



## 19.5 Connecting the Decision to the Process

1. Navigate back to the model by clicking on the *Back to order-approval* link at the top-left of the editor.
2. Our DMN model is now complete. Make sure to save your model.
3. With our DMN model implemented, we can now revisit our **Business Rules Task** in our BPMN2 model. Open the `order-management` process definition and click on the `Auto Approval Order` node.
4. Open the node's properties in the property-panel on the right side of the editor, open the **Implementation/Execution** section and set:
  - Namespace: `http://www.redhat.com/dmn/lab/order-approval-dmn`
  - Name: `order-approval`
5. In the same properties panel, expand the *Data Assignments* section and open the *Assignments* editor
6. Implement the following data input and output assignments.



7. Our BPMN model is now complete. Make sure to save the model.

Now, we should be able to create and implement our forms.

## 19.6 Creating Forms

In this section we are going to create the process start and user-task forms. We could simply generate these forms with the click of a button, which gives us some standard forms based on the process and task data.

In this lab however, we will be creating these forms using the **Form Modeler** tool. This allows us to design these forms to our specific needs.

### 19.6.1 PROCESS START FORM

Let's start with the process start form. We want to create the following form:

Business Central Menu ▾

Spaces > MySpace > Order-Management > master > OrderManagement-taskform

Form Modeler [OrderManagement-taskform] ▾

Save | Delete | Rename | Copy | Download | Latest Version | View Alerts | X | Edit

Model Overview

Order Information

item name  
-- Select a value --

urgency  
 low  high

TargetPrice  
TargetPrice

SupplierInfo

supplier name  
username

- In the project's library view, click on **Add Asset**. Filter on **Form**, click on the **Form** tile. Enter the details as shown in the screenshot below:

**Create new Form**

**Form \***

**Package**

**Select Model type:**

Business Process

**Select Process:**

**Select Form:**

Data Object

**+ Ok** **Cancel**

- On this form we want to specify the initial order. We therefore require fields from the `orderInfo` and `supplierInfo` process variable. When we expand the `Model Fields` section, we can see our 2 process variables (`orderInfo` and `supplierInfo`). These are both complex objects. To work with complex objects (as opposed to simple types like integers and booleans), we require a data-form for that specific object. We therefore first need to create a data-form for our `orderInfo` and `SupplierInfo` objects.

- Go back to the project's library view, click again on **Add Asset** and create a new form. Use the following details:

## Create new Form

**Form \***

**Package**

**Select Model type:**

- Business Process
- Data Object

**Select Data Object from project:**

+ Ok
Cancel

4. Using the Form Modeler constructs, create the following form:

The screenshot shows the 'Form Modeler [OrderManagement-Order]' interface. On the left, there's a sidebar with 'Components' expanded, showing 'Model Fields' and 'Form Controls'. The main area is titled 'Model' and contains three fields:
 

- 'item name': A dropdown menu with the placeholder 'Select a value'.
- 'urgency': A radio button group with two options: 'low' and 'high'.
- 'TargetPrice': A text input field with the placeholder 'TargetPrice'.

 At the top right, there are standard save, delete, rename, copy, download, and alert buttons.

5. To create this form, drag both the `item`, `urgency` and `targetPrice` onto the canvas and configure them as follows.

6. List Box:

## Field properties

### Field Type

ListBox

### Label

item name

### Options

<a href="#" style="color: blue; font-weight: bold;">+ New Instance</a>			
Value	Text		
Huawei P10	Huawei P10	<a href="#" style="color: blue;">Edit</a>	<a href="#" style="color: red;">Delete</a>
Dell XPS 15	Dell XPS 15	<a href="#" style="color: blue;">Edit</a>	<a href="#" style="color: red;">Delete</a>

« < 1-2 of 2 > »

### Default Value

-- Select a value --

Add default empty option

Required

Read Only

Validate on change field value

### Help Message i

### Field binding

item

[+ Ok](#) [Cancel](#)

7. Radio Group:

## Field properties

### Field Type

RadioGroup

### Label

urgency

### Options

[+ New Instance](#)

Value	Text		
low	low	<a href="#">Edit</a>	<a href="#">Delete</a>
high	high	<a href="#">Edit</a>	<a href="#">Delete</a>

« < 1-2 of 2 > »

### Default Value

-- Select a value --

Show options inline

Required

Read Only

Validate on change field value

### Help Message

### Field binding

urgency

[+ Ok](#)

[Cancel](#)

8. Decimal Box:

**Field properties****Field Type**

DecimalBox

**Label**

TargetPrice

**PlaceHolder**

TargetPrice

**Max. Length**

100

 Required Read Only Validate on change field value**Help Message** **Field binding**

targetPrice

 Ok  Cancel9. Save the form and create another new form for our `supplierInfo`. Use the following details.

**Create new Form**

**Form \***

**Package**

**Select Model type:**

- Business Process
- Data Object

**Select Data Object from project:**

**+ Ok** **Cancel**

10. Using the Form Modeler constructs, create the following form:

The screenshot shows the 'Form Modeler [OrderManagement-SupplierInfo]' interface. On the left, there's a sidebar with 'Components' expanded, showing 'Model Fields' and 'Form Controls'. The main area is titled 'Model Overview' and contains a single text input field with the label 'supplier name' and a placeholder 'username'. The top navigation bar includes 'Business Central', 'Menu', and user information like 'pamAdmin'. Action buttons at the top right include 'Save', 'Delete', 'Rename', 'Copy', 'Download', 'Latest Version', and 'View Alerts'.

11. To create this form, drag the `user` field onto the canvas and configure it as follows.

### Field properties

**Field Type**  
TextBox

**Label**  
supplier name

**PlaceHolder**  
username

**Max. Length**  
100

Required

Read Only

Validate on change field value

**Help Message** 

**Field binding**  
user

**+ Ok** **Cancel**

12. Save the form and open the `OrderManagement` form (the first form we created).
13. Drag the `orderInfo` process variable onto the canvas. In the pop-up form, set the `OrderManagement-Order` form we just created as the **Nested Form**:

**Field properties**

**Field Type**  
SubForm

**Label**  
Order Information

**Nested Form**  
OrderManagement-Order

Required

Read Only

Validate on change field value

**Help Message** ⓘ

**Field binding**  
orderInfo

**+ Ok** **Cancel**

14. Drag the `supplierInfo` process variable onto the canvas. In the pop-up form, set the `OrderManagement-SupplierInfo` form we just created as the **Nested Form**:

**Field properties****Field Type**

SubForm

**Label**

SupplierInfo

**Nested Form**

OrderManagement-SupplierInfo

 Required Read Only Validate on change field value**Help Message** **Field binding**

supplierInfo

**+ Ok** Cancel

## 19.6.2 PREPARE OFFER FORM

Next, we will create the form for the `Prepare Offer` **User Task**.

1. Create a new form. Provide the following details:

**Create new Form**

**Form\***  
PreparedOffer-taskform

**Package**  
com.example.order\_management

**Select Model type:**  
 Business Process

**Select Process:**  
OrderManagement

**Select Form:**  
Task Form for 'PrepareOffer' (PreparedOffer-taskform)

+ Ok Cancel

2. Our aim is to create a form that looks as such:

Business Central Menu

Spaces > MySpace > Order-Management > master > PreparedOffer-taskform

Form Modeler [PreparedOffer-taskform]

**Model** Overview

**OrderInfo**

item name urgency  
low high

**Your Best Offer**

delivery date best offer  
best offer

Save Delete Rename Copy Download Latest Version Hide Alerts

3. As with the process start form, this user-task form operates on 2 variables, `orderInfo` and `supplierInfo`. And, as with the process start form, we need to create a data-object form for each of these variables. Technically, data-object forms for a certain data-object can be reused in multiple task-forms. However, creating a data-object form per task-form allows us to design these data-object forms aimed for that specific task.

**PrepareOffer-OrderInfo**

The screenshot shows the 'Form Modeler [PrepareOffer-OrderInfo]' interface. On the left, there's a sidebar with 'Components' expanded, showing 'Model Fields' and 'Form Controls'. The main area is titled 'Model Overview' and contains two fields: 'item name' (with a text input field containing 'item name') and 'urgency' (with a radio button group showing 'low' and 'high'). A toolbar at the top right includes 'Save', 'Delete', 'Rename', 'Copy', 'Download', 'Latest Version', 'Hide Alerts', and a trash icon.

**PrepareOffer-SupplierInfo**

The screenshot shows the 'Form Modeler [PrepareOffer-SupplierInfo]' interface. On the left, there's a sidebar with 'Components' expanded, showing 'Model Fields' and 'Form Controls'. The main area is titled 'Model Overview' and contains two fields: 'delivery date\*' (with a text input field containing 'delivery date') and 'best offer\*' (with a text input field containing 'best offer'). A toolbar at the top right includes 'Save', 'Delete', 'Rename', 'Copy', 'Download', 'Latest Version', 'Hide Alerts', and a trash icon.

Finally, we need to create the task form for the `Approve` task.

1. Create a new form. Provide the following details.

**Create new Form**

**Form \***  
Approve-taskform

**Package**  
com.example.order\_management

**Select Model type:**

Business Process

Select Process:  
OrderManagement

Select Form:  
Task Form for 'Approve' (Approve-taskform)

Data Object

**+ Ok** **Cancel**

2. Our aim is create a form that looks like this:

Business Central

Spaces > MySpace > Order-Management > master > Approve-taskform

Form Modeler [Approve-taskform] ▾

Model Overview

SupplierInfo

user	delivery date	best offer
user	delivery date	best offer

OrderInfo

item name	urgency	TargetPrice
item name	urgency	TargetPrice
<input type="checkbox"/> approved		

3. As with the other forms, this user-task form operates on 2 variables, `orderInfo`, `supplierInfo`. And, as with the other forms, we need to create a data-object form for each of these variables.

## Approve-SupplierInfo

The screenshot shows the Form Modeler interface for the 'Approve-SupplierInfo' form. On the left, there's a sidebar with 'Components' and 'Model Fields' sections. The main area displays three input fields: 'user' (with value 'user'), 'delivery date' (with value 'delivery date'), and 'best offer' (with value 'best offer'). A toolbar at the top right includes 'Save', 'Delete', 'Rename', 'Copy', 'Download', 'Latest Version', 'Hide Alerts', and other icons.

## Approve-OrderInfo

The screenshot shows the Form Modeler interface for the 'Approve-OrderInfo' form. On the left, there's a sidebar with 'Components' and 'Model Fields' sections. The main area displays four input fields: 'item name' (with value 'item name'), 'urgency' (with value 'urgency'), 'TargetPrice' (with value 'TargetPrice'), and a checkbox labeled 'approved'. A toolbar at the top right includes 'Save', 'Delete', 'Rename', 'Copy', 'Download', 'Latest Version', 'Hide Alerts', and other icons.

**Don't forget to save all your forms!!!**

The implementation of our process is complete. It's now time to deploy and test our application.

### 19.7 Deploying the Process Service

With our **Order Management** project's process, decisions and forms completed, we can now package our project in a Deployment Unit (KJAR) and deploy it on the Execution Server. To do this:

1. Go back to our project's Library View (for example by clicking on the `Order Management` link in the breadcrumb navigation in the upper-left of the screen).
2. Click on the **Deploy** button in the upper-right corner of the screen. This will package our project in a Deployment Unit (KJAR) and deploy it onto the Execution Server (KIE-Server).
3. Go to the **Execution Servers** perspective by clicking on "Menu → Deploy → Execution Servers". You will see the **Deployment Unit** deployed on the Execution Server.

### 19.8 Execute the process

In this section, you will execute the process deployed on the Process Execution Server via the Business Central workbench.

1. Navigate to **Menu → Manage → Process Definitions**. If everything is correct, the `order-management` process will be listed. Click on the kebab icon of the `order-management` process and click on **Start**.

The screenshot shows the Business Central interface with the title 'Manage Process Definitions'. A single row is listed in the table:

Name	Version	Deployment	Actions
OrderManagement	1.0	order-management_1.0-SNAPSHOT	<span style="color: blue;">⋮</span>

At the bottom, there are buttons for 'Save Filters' and 'Clear All', and a page navigation bar showing '10 Items' and '1 of 1'.

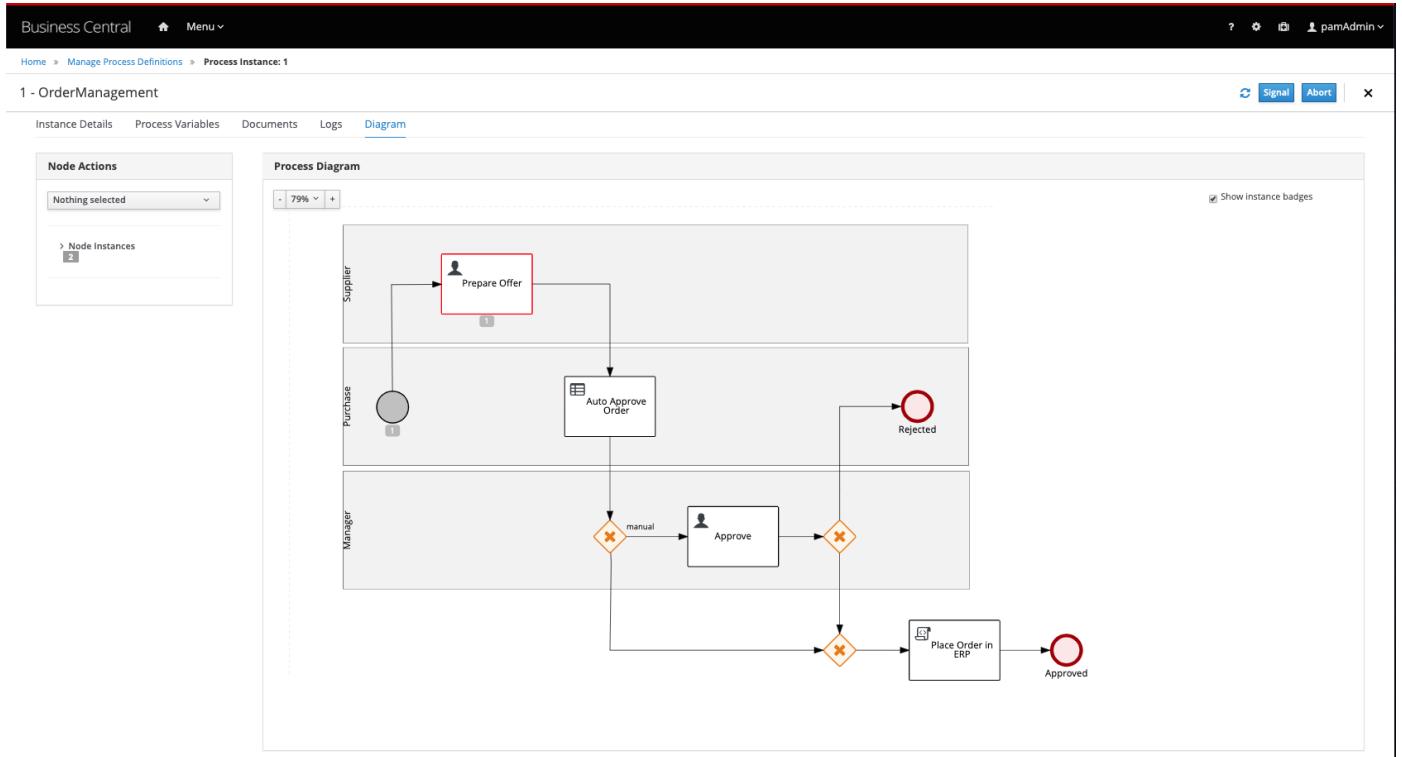
2. In the form that opens, pick the **Huawei P10 Phone** as the item and set the urgency to **low**. Set the target price to **700** and set the supplier name to the name of your own Business Central user (e.g. `pamAdmin`). Click on **Submit**.

The screenshot shows the 'OrderManagement' form with the following data:

- Correlation key:** (Collapsed)
- Form:** (Collapsed)
- Order Information:**
  - item name:** Huawei P10
  - urgency:** low (selected radio button)
- TargetPrice:** 700
- SupplierInfo:**
  - supplier name:** pamAdmin

A large blue 'Submit' button is located at the bottom right of the form.

3. In the process instance details screen that opens, click on the **Diagram** tab to open the process instance diagram, which shows the current state of the process.



4. The process is in a wait state at the `Prepare Offer` task. Navigateto **Menu** → **Track Task Inbox**\*\*. Click on the `Prepare offer` task to open its task window.
5. Click on the **Start** button to start working on the task. Because the task has been assigned to a single user (via `#{{supplierInfo.user}}`), you don't have to first *claim* the task.
6. Select a random delivery date. Set the best offer to **900**. Click on **Complete**.

**1 - Prepare Offer**

**Work** Details Assignments Comments Admin Logs

**OrderInfo**

item name: Huawei P10      urgency:  high

**Your Best Offer**

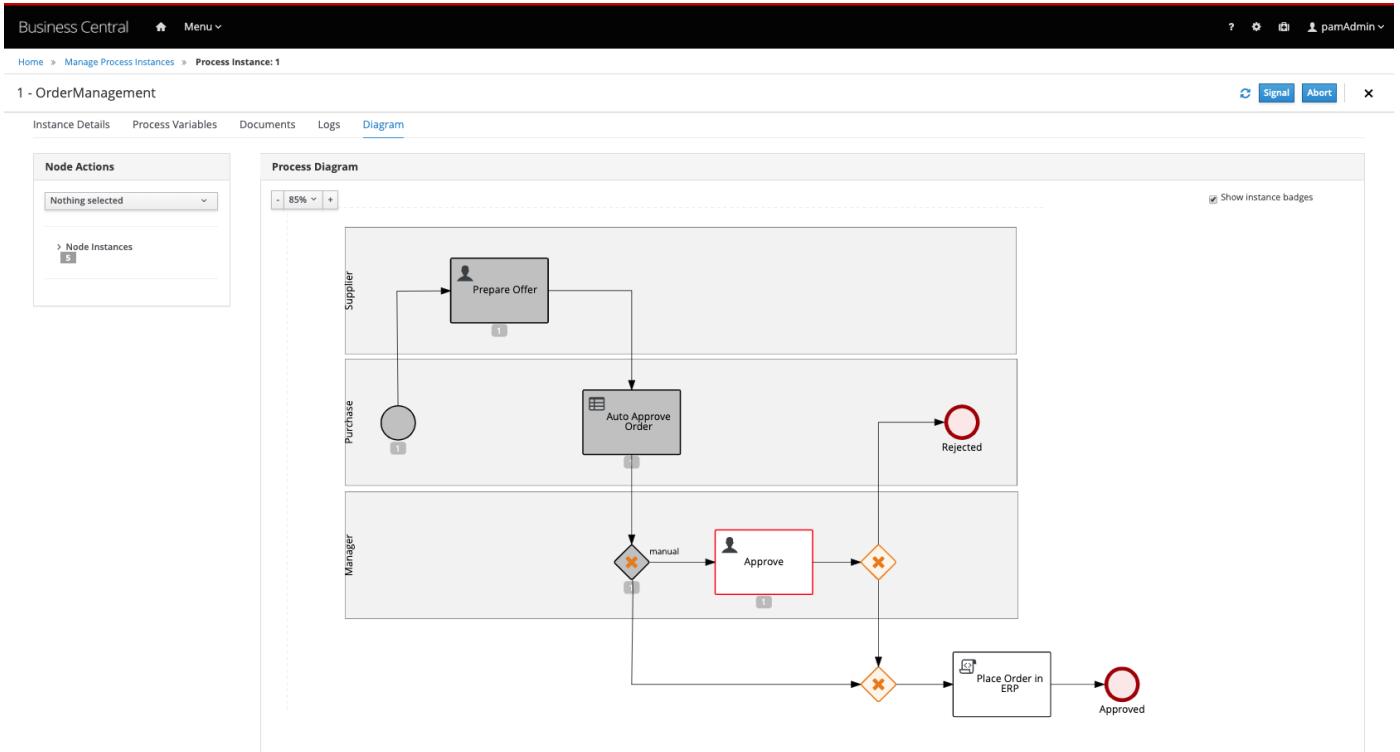
delivery date\*: 03/11/2020      best offer\*: 900

**Comments**

Add a comment.

There are no Comments for this task.

7. The process will continue to the `Auto Approve Order` decision node. Because of the target prices set, and the offered price, the decision will evaluae to `false`. Hence, the process will continue to the `Approve` task.



