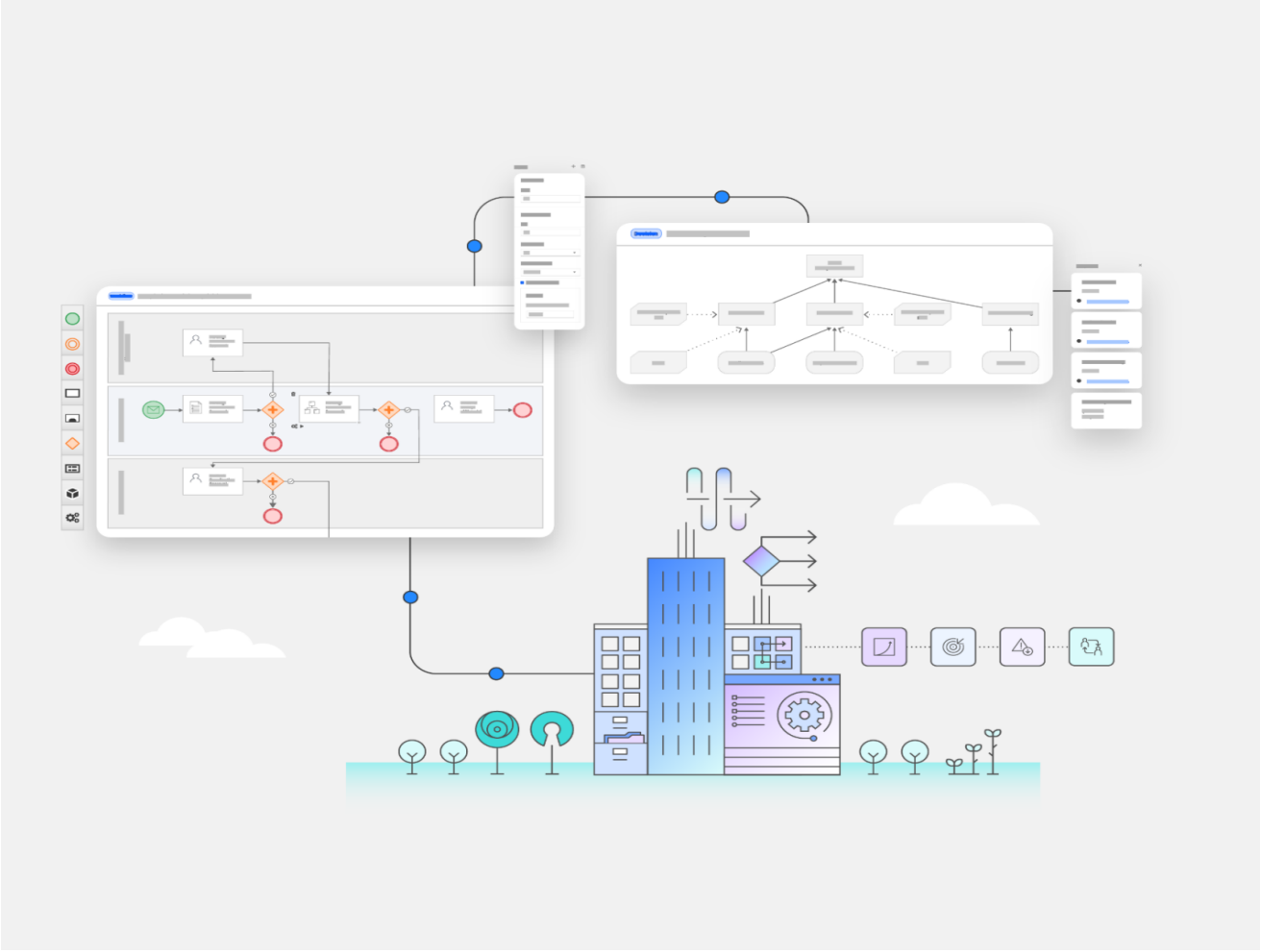
IBM Business Automation Manager Open Editions

Lab Environments
W420G - IBM BAMOE Skills Academy



Barry Lulas
barry.lulas@ibm.com
Principal Technical Sales Leader, Business Automation SME, Americas

Jason Porter
jporter@ibm.com
Software Engineer

Document edition: March 2025

Table of Contents

Introduction

Welcome to the **IBM BAMOE Skills Academy**. The labs included in this course will cover using **IBM Business Automation Manager Open Edition**'s capabilities for decision and process automation, including guidance on how to build complete decision or process services, as well as deployment and execution on Red Hat OpenShift.

IBM Business Automation Manager Open Editions (IBM BAMOE) is a cloud-native business automation technology for building cloud-ready business applications. It is built from various open-source projects including Drools, jBPM, and most notably Kogito. The letter *"K"* in Kogito refers to Kubernetes, the base for Red Hat OpenShift, as the target cloud platform for IBM BAMOE and the Knowledge is Everything (KIE) open-source business automation project for which IBM BAMOE originates.

IBM BAMOE is optimized for a hybrid cloud environment and adapts to your domain and tooling needs. The core objective of IBM BAMOE is to help you mold a set of business processes and decisions into your own domain-specific cloud-native set of services.

When you use IBM BAMOE, you are building a cloud-native application as a set of independent domain-specific services to achieve some business value. The processes and decisions that you use to describe the target behavior are executed as part of the services that you create. The resulting services are highly distributed and scalable with no centralized orchestration service, and the runtime that your service uses is optimized for what your service needs.

