Server Lifecycle Manager

Service Lifecycle Manager

PREPARED FOR

IAS PROJECT - GROUP 4

IIIT-Hyderabad

PREPARED BY

TEAM 2

Amisha Bansal

Devesh Jha

Parul Ansal

1. Overview

1.1. Introduction:

The goal of this distributed IoT platform is to provide all the generic functionality for an IOT application so that developers can focus more on building features that differentiate the product and add value to them.This project constitutes building an IOT platform which provides functionalities to develop an application involving interaction with multiple sensors, analyzing the data from the sensors and take action on the sensors.

Our module in specific is responsible for maintaining the health of the platform making sure there is no disruption of any kind.

Server Lifecycle Manager:

To make sure application servers are running properly without any interruption or overloading, Server Lifecycle will check the stats of each server regularly. It does so by making sure there is no overload on any server and even if the server crashed due to any reason, it restarts the server immediately by re-running the application.

Service Lifecycle Manager:

There shouldn't be any interruption of any kind in the application running due to failure of any platform service. Service Lifecycle manager will manage all services within the platform and restarts the service if required.

1.2. Scope:

This platform can be used as a host to applications. Application developers can use this platform as an environment to deploy, store and run his application without being bothered. The platform will support a full application lifecycle. The platform and the application deployed is fault tolerant and load balanced.

Functional Overview

2. Server Lifecycle Manager:

It manages servers on the platform.

2.1. Responsibilities:

- Setting up new servers
- Checking the status of servers whether it’s running or not
- Restarting the server if it’s not running
- Stopping the server
- Running the dockerfile received from Deployer.

3. Service Lifecycle Manager:

Service Lifecycle manages all the services in the platform like scheduler, deployer, sensor manager etc. It makes sure that there is no disruption in the platform and within its services.

3.1. Responsibilities:

- It continuously checks the status of every service in the platform.
- Restarts a service if stopped.

4. Interaction between services of platform

Status of services and servers will be shared to the module.

4.1. Server Lifecycle

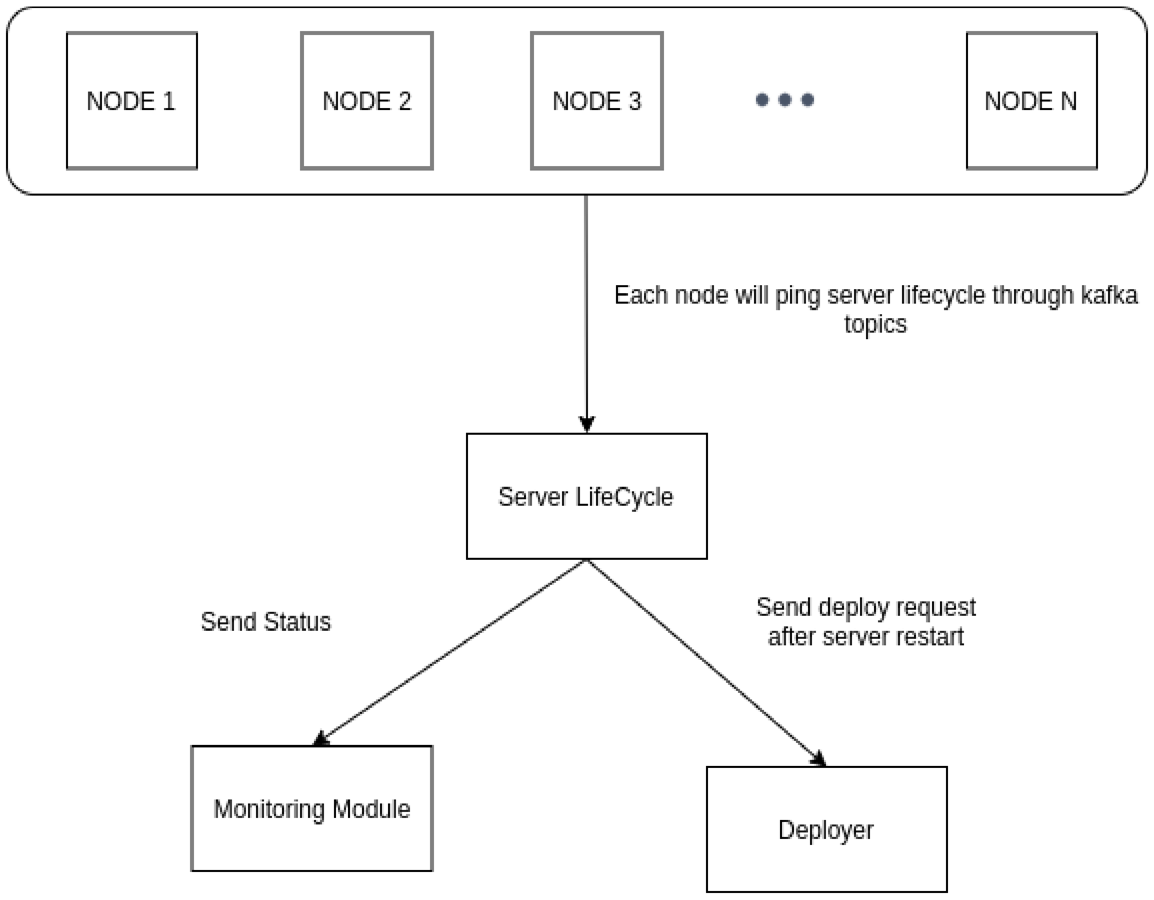
All servers will communicate with Server Lifecycle via kafka topic.

