Team 1 Requirements

PREPARED FOR

IAS PROJECT - GROUP 4

IIIT-Hyderabad

PREPARED BY

TEAM 1

Akshay M 2020201023

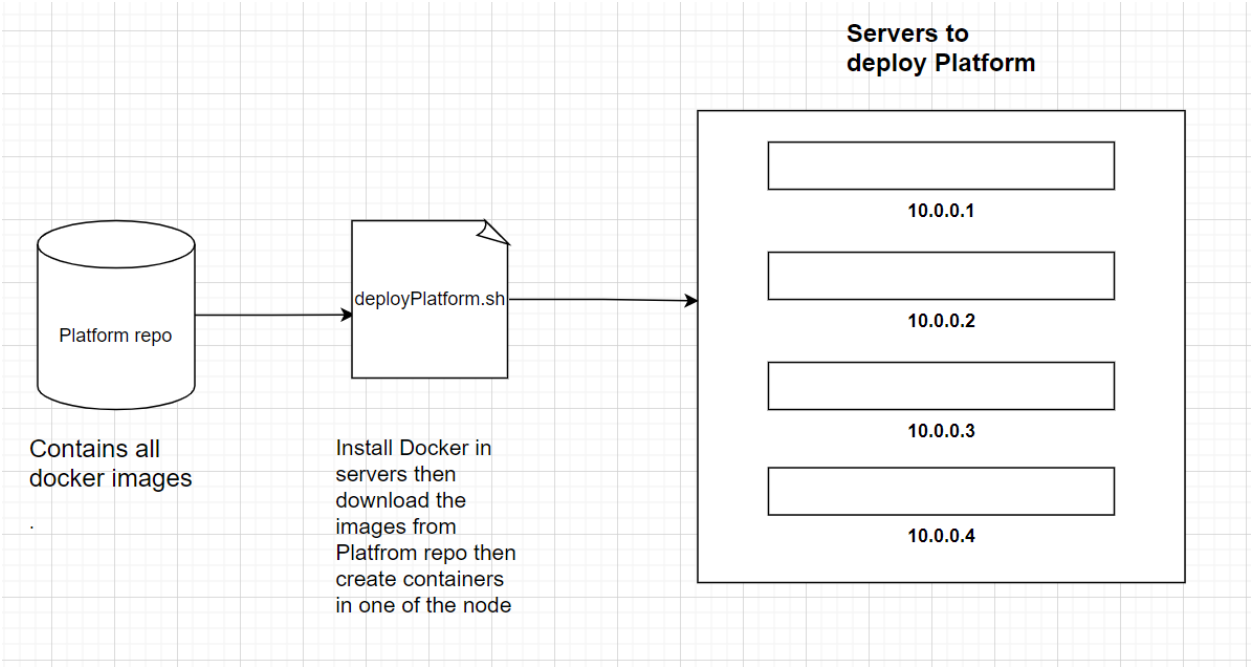
Raman Shukla 2020201098

Varun Nambigari 2020201079

Our sub team worked on the following modules

1) Provide UI and Dashboard
2) Platform Manager
3) Validation
4) Scheduler
5) Action Service
6) Sample Application

Deploying the platform:



Platform repo: (Search for docker repo)

Platform repo contains the docker images of all the services which are supposed to run in the platform. The following docker image files should be present

1) Platform Manager
2) Scheduler
3) Node Manager
4) Deployment Manager
5) Authorization Service
6) Monitoring Services
7) Sensor Manager
8) Registering a Sensor Service
9) Action Manager

DeployPlatform.sh :

Input : List of IP addresses where the platform needs to be deployed, location of the platform repo.

Functionalities:

1) Should install docker in all the servers
2) In one of the server (say 10.0.0.1) download all the images from the platform repo and start the container.
3) Now we have servers (10.0.0.2, 10.0.0.3, 10.0.0.4) for deploying the user Applications
4) Install Kafka as a messaging queue

Assumptions :

Sensor is acting like a server. Where we can ping the data and get response.
Client code to interact with sensor is provided by the user of this platform.
All the codes are written in python.