What you will need to access

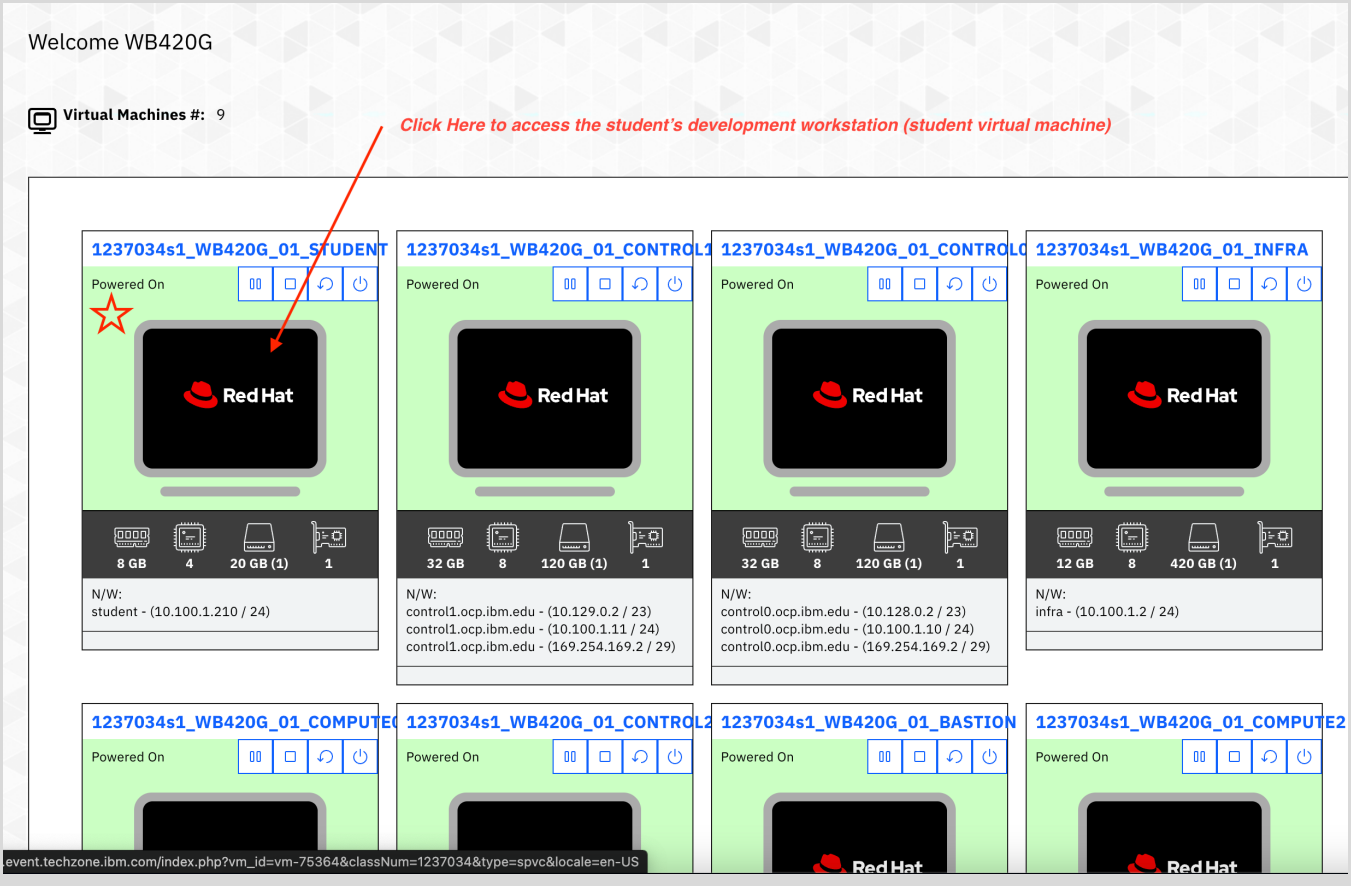
This lab utilizes two environments:

- **IBM BAMOE Developer Workstation** which provides the student with all the necessary tools and technologies for building, testing, executing, and deploying IBM BAMOE applications. This environment is based on Red Hat Linux v9 and represents a common developer workstation for IBM BAMOE. Students will have direct access to the OpenShift cluster from the Student VM, as well as have administrative console access via a browser and command-line interface.

- **Red Hat OpenShift Cluster** which provides infrastructure services in support of IBM BAMOE applications. This environment is based or Red Hat OpenShift and represents a common target execution environment for IBM BAMOE. The student will not necessarily need to login to this environment, as they have access via the OpenShift command-line interface or the web-based OpenShift administrative console.

*We will use the following environments for the Student's Developer Workstation (**please see the environment assignment sheet provided by the instructor for the environment that you are assigned. The assignment sheet is also located in the bamoe-skills-academy-documents folder under the environments folder**). Be sure to select the **WB420_STUDENT** virtual machine from the list of available virtual machines of this environment.*

Important: We will use the following account information to login and use the student's virtual machine. Please see the environment assignment sheet, provided by the instructor. Be sure to select the *WB420G_XX_STUDENT virtual machine, from the list of other virtual machines when you visit the assigned environment URL, as seen in the following example:



Virtual Machine Accounts

Once you have successfully accessed your assigned student virtual machine, the following table depicts the user accounts and passwords used in the lab environments:

Table 1: User IDs and passwords for your lab environment

VM name	Account	Password	Comment
Student Virtual Machine (Developer Workstation)	ibmuser	Passw0rd	BAMOE Developer Workstation (RHEL9)
	Root	1l0veibmrh	Use sudo
Lab OpenShift Cluster Administration	ocadmin	ibmrhocp	default-route-openshift-image-registry.apps.ocp.ibm.edu

Important: From time to time, you may be requested to enter the student's account password in various tools, such as VS Code or Google Chrome. If that happens, simply use the student virtual machine account password, noted above.

High Level Lab Architecture

The following diagram depicts the high-level architecture as it relates to typical IBM BAMOE application life-cycle management. Students, playing the role of **developer** or **modeler**, will utilize the **Student Virtual Machine**, which provides a typical example of how a developer machine would be configured to develop BAMOE applications. Applications built during the labs will be automatically deployed to the **Lab OpenShift Cluster**, which provides a typical example of how an OpenShift or Kubernetes cluster would be configured to deploy, test, and execute BAMOE applications. Positioned in the middle of the developer workstation and the OpenShift cluster, is the enterprise Git repository management system, for which all IBM BAMOE applications are published and utilized.

The developer workstation is installed with typical Java-based development tools, such as the VS Code IDE, with the IBM BAMOE Developer Tools extensions already installed. In addition, supporting tools and technology, such as the Java Development Kit v17, Git command-line interface, Maven command-line interface, and the OpenShift command-line interface all provide the developer with the necessary tools to build, test and deploy IBM BAMOE applications.

