Business Process Automation Scenario using
AWS RDS/EC2 + Quarkus + docker + Camunda

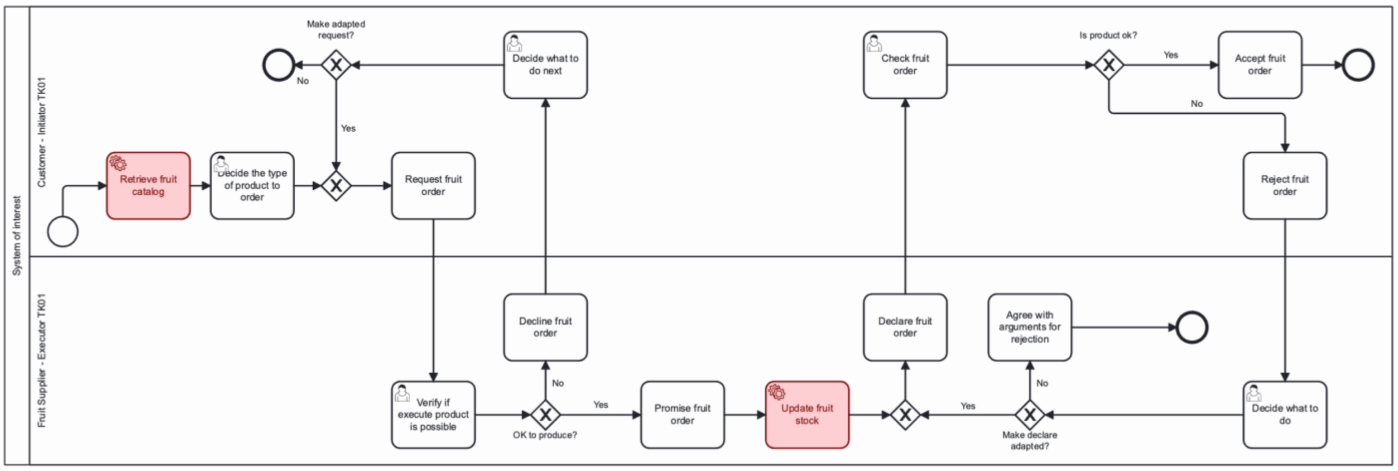
Instituto Superior Técnico
Universidade de Lisboa
Integração Empresarial / Enterprise Integration

Managing a business process of ordering fruit

The purpose of this integration scenario is to illustrate the design, the development, the deployment, and the testing of a business process of ordering fruit to a supplier by a customer.

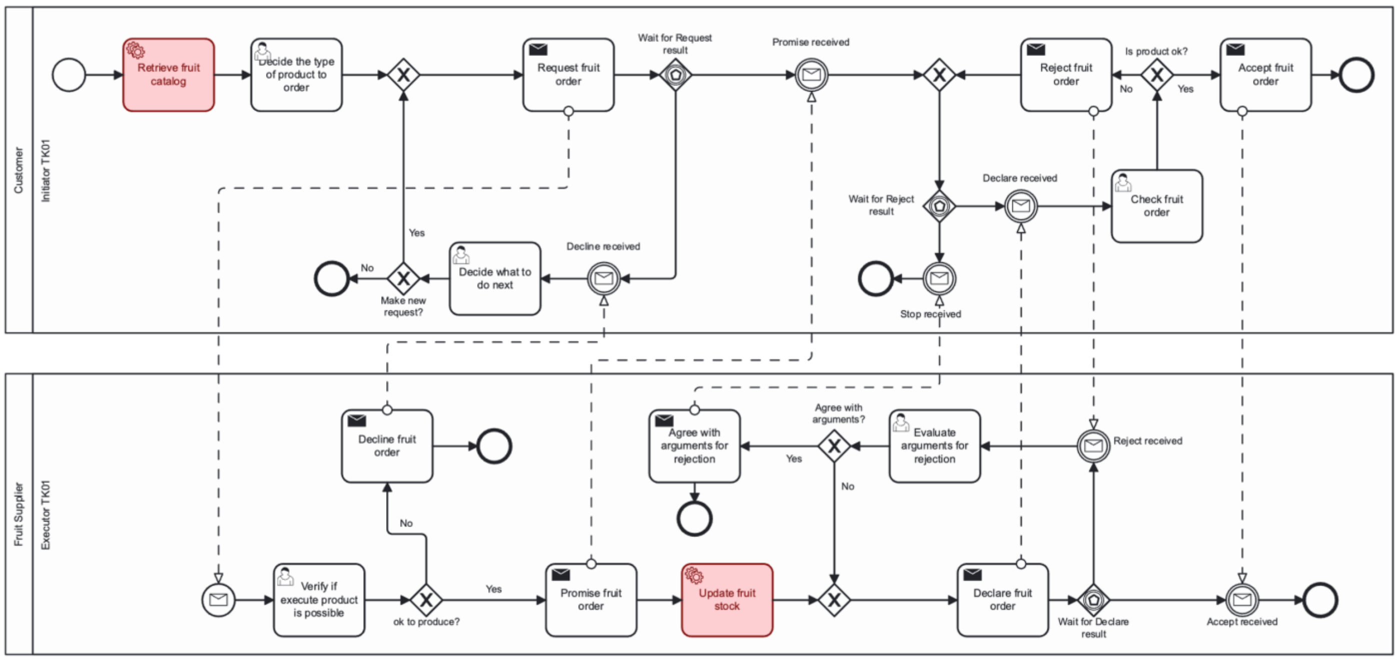
To that end, the customer decides what type, and quantity, of fruit to order accordingly with the stock availability. Then, the customer requests the desired fruit and the supplier promise or decline that order. After that, the quantity stock of the fruit is updated, and the supplier declares the order to the customer. Finally, the customer accepts or rejects the order. If accepted by the customer, the process ends, otherwise both actors need to agree upon an ending.

