Note: DO NOT run the commands in grey. Only execute the commands in red.

Accessing your devices(

Each team has been assigned one server container and 2 or 3 raspberry Pis, which will host the clients containers.

In the sheet shared with you, you will notice two to three ips addresses, one for the server and one for the MQTT container, that has been predeployed for your convenience.
(Example IP's Not for use)

DEVICE CONFIGURATION	DEVICE IP	Containers	Student Names		Server Container IP:port	MQTT IP:Port	Network
Raspberry Pi 4B - 2GB	10.24.24.125	4	A	D	10.24.24.32:2222	10.24.24.32:1884	flotilla-net-32
Raspberry Pi 4B - 8GB	10.24.24.170	4	B	E			
			C	F			

ssh into the Raspberry Pis:

```
ssh wschool@<DEVICE IP>MQ
```

Password: wschool

Once you have ssh-ed, your home directory should look like this:

```
wschool@pi-4b-5:~ $ ls
data  flotilla-client-arm.tar.gz  models  run_client.sh
```

The flotilla-client-arm.tar.gz is a tarball of the docker image we are supposed to run.

Load this docker image:

```
docker load -i flotilla-client-arm.tar.gz
```

It might take a while(~15min) to load, so be patient! Check if the image has loaded properly:

```
docker image ls
```

If the "flotilla-client" appears in images, you're all set.

```
IMAGE                ID             DISK USAGE   CONTENT SIZE   EXTRA
flotilla-client:arm  288afa8cb49b       3.04GB         1.48GB   U
```

Setup the Network on the Pis:

On the server we initialize the swarm with the following command:

```
docker swarm init --advertise-addr <server ip>
```

On the server, a network overlay is running that allows the containers in the swarm to message each other. The command is something like the following and gives 256 ips. Name of the network will be flotilla-net(or whatever you want). The mqtt, the server and all the client containers will attach to this network:

```
docker network create --driver overlay --attachable
--subnet=10.0.9.0/24 <network-name>
```

The mqtt containers can then be brought up:

```
docker compose -f <mqtt-compose>.yaml up -d
```

Your Pi's need to join this swarm. The 16 teams are divided equally among 2 server. Check your server IP from the sheet and then execute the following T1-T8, T9-T16 commands on all the pi's assigned to your group. The client containers can do this(Automatically) by passing the token the server generates when the swarm is made

For T1 to T8 please use:

```
docker swarm join --token
SWMTKN-1-3uy7wsk54037m7rzt6toxmdwrbpmb8upwe643mtn98gynqtv91-celnjo
qb6ize6buo3mgnm8h91 10.24.24.32:2377
```

For T9 to T16 please use:

```
docker swarm join --token
SWMTKN-1-3x45gzklzr94wooe0hvznf2itccrm1wucezs22yk95v42lwf9m-f1wxes
3ajjxcrwnpbrp5yhar4 10.24.24.31:2377
```

Setup Pis to run the containers.

The following will allow us to assign memory to docker containers.
From a sudo account, run the following:

```
sudo nano /boot/cmdline.txt
```

Add the following to the end of the line. Make sure there is only as single line (after a space) and not two lines.

```
cgroup_enable=cpuset cgroup_enable=memory cgroup_memory=1
```

Now we are all set to run the client containers on the Pis.

Setup the run file:
The run_client.sh file will be something like:

```
docker rm -f $(docker ps -aq)
cpu=0
for i in $(seq 0 3); do
        mkdir -p logs/client_$i
        docker run --network <network-name> \
                --env MQTT_IP=<MQTT_IP> --env MQTT_PORT=<MQTT_PORT> --env CLIENT_ID=%i\
                --name="client_$i" \
                --memory="<MEMORY_IN_MB>m" \
                --env DOCKER_RUNNING=True \
                --cpuset-cpus="$cpu" \
                --mount type=bind,source=/home/wschool/logs/client_$i,target=/fedml-ng/logs \
                --mount type=bind,source=/home/wschool/data/<dataset-id>/part_$i,target=/fedml-ng/data \
                --log-driver=json-file \
                -dti flotilla-client:arm
        echo "$cpu"
        cpu=$(($cpu+1)) >> logs/docker_output_client_$i.log
done
sec=0
while true; do
        echo $sec\_log_entry_name_cpu_memory >> logs/util_stats_docker.log
        docker stats --all --format "{{.Name}}:{{.CPUPerc}}:{{.MemUsage}}" --no-trunc --no-stream >> logs/util_stats_docker.log
        sec=$(($sec+1))
        sleep 1
done
```

You have to fill in a few parameters here.

<network-name> is mentioned in the sheet under “Network” and is dependent on the machine Flotilla server is running on.

The <MQTT_IP> and <MQTT_PORT> that your team has to use is mentioned in the sheet provided under the column “MQTT IP:Port”. This is the IP to the container hosting the MQTT broker.

<memory_in_mb> is the amount of memory you want to assign to each Pi. This is dependent on how much memory your Pi has and how many containers you want to run on the Pi, which is configured in the for loop index. Check how much memory your pi has using `free -h`. For this exercise, set to 2048 for Pi4b with 8GB memory, 512 for Pi4b with 2GB memory and 100 for Pi 3b for Pi3b.

In the sheet it documented how many containers you can launch on the Pi assigned to your group under the column “Containers”.

This code pins one container to one CPU core, but you can pin multiple cores to one container. You can change the run file to do that too! Give it a try.

We have facilitated three datasets for this tutorial:

```
wschool@pi-4b-1:~ $ ls data
CIFAR10_NIID3_12  EMNIST_NONIID3  OpenEIA-CA
```

Make sure you run the dataset mentioned in the configuration file you pass to the Flotilla session matches the dataset in the run_client.sh file. This will be the only dataset available to the container!

Once you have run_client.sh file setup, run the following command to get the containers up.

```
bash run_client.sh
```

Setup the Flotilla server

The Flotilla server has been configured and run for your convenience. You have to open **two terminals** to ssh into the Flotilla server container assigned to you. **On both**, do:

```
ssh root@<Server Container IP> -p <Port>
```

Once inside the container, run:

```
cd /fedml-ng && export PATH=/opt/conda/bin:$PATH
```

Now you are all set to run the experiments on Flotilla!

On one terminal run:

```
python flo_server.py
```

You should see MQTT reports from your client containers on your server now.

On the other terminal, start the FL session!

```
cd /fedml-ng && export PATH=/opt/conda/bin:$PATH

python flo_session.py <config> --federated_server_endpoint
localhost:12345
```