IAS PROJECT REQUIREMENT DOCUMENT

TEAM 2

Swarnali Dey [2021201088]
Shaon Dasgupta [2021201068]
Ayush Mittal [2021201030]

Introduction

The deployment manager is responsible for deploying the application or the service on the machine. It will receive the configuration file of the application to locate and deploy all the dependencies. It is also responsible for applying the load balancing algorithm and initiating the nodes on the platform to run the service.

Deployment manager is monitoring all the deployment nodes and their status. It is listening to scheduler and deploying the applications to the deployment nodes.

Intended uses - Provides an interface to data scientists and app developer using this data scientist and app developer will upload the required documents. Using the contract file an end point will be created on which model will be running to provide service. And the config file will be used to develop a docker file which will help in deployment of the model.

Using docker file, to deploy applications and AI models on separate machines. This provides an isolated environment and helps to achieve fault tolerance at application level.

Assumptions - The applications and AI models are written/developed in python programming language. The end app developer and the data scientist would provide appropriate contracts that can be used for proper functioning of the models and applications after deployment.

The data scientist would provide a sufficiently trained model for deployment in the platform as the platform does not provide training and testing facilities for AI Models.

Actors -

- Data Scientist - Makes the AI model, should be able to package and upload the model based on contract provided by platform. Platform will avail it as service for applications on the platform.
- Application developer - Develops the application based on the contract provided by platform on how to package application and configuration files. Application will use models available based on contracts provided by data

scientists. App will be based on models, sensor type, controller type available on platform.

Features and Requirements -
Functional Requirements-

- **Application Isolation -** Using docker for deployment of AI models and Applications on different machines provides isolated environment at application level, so that there will be no conflict in requirement of any application/model.

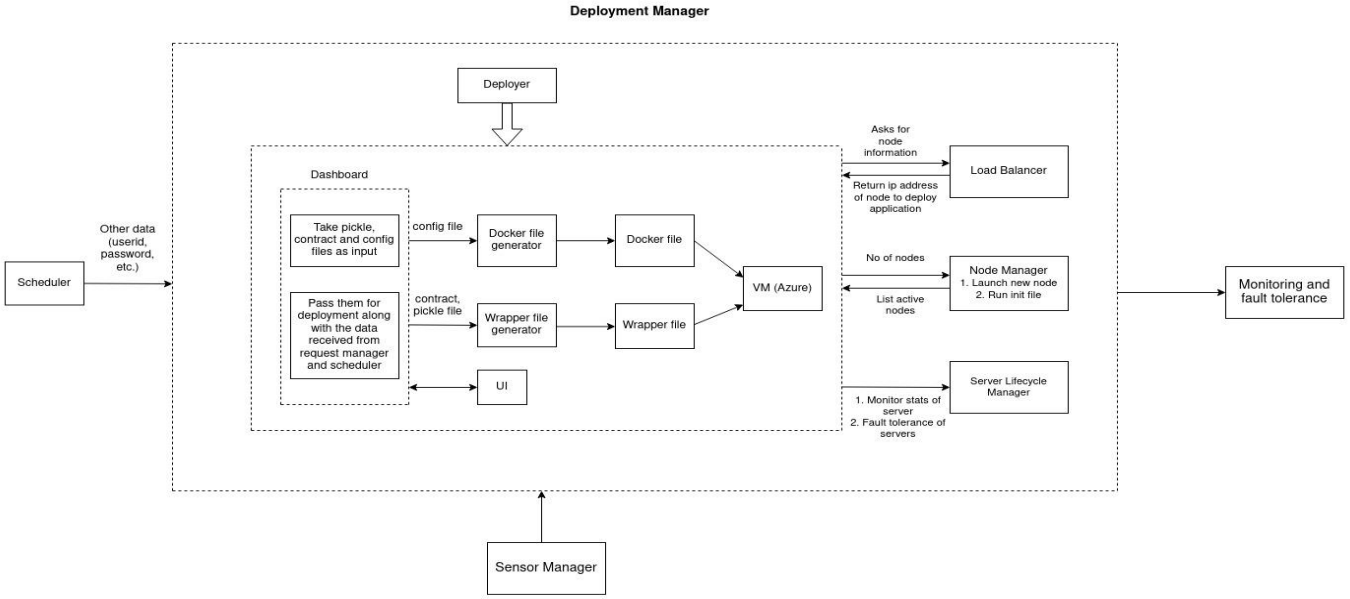
- **Load Balancing -** As the application request is received by the deployer. It gets the free nodes from the platform for deployment. A load balancing algorithm is then applied, and the active workload is calculated for each of the available nodes by noting the CPU usage and the RAM usage of each. In case of high workload, the load balancer requests for more nodes from the manager.

- **Deployment of application/model on the platform -** The deployer sends the request for deploying the application to the platform which is saved in the queue. This job is picked by the scheduler. The Scheduler analyses the configuration information and the algorithms to get a better understanding of the application. It then specifies the start time and duration for which it should run. The deployer demands for a single or more modes to deploy the request. The application is then sent for deployment on the platform and a load balancing algorithm is applied to distribute the workload. The deployer will then have to monitor the health of each node.

Non Functional Requirements-

- **Fault Tolerance -** Deployer also need to manage all the abnormally exited applications and restart them again, with this platform should be able to distinguish between applications which are intentionally stopped by an user and which are actually abnormally stopped.
- **Scalability -** : Scalability will be provided on the platform by allowing more than one instance of an application to run concurrently and letting deployed applications could be re-deployed after scaling or extending it by the entitled developer.
- **Persistence -** The state of the platform that includes the state of applications scheduled and the sensors is saved when the platform shuts-down and is restored when the platform starts up again.

Block Diagram -



Systems -

- **Deployer:** The deployer takes pickle, config, contract files as input and generates the docker file and wrapper file to be deployed on the VM (Azure). The UI will be used for uploading the pickle, config, contract files in case of data scientist and source files, config and contract files, in case of application developer.
  Other than serving the deployment requests, it is responsible for monitoring the applications and managing their life cycles.

- **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 the chosen node's ip address to the deployer for deployment of applications.

- **Server Lifecycle Manager**: This module manages the deployment node servers and their status. It is responsible for the fault tolerance of the nodes and initiating the nodes with necessary packages and modules.

- **Node Manager:** It is responsible for initiating node instances as requested by the deployment manager. It is also responsible for init file generation which helps in communication with the scheduler, deployer and sensor manager.

Services -

- **Upload and deployment of AI models and Applications -** It provide separate interface for AI developer and App developer to upload required files for deployment.
- **Rest End points for App developers to use AI model -** based on contract provided by AI developer an rest end point is created in the form of wrapper class. App developer can send data in specified formate and these end point will return the predicted result.
- **Provides required environment -** Based on the config file, separate environment is created in the form of containers for respective application and model to run.
- **Health check of deployed application ans models -** it checks it all the deployed applications and models and working or not, at regular intervals.

Interaction with other parts -

The deployment manager will interact with the scheduler to get details like userid, password and scheduler information. It will interact with the monitoring service to get status of the tasks of an application and display the results. The running application will interact with the sensor manager to get data from the sensor.