Internals of Application Server

# IOT Platform: Sensor Manager & Data Binding

*Team Design Document*

Team 1 (Group 5)

***Submitted by:***

Avi Agrawal(2020201046)

Sailee Shingane(2020201004)

Shubham Swetank(2020201025)

# Table of Contents

## Introduction

Sensor Manager is the part of the Distributed Application Deployment platform which will be mainly responsible for collecting and maintaining the data generated from various sensors fitted on IoT devices. This data will be provided to the applications that are deployed on the platform for analyses and processing.

It will mainly provide various APIs to the application developers for accessing the sensor data directly in their algorithms. The application deployer will be able to register the types of sensor required by the application and then install as many instances of those sensors as required by the application. The data will be entirely managed and scaled by the Sensor Manager thus freeing the developers from hassle of dealing with the sensors and hence provide an easy way for developing and deploying IoT applications with minimal effort.

## Implemented Module

| S.No | Modules/sub systems Implemented |
|---|---|
| 1. | Sensor Registration & Installation |
| 2. | Sensor Manager |
| 3. | Sensor Data Binding |

## Actors

- **Platform Developer** - Will develop and maintain services provided by sensor manager
- **Admin** - Admin will be responsible to request the sensors required by the applications to run. Sensor manager will respond with the sensors available in the registry depending upon the location/placeholder id provided by the admin
- **App Deployer** - App Developer can use the Sensor Manager services to register new types of sensors and install valid sensor instances on the platform

## Block Diagram

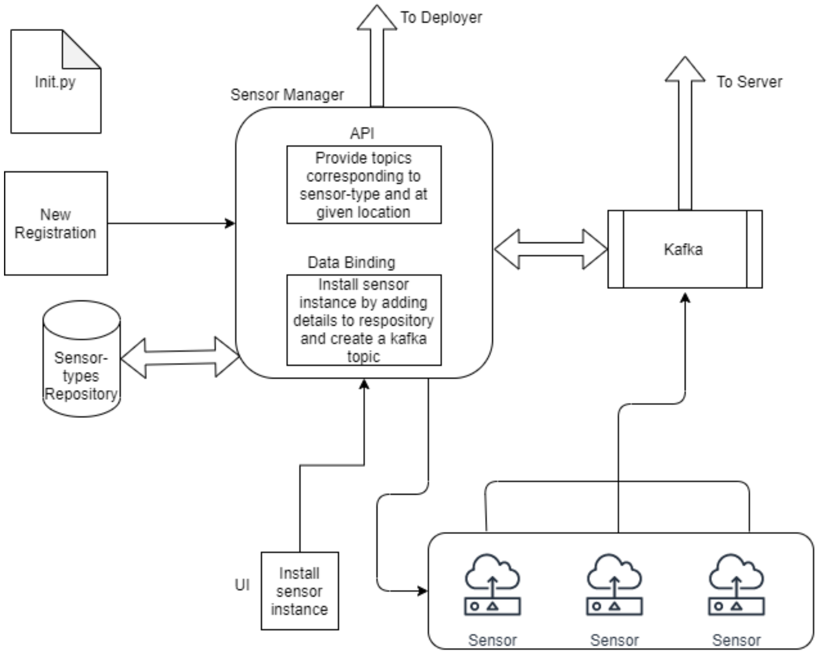

To Deployer
Init.py
Sensor Manager
To Server
API
Provide topics corresponding to sensor-type and at given location
New Registration
Kafka
Data Binding
Install sensor instance by adding details to respository and create a kafka topic
Sensor-types Repository
UI
Install sensor instance
Sensor
Sensor
Sensor

**Fig.: Sensor Manager & Data Binding**

## Solution design considerations

- Sensor Manager Initialiser
    - For initialising the sensor manager and setting it up for the further communication with other modules and sensor instances
    - After initialisation sensor registration, sensor installation and APIs to get sensor data and controller function would be available for use.
    - The sensor manager will be ready to manage sensor data using kafka

- New Sensor Registration
    - New sensors required for the application can be registered on the platform via '/new-sensor' api exposed by this module
    - The sensor information model will be validated by sensor manager and will be added to sensor repository if not already present in it.
- Sensor Instance Installation
    - After registering sensor types its multiple instances can be installed on the platform using '/install-sensor' api
    - The installed sensor instances are added in sensor registry of the platform
    - After installation the data generated by the sensor instance will be available for use within seconds after installation by any application on the platform.
- Sensor Manager
    - Sensor Manager will expose the APIs required for sensor registration/installation as well as for providing sensor data required by the application and to handle the controllers.
    - It will generate unique topic ids for the sensor instances and pass it on to the data binding submodule for creating a kafka topic on which the data of that sensor instance will be stored.
    - It will make sure how the data is being sent to the application and control the data rate as well.
    - It is also responsible for consuming the data from the kafka topics and passing them to the algorithm that has asked for it.
- Data Binding
    - Data Binding component will be mainly responsible to receive data generated by the various sensors in the sensor registry of the platform
    - It will get the data from the sensors mainly by connecting to the sensors instances via network.
    - The data thus received will be published on the unique kafka topic provided by the sensor manager for the corresponding sensor instance.

## Overall 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 has 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.