8. Go back to the **Task Inbox** and open the **Approve** task. Click on **Claim** and on **Start**. In this form we can approve or disapprove the order via the **approved** checkbox, and specify a **rejection reason** if we reject the order. Approve the task by checking the **approved** checkbox and clicking on **Complete**:

The screenshot shows the Business Central Task Inbox for task 2. The top navigation bar includes 'Business Central', 'Home', 'Task Inbox', and 'Task: 2'. The main area displays the task details:

**SupplierInfo**

user	delivery date	best offer
pamAdmin	delivery date	900

**OrderInfo**

item name	urgency	TargetPrice
Huawei P10	low	700

approved

**Comments**

Add a comment.

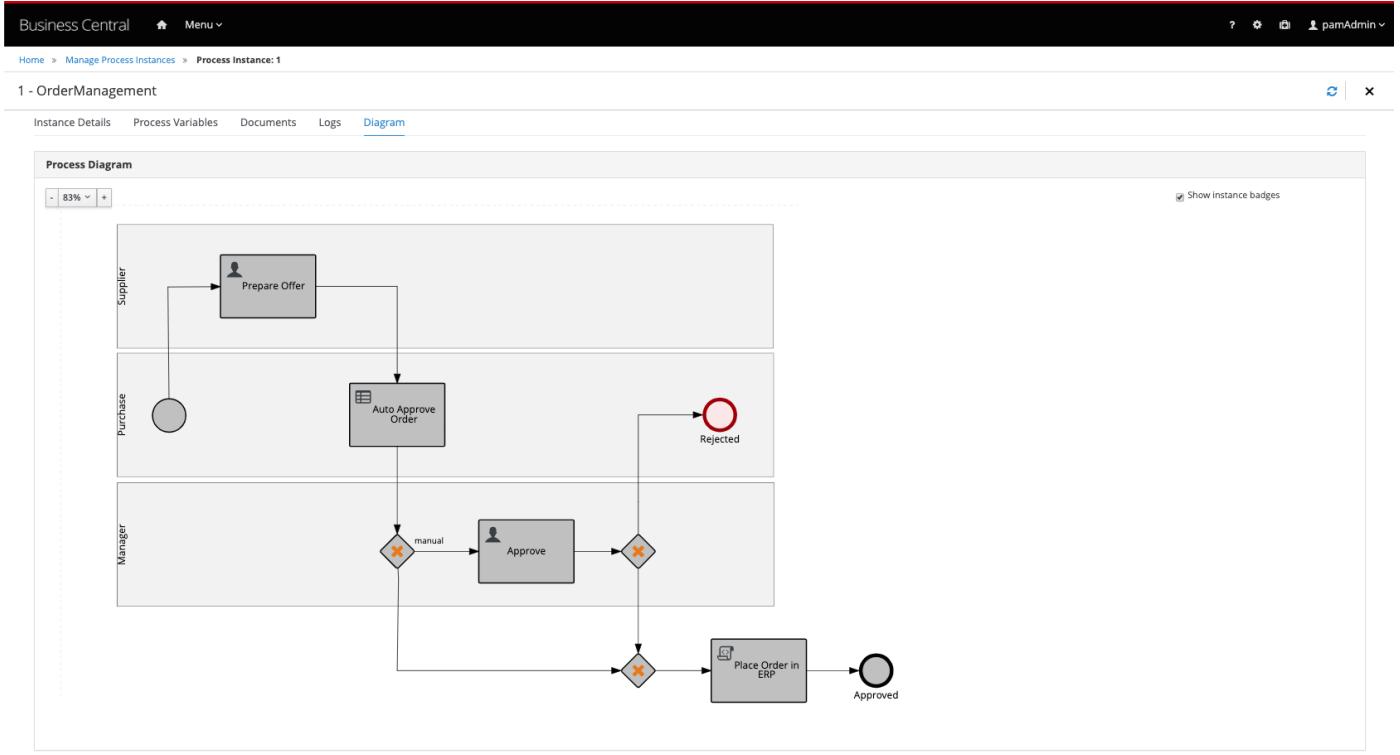
There are no Comments for this task

Buttons at the bottom: Save, Release, Complete.

9. Go back to the process instance view and observe that the process instance is gone.

10. Enable the **Completed** checkbox in the **State** filter on the left-hand-side of the screen. Observe that we can see our process instance in the list.

11. Open the process instance, open its **Diagram** tab. Observe that the order has been accepted:



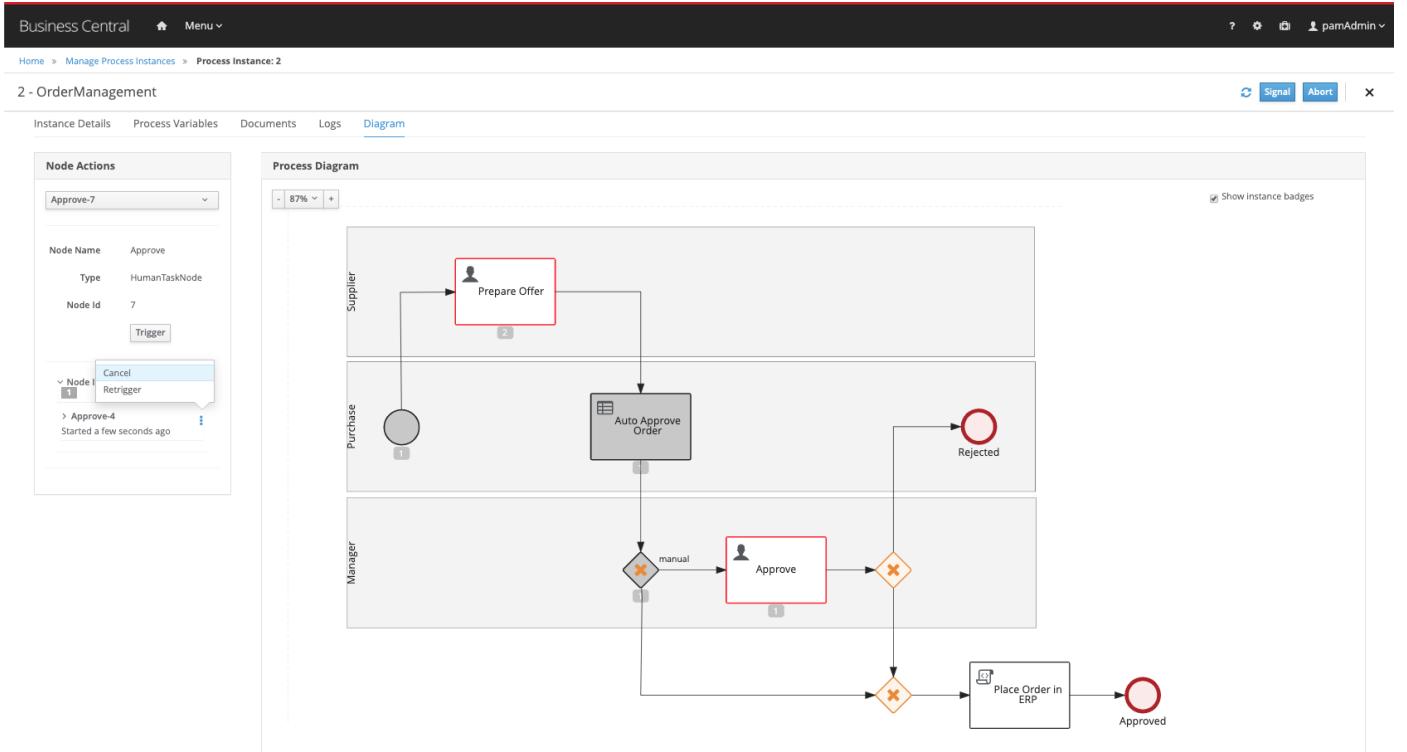
Run a couple more process instances with different values to test, for example, the functionality of the [Automated Approval Rules](#).

### 19.9 Correcting problems and errors

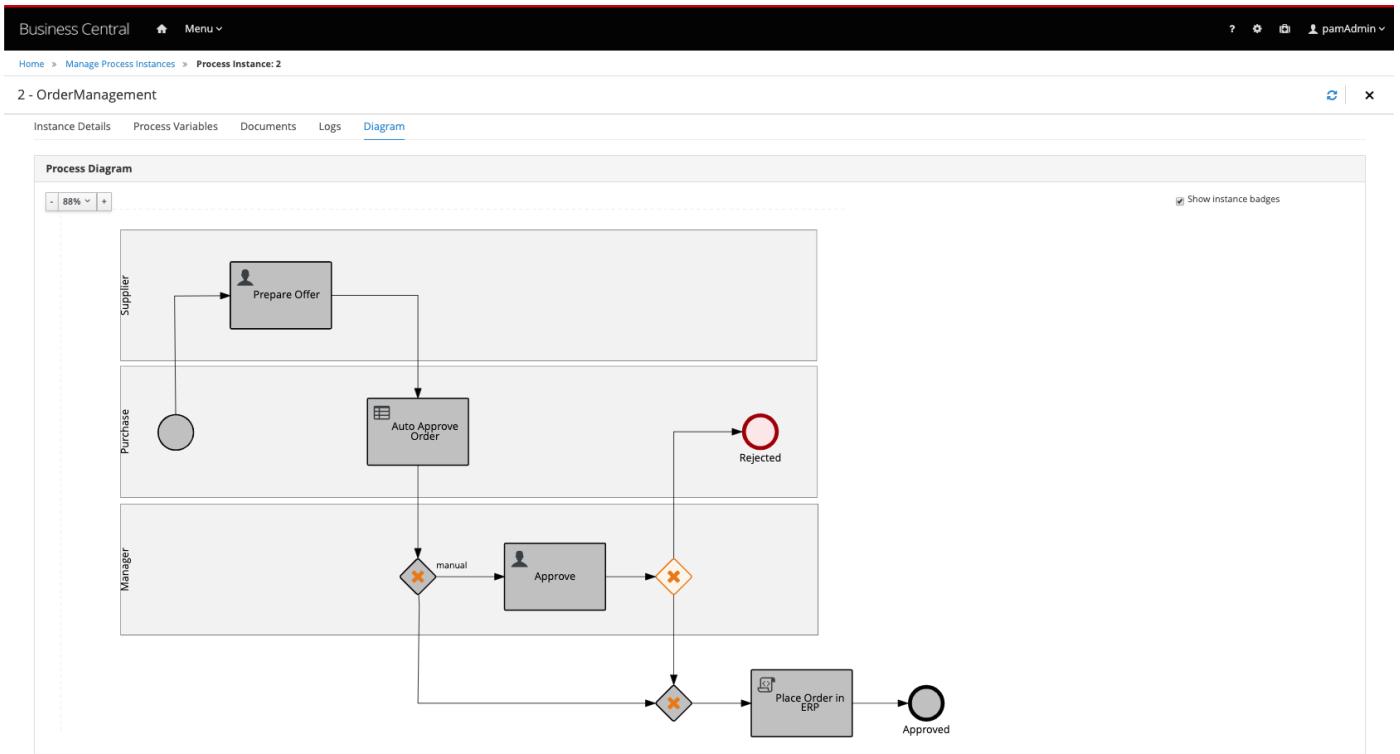
During process instance execution, a lot of things can go wrong. Users might fill in incorrect data, remote services are not available, etc. In an ideal world, the process definition takes a lot of these possible problems into account in its design. E.g. the process definition might contain exception handling logic via boundary catching error events and retry-loops. However, there are situations in which an operator or administrator would like to manually change the process to another state for example,

restart an already completed **User Task**. In the latest version of IBM Business Automation Open Edition 8.0 this is now possible via the **Process Instance interface** in Business Central.

1. Start a new process instance of our **Order Management** process.
2. Complete the `Prepare Offer` task in such a way that the order is not automatically approved and the process will hit the `Approve User Task` wait state.
3. Go to the **Process Instances** view and select the process instance. Navigate to the **Diagram** tab. Observe that the process is waiting in the `Approve User Task`.
4. Click on the `Prepare Offer` node to select it. In the **Node Actions** panel on the left-hand-side of the screen, verify that the `Prepare Offer` node is selected and click on **Trigger**. Observe that the `Prepare Offer User Task` has been activated.
5. Although we have re-activated the `Prepare Offer` node, we have not yet de-activated the `Approve` task. Click on the active `Approve` task and expand the **Node Instances** section in the **Node Actions** panel. Click on the kebab icon of the active `Approve` instance and click on **Cancel**:



6. Open the **Task Inbox**. Observe that the `Approve User Task` is gone and that we have a new `Prepare Offer` task.
- Open the `Prepare Offer` task, set the price to a price which will trigger the rules to automatically approve the order, and complete the task. Go to the process instances view and observe that the process instance has been completed. Enable the **Completed** filter in the **State** filter panel on the left-hand-side of the screen. Open the completed process instance and open its *Diagram* tab.



#### 19.10 Execute the process via APIs

The Execution Server provides a rich RESTful API that allows user to interact with the process engine and deployed processes via a REST. This powerful feature allows users to create modern user interface and applications in their technology of choice (e.g. Entando DXP, ReactJS/Redux, AngularJS, etc.) and integrate these applications with the process engine to create modern, process driven, enterprise applications.

The Swagger interface provides the description and documentation of the Execution Server's RESTful API. At the same time, it allows the APIs to be called from the UI. This enables developers and users to quickly test a, in this case, a deployed business process.

1. Navigate to the [KIE Server Swagger Page](#)
2. Locate the **Process instances** section. The Process Instances API provides a vast array of operations to interact with the process engine.
3. Locate the **POST** operation for the resource `/server/containers/{containerId}/processes/{processId}/instances`. This is the RESTful operation with which we can start a new process instance. Expand the operation:

The screenshot shows the KIE Server Swagger UI for the POST operation at `/server/containers/{containerId}/processes/{processId}/instances`. The operation description is "Starts a new process instance of a specified process".

**Parameters** (Section):
 

- containerId** (required, string, path): container id where the process definition resides
- processId** (required, string, path): process id that new instance should be created from
- body**: optional map of process variables

**Example Value** (Model):
 

```
{
      "age": 25,
      "person": {
        "Person": {
          "name": "john"
        }
      }
    }
```

**Parameter content type**: application/json

**Responses** (Section):
 

- Code**: 201, **Description**: Process instance started

**Response content type**: application/json

4. Click on the **Try it out** button.

- Set the **containerId** to `order-management` (in this case we use the alias of the container).
- Set the **processId** to `order-management.OrderManagement`.
- Set **Parameter content type** to `application/json`.

5. Set the **Response content type** to `application/json`.

6. Set the **body** to:

```
{
  "orderInfo" : {
    "com.myspace.order\_management.OrderInfo" : {
      "item" : "Huawei P10", "urgency" : "low", "price" : 0.0, "targetPrice": "700.0"
    }
  },
  "supplierInfo" : {
    "com.myspace.order\_management.SupplierInfo" : { "user" : "bamAdmin" }
  }
}
```

**POST** /server/containers/{containerId}/processes/{processId}/instances Starts a new process instance of a specified process.

**Parameters**

Name	Description
<b>containerId</b> * required string (path)	container id where the process definition resides order-management
<b>processId</b> * required string (path)	process id that new instance should be created from order-management.OrderManagement
<b>body</b> (body)	optional map of process variables

Example Value | Model

```
{
  "orderInfo" : {
    "com.myspace.order_management.OrderInfo" : {
      "item" : "Huawei P10",
      "urgency" : "low",
      "price" : 0.0,
      "targetPrice": "700.0"
    }
  },
  "supplierInfo" : {
    "com.myspace.order_management.SupplierInfo" : {
      "user" : "bamAdmin"
    }
  }
}
```

**Cancel**

Parameter content type  
application/json

**Execute** **Clear**

7. Click on the *Execute* button.
  8. If requested, provide the username and password of your **Business Central** and **KIE-Server** user (in this example we have been using u: `bamAdmin`, p: `ibmpam1!`).
  - INFO: If you're using the Linux environment on Skytap use the `pamadmin:pamadm1n` information
  9. Inspect the response. Note that the operation returns the process instance id of the started process.
  10. Go back to the Business Central workbench. Go the process instances view and inspect the process instance we have just started.
-

The RESTful API provides many more operations. Let's use the API to fetch our **Task List** and complete the Request Offer task.

1. In the Swagger API, navigate to the **Process queries** section.
2. Find the **GET** operation for the resource `/server/queries/tasks/instances/pot-owners`. Expand the operation and click on the **Try it out** button.
3. Make sure the **\*Response content type** is set to `application/json`. Leave all the other fields set to their default values.
4. Click on the **Execute** button. This will return all the tasks for our user (in the case of this example this is the `bamAdmin` user).

Server response

Code	Details
200	<p>Response body</p> <pre>{   "task-summary": [     {       "task-id": 15,       "task-name": "Prepare Offer",       "task-subject": "Prepare Offer for Huawei P10",       "task-description": "Prepare Offer for Huawei P10",       "task-status": "Reserved",       "task-priority": 0,       "task-is-skippable": false,       "task-actual-owner": "bamAdmin",       "task-created-by": null,       "task-created-on": {         "java.util.Date": 1583758866991       },       "task-activation-time": {         "java.util.Date": 1583758866991       },       "task-expiration-time": null,       "task-proc-inst-id": 12,       "task-proc-def-id": "order-management.OrderManagement",       "task-container-id": "order-management_1.0-SNAPSHOT",       "task-parent-id": -1     }   ] }</pre> <p>Response headers</p> <pre>cache-control: no-cache, no-store, must-revalidate connection: keep-alive content-length: 698 content-type: application/json date: Mon, 09 Mar 2020 13:01:34 GMT expires: 0 pragma: no-cache server: JBoss-EAP/7 x-powered-by: Undertow/1</pre>

5. We can see the `Prepare Offer` task that is available in our inbox. Let's complete this task.
6. Go to the **Task Instances** section in the Swagger interface and locate the **PUT** operation of the `/server/containers/{containerId}/tasks/{taskId}/states/completed` resource. This is the operation with which we can complete a task.

- Set the **containerId** to `order-management`.
- Set the **taskId** to the id of the task instance you want to complete. The task instance id can be found in the list of task instances we got back from our previous REST operation.
- Set **auto-progress** to `true`. This controls the auto progression of the tasks through the various states of the task lifecycle (i.e. claimed, started, etc.)
- Set the **Parameter content type** to `application/json`.
- Set the **Response content type** to `application/json`.
- Set the **body** to:

```
{
  "supplierInfo": {
    "com.myspace.order\_management.SupplierInfo" : {
      "user": "bamAdmin",
      "offer": "900",
      "deliveryDate": "2020-03-11T12:00:00.000Z"
    }
  }
}
```

 INFO: If you're using the Linux environment on Skytap use the following.

```
{
  "supplierInfo": {
    "com.myspace.order\_management.SupplierInfo" : {
      "user": "pamadmin",
      "offer": "900",
      "deliveryDate": "2020-03-11T12:00:00.000Z"
    }
  }
}
```

```
}
```

- Click on the **Execute** button. If you've entered everything correctly, the task will be completed and the process will move to the next wait state, the `Prepare Offer` task. Go back to the Business Central workbench. Go to the process instances view. Select the process instance of the task you've just completed. Observe that the `Prepare Offer` task has been completed and that the process is now waiting on the `Approve User Task`.

The rest of the tasks can be completed in the same way via the API.

### 19.11 Using the KIE-Server Client

IBM Business Automation Open Edition 8.0 provides a KIE-Server Client API that allows the user to interact with the KIE-Server from a Java client using a higher level API. It abstracts the data marshalling and unmarshalling and the creation and execution of the RESTful commands from the developer, allowing him/her to focus on developing business logic.

In this section we will create a simple Java client for our Order Management process.

