Action Manager , Fault Tolerance and Bootstrap

Group 1, Team 1

May 8, 2021

Souptik Mondal, Shubhankar Saha, Ramanjaneyulu Payala

Mentor and Professor : Prof. Ramesh Loganathan

TA: Pratik Tiwari, Shubham Agarwal, Jay Krishna

Contents

1 Introduction

One of the most important feature of any platform is monitoring and fault tolerance of that platform. These two are very related concepts. In order to add fault tolerance feature, all the components should be monitored continuously. Our team also implemented the platform initialization feature and scaling concept. When any platform boots up, each of the components should be initialized properly and in a specified order, so that all the dependencies are fulfilled. So, it is also a very important feature of our platform.

2 Action Manager

The Action Manager performs all the necessary actions as requested by the host machines , after they had completed the execution of a service. More specifically it performs 3 different types of tasks :

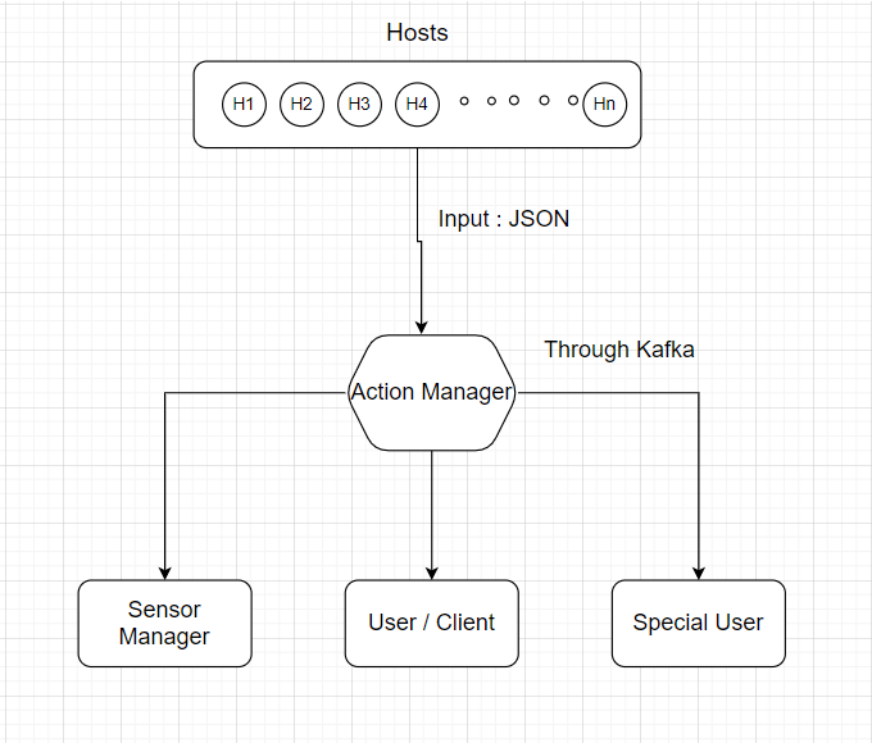

Figure 1: Action Manager

- It will receive instructions from the host machine using kafka_topic named "action_manager".

Then it will parse all the information and differentiate what to do next i.e the instruction is for users or sensor manager.

- It communicates with the sensor manager using a unique kafka_topic for controlling specific controller/sensor.
- It communicates with the user/client who requests the service.
- It notifies via SMS ,email to a list of users as mentioned in the input json file.
- It will send the specific user the email with the message. We have used smtplib. Used SMTP protocol for sending the mail. Connected to smtpserver with default port 587. Provided sender login details to smtpserver. After authentication using sendmail functionality of smtpserver mail is sent to the respective users.

2.1 Technology used:

- The action manager will be a standalone module . It will run inside a docker container.
- It accepts the request/command as a Json.
- It notifies via SMS ,email to a list of users as mentioned in the input json file , inside the action manager module.
- For other communication it will use Kafka.

2.2 Interaction with other modules :

- The action manager will receive information from the host machines. The received information will contain the following details
  - User details like email id, mobile number
  - Commands for the Sensor Manager.
- The action manager will communicate with the sensor manager
  - It will receive instructions from the host machine using kafka_topic named "action_manager". Then it will parse all the information and differentiate what to do next i.e the instruction is for users or sensor manager.
  - It will communicate with the Sensor Manager via kafka_topic named "sensor_manager" and send the commands to the sensor manager. The Sensor Manager will communicate with Sensor Controllers to execute those commands.
- There is a heartbeat function in the Action manager. It will send status to the Monitoring module via a kafka topic named "HeartBeat".

