Open Editions BootCamp

None

IBM

None

Table of contents

1. Open Editions Bootcamp

Welcome to the Open Editions Bootcamp. This series of hands-on exercises is designed to guide you through the essential aspects of process automation using BAMOE. Whether you are a developer or an architect, these labs will equip you with the skills needed to effectively leverage BAMOE for modern, cloud-native business automation solutions.

Throughout this bootcamp, you will gain practical experience in designing, implementing, and deploying business processes. You will also explore how to integrate decision services and external systems within your automation workflows, all while utilizing the powerful tools provided by BAMOE.

Some features covered in this bootcamp may be in tech preview. For the latest information on fully supported and tech preview features, please refer to the product documentation.

By the end of these labs, you will:

- Understand the core components and architecture of BAMOE.
- Gain practical skills in designing, implementing, and deploying business processes.
- Learn to integrate decision services and external systems within your processes.
- Develop proficiency in using Canvas and VS Code for process authoring.

1.1 About the Bootcamp

In today's rapidly evolving business landscape, lightweight automation is key to efficiency and scalability on the cloud. Through these guides, developers and architects can gain hands-on experience with the latest capabilities of IBM Business Automation Manager Open Edition.

Participants will explore the development experience and experiment with multiple capabilities for efficiently creating process automation solutions backed by open-source practices and IBM.

1.1.1 Who Is This For

This bootcamp is designed for technical users, such as developers and architects, who are seeking to explore the latest in business automation. It is ideal for professionals looking to deepen their knowledge of IBM BAMOE and how it can be used to create cloud-native solutions.

1.1.2 What You'll Learn

- **Get Started with Cloud-Native Business Automation**: Discover the principles and benefits of cloud-native business automation and how IBM BAMOE enables it.
- **Explore the Features of Development Tools**: Dive into IBM BAMOE Canvas and IBM BAMOE Developer Tools for Microsoft VSCode, understanding their functionalities and how they enhance the development experience.
- **Getting Started with Process Services**: Learn the basics of creating and managing process services, and understand their role in business automation.
- **Explore the Concepts of Process Automation**: Understand the fundamental concepts and workflows involved in process automation, including modeling, execution, and monitoring.
- **Leverage Event-Driven Capabilities in Process Automation**: Explore how event-driven architecture can be integrated into process automation to enhance responsiveness and scalability.
- **Deploying on OpenShift**: Gain insights into deploying process and decision services on OpenShift, ensuring scalable and resilient operations.

This bootcamp will equip you with the skills and knowledge needed to effectively utilize IBM Business Automation Manager Open Editions for modern, cloud-native business automation.

Last update: 2024-07-01

2. Guides

2.1 Getting Started

2.1.1 1. Getting started

In this section you will become familiar with foundational concepts of the product and use the main features of Canvas with decisions based on DMN.

Goals:

1. Know the key components of IBM BAMOE.
2. Set up your development environment.
3. Become familiar with the use case.
4. Explore the key features of Canvas through a Decision Automation perspective.

Let's move forward and get a quick overview of IBM BAMOE and its components.

Last update: 2024-07-01

2.1.2 2. Overview of IBM Business Automation Manager Open Editions

2.1 Components

IBM Business Automation Manager Open Editions (IBM BAMOE) is a comprehensive platform for automating business processes and decision services. It provides a range of tools and components that enable you to design, implement, and manage business automation solutions.

2.2 Key Components

IBM BAMOE consists of several core components essential for development, execution, and management of business automation processes.

Development Tools

- `Canvas` : A web-based environment for designing and modeling business processes and decisions.
- `IBM BAMOE Developer Tools for VS Code` : Tools for developing business processes and decisions within the Visual Studio Code environment.
- `Forms CLI` : A command-line tool for generating web forms required for process execution.

Execution

- `Decision and Rules Engine` : Executes decision models and business rules.
- `Process Engine` : Executes business process models.

Management

- `Management Console` : Provides an interface for managing and monitoring process instances and tasks.
- `Task Inbox` : A user interface for viewing and interacting with user tasks in BAMOE process services.

With a solid understanding of IBM BAMOE's main components, we will introduce the primary use case that will guide our hands-on exercises.

Last update: 2024-07-01

2.1.3 3. Use Case Overview

The Credit Card Application Process is a comprehensive workflow that automates the evaluation and approval of credit card applications. This process ensures that applications are thoroughly evaluated and processed accurately.

3.1 Objectives of the Business Process

The primary objective of this process is to evaluate applicant data and decide whether to approve or reject the application. This ensures that only qualified applicants receive a credit card while minimizing risk for the financial institution.

3.2 Key Steps in the Process

1. **Receive Application**:
   • Receive and parse the application.
   • Validate initial data and assign a unique application ID.
2. **Evaluate Application**:
   • Assess the applicant's credit score and financial information.
   • Applications with a credit score below 550 are automatically rejected.
   • Applications with scores between 550 and 649 are sent for manual review.
   • High scores (650 and above) are considered for further processing.
3. **Manual Review**:
   • Further scrutiny by a financial officer.
   • Detailed analysis of the applicant's financial history and relevant factors.
4. **Generate Card Details**:
   • Once approved, generate necessary card details such as card number, CVV, expiration date, credit limit, and APR.
5. **Approval and Notification**:
   • Approved applicants receive their card details, while rejected applicants are informed of the reasons for rejection.

3.3 Business Rules and External Validations

The process incorporates several business rules to ensure accurate decision-making: - **Credit Score Assessment**: Determines initial eligibility based on predefined thresholds. - **Debt-to-Income Ratio Check**: Ensures applicants have a manageable level of debt relative to their income. - **Fraud Risk Assessment**: Uses external AI services to detect potential fraud, adding an extra layer of security.

3.3.1 APPROVALS AND VALIDATIONS

- **Automated Approvals**: Based on clear-cut criteria such as credit score and debt-to-income ratio.
- **Manual Approvals**: Required for borderline cases where human judgment is necessary.
- **External Service Validations**: Involving AI-based fraud detection and other external checks to ensure data authenticity and reliability.

Next, let's delve into the environment setup required to begin our journey with BAMOE.

Last update: 2024-07-01

2.1.4 4. Environment Setup

In this section, we will set up the necessary environment to start working with IBM Business Automation Manager Open Editions (BAMOE).

4.1 Prerequisites

Below is the list of technologies you will need.

- Java JDK 17
- Maven 3.9.6
- OpenShift 4.15 (*Need an OpenShift? Try the Dev Sandbox for Red Hat OpenShift for free.*)
- Git CLI
- Microsoft VSCode (latest version) *(Need the IDE? Get it here: VSCode download page.)*

> **Tip**
>
> For more information about the supported environments of IBM BAMOE, refer to the official documentation

You may need an account on these websites in order to rely on easy to use cloud services in some of the exercises:

- GitHub - for versioning projects on GitHub
- Red Hat - For those using Dev Sandbox for OpenShift
- IBM - For using IBM's managed Kafka service

Verifying Installed Software & Version

To verify if you have the correct versions installed, you can use the following commands:

```
java -version
mvn -version
oc version
git --version
```

4.1.1 PREPARING VSCODE AND OPENSHIFT DEV SANDBOX

If you have an OpenShift instance ready to use, and if the IBM BAMOE Dev Tools extension is already installed your VSCode IDE, you can jump to 4.1.2 installing canvas

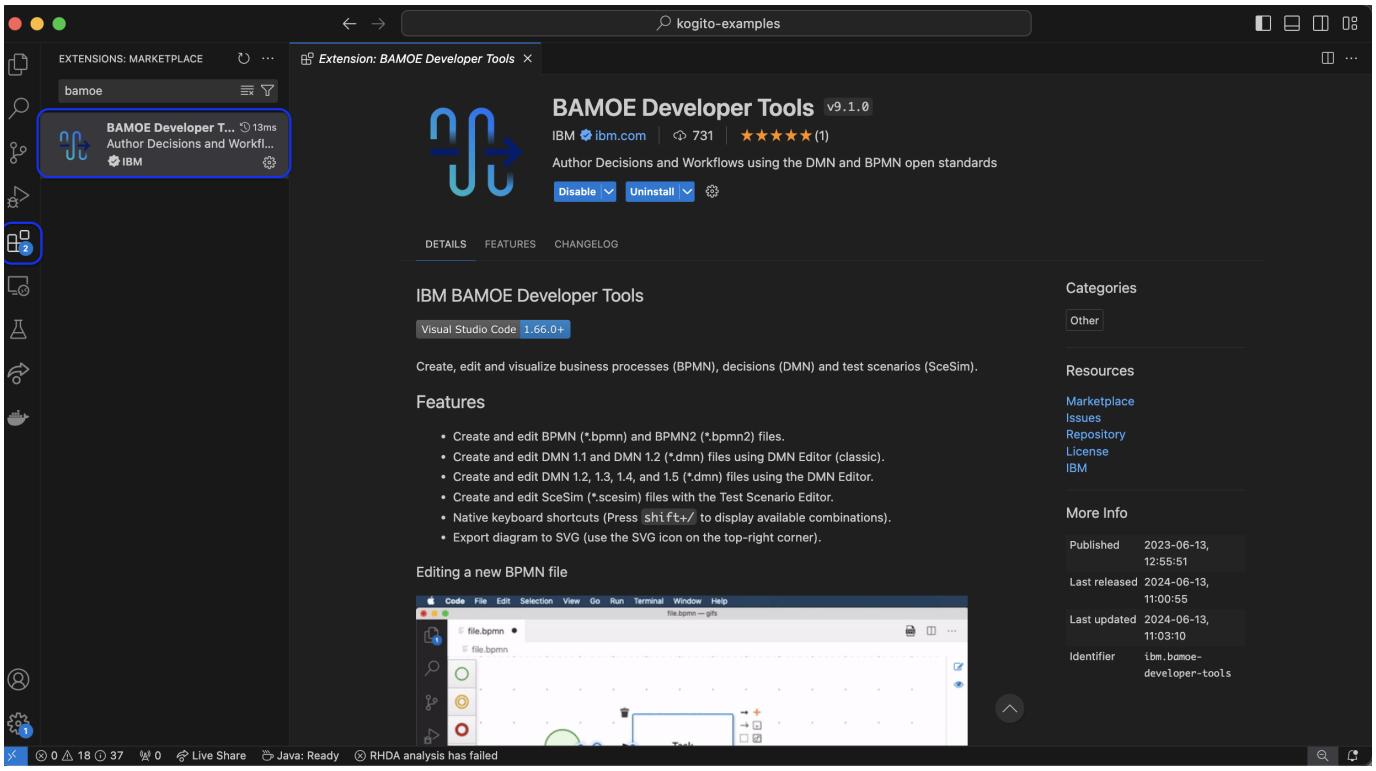
4.1.1.1 Setting Up Local Development Environment with VS Code

For running through the exercises, we'll use **Visual Studio Code** combined with IBM BAMOE Developer Tools Extension for Microsoft VSCode.

4.1.1.2 Setting Up IBM BAMOE Developer Tools for VS Code

To install BAMOE Dev Tools, open your IDE and go through the following steps:

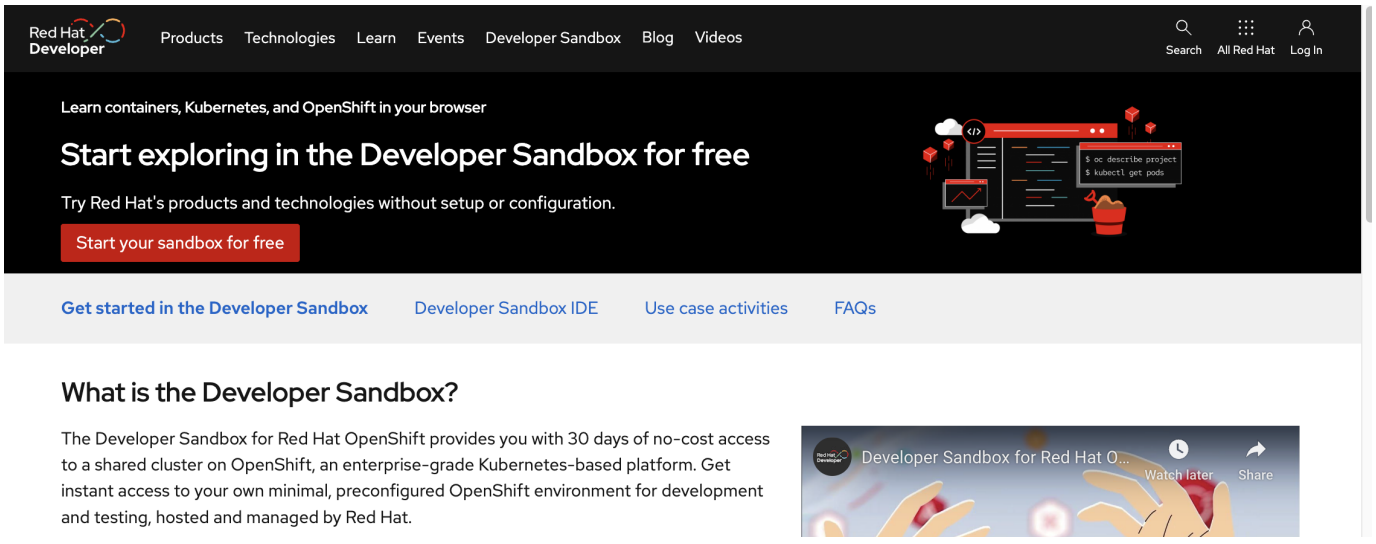
1. Go to the Extensions view by clicking on the Extensions icon in the Activity Bar on the side of the window or by pressing `Ctrl+Shift+X`.
2. In the search box, type `IBM BAMOE Developer Tools`.
3. Click the Install button for the IBM BAMOE Developer Tools extension.



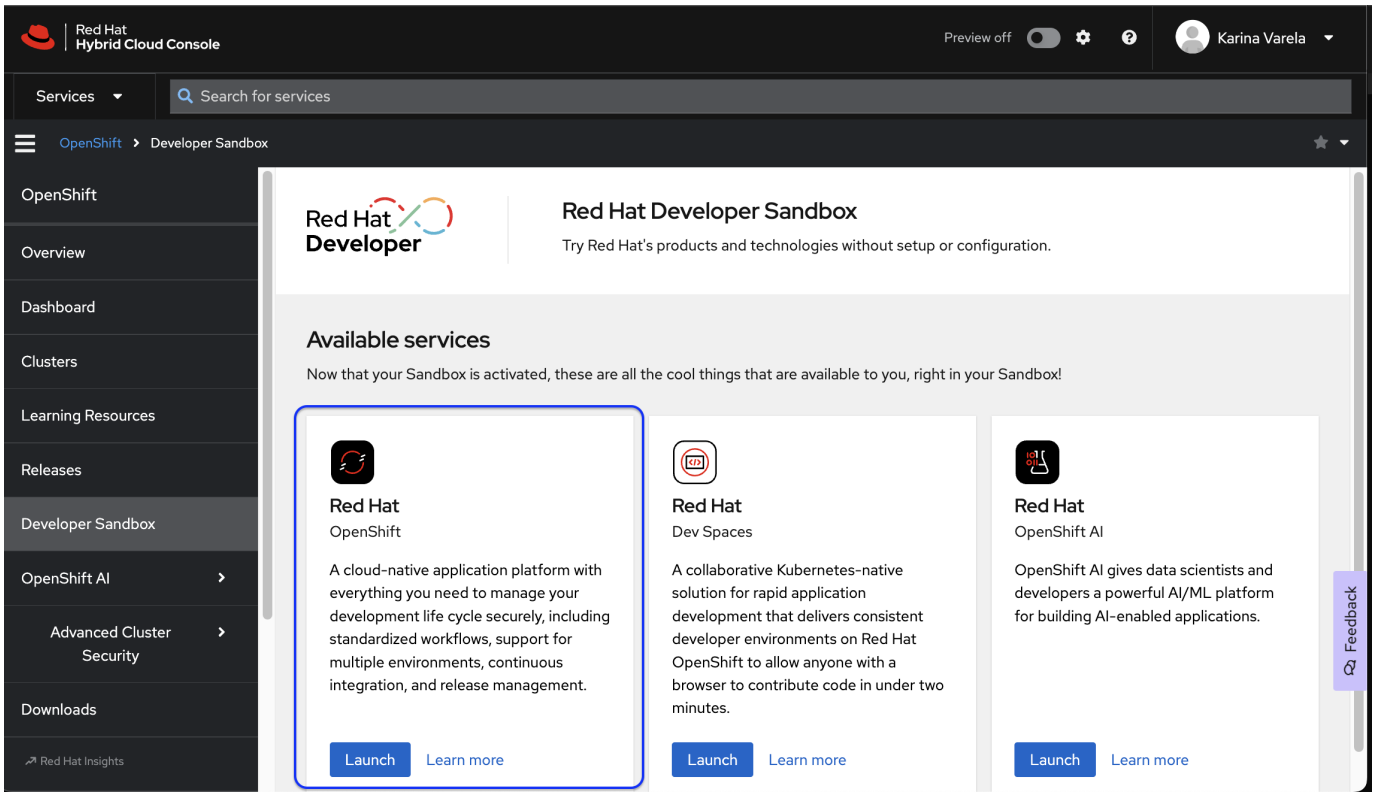
4.1.1.2.1 Need an OpenShift Environment? Developer Sandbox for Red Hat OpenShift

To go through the OpenShift exercises, you'll need an OpenShift environment up and running. You can use any instance where you have access within a project. If you need a free, easy-to-setup OpenShift for our learning purposes, you can use the Developer Sandbox for Red Hat OpenShift.