Roles :

Application Developer: (provides app.zip -> contains source code, appConfig.json and interface.json)

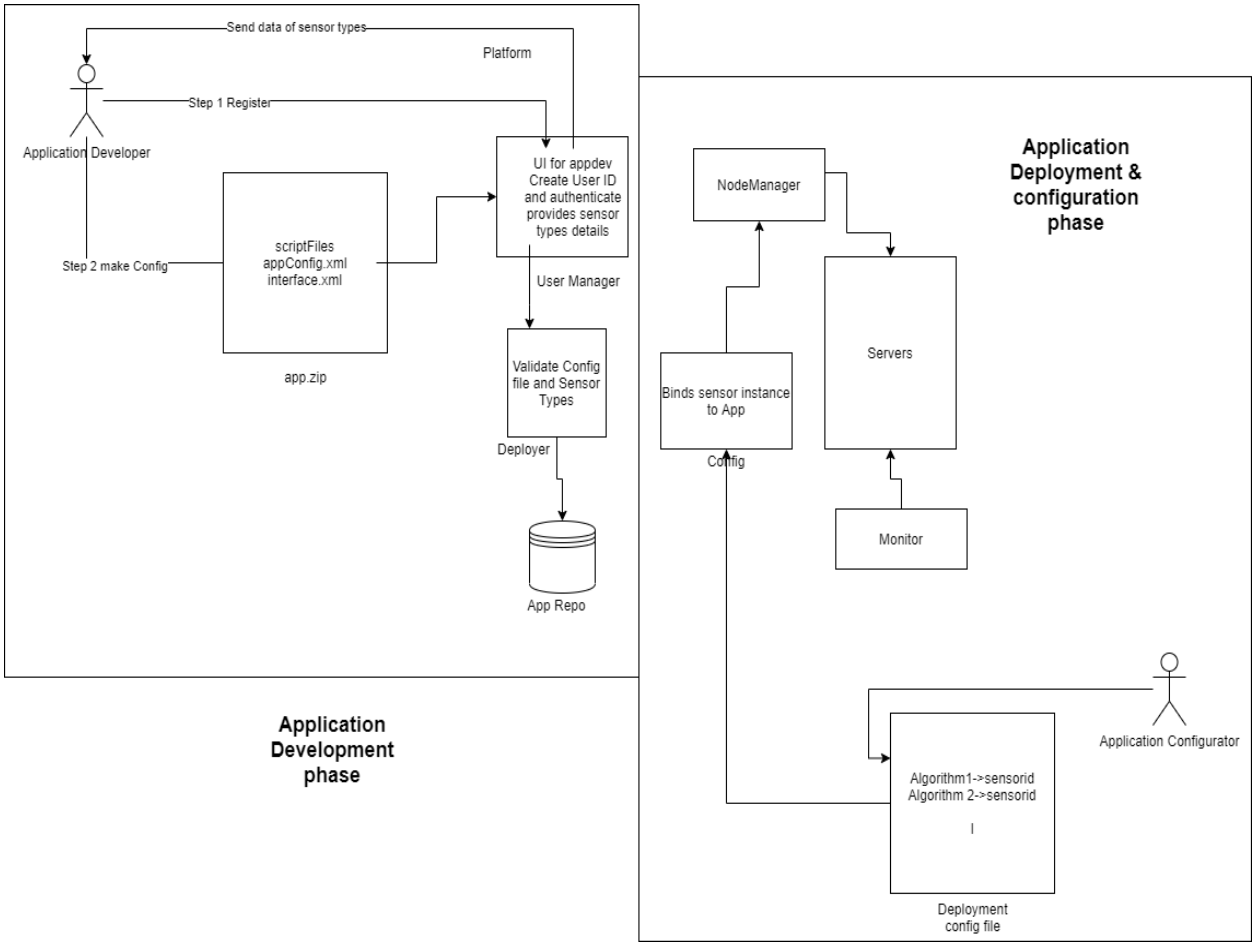
- Write the algorithms by getting and setting data from the sensor client instances provided by the platform.
- Write a proper appConfig.json and interface.json files.

Application Configurator(provides catalogue.json, deployConfig.json):

- Register the sensors type using catalogue.json, this file contains the code to connect with the sensor (client to the sensor server). Only need to pass ip and port number.
- Provides a sensor_client_instance.json which contains the details of the sensors with which we have to connect. Ex ip,port, meta data (key value format)
- Provides a deployConfig.json. This contains name of the algorithm to be deployed on which sensors which can be sorted based on given key value.