1. Create a new Maven Java JAR project in your favourite IDE (e.g. IntelliJ, Eclipse, Visual Studio Code).
2. Add the following dependency to your project:

```
<dependency>
<groupId>org.kie.server</groupId>
<artifactId>kie-server-client</artifactId>
<version>7.67.2.Final-redhat-00006</version>
<scope>compile</scope>
</dependency>
```

3. Create a Java package in your `src/main/java` folder with the name `com.myspace.order_management`.
4. Download the `OrderInfo.java` file from [this](#) location and add it to the package you've just created.
5. Download the `SupplierInfo.java` file from [this](#) location and add it to the package.
6. Create a new Java class called `Main`. Add a `public static void main(String[] args)` method to your main class.
7. Before we implement our method, we first define a number of constants that we will need when implementing our method (note that the values of your constants can be different depending on your environment, model namespace, etc.):

```
private static final String KIE_SERVER_URL = "http://localhost:8080/kie-server/services/rest/server";
private static final String CONTAINER_ID = "order-management";
private static final String USERNAME = "bamAdmin";
private static final String PASSWORD = "ibmpam1!";
private static final String PROCESS_ID = "order-management.OrderManagement";
```

 INFO: If you're using the Linux environment on Skytap use the following.

```
private static final String KIE_SERVER_URL = "http://localhost:8080/kie-server/services/rest/server";
private static final String CONTAINER_ID = "order-management";
private static final String USERNAME = "pamadmin";
private static final String PASSWORD = "pamadmin";
private static final String PROCESS_ID = "order-management.OrderManagement";
```

8. KIE-Server client API classes can mostly be retrieved from the `KieServicesFactory` class. We first need to create a `KieServicesConfiguration` instance that will hold our credentials and defines how we want our client to communicate with the server:

```
KieServicesConfiguration kieServicesConfig = KieServicesFactory.newRestConfiguration(KIE_SERVER_URL, new EnteredCredentialsProvider(USERNAME, PASSWORD));
```

9. To allow the KIE-Server Client's marshaller to marshall and unmarshall instances of our domain model, we need to add our domain model classes to the `KieServicesConfiguration`.

```
Set<Class<?>> extraClasses = new HashSet<()>;
extraClasses.add(OrderInfo.class);
extraClasses.add(SupplierInfo.class);
kieServicesConfig.addExtraClasses(extraClasses);
```

10. Next, we create the `KieServicesClient`:

```
~~~java KieServicesClient kieServicesClient = KieServicesFactory.newKieServicesClient(kieServicesConfig); ~~
```

11. From this client we retrieve our `ProcessServicesClient`:

```
ProcessServicesClient processServicesClient = kieServicesClient.getServicesClient(ProcessServicesClient.class);
```

12. We now create a `Map` which we will use to pass the process input variables. We create a new `OrderInfo` instance and `SupplierInfo` instance and put them in the `Map`.

```
Map<String, Object> inputData = new HashMap<()>;
OrderInfo orderInfo = new OrderInfo();
orderInfo.setItem("Huawei P10");
orderInfo.setUrgency("low");
inputData.put("orderInfo", orderInfo);

SupplierInfo supplierInfo = new SupplierInfo();
supplierInfo.setUser("bamAdmin"); inputData.put("supplierInfo", supplierInfo);
```

 INFO: If you're using the Linux environment on Skytap use `pamadmin:pamadm1n` as the username password

```
Map<String, Object> inputData = new HashMap<()>;
OrderInfo orderInfo = new OrderInfo();
orderInfo.setItem("Huawei P10");
orderInfo.setUrgency("low");
inputData.put("orderInfo", orderInfo);
```

```
SupplierInfo supplierInfo = new SupplierInfo();
supplierInfo.setUser("pamadmin"); inputData.put("supplierInfo", supplierInfo);
```

13. We can now start a new process instance via the `ProcessServicesClient`.

```
Long processInstanceId = processServicesClient.startProcess(CONTAINER_ID, PROCESS_ID, inputData);
```

14. Finally, we can print the process instance id to `System.out`.

```
System.out.println("New *Order Management* process instance started with instance-id: " + processInstanceId);
```

15. Compile your project and run it. Observe the output in the console, which should say: **New Order Management process instance started with instance-id**

---

The complete project can be found here: <https://github.com/timwuthenow/rhpam7-order-management-demo-repo>

The KIE-Server Client provides more services to interact with the Execution Server:

- `UserTaskServicesClient` : provides functionality to interact with the **UserTask** services, for example to claim, start and complete a **User Task**.
- `CaseServicesClient` : provides functionality to interact with the **Case Management** features of the Execution Server.
- `ProcessAdminServicesClient` : provides the administration API for processes.
- etc.

We leave as an exercise to the reader to try to complete a **User Task**, of the process instance we've just created, using the `UserTaskServicesClient`.

---

Last update: 2023-12-06

## 3.7 Working with event driven processes in IBAMOE

---

### 3.7.1 20. Introduction

In this guided lab let's see in practice how we can use process automation applications that fits within event-driven architectures. We can list at least three ways to adjust our business application fit within EDA:

- We can build processes that can react to events that happen in the ecosystem;
- From within the process, we can emit events to notify the ecosystem about key activities in the business process and interact with external services via events;
- We can track every transaction committed either for business processes, cases (case management), or human tasks by publishing events for.

#### 20.1 The alignment of tech evolution and business standards like BPMN

When providing an implementation for a specification, each provider has the opportunity to deliver the solution of choice. It is not different for the BPMN specification. It allows different implementations for its diagram elements, and this is how IBAMOE delivers the most recent tech concepts by still allowing business users to use the modeling notation they are used to.

In IBAMOE (a.k.a. jBPM), it is possible to make use of message events (starting, intermediate or ending) to interact via events. In this case, the KIE Server Kafka extension makes sure the communication occurs effectively with the event streaming brokers.

In the upcoming labs we will learn how to model the processes and when and how to add configurations to the business project and KIE Server.

---

Last update: 2023-12-06

### 3.7.2 21. IBAMOE 9.0 Kafka extension

In order to be able to start processes based on new events, we will need to configure the IBAMOE Kafka extension.

The IBAMOE Kafka extension allows the KIE Server (process and decision engine) to react to events and publish events to kafka topics.

 INFO: There are several options in IBAMOE to customize the Kafka address, topic names, etc. In our case, we're using the default Kafka address, which is, localhost:9092. More customization information can be found in the official Red Hat product documentation: [Configuring a KIE Server to send and receive Kafka messages from the process](#).

In this setup steps, we will configure IBAMOE only in the *server level - we are not yet configuring the business project*. We will see how to configure the project as we move forward on the labs.

#### 21.1 Enabling the Kafka extension

We can configure the engine to support different capabilities. In order to enable processes to be started through eventing, we only need to enable the extension via system property.

- With IBAMOE up and running, execute the following, where `$JBoss_HOME` is the installation directory of the JBoss EAP instance you're running KIE Server from:

```
$JBoss_HOME/bin/jboss-cli.sh -c
[standalone@localhost:9990 /] /system-property=org.kie.kafka.server.ext.disabled:add(value=false)
[standalone@localhost:9990 /] :shutdown(restart=true)
```

The first command will enable the Kafka extension. Next, we're restarting EAP so that the new configuration is active. You can check EAP logs to confirm it is restarting.

The following output will show up in IBAMOE logs:

```
INFO [org.kie.server.services.impl.KieServerImpl] (ServerService Thread Pool -- 74) Kafka KIE Server extension has been successfully registered as server extension
```

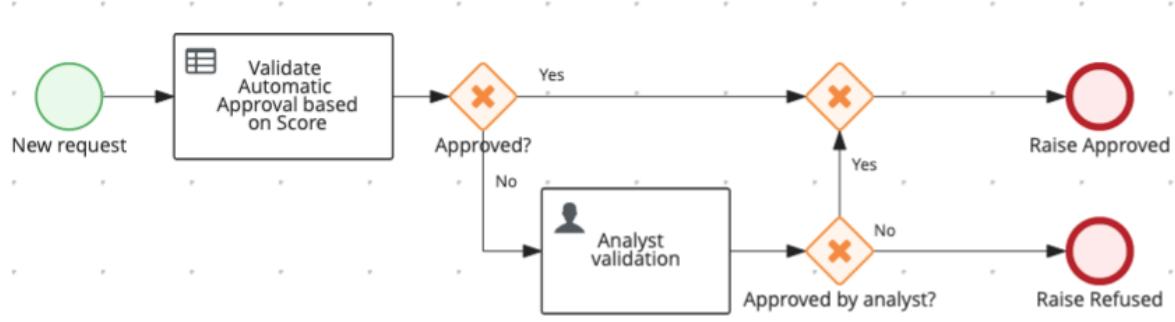
This is the only configuration you will need in a server level to be able to start processes using events.

---

Last update: 2023-12-06

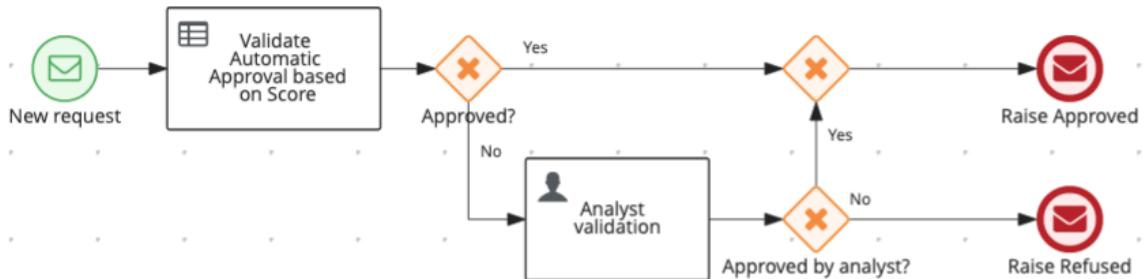
### 3.7.3 22. The Credit Card Raise Approval Project

In this use case we would like to handle the automation of a credit limit increase approval process. Most card issuers allow customers to request an increased credit limit through various entry points such as: websites, their mobile applications or over the phone with customer service. Let's consider we need to deliver this automation for a bank that wants to achieve a similar use case within an event-driven architecture.



The existing process is started via REST. It has a step for automatic request validation using DMN, and if the request not approved, it goes to a manual review queue. If approved, the service responsible for updating the cc limit is invoked via REST (the diagram only represents this REST call with a script task since this is not relevant for this lab's scenario). Finally, the process ends either with an approved or denied request.

Now, with the architecture shift, the service responsible for increasing the credit card limit should not be invoked via REST anymore. The external service now listens to the topic "request-approved" in order to track when to execute the limit raise. The business process should get started based on events, and whenever the process finishes, it should post a message to a specific topic depending on whether the request was approved or not.



Process v2.

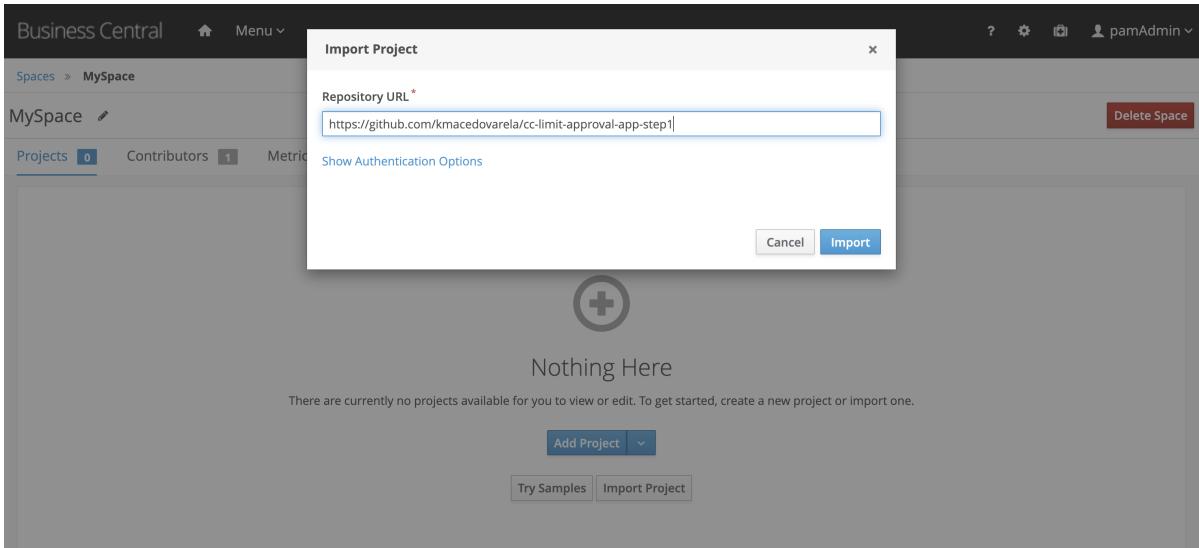
Whenever a new event happens in a topic, a new instance will be triggered. Depending on how this process ends, an event is published in a different topic, therefore, different services can react based on the approval status.

In this strategy we have a resilient way of communication between services where the broker is responsible for storing and providing the events. Adding to that, the tech team can evolve the solutions by using the features available in Kafka itself, like the possibility to replay all the events that happened in a specific time, in chronological order.

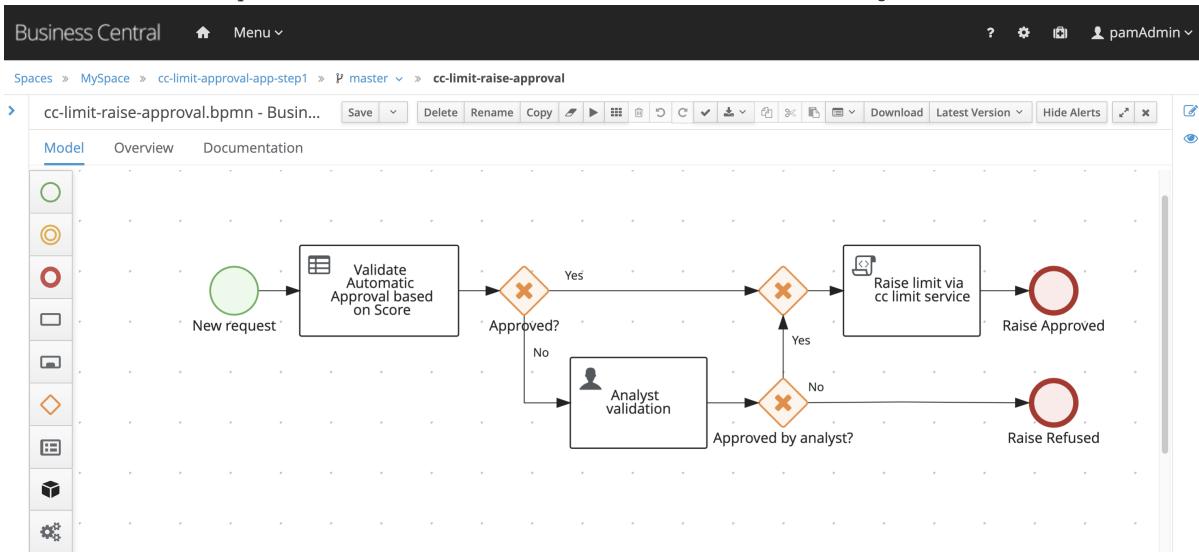
## 22.1 Importing the project

Let's import the existing project so we can start implementing the eventing capabilities.

1. Access Business Central, and import the following project: <https://github.com/kmacedovarela/cc-limit-approval-app-step1>



2. Let's check the existing project. Open the cc-limit-raise-approval process. Notice that currently it is a traditional process, with a standard start and stop node. Processes like this can be started either via REST or JMS.



## 22.2 Reacting to events

The first task we'll do, is to enable the existing process to react to events that are published in a specific topic. Whenever a new event is published, a new process instance should be created.

1. To allow this process definition to be started with events, the first step is to change the start event to a *Start Message Event* by clicking the node in the editor and selecting the *envelope* icon to "Convert into Start Message":



2. Whenever a customer make a new request (independently of the channel used) an event should be published on the `incoming-requests` Kafka topic. This way as new channels are ready to be added, they just need to point to this request instead of modifying for the process end point. With that, a new process instance should be started whenever a new event is published in this topic. Let's configure the *Start Message Event* with the `incoming-requests` topic:

#### ▼ Implementation/Execution

##### Message

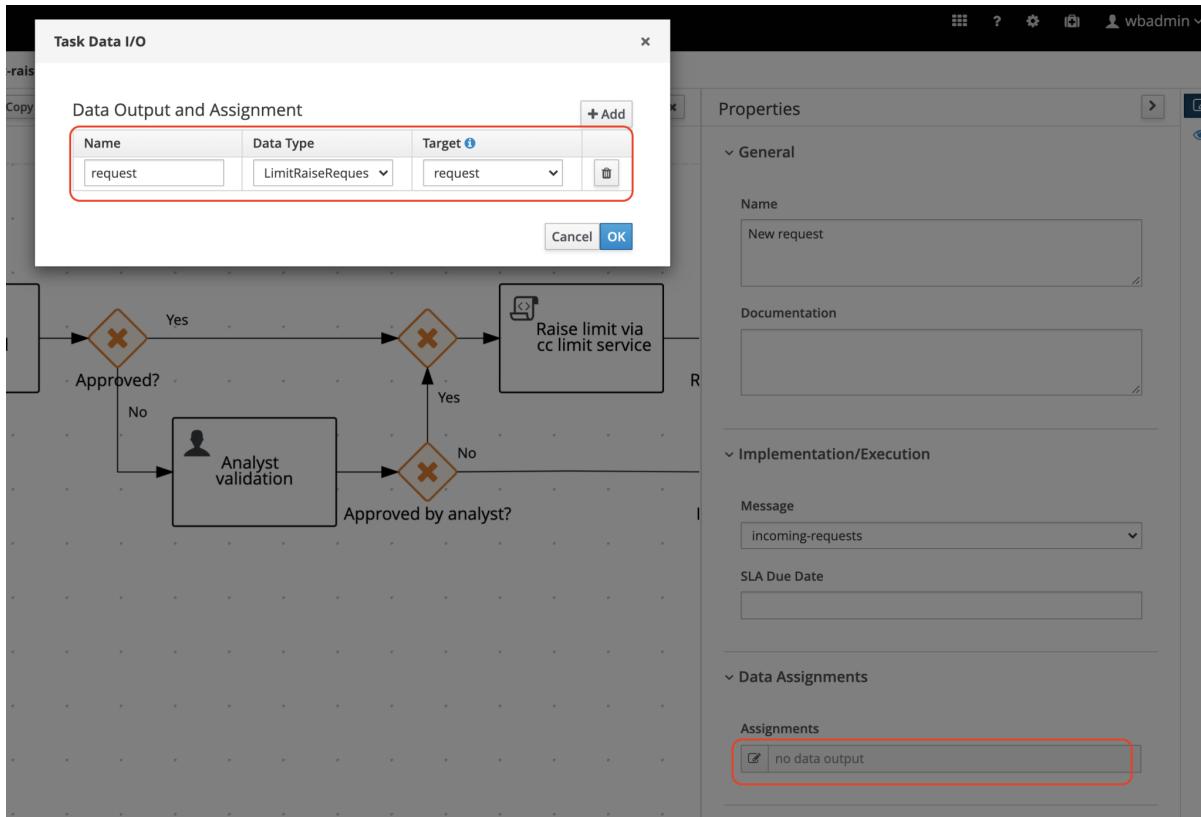
`incoming-requests`

##### SLA Due Date

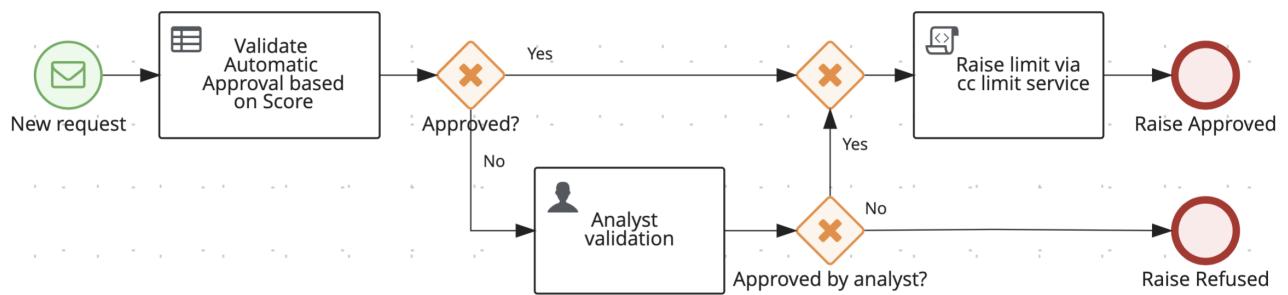
**WARN:** we need to receive the data that is the event data. The KIE Server provides automatic marshalling to help us mapping the input directly to a *Data Object* (a POJO). This project has an object named `LimitRaiseRequest.java` which we will use to receive the incoming data and feed it in the process.

3. On the properties panel of the *Start Message Event*, configure the input data:

- Name: request
- Data Type: `LimitRaiseRequest`
- Target: request



4. Save the process. Your process should now look like this:



### 22.3 Deploying the project

Now, let's deploy and test the project.

1. On the breadcrumb, click on "cc-limit-approval-app-step1" to go back to the Project Explorer view.
2. Click on the "Deploy" button.

## 22.4 Testing the project

Let's publish a new event in the `incoming-requests` topic using the Kafka producer CLI tool.

1. Open a new tab in your terminal and access the `strimzi-all-in-one` project folder.

```
cd ~/enablement/amq-examples/strimzi-all-in-one
```

2. Next, use the Kafka producer to publish new messages on the topic `incoming-requests`.