As stated by Red Hat: *"The Developer Sandbox for Red Hat OpenShift provides you with 30 days of no-cost access to a shared cluster on OpenShift, an enterprise-grade Kubernetes-based platform. Get instant access to your own minimal, preconfigured OpenShift environment for development and testing, hosted and managed by Red Hat."*



After you log-in and create a sandbox on the page above, you'll get to access your OpenShift:

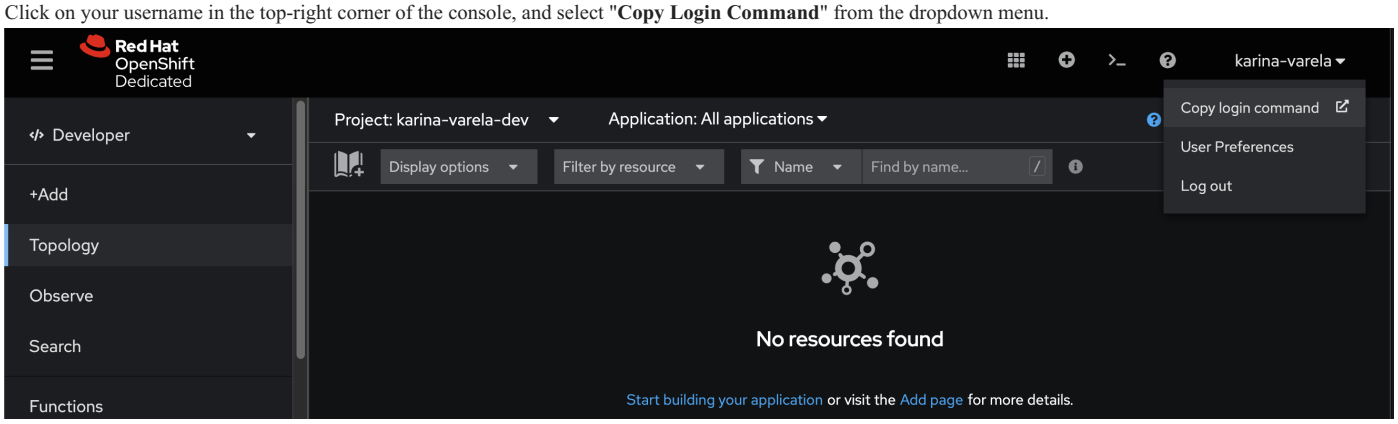


In this environment, you have access to a namespace named with `your-username-dev` (e.g. `karina-varela-dev`).

4.1.1.3 Login to access OpenShift from your Terminal

To deploy BAMOE Canvas to OpenShift, you will need your OpenShift login token. Follow these steps to retrieve it using the OpenShift web console:

1. **Log in to the OpenShift Web Console**:
2. Open your web browser and navigate to your OpenShift cluster's web console URL.
3. Enter your username and password to log in.
4. **Accessing the Token**:
5. Click on your username in the top-right corner of the console, and select "**Copy Login Command**" from the dropdown menu.



6. A new page will open with your login token. Click on the "**Display Token**" button to view your token.
7. Copy the login token provided.
8. **Using the Token to log-in in your terminal**:
9. Open your terminal.
10. Log in to the OpenShift cluster using the `oc` CLI with the following command:

```
oc login --token=<your-token> --server=<your-openshift-api-url>
```

Next, let's install Canvas on OpenShift.

4.1.2 INSTALLING CANVAS ON RED HAT OPENSHIFT CONTAINER PLATFORM (OCP)

IBM BAMOE Canvas is a web application that provides authoring tools for standards based business assets, directly in the browser. It allows users to create, edit, and manage decisions and processes, integrate with git for syncing repositories, and during development phase, deploy files to OpenShift and Kubernetes.

To install Canvas on a container platform, we need to install three resources:

- Extended Services
- CORS Proxy
- IBM BAMOE Canvas

Let's go through the installation and recall the purpose of these resources.

4.1.2.1 Installation Steps

In this section, we will install BAMOE Canvas on an OpenShift cluster. By using a manual installation process, you can have a better view of the resources created on OCP during the installation.

> **Note**
>
> Users who can install `helm` locally, can refer to the official documentation on how to Install Canvas with Helm.

4.1.3 INSTALLING CANVAS ON OPENSHIFT

Let's install **Canvas**, the environment we'll explore on the next lab for experimenting with automation services development with IBM BAMOE.

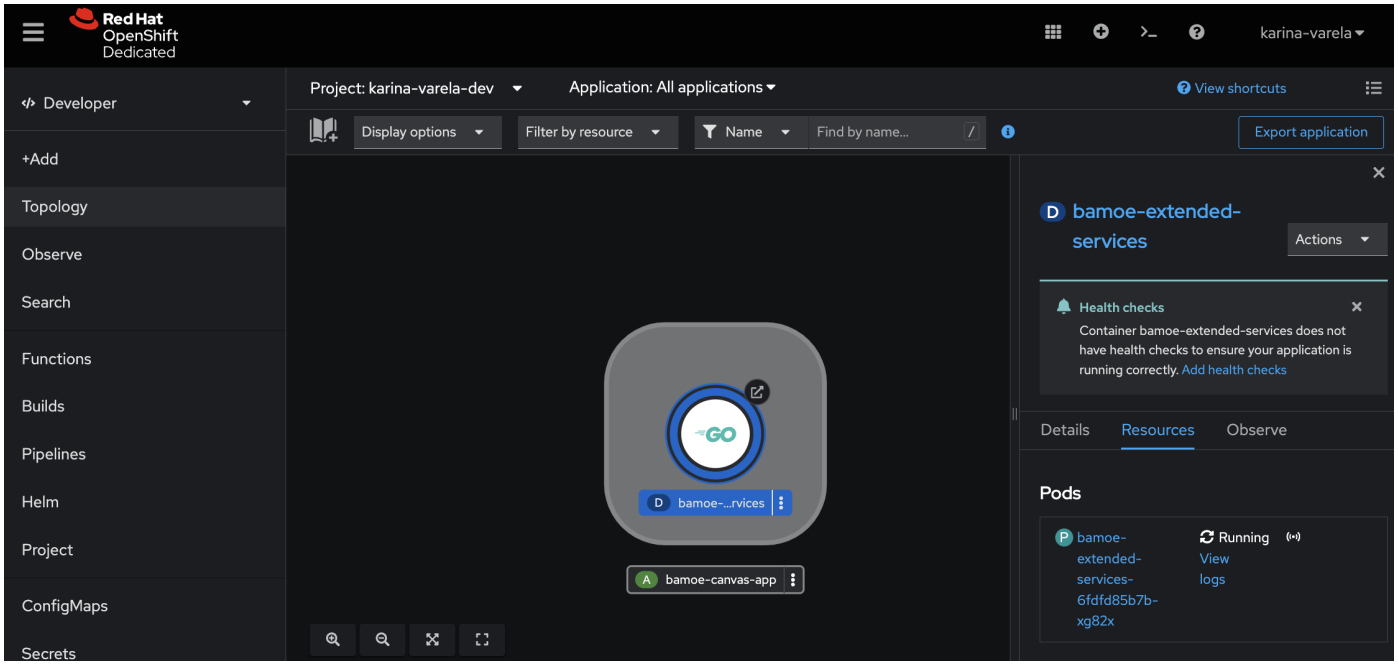
Canvas is a powerful web application that provides tools for authoring decisions and workflows directly in the browser. It integrates seamlessly with Git for version control and with OpenShift for deploying your models for development validation purposes.

1. **Deploy Extended Services**

   Extended Services are back-end services that provide additional features to Canvas, such as the DMN Runner (execution and validation of decision models) and a proxy (enables communication with OpenShift and Kubernetes clusters).

```
export APP_PART_OF=bamoe-canvas-app
export APP_NAME_EXTENDED_SERVICES=bamoe-extended-services
oc new-app quay.io/bamoe/extended-services:9.1.0-ibm-0001 --name=$APP_NAME_EXTENDED_SERVICES
oc create route edge --service=$APP_NAME_EXTENDED_SERVICES
oc label services/$APP_NAME_EXTENDED_SERVICES app.kubernetes.io/part-of=$APP_PART_OF
oc label routes/$APP_NAME_EXTENDED_SERVICES app.kubernetes.io/part-of=$APP_PART_OF
oc label deployments/$APP_NAME_EXTENDED_SERVICES app.kubernetes.io/part-of=$APP_PART_OF
oc label deployments/$APP_NAME_EXTENDED_SERVICES app.openshift.io/runtime=golang
```

   If you check your OpenShift console, you should be able to see the new pod is up and running. You can do the same observation for the next two deployments as well.



2. **Deploy CORS Proxy**

   The CORS Proxy allows BAMOE Canvas to communicate with Git providers like GitHub, Gitlab and Bitbucket.

```
export APP_NAME_CORS_PROXY=bamoe-cors-proxy
oc new-app quay.io/bamoe/cors-proxy:9.1.0-ibm-0001 --name=$APP_NAME_CORS_PROXY
oc create route edge --service=$APP_NAME_CORS_PROXY
oc label services/$APP_NAME_CORS_PROXY app.kubernetes.io/part-of=$APP_PART_OF
oc label routes/$APP_NAME_CORS_PROXY app.kubernetes.io/part-of=$APP_PART_OF
oc label deployments/$APP_NAME_CORS_PROXY app.kubernetes.io/part-of=$APP_PART_OF
oc label deployments/$APP_NAME_CORS_PROXY app.openshift.io/runtime=nodejs
```

3. **Deploy BAMOE Canvas**

   Finally, deploy the BAMOE Canvas image, setting the environment variables required to connect it to the Extended Services and CORS Proxy backends.

```
sh
export APP_NAME_BAMOE_CANVAS=bamoe-canvas
oc new-app quay.io/bamoe/canvas:9.1.0-ibm-0001 --name=$APP_NAME_BAMOE_CANVAS \
    --env=KIE_SANDBOX_EXTENDED_SERVICES_URL=https://$(oc get route $APP_NAME_EXTENDED_SERVICES --output jsonpath={.spec.host}) \
    --env=KIE_SANDBOX_CORS_PROXY_URL=https://$(oc get route $APP_NAME_CORS_PROXY --output jsonpath={.spec.host})
oc create route edge --service=$APP_NAME_BAMOE_CANVAS
oc label services/$APP_NAME_BAMOE_CANVAS app.kubernetes.io/part-of=$APP_PART_OF
oc label routes=$APP_NAME_BAMOE_CANVAS app.kubernetes.io/part-of=$APP_PART_OF
oc label deployments/$APP_NAME_BAMOE_CANVAS app.kubernetes.io/part-of=$APP_PART_OF
oc label deployments/$APP_NAME_BAMOE_CANVAS app.openshift.io/runtime=js
```

4. Access IBM BAMOE Canvas

If all went well, you should be able to see three pods - either on the OCP web console or using the cli to retrieve the pods of the current namespace: `oc get pods`. You should see something like:

```
oc get pods
NAME                                    READY   STATUS    RESTARTS
bamoe-canvas-54f87f584c-dpjr6           1/1     Running   0
bamoe-cors-proxy-578bf787cb-rc4t8       1/1     Running   0
bamoe-extended-services-6fdfd85b7b-xg82x   1/1     Running   0
```

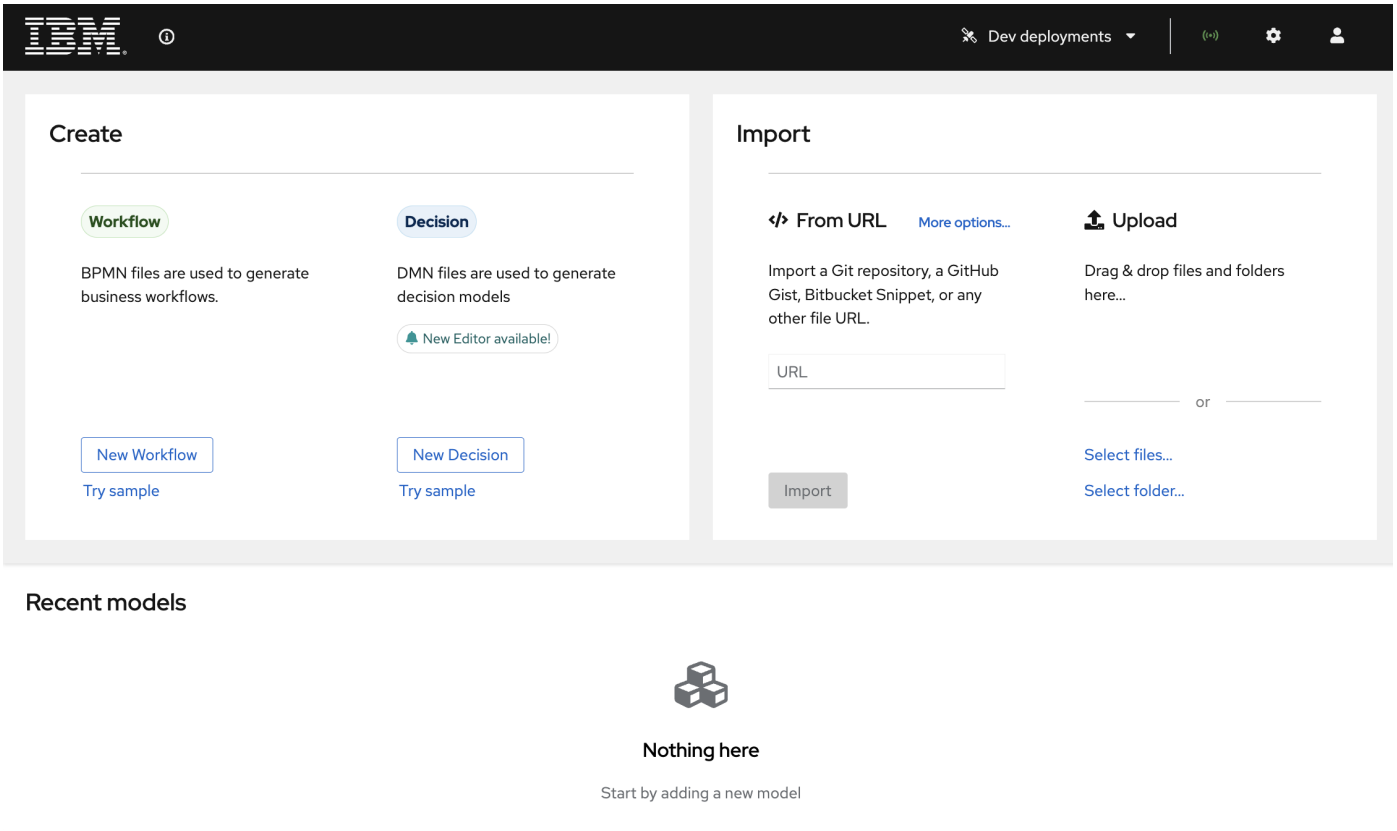
Your IBM BAMOE Canvas instance should be up and accessible. To get Canvas' URL, run this command:

```
oc get route $APP_NAME_BAMOE_CANVAS --output jsonpath={.spec.host}; echo
```

The retrieved URL would look something like

`bamoe-canvas-username-dev.apps.sandbox-id.p1.openshiftapps.com/#/`.

The URL should lead to your new installation of Canvas:



With your development tools prepared, you're now ready to start exploring the solution in more detail. Let's move on to exploring Canvas with Decision Automation.