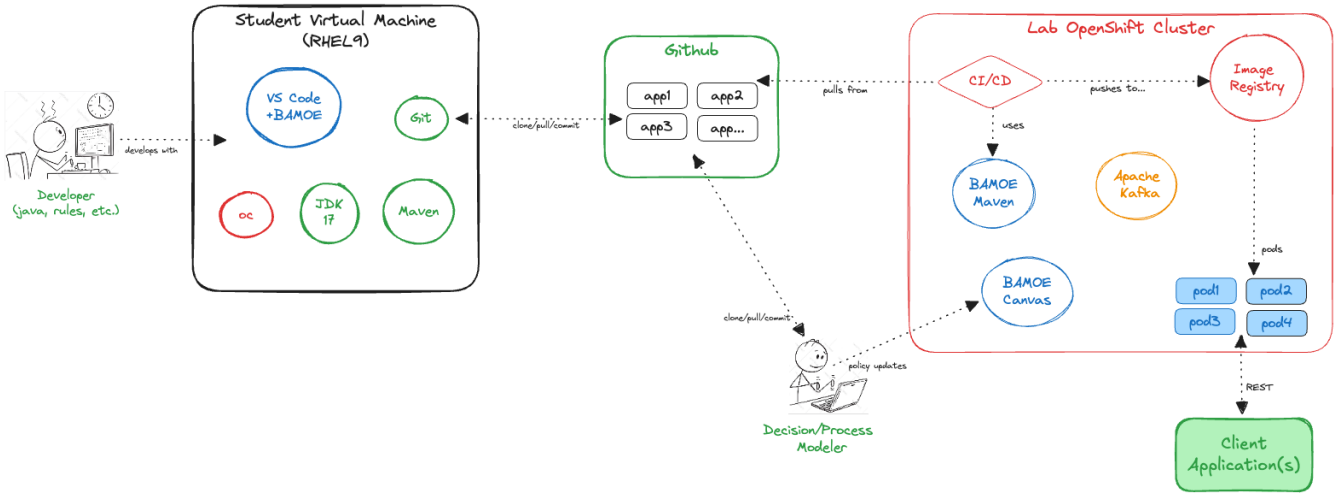
The OpenShift cluster is configured with all necessary infrastructure services that IBM BAMOE needs. In the case of Decision Automation projects, those services include the BAMOE Maven Repository, BAMOE Canvas, as well as supporting infrastructure services such as Apache Kafka (*for event-based applications*). All IBM BAMOE applications are deployed to the OpenShift cluster as container images and automatically create all OpenShift objects, such as deployments, pods, services, and routes. Images are pushed to the OpenShift container image registry, from either the developer workstation or a CI/CD pipeline, using standard Maven commands.

All IBM BAMOE projects are standard Maven projects and are stored in an accessible Git repository. Tools such as the BAMOE Developer Tools or BAMOE Canvas access these Maven projects directly from the tool, simply by using standard Git commands.

Lab Architecture for Decision Automation

The following diagram represents the typical architecture for decision automation applications, using IBM BAMOE:

Figure 1: Lab architecture for Decision Automation applications

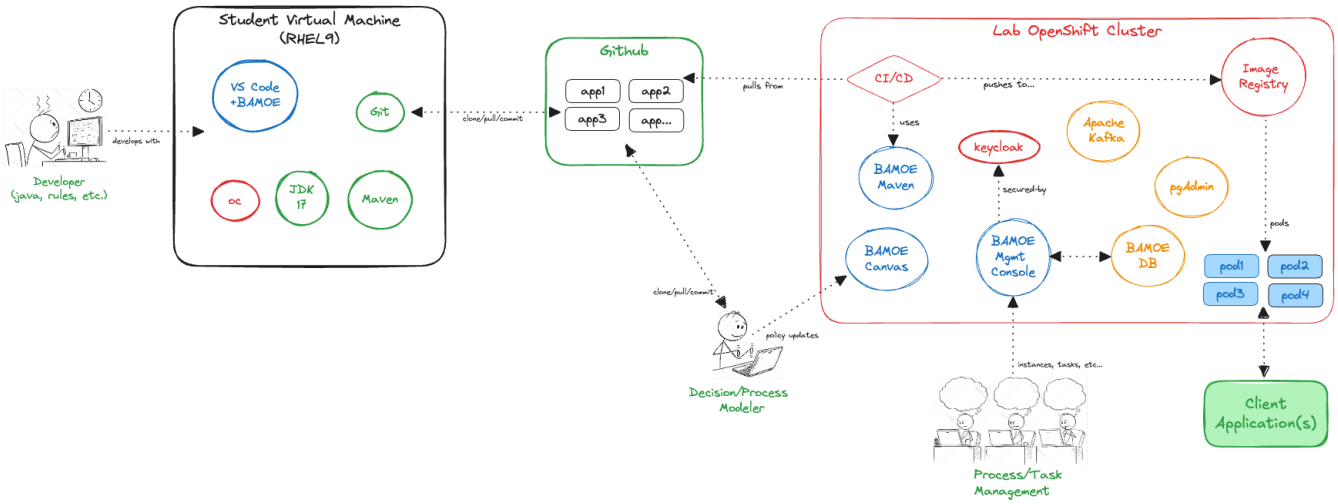


The decision automation labs described in this document focus on how to create BAMOE projects, add business automation resources, including technical rules written in Drools Rule Language (DRL), decision models written in Decision Modeling & Notation (DMN), and orchestrated with stateless workflow written in Business Process Modeling & Notation (BPMN). You will learn how to configure the features of each decision service through adding various Maven dependencies and updating the service's property setting for each target deployment profile. You will also learn how to deploy and execute decision services as standalone applications or as container images on Docker and Kubernetes. Finally, you will learn how to properly test your decision services.

Lab Architecture for Process Automation

The following diagram represents the typical architecture for process automation applications, using IBM BAMOE:

Figure 2: Lab architecture for Process Automation applications



The process automation labs described in this document focus on how to create BAMOE projects, add business automation resources, including technical rules written in Drools Rule Language (DRL), decision models written in Decision Modeling & Notation (DMN), and orchestrated with stateless or stateful workflow written in Business Process Modeling & Notation (BPMN). You will learn how to configure the features of each process service through adding various Maven dependencies and updating the service's property setting for each target deployment profile. You will also learn how to deploy and execute process services as standalone applications or as container images on Docker and Kubernetes. Finally, you will learn how to properly test your process services.

Lab Environment Setup and Basic Usage

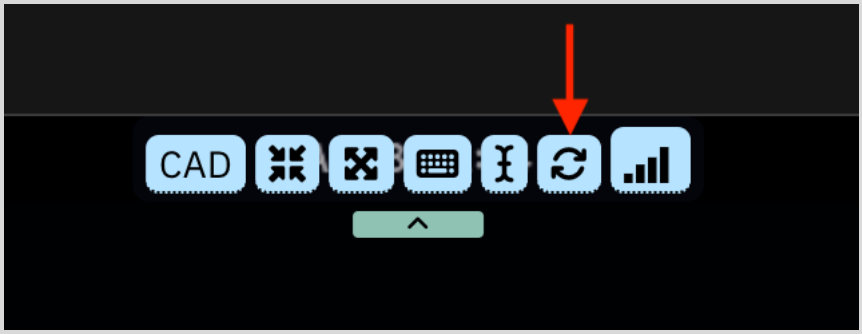
The lab environment representing the Developer Workstation (*student virtual machine*) has been pre-configured with all student lab guides and lab project repositories. However, there may be situations where you need to refresh these artifacts, as instructors may be required to make minor updates during the delivery of the course, and the instructor will show you how to do that in the first class session.

