

FM SETUP INSTRUCTIONS

Index

1. [Setup FM with push_fm script](#)
2. [Setup FM by copying built docker images](#)
3. [Start/Restart FM](#)
4. [Run FM Simulator](#)
5. [Setup sherpas](#)
6. [Setup plugin](#)
7. [Setup optimal dispatch config](#)
8. [Push mule docker image to local docker registry](#)
9. [Fleet maintenance](#)

FM Installation

FM installation prerequisites

1. Install docker(<https://docs.docker.com/engine/install/>)
2. Install docker-compose(<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04>)
3. Works only on x86 arch

Setup FM with push_fm script

1. Clone fleet manager repository and setup git config for submodule

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

# open and edit .git/config file, add branch=dev to submodule mule entry
[submodule "mule"]
url = https://<token>@github.com/AtiMotors/mule.git
active = true
branch = dev
```

git submodule update

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, cryptography 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

4.1. Create map folders for each of the fleets

```
mkdir static/fleet_1/map/
copy all the map files of fleet_1 to static/fleet_1/map/

mkdir static/fleet_2/map/
copy all the map files of fleet_2 to static/fleet_2/map/
```

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 -Wi 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 -W
```

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_v<fm_version>_docker_images), run the following commands

- a. Load base images on server/localhost

```
ssh username@ip  
cd FM_v<fm_version>_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 -Wbi 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 -Wb
```

7. [Setup plugins](#) if any.
8. [Setup sherpas](#).
9. [Setup optimal dispatch config](#)
10. [Push mule docker image to local docker registry](#)
11. To start using fleet_manager, follow [Start or Restart FM](#)

Setup FM by copying built docker images

1. Copy built docker images to the FM server from Ati server(data@192.168.10.21:/atidata/datasets/FM_v<fm_version>_docker_images)

2. Load docker images

```
cd FM_v<fm_version>_docker_images  
bash load_docker_images.sh
```

3. Backup the current static directory if already present. Copy "static" directory from fleet_manager repository/FM_v<fm_version>_docker_images folder to the FM server, update it with the info from backup.
4. Copy docker-compose.yml from <fm_repository>/misc/ or FM_v<fm_version>_docker_images folder to the static folder.
5. Create cert files if not already present by following [Setup FM with push_fm script](#) steps 1-3.
6. Copy the cert files generated (fm_rev_proxy_cert.pem, fm_rev_proxy_key.pem) to the static/certs/ directory in the FM server
7. Follow steps 7-10 in [Setup FM with push_fm script](#)
8. To start using fleet_manager, follow [Start or Restart FM](#)

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. Start/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
```

4. Addition/Deletion of fleets, sherpas should be done through Configure page on the dashboard. Adding it to fleet_config.toml will have no effect. Fleets names have to be same as map_names. Copy the map files to the static directory on FM server by following [Setup FM with push_fm script](#) step 4 before trying to add it through dashboard.

5. Please restart FM using docker-compose commands step 2, after adding sherpas/fleets.
6. 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
7. Follow [Fleet maintenance](#) if needs be

Run FM Simulator

- a. Follow [Setup FM with push_fm script](#), steps 1-2
- b. Set simulate in static/fleet_config/fleet_config.toml

```
[fleet.simulator]
simulate=true

[fleet]
http_scheme="http"
```

- c. To get trip bookings done automatically add routes(list of station names), trip booking frequency(seconds) to fleet_config.

```
[fleet.simulator.routes]
route1 = [["Station A", "Station B"], [10]]
route2 = [["Station A", "Station C"], [60]]
```

- d. Make sure all the stations mentioned in gmaj file(<fleet_name>/map/grid_map_attributes.json) has only the below mentioned tags. Tags like conveyor, auto_hitch, auto_unhitch will not work in simulator mode.

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

- e. If you want to start sherpas at particular station add this patch to config

```
[fleet.simulator.initialize_sherpas_at]
sample_sherpa="Station A"
```

- f. If you want to simulate transit visas set visa handling in fleet.simulator config

```
[fleet.simulator]
visa_handling=true
```

g. Follow remaining steps in [Setup FM with push_fm script](#), steps 3-7. [Setup Sherpa](#) not required for simulation

Setup sherpas

- a. Copy fm cert file(fm_rev_proxy_cert.pem) generated in [Setup FM with push_fm script 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"
http_url = "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 mmits_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 mmits_utils.sh
sudo chmod ugo+rwx ati_mule_maintenance.sh
sudo chmod ugo+rwx /opt/ati/uniflash
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_v<fm_version>_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
```

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.

```
{
  "E2bKHiYNMk5kCvSKZf0VThr5t8oUQ_8mrot36QVrk9E_CONV1": {"name": "Conveyor1",
    "nearest_chute": "Meeting Room 1"},

  "B2bKHiYNMk5kCvSKZf0VThr5t8oUQ_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 parameter \$Z\$

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

where,

eta - expected time of arrival computed for the sherpa to reach the first station of the trip booked,
 priority - measure of how long a trip has been pending,
 p1 - Time since booking of current trip,
 p2 - Minimum of time since booking across all the pending trips,

```
a - eta power factor , 0<a<1,
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 dispatch 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 chosen according to the frequency of trip bookings, route length between the stations to maximise the throughput.
- For good takt time, eta power factor should be higher, for fair scheduling priority power factor should be set higher.
- Sherpas can also be restricted from running on certain routes/station by setting up exclude_stations config. The below config will stop "sample_sherpa"(sherpa_name) from getting assigned with the trips with route having any of ["Station A", "Station B"]

```
[optimal_dispatch.exclude_stations]
sample_sherpa=["Station A", "Station B"]
```

Push mule docker image to local docker registry

- 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 on fm server

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

3. Setup certs for docker push on fm server

```
sudo mkdir /etc/docker/certs.d/<fm_ip>:443
sudo cp <fm_static_dir>/certs/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
```

Fleet maintenance

Update map files

1. Copy all the new map files to <fm_static_directory>/<fleet_name>/map/ folder
2. Select the fleet which needs the map update from the webpage header in the dashboard and press update_map button on the webpage header(present along with start/stop fleet , emergency_stop fleet etc.)
3. Restart of FM is not required - for map updates

Swap sherpas between fleets

1. Delete the current sherpa entry and then again add it to a different fleet. Sherpa's can't be swapped directly. FM restart would be required post addition of sherpa to the new fleet.

Generate api keys for sherpas/conveyor/summon_button/any hardware

1. Run utils/api_key_gen.py in utils directory in fleet_manager - You will fleet_manager repository access, python installed in your machine to run this. Python dependecies required: secrets, click

```
cd <path_to_fleet_manager_repository>/utils
python3 api_key_gen.py --hw_id <unique_hwid>
```

Add/Remove frontendusers

1. Run utils/gen_hashed_password.py in utils directory in fleet_manager - You will fleet_manager repository access, python installed in your machine to run this. Python dependecies required: hashlib,

click

```
python3 utils/gen_hashed_password.py --password <password>
```

2. The generated hashed password can be added to
<fm_static_directory>/fleet_config/frontend_users.toml

```
[frontenduser.<new_user>]  
hashed_password=<hashed password>
```

3. Remove unwanted entries from <fm_static_directory>/fleet_config/frontend_users.toml if any.