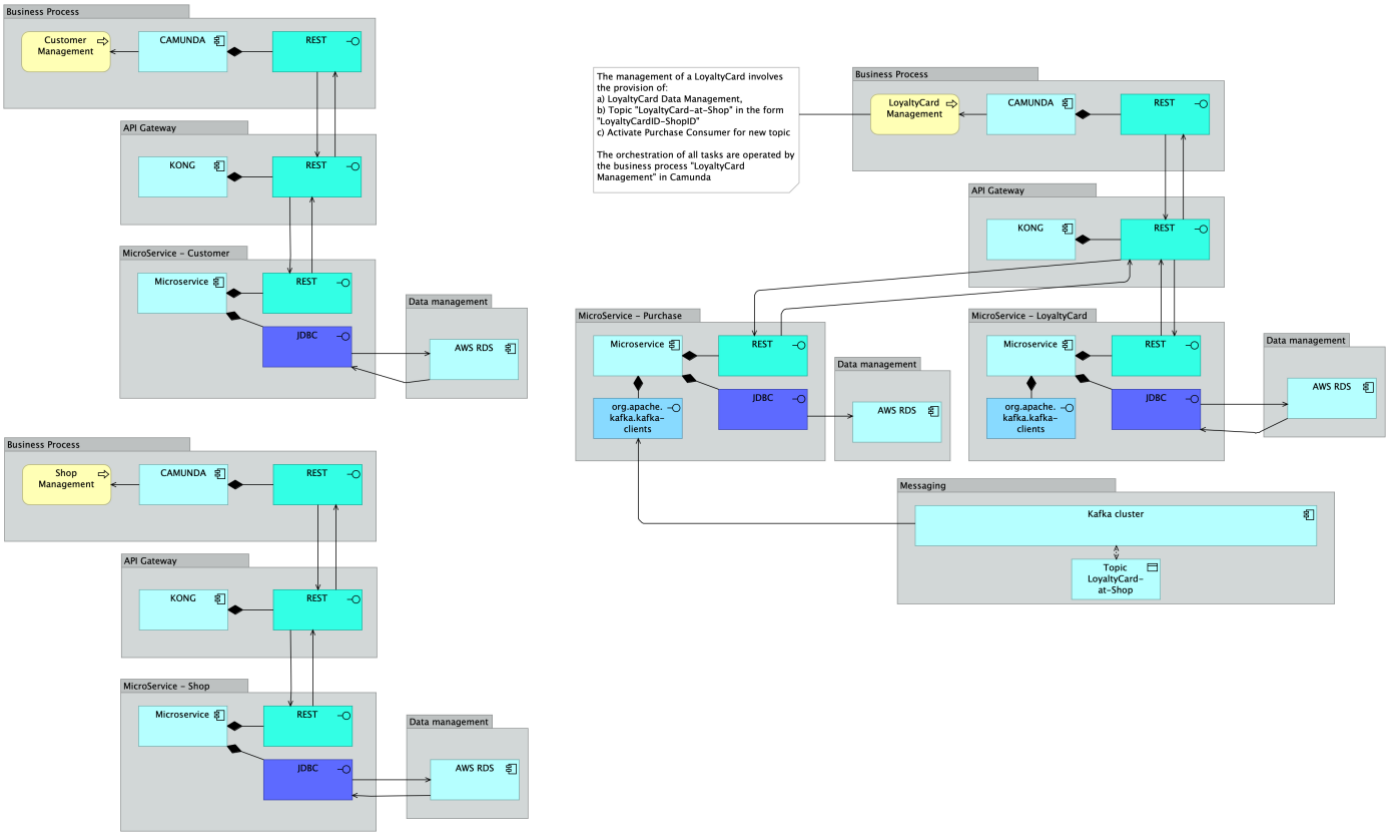
DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Project Automation Scenario
using All IE technologies

Instituto Superior Técnico
Universidade de Lisboa
Integração Empresarial / Enterprise Integration

LaaS Proof-of-Concept (PoC)

The purpose of this integration scenario is to illustrate the design, implementation, deployment and management of the IE technologies within the scope of your project. To that end, **three example business processes**[1] are available: Customer management, Creation of Shop Management, and LoyaltyCard Management. **Four example microservices** are also available[2]: Customer, Shop, LoyaltyCard, and Purchase. The application architecture is a subset of your project application architecture, as follows:



The business processes examples and microservices examples offers you a LaaS PoC for your study, test, reuse, and adaptation. None of the components are fully developed, neither fully traceable to all the project' requirements. You will need to understand them, and then, adapt.

[1] For demonstration purposes, each business process is using one of the patterns provided.
[2] The required versions of Apache and Java are: Apache Maven 3.9.6 and Java version: 17. Also requires a docker hub free account.