3 Monitoring

This module will continuously monitor all the critical components so that the platform can detect if there is any failure of service or node.

- It is implemented by using heartbeat signal method. In this method we are continuously checking all the components by sending heartbeat signal and analysing the responses from those components.
- This module maintains a run time data structure to store status of all the modules running in the platform.
- It stores an instance of all the modules and again takes another instance after 3 sec. It then matches the status of each module. If there is not any change in any module's state, then it notifies that module name to the fault tolerance module.
- It uses an flask API call "http://fault_tolerance_url/faulty". It will send a json data using a standard format like : { "module_name" : "action_manager" }

3.1 Technology used:

- The Monitoring Module will be a standalone module. It will run inside a docker container.
- For communication with platform modules it will use Kafka.
- For communication with fault tolerance module it will use API.

3.2 Interaction with other modules :

- In monitoring, implemented heartbeat kind of mechanism which keep track of all micro services/modules mainly critical components.
- Basically this module will communicate with each critical modules to get the status of that module for example Dead or Alive. It uses a kafka topic named "Heart_Beat" for this purpose.
- If any module is down it will notify the fault tolerance module using API call.

4 Bootstrap

The bootstrap module is responsible for initializing all the modules in the platform.

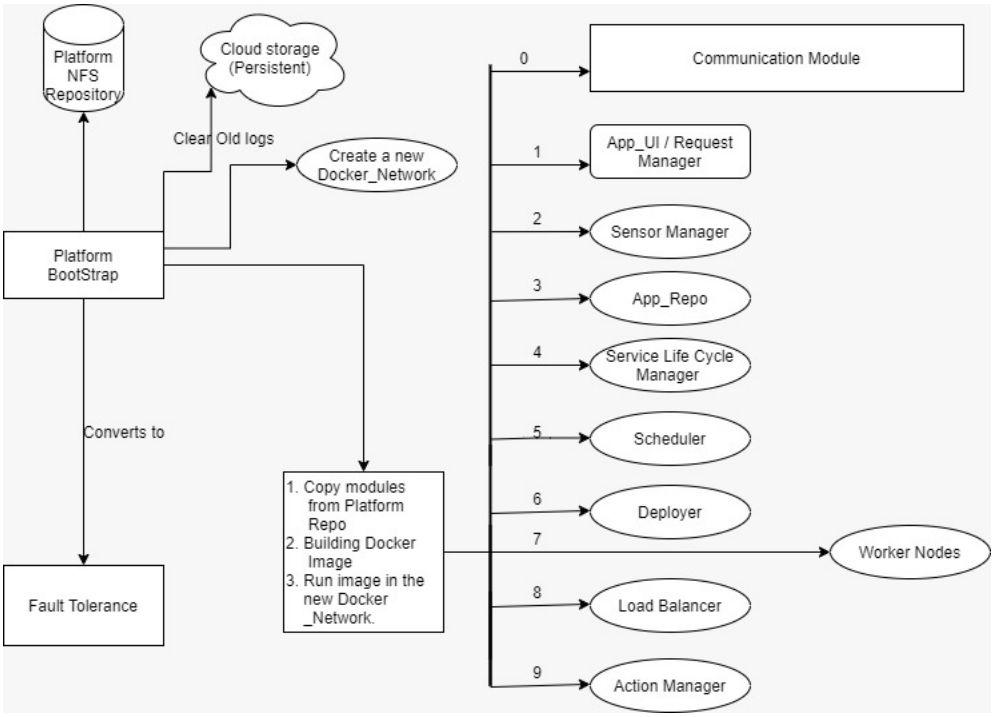

Figure 2: Bootstrap

- This module clears any previous log that are stored in database. We have used mongo DB.
- It creates a new docker network where all the platform modules are to be deployed(run).
- It build the zookeeper and kafka image and puts them in the docker and exposes the relevant ports. Both of the together act as a communication model for the whole platform.
- It will get the modules form the platform repository and then build the corresponding docker images. While doing that it also exposes relevant ports for each module.
- The bootstrap follows a sequence to initialize the modules. The sequence is like :
  - communication module
  - app_ui
  - sensor_manager
  - app_repo
  - service_lcm
  - scheduler

- load_balancer
- worker_nodes
- deployer
- action_manager
- monitoring

5 Fault Tolerance

The fault tolerance module will make sure that if any module or service is down, it will restart it again.