```
docker-compose exec kafka bin/kafka-console-producer.sh --topic incoming-requests --bootstrap-server localhost:9092
```

3. You can send the following data, and press enter:

```
{"customerId": 1, "customerScore": 250, "requestedValue":1500}
```

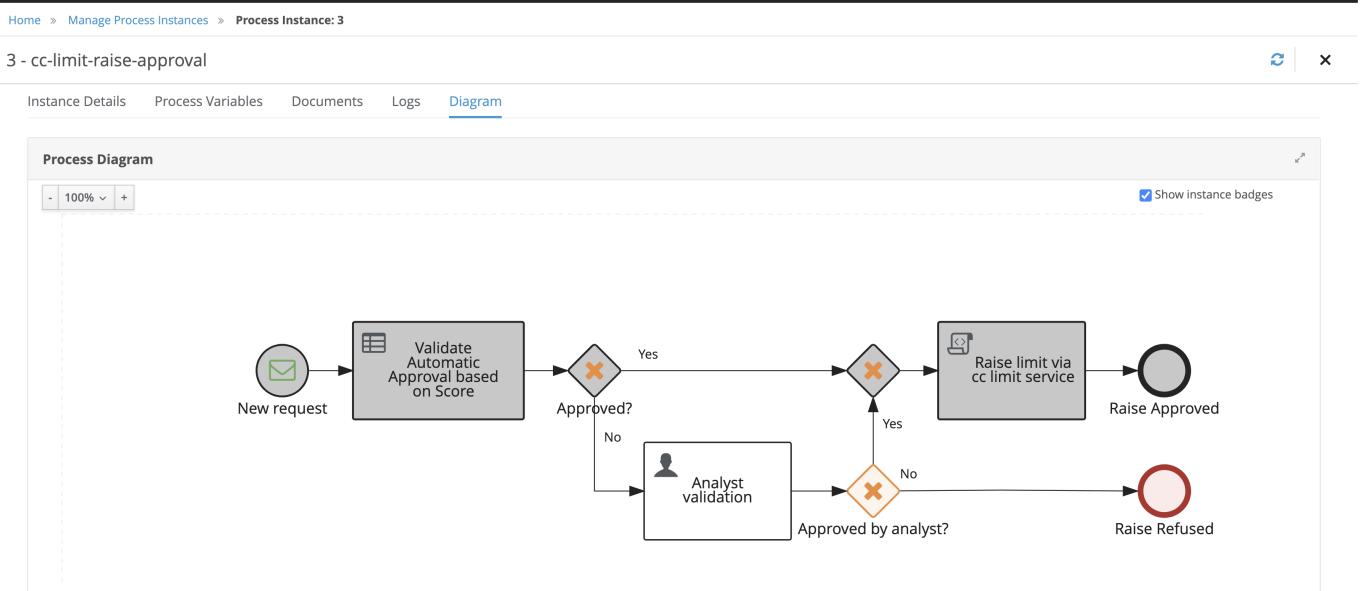
```
± lmaster ✘ ~ docker-compose exec kafka bin/kafka-console-producer.sh --topic incoming-request
s --bootstrap-server localhost:9092
>{"data" : {"customerId": 1, "customerScore": 250, "requestedValue":1500}}
>
```

4. Back to the browser, open Business Central. On the top menu, go to **Menu -> Manage -> Process Instances**.

5. On the left column, filter by "Completed" State. You should see as many instances as the number of events you published on Kafka:

	ID	Name	Description	Version	Last update	Errors	Actions
<input type="checkbox"/>	3	cc-limit-raise-app...	cc-limit-raise-app...	1.0	13-Apr-2021 18:4...	0	⋮
<input type="checkbox"/>	2	cc-limit-raise-app...	cc-limit-raise-app...	1.0	13-Apr-2021 18:3...	0	⋮
<input type="checkbox"/>	1	cc-limit-raise-app...	cc-limit-raise-app...	1.0	13-Apr-2021 18:3...	0	⋮

6. Select a process instance, and next, select the tab **Diagram**. You should see something like:



Last update: 2023-12-06

### 3.7.4 23. Emmitting Events from Business Processes

---

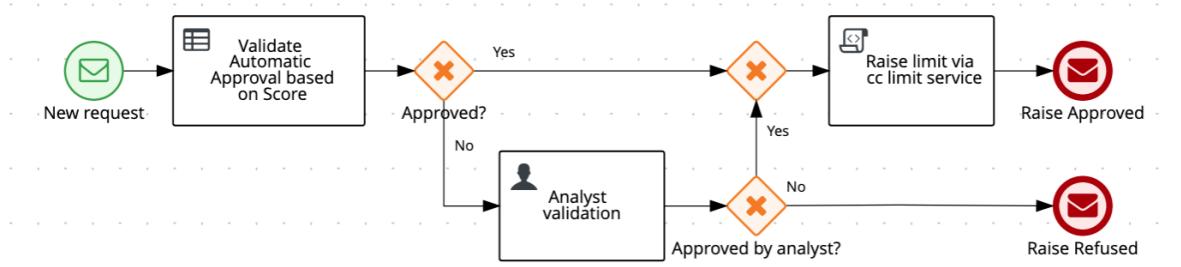
In order to be able to finish processes based on new events, we will need to set up our environment.

In this setup we will:

- Check the required configuration in the business project
- Add bpmn components to emit messages

#### 23.1 Emitting events in a business process

1. Now, we need to *End Message Events* instead of two *End Events*. In Business Central, open the cc-limit-approval-app process.
2. Convert the two *End Events* to *End Message Events*. It should look like this:



1. Next, configure the Kafka topic name in the message name for both nodes as following:

- **Raise Approved** message name: `requests-approved`
- **Raise Denied** message name: `requests-denied`

See below one of the nodes, the *Raise Denied* node configuration:

Properties	
<b>General</b>	
Name	Raise Refused
Documentation	(empty)
<b>Implementation/Execution</b>	
Message	requests-denied
<b>Data Assignments</b>	

2. Save the process definition.

3. On the breadcrumb, click on "cc-limit-approval-app-step1" to go back to the Project Explorer view.

4. Click on the "Deploy" button.

## 23.2 Configuring the business application

1. In Business Central, navigate to the **Project Settings -> Deployments -> Work Item handlers**:

Name	Value	Resolver type	Parameters
Send Task	new org.jbpm.bpmn2.handler.SendTaskHandler()	MVEL	Parameters (0)

Observe that there is a task configured named `Send Task`. In IBAMOE {{ version }} you need this configuration to be able to use any `Message Events` (ending and throwing) that would emit events.

## 23.3 Consuming the events from Kafka topic using Kafka Consumer CLI

In order to validate if our process is emitting processes as we expect, we need to listen to the Kafka topics `requests-approved` and `requests-denied` to validate if the messages were emitted correctly.

1. Open a new terminal tab, and navigate to the Kafka project folder.

```
cd ~/enablement/amq-examples/strmzi-all-in-one/
```

2. Start the Kafka command line tool that allows us to consume events that happen in a topic, and therefore, will allow us to know if IBAMOE published the events when the process ended. The tool is `kafka-console-consumer.sh`. Let's check if the process emitted events on the topic `requests-approved`.

```
docker-compose exec kafka bin/kafka-console-consumer.sh --topic requests-approved --from-beginning --bootstrap-server localhost:9092
```

## 23.4 Testing the solution

To test the solution, we will start a new process instance that will start, be automatically approved, and end without any human interaction. A new process instance should get started whenever you publish a new event on the `incoming-requests` topic, and,

when there is an automatic approval, the process will end and publish an event to the `requests-approved` topic. Let's see this in action:

- Like we did on the first lab, let's start a new process instance by publishing a message in the `incoming-requests` topic. If you canceled the execution of the kafka producer, here's how you can start it:

```
docker-compose exec kafka bin/kafka-console-producer.sh --topic incoming-requests --bootstrap-server localhost:9092
```

You can use the following data in your event: `{"data" : {"customerId": 1, "customerScore": 250, "requestedValue":1500}}`

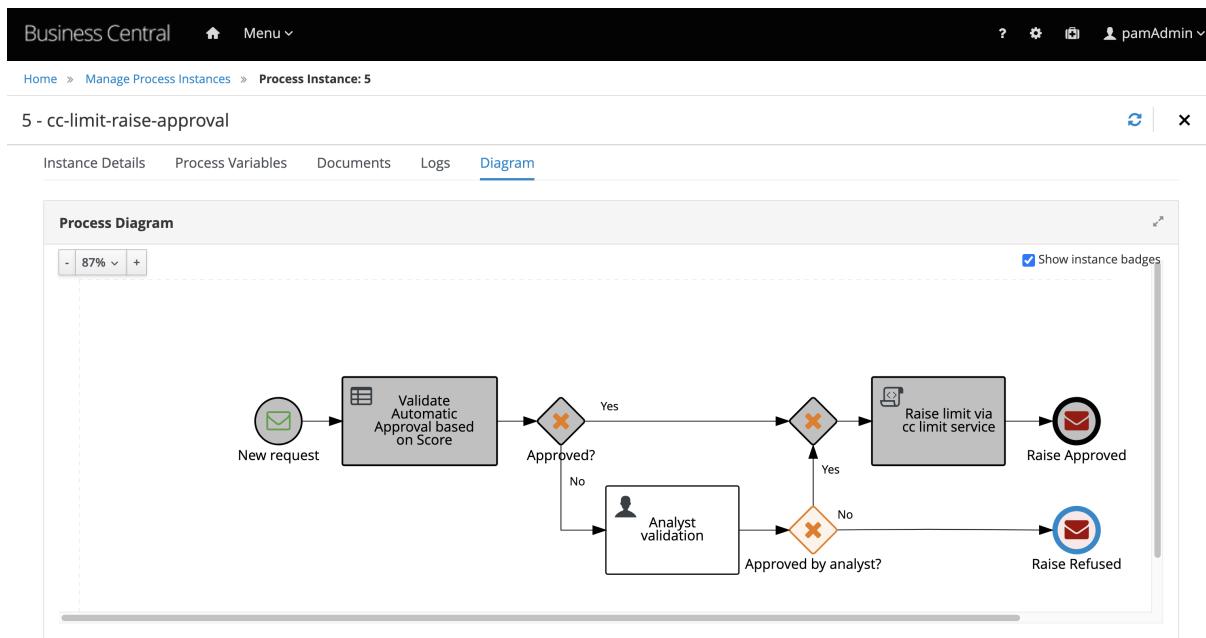
- When you hit return, the data is published to the `incoming-requests` topic
- Kafka reads the event from the `incoming-requests` and automatically instantiates a new process with this data.
- Now check the terminal where you are consuming the messages in the `requests-approved` topic. You should see a new event published by your process. The event will look like this (though not on multiple lines):

```
{
  "specversion": "1.0",
  "time": "2021-04-14T18:04:42.532-0300",
  "id": "25ba2dd0-a8d0-4cfc-9ba4-d2e556ffb4d0",
  "type": "empty",
  "source": "/process/cc-limit-approval-app.cc-limit-raise-approval/5",
  "data": null
}
```

- Identify the process ID on the event above. In this example, the process instance that emitted this event was process of ID **5**.
- Let's check this same process instance in Business Central. In Business Central, open the **Menu -> Manage -> Process Instances**.
- On the left column, filter by "Completed" State. You should see as many instances as the number of events you published on Kafka.

Id	Name	Description	Version	Last update	Errors	Act
6	cc-limit-raise-...	cc-limit-raise-...	1.0	14-Apr-2021 ...	0	
5	cc-limit-raise-...	cc-limit-raise-...	1.0	14-Apr-2021 ...	0	
4	cc-limit-raise-...	cc-limit-raise-...	1.0	14-Apr-2021 ...	0	
3	cc-limit-raise-...	cc-limit-raise-...	1.0	13-Apr-2021 ...	0	
2	cc-limit-raise-...	cc-limit-raise-...	1.0	13-Apr-2021 ...	0	
1	cc-limit-raise-...	cc-limit-raise-...	1.0	13-Apr-2021 ...	0	

- Identify your process instance ID. In this example, instance with id **5**. Select the process instance.
- Next, select the tab `Diagram`. You should see something like:



Last update: 2023-12-06

### 3.7.5 24. Auditing with Kafka

When using the Kafka extension in IBM Business Automation Open Edition 8.0, every transaction for processes, cases and tasks execution can be tracked via events. For each of these categories, we'll have an event emitted to a Kafka topic, in other words, we'll have three topics here: `jbpm-processes-events`, `jbpm-tasks-events`, `jbpm-cases-events`.

To enable this feature, you need to add the `jbpm-event-emitters-kafka` library to the engine, KIE Server. This can either be downloaded in the [community repository for jBPM](#) or via the Red Hat customer portal: [rhpam-7.10.0-maven-repository.zip](#).

The maven repository have ~1.5GB. In order to facilitate the execution of this lab, you can download the `jbpm-event-emitters-kafka` for IBAMOE {{ version }} [here](#); **FIGURE OUT THE REPLACEMENT**

1. Stop IBAMOE.
2. Download the `jbpm-event-emitters-kafka`. It's name will be similar to `jbpm-event-emitters-kafka-7.x.x.Final-redhat-x.jar`.
3. Since this is a behavior only needed by the engine, place the library inside the `kie-server.war` folder, inside the `WEB-INF` directory.

**TIP:** If you downloaded the maven repository zip file in the Red Hat Customer Portal, you can find the jar inside the folder `maven-repository/org/jbpm/jbpm-event-emitters-kafka/7.67.2.Final-redhat-00006/jbpm-event-emitters-kafka-7.67.2.Final-redhat-00006.jar`

```
cp jbpm-event-emitters-kafka-7.67.2.Final-redhat-00006.jar $JBoss_EAP/standalone/deployments/kie-server.war/WEB-INF/lib/
```

4. Next, start IBAMOE server.

Let's check the auditing behavior.

#### 24.1 Testing the feature

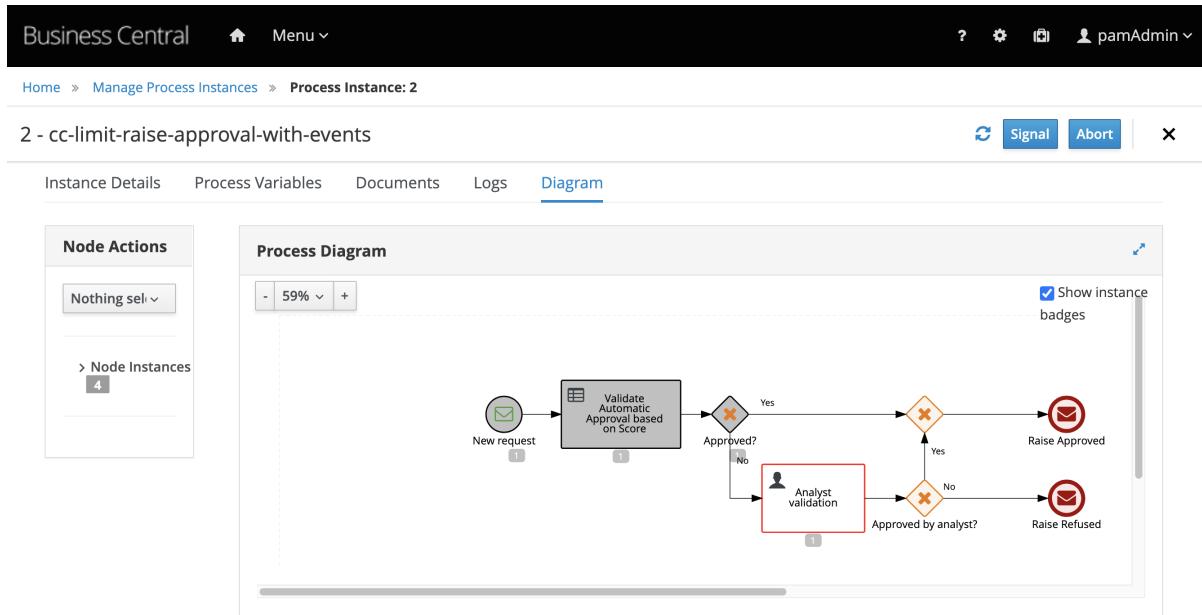
To check the auditing capabilities you can start new processes, interact with human tasks and track the events that are being published on the `jbpm-tasks-events` and `jbpm-processes-events` topics. The event tracking are active also for processes that doesn't use *message events* elements.

In this example we will check the behavior for our event driven business application.

1. Start a new process by emitting an event. Let's start a process that will not be automatically approved. In this way, we will also have a human task created. You can emit the following event to the `incoming-requests` topic:

```
{"customerId": 1, "customerScore":100, "requestedValue": 1200}
```

2. You should be able to see a new process instance can be seen in Business Central in the following status:



1. You can use the kafka consumer CLI script to check the messages that were emitted on the topics: `jbpm-processes-events` and `jbpm-tasks-events`.

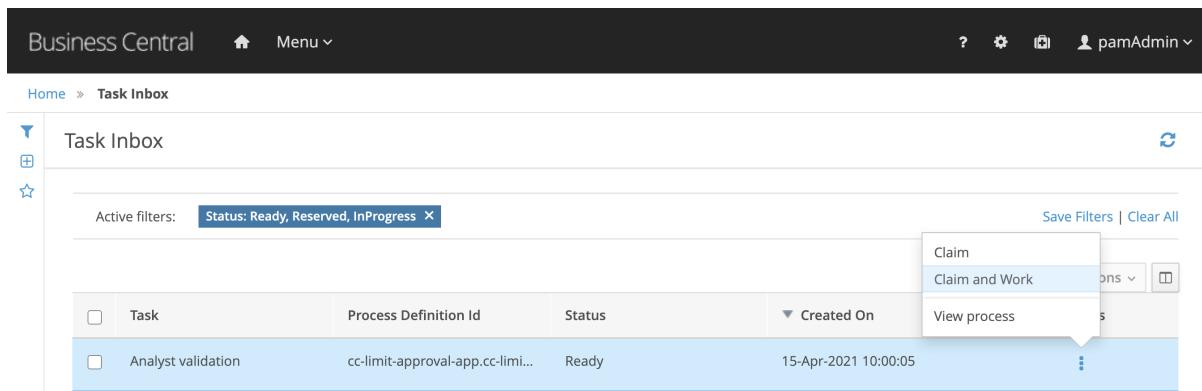
2. You should be able to see an event like this published on the `jbpm-process-events`:

```
{"specversion": "1.0", "time": "2022-09-15T10:00:05.609-0300", "id": "28e13bc0-1c92-42fd-8909-b48a206325d3", "type": "process", "source": "/process/cc-limit-approval-app.cc-limit-raise-approval-with-end-events/2", "data": {"compositeId": "default-kieserver_2", "id": 2, "processId": "cc-limit-approval-app.cc-limit-raise-approval-with-end-events", "processName": "cc-limit-raise-approval-with-events", "processVersion": "1.0", "state": 1, "containerId": "cc-limit-approval-app_1.0.0-SNAPSHOT", "initiator": "unknown", "date": "2021-04-15T10:00:05.608-0300", "processInstanceId": "cc-limit-raise-approval-with-events", "correlationKey": "2", "parentId": -1, "variables": {"request": {"customerId": 1, "requestedValue": 1200, "customerScore": 100, "denyReason": null}, "approval": false, "initiator": "unknown"}}}
```

3. You should be able to see an event like this published on the `jbpm-tasks-events`:

```
{"specversion": "1.0", "time": "2022-09-15T10:00:05.612-0300", "id": "2ac83d91-40d7-49f3-a114-2b72816a20a4", "type": "task", "source": "/process/cc-limit-approval-app.cc-limit-raise-approval-with-end-events/2", "data": {"compositeId": "default-kieserver_2", "id": 2, "priority": 0, "name": "Analyst validation", "subject": "", "description": "", "taskType": null, "formName": "Task", "status": "Ready", "actualOwner": null, "createdBy": null, "createdOn": "2021-04-15T10:00:05.522", "processInstanceId": 2, "parentId": -1, "processId": "cc-limit-approval-app.cc-limit-raise-approval-with-end-events", "containerId": "cc-limit-approval-app_1.0.0-SNAPSHOT", "potentialOwners": ["kie-server"], "excludedOwners": [], "businessAdmins": ["Administrator", "Administrators"], "inputData": {"Skippable": "false", "request": {"customerId": 1, "requestedValue": 1200, "customerScore": 100, "denyReason": null}, "TaskName": "Task", "NodeName": "Analyst validation", "GroupId": "kie-server"}, "outputData": null}}
```

4. Using Business Central, interact with the human task Analyst Validation, and check the events emitted on the `jbpm-tasks-events`.



You should be able to see at every task change, a new event in the `jbpm-tasks-events`. Also, for every transaction committed for the process, you should see new events on the `jbpm-process-events`.

By now, you have an event-driven process, that can be integrated within an event driven architecture, and furthermore, can be tracked and monitored in an asynchronous way by the usage of events.

The complete project can be found at: <https://github.com/timwuthenow/cc-limit-approval-app>

Last update: 2023-12-06

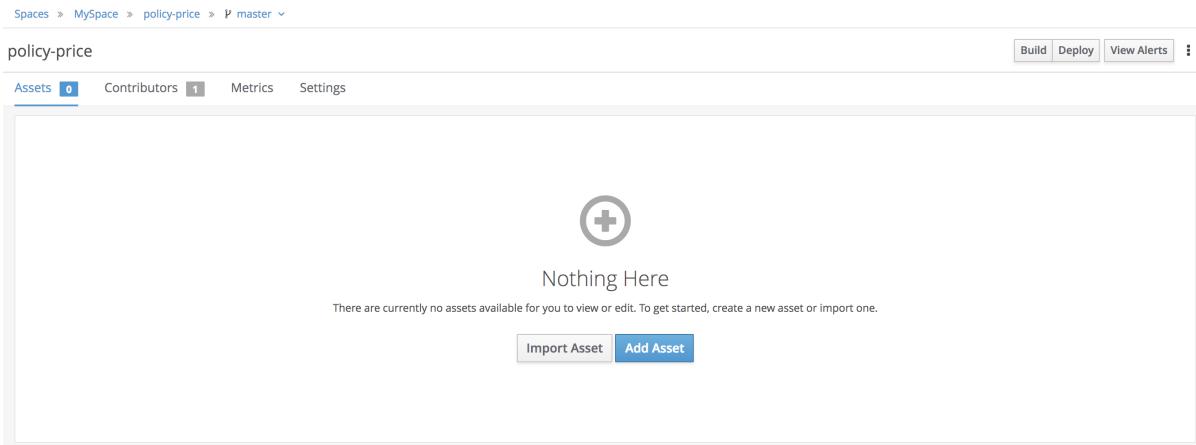
## 3.8 25. Getting started bc

---

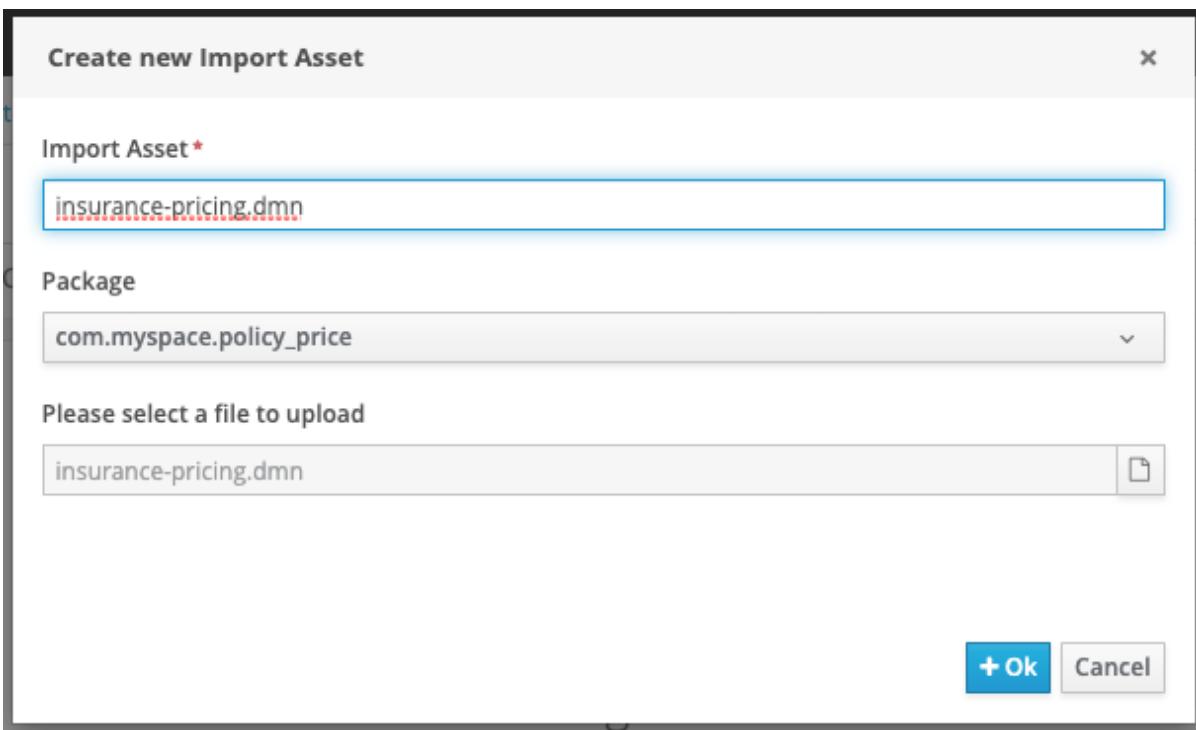
### 3.8.1 25.1 Importing a DMN in Business Central

If you want to try this in Business Central, you can go through the following steps, but it is not required.

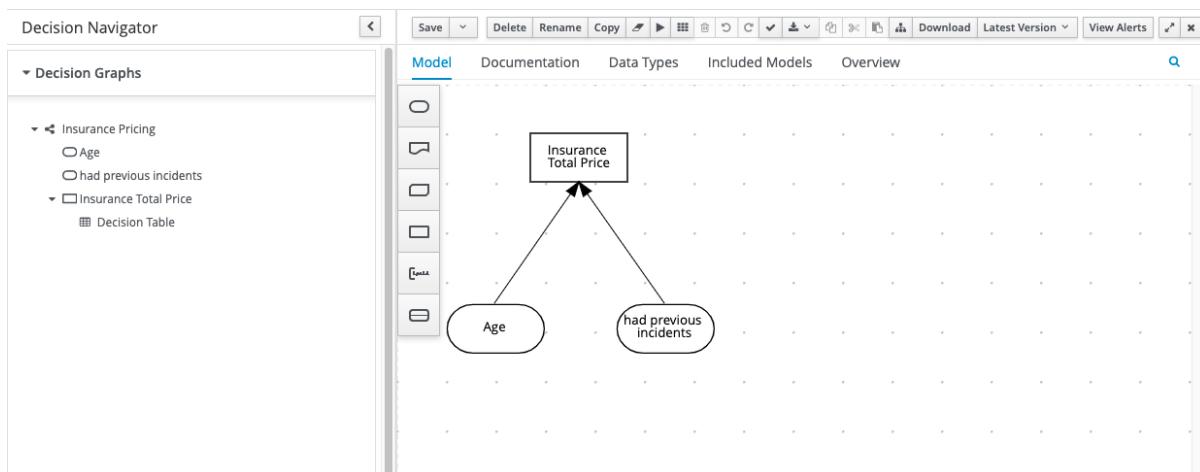
1. From the GitHub web page, click **Clone or download** on the right and then select **Download ZIP**.
2. Using your favorite file system navigation tool, locate the downloaded ZIP file and unzip it to a directory in your file system.
  - From this point forward, this location is referred to as `$PROJECT_HOME`.
3. Log in to Business Central. You can use either `{{ bamAdmin }}:{{ bamPass }}` to do so or whatever login you have created on your instance.
4. Create a project in Business Central called `policy-price`.
5. In the empty project library view for the `policy-price` project, click **Import Asset**.



6. In the **Create new Uploaded file** dialog, enter `insurance-pricing.dmn` in the **Uploaded file** field:



7. Using the browse button at the far right of the field labeled **Please select a file to upload**, navigate with the file browser to the `$PROJECT_HOME` directory where the unzipped Git repository is located.
8. Select the `$PROJECT_HOME\policy-price\insurance-pricing.dmn` file.
9. Click **Ok** to import the DMN asset.
10. The diagram will open and you will be able to see the DRD. Explore the diagram nodes to check the decision policies of this diagram.



11. Close the diagram. You should now be on the library view for the `policy-price` project.

12. You should see the `insurance-pricing` asset is added to your project assets:

13. From the `policy-price` project's library view, click **Build**, then **Deploy** to deploy the project to the execution server.

14. After receiving the build confirmation, navigate to the container deployment list by clicking the "View deployment details" link in the confirmation pop-up, or by selecting **Menu** → **Deploy** → **Execution Servers**.

15. Verify that `policy-price_2.0.0` shows a *green* status:

### 3.8.2 25.2 Testing the Decision Service on KIE Server

In this section, you test the DMN solution using the REST endpoints available in the Decision Server (a.k.a. KIE Server).

1. Open your Decision Server (a.k.a KIE Server) on the url "/docs". You should see something like this:

2. Next, under `DMN Models`, click on the `POST /server/containers/{containerId}/dmn` and select "Try it out":

POST /server/containers/{containerId}/dmn Evaluates decisions for given input

Parameters

Name Description

**containerId** \* required Container id to be used to evaluate decisions on  
string  
(path)

**body** \* required DMN context to be used while evaluation decisions as DMNContextKS type  
(body)  
Example Value Model  
<?xml version="1.0" encoding="UTF-8"?>  
<!-- XML example cannot be generated -->

Parameter content type  
application/xml

Try it out

1. Now use the following data:
2. Container ID: policy-price
3. Body (dmn context): {"dmn-context": {"Age": 20, "had previous incidents": false}}
4. Parameter content type: application/json

**POST** /server/containers/{containerId}/dmn Evaluates decisions for given input

**Parameters**

**Name** **Description**

**containerId** \* required string (path) Container id to be used to evaluate decisions on policy-price

**body** \* required (body) DMN context to be used while evaluation decisions as DMNContextKS type

Example Value Model

```
{"dmn-context": {"Age": 20, "had previous incidents": false}}
```

**Cancel**

Parameter content type application/json

1. Click on the **execute** button. You should see the server response `200` and the results of the decision.

**Server response**

Code	Details
200	<b>Response body</b> <pre>{   "type": "SUCCESS",   "msg": "OK from container 'policy-price'",   "result": {     "dmn-evaluation-result": {       "messages": [],       "model-namespace": "http://www.trisotech.com/definitions/_bb8b9304-b29f-462e-9f88-03d0d868aec5",       "model-name": "Insurance Pricing",       "decision-name": [],       "dmn-context": {         "had previous incidents": false,         "Insurance Total Price": 2000,         "Age": 20       },       "decision-results": {         "_7c68efef-3b20-4807-8d15-7f55995cc8fd": {           "messages": [],           "decision-id": "_7c68efef-3b20-4807-8d15-7f55995cc8fd",           "decision-name": "Insurance Total Price",           "result": 2000,           "status": "SUCCEEDED"         }       }     }   } }</pre>

2. Try out the Decision with different values for the age and accident history, and compare the results with the decision table:

**Insurance Total Price (Decision Table)**

U	Age (number)	had previous incidents (boolean)	Insurance Total Price (number)	Enter Text
1	>25	false	1000	
2	>25	true	1250	
3	[18..25]	false	2000	
4	[18..25]	true	3000	

---

Last update: 2023-12-06

## 3.9 Deploying PAM on OpenShift with Operators

---

### 3.9.1 26. Red Hat PAM Operator on OpenShift 4

In this lab we will use the enhanced Business Automation Operator **7.10+** to deploy a number of IBM Business Automation Open Edition 9.0 environments on OpenShift **4**.

#### 26.1 Goal

- Install the Business Automation Operator on OCP 4.
- Use the Business Automation Operator 8.0.4 to deploy a number of IBM Business Automation Open Edition 9.0 environments.
- Change the KIE-App deployment CRDs to show reconciliation.
- Change Operator ConfigMaps to make advanced configuration changes to the KIE-App.

#### 26.2 Problem Statement

In this lab, the goal is to provision and manage various IBM Business Automation Open Edition 9.0 architectures using the Business Automation Operator on OpenShift 4.

- We deploy an IBAMOE Trial environment, which is a basic ephemeral environment that does not require any form of storage (e.g. persistent volume, database).
- We explore Operator reconciliation features by removing provisioned resources like Services and Deployment Configs.
- We alter the deployment through the Operator to show how the provisioned environment changes.
- We change a KIE configuration parameter in the Business Automation Operator ConfigMap to demonstrate advanced configuration changes.
- We provision a more sophisticated Production environment, to show creation of PVCs, deployment of databases and integration with IBAMOE Smart Router.
- We use the Operator Installer console to install a new KIE-App deployment.

### 26.3 First steps

If you are using your own OpenShift environment, follow the steps below to create a project and install the operator. If you are trying this lab in an environment provisioned by the Red Hat team, skip to the section [Inspect the Lab environment](#).

1. Create a new project in OpenShift. We suggest the name `ibamoe-install`.

## Create Project

An OpenShift project is an alternative representation of a Kubernetes namespace.

[Learn more about working with projects](#)

**Name \*** [?](#)

**Display name**

**Description**

Used to deploy IBM Business Automation Manager Open Edition

[Cancel](#) [Create](#)

2. Navigate to `Operators`, `Operator Hub`, and search for `Business Automation`:

The screenshot shows the Red Hat OpenShift OperatorHub interface. The left sidebar is collapsed. The main header says "Red Hat OpenShift on IBM Techzone". The top right shows a notification count of 5 and a user "kube:admin". The navigation bar includes "Home", "Overview", "Projects", "Search", "API Explorer", "Events", "Operators" (selected), "Installed Operators", "Workloads", "Networking", "Storage", "Builds", "Observe", "Compute", "User Management", and "Administration". The "Operators" section has a sub-menu with "OperatorHub" (selected) and "Installed Operators". The "Business Automation" category is selected in the center panel. A card for "IBM Business Automation" is highlighted with a red border. The card details: "IBM Business Automation provided by IBM", "Deploys and manages IBM Business Automation Manager and Red Hat Decision Manager...", and "IBM". The bottom left of the panel shows a "Source" section with "Red Hat (2)", "Certified (0)", and "Community (0)". The top right of the panel shows "2 items".

3. Click on the Business Automation and then, click `Install`.

**IBM Business Automation**

8.0.3-4 provided by IBM

**Install**

---

**Latest version** Deploys and manages IBM Business Automation Manager Open Editions environment.

8.0.3-4

- **IBM Process Automation Manager** is a platform for developing containerized microservices and applications that automate business decisions and processes. It includes business process management (BPM), business rules management (BRM), and business resource optimization and complex event processing (CEP) technologies. It also includes a user experience platform to create engaging user interfaces for process and decision services with minimal coding.

**Capability level**

- Basic Install
- Seamless Upgrades
- Full Lifecycle
- Deep Insights
- Auto Pilot

**Source** Red Hat

**Provider** IBM

**Infrastructure features** Disconnected

**Valid Subscriptions** IBM Business Automation Manager Open Edition

**Repository**

<https://github.com/kiegroup/kie-cloud-operator>

---

**Container image**

[registry.redhat.io/ibm-pam](https://registry.redhat.io/ibm-pam)

4. You can select the following options, and click on `Submit`:

## Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

### Update channel \*

8.x-stable

### Installation mode \*

All namespaces on the cluster (default)

This mode is not supported by this Operator

A specific namespace on the cluster

Operator will be available in a single Namespace only.

### IBM Business Automation

provided by IBM

#### Provided APIs



KieApp

A project prescription running an IBM BAMOE environment.

### Installed Namespace \*

PR ibamoe-install

### Update approval \*

Automatic

Manual

Install

Cancel

5. Once subscribed, you should wait for the operator to get provisioned. Then you can proceed with the lab.

## Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the [Understanding Operators documentation](#). Or create an Operator and ClusterServiceVersion using the [Operator SDK](#).

Name	Managed Namespaces	Status	Last updated	Provided APIs
IBM Business Automation	ibamoe-install	Succeeded Up to date	Just now	KieApp

Last update: 2023-12-06

### 3.9.2 27. Deploying an IBAMOE Trial Environment

- From the **Business Automation** page in your OpenShift Console, open the **KieApp** tab and click on **Create KieApp**.

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, there's a sidebar with options like Home, Projects, Search, Explore, Events, Operators (with 'Installed Operators' selected), and Workloads. The main content area is titled 'Project: rhpm710-operator-lab-user1'. Under 'Operators', it shows 'Business Automation 7.10.1-1 provided by Red Hat'. The 'KieApp' tab is active. Below it, a section titled 'KieApps' shows a message 'No operands found' and a note: 'Operands are declarative components used to define the behavior of the application.' A blue button labeled 'Create KieApp' is visible.

- A form will be displayed for you to choose which installation option you want to have. notice the `environment` field. In this field we define the type of the environment we want to provision. In this case we want to provision the **Trial** environment, so we accept the default values.

The screenshot shows the 'Create KieApp' form. The sidebar on the left is identical to the previous screenshot. The main form has a header 'Project: rhpm710-operator-lab-user1' and 'Business Automation > Create KieApp'. It includes a note: 'Create by completing the form. Default values may be provided by the Operator authors.' and a choice between 'Configure via: Form view' (selected) and 'YAML view'. A tip message says: 'Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.' The form fields include 'Name\*' (set to 'rhpm-trial'), 'Labels' (set to 'app=frontend'), and 'Environment\*' (set to 'rhpm-trial'). To the right, there's a 'KieApp' icon with the text 'provided by Red Hat' and 'A project prescription running an RHPAM/RHDM environment.' A 'TIP:' label is located at the bottom right of the form area.

You also have the YAML definition option if you want to do customizations that are not available in the form above.

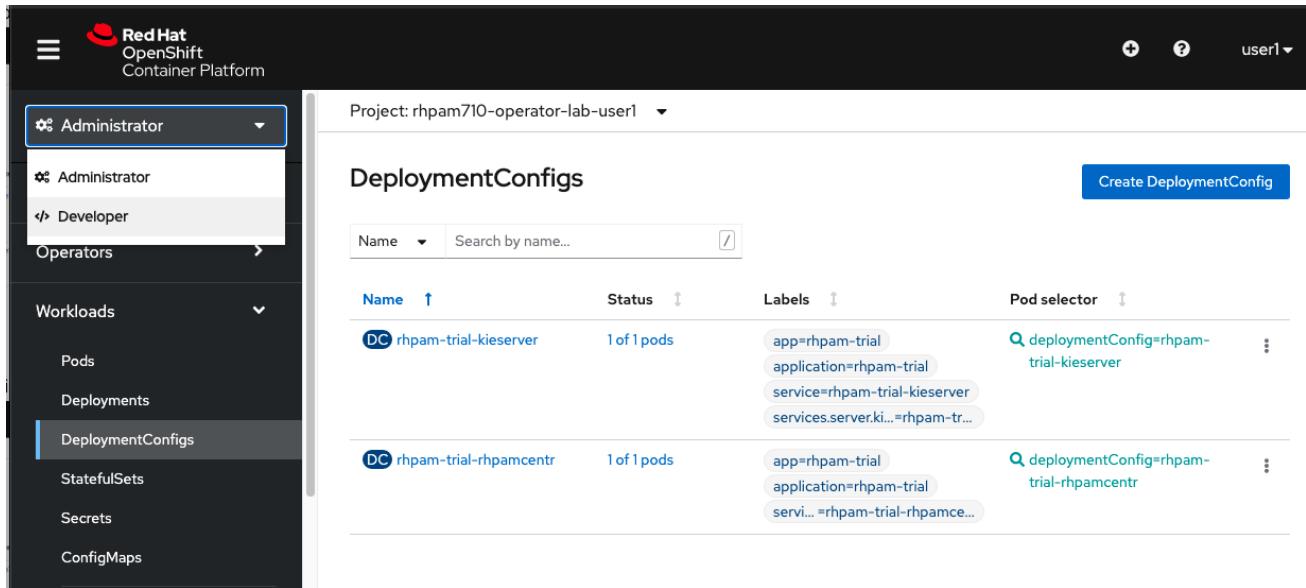
- Click on the **Create** button at the bottom of the page.
- In the **KieApp** tab, we can see our new **rhpm-trial** environment being listed.

The screenshot shows the Red Hat OpenShift Container Platform web interface. On the left, there is a sidebar with navigation links: Home, Projects, Search, Explore, Events, Operators (with 'Installed Operators' selected), and Workloads (with 'KieApps' selected). The main content area is titled 'Project: rhpam710-operator-lab-user1'. It shows the 'Business Automation' operator details, specifically the 'KieApp' tab. A table lists a single KieApp named 'rhpam-trial' with status 'Phase: Deployed' and no labels. There is a 'Create KieApp' button at the top right of the table.

5. Expand the **Workloads** menu on the left side of the screen. Click on **Deployment Configs**. Observe that the Operator has created 2 Deployment Configs, one for Business Central and one for KIE-Server.

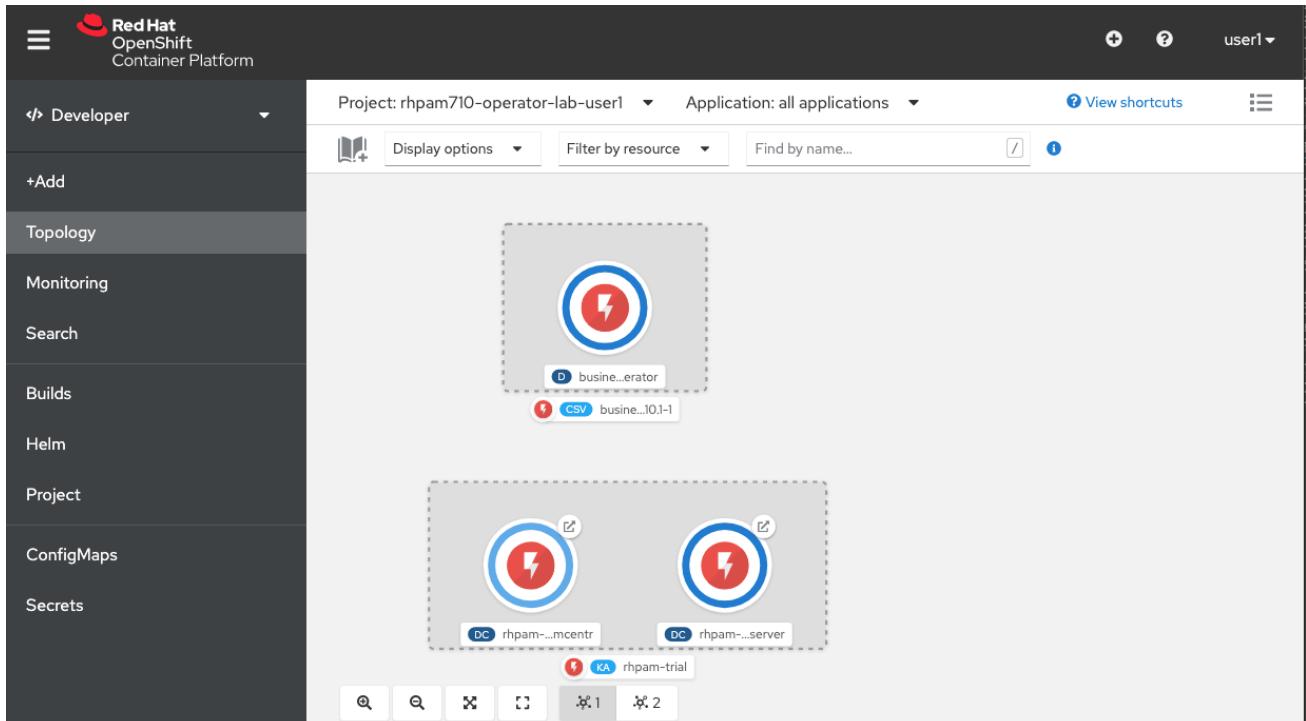
The screenshot shows the Red Hat OpenShift Container Platform web interface. The sidebar shows 'Workloads' expanded, with 'DeploymentConfigs' selected. The main content area is titled 'Project: rhpam710-operator-lab-user1'. It shows the 'DeploymentConfigs' table with two entries: 'rhpam-trial-kieserver' and 'rhpam-trial-rhpamcentr'. Both entries show '1 of 1 pods' status. The 'Pod selector' column shows labels: 'app=rhpam-trial', 'application=rhpam-trial', 'service=rhpam-trial-kieserver', 'services.server.ki...=rhpam-tr...', and 'deploymentConfig=rhpam-trial-kieserver' for the first entry, and 'deploymentConfig=rhpam-trial-rhpamcentr' for the second entry.

6. Open the **Developer Console** by clicking on the link in the dropdown box at the top left of the screen.



Name	Status	Labels	Pod selector
DC rhpam-trial-kieserver	1 of 1 pods	app=rhpam-trial application=rhpam-trial service=rhpam-trial-kieserver services.server.kie...=rhpam-tr...	Q deploymentConfig=rhpam-trial-kieserver
DC rhpam-trial-rhpamcentr	1 of 1 pods	app=rhpam-trial application=rhpam-trial servi... =rhpam-trial-rhpamce...	Q deploymentConfig=rhpam-trial-rhpamcentr

7. Click on the **Topology** link to show a graphical representation of the topology of our namespace, which includes an Operator DC, a Business Central DC, and a KIE-Server DC.



8. Go back to the **Administrator** console.

9. Open **Networking** → **Routes** menu to see all the available routes to our KIE application deployed in this namespace.

Name	Status	Location	Service
console-cr-form	Accepted	https://console-cr-form-rhampm710-operator-lab-user1.apps.cluster-486f486f.sandbox1697.opentlc.com	console-cr-form
rhampm-trial-kieserver	Accepted	https://rhampm-trial-kieserver-rhampm710-operator-lab-user1.apps.cluster-486f486f.sandbox1697.opentlc.com	rhampm-trial-kieserver
rhampm-trial-kieserver-http	Accepted	http://rhampm-trial-kieserver-http-rhampm710-operator-lab-user1.apps.cluster-486f486f.sandbox1697.opentlc.com	rhampm-trial-kieserver
rhampm-trial-rhampcentr	Accepted	https://rhampm-trial-rhampcentr-rhampm710-operator-lab-user1.apps.cluster-486f486f.sandbox1697.opentlc.com	rhampm-trial-rhampcentr
rhampm-trial-rhampcentr-http	Accepted	http://rhampm-trial-rhampcentr-http-rhampm710-operator-lab-user1.apps.cluster-486f486f.sandbox1697.opentlc.com	rhampm-trial-rhampcentr

10. Identify the **Business/Decision Central URL** link to navigate to the IBAMOE Business Central workbench. It should be named `rhampm-trial-rhampcentr-http` for the http option, or `rhampm-trial-rhampcentr` for https.
11. As the Operator is responsible for deployment and configuration of the IBAMOE environment, we can find the details if this deployment in the **KieApp** instance details screen. Open your KieApp in **Operators → Installed Operators → Business Automation → KieApp → rhampm-trial**, and click on the **YAML** tab. CHANGE THIS LINK
12. We can see in the YAML description that the **adminPassword** has been set to `RedHat`. Navigate back to the **Business Central** workbench and login with u: `adminUser` p: `RedHat`.
13. Explore the **Business Central** application. In particular, go to **Menu → Deploy → Execution Servers** to see the **Execution Server** connected to the workbench.

Capabilities:	Deployment Units	Remote Servers
<input checked="" type="checkbox"/> Decision <input checked="" type="checkbox"/> Process <input checked="" type="checkbox"/> Planner	+ Add Deployment Unit	<b>No Associated Containers</b>  This Remote Server does not have any associated Deployment Units. You can create deployment units on this server by adding them to corresponding server configuration.

Last update: 2023-12-06

### 3.9.3 28. Version Upgrades

The Operator of IBAMOE is also capable of doing both patch and minor upgrades. This means that, for example, the Operation can upgrade an IBAMOE environment from 7.10.0 to 7.10.1 or from 7.10.1 to 7.11.0.

- When creating a new KieApp, you can find the option to enable the version updates.

Name	Status
rhpam-trial	Deployed

**Annotations:** 1 Annotation

**Created At:** 8 minutes ago

**Business/Decision Central URL:** [rhpam-trial-rhpamcenter-rhpam75-operator-lab-user1.apps-crc.testing](#)

- If both the **Enable Upgrades** and **Include minor version upgrades** settings are set to true, the KieApp YAML configuration will include the following spec:

```

1 apiVersion: app.kiegroup.org/v2
2 kind: KieApp
3 metadata:
4   creationTimestamp: '2021-04-15T21:52:58Z'
5   generation: 1
6   managedFields:-
7     name: rhpam-trial
8     namespace: rhpam710-operator-lab-user1
9     resourceVersion: '308712'
10    selfLink: >
11      /apis/app.kiegroup.org/v2/namespaces/rhpam710-operator-lab-user1/kieapps/rhpam-trial
12    uid: d7f19a54-00fd-4b42-b20f-bc28f843cf7
13  spec:
14    environment: rhpam-trial
15    upgrades:
16      enabled: true
17      minor: true

```

With this configuration the version upgrade mechanism of the Operator should be enabled for the given KieApp.

### 3.9.4 29. Reconciliation

The OpenShift Operators provide functionality to reconcile an existing environment in order to bring it back to its expected state. We will now test this feature by removing one of the required resources from our deployment.

1. Open the **Resources** tab. This will show all the resources of the application deployed and managed by the Operator.

Name	Kind	Status	Created
rhpam-trial-businesscentral-app-secret	Secret	Created	Apr 15, 6:52 pm
rhpam-trial-kieserver	DeploymentConfig	Created	Apr 15, 6:52 pm
rhpam-trial-kieserver	Route	Created	Apr 15, 6:52 pm
rhpam-trial-kieserver	Service	Created	Apr 15, 6:52 pm
rhpam-trial-kieserver-app-secret	Secret	Created	Apr 15, 6:52 pm

2. On the fourth row, we can see the `rhpam-trial-kieserver` **Service** resource.

3. In the left menu, go to **Networking → Services**. Open `rhpam-trial-kieserver`.

4. Delete the **Service** by clicking on the **Actions** button at the upper right of the screen and clicking on **Delete**.

5. Notice the **Service** disappearing and immediately reappearing. This is the Operators reconciliation logic at work, bringing the environment back in its expected state.

### 3.9.5 30. KIE App Configuration

The definition of the expected state of KIE-App environment is defined in the YAML definition the KIE-App. In this section we will slightly change this configuration to see how the Operator applies changes in the configuration of your IBM Business Automation Open Edition 9.0 environment.

#### 30.1 Changing Credentials

1. Go back to the YAML definition of your `rhpam-trial` KieApp.
2. Add a `commonConfig` section, with the `adminUser` to the value `bamAdmin`, and the `adminPassword` to `ibmpam1!`. Click on the **Save** button.

```

1 apiVersion: app.kiegroup.org/v2
2 kind: KieApp
3 metadata:
4   creationTimestamp: '2021-04-15T21:52:58Z'
5   generation: 1
6   managedFields: ...
66  name: rhpam-trial
67  namespace: rhpam710-operator-lab-user1
68  resourceVersion: '317761'
69  selfLink: >-
70    /apis/app.kiegroup.org/v2/namespaces/rhpam710-operator-lab-user1/kieapps/rhpam-trial
71  uid: d7f19a54-00fd-4b42-b20f-bc28f843cf7
72 spec:
73   environment: rhpam-trial
74   commonConfig:
75     adminPassword: pamAdmin
76     adminUser: redhatpam1!
77 status:
78   applied:
79     commonConfig:
80       adminPassword: Redhat
81       adminUser: adminUser
82       amqClusterPassword: RedHat
83       amqPassword: Redhat
84       applicationName: rhpam-trial
85       dbPassword: Redhat
86
87
88
89
90
91
92
93
94
95

```

Specified values:

```

spec:
  commonConfig:
    adminPassword: bamAdmin
    adminUser: ibmpam1!

```

3. Click the **Reload** button to reload the YAML view.
4. Click on the **Overview** tab. Notice the deployments re-deploying.
5. Click on the **Business/Central Central URL** to open the Business Central console.
6. Log in with the new username and password: `bamAdmin / ibmpam1!`.

### 30.2 Adding a KIE-Server

Apart from changing some configuration parameters, we can also change the topology our deployment in the KieApp YAML file.

1. Go back to the YAML definition of your `rhpam-trial` KieApp.

2. Add a `servers` section and set the `replicas` parameter of the `rhpam-trial-kieserver` to **2**.

```

apiVersion: app.kiegroup.org/v2
kind: KieApp
metadata:
  creationTimestamp: '2021-04-15T21:52:58Z'
  generation: 1
  managedFields:-
    name: rhpam-trial
    namespace: rhpam710-operator-lab-user1
    resourceVersion: '317761'
    selfLink: >-
      /apis/app.kiegroup.org/v2/namespaces/rhpam710-operator-lab-user1/kieapps/rhpam-trial
    uid: d7f19a54-00fd-4b42-b20f-bc28f843cf7
spec:
  environment: rhpam-trial
  status:
    applied:
      commonConfig:
        adminPassword: pamAdmin
        adminUser: redhatpam1
        amqClusterPassword: RedHat
objects:
  servers:
    - deployments: 1
      name: rhpam-trial-kieserver
      replicas: 2

```

3. Click the **Save** button.

4. Go to **Workloads → Deployment Configs**. Note that there are now 2 KIE-Server Deployment Configs.

```

apiVersion: app.kiegroup.org/v2
kind: KieApp
metadata:
  creationTimestamp: '2021-04-15T21:52:58Z'
  generation: 1
  managedFields:-
    name: rhpam-trial
    namespace: rhpam710-operator-lab-user1
    resourceVersion: '323956'
    selfLink: >-
      /apis/app.kiegroup.org/v2/namespaces/rhpam710-operator-lab-user1/kieapps/rhpam-trial
    uid: d7f19a54-00fd-4b42-b20f-bc28f843cf7
spec:
  commonConfig:
    adminPassword: pamAdmin
    adminUser: redhatpam1
    environment: rhpam-trial
  objects:
    servers:
      - deployments: 1
        name: rhpam-trial-kieserver
        replicas: 2
status:
  applied:

```

5. Go back to the YAML definition of your `rhpam-trial` KieApp.

6. Navigate to the `servers` section and add the property `deployments` with the value **2**.