Access the Developer Workstation (Student Virtual Machine)

The Developer Workstation is a virtual machine configured as a Red Hat Linux v9 workstation and made available to students through their own laptop's browser. Once you login, you will see the following:

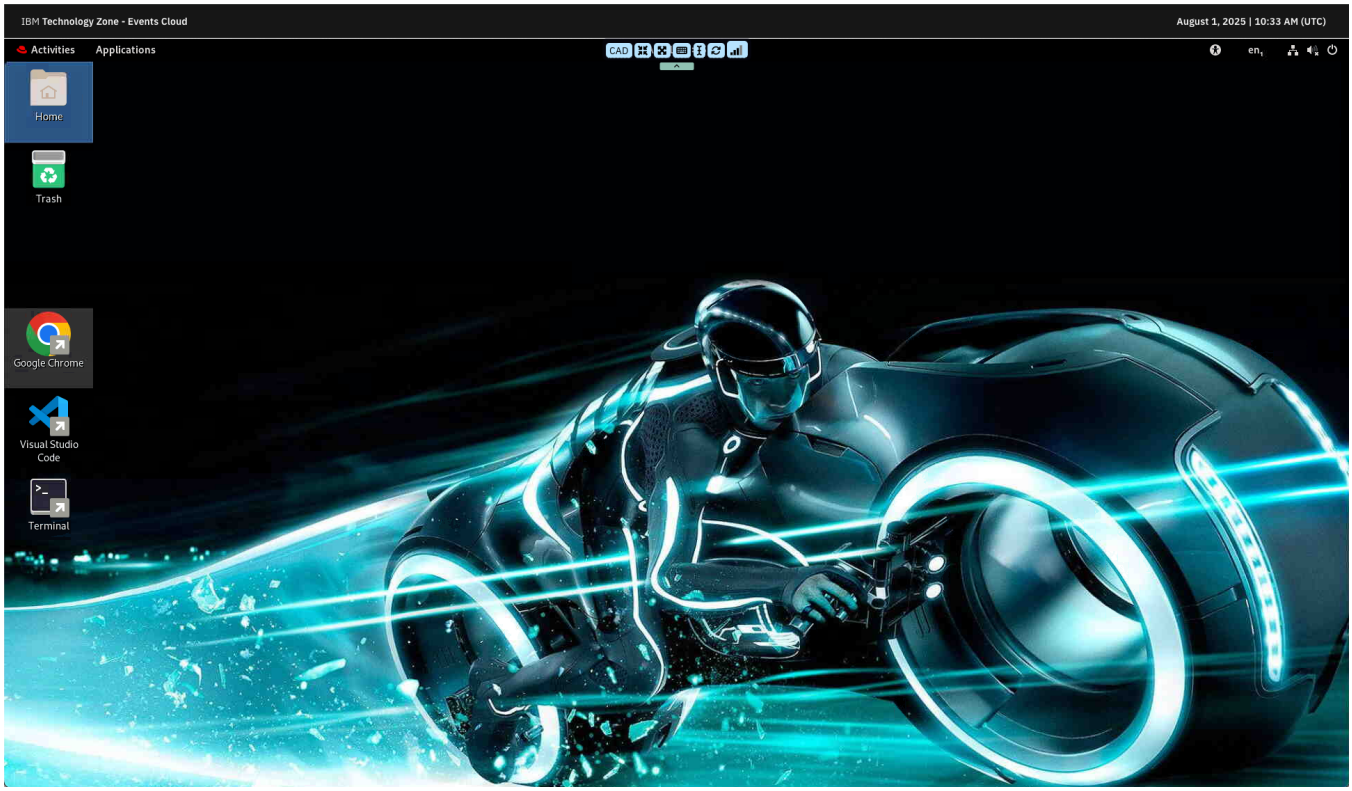
Important: While any browser will work for accessing the student virtual machine from the student's personal laptop, is it recommended to use Google Chrome. The student virtual machine comes pre-configured with Google Chrome, which is one of the recommended browsers for IBM BAMOE Canvas and IBM BAMOE Management Console.

It is also important to note that the performance of the browser-based access to the virtual machines can at sometimes become a little slow. You may need to refresh the desktop occasionally, especially if you get into situations where mouse clicks and typing is slow. At the top of the virtual machine window is a toolbar, show below. **Click the icon directly below the red arrow**, show below, to refresh the virtual machine:



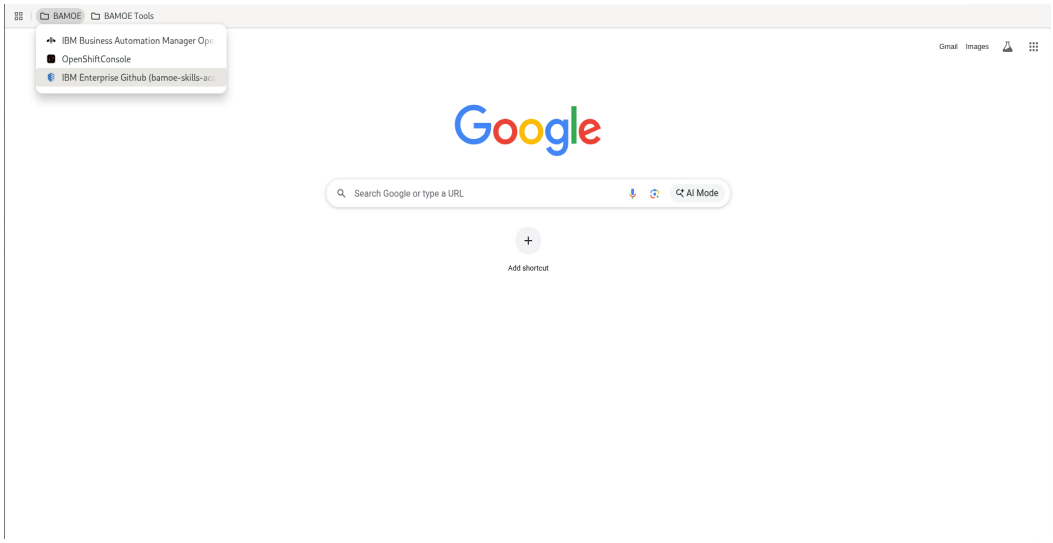
From here, your virtual machine desktop should look like the following:

Figure 3: Developer Workstation (student virtual machine) Desktop

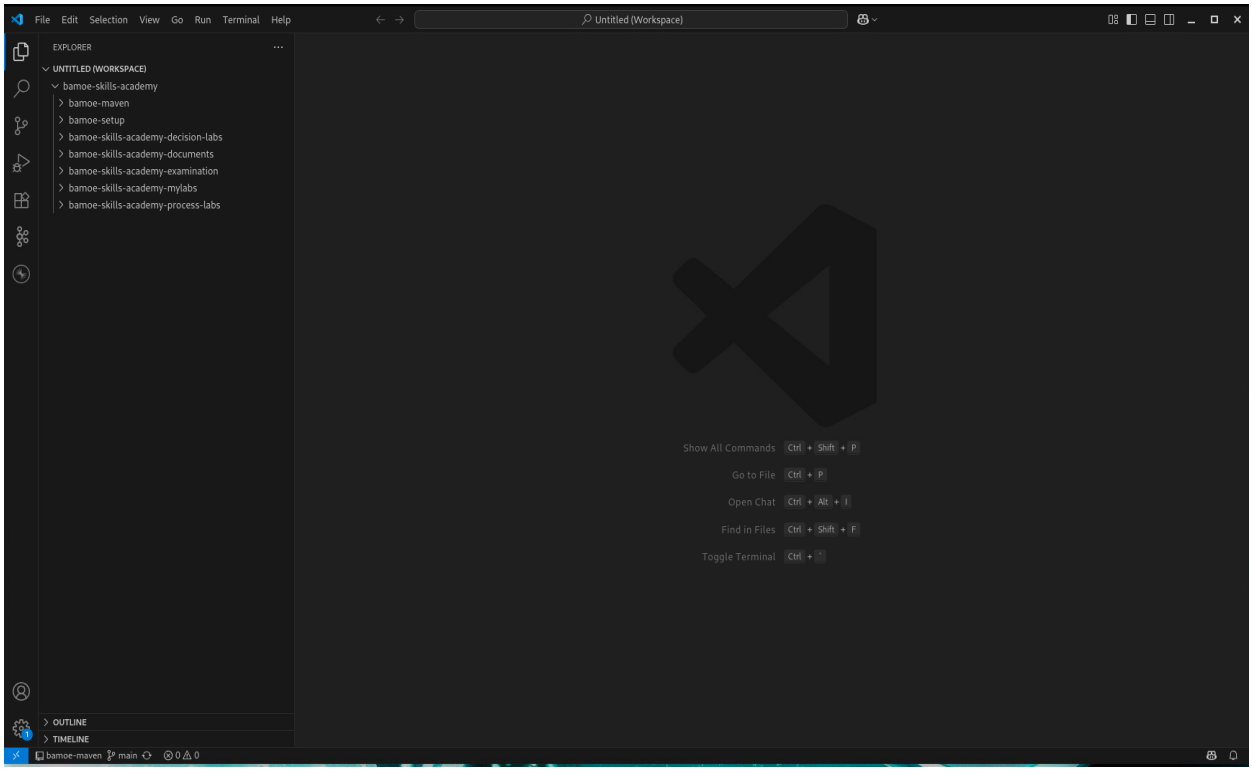


There are three main desktop shortcuts available to the student:

- **Google Chrome**, which is used to access web-based tools such as IBM BAMOE Canvas, IBM BAMOE Management console, pgAdmin (database) console, and the OpenShift administration console. Please see the provided Google Chrome bookmarks for access to these applications.



- **Microsoft Visual Studio Code (VS Code)**, which is the IDE used for IBM BAMOE project development. The IBM BAMOE extensions for BPMN, DMN, and Test Scenarios are already installed into VS Code, as well as extensions for Kafka and testing REST API endpoints. In addition, all lab repositories are already added to the VS Code workspace.

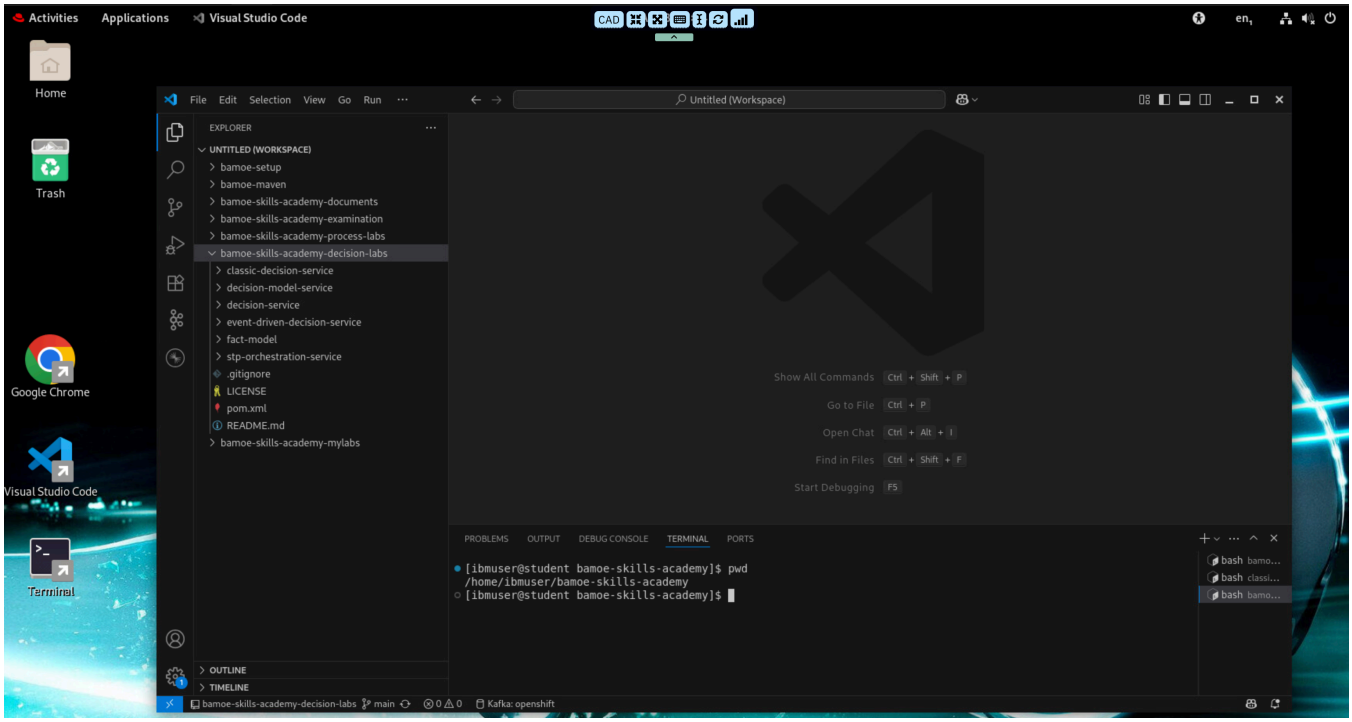


- **Terminal Window**, which is used for issuing Maven and Git commands on IBM BAMOE projects. You can also use the Terminal Window capabilities of VS Code vs using the desktop terminal window.