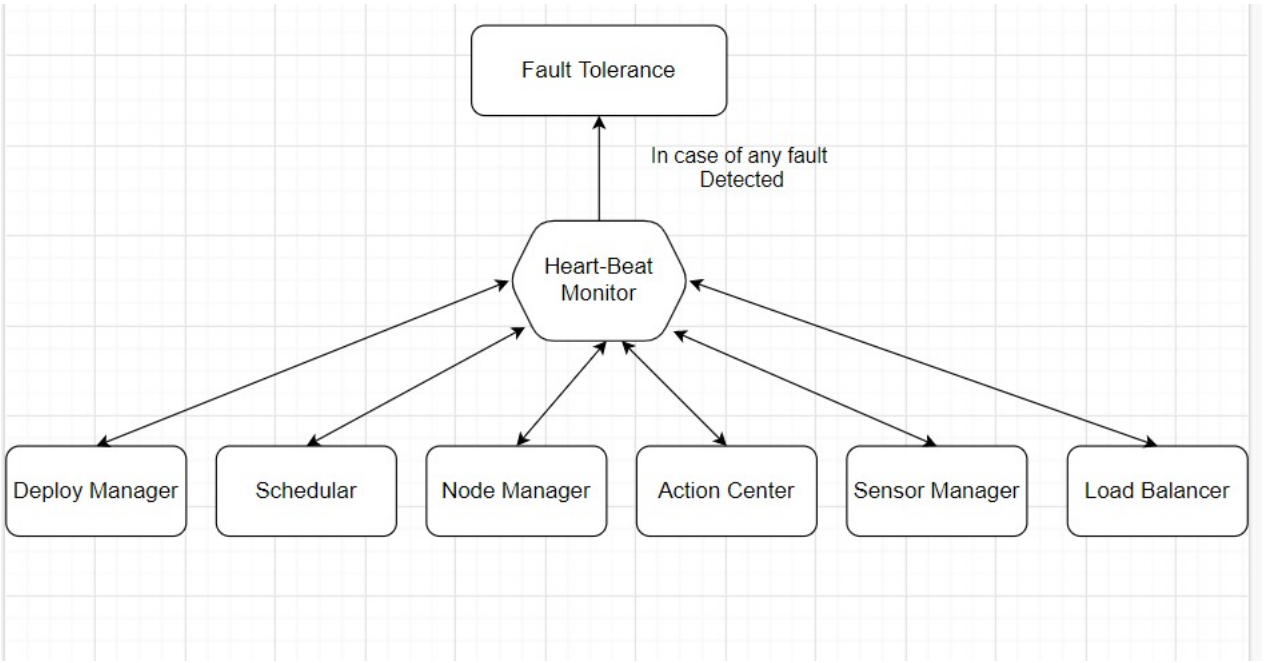

Figure 3: Monitoring

- It checks if it is a platform module or a worker node.
- If any requested module is already running, then it returns the message like the module is already running.
- If the module is not running but the container is still present in the docker network then it removes that container.

- Finally after all these checking it search for an existing image or build a new image of the module.
- It deploys/runs the module and stores the ip of that module in the "ip.json" file and return the success message along with the new ip of that module.
- If the request is for a worker node from the load balancer module, then it sees if the request is for a new worker node.
- if the request is for a new worker node, then it deploys a new worker node and send the details to the load balancer. Hence this is the scaling feature of our platform.

5.1 Technology used :

- The Bootstrap and Fault Tolerance Module will be a standalone module. It will run outside a docker container.
- For communication it will use Kafka and Flask.

5.2 Interaction with other modules :

- The fault tolerance module will communicate with the monitoring module using an API.

6 Scaling

The scaling will add new worker node with the help of load balancer module if required.

- If the load of the running worker nodes are high then the scaling will add new worker node to reduce the load by the help of the load balancer module.
- The new requests will be run on the new worker node.