End User:

- Interacts with the application UI which is provided by application developers.
- Will receive notification from the action server.
- May trigger a particular action on a sensor.



Use Case:

1. **Farm humidity controller:** This application will measure the humidity around the land and if the humidity is below a certain threshold then we use a controller (water sprinklers) to raise the humidity.

   Input: humidity sensor 1
   Controller: Water sprinklers
   Code: HumidityController.py

2. **Farm ph controller:** This application will measure the ph around the land and if the ph is below a certain threshold then we will spread the ammonia and other substances around the land.

Input: ph sensor 1
Controller: Ammonia spreaders
Code: pHController.py

Steps to use the platform :(using example of humidity sensor)

1. *Application configurator* will provide catalogue.json which adds a sensor type(humiditysensor type is added) to the platform. Catalogue.json contains humidityclient.py to communicate with the sensor. (taking input ip and port)
2. Application developers will write the algorithms. These algorithms should take a client in the constructor to communicate with the sensor.
3. *Application configurator* will provide sensor_client_instance.json. Create_sensor_instance.py makes a database with the above details of the sensor.
4. Binding part : deploy.py will read deployConfig.py provided by the *Application configurator* and create instances of the sensorClient needed and passes the instances to the algorithm mentioned.

Dashboard

- The platform manager provides APIs for the dashboard.
- The dashboard shows the the following:
    - The logs from the running applications.
    - The status of various containers.
    - The logs of various containers.

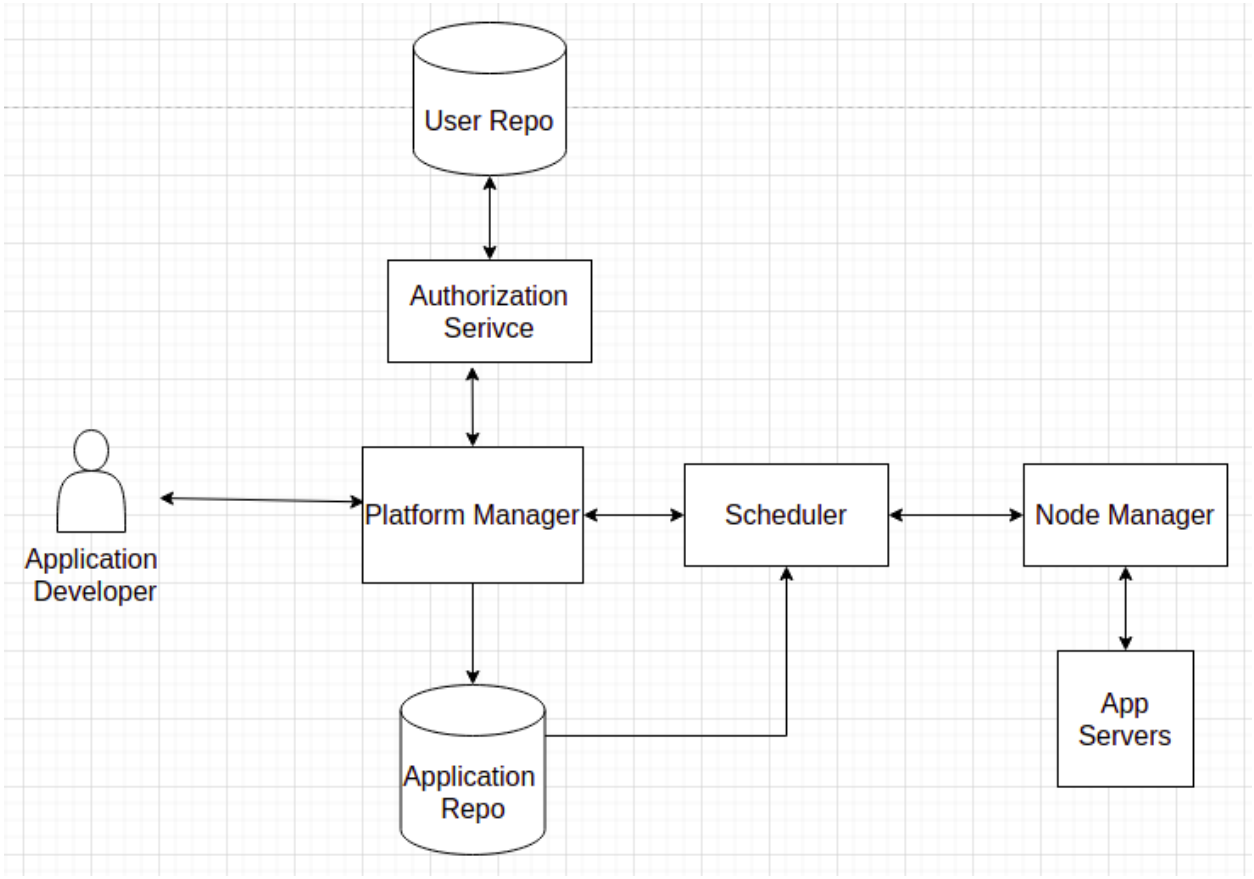
Communication module:
UI - HTTP (client server)
Internal Services (Kafka)

