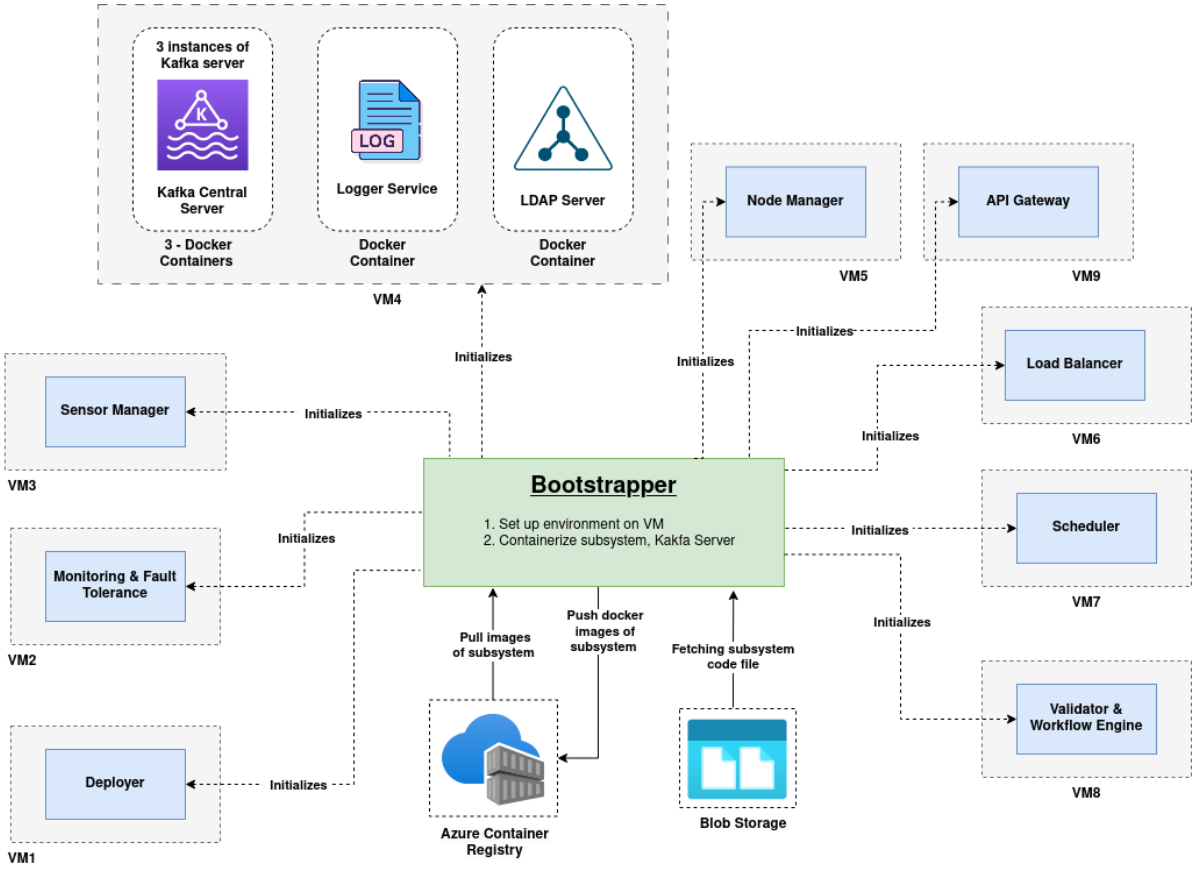
BOOTSTRAPPER SUB-SYSTEM

Introduction of Bootstrapper

- Boostrapper basically is responsible for deploying all the subsystems code-base of our IOT platform.
- It initialises nodes/VM, does env setup on each VM/node, starts kafka central server, starts containers of each sub-system on respected VM?node.
- Stores all the information of each subsystem/service deployed by bootstrapper, like on which node/Vm inside which container which sub-system is running on which port in the service registry.

Boostrapper diagram



Command to run Bootstrapper: Download bootstrapper from github first and then execute **python3 main.py**

Working of Bootstrapper

1. It initialises the kafka central server on a specified node. It stores IP and port of Kafka Central server in service registry which can be used by other

subsystems. Docker-compose file is being used for running Kafka central server with 3 brokers running one on each container and red-panda UI on the fourth container.