## Key Data structures

- Sensor instance request

  Generated while requesting for instances in required area or installed at particular place,

```
{
        placeholder_id:
        location:
}
```

- Sensor registration interface

  This will be generated after registration of new sensor-type

```
{
        sensor_id:
        sensor_type:
        sensor_name:
        controller_type:
}
```

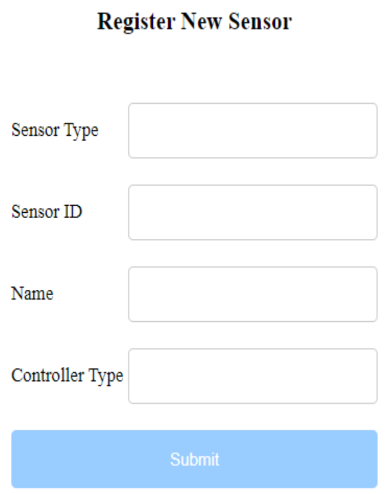
- Sensor instance registration

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

```
{
        sensor_type:
        location:
        sensor_ip:
        sensor_port:
        placeholder_id:
        controller_id:
}
```

# Interactions & Interfaces

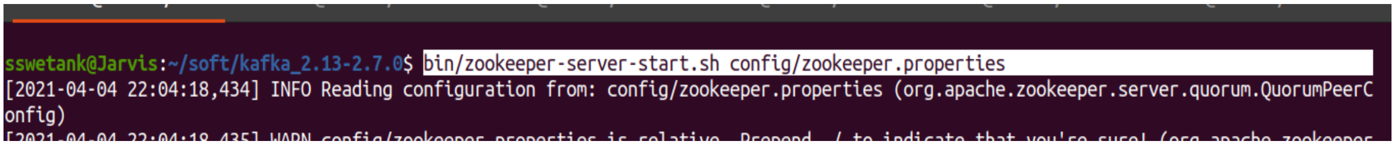
## Sensor Registration



**Register New Sensor**

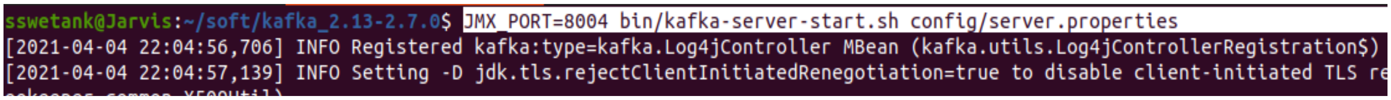
| | |
|---|---|
| Sensor Type | |
| Sensor ID | |
| Name | |
| Controller Type | |

## Launching Zookeeper Server

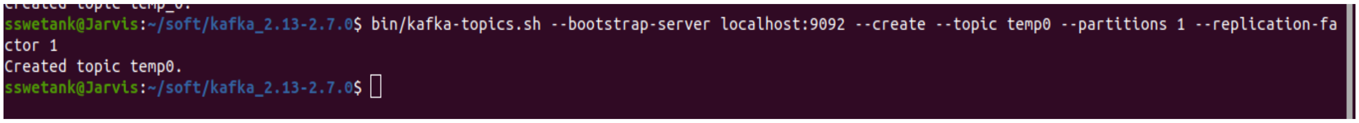


```
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$ bin/zookeeper-server-start.sh config/zookeeper.properties
[2021-04-04 22:04:18,434] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerC
onfig)
[2021-04-04 22:04:18,435] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper
```

## Starting Kafka server

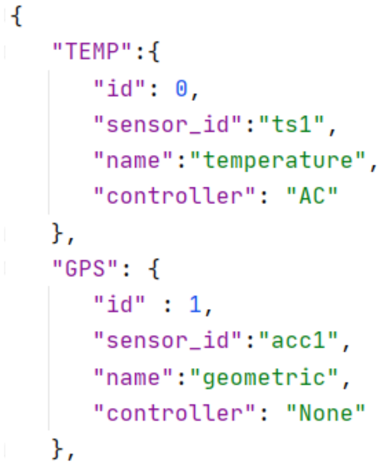


```
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$ JMX_PORT=8004 bin/kafka-server-start.sh config/server.properties
[2021-04-04 22:04:56,706] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2021-04-04 22:04:57,139] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS re
ookeeper.common.X509Util)
```



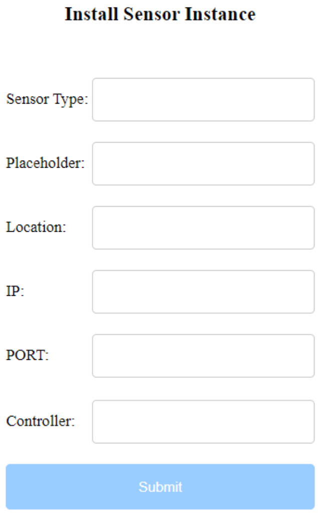
```
Created topic temp_0.
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic temp0 --partitions 1 --replication-fa
ctor 1
Created topic temp0.
sswetank@Jarvis:~/soft/kafka_2.13-2.7.0$ 
```

## Sensor Simulation



```
{
    "TEMP":{
        "id": 0,
        "sensor_id":"ts1",
        "name":"temperature",
        "controller": "AC"
    },
    "GPS": {
        "id" : 1,
        "sensor_id":"acc1",
        "name":"geometric",
        "controller": "None"
    },
```

## Sensor Installation



**Install Sensor Instance**

Sensor Type:

Placeholder:

Location:

IP:

PORT:

Controller:

# Technologies Used

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