4.1.4 PREPARING CANVAS FOR INTEGRATING WITH GIT AND OPENSHIFT

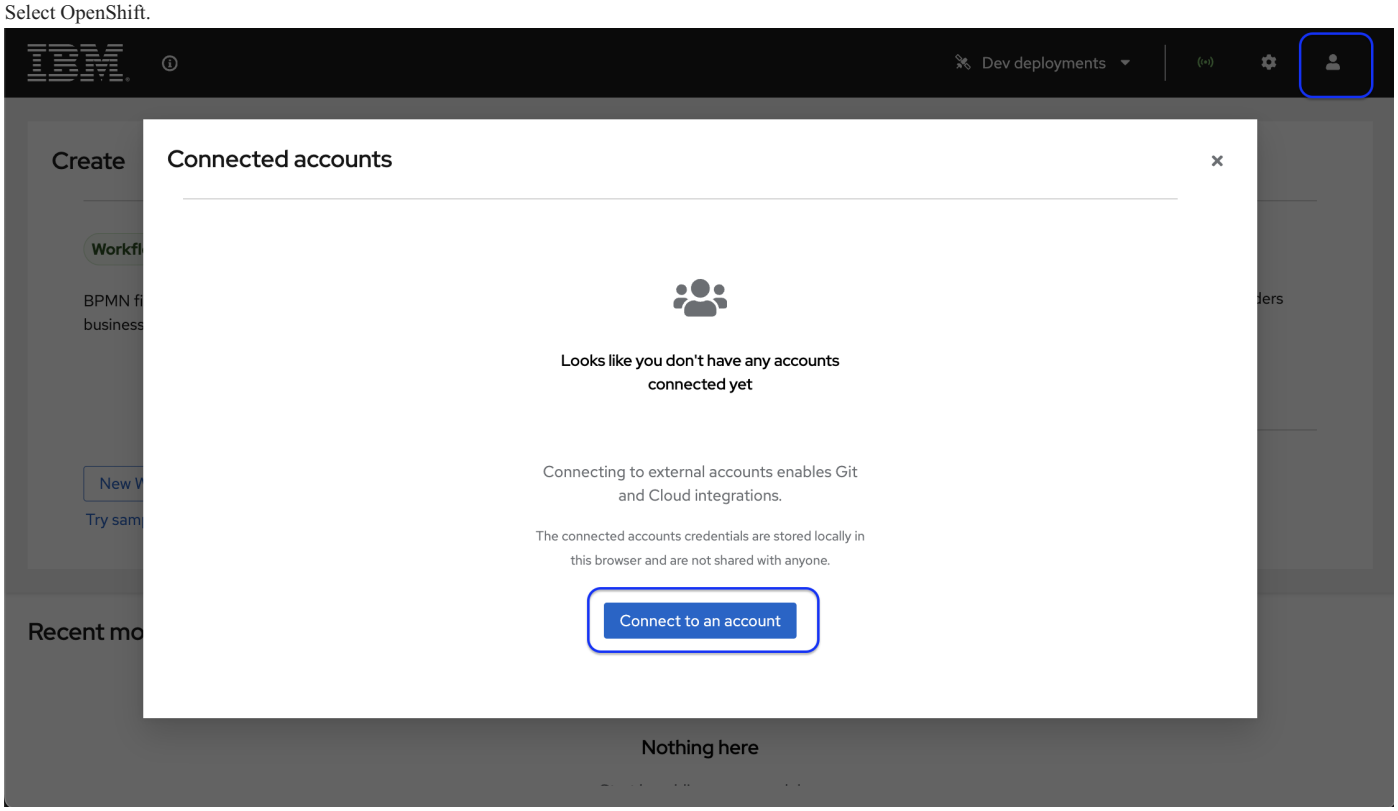
To be able to use the maximum potential of Canvas, let's set up the integration with Git and OpenShift.

In Canvas, execute the following procedures to set up the integration:

4.1.4.1 Connect Canvas to OpenShift for Deployment and Integration

1. In Canvas, click on the "User" icon on the top-right corner, and click on "Connect to an account".

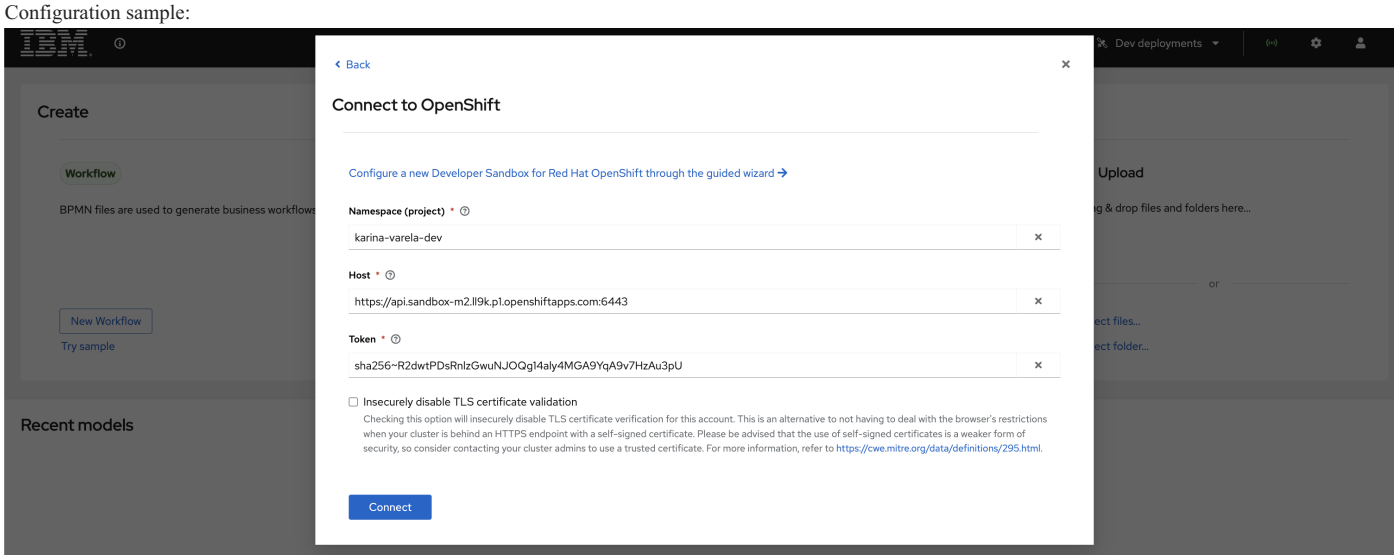
2. Select OpenShift.



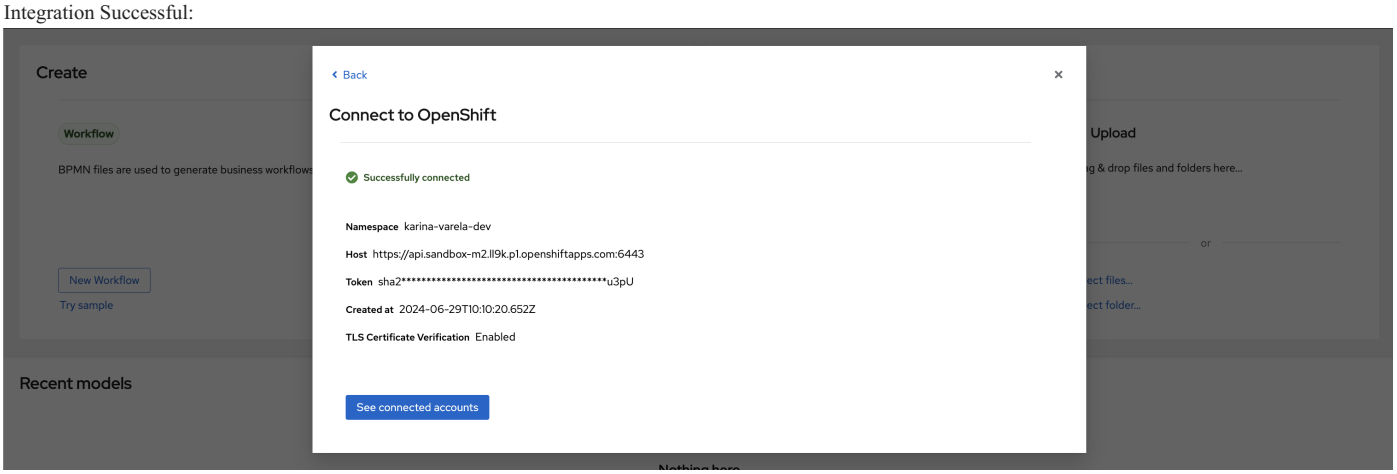
3. Enter the OpenShift API URL and your project name. You can obtain them through the web console or using the CLI commands below to retrieve the namespace, API URL, and authentication token respectively:

```
oc project -q
oc whoami --show-server
oc whoami --show-token
```

Configuration sample:



Integration Successful:

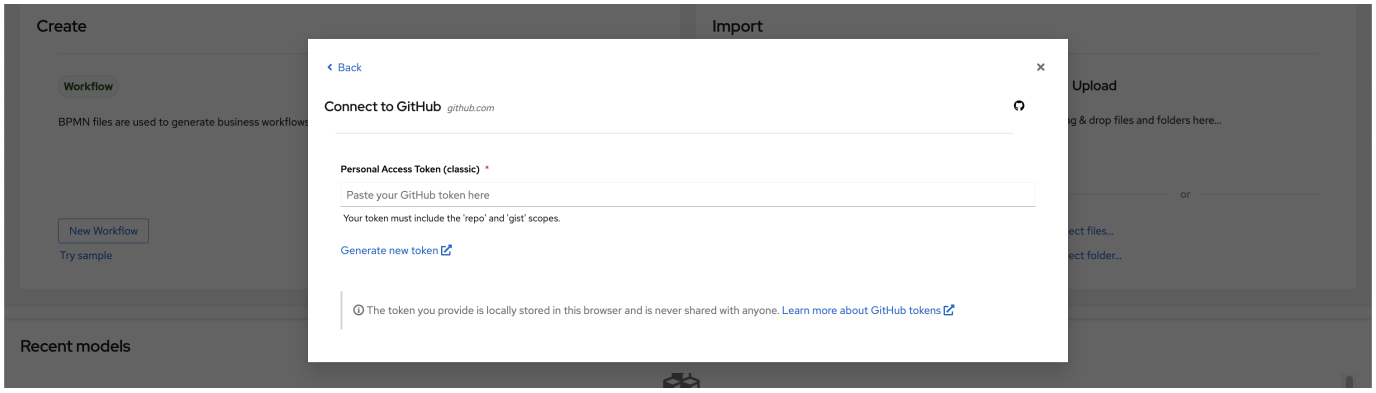


4. Save the settings.

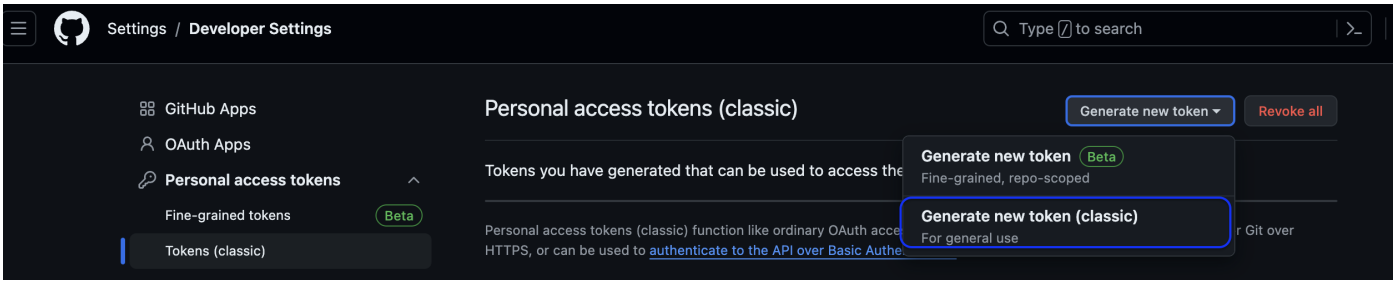
Your Canvas can now run dev deployments of your decision services on OpenShift! To wrap up, let's configure github integration.

4.1.5 CONNECT CANVAS TO GITHUB FOR VERSION CONTROL AND COLLABORATION

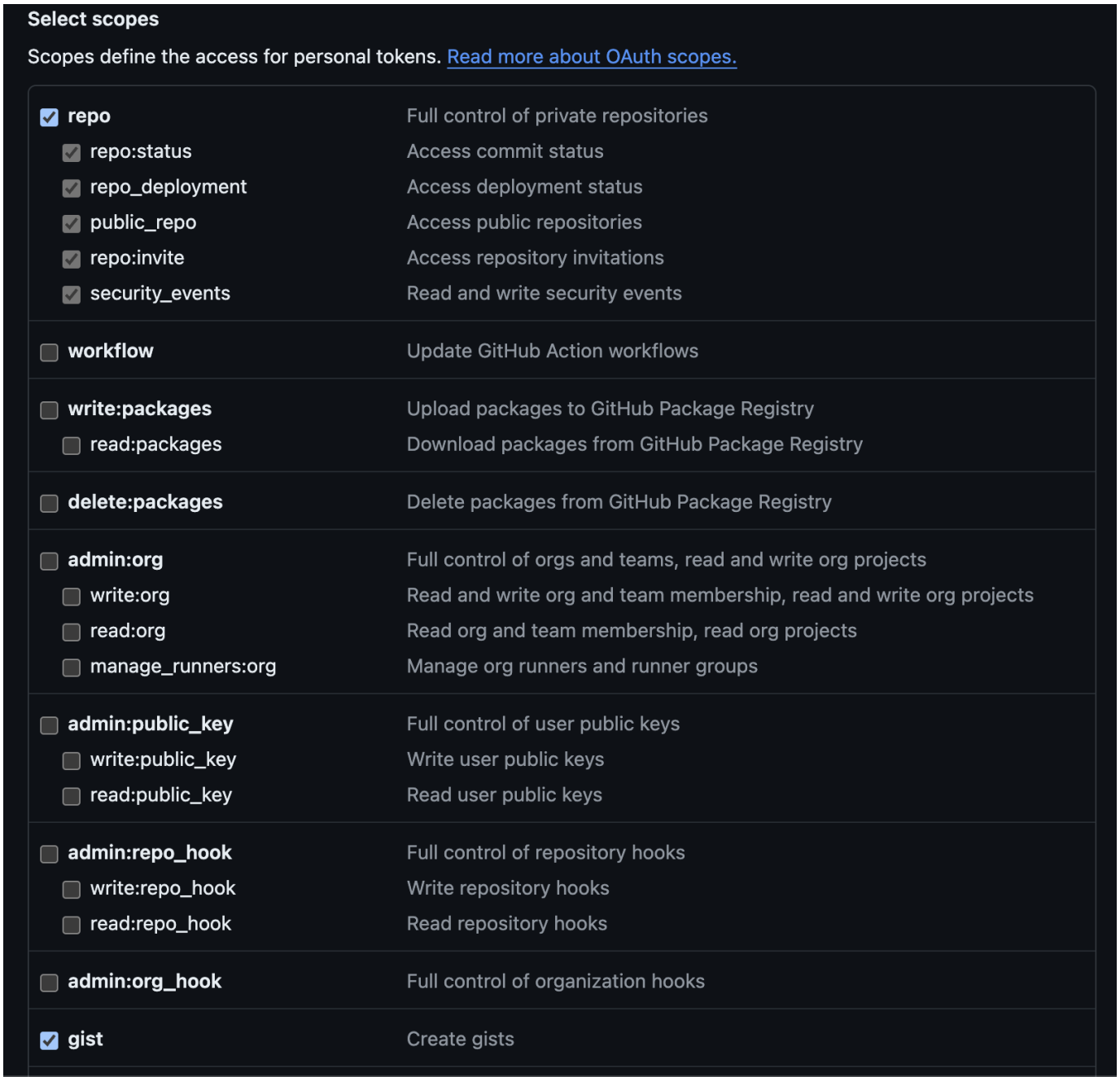
1. In Canvas, go to the settings menu and select "GitHub Integration".!
2. Click on **Generate new token**. You'll be redirected to the GitHub page where you can create a token for Canvas to authenticate on GitHub with the given permissions.



1. In GitHub, create a new classic token



2. Choose a `note` (any description for this token) and the permissions for 'repo' and 'gist' scopes.



1. On the bottom of the page, click on "**Generate Token**". Copy and save the generated token, you'll need it in Canvas.
2. Now back in Canvas, enter your GitHub repository URL and personal access token. You should see a green message informing you have **Successfully connected**.

Awesome!! You're all set! With your development tools prepared, you're now ready to start exploring the solution in more detail. Let's move on to exploring Canvas with Decision Automation.

Last update: 2024-07-01

2.2 Exploring Canvas & Decisions

2.2.1 5. Getting Started with Exploring Canvas & Decisions

In this section, you will become familiar with the foundational concepts of IBM Business Automation Manager Open Editions (BAMOE) and use the main features of Canvas for decision automation using DMN.

5.1 Goals:

- Get to know Canvas and DMN
- Import and Explore Projects:
- Test your decision models directly in Canvas.
- Deploy your decision models on OpenShift using Canvas Dev Deployment.
- Explore and try the REST endpoints provided by decision services in BAMOE.

Let's move forward and get a quick overview of the value of open standards for decision automation.