To deploy this integration scenario the following plan is defined:

	Project task description	Support tool	Type of activity
0	Configure the batches with the configuration of your target environment	Vscode	Configure
1	Deploy the environment setup with the script "DeploymentAutomation.sh"	Terraform	Deploy
2	Configure the BPMN processes configuration (service tasks) with Quarkus EC2 instance identification for microservices	Camunda modeler	Configure
3	Deploy the BPMN processes	Camunda modeler	Deploy
4	Test the BPMN processes *Hint: to invoke a process within a single BPMN file with more than one process definition, go directly to the start endpoint, for instance, using curl:* curl -H "Content-Type: application/json" -X POST -d '{"businessKey":"chaveexecucao","variables":{"businessKey":{"value":"chaveexecucao","type":"string"}}}' http://EC2_NAME_CAMUNDA:8080/engine-rest/process-definition/key/BusinessActor1CustomerManagement/start curl -H "Content-Type: application/json" -X POST -d '{"businessKey":"chaveexecucao","variables":{"businessKey":{"value":"chaveexecucao","type":"string"}}}' http://EC2_NAME_CAMUNDA:8080/engine-rest/process-definition/key/BusinessActor1LoyaltyCardManagement/start	Camunda apps	Test

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
yourDockerUsername=YOUR DOCKER HUB USERNAME
yourDockerPassword=YOUR DOCKER HUB PASSWORD
```

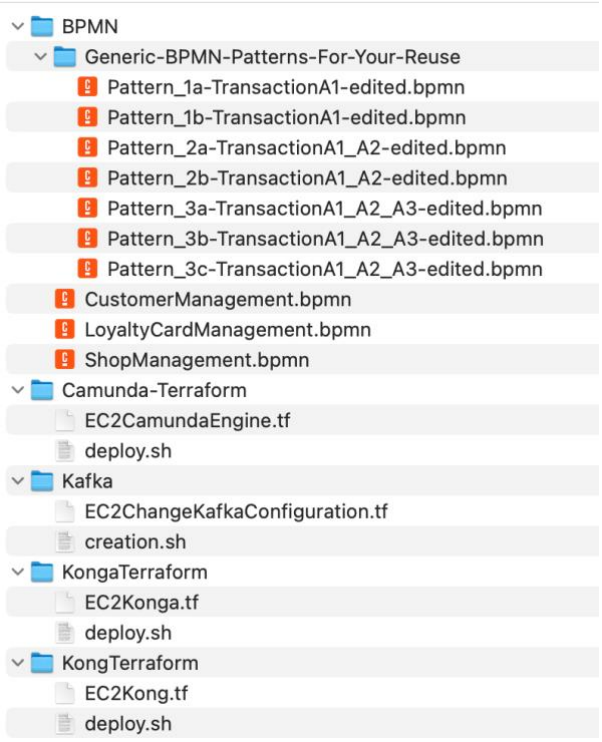
- **/ Quarkus-Terraform/*/EC2InstallQuarkus.sh**

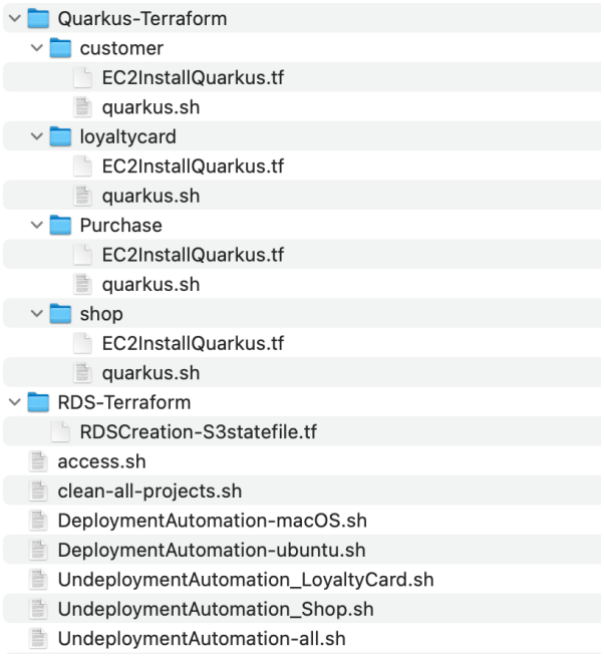
```
# This is an Amazon Linux ARM AMI built by Amazon Web Services - FOR DOCKER image COMPATIBILITY (check where you compiled your docker image and adapt the ami accordingly)
ami = "ami-0cd7323ab3e63805f"
```

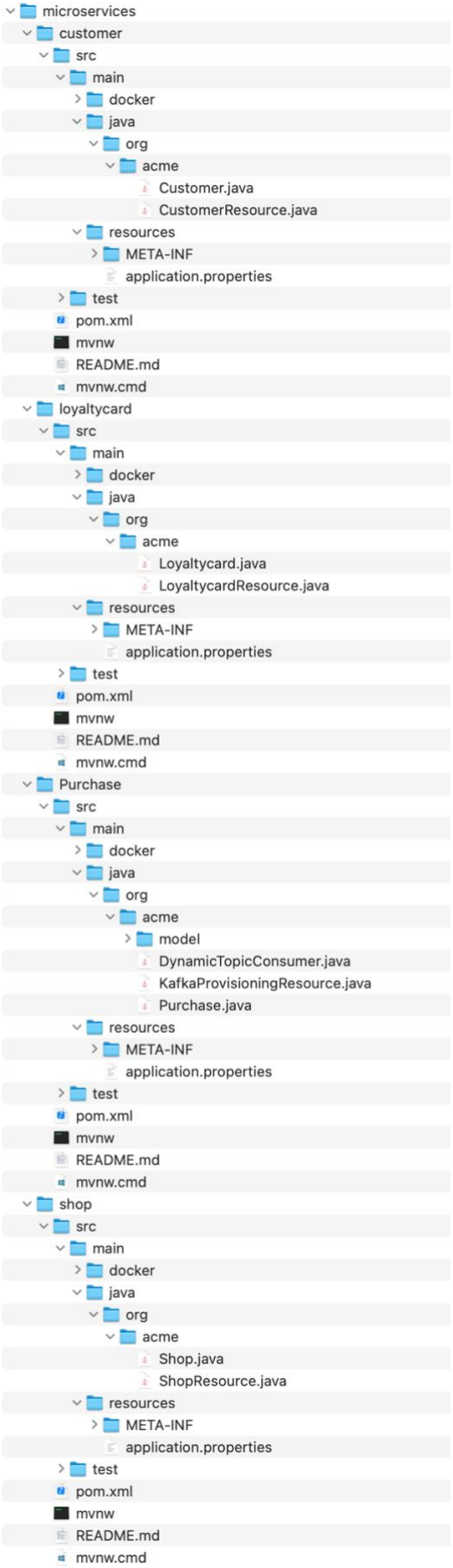
- Change the URL of service tasks in the Camunda BPMN to Kong