4.2. Service Lifecycle

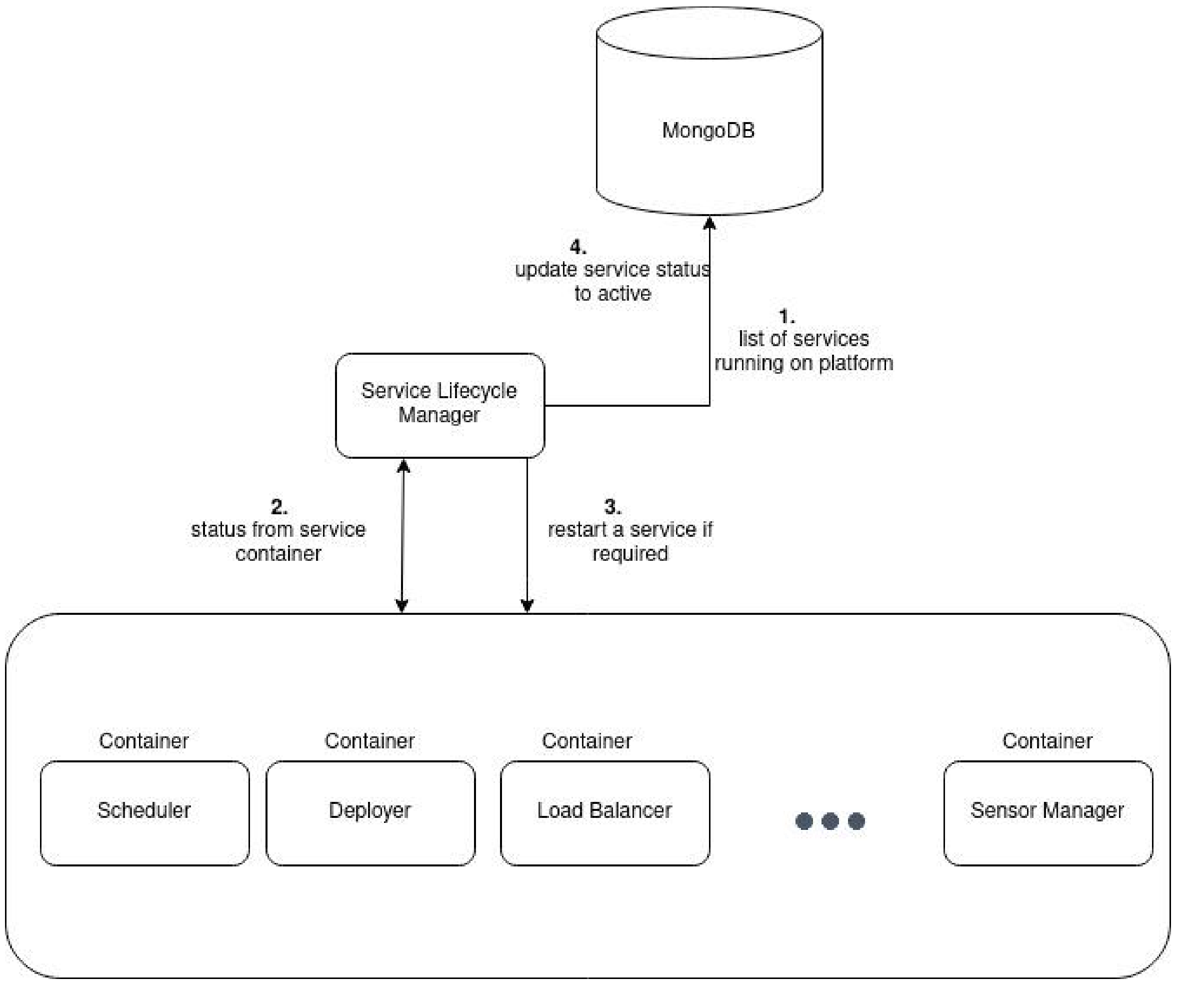
Service Lifecycle will interact with services of the platform to get their status. Container logs of respective service will be checked continuously.