Last update: 2024-07-01

2.2.2 6. Canvas & Decision Model and Notation (DMN)

6.1 The Value of Open Standards for Decision Automation

6.1.1 WHY DMN, AND WHAT IS IT

Decision Model and Notation (DMN) is an open standard that provides a powerful framework for decision automation. It offers a visual notation for decision modeling, a standardized approach to defining decision logic, and ensures interoperability between different tools and platforms. Designed to be easily understood by all stakeholders, from business users to developers, DMN enables the creation of executable models that can be run on various platforms, ensuring consistent decision-making processes.

DMN is particularly valuable because it: - Enhances communication between business and technical teams through visual notation - Ensures consistency and clarity in decision logic across different systems - Facilitates sharing and reuse of models across various tools and platforms - Supports a wide range of decision logic, from simple rules to complex decision trees

6.1.2 FEEL: A STANDARD FRIENDLY ENOUGH EXPRESSION LANGUAGE

FEEL (Friendly Enough Expression Language) is a key component of DMN that provides an intuitive way to express decision logic. Here are some examples of FEEL to give you an idea of its simplicity:

Simple Conditional Logic: Checking if an applicant's age is above 18:

```
if applicant.age > 18 then "Eligible" else "Not Eligible"
```

String Manipulation: Concatenating the first name and last name of an applicant:

```
applicant.firstName + " " + applicant.lastName
```

List Operations: Calculating the average score from a list of scores:

```
sum(scores) / count(scores)
```

Date Operations: Calculating the number of days between two dates:

```
days between startDate and endDate
```

6.2 IBM & OMG: Working towards the evolution of DMN

The collaboration between IBM and the Object Management Group (OMG) has been instrumental in the development and evolution of DMN. This partnership ensures continuous improvement of the standard, alignment with industry needs, and broader adoption across various sectors.

> **Did you know**
>
> The team behind Drools, an open-source decision management system, has been actively involved with OMG since the early days of DMN. They have consistently implemented the standard, even in its early versions. For more detailed information on Drools' implementation of DMN, you can explore Drools' documentation.

Next, let's explore the development experience with Canvas, starting with connecting Canvas to OpenShift.

6.3 Development Experience with Canvas

> **Note**
>
> If you haven't installed Canvas yet, please follow the installation guide here: Installing Canvas on OpenShift

6.3.1 CONNECTING CANVAS TO GITHUB

To connect BAMOE Canvas to GitHub, follow these steps:

1. **Create a GitHub Personal Access Token:**
   - Go to GitHub Settings
   - Click on "Generate new token"
   - Select the required scopes for accessing your repositories
   - Generate the token and copy it
2. **Configure Canvas with GitHub Token:**
   - Access BAMOE Canvas web interface
   - Go to Settings
   - Enter your GitHub personal access token in the provided field
   - Save the configuration

With these steps, BAMOE Canvas will be connected to your GitHub account, allowing you to import projects and synchronize your work.

Great job! You now have a solid understanding of DMN and its value in decision automation. Let's dive into the development experience with Canvas.

Last update: 2024-07-01

2.2.3 7. Exploring Canvas & Decisions

In this section, we will import an existing DMN project into BAMOE Canvas and explore its decision models and automation capabilities.

7.1 Value of DMN Automation

1. **Consistency**: Ensures uniform application of eligibility criteria across all applications.
2. **Efficiency**: Rapidly processes applications, reducing manual workload.
3. **Flexibility**: Allows easy updates to decision logic without changing the underlying process.
4. **Transparency**: Provides clear decision rationale, aiding in regulatory compliance.
5. **Scalability**: Handles increasing application volumes without proportional increase in resources.

7.2 Importing DMN Projects into Canvas

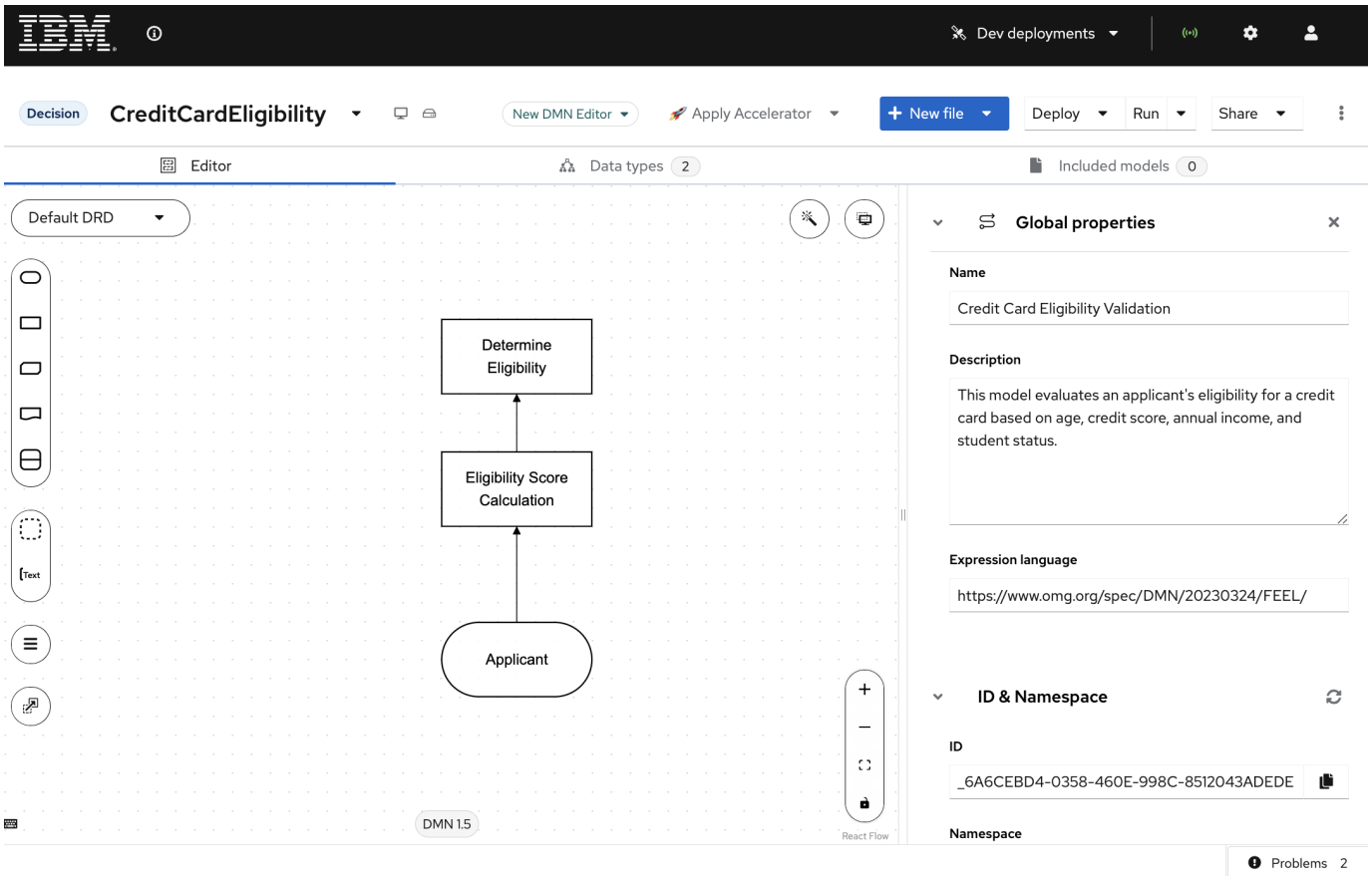
To start, let's import a DMN file into Canvas. We will use a credit card eligibility use case to illustrate this process.

1. Open Canvas in your browser.
2. To import the DMN file, in Canvas go to the Import menu and select "From URL".
3. Copy and paste the following link and click **Import**:

   https://github.com/kmacedovarela/cc-application-approval-starter/blob/main/src/main/resources/CreditCardEligibility.dmn

7.3 Credit Card Eligibility DMN Model

This DMN (Decision Model and Notation) model automates the initial eligibility assessment for credit card applications.



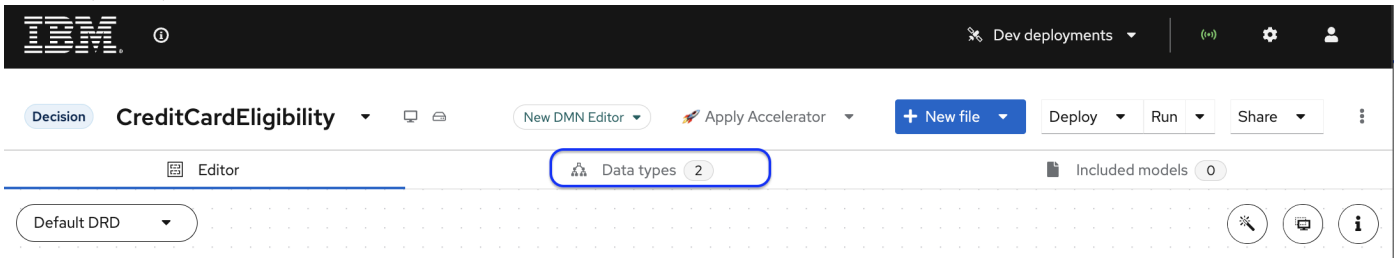
It evaluates applicant information to determine whether an application should be automatically approved, rejected, or sent for manual review.

7.3.1 KEY ELEMENTS

1. **Input Node**: Applicant information (age, credit score, annual income, student status)
2. **Decision Nodes**:
3. Eligibility Score Calculation: Assigns a score based on applicant criteria
4. Determine Eligibility: Interprets the score to make a final decision
5. **Output**: Eligibility result (Approved, Rejected, or Manual Review)

7.4 Exploring the Decision Model

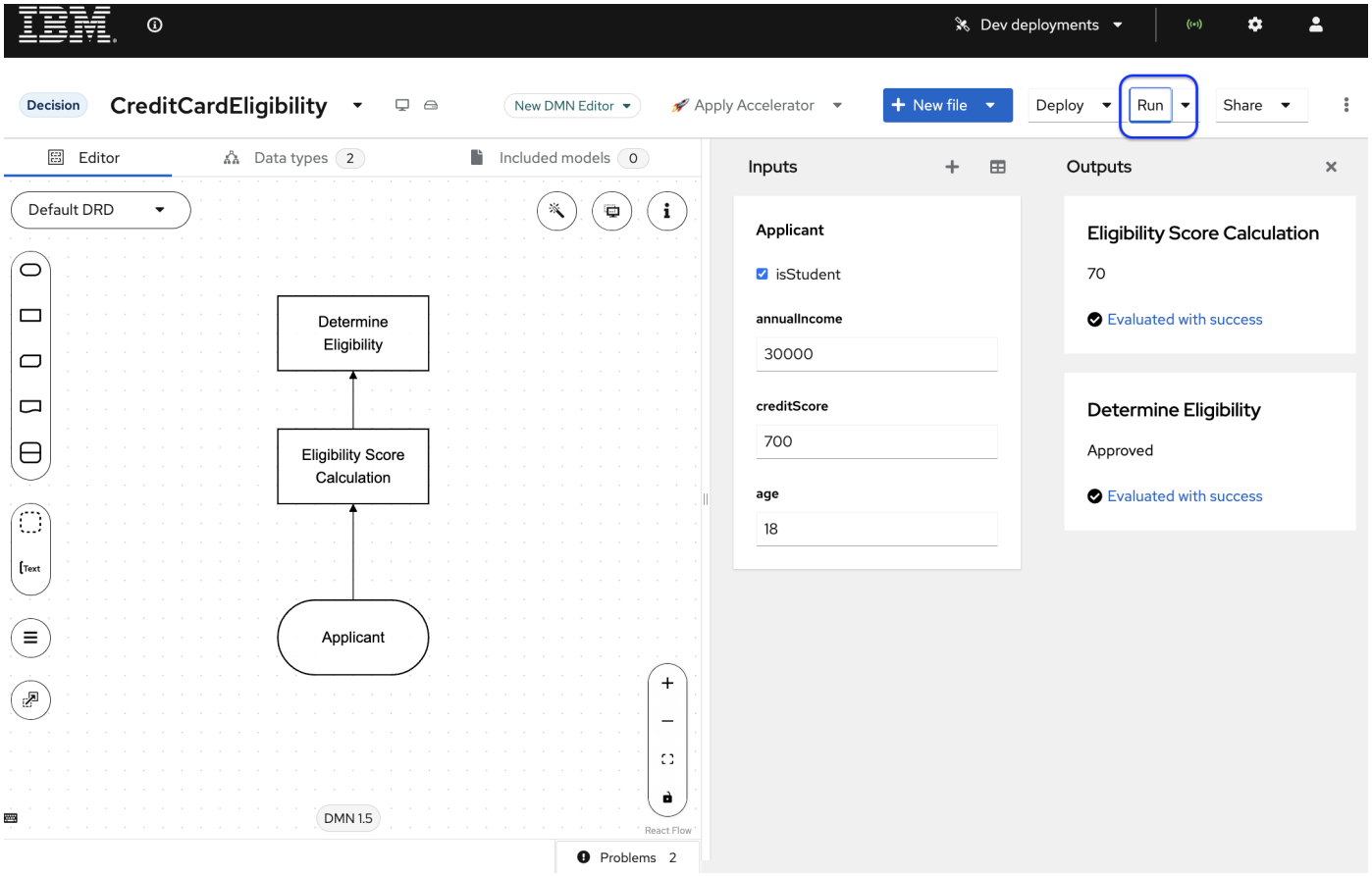
1. Click on the Data Types tab. In it, you can see the `Applicant` type, which includes the attributes: age (Number), creditScore (Number), annualIncome (Number), and isStudent (Boolean).



2. Now, click on the `Editor` tab and let's check the decision nodes. Click on the `Eligibility Score Calculation` node and observe it is of type Number and contains a decision table for calculating the eligibility score based on the applicant's criteria.
3. Next, check the `Determine Eligibility` decision node, which is of type String and uses a FEEL expression to determine the eligibility result based on the calculated score.

7.5 Running the Decision

1. Click on the Run button to see the form that was automatically generated on the right side of the screen.



2. Try out the decision using the test scenarios below.

Test Scenarios

Some examples of input and output data that can validate the decision:

Is Student	Annual Income	Credit Score	Age	Expected Output
false	30000	700	17	Rejected
false	40000	500	25	Manual Review
false	60000	720	30	Approved
true	5000	600	19	Approved
false	80000	650	45	Approved

7.6 Using the Dev Deployment

7.6.1 ABOUT THE DEV DEPLOYMENT

Dev deployments in BAMOE Canvas allow you to deploy Decisions to both local Kubernetes instances and remote Kubernetes/OpenShift environments for development purposes.

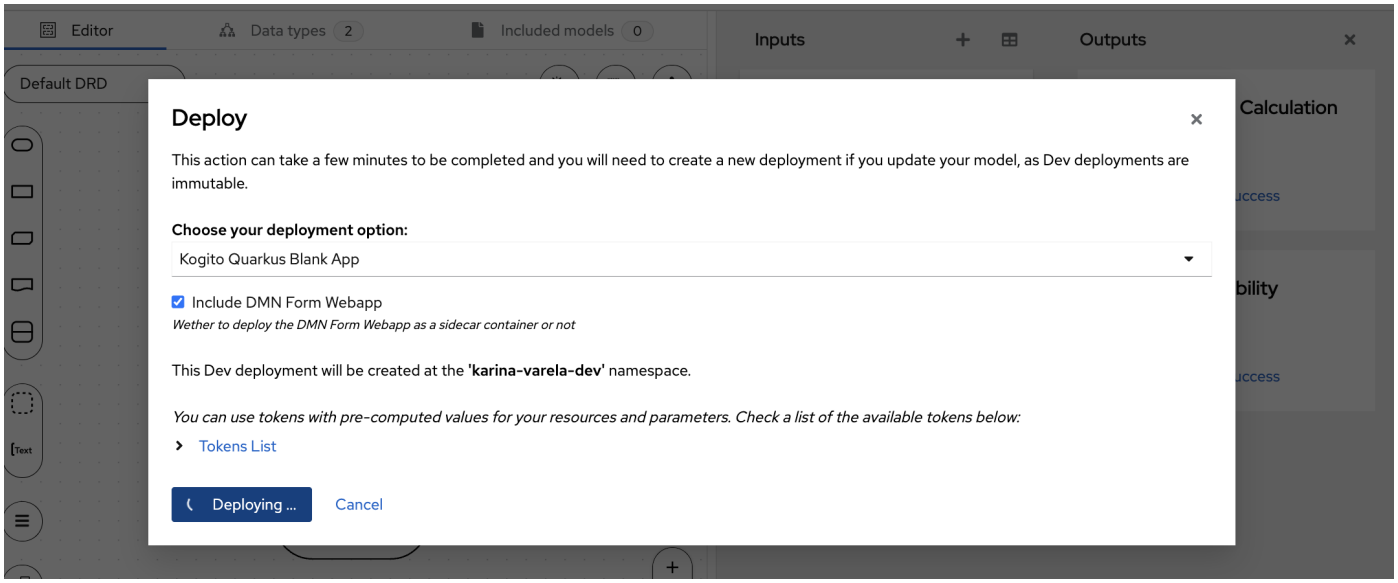
With it, you can deploy all Decisions within a project. After a short deployment process, you will be able to access a web application provided by BAMOE Canvas for testing and interacting with your Decisions. This web application includes a form similar to the one you used when running a decision in Canvas. Along with the form, it also provides access to the Swagger UI with the Decisions' API information.

