FM SETUP INSTRUCTIONS

Index

FM Installation

FM installation prerequisites

1. Install docker
2. Install docker-compose
3. Works only on x86 arch

Setup FM

1. Clone fleet manager repository

```
git clone https://github.com/AtiMotors/fleet_manager
cd fleet_manager
git pull
git submodule update --init --recursive
```

2. Checkout to release/branch, update mule submodule.

```
git checkout <branch>
git submodule update --remote
git submodule update
```

3. Setup cert files - You will need python installed in your machine to carry out this step

   3.1 *Update all_server_ips, http_scheme in static/fleet_config/fleet_config.toml, add wireguard ip of the server as well if wg access is present*

```
for example:
    all_server_ips=["192.168.6.11", "10.9.0.168", "127.0.0.1"]
```

```
    http_scheme="https"
```

3.2 *Install toml, crytography in your machine(not server)- these packages are required to generate cert files*

```
pip install toml cryptography
```

3.3 *Upon successful installation of the above mentioned packages, run setup_certs.py to generate FM cert file*

```
cd utils && python3 setup_certs.py
../static/fleet_config/fleet_config.toml ../static
```

3.4 *Copy the cert file(static/certs/fm_rev_proxy_cert.pem) to all the sherpas(/opt/ati/config/fm_rev_proxy_cert.pem)*

4. Update static directory with map_files, sherpa_details

4.1 *Add fleet names, customer details, server_ip to the static/fleet_config/fleet_config.toml*

```
server_ip="xyz"
fleet_names=["sample_fleet", "sample_fleet_1"]
customer="xyz"
site="xyz"
location="xyz"
```

4.2. *Add sherpa details of all the sherpas to the fleet_config following the sample given below- make sure sherpa names match hostname in the mule*

```
[fleet_sherpas.<sherpa_name>]
hwid="abc"
api_key="qZhoteD9zOHn_wBoYW04vgeaiLSBIoWP_jaVy5TQLp0_T30-789PAI"
fleet_name="sample_fleet"

[fleet_sherpas.<sherpa_name>]
hwid="xyz"
api_key="ffZhoteD9zOHn_wBoYW04vgeaiLSBIoWP_jaVy5TQLp0_T30-789PAI"
fleet_name="sample_fleet_1"
```

4.3. *Create map folders, map_files.txt for all the fleet names present in fleet_config.toml. Make sure grid_map_attributes.json is present*

```
mkdir static/sample_fleet/map/
copy all the map files to sample_fleet/map/
cd sample_fleet/map/
ls > map_files.txt
```

5. If server has internet, allows you to download open-source packages (Recommended to use step 6 instead of this step)

   a. If you want to setup fm on a remote location, run push_fm script to create all the docker images on the server

   ```
   ./scripts/push_fm.sh -WDi username@ip
   ```

   b. If you want to setup fm on your machine, run push_fm script to create all the docker images on your machine

   ```
   ./scripts/push_fm.sh -WD
   ```

6. If server doesn't have internet access, copy built docker images to the server from Ati server(data@192.168.10.21:/atidata/datasets/FM_v2.1_docker_images), run the following commands

   a. Load base images on server/localhost

   ```
   ssh username@ip
   cd FM_v2.1_docker_images
   bash load_docker_images.sh
   exit
   ```

   b. If you want to setup fm on a remote location, run push_fm script from your machine to create all the docker images on the server

   ```
   ./scripts/push_fm.sh -WbDi username@ip
   ```

   c. If you want to setup fm on your machine, run push_fm script from your machine to create all the docker images on your machine

   ```
   ./scripts/push_fm.sh -WbD
   ```

7. Setup plugins if any.
8. Setup sherpas.
9. Setup optimal dispatch config
10. Push mule docker image to local docker registry

Start or Restart FM

1. Modify timezone if required by setting environment variables TZ, PGTZ in services fleet_manager, db enlisted in static/docker-compose.yml.
2. Restart FM

```
cd static
docker-compose -p fm down
docker-compose -p fm up
```

3. Use FM through UI, if running FM on localhost use ip as 127.0.0.1

```
https://<ip>/login
username: admin
password: 1234
```

3. Induct all the sherpas that you want to use
   a. Press enable for trips button from sherpa card
   b. Only those sherpas that has been enabled for trips will get assigned with a trip

Run FM Simulator

a. Follow Setup FM , steps 1-2

b. Set simulate in static/fleet_config/fleet_config.toml

```
simulate=true
http_scheme="http"
```

c. Make sure all the station below mentioned tags in grid_map_attributes.json pertaining to all the fleets(<fleet_name>/map/grid_map_attributes.json). Tags like conveyor, auto_hitch, auto_unhitch doesn't work in simulator mode.

```
 "station_tags": [
  "parking",
  "dispatch_not_reqd"
 ]
```

d.Follow Setup FM, steps 3-5

Setup sherpas

a. Copy fm cert file(fm_rev_proxy_cert.pem) generated in Setup FM step 3 to sherpa's /opt/ati/config directory

b. Add this patch to /opt/ati/config/config.toml in the mule

```
[fleet]
api_key = " "
chassis_number = " "
data_url = "https://<fm_ip_address>:443/api/static"
fm_ip = "https://<fm_ip_address>:443"
ws_url = "wss://<fm_ip_address>:443/ws/api/v1/sherpa/"
fm_cert_file="/app/config/fm_rev_proxy_cert.pem"
```