5. Block Diagrams

5.1. Server Lifecycle


NODE 1
NODE 2
NODE 3
NODE N
Each node will ping server lifecycle through kafka topics
Server LifeCycle
Send Status
Send deploy request after server restart
Monitoring Module
Deployer

5.2. Service Lifecycle



6. Communication Model

For tracking servers running over the platform, Kafka has been used as a communication between server lifecycle manager and servers.
A Kafka topic is created for each server and the server will produce status at a particular interval which will be sent to the server lifecycle manager. In case the server stops, the server lifecycle manager will stop receiving status from Kafka topic of a server.

7. Functionalities:

7.1. Fault Tolerance:

Our platform is fault tolerant. Even if a service stops working, it will immediately be deployed again. So there won't be any discontinuation in running applications.

7.2. Persistence:

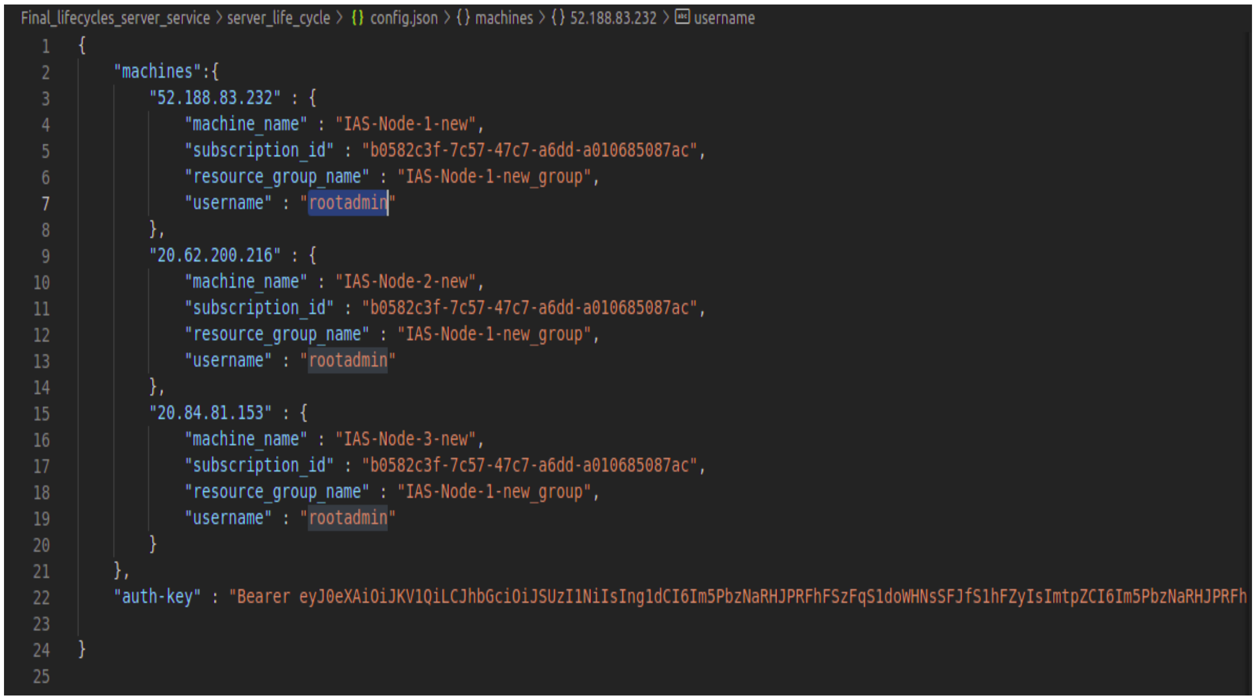
Again our platform is persistent due to fault tolerance, a service or a server will be restarted without hampering the application in running.

7.3. Health management:

Service Lifecycle will take care of the overall health of the platform by continually checking the status of each service. It will restart the service before it affects the application.