Build and Deploy the Decision Automation Labs to OpenShift

Once you have accessed your student environment, the next step is to build and deploy the pre-supplied decision automation labs to the student's OpenShift cluster. We will leave the process automation labs until Day 3, as they require a little more discussion to understand. All student learning material has already been downloaded to the /home/ibmuser/bamoe-skills-academy folder on the student's virtual machine.

You can either use the supplied **Terminal** application or **VS Code's terminal** window, to submit commands to build and deploy the lab projects. It is easier if you use VS Code, as in the following:



As you can see from the image above, VS Code has a terminal window open, and the folder containing the labs is accessible from this terminal window. In this lab, we will build, deploy, and test the decision automation labs to test that the environment is running correctly. Perform the following steps to verify the labs:

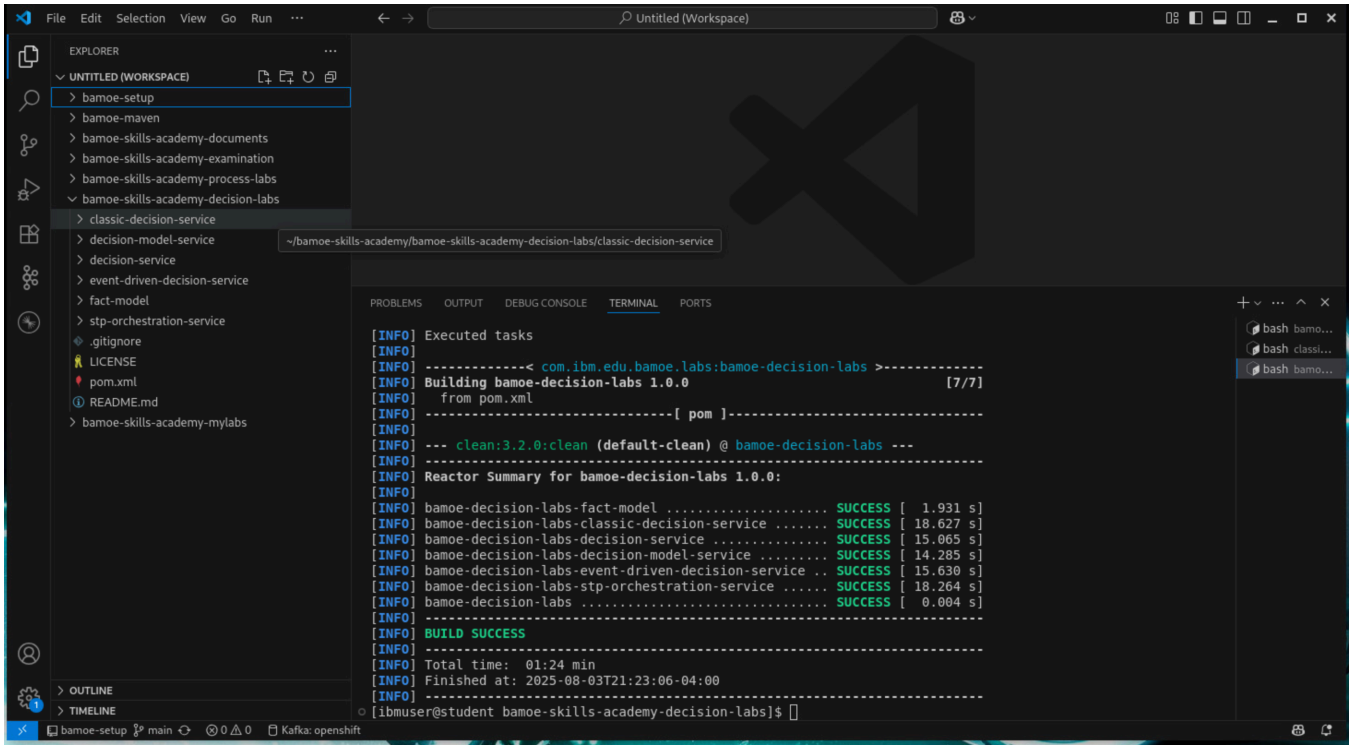
1. First step is to build the decision labs using Maven. In a terminal window type:

```
cd /home/ibmuser/bamoe-skills-academy
cd bamoe-skills-academy-decision-labs
mvn clean install
```

2. Once you have successfully built all decision labs, then the next step is to build the container images and deploy them to the OpenShift cluster. This is done automatically when you type:

```
mvn package -Popenshift
```

When you execute this command, Maven will package each project as a Docker container image, using Quarkus JIB. Once the container image has been created successfully, the container image is automatically deployed to the OpenShift cluster (based on the application property called ***quarkus.container.image.push=true)***, which is stored in each project's /src/main/resources/application.properties file. If you type this command at the root folder of the decision labs repository, it will package each lab project, create the images, and deploy to OpenShift for all projects. You can perform the same action on a per-project basis, simply by navigating into the actual lab folder, such as ***classic-decision-service.*** The output should look something like this:



If you don't see success from the Maven build, then you may need to re-login to both the OpenShift cluster as well as Docker, since Quarkus JIB uses Docker to create the container image vs using Dockerfiles, which are hard to maintain and understand. To login to both systems, go to a terminal window and type the following:

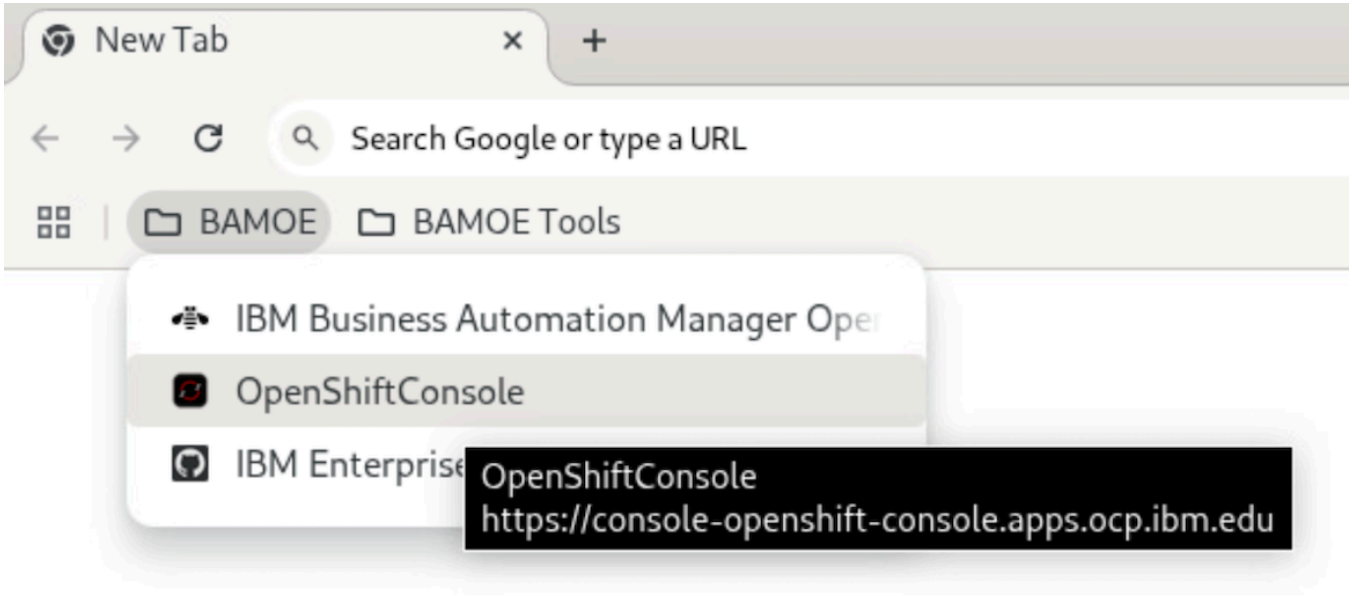
```
cd /home/ibmuser/bamoe-skills-academy
cd bamoe-setup/openshift/docker
./setup-quarkus-jib.sh
```

The **setup-quarkus-jib.sh** script automatically logs the user into the OpenShift console using the command line interface, but then also logs you into docker. This may be required if more than 24 hours elapses. From there you can go back to the lab folders and re-run the previous command:

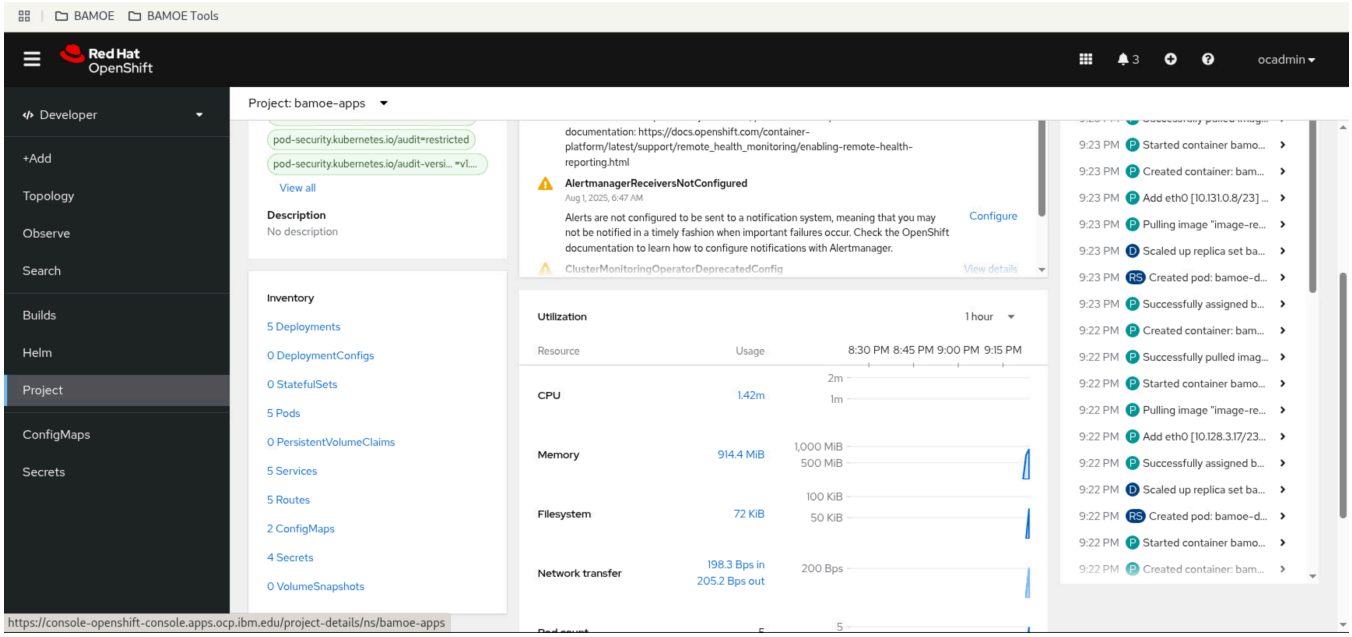
```
cd /home/ibmuser/bamoe-skills-academy
cd bamoe-skills-academy-decision-labs
mvn package -Popenshift
```

Verify the Deployment(s)

Once you have built and deployed the labs to OpenShift, the next step is to log onto the OpenShift console and verify that all project **deployments**, **pods**, **services**, and **routes** were created. Using the supplied bookmark in Google Chrome, log into the OpenShift console:



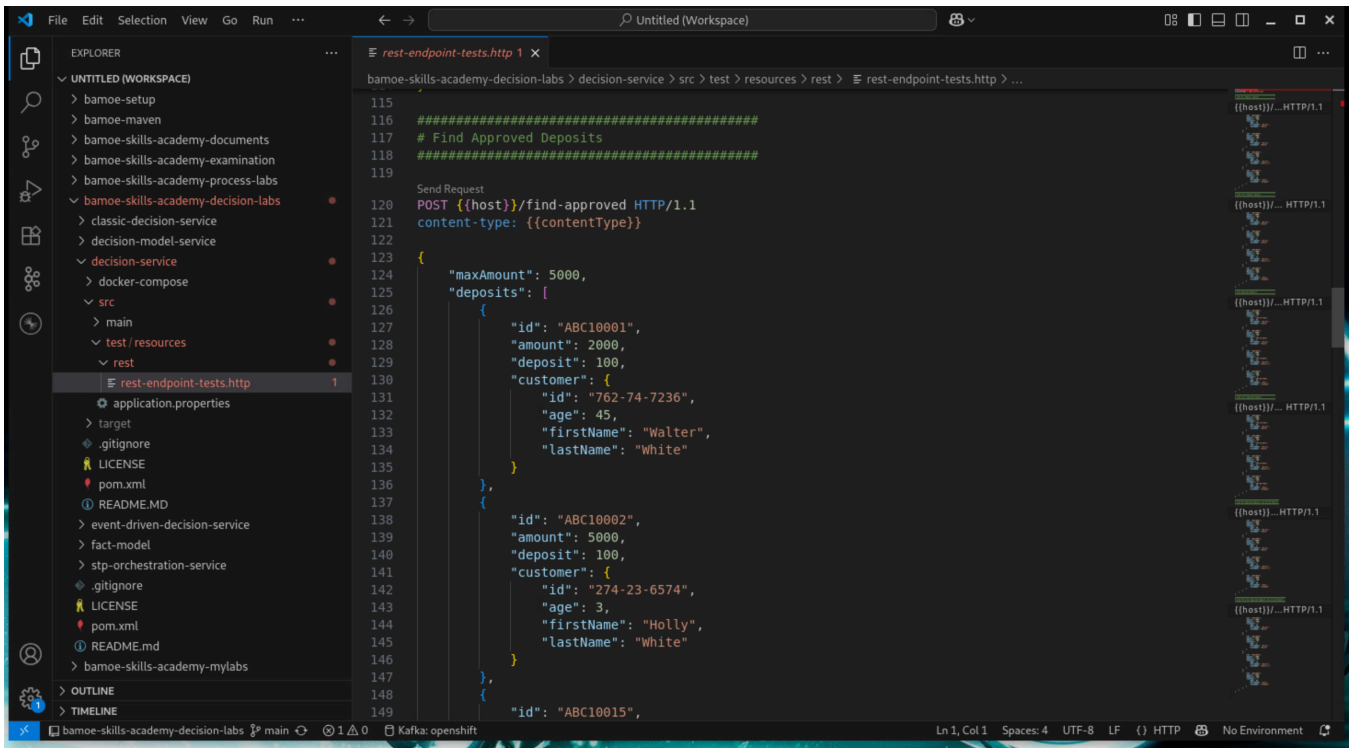
and under the **Developer** role, make sure the **bamoe-apps** project has been selected:



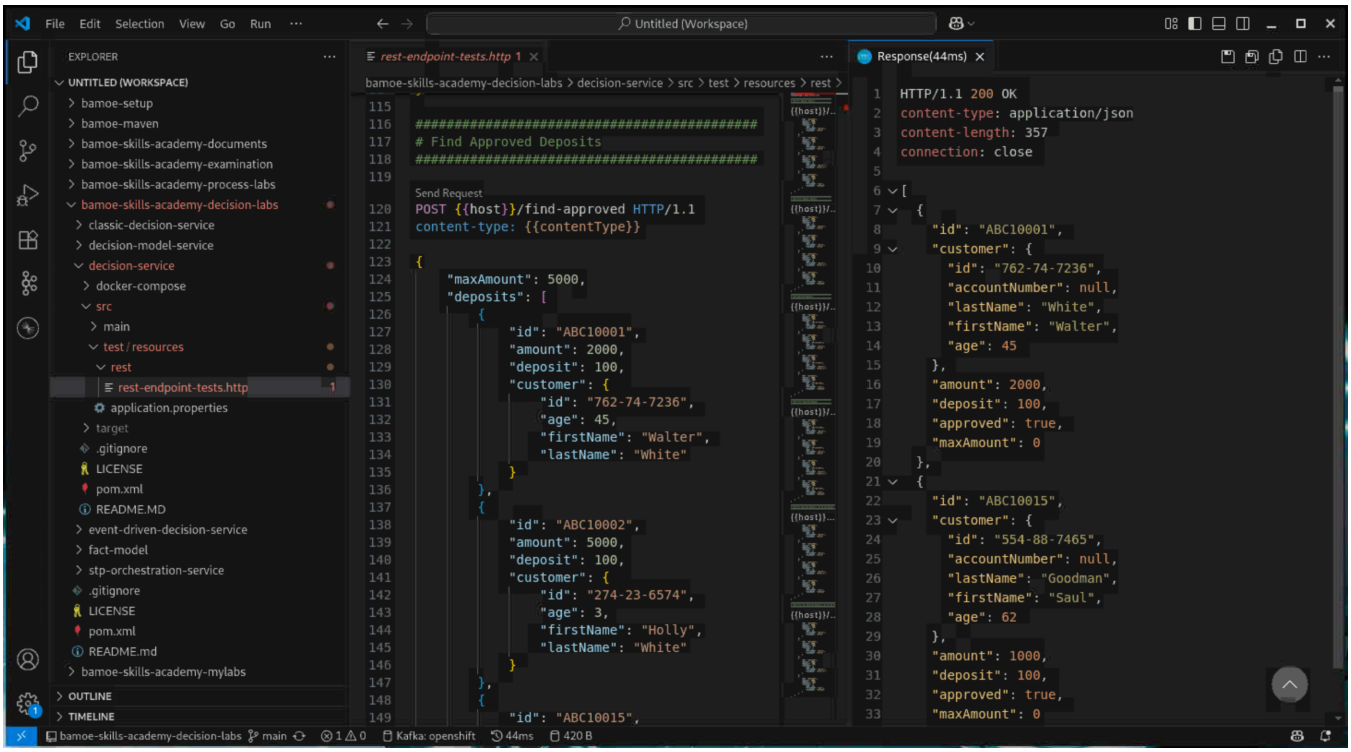
If your build and deployment was successful, you should see 5 **deployments**, **pods**, **services**, and **routes** representing the 5 decision automation lab projects. If you do not see these objects then go back to the step where you needed to login to docker, which is typically the issue with automatic deployment.

Test the Labs

Once you have built and deployed the labs to OpenShift, the next step is to run all unit tests for each lab. This involves using VS Code and the REST Client extension. Each lab project includes a set of unit tests in the src/test/resources/rest folder. In VS Code, select the project you want to test from the VS Code Navigator window, and open the file named **rest-endpoint-tests.http**:



Once you have the test open, as you can see on line **#118** there is a link called "**Send Request**", which will send the JSON to the exposed REST API endpoint for the decision service route. Click on "**Send Request**" and then a response window will be opened, showing the results from calling the decision service from VS Code to OpenShift:



Important: Each http test file contains a set of variables, notated by the "@" symbol, which describes the URL where the decision service is deployed. Each test file contains both a localhost and a remote (OCP) URL. Make sure the **@host={{remote-host.url}}** is set so that your target is the deployed services in OpenShift:

```
###############################################
# REST Endpoint Tests
###############################################

0 references
@local.host.url = http://localhost:8080
1 reference
@remote.host.url = https://bamoe-decision-labs-decision-model-service-bamoe-apps.apps.ocp.ibm.edu
1 reference
@host={{remote.host.url}}
1 reference
@contentType = application/json
```

This concludes this lab exercise!

Summary

In this lab you have:

- Learned how to access and use your student lab environment
- Built and deployed all decision automation labs to the OpenShift environment
- Tested at least one (***decision-service***) lab on OpenShift
- You are now ready to start building and deploying your own lab projects!