Internals of Application Server

Hackathon 2:
Sensor Manager and Data Binding

Team 1 (Group 5)

Avi Agrawal	(2020201046)
Sailee Shingane	(2020201004)
Shubham Swetank	(2020201025)

Table of Contents

Introduction :

Our platform provides various microservices which will facilitate application development. Communication module will provide useful APIs to integrate independent applications on it. All the communication is done by kafka and the sensor will produce the data on the partitions of a kafka topic.

Sensor manager module is responsible for new sensor instance registration and communication module then will be binding sensor data with deployed algorithms and sending the data as per the configuration provided by the application developer.

Implemented Module

S.No	Modules/sub systems Implemented
1.	Sensor Registration
2.	Sensor Manager
3.	Sensor Data Binding with the running algorithm using kafka topic

Actors

- **Platform Developer** - We are developing demo algorithm which is actually developer by platform develop and he will make use of Application manager and authenticates himself uploads the application
- **Admin** - Admin will make use of scheduler to create request with deployer manager to deploy specified application in runtime server
- **End User** - To receive notifications sent by applications running in runtime server via action manager service.

Block Diagram:

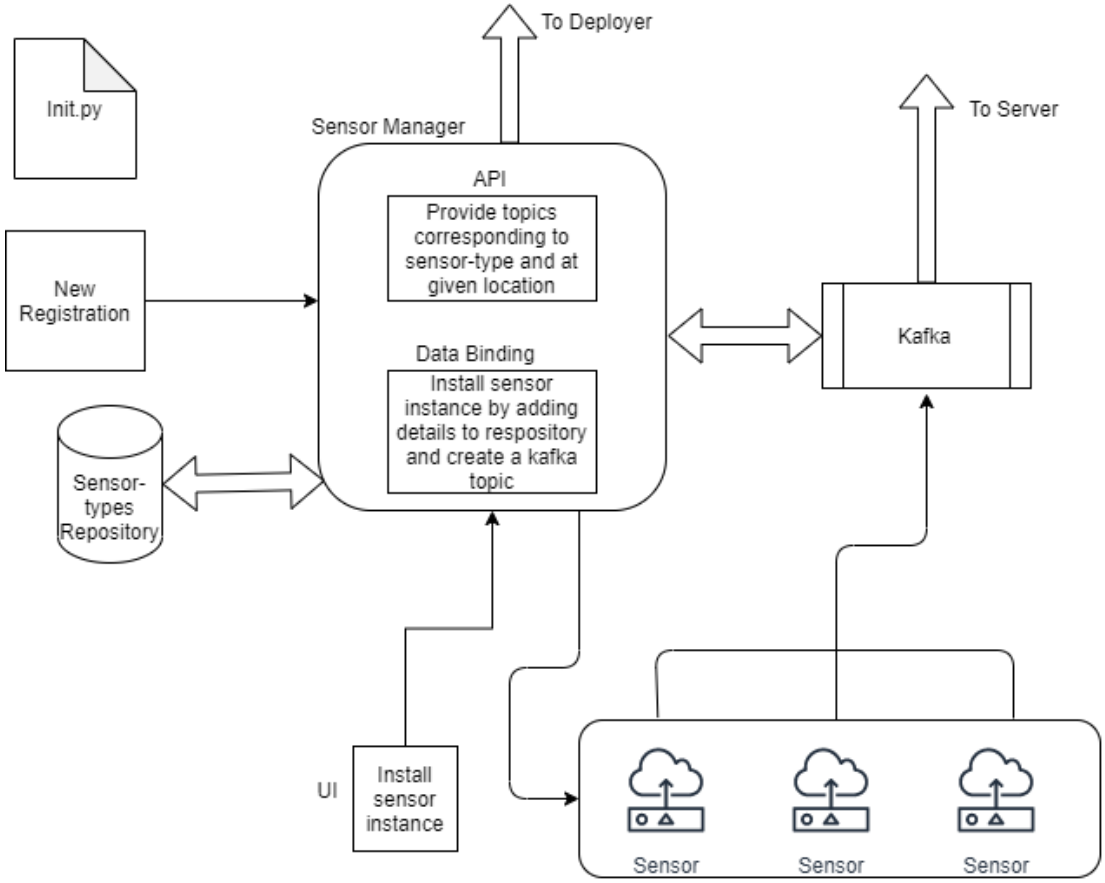


Fig.: Sensor Manager & Data Binding

Primary components:

- Sensor Manager Initialiser -
    - For initialising the sensor manager and setting it up for the further communication with the kafka topics and sensor instances

- New Sensor Registration -
    - Admin will install sensors to the platforms via UI
    - Admin will provide meta/config file with information like Sensor Type ,Sensor Geo Location , Sensor ID
    - Sensor registration module will validate the config file and store sensor information in the sensor repository

- Sensor Manager -
    - Sensor manager module is responsible for new sensor instance registration and communication module then will be binding sensor data with deployed algorithms and sending the data as per the configuration provided by the application developer.
    - It will make sure how the data is being sent to the running application and control the data rate as well.
    - Sensor Manager will also make sure there is a partition created for binding the sensor instances with some topic of Kafka for the data transfer between the modules.

- Sensor topic Consumer: -
    - Responsible for fetching data from message queues on sensor topics.
    - Push data on message queue on algorithm topic

Application Flow:

1. Each sensor will have a gateway point which will be responsible for making the sensor data in and out.
2. We assume the admin have collected the sensor instance information and types and it sends a json of the same to the sensor manager for the sensor registration.
3. Sensor registration: registering the sensor with their location and sensor id. This module is responsible for the initial setup of the sensors and binds it to some gateway for the data.
4. UI will be provided to the admin for installing sensors to the platform. Admin can register sensors by providing meta/config file consisting information about sensors like Sensor Type ,Geo Location , output format ,Sensor ID
5. Sensor Manager from config file(send by deployer) will get request of sensor topic
6. This config file specifies whether to get data from the sensor or control the sensor by setting some control attribute of the sensor instance.
7. Each time for every request sensor manager will detach a thread for serving the request.
8. This thread will be responsible for reading data from the sensor topic, writing the data to some other temp topic at some rate as provided by the sensor manager.
9. Sensor Manager will listen to a common topic as per defined which is defined at the start time only and the system is aware of that topic.

Interaction with other module

- On receiving sensor data requests from running algorithms , the sensor manager will be checking the existence of the sensor from the platform repository to validate the request.
- Deployer communicates with the sensor manager using Kafka for assigning appropriate sensor nodes for algorithm and sensor communication.
- Sensor manager will get requests for sensor topics from init.py script running at some node. Sensor manager will start a thread to serve the get request of the sensor topic, this thread will be getting the sensor data from the sensor topic.

Config Files:

- **Sensor_instance_request:**

  Must be sent by deployer to request the instances required for binding sensors to the algorithms

  {

  sensor_type_id:

  location:

  }

- **Sensor_registration interface:**

  This will be provided for registration of new sensor-type by the admin.

  {

  sensor_id:

  sensor_type:

  sensor_name:

  Sensor_output_format:

  sensor_control_type:

  }

- **Sensor_instance_registration:**

  This will be provided for adding instances of various sensor-types already present in the sensor repository.

  {

  sensor_type:

  geo_location:

  sensor_ip:

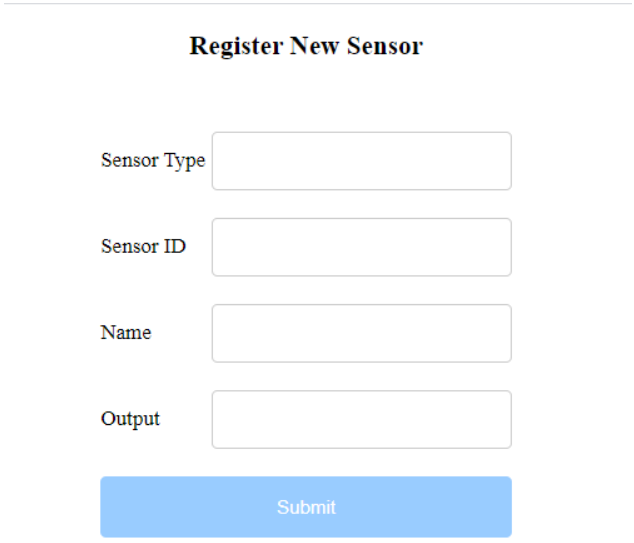
  sensor_port:

  Sensor_description:

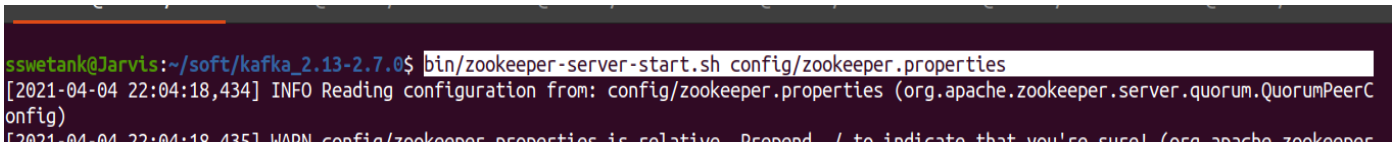
  }

Code and UI Snippet:

1. Sensor Registration



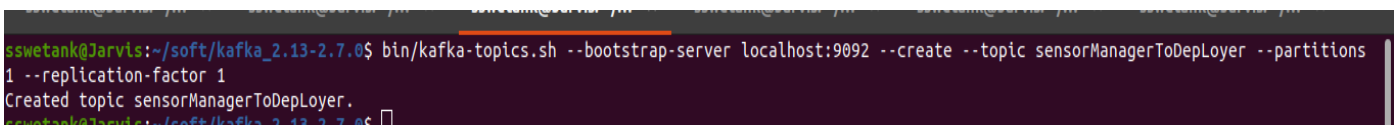
2. Launching Zookeeper Server



3. Starting Kafka server

4. Topic Creation:

a. SensorManagerToDeployer



b. DeployerToSensorManager

```
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic DeployerToServerManager --partitions
1 --replication-factor 1
Created topic DeployerToServerManager.
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$
```

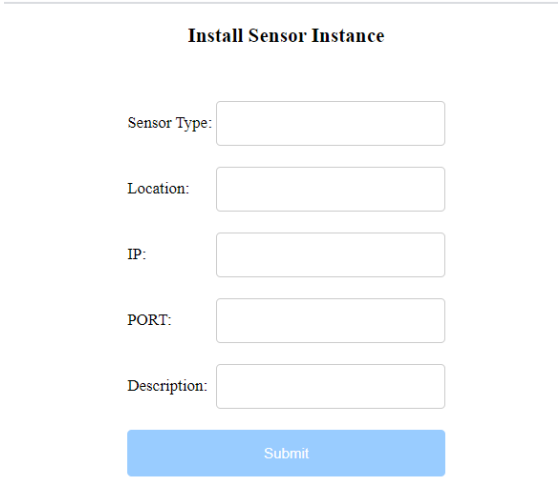
c. Creating topic for sensors

```
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic temp0 --partitions 1 --replication-fa
ctor 1
Created topic temp0.
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$
```

5. Sensor Simulation:

```
{
   "temp":{
      "id": 0,
      "sensor_id":"ts1",
      "name":"temperature",
      "output": {
         "data": "float"
      }
   },
   "AC": {
      "id" : 1,
      "sensor_id":"acc1",
      "name":"temperature",
      "output": {
         "status" : -1
      }
   }
}
```

6. Sensor Installation:



Technology Used

- Kafka for queuing data stream to the model
- Python
- Interaction with MongoDB by handling JSON files.
- Bash shell scripts
- Sensor simulator