2. Do registration of the terminal where bootstrapper is running to access other node terminals using ssh.
3. Inserting Node info like ip, pass, user_name, status stored in node_info.py into the nodeDB collection.

```
# done
#sshpass -p Jeet@deployer ssh jeetdeployer@20.2.81.4
node_1 =  {
        "ip" : "20.2.81.4",
        "password" : "Jeet@deployer",
        "user_name" : "jeetdeployer",
        "node_name": "node_1",
        "status": "active"
}

#done
#sshpass -p Abcdefgh@1234 ssh prannema@20.127.0.89
node_2 = {
    "ip" : "20.127.0.89",
    "password" : "Abcdefgh@1234",
    "user_name" : "prannema",
    "node_name" : "node_2",
    "status": "active"
}
```

4. Inserting the configuration details stored in config.py into platform_config collection.

```
platform_config = [
    {
        "name" : "logger-db",
        "server_url": "mongodb+srv://pranshu_mongo:iasproject@cluster0.svcqjdj.mongodb.net/?retryWrites=true&w=majority",
        "db_name": "LOGS"
    },
    {
        "name" : "platform-db",
        "server_url": "mongodb+srv://pranshu_mongo:iasproject@cluster0.svcqjdj.mongodb.net/?retryWrites=true&w=majority",
        "db_name": "IAS_test_1"
    },
    {
        "name" : "monitoring-db",
        "server_url": "mongodb+srv://pranshu_mongo:iasproject@cluster0.svcqjdj.mongodb.net/?retryWrites=true&w=majority",
        "db_name": "M_FT"
    },
    {
        "name": "acr_info",
        "user_name":  "testimages01",
        "password": "EnQFZBylKmDFlOEPuf1LQ3ZYvKxxlbb1Qd8uYdXGQw+ACRBcl3xB",
        "login_server": "testimages01.azurecr.io"
    },
    {
        "name": "blob_storage",
        "server_url": "https://applicationrepo326.blob.core.windows.net/",
        "container_name": "appdb"
    }
]
```

BOOTSTRAPPER SUB-SYSTEM

5. Do environment setup on each node by executing node_env_setup.sh(copy file to the resp node using scp command) file on the terminal of the respected node where we want to do env setup.
6. Now deploy each subsystem by reading the platform_info.py file. Steps performed to deploy each sub-system as as follows:

```
"deployer" : {
    "port" : 8006,
    "node_info" : node_info.node_6
},
"logger" : {
    "port" : 8002,
    "node_info" : node_info.node_1
},
"api-gateway": {
    "port": 8013,
    "node_info": node_info.node_5
},
"node-manager" : {
    "port" : 8004,
    "node_info" : node_info.node_4

},
"load-balancer" : {
    "port" : 8005,
    "node_info" : node_info.node_4

},
```

   a. Download the zip file from blob-storage, unzip it, and generate a Dockerfile of that folder by reading the config.json file inside that folder.
   b. Now build a docker image by executing Dockerfile inside the folder.
   c. Push the generated image to Azure Container Registry (ACR).
   d. Store ACR image path and port of the code on which it is listening.
   e. Take the ssh connection of the node where we want to deploy a respected sub-system.
   f. Pull the docker image from ACR and run it.
   g. While running the docker image we will pass three env variables to each container node_name(node on which the sub-system container is running), container_name (name of container inside which sub-system is running), container_start_time.
   h. While running docker image of deployer sub-system apart from passing env variables we also do volume mounting.
      i. -v <path_of_node>:<path_inside_deployer>
      ii. **-v $(pwd)/platform-deployer/service:/deployer/services**
   i. Store all the info of each sub-system deployed into the service registry

j. Service registry mongo-db entry

```
'''
service_info_entry = {
  "app_name" : "",
  "service_name": "",
  "app_id": "",
  "port": "",
  "container_up_time": "",
  "container_name": "",
  "container_id": "",
  "node_name": "",
  "ip": ""
}
'''
```

k. nodeDB mongo-db entry

```
'''
node_info = {
    "user_name": "",
    "ip": "",
    "password": "",
    "node_name": "",
    "status": "active/inactive"
}
'''
```

Work That Can Be Done

- Make one boostrapper.py file which will download bootstrapper from blob-storage, unzip it and then execute command **python3 main.py**.

SSH Commands

- Install sshpass:
  - **sudo apt-get install sshpass**
- Register our terminal to the target node
  - **ssh-keyscan -H <ip_of_target_node> ~/.ssh/known_hosts**
- ssh connection:
  - **sshpass -p <password> ssh <username>:<ip>**
  - **ssh -i PATH_TO_PRIVATE_KEY USERNAME@EXTERNAL_IP**

BOOTSTRAPPER SUB-SYSTEM

- copy file/folder from current machine to some node:
    - **sshpass -p <password> scp -r <source_path> jeetdeployer@20.2.81.4:~/<destination_path_after_root_dir_of_target_node_where_we_want_to_copy_file_or_folder>**