c. Setup/update ati_mule_maintenance service

```
git clone https://github.com/AtiMotors/system

copy latest mmts_utils.sh, ati_mule_maintenance.sh from ati_core folder to
/etc/systemd directory in sherpa
copy ati_mule_maintenance.service from ati_core folder to
/etc/systemd/system directory in sherpa

#stop the ati_mule_maintenance service and delete maintenance fifo file
ssh into mule
sudo systemctl stop ati_mule_maintenance
sudo systemctl disable ati_mule_maintenance
sudo rm /opt/ati/run/maintenance_req_fifo

#start maintenance service
ssh into mule
cd /etc/systemd
sudo chmod ugo+rwx mmts_utils.sh
sudo chmod ugo+rwx ati_mule_maintenance.sh
sudo systemctl enable ati_mule_maintenance
sudo systemctl start ati_mule_maintenance

#enable 443 port
sudo ufw allow 443
```

d. Setup mule nginx container (if not already present)

1. Check if mule nginx container is running: (below command should show container running)

```
ssh into mule
docker ps | grep mule_nginx
```

2. Build container (if nginx container is not present):

- Make a new folder, shell_scripts

```
cd /opt/ati
mkdir shell_scripts
```

- Copy load_mule_nginx.sh file from server(data@192.168.10.21:/atidata/datasets/FM_v2.1_docker_images/load_mule_nginx.sh) to mule (/opt/ati/shell_scripts). (DO NOT copy this script to /opt/ati folder on mule)

- Run load_mule_nginx.sh

```
cd /opt/ati/shell_scripts
bash load_mule_nginx.sh
```

- Restart mule docker

```
docker restart mule
```

e. Copy cert files for docker pull

```
sudo mkdir /etc/docker/certs.d/<fm_ip>:443
sudo cp /opt/ati/config/fm_rev_proxy_cert.pem
/etc/docker/certs.d/<fm_ip>:443/domain.crt
```

Setup Plugin

a. Setup IES

b. Setup conveyor booking

Setup IES

a. Add IES plugin to static/fleet_config/plugin_config.toml

```
all_plugins=["ies"]
```

b. Modify static/plugin_ies/locationID_station_mapping.json file. Map IES station names to corresponding ati station names as the template indicates.

```
{
    "Warehouse_Pick": "ECFA start",
    "HP02_FA02": "ECFA-2",
    "HP03_FA01": "ECFA-1",
}
```

Setup conveyor booking

a. Add conveyor plugin to static/fleet_config/plugin_config.toml

```
all_plugins=["conveyor"]
```

b. Modify static/plugin_conveyor/api_key_conveyor_mapping.json. Map api keys to conveyor station names, also specify the nearest chute station.

```
{

"E2bKHiYNMk5kCvSKZfOVThr5t8oUQ_8mrot36QVrk9E_CONV1": {"name": "Conveyor1",
"nearest_chute": "Meeting Room 1"},

"B2bKHiYNMk5kCvSKZfOVThr5t8oUQ_8mrot36QVrk9K_CONV2": {"name": "Conveyor2",
"nearest_chute": "Meeting Room 2"}

}
```

Setup optimal dispatch config

Optimal dispatch logic tries to allocate the pending trips with the best sherpa available. Choice of best sherpa is made with the paramter Z

$\Z=(eta)^a/(priority)^b$
$priority=p1/p2$

```
where,
    eta - expected time of arrival for sherpa pose to first station of the
trip booked
    a - eta power factor , 0<a<1,
    priority - measure of how long a trip has been pending,
    p1 - Time since booking of currrent trip,
    p2 - Minimum of time since booking across all the pending trips,
    b - priority power factor , 0<b<1,
```

1. **Maximise number of trips done**: To get maximum number of trips done in a given time frame eta_power_factor can be set to 1, priority_power_factor can be set to 0. This will make the optimal disaptch logic to lean towards trips that can be started faster. The trip booking order will not be followed.

```
[optimal_dispatch]
method="hungarian"
prioritise_waiting_stations=true
eta_power_factor=1.0
priority_power_factor=0.0
```

2. **Fair scheduling**: To configure optimal dispatch logic to take trips in the order they were booked eta_power_factor can be set to 0, priority_power_factor can be set to 1.

```
[optimal_dispatch]
method="hungarian"
prioritise_waiting_stations=true
eta_power_factor=0.0
priority_power_factor=1.0
```

3. **Custom configuration**: There is no ideal combination of eta_power_factor, priority_power_factor. They should be choosen according to the frequecy of trip bookings, route length between the stations to maximise the throughtput.

4. For good takt time, eta power factor should be higher, for fair scheduling priority power factor should be set higher.

Push mule docker image to local docker registry

1. Copy mule docker image tar file to fm_server and load the image

```
docker load -i <mule_image tar file>
```

2. Tag mule image with registry ip, tag

```
docker tag mule:<mule_tag> <fm_ip>:443/mule:fm
```

3. Setup certs for docker push

```
sudo mkdir /etc/docker/certs.d/<fm_ip>:443
sudo cp /opt/ati/config/fm_rev_proxy_cert.pem
/etc/docker/certs.d/<fm_ip>:443/domain.crt
```

4. Push mule docker image to FM local registry

```
docker push <fm_ip>:443/mule:fm
```