Tip

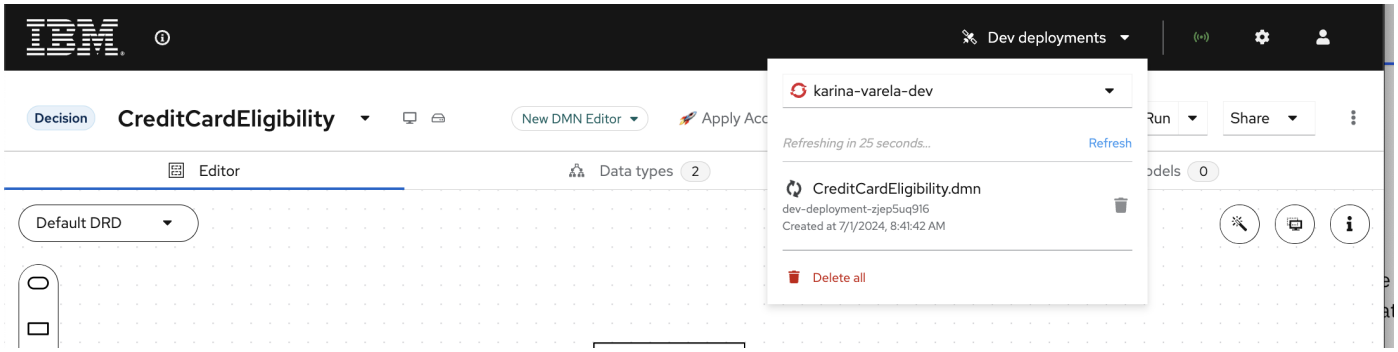
Dev deployments are immutable, meaning if you need to make changes or if an error occurs, you must create a new deployment. Dev deployments can be easily managed and deleted from the Dev deployments dropdown.

7.6.2 STEPS TO DEPLOY

1. With the decision model opened, click on deploy.
2. In the pop-up that opens up, choose the option "Kogito Quarkus Blank App" for "Choose your deployment option".



3. Check the checkbox for "Whether to deploy the DMN Form Webapp as a sidecar container or not".
4. Click on confirm.
5. Click on "Dev Deployments" on the top menu bar to show the deployment status.



6. When available (shows a green check), click on it to open the deployed service on a new page.

A new browser tab will open on a URL similar to this: `https://dev-deployment-yliell9512.apps.sandbox-m2.ll9k.p1.openshiftapps.com/form-webapp/#/form/Credit%20Card%20Eligibility%20Validation`, showing an automatically generated form.

!!! **Note**: Dev deployments are intended to be used for development purposes only, so users should not use the deployed services in production or for any type of business-critical workloads.

7.6.3 TESTING THE REST APIS

After opening the deployed service by opening the service URL, append `/q/swagger-ui` to the URL to access the APIs.

This is based on OpenAPI. You can access the OpenAPI file by opening `/q/openapi`.

7.6.3.1 Evaluating a Decision using REST

Let's try using the POST endpoint available in swagger to fire a decision.

1. Access the swagger ui in the deployed decision service. Locate and click on the endpoint: POST `/Credit Card Eligibility Validation`
2. Click on edit to use the input:

```
{
  "Applicant": {
    "isStudent": false,
    "annualIncome": 30000,
    "creditScore": 700,
    "age": 17
  }
}
```

1. Observe the output, which should be similar to:

```
{
  "Eligibility Score Calculation": 700,
  "Determine Eligibility": "Rejected",
  "Applicant": {
    "isStudent": false,
    "annualIncome": 30000,
    "creditScore": 700,
    "age": 17
  }
}
```

Awesome!! You've successfully imported and explored a DMN project in Canvas. Now that you've seen how to model and test decisions.

Last update: 2024-07-26

2.3 Process Automation: Getting Started

2.3.1 8. Getting Started with Process Automation

In this section, you will become familiar with the foundational concepts of IBM Business Automation Manager Open Editions (BAMOE) and use the main features of Canvas for process automation using BPMN.

8.1 Goals:

- Clone the starter project created with the accelerator using git
- Download to the computer to work on VSCode using the Dev Tools extensions
- Use Quarkus dev mode and use the management console and task inbox
- Expand the process design: Script Task, Human Task, Timer Event, Service Task

Let's get started.

Last update: 2024-07-03

2.3.2 9. Running a basic process service

9.1 About BPMN in BAMOE

BPMN nodes provide a variety of functionalities to automate business processes. Here's a quick overview of some key BPMN nodes and their purposes:

BPMN Node	Purpose	When to Use	Example
Script Task	Execute code within process	Quick, simple logic	Calculate a value in JavaScript
Service Task	Invoke external services or custom Java code	Integrate with other systems or process custom logic	Call a REST API, or trigger a logic in a java class
Human Task	Involve human interaction	Manual input/approval needed	Assign approval to manager
Timer Event	Time-based events	Scheduling, delays, timeouts	Wait 24h before reminder
Gateway	Control process flow	Decision points, parallel processing	Route high-value orders differently
DMN Task	Execute complex decision logic	automated decisions	Determine credit eligibility

9.1.1 NOTE ON AUTOMATIC MARSHALLING

BAMOE auto-maps process variables to DMN inputs/outputs by name and type. Here's a simplified explanation:

Automatic Marshalling in BAMOE:

The **basic principle** of this capability relies on BAMOE usage of Java reflection to convert POJOs (Plain Old Java Objects) to DMN data types and vice versa.

It can be seen when:

- When passing data from process to DMN: It maps POJO fields to corresponding DMN input fields.
- When receiving data from DMN to process: DMN output is converted back into the appropriate Java object.

> **Tip**
>
> Matching Criteria: Field names in the POJO should match DMN input/output names. Data types should be compatible (e.g., Java String to DMN string).

This automatic marshalling simplifies data exchange between BAMOE processes and DMN decision services, reducing the need for manual data transformation in many cases.

9.2 Cloning the Project and Running Quarkus Dev Mode

First, let's clone the starter project to your local repository and run it using Quarkus dev mode.

9.2.1 STEPS:

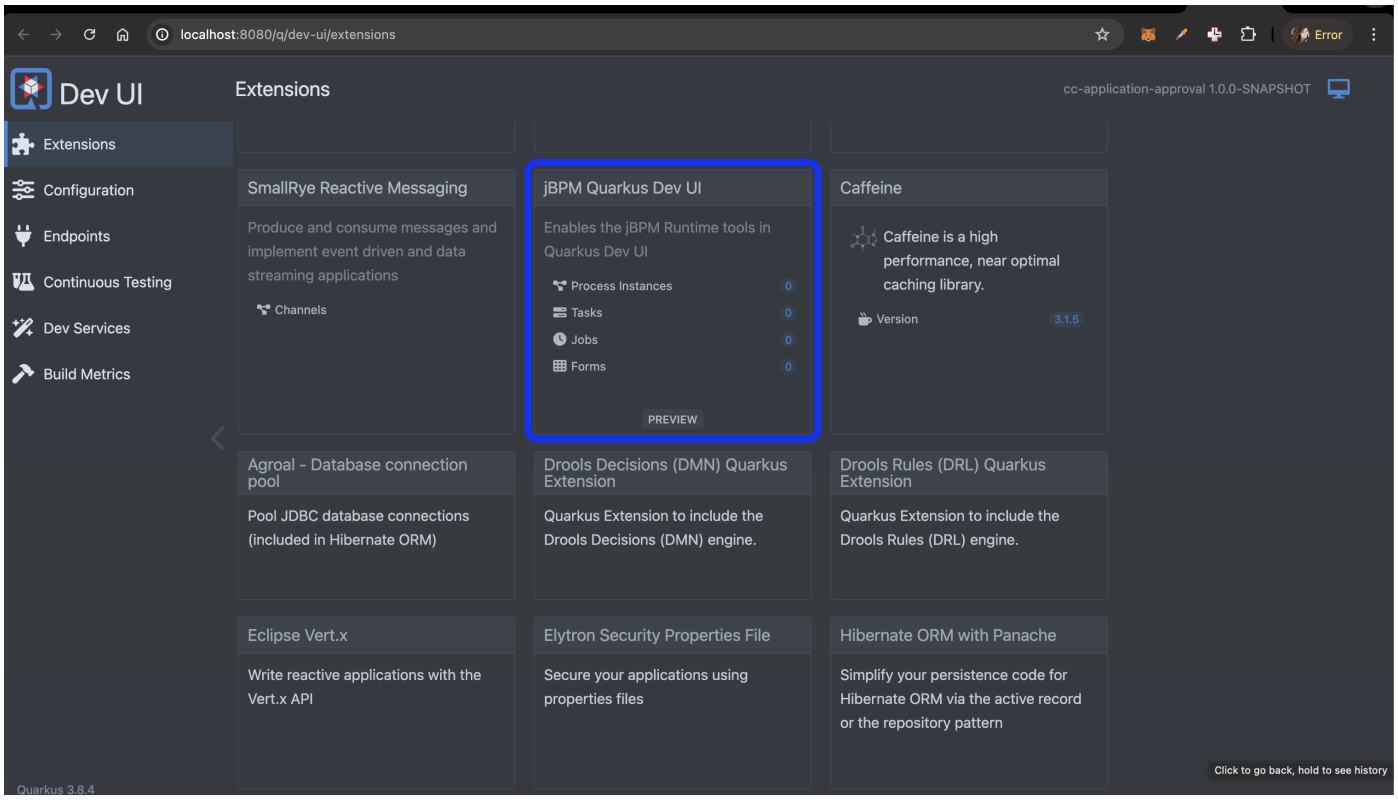
1. **Clone the Repository**:

```
git clone https://github.com/kmacedovarela/cc-application-approval-starter
cd cc-application-approval-starter
```

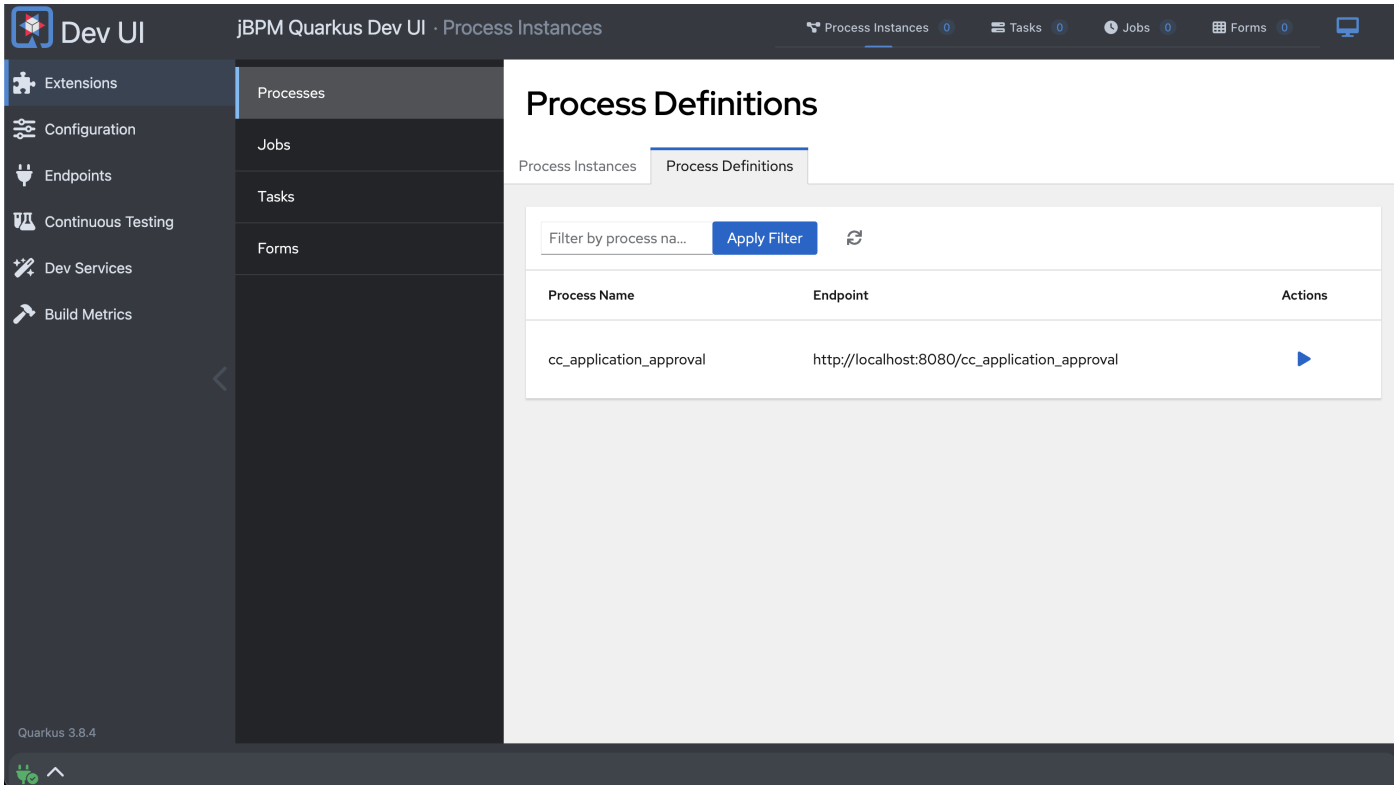
2. Run Maven Quarkus Dev Mode:

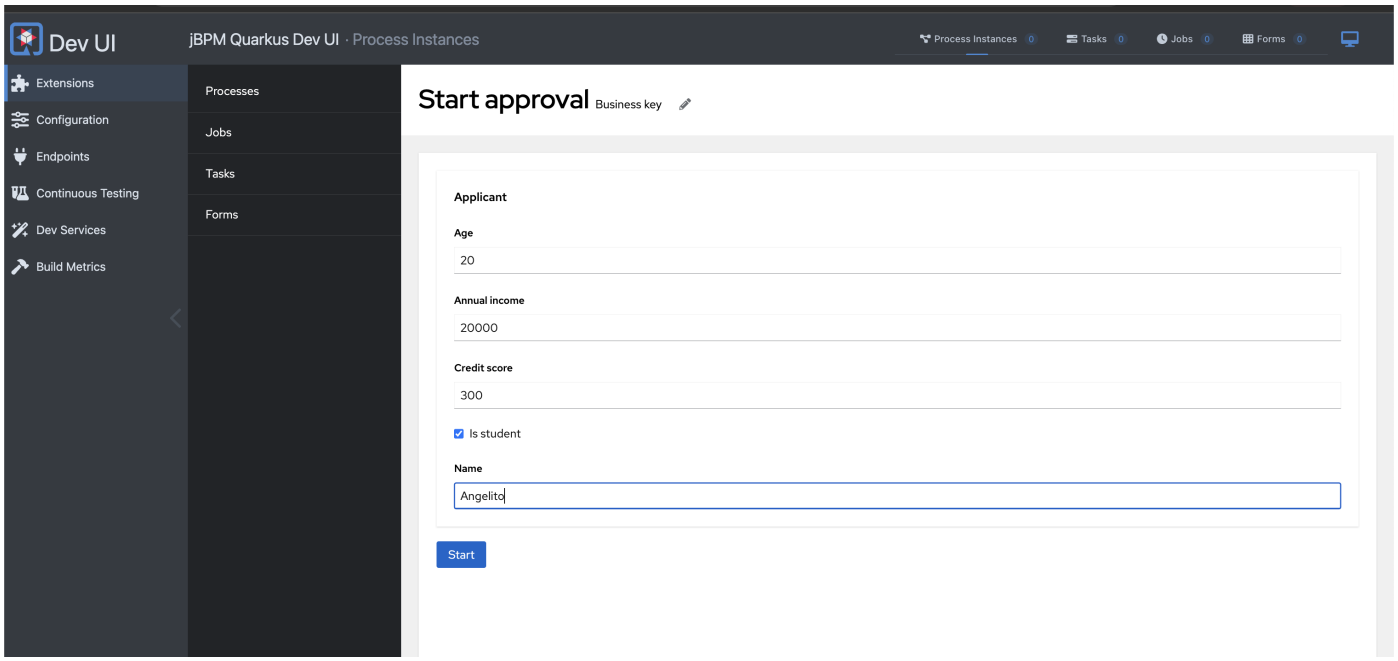
```
mvn quarkus:dev
```

3. Open the Browser and navigate to http://localhost:8080/q/dev-ui.
4. Locate and click on *Process instances* in the **jBPM Quarkus Dev UI add-on**.

