**Project document structure and steps**

This document describes the folder content and the main commands to use to run a basic version of the project.

**Folders :**

**Docker**: contains DockerFile and DockerCompose.yml files to set up the project environment.

**Data**: contains an example of health dataset

**PyPrograms**: contains necessary python programs to create mangoDB database, kafka topics, kafka producers and kafka consumers.

**STEP1: Docker environment set-up**

# For windows users, sudo may be required

#sudo docker-compose up -d --build

docker-compose up -d --build

**STEP2: Kafka topics’ creation**

For testing topics’ creation in Kafka, use the following python programs that are in PyPrograms folder:

Update the programs to create 'urgent\_data' and 'normal\_data' topics.

For that, python-kafka is needed.

! pip install kafka-python [[1]](#footnote-0)

python3 createKafkaTopic.py

You can check with shell command line the created topics using a navigator

http://localhost:9021

python3 kafkaConsumer.py

python3 kafkaProducer.py

**STEP3: MongoDB and connectors’ configuration**

Please refer to MongoDB Lab, to access to mongo container and create Patient DB and collections.

To add the DB content to MongoDB, you can use the provided script at PyPrograms

python3 pyMango.py

To add a MongoDB Kafka Sink Connector for 'urgent\_data' and 'normal\_data' topics into the 'patient.sensorlogs' collection in mongodb" :

curl -X POST -H "Content-Type: application/json" --data '

{"name": "mongosinkUrgent\_data",

"config": {

"connector.class":"com.mongodb.kafka.connect.MongoSinkConnector",

"tasks.max":"1",

"topics":"urgent\_data",

"connection.uri":"mongodb://root:root@mongo:27017",

"database":"patient",

"collection":"sensorlogs",

"key.converter":"org.apache.kafka.connect.storage.StringConverter",

"key.converter.schemas.enable":false,

"value.converter":"org.apache.kafka.connect.storage.StringConverter",

"value.converter.schemas.enable":false

}}' http://localhost:8083/connectors -w "\n"

curl -X POST -H "Content-Type: application/json" --data '

{"name": "mongosinkNormal\_data",

"config": {

"connector.class":"com.mongodb.kafka.connect.MongoSinkConnector",

"tasks.max":"1",

"topics":"normal\_data",

"connection.uri":"mongodb://root:root@mongo:27017",

"database":"patient",

"collection":"sensorlogs",

"key.converter":"org.apache.kafka.connect.storage.StringConverter",

"key.converter.schemas.enable":false,

"value.converter":"org.apache.kafka.connect.storage.StringConverter",

"value.converter.schemas.enable":false

}}' http://localhost:8083/connectors -w "\n"

echo "\nmongod connector configured"

sleep 2

echo "\nKafka Connectors:"

curl -X GET "http://localhost:8083/connectors/" -w "\n"

sleep 2

echo '''

==============================================================================

At this stage, the MongoDB server and Kafka broker are ready to receive data.

**STEP4 : Running the e-health scenario**

You can execute the alert\_nurse.py program to simulate the nurse waiting for alert messages.

python3 alert\_nurse.py

You can use the gateway.py program to simulate the gateway reading data from heart.csv and sending messages to the appropriate topic in Kafka depending on the ‘target’ field (either ‘urgent\_data’ or ‘normal\_data’).

Check that the sent messages are all persisted in MongoDB.

Check that alert messages are received by the nurse.

1. You may need to install pip , for ex (on MAC)

   curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

   python3 get-pip.py [↑](#footnote-ref-0)