IAS Project
Team Requirement Report
Deployment Manager

Group 6 Team 2

Submitted By :
AbhiRam DV (2020201030)
Janmejay Singh(2020201089)
Vinaya Sai Revanth Bachu(2020201090)

Table of Contents

Introduction

Server Lifecycle Manager – This module will monitor all the deployment nodes and its status. It initially launches the nodes specified by the platform developer and is responsible for handling the fault tolerance at server level.

Service Lifecycle Manager– This module will be responsible for Listening to requests from scheduler and deploying the applications to the deployment nodes. It also abstracts out the node level deployment details and provides the platform a way to monitor the application status and get the logs of the application.

Intended Use –

- Provide a kafka endpoint to the Scheduler for interacting with the deployer for deployment of application and send stop requests and log requests for application.
- Running the applications in an isolated environment using docker containers and managing the fault tolerance at application level.
- Providing a way for the platform to monitor the application execution status.
- Monitoring the deployment servers and server level fault tolerance.

Assumptions and Dependencies –

- Only the scheduler of the platform will be able to send the requests to the deployer.
- The configuration files need to be in pre-defined format.
- The application to be uploaded on the platform needs to be in a zipped file which also contains the application configuration file and is stored in specified directory in a network shared file system.
- Applications that are being deployed are written in supported languages (Python).

Features and Requirements –

Functional Requirements –

1. **Application Isolation –** The applications that are being deployed should be running in an isolated environment. There must not be any python package conflicts as several applications that can be running in single might need several versions of the same package.
2. **Application Fault Tolerance –** The applications that have been exited abnormally have to be monitored and restarted automatically. However the platform must be able to distinguish between the applications that are abnormally stopped and the ones which the user force stops from execution.
3. **Application Monitoring –** The deployer should be able to monitor the running applications and should be able to serve the requests from the scheduler for getting logs. After an application terminates normally the status must be updated as well.
4. **Deployment Nodes monitoring –** The deployer should be able to monitor the deployment nodes running in the server and must be able to handle the fault tolerance of the servers.

Non-Functional Requirement –

1. **Robustness –** The Deployer should not entirely crash when one / many of the worker nodes goes offline. Instead it should choose one of the running nodes and must deploy the applications there
2. **Security –** Authorization will prevent the users who do not have access rights to see or modify the application. Only authorized users will be able to send the requests to the platform.
3. **Ease of Accessibility –** The web GUI will provide access to the platform through the internet. The Rich Interactive Web GUI will allow users to access the platform more effectively which immensely helps user to see logs and send stop requests for the algorithms.