```

objects:
  servers:
    - deployments: 2
      name: rhpam-trial-kieserver
      replicas: 2

```

7. Click the **Save** button.

Last update: 2023-12-06

### 3.9.6 31. ConfigMaps

The Operator stores its configuration in a number of *ConfigurationMaps*. These ConfigurationMaps can be used to change more advanced configurations that can not be configured in the KieApp YAML. In the case the Operator upgrades the version of your IBAMOE environment, the Operator is aware that one of the ConfigMaps has changed and will make a backup of it during the upgrade.

#### 31.1 Viewing and editing ConfigMaps

A powerful feature of OpenShift are ConfigMaps which provide mechanisms to inject containers with configuration data while keeping containers agnostic of OpenShift Container Platform. A ConfigMap can be used to store fine-grained information like individual properties or coarse-grained information like entire configuration files or JSON blobs. In this section, we will modify some of the default health properties that are different than the defaults provided with IBAMOE and we want them to roll out to every container that gets created with the operator.

1. In the OpenShift Console, open **Workloads → Config Maps**.

Name	Namespace	Size	Created
kie-cloud-operator-lock	rhpam75-operator-lab-user1	0	14 hours ago
kieconfigs-7.4.0	rhpam75-operator-lab-user1	1	14 hours ago
kieconfigs-7.4.0-dbs	rhpam75-operator-lab-user1	4	14 hours ago
kieconfigs-7.4.0-envs	rhpam75-operator-lab-user1	9	14 hours ago
kieconfigs-7.4.0-jms	rhpam75-operator-lab-user1	1	14 hours ago
kieconfigs-7.4.1	rhpam75-operator-lab-user1	1	14 hours ago
kieconfigs-7.4.1-dbs	rhpam75-operator-lab-user1	4	14 hours ago
kieconfigs-7.4.1-envs	rhpam75-operator-lab-user1	9	14 hours ago
kieconfigs-7.4.1-jms	rhpam75-operator-lab-user1	1	14 hours ago
kieconfigs-7.5.0	rhpam75-operator-lab-user1	1	14 hours ago
kieconfigs-7.5.0-dbs	rhpam75-operator-lab-user1	4	14 hours ago
kieconfigs-7.5.0-envs	rhpam75-operator-lab-user1	9	14 hours ago

2. Note that the Operator keeps the current ConfigMaps, and the ones of the last 2 versions.
3. Click on the `kieconfigs-7.10.1` ConfigMap and open the YAML tab.
4. Explore the configuration options.
5. Set the `initialDelaySeconds` of the `livenessProbe` of the Business Central console from 180 to 240.
6. Click the **Save** button to save the configuration.
7. Go to "Workloads → Deployment Configs", open the `rhpam-trial-rhpamcentr` Deployment Config and open the YAML tab.
8. Find the LivenessProbe `initialDelaySeconds` configuration and notice that it's still set to 180.
9. Delete the DeploymentConfig. This will have the Operator reconciliation recreate the DC.
10. Open the YAML configuration of this recreated DeploymentConfig.
11. Find the LivenessProbe `initialDelaySeconds` configuration and note that this time it has been set to 240, the value set in the ConfigMap.

### 31.2 Deleting an application

Apart from provisioning an IBAMOE application, the Operator also allows us to easily delete an application.

1. Navigate to **Operators → Installed Operators → Business Automation → KieApp**.

2. Click on the kebab icon of the `rhpam-trial` KieApp and click **Delete**.

Name	Labels	Kind	Status	Version	Last Updated
<code>rhpam-trial</code>	No labels	KieApp	Deployed	7.5.0	Nov 13, 10:49 am

3. Navigate back to **Workloads → Deployment Configs** and note that the IBAMOE Deployment Configs have been removed.

Last update: 2023-12-06

### 3.9.7 32. Operator Wizard Walkthrough

The Business Automation Operator contains an *Operator Installer Console*. This console gives you a *wizard* experience to deploy IBM Business Automation Open Edition 9.0 environments.

#### 32.1 Wizard walk through

##### 1. Go the Business Automation Operator and click in **Installer** link.

Project: ibamoe-install ▾

Installed Operators > Operator details

**IBM Business Automation**  
8.0.3-4 provided by IBM

Actions ▾

**Details** YAML Subscription Events KieApp

**Provided APIs**

**KieApp**  
A project prescription running an IBM BAMOE environment.  
[Create instance](#)

**Provider**  
IBM

**Created at**  
1 minute ago

**Links**  
Installer  
<https://console-cr-form-ibamoe-install.apps.652e56ef1ebcb00178b28f2.cloud.techzone.ibm.com>

**Product Page**  
<https://ibm.com>

**Documentation**  
<https://www.ibm.com/docs/en/ibamoe>

**Maintainers**  
IBM  
[mysphelp@us.ibm.com](mailto:mysphelp@us.ibm.com)

Description

Deploys and manages IBM Business Automation Manager Open Editions environment.

- IBM Process Automation Manager is a platform for developing containerized microservices and applications that automate business decisions and processes. It includes business process management (BPM), business rules management (BRM), and business resource optimization and complex event processing (CEP) technologies. It also includes a user experience platform to create engaging user interfaces for process and decision services with minimal coding.

To use the guided installer to provision an environment, open the **Installer** link, in the **Links** section on the left side of this page.

#### ClusterServiceVersion details

**Operator installer**

RHPAM installer

**1 Installation**

**Installation**

Application name \*

Environment \*

rhpam-trial

The name of the environment used as a baseline

Enable Upgrades

Set true to enable automatic micro version product upgrades.

Use Image Tags

Set true to enable image tags, disabled by default. This will leverage image tags instead of the image digests.

Disable SSL Routes

If checked, external routes to the services will use plain text communication (http).

**Truststore**

**Next** **Back** **View YAML** **Finish**

##### 2. Login with Openshift. A page will show up asking for authorization. Select all options and click on "Allow selected permissions".

3. Give the application the name `my-ibamoe-authoring`.
4. Select the `rhpam-authoring` for the Environment.
5. Check the **Enable Upgrades** checkbox.
6. Scroll down and set the Username and Password to `bamAdmin : ibmpam1`.
7. Click the **Next** button.
8. Don't change any values in the **Security** section. Click on **Next**.
9. Go through the **Components** section of the installer and observe the possible options. Don't change any values for now.
10. Keep clicking next until you reach the **Confirmation** screen and click **Deploy**.
11. Go back to the OpenShift Console.
12. Navigate to **Workloads → Deployment Configs** and observe that a new IBM Business Automation Open Edition 9.0 production environment has been deployed. Note that this environment has a PostgreSQL database deployed. Also note that both the Business Central and KIE-Server Deployment Configs have their ReplicationController set to 3 pods.

Name	Status	Labels	Pod selector
DC my-rhpam-prod-kieserver	3 of 3 pods	app=my-rhpam-prod application=my-rhpam-prod service=my-rhpam-prod-kieserver services.serverkie.org/kie...=my-rhpam-prod-...	deploymentConfig=my-rhpam-prod-kieserver
DC my-rhpam-prod-kieserver-postgresql	1 of 1 pods	app=my-rhpam-prod application=my-rhpam-prod service=my-rhpam-prod-kieserver-postgresql	deploymentConfig=my-rhpam-prod-kieserver-postgresql
DC my-rhpam-prod-rhpamcentrmon	3 of 3 pods	app=my-rhpam-prod application=my-rhpam-prod service=my-rhpam-prod-rhpamcentrmon	deploymentConfig=my-rhpam-prod-rhpamcentrmon
DC rhpam-trial-kieserver	2 of 2 pods	app=rhpam-trial application=rhpam-trial service=rhpam-trial-kieserver services.serverkie.org/kie...=rhpam-trial-kie...	deploymentConfig=rhpam-trial-kieserver
DC rhpam-trial-kieserver-2	2 of 2 pods	app=rhpam-trial application=rhpam-trial service=rhpam-trial-kieserver-2 services.serverkie.org/kie...=rhpam-trial-kies...	deploymentConfig=rhpam-trial-kieserver-2
DC rhpam-trial-rhpamcentr	1 of 1 pods	app=rhpam-trial application=rhpam-trial service=rhpam-trial-rhpamcentr	deploymentConfig=rhpam-trial-rhpamcentr

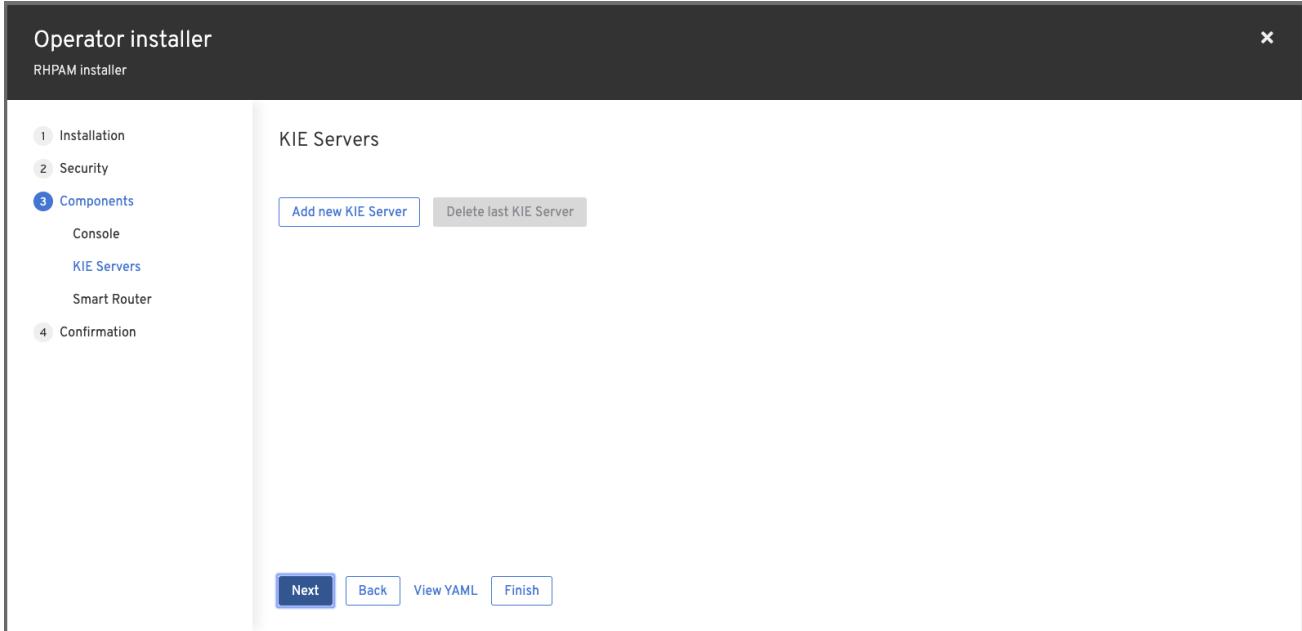
13. Go back to the **Operators → Installed Operators → Business Automation → KieApp**.

14. Delete the `my-rhpam-authoring` we've just deployed with the Installer.

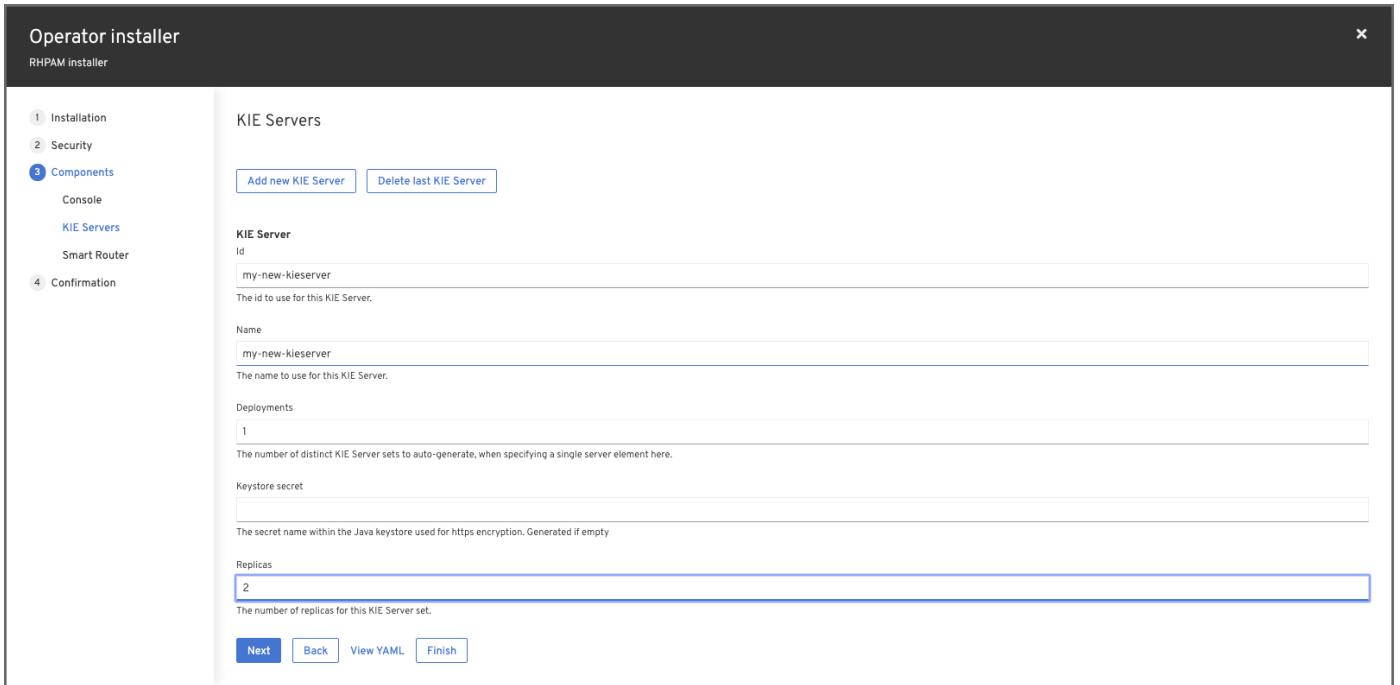
### 32.2 How to KIE Server with more replicas

We will now deploy a new production environment using the installer, but this time we will configure our KIE-Server in the wizard and set the replications of the KIE-Server to 2 instead of 3.

1. Go back to the Business Automation Operator, and open the Wizard.
2. Create a new IBAMOE Production Environment. Continue until you reach the **KIE Servers** screen.



3. Click **Add new KIE Server** and use the following configuration for your KIE-Server.



4. Click through the rest of the screens until you can press the **Deploy** button to deploy the environment.
5. Navigate to the **Workloads → Deployment Configs** screen to see your IBAMOE production environment, including the KIE-Server you configured.

### 32.3 Conclusion

This concludes the lab on the Business Automation Operator. If you have time left, feel free to explore more features of the operator.

---

Last update: 2023-12-06

## 3.10 Tools

---

### 3.10.1 33. Introduction

This is a series of guided exercises that will allow you to experiment the authoring tools in VSCode and deployment in IBAMOE engine, KIE Server.

#### 33.1 Tooling Set

In Red Hat PAM and DM you can author decisions using:

- Business Central (in IBAMOE) or Decision Central (in RHDM)
- A more business friendly UI;
- [Business Automation VSCode Extension](#)
- A developer IDE ([Visual Studio Code](#)) extension that allows the visualization and editing of BPMN, DMN and Test Scenarios inside VSCode.

There is also a set of community tooling that's also available for use. All the tools below are backed by IBM and Red Hat:

- [DMN FEEL Handbook](#)
- A handbook for the FEEL expression language from the DMN specification, as implemented by the Drools DMN open source engine.
- [Learn DMN in 15 minutes](#)
- A guided tour in a website through the elements of DMN
- [GitHub Chrome Extension](#)
- A browser extension that allows you to visualize and edit BPMN, DMN and Test Scenario files directly in GitHub.
- [Online Editors](#)
- [BPMN.new](#) - A free online editor for business processes;
- [DMN.new](#) - A free online editor for decision models;
- [PMML.new](#) - A free online editor for scorecards;
- [Business Modeler Hub](#)
- Allows for the download of the: VSCode extension, GitHub Chrome Extension, and Desktop App

---

Last update: 2023-12-06

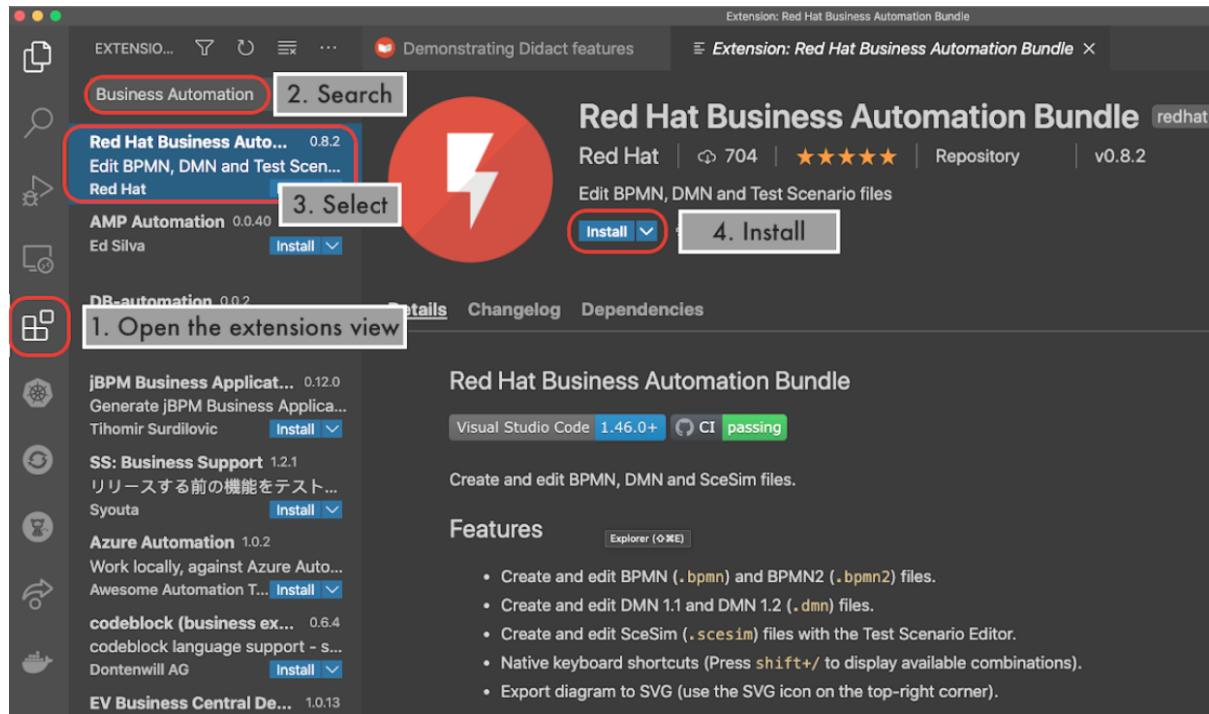
### 3.10.2 34. Business Automation projects in VSCode

To start working with business automation projects in VSCode, you'll need to install the VSCode Extension that allows you to work with BPMN, DMN and Test Scenarios through graphical editors.

#### 34.1 Installing the VSCode extension

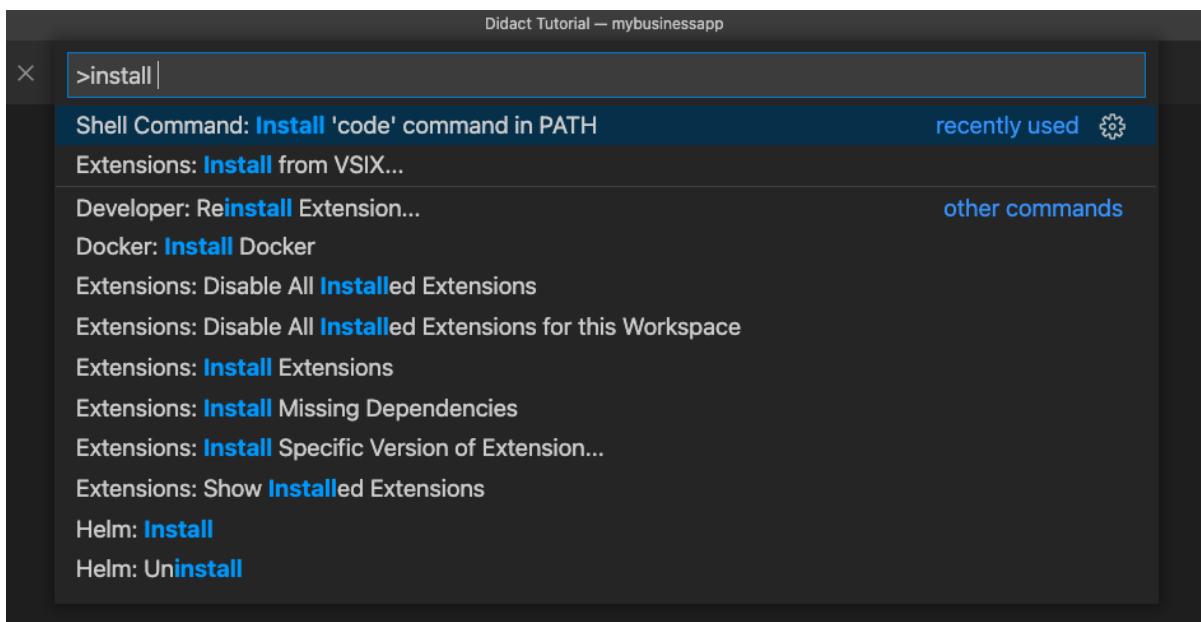
To install the extension in VSCode, open the extensions menu, and search for Business Automation.

You should find the Red Hat Business Automation Bundle.



Click on install.

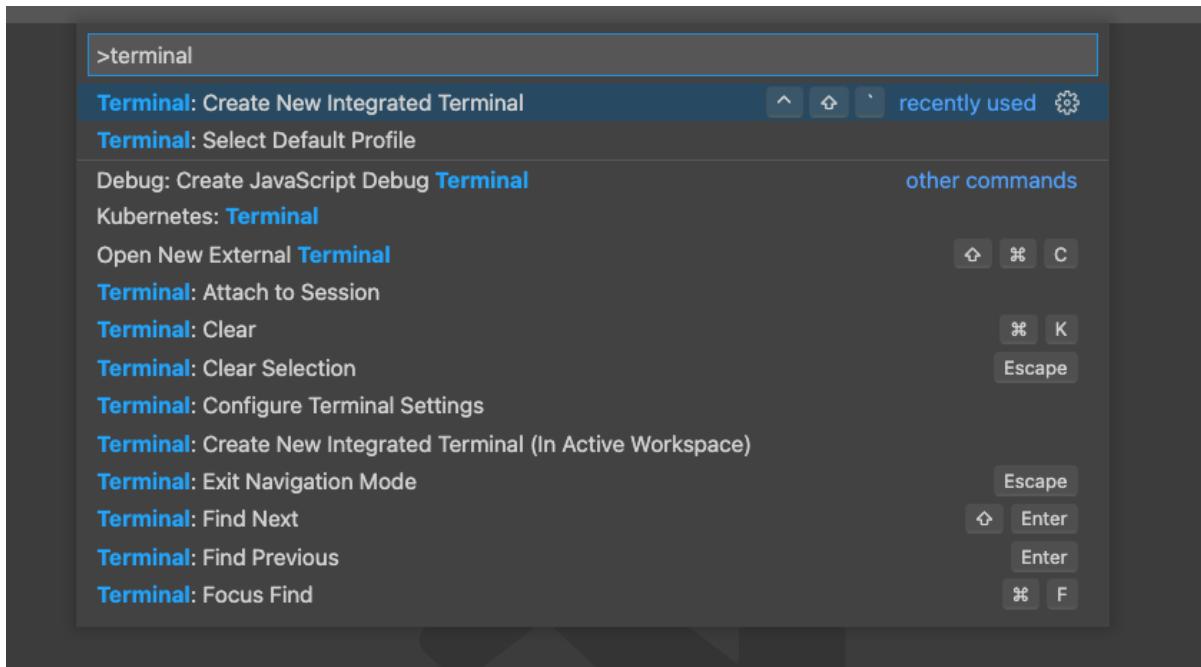
If this is the first time you are using VSCode, it would be interesting to also install the `code` command in path, so that you can open projects directly from the terminal. To do so, press `cmd+shift+p` (or `ctrl+shift+p`) to launch `vscode Quick Open` menu. And next, search for **Instal code command in PATH:**



### 34.2 Create new a project

Let's create a new project using the maven archetype. This project should contain the structure and files that Business Central expects, so this project should be editable and authored in both VScode and Business Central.

1. Now we will use the terminal. You can either use your terminal or use the built-in terminal In VScode. To use the terminal in VSCode you can press `cmd+shift+p` (or `ctrl+shift+p`) to launch `vsCode Quick Open` menu. And next, open a `new intergrated terminal`:



1. Next, in the terminal navigate to the directory where you would like to create the new project. Let's call it `$PROJECT_DIR` from now on. Create a new folder named `tooling-labs`.

```
$ cd $PROJECT_DIR
$ mkdir tooling-labs
$ cd tooling-labs
```

- Now, use the maven archetype to create a new project in the `tooling-labs` directory:

```
mvn archetype:generate \
-DarchetypeGroupId=org.kie \
-DarchetypeArtifactId=kie-kjar-archetype \
-DarchetypeVersion=7.67.2.Final-redhat-00006
```

```
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
○ → cd tooling-labs/
2021-04-11 11:14:35 🕒 Karinas-MBP in ~/projetos/enablement/tooling-labs
○ → mvn archetype:generate \
>   -DarchetypeGroupId=org.kie \
>   -DarchetypeArtifactId=kie-kjar-archetype \
>   -DarchetypeVersion=7.48.0.Final-redhat-00004
```

**TIP:** If you need to create a case project, you can use the parameter `-DcaseProject=true`.

- Maven will download the libraries, and once it finishes, it will confirm if you want to create the project using the default GAV (group:artifact:version). Type "Y" and press enter.

