EDGE ANALYTICS FOR BROADCASTING NETWORK MONITORING SYSTEMS

Requirements

- **Docker**
- **Docker-compose**
- **Docker Desktop (Not mandatory)**

Installation with github repo

After cloning and opening the GitHub repository, you will find two main folders corresponding to the two applications, WebPlatform and AsyncController.
Both applications contain two docker-compose files:

- docker-compose.dev.yaml: This is the file for developers to use during development. It allows interaction with all the components within the Docker environment via localhost.
- docker-compose.prod.yaml: This is the file for final distribution. It allows interaction only with the components that need to be contacted from the outside.

In order to run both the applications, you should create a network in docker. To do so, run this command in a terminal: *docker network create int-net* .

AsyncController:
To create and run the AsyncController, navigate to the folder where the docker-compose file is located: "/EdgeAnalytics/AsyncController". Here, open a terminal and execute the command

docker-compose -f docker-compose.prod.yaml build

to create the container and the command

docker-compose -f docker-compose.prod.yaml up -d

to run it. You can also use the command

docker-compose -f docker-compose.prod.yaml up --build -d

to perform both operations together.

WebPlatform:
To create and run the WebPlatform, navigate to the folder where the docker-compose file is located: "/EdgeAnalytics/WebPlatform". Here, open a terminal and execute the command

docker-compose -f docker-compose.prod.yaml build

to create the container and the command

docker-compose -f docker-compose.prod.yaml up -d

to run it. You can also use the command

docker-compose -f docker-compose.prod.yaml up --build -d

to perform both operations together. Once the container is running, simply open a browser (not Safari) and enter the URL localhost:3050 to use the application.

Installation with container files .tar

You should have a finalDelivery.zip file. In order to build both the applications, you should unzip it and also unzip both the files