Systems -

1. Deployer

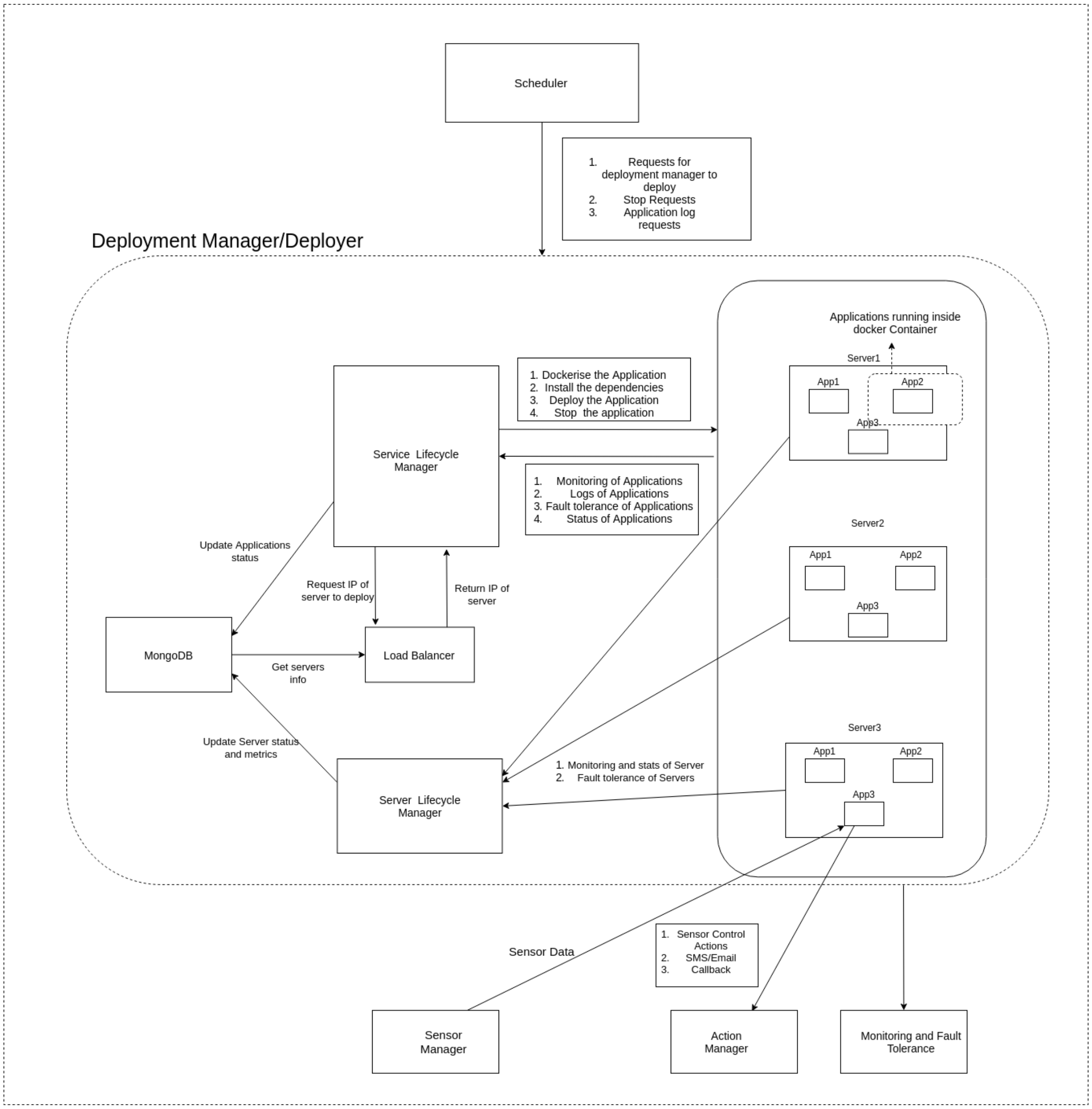
1.1 Functional Overview –

This part of the project deals with handling the deployment requests of the applications uploaded by the user along with authentication and authorization of the user. The interface will be in the form of Graphical User Interface. The developer can sign-up/sign-in from the interface and then upload the code/algorithms of the application. This module will also provide an interface to manage those applications and view the algorithms running on the platform by application developers and end users respectively.

Other than serving the deployments requests, it is also responsible for managing the life cycle of applications which include monitoring the status of the applications and handling the fault tolerance.

All the described functionality will be abstracted out by the deployer and provides the platform a simpler way to interact with the applications.

1.2 Block Diagram –


Scheduler
1. Requests for deployment manager to deploy
2. Stop Requests
3. Application log requests
Deployment Manager/Deployer
Applications running inside docker Container
Server1
App1
App2
App3
1. Dockerise the Application
2. Install the dependencies
3. Deploy the Application
4. Stop the application
Service Lifecycle Manager
1. Monitoring of Applications
2. Logs of Applications
3. Fault tolerance of Applications
4. Status of Applications
Server2
Update Applications status
Request IP of server to deploy
Return IP of server
MongoDB
Load Balancer
Get servers info
Server3
Update Server status and metrics
1. Monitoring and stats of Server
2. Fault tolerance of Servers
Server Lifecycle Manager
Sensor Data
1. Sensor Control Actions
2. SMS/Email
3. Callback
Sensor Manager
Action Manager
Monitoring and Fault Tolerance

1.3 Sub-Systems –

1.3.1 Service Lifecycle Manager – This module handles the Deployment requests made by the scheduler and deploys the application. It is also responsible to monitor the applications and manage their Lifecycle.

1.3.2 **Load Balancer** – This module is responsible for finding out the best node to deploy the application. It looks for relevant stats like cpu usage, and finally returns its ip address to the Service Lifecycle manager to deploy the applications.

1.3.3 **Server Lifecycle Manager** – This module manages the Deployment Node servers and their status. It is responsible for the fault tolerance of the nodes and is responsible to initiate the node with necessary packages and modules.

1.3.4 **Node Api** – The is a module which will be running in the deployment servers. It is responsible to take requests from the service Lifecycle manager and run the applications inside a docker container by installing the specified requirements first.

1.4 Services –

1.4.1 **Deployment requests–** This part of the platform provides access to make requests for deployments by scheduler.

1.4.2 **Stop Requests –** This part of the platform provides a way for scheduler to make stop requests for running applications.

1.4.3 **Get Logs –** It provides a service for retrieving the logs of applications running in the platform.

1.5 Interaction with other parts –

The module will interact with monitoring service to get status of the tasks of an application and display the results.
The module will interact with the scheduler to get algorithms' information which is to be Deployed.
The application running will interact with the sensor manager to get the data of the sensor.

The application might also make requests to the action manager to trigger some actions that deemed necessary.