```
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
[INFO] Using property: groupId = org.kie.businessapp
[INFO] Using property: artifactId = mybusinessapp
[INFO] Using property: version = 1.0
[INFO] Using property: package = org.kie.businessapp
[INFO] Using property: archetypeCatalog = local
[INFO] Using property: caseProject = false
[INFO] Using property: kieVersion = 7.48.0.Final-redhat-00004
Confirm properties configuration:
groupId: org.kie.businessapp
artifactId: mybusinessapp
version: 1.0
package: org.kie.businessapp
archetypeCatalog: local
caseProject: false
kieVersion: 7.48.0.Final-redhat-00004
Y: : █
```

- You should get a new project named `mybusinessapp`. If you are in VSCode built-in terminal, you can open the project with:

```
$ code -r mybusinessapp/
```

- In VSCode, navigate through the project structure and confirm that it has a `kie-deployment-descriptor.xml` and a `kmodule.xml`. These are the files that Business Central needs to understand that this is a business project that should be packaged in a kjar. These files are also needed by KIE Server.

### 34.3 Next Steps

Now, let's author a DMN file, test it and deploy it to KIE Server.

Last update: 2023-12-06

### 3.10.3 35. Authoring a Decision

---

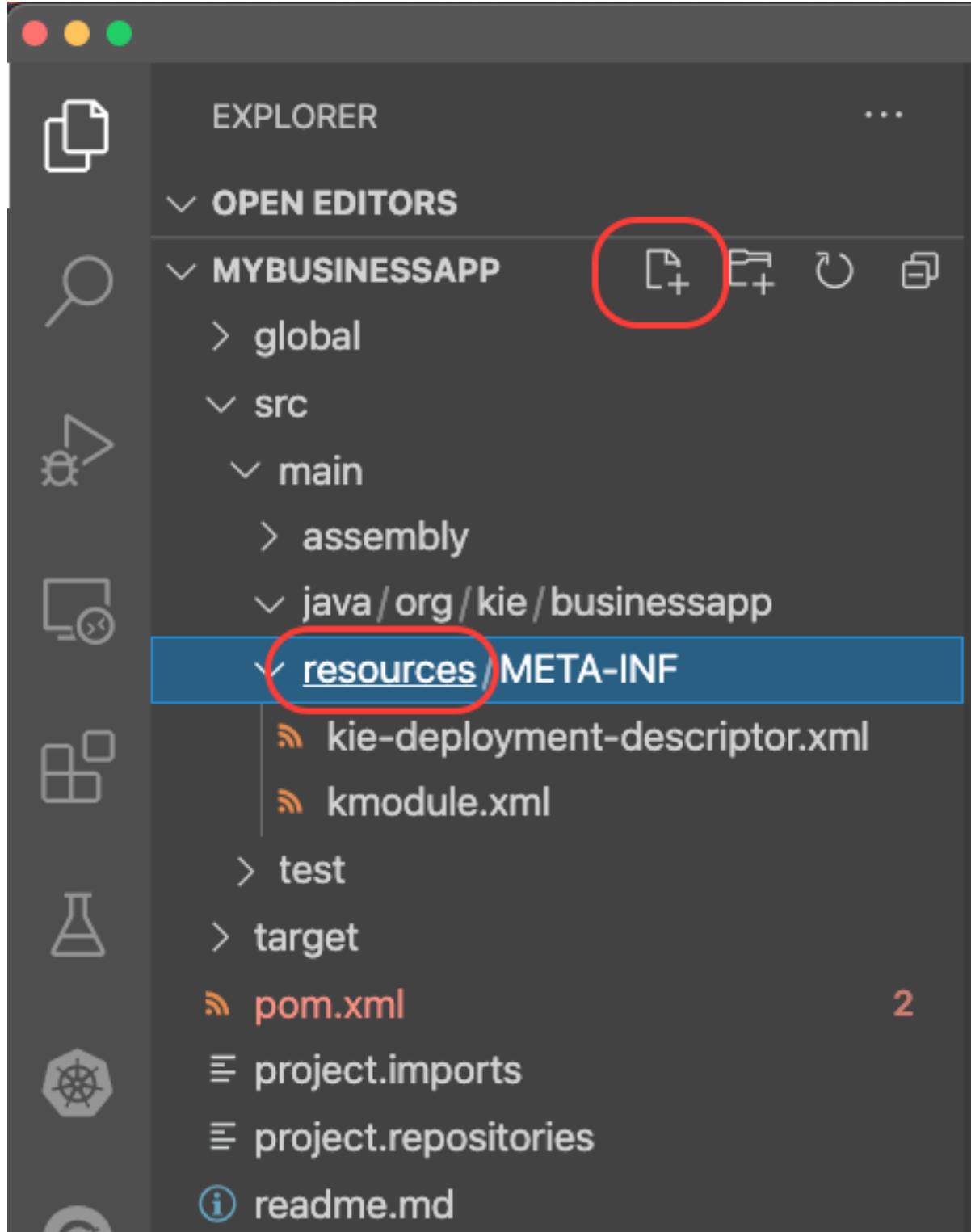
Let's author a simple decision and test it.

The use case we'll try out is the automation of a repeated decision for requests approval.

### 35.1 Create a new Decision

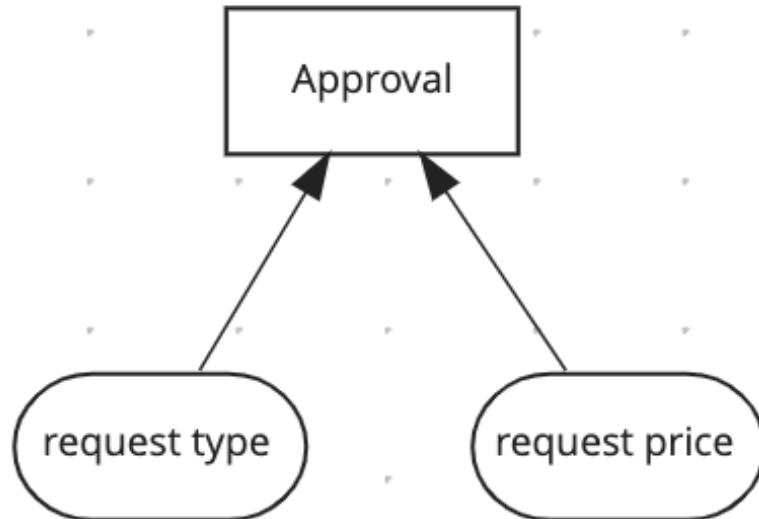
In the project we've just created:

1. Select the folder where you want to create the new file. Click on `resources`. Next, click on the *new folder* icon:



2. Name the file `automated-request-approval.dmn` and press enter. The file should open in the DMN Editor.

3. Create the following DMN:



4. This DRD contains:

- A decision node `Approval` of type `boolean`;
- Two inputs:
- A `request type`, which is `string`
- A `request price`, that is a `number`.

5. Implement the following decision table, in the decision node:

### Approval (Decision Table)

U	request price (number)	request type (string)	Approval (boolean)	annotation-1
1	<=150	"low", "medium", "urgent"	true	
2	(150..400]	"medium", "urgent"	true	
3	(150..400]	"low"	false	
4	(400..1500]	"urgent"	true	
5	(400..1500]	"medium", "low"	false	
6	>1500	-	false	

6. Save the diagram

## 35.2 Testing the decision

Before deploying the decision service in KIE Server, let's do some unit testing using the Test Scenario Simulation tooling.

### 35.2.1 CONFIGURING THE PROJECT

In order to use test scenarios, you need to add at least three dependencies to your project:

- junit:junit
- org.drools:drools-scenario-simulation-backend
- org.drools:drools-scenario-simulation-api

Open the pom.xml file and add the following dependencies:

```

<dependencies>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-scenario-simulation-api</artifactId>
    <version>${version.org.kie}</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-scenario-simulation-backend</artifactId>
    <version>${version.org.kie}</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>
</dependencies>

```

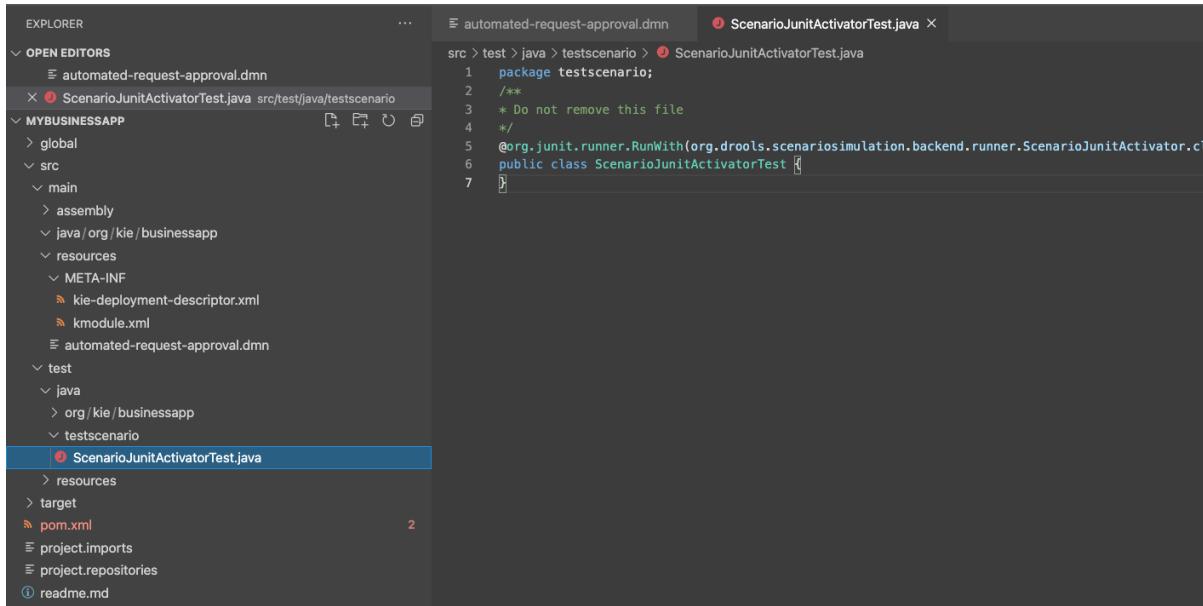
### 35.2.2 ADDING THE JUNIT RUNNER

In this scenario, the `version.org.kie` should be compatible with the product version you want to use. In this scenario, we are using IBAMOE 7.10, which would be `<version.org.kie>7.67.2.Final-redhat-00006</version.org.kie>`.

1. Create a new folder `testscenario`: `/src/test/java/testscenario`
2. In the folder you just created, add a file and name it `ScenarioJUnitActivatorTest.java`
3. In this class, you should add a the Scenario Activator. This class allows the test scenarios to run along with the junit tests.

```
package testscenario;
/**
 * Do not remove this file
 */
@org.junit.runner.RunWith(org.drools.scenariosimulation.backend.runner.ScenarioJUnitActivator.class)
public class ScenarioJUnitActivatorTest {
```

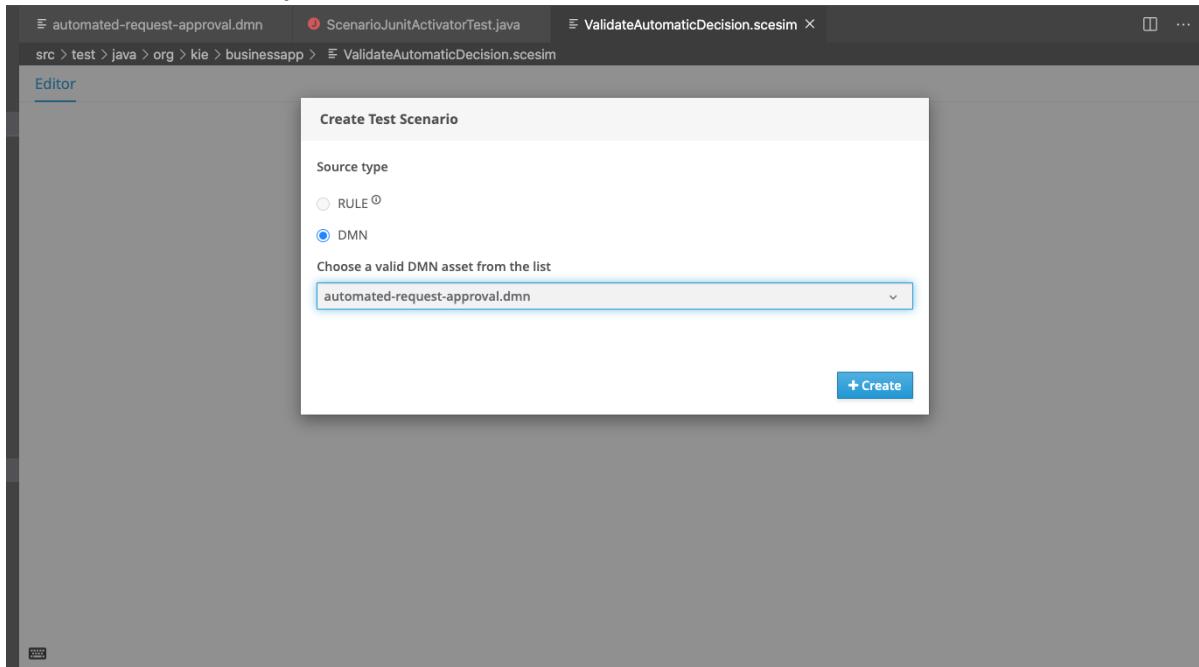
It should look like this:



### 35.2.3 CREATING THE TEST SCENARIO

1. On the folder `src/test/resources/org/kie/businessapp` create a new file named `ValidateAutomaticDecision.scesim`.
2. The editor should open up with the option to choose the **Source type**. This is the type of rule you want to test.

3. Select DMN, next, choose your DMN file and click on the **create** button.



4. The tool will already bring the inputs and expected result columns based on your DMN. Now, implement the following test:

#	Scenario description	GIVEN		EXPECT
		request price	request type	
1	Should approve requests under 150	150	"urgent"	true
2	Should not approve low priority requests higher than 150	160	"low"	false
3	Should approve urgent requests until 1500	1500	"urgent"	true
4	Should not automatically approve any request higher than 1500	1501	"urgent"	false

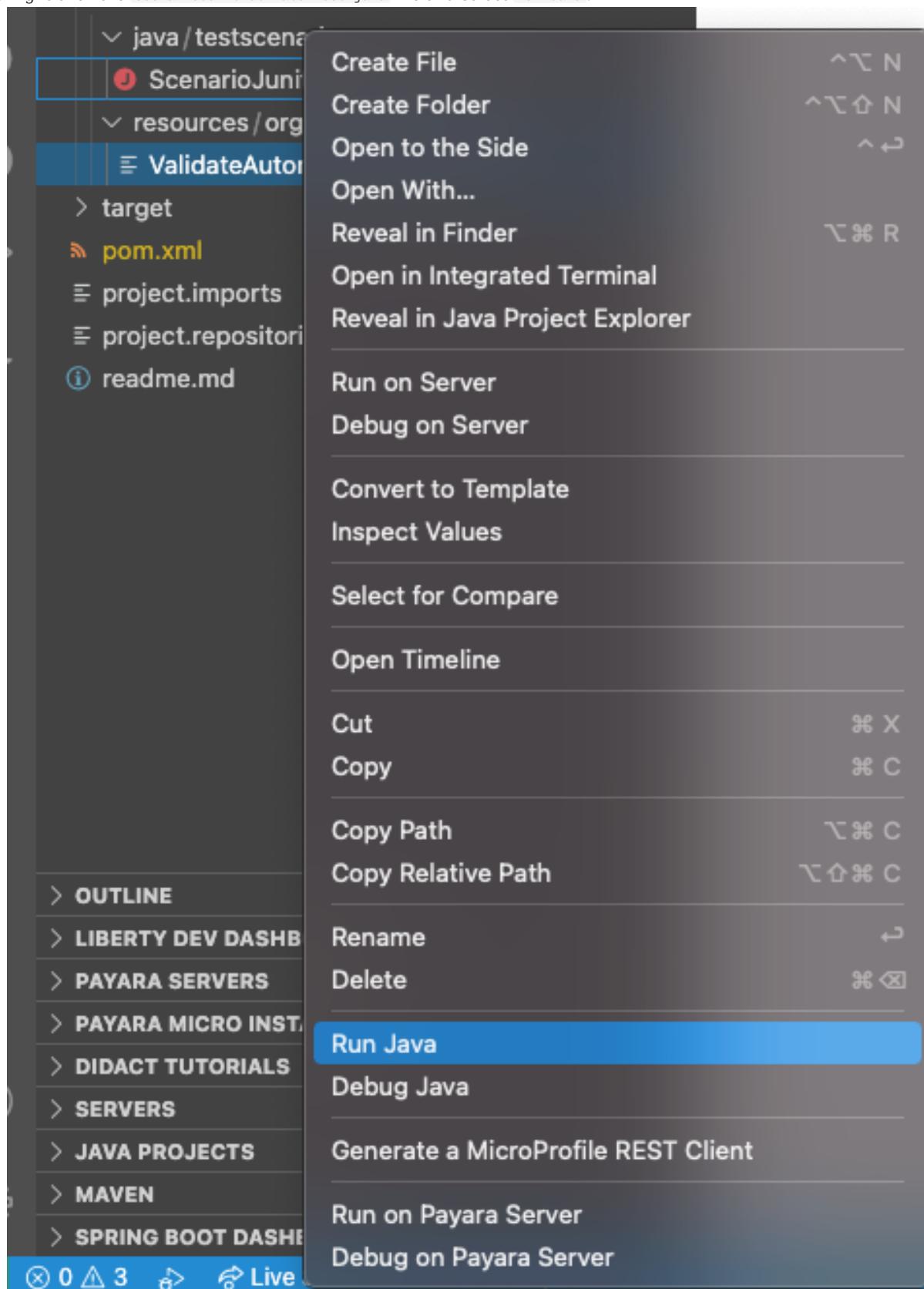
#### 35.2.4 RUNING THE TESTS

You can run the tests in two ways: using maven or the JUnit activator class. To run the test with maven you can for example run:

```
mvn test
```

If you want to run the tests using the activator class:

1. Right click the `ScenarioJUnitActivatorTest.java` file and select `Run Java`:



2. The execution results should show up:

#	Scenario description	GIVEN		EXPECT
		request price	request type	
1	Should approve requests under 150	150	"urgent"	true
2	Did not approve low priority requests higher than	160	"low"	false
3	Should approve urgent requests until 1500	1500	"urgent"	true
4	not automatically approve any request higher tha...	1501	"urgent"	false

3. Try changing the line one expected result from true to false . Click on the re-run button to see the results.

#	Scenario description	GIVEN		EXPECT
		request price	request type	
1	Should approve requests under 150	150	"urgent"	true
2	Did not approve low priority requests higher than	160	"low"	false
3	Should approve urgent requests until 1500	1500	"urgent"	true
4	not automatically approve any request higher tha...	1501	"urgent"	false

Finally, adjust the tests and make sure that your project can compile when you run:

```
mvn clean install
```

### 35.3 Next Steps

Now it's time to deploy our project to KIE Server and test it out.

Last update: 2023-12-06

### 3.10.4 36. Deploying the project in KIE Server

It's time to deploy our business application in KIE Server.

#### 36.1 Deployment

We can deploy the project directly in KIE Server without the need to use Business Central. To do so, we can use the available REST API.

1. Open KIE Server REST API. (i.e. <http://localhost:8080/kie-server/docs>)
2. Under "KIE Server and KIE container" category select the following:
3. `PUT /server/containers/{containerId}` Creates a new KIE container in the KIE Server with a specified KIE container ID
4. Click on "Try it out"
5. Insert your project details. The GAV can be found for example, in your `pom.xml`. See an example:
6. **containerId**: mybusinessapp
7. **body**:

```
{"container-id" : "mybusinessapp",
 "release-id" : {
   "group-id" : "org.kie.businessapp",
   "artifact-id" : "mybusinessapp",
   "version" : "1.0"
 }
}
```

8. Click on the blue button "Execute". You should get a 201 result as follows:

Code	Details
201	<b>Response body</b> <pre>{   "type": "SUCCESS",   "msg": "Container mybusinessapp successfully deployed with module org.kie.businessapp:mybusinessapp:1.0.",   "result": {     "kie-container": {       "container-id": "mybusinessapp",       "release-id": {         "group-id": "org.kie.businessapp",         "artifact-id": "mybusinessapp",         "version": "1.0"       },       "resolved-release-id": {         "group-id": "org.kie.businessapp",         "artifact-id": "mybusinessapp",         "version": "1.0"       },       "status": "STARTED",     }   } }</pre>

#### 36.2 Testing the Automated Approval Decision

Now, using the KIE server REST API, we'll consume the decision we've just deployed.

1. Under the section **DMN Models** locate:
2. `POST /server/containers/{containerId}/dmn` Evaluates decisions for given input
3. Click on try it out
4. Use the following data:
5. **ContainerID**: mybusinessapp
6. **Body**:

```
{
  "dmn-context": {
```

```

    "request type": "urgent",
    "request price": "250"
}
}

```

### 36.3 Extra Lab: Business Central

Finally, you can import this project in Business Central. In order to do so, this needs to be a git-based project and Business Central needs to have access to the git repository where the project is stored. The following steps consider a local environment scenario.

1. Access your application folder in the terminal.
2. Initialize the git repository and do the first commit
3. `git init`
4. `git add -A`
5. `git commit -m "first commit"`
6. With this you can already import the project in Business Central. Open Business Central and select the **import the project** option.
7. In the pop-up, in the **Repository URL** field, you should insert the git repository. If it is on your local machine you can inform something like: `/$PROJECT_DIR/tooling-labs/mybusinessapp`. Confirm the operation.
8. You should see the project. Select it and click the **ok** button.

The screenshot shows the Business Central interface. At the top, there is a dark header bar with the text "Business Central" on the left, a home icon in the center, and "Menu" with a dropdown arrow on the right. Below the header, the navigation path "Spaces > MySpace > Import Projects" is visible. The main content area has a light gray background and features a search bar at the top. Below the search bar, a white rectangular card displays the imported project information: "mybusinessapp" followed by a small blue circular icon with a checkmark, and the text "Example 'mybusinessapp' module". A green checkmark icon is located at the bottom left of the card.

Feel free to explore the project and validate the test scenario and deployment through Business Central.

Business Central    Home    Menu ▾

Spaces » MySpace » mybusinessapp » master

mybusinessapp

Assets 6 Change Requests 0 Contributors 2 Metrics Settings

All ▾   1-6 of 6 < 1 > of 1 Import Asset Add Asset

 automated-request-approval	DMN	Last modified Today	Created Today
 automated-request-approval	DMN	Last modified Today	Created Today
 ScenarioJUnitActivatorTest	Data Objects	Last modified Today	Created Today

Alerts

Level	Text	File	Column	Line
Info	Completed indexing of MySpace/mybusinessapp/master	-	0	0

Last update: 2023-12-06