To implement the whole business process, a single execution system can be considered, as depicted in the following figure (a BPMN process) where both business actors use the same system (they are represented by different lanes).



Two service tasks are identified, marked as red in the above figure, the first one to retrieve the fruits catalog, and the second to update it. Both will be enforced by Quarkus microservices provided by a REST API (as introduced in the correspondingly tutorial).

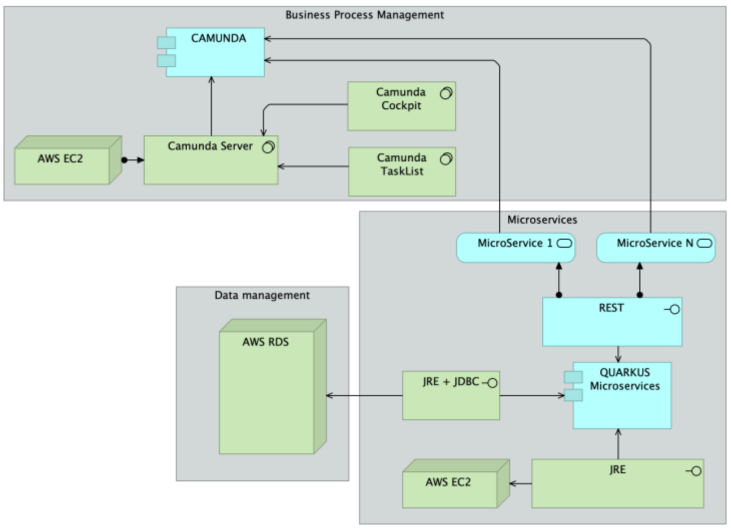
Other implementation options can (and, in the industry, very often it does!) include multiple business processes execution systems that require integration between them. In that context, each actor interacts with one or more process execution system to execute their activities. In this scenario, the alternative implementation option with one business process engine by actor is depicted in the following figure (a BPMN collaboration).



Again, two service tasks are identified, the first one to retrieve the fruits catalog, and the second to update it. Both will be enforced by Quarkus microservices provided by a REST API (as introduced in the correspondingly tutorial). The difference is that two systems are identified, the first retrieves the fruit catalog and the second updates the fruit catalog. Herein, the usage of microservices has the advantage of decoupling the business entities from the business process design.

Between the two previous options[1], the best implementation option decision depends on the decoupling of the systems, usually each enterprise has its own systems, and it is not possible for an external actor to use it. The solution is thus to integrate the available systems from the different enterprises, where each enterprise is responsible for its own set of tasks, as clearly specified in the second BPMN diagram.

Next step is to design the technological architecture. Considering the first implementation option, the correspondingly technological integration scenario is settled: two REST API are available (`Retrieve fruit catalog`, `Update fruit stock`) at AWS service and are consumed by the Camunda platform using its Camunda `http-connector`. Quarkus is the framework of this REST API provider. Each REST API request depend on a database that is also provided by AWS – the RDS. Finally, the Camunda platform is also deployed in AWS EC2. The following figure depicts the desired technology architecture.