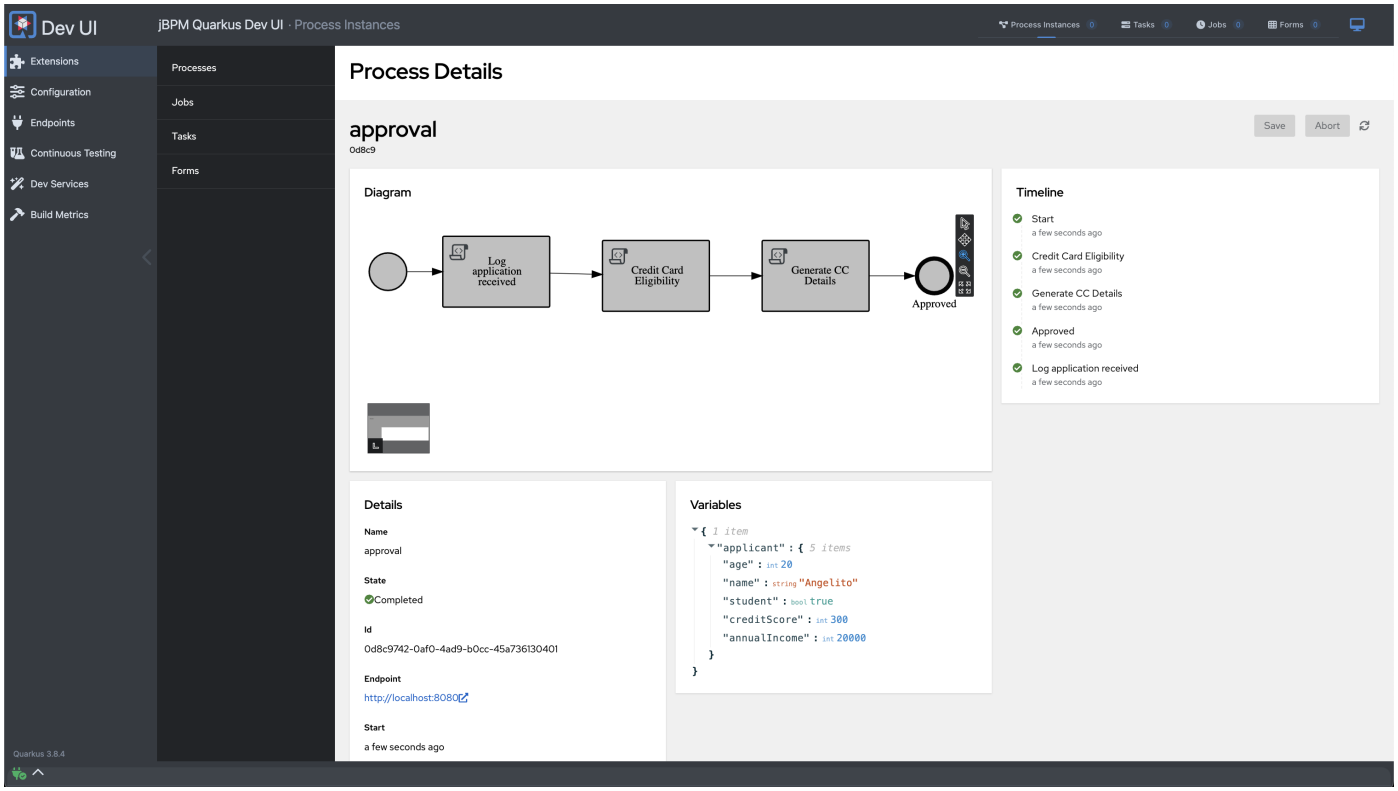


5. Start a new process from the process definition list.





6. Open the process details, and check the process instance progress, and the variable values.



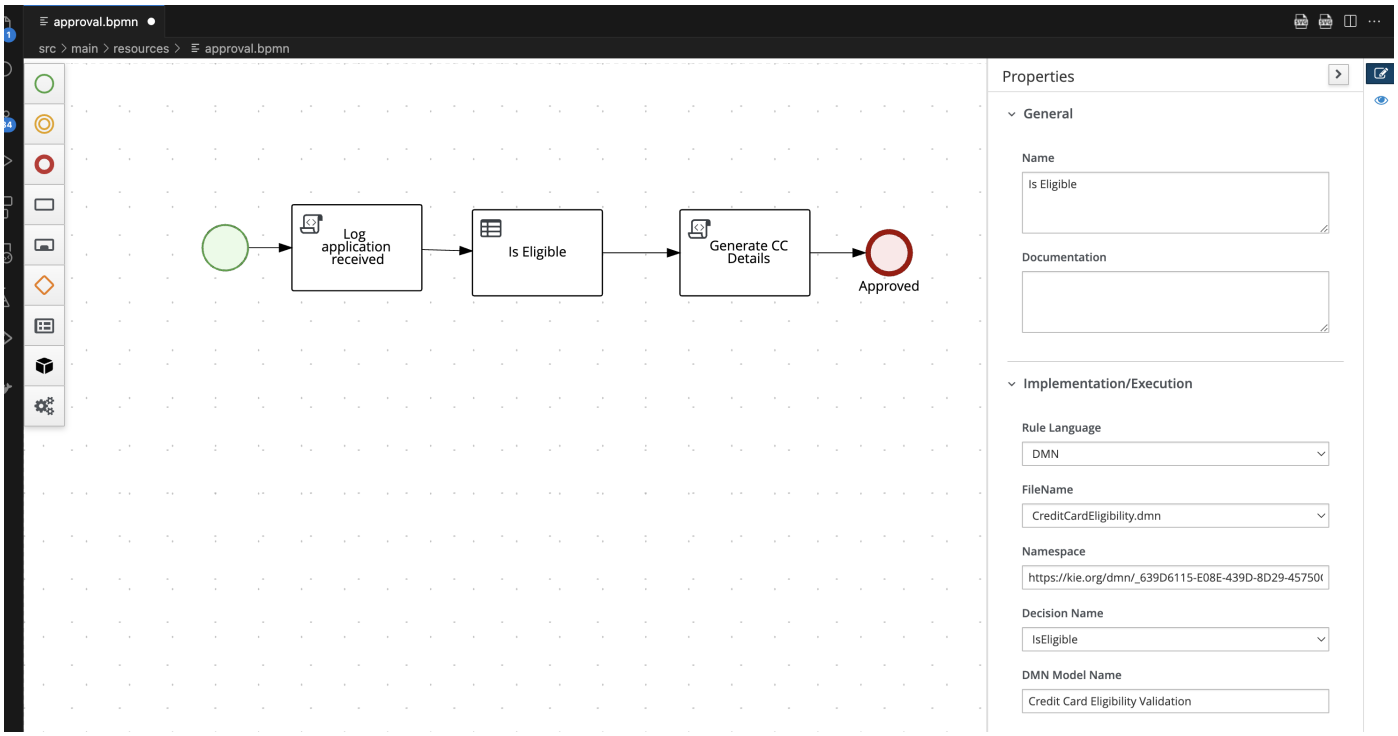
Awesome!! You've successfully cloned the project and started it using Quarkus dev mode. Next, let's explore how to extend the process design with various tasks in Canvas.

Last update: 2024-07-03

2.3.3 10. Using DMN Decisions in Processes

In this section, you will learn how to use DMN-based business rules task to automate decisions in your process, leveraging process variables and data types.

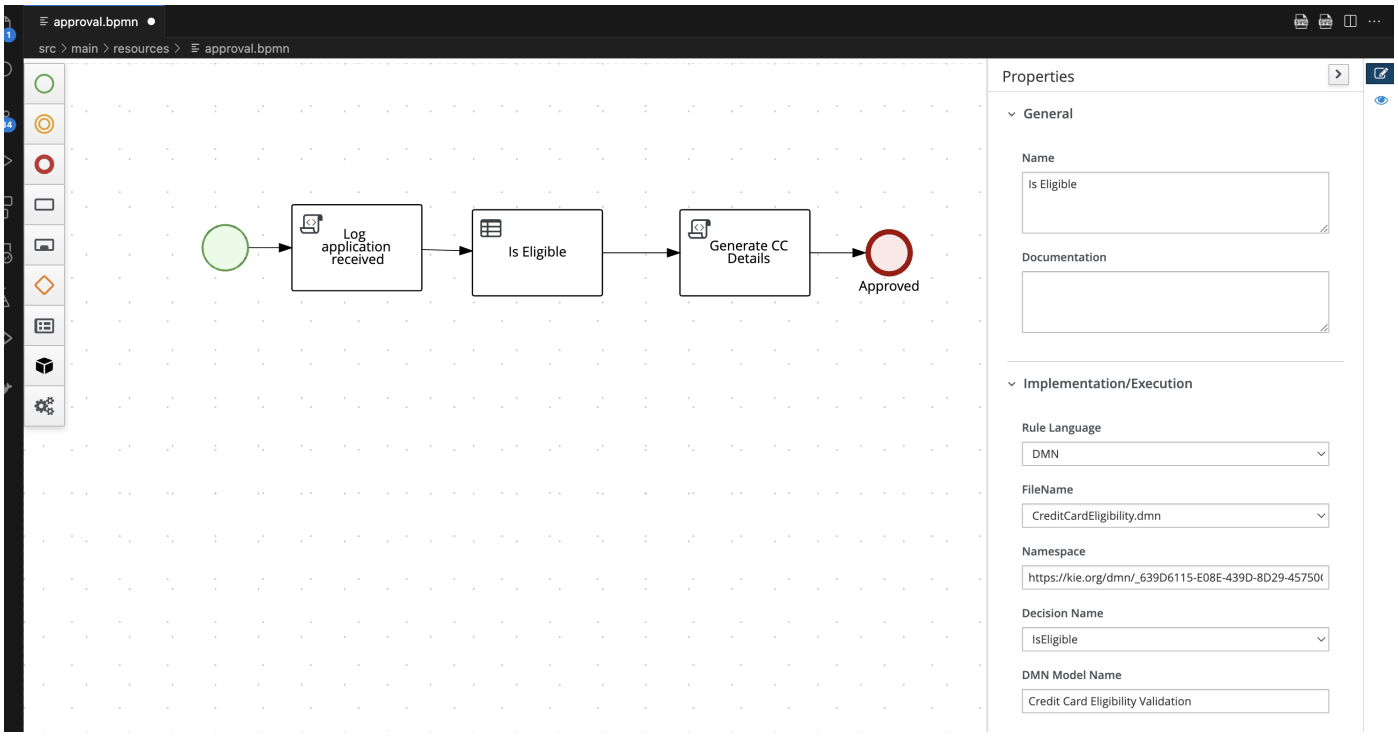
1. In VSCode, open the process `approval.bpmn`, and delete the script task named `Check Card Eligibility`.
2. Add a new **business rules task** to the process diagram, named **Is Eligible**.



10.1 Configuring the Business Rules Task

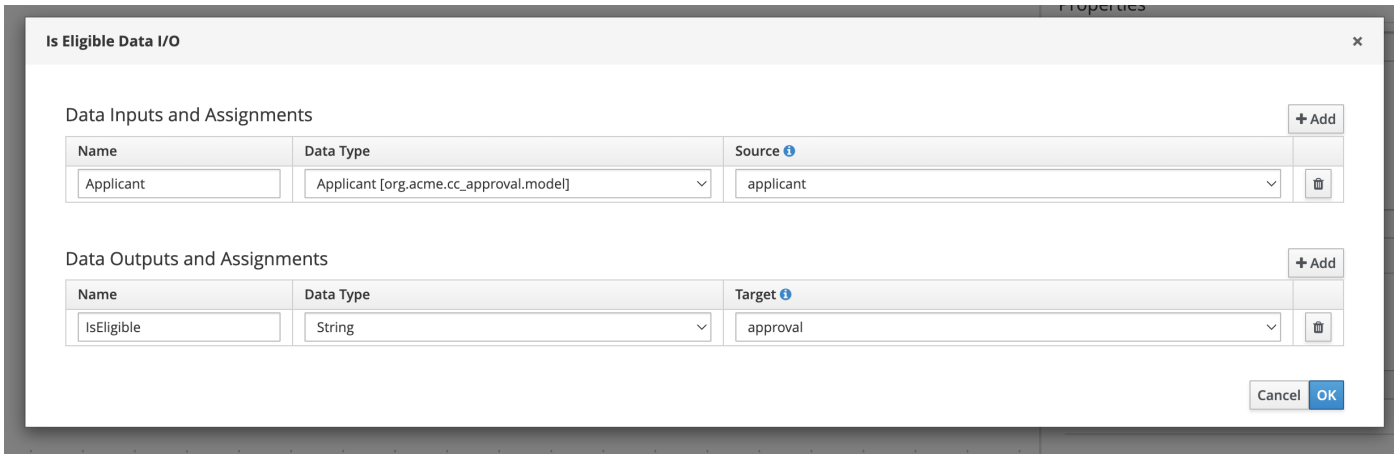
1. Configure the business rules task to consume the decision model. You can find and double check the information below, in the DMN file `CreditCardEligibility.dmn` available in your project:

```
Rule Language: DMN
Filename: CreditCardEligibility.dmn
Namespace: https://kie.org/dmn/_639D6115-E08E-439D-8D29-45750C32DB28
Decision Name: IsEligible
DMN Model Name: Credit Card Eligibility Validation
```



2. Configure the inputs and outputs of the task, by clicking on **Assignments** and using the following info:

Input: `Applicant`, `org.acme.cc_approval.model.Applicant`, `applicant` **Output**: `IsEligible`, `String`, `approval`



10.2 Validating the DMN Configuration

1. Open the DMN File and verify that the `Applicant` data type is part of the DMN model and matches the process variable.
2. Now, open the class `org.acme.cc_approval.model.Applicant` and check the attributes there. See the similarity with the data type?

The process variable `applicant` matches the data type in the DMN model, ensuring seamless data flow between the process and the decision model.

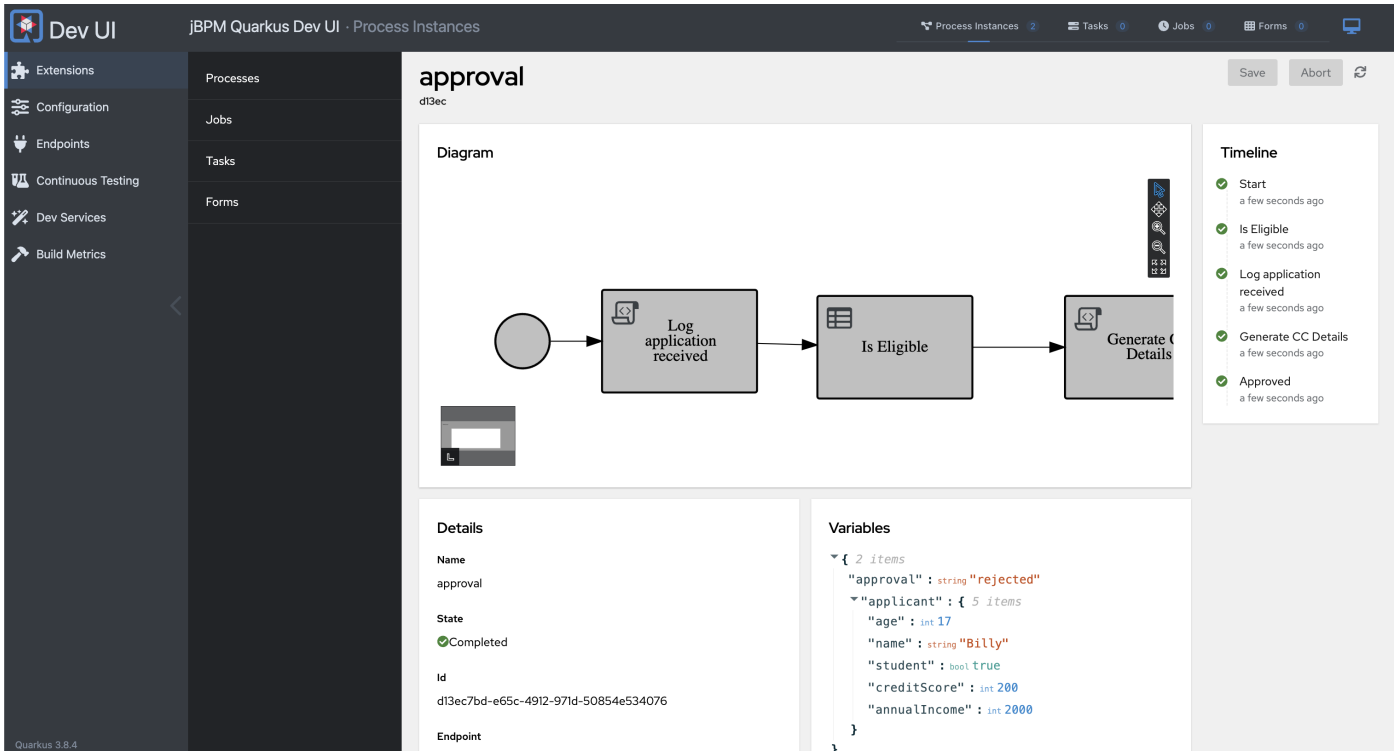
10.3 Running the Process with DMN Automation

1. Start Quarkus in Dev Mode:

```
mvn quarkus:dev
```

2. Open the Dev UI, navigate to `http://localhost:8080/q/dev-ui`.
3. Start a new process instance from the approval process definition.