- In Kong, create the routes and the services to your microservices provided address. For testing purposes you can simplify integrating the Camunda directly to your microservices.

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
Generic-BPMN-Patterns-For-Your-Reuse
Pattern_1a-TransactionA1-edited.bpmn
Pattern_1b-TransactionA1-edited.bpmn
Pattern_2a-TransactionA1_A2-edited.bpmn
Pattern_2b-TransactionA1_A2-edited.bpmn
Pattern_3a-TransactionA1_A2_A3-edited.bpmn
Pattern_3b-TransactionA1_A2_A3-edited.bpmn
Pattern_3c-TransactionA1_A2_A3-edited.bpmn
CustomerManagement.bpmn
LoyaltyCardManagement.bpmn
ShopManagement.bpmn
Camunda-Terraform
EC2CamundaEngine.tf
deploy.sh
Kafka
EC2ChangeKafkaConfiguration.tf
creation.sh
KongaTerraform
EC2Konga.tf
deploy.sh
KongTerraform
EC2Kong.tf
deploy.sh


Quarkus-Terraform
customer
EC2InstallQuarkus.tf
quarkus.sh
loyaltycard
EC2InstallQuarkus.tf
quarkus.sh
Purchase
EC2InstallQuarkus.tf
quarkus.sh
shop
EC2InstallQuarkus.tf
quarkus.sh
RDS-Terraform
RDSCreation-S3statefile.tf
access.sh
clean-all-projects.sh
DeploymentAutomation-macOS.sh
DeploymentAutomation-ubuntu.sh
UndeploymentAutomation_LoyaltyCard.sh
UndeploymentAutomation_Shop.sh
UndeploymentAutomation-all.sh


microservices
customer
src
main
docker
java
org
acme
Customer.java
CustomerResource.java
resources
META-INF
application.properties
test
pom.xml
mvnw
README.md
mvnw.cmd
loyaltycard
src
main
docker
java
org
acme
Loyaltycard.java
LoyaltycardResource.java
resources
META-INF
application.properties
test
pom.xml
mvnw
README.md
mvnw.cmd
Purchase
src
main
docker
java
org
acme
model
DynamicTopicConsumer.java
KafkaProvisioningResource.java
Purchase.java
resources
META-INF
application.properties
test
pom.xml
mvnw
README.md
mvnw.cmd
shop
src
main
docker
java
org
acme
Shop.java
ShopResource.java
resources
META-INF
application.properties
test
pom.xml
mvnw
README.md
mvnw.cmd