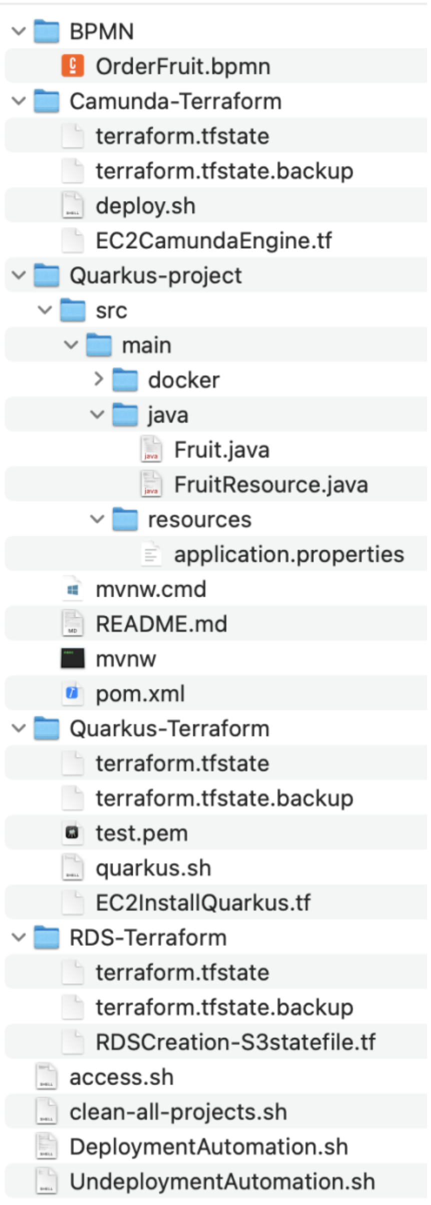


[1] You can create similar BPMN diagrams automatically, using the SemantifyingBPMN tool. It is public and available at https://github.com/SemantifyingBPMN/SemantifyingBPMN

To deploy this integration scenario the following plan is defined:

	Project task description	Support tool	Type of activity
1	Develop the Quarkus microservices	VS Code + mvn	Develop
2	Design the BPMN process	Camunda modeler	Design
3	Create an AWS RDS database	Terraform	Create
4	Develop the datamodel	SQL database creation	Develop
5	Configure Quarkus project configuration with AWS RDS identification and push its image into dockerhub	VS Code + mvn + docker	Configure
6	Create a EC2 instance and pull your previous Quarkus image	Terraform + docker	Create
7	Create a EC2 instance and pull Camunda image	Terraform + docker	Create
8	Configure BPMN process configuration (service tasks) with Quarkus EC2 instance identification	Camunda modeler	Configure
9	Deploy the BPMN process	Camunda modeler	Deploy

For your learning purposes, you can find the required artifacts already available in the following folders in a zip file, accordingly with this organization:



```
BPMN
    OrderFruit.bpmn
Camunda-Terraform
    terraform.tfstate
    terraform.tfstate.backup
    deploy.sh
    EC2CamundaEngine.tf
Quarkus-project
    src
        main
            docker
            java
                Fruit.java
                FruitResource.java
            resources
                application.properties
    mvnw.cmd
    README.md
    mvnw
    pom.xml
Quarkus-Terraform
    terraform.tfstate
    terraform.tfstate.backup
    test.pem
    quarkus.sh
    EC2InstallQuarkus.tf
RDS-Terraform
    terraform.tfstate
    terraform.tfstate.backup
    RDSCreation-S3statefile.tf
access.sh
clean-all-projects.sh
DeploymentAutomation.sh
UndeploymentAutomation.sh
```

To test it, all you must do is to change for the configuration to your own environment, at the following locations:

- **/access.sh**

```
aws_access_key_id="YOUR AWS ACCESS KEY"
aws_secret_access_key="YOUR AWS SECRET KEY"
aws_session_token="YOUR AWS TOKEN"
```

- **/Quarkus-project/mysql-RDS-CRUD/src/main/resources/application.properties**

```
quarkus.datasource.reactive.url=mysql://YOUR_RDS_ADDRESS:3306/quarkus_test_all_operations
quarkus.container-image.group=YOUR GIT USERNAME
```