4. Check Process Variables: Verify the variables of the completed instance to ensure the DMN decision was executed correctly.



5. These test scenarios help you validate different outcomes based on the input data for the DMN-based decision automation. Test with different data that will result on different outcomes in the decision.

Scenario	Is Student	Annual Income	Credit Score	Age
Automatic Approval	false	15000	750	25
Automatic Rejection	false	15000	750	17
Manual Review	false	30000	600	20

6. Stop Quarkus and close the dev ui in your browser.

10.4 Configuring Different Outcomes Based on Automated Decisions

1. Now, with the process working, let's add the gateways to handle the three possible scenarios: **Automatic approval**, **Automatic rejection**, **Manual Approval**.
2. Add an exclusive gateway after the decision node, and let's configure the possible outcomes:

   a. **Automatic Approval**: - For the sequence flow leading to the approval end event, use the condition:

```
return approval.toLowerCase().equals("approved");
```

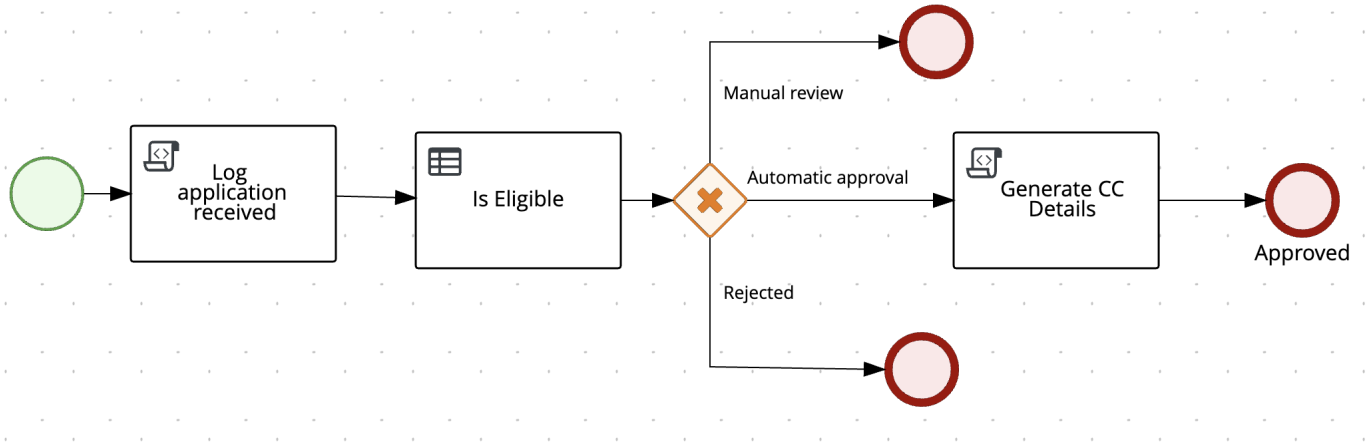
   b. **Manual Approval**: - For the sequence flow leading to the manual review end event, use the condition:

```
return approval.toLowerCase().equals("manual");
```

   c. **Automatic Rejection**: - For the sequence flow leading to the rejection end event, use the condition:

```
return approval.toLowerCase().equals("rejected");
```

3. Add three different paths from the gateway, each leading to an end event.



4. Connect each sequence flow to a respective end event.
5. Generate the SVG diagram in VSCode to visualize the process.
6. Run `mvn quarkus:dev` to start Quarkus in development mode.
7. Start a new process instance and test the three different scenarios (approved, manual, rejected).
8. Verify the instance details for each scenario.
9. Close the dev-ui and stop quarkus.

Awesome! You've successfully configured the process to handle different outcomes based on automated decisions. With the exclusive gateway and the conditions set for approval, manual review, and rejection, your process is now more dynamic and responsive. Next, let's proceed with using the service task to further enhance our process automation capabilities.

Last update: 2024-07-03

2.3.4 11. Custom Logic with Service Tasks

11.1 Adding a Service Task for Credit Card Generation

When a card application is approved, we need to generate the credit card details. We can use Java to process this generation, providing a more flexible and powerful way to create the card information.

11.1.1 UNDERSTANDING THE SERVICE CLASS

Open the `CreditCardService` class in your project. This class contains the logic for generating credit card details:

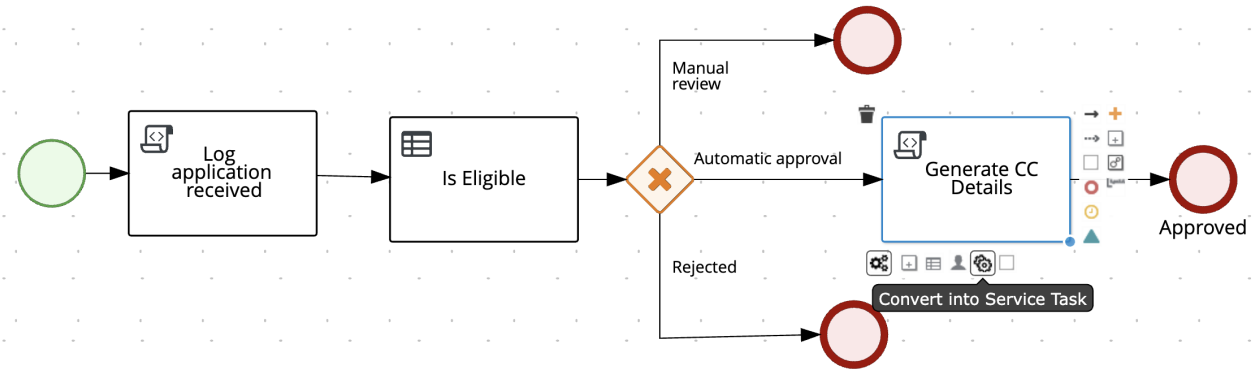
```
@ApplicationScoped
public class CreditCardService {
    public CreditCard generateCreditCardDetails(Applicant applicant) {
        double creditLimit = applicant.getAnnualIncome() * 0.3;
        return new CreditCard(applicant.getName(), creditLimit);
    }
}
```

This service calculates a credit limit based on 30% of the applicant's annual income and creates a new `CreditCard` object with the applicant's name and the calculated credit limit.

11.1.2 ADDING THE SERVICE TASK TO YOUR BPMN

You can incorporate custom Java code into your processes using service tasks. Here's how to add and configure a service task for credit card generation:

1. Open the approval process in VSCode.
2. Add a new service task on the approved path, after the existing gateway and before the end node.

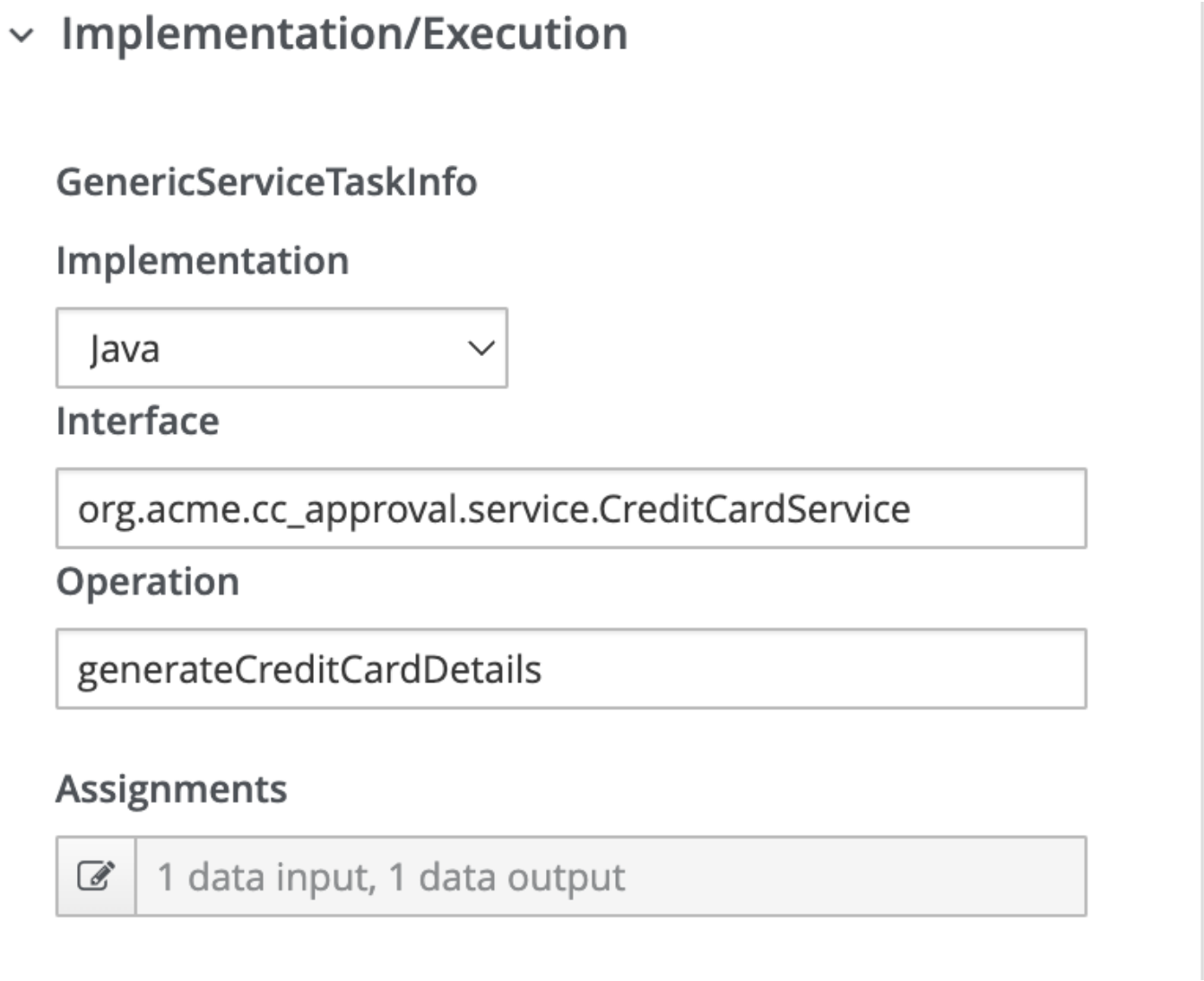

Log application received
Is Eligible
Manual review
Automatic approval
Rejected
Generate CC Details
Approved
Convert into Service Task

1. Name the new service task "Generate CC Details".
2. Configure the service task with these attributes:

• Implementation: `java`

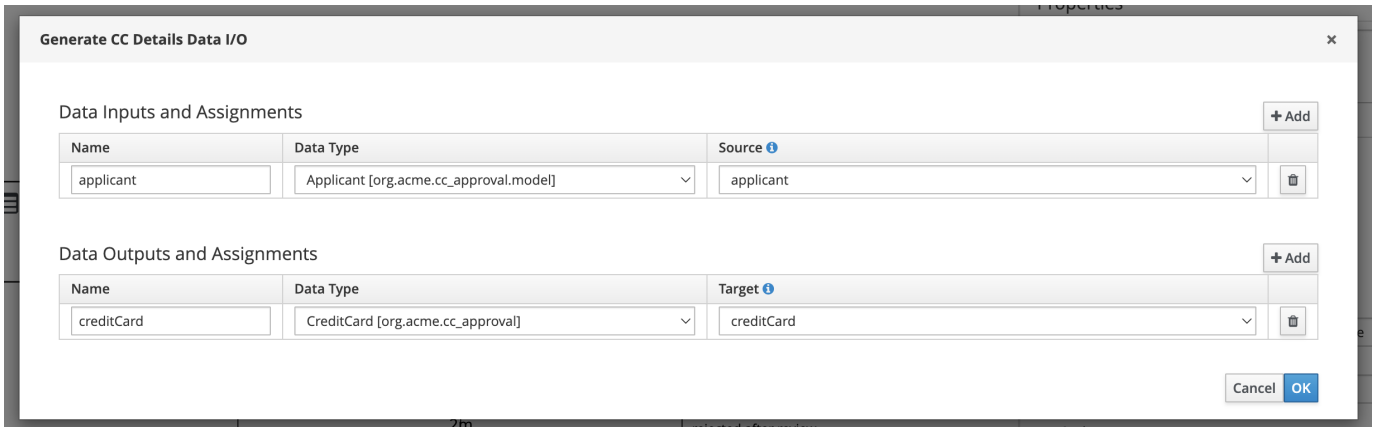
• Interface: `org.acme.cc_approval.service.CreditCardService`

• Operation: `generateCreditCardDetails`



The input assignment sends the process data as a parameter to the method, while the output assignment brings the method's return value back to the process and assigns it to a process variable.

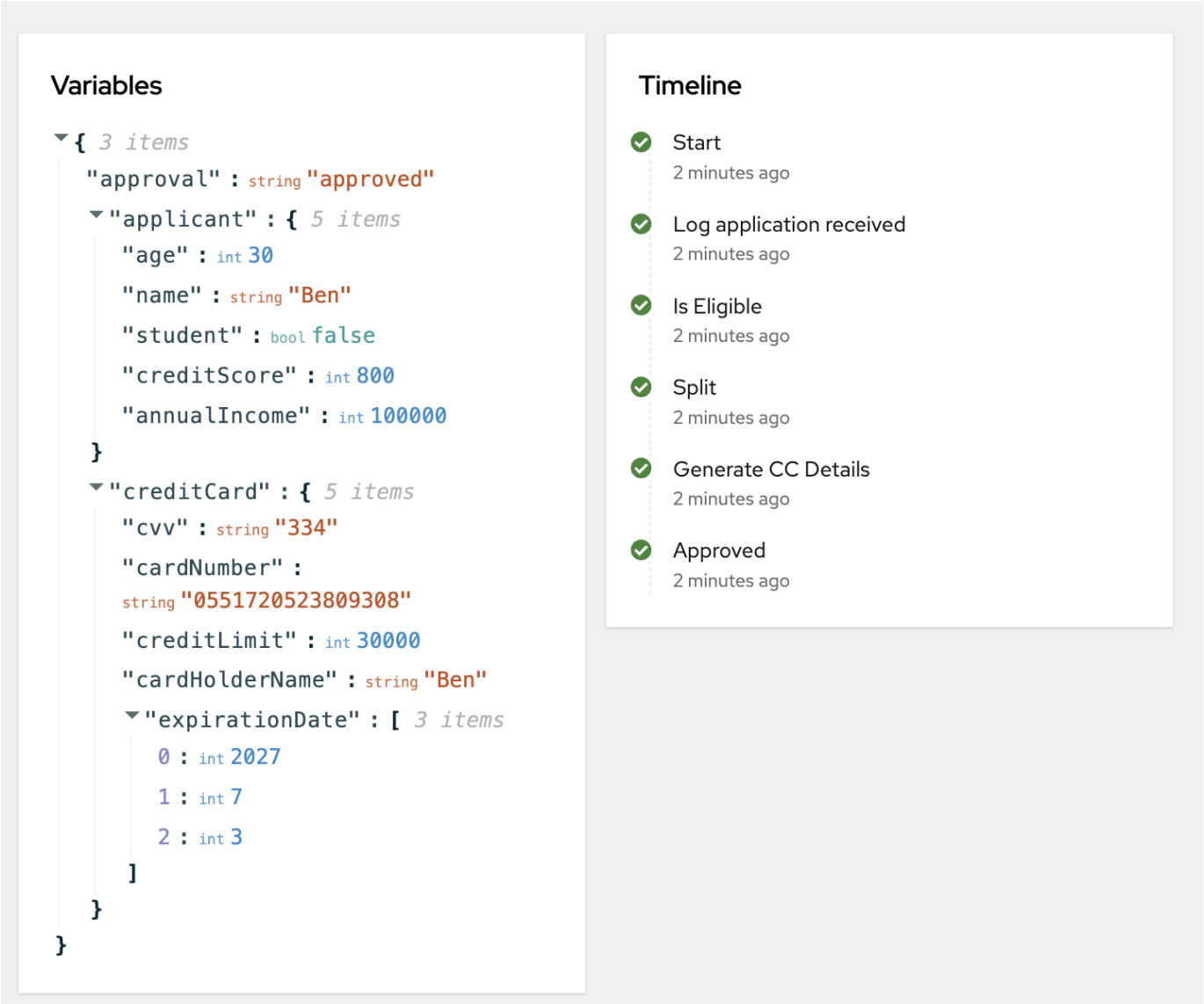
Assignments: - Input Assignment: - Name: `applicant` - Data Type: `Applicant` - Source: `applicant` - Output Assignment: - Name: `creditCard` - Data Type: `CreditCard` - Target: `creditCard



11.1.3 TESTING THE UPDATED PROCESS

To test your updated process:

1. Start Quarkus in dev mode using the command line.
2. Open the Quarkus Dev UI in your browser.
3. Start a new process instance with an approved application to see the card details in the instance data.
4. Examine the process instance details, paying particular attention to the generated credit card values.