Validator Service

This service checks the folder structure of app.zip file.
Also checks the json format of the files given.

Platform Manager:

The platform can provide a library file to the application developer to import in their code.



Roles and Responsibilities :

1) Provides UI for registering and logging in for the user of the platform(Uses Authorization service)
2) Service to upload or update the respective users application in the Application Repo

3) If the User deploys the application Platform Manager will send the data to the Scheduler to schedule the deployment.

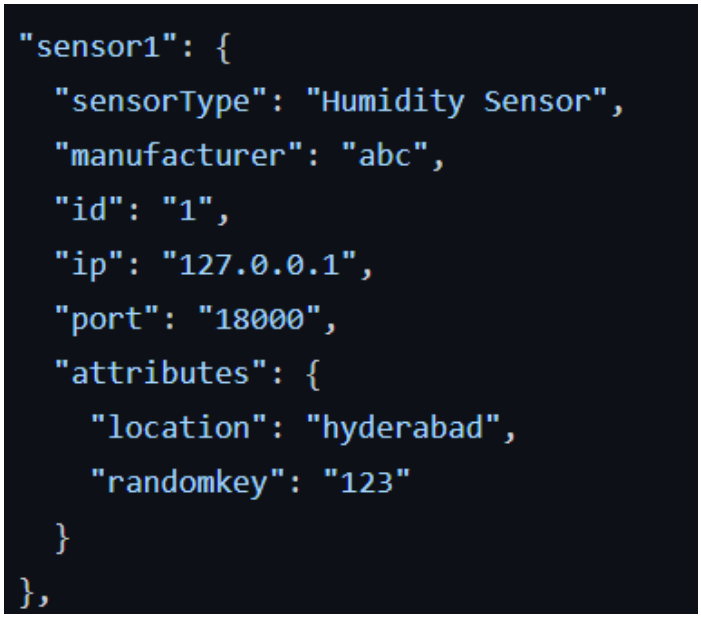
How platform will know about sensor types: (sensortypes.json)

- *Application configurator* will provide catalogue.json which adds a sensor type to the platform.
- This also provides client code to interact with the sensor server.

```
"sensor1": {
  "sensor_name": "Humidity Sensor",
  "sensor_data_type": "float",
  "sensor_location": "None",
  "manufacturer": "abc",
  "ip": "None",
  "port": "None",
  "class_code": "humidityClient.py",
  "functions": {
    "getHumidty": {
      "params": "None",
      "returntype": "float"
    },
    "setSprinkler": {
      "params": "None",
      "returntype": "None"
    }
  }
},
```

How platform will know which sensor to register:(sensor_client_instance.json)

- *Application configurator* will provide sensor_client_instance.json which adds a sensor type to the platform.
- These details of the sensor are saved in database.json.
- During the time of binding we get the sensors from here and the aplgorithm name from the deployConfig.json and the application is binded.



How platform will know which algorithm to run on which sensor:(deployConfig.json)

- The deployer.py file will use the deployConfig.json file where the algorithms are mapped with their sensor instances.
- The sensor details are stored in the database.json file which are passed in the deployConfig file.
- The database.json file is created by the create_sensor_instance.py file.

How platform will know how to deploy an application:(appConfig.json)

- The deployer.py file will check the appConfig file to check the dependencies and environment required for the running of the application.
- There is sensor details registered by the application configurator and their respective class code names which are used to bind to the sensor are also provided.