- **/Quarkus-Terraform/quarkus.sh**

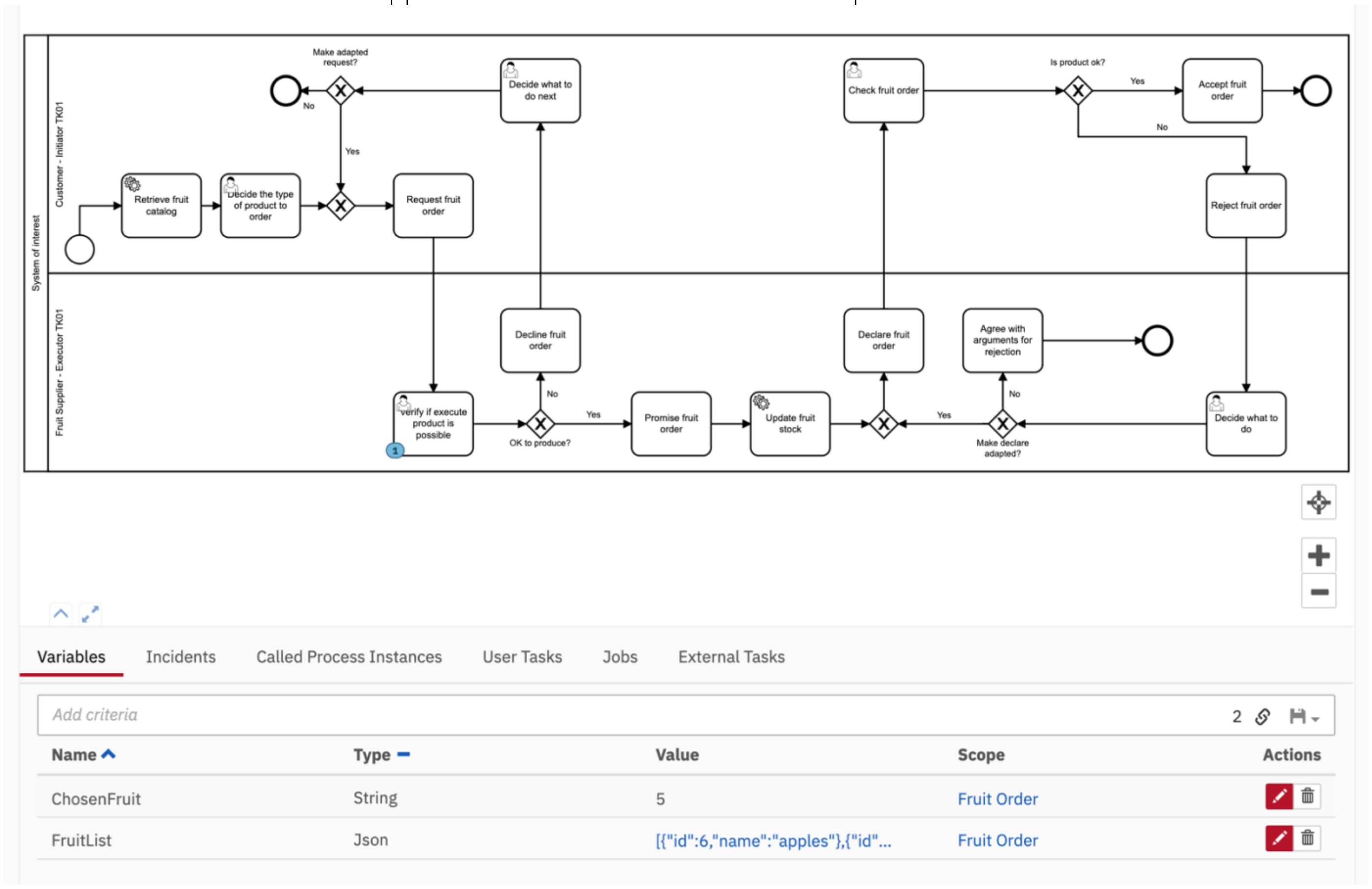
```
sudo docker login -u "YOUR GIT USERNAME" -p "YOUR GIT PASSWORD"
sudo docker pull YOUR GIT USERNAME/scenario1:1.0.0-SNAPSHOT
sudo docker run -d --name scenario1 -p 8080:8080 YOUR GIT USERNAME/scenario1:1.0.0-SNAPSHOT
```

- Change the url of "Retrieve fruit catalog" and "Update fruit catalog" tasks, in Camunda BPMN file, to your /Quarkus-Terraform/quarkus.sh provided address.

Finally, for playing and debugging you may use the following tools:

- POSTMAN
- SQL Workbench
- CAMUNDA Cockpit

Here is an execution snippet of an instance of the Fruit Order process:



Name	Type	Value	Scope	Actions
ChosenFruit	String	5	Fruit Order	
FruitList	Json	[{"id":6,"name":"apples"},{"id"...	Fruit Order	

Homework to follow-up this Integration Scenario:

As you noticed, the customer is choosing a single type of fruit and ordering it, while the fruit Supplier deletes that type of fruit from the catalogue.

Modify the implementation to allow the ordering of a specific quantity of a fruit type by the customer. In practice, you will have to consider not only the type of fruit, but also the ordered quantity, and manage it.

What do you have to change in the BPMN process? How?

What do you have to change in the Quarkus microservices? How?

What do you have to change in the datamodel? How?