1. After testing, close the Dev UI and stop Quarkus.

By following these steps, you've successfully integrated a Java service task into your BPMN process to generate credit card details upon approval. This demonstrates how you can leverage custom Java code to enhance your business processes with complex logic and data manipulation.

Last update: 2024-07-03

2.3.5 12. User tasks for manual approval

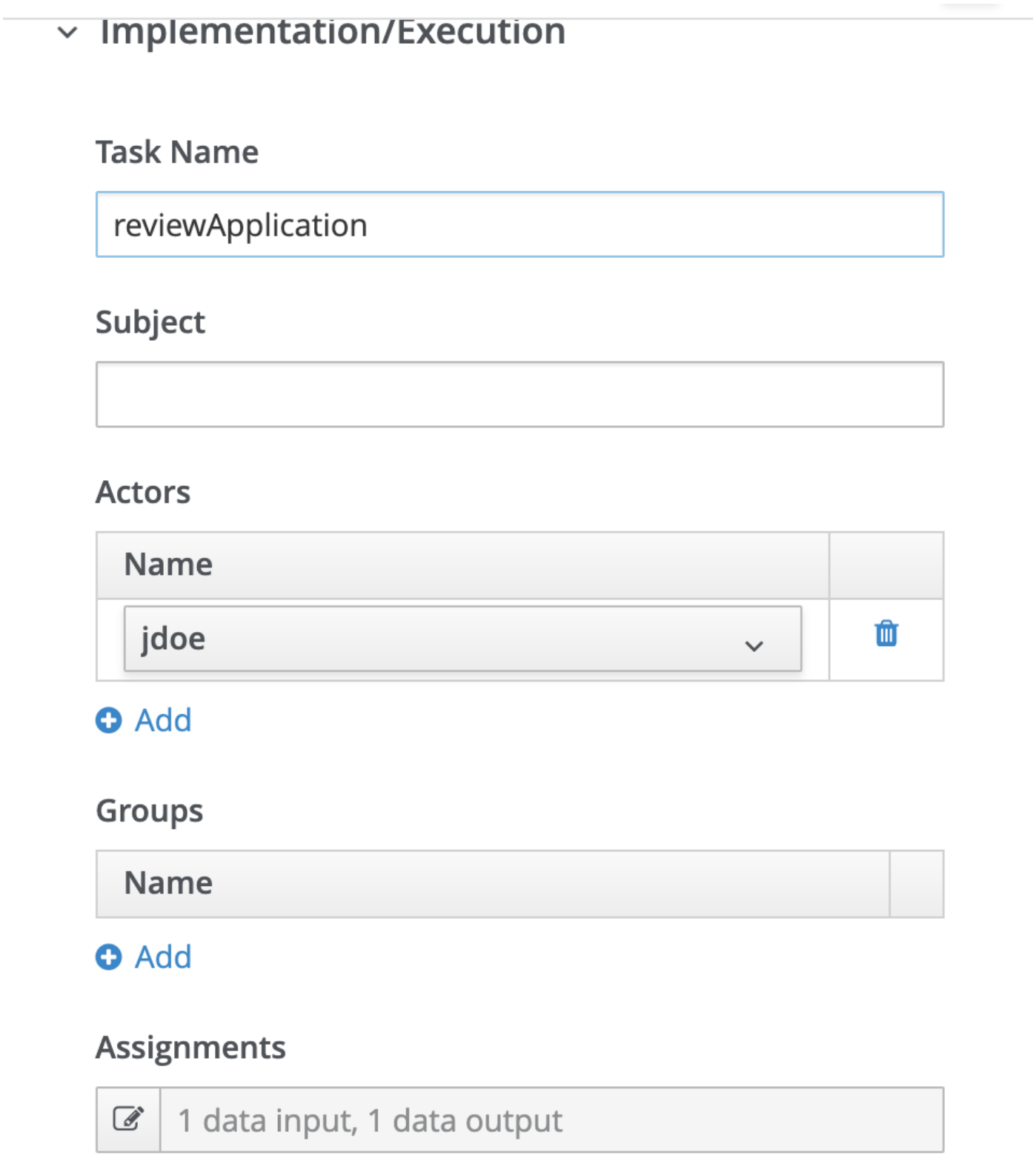
When the decision outcome indicates the need of a manual approval, the flow should move forward to a user task. Let's add this manual step in the process, and see it in action.

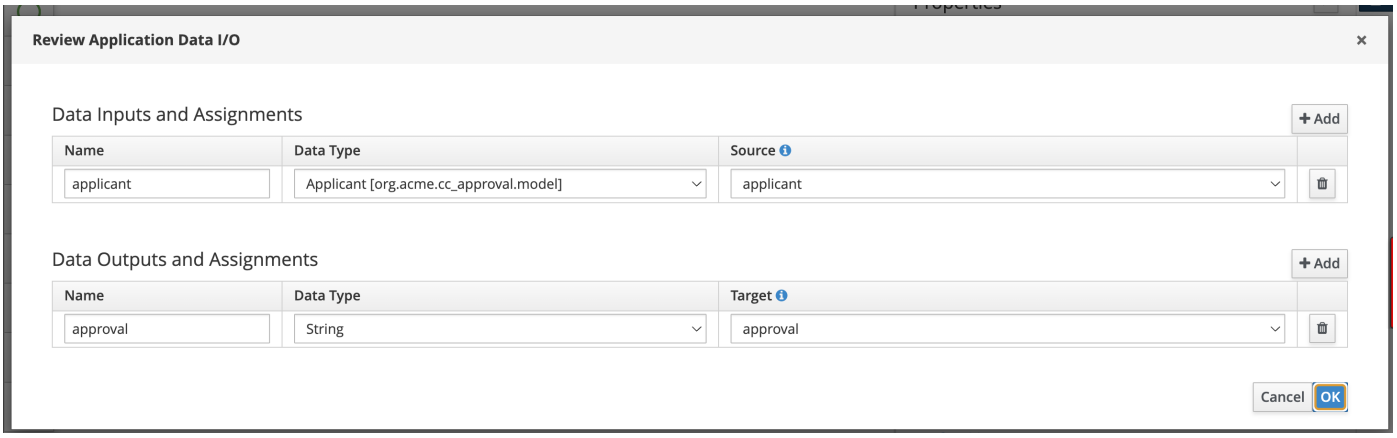
About the process: - When an application requires manual review, it now goes to the "Review Application" task. - The assigned user (jdoe) reviews the application and decides to approve or reject it. - Based on the decision (stored in the `approval` variable), the process takes one of two paths: 1. If approved, it proceeds to generate credit card details. 2. If rejected, it ends the process immediately.

12.1 Manual approval

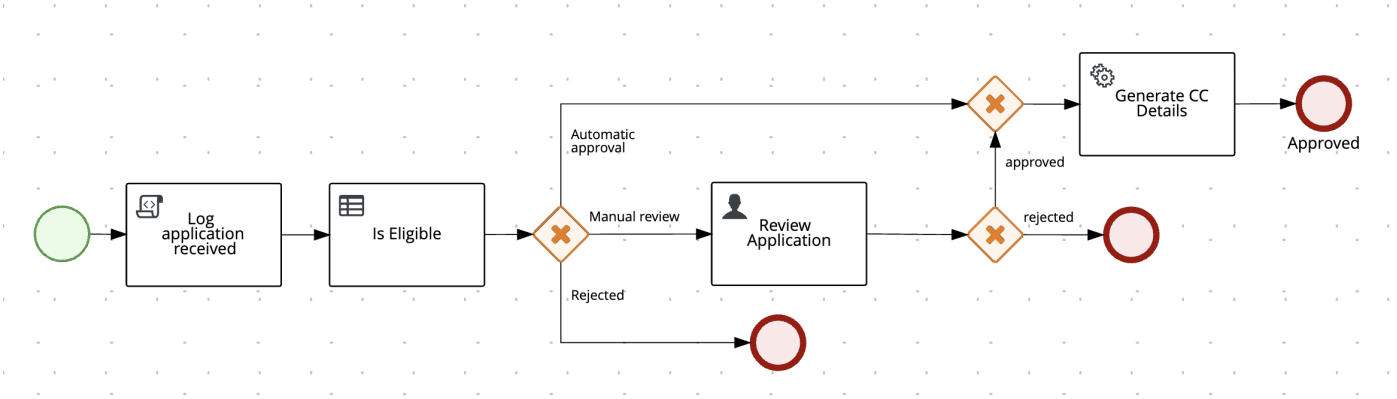
1. Add a new user task on your process, on the "manual approval" path.
2. Configure the task with the name "Review Application", and:
3. Set the Actor to: `jdoe`

- Input Assignment:
- Name: `applicant`
- Data Type: `Applicant`
- Source: `applicant`
- Output Assignment:
- Name: `approval`
- Data Type: `String`
- Target: `approval`


Implementation/Execution
Task Name
reviewApplication
Subject
Actors
Name
jdoe
Add
Groups
Name
Add
Assignments
1 data input, 1 data output



4. Now, re-model your process so that it looks like this:



5. You can configure the sequence flows after the human task such as:

6. For the "approved" path: - Set the condition to: `return approval.toLowerCase().equals("approved");`

7. For the "rejected" path:

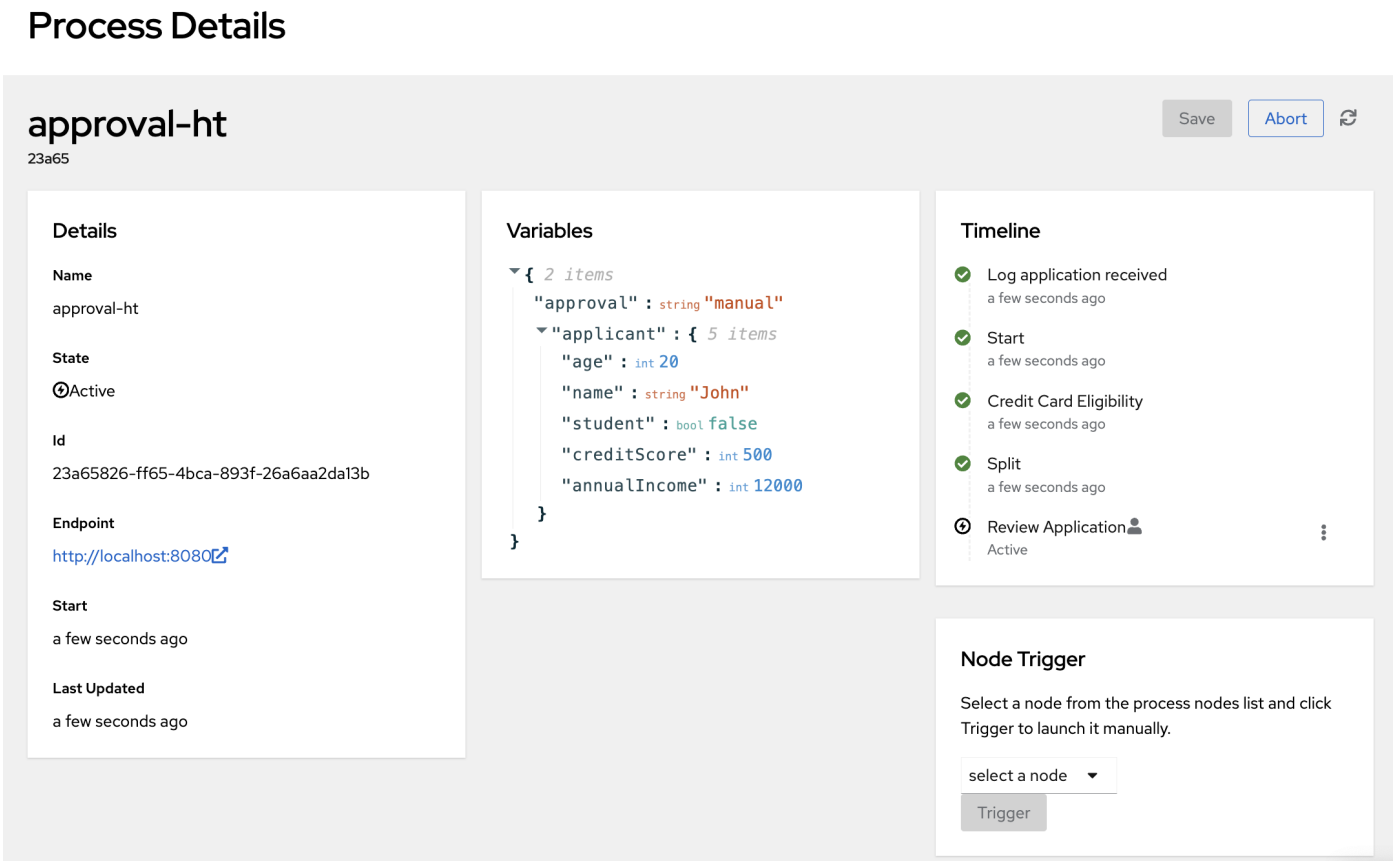
• Set the condition to: `return approval.toLowerCase().equals("rejected");`

12.2 Testing the Updated Process

1. Start Quarkus in dev mode.

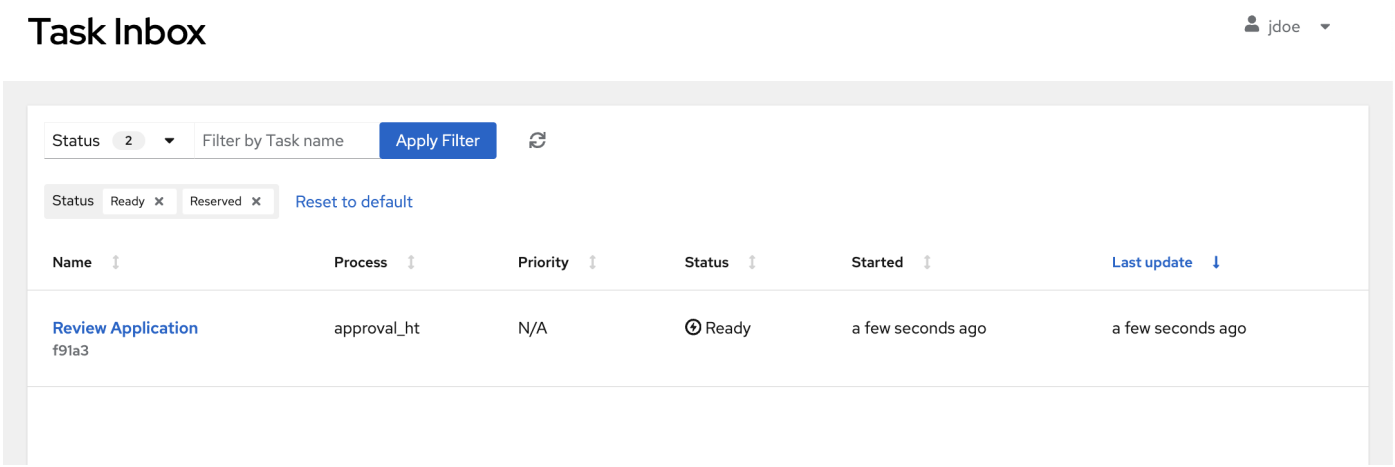
2. Open the Quarkus Dev UI in your browser.

3. Start a new process instance that would route to manual review.



4. You should see a task assigned to user "jdoe".

5. Switch to "jdoe" and complete the task, choosing either "approved" or "rejected".



6. Observe how the process follows the appropriate path based on your decision.

7. If approved, verify that credit card details are generated.

8. If rejected, confirm that the process ends without generating card details.

By adding this human task, you've introduced a crucial manual review step into your process, allowing for human decision-making in complex or borderline cases. This enhances the flexibility and robustness of your credit card approval process.

Last update: 2024-07-26