8. Configuration file

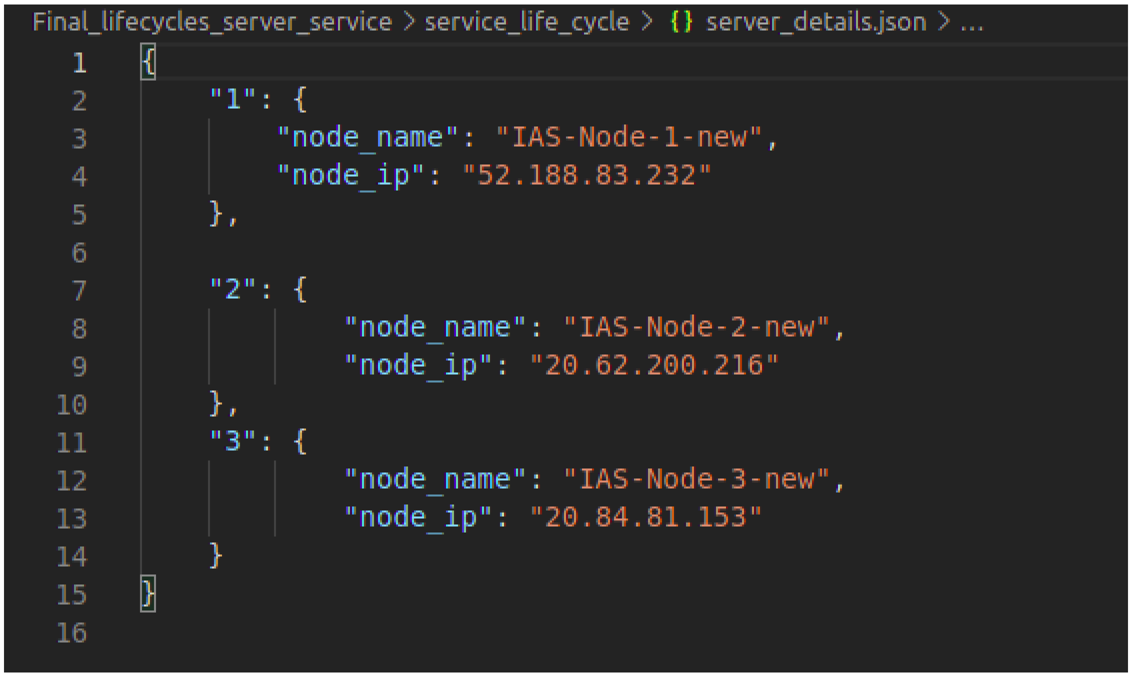
config.json



```
Final_lifecycles_server_service > server_life_cycle > {} config.json > {} machines > {} 52.188.83.232 > username
{
    "machines":{
        "52.188.83.232" : {
            "machine_name" : "IAS-Node-1-new",
            "subscription_id" : "b0582c3f-7c57-47c7-a6dd-a010685087ac",
            "resource_group_name" : "IAS-Node-1-new_group",
            "username" : "rootadmin"
        },
        "20.62.200.216" : {
            "machine_name" : "IAS-Node-2-new",
            "subscription_id" : "b0582c3f-7c57-47c7-a6dd-a010685087ac",
            "resource_group_name" : "IAS-Node-1-new_group",
            "username" : "rootadmin"
        },
        "20.84.81.153" : {
            "machine_name" : "IAS-Node-3-new",
            "subscription_id" : "b0582c3f-7c57-47c7-a6dd-a010685087ac",
            "resource_group_name" : "IAS-Node-1-new_group",
            "username" : "rootadmin"
        }
    },
    "auth-key" : "Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Im5PbzNaRHJPRFhFSzFqS1doWHNsSFJfS1hFZyIsImtpZCI6Im5PbzNaRHJPRFh

}
```

server_details.json



Final_lifecycles_server_service > service_life_cycle > {} server_details.json > ...

```json
{
    "1": {
        "node_name": "IAS-Node-1-new",
        "node_ip": "52.188.83.232"
    },

    "2": {
            "node_name": "IAS-Node-2-new",
            "node_ip": "20.62.200.216"
    },
    "3": {
            "node_name": "IAS-Node-3-new",
            "node_ip": "20.84.81.153"
    }
}

```

9. Test Cases

9.1. Server Lifecycle Manager

9.1.1. A node crashed:

If we don't get a ping from a particular node for a certain amount of time, the server lifecycle will detect it and will restart the node again. If multiple applications were running on a machine that crashed, it will re-deploy each application with the help of load balancer so that there won't be any overload.

9.2. Service Lifecycle Manager

9.2.1. Checking status of each service

Will continuously monitor the health of running services by checking through container logs.

- If a service is no longer running status in DB should be INACTIVE.
- If a service is running status in DB should be ACTIVE.
- After restarting the service, the new container id and the ip of the machine on which it is running should get updated in DB.

10. System Requirement

64-bit OS(Linux)

RAM needed - 4GB(atleast)

Processor: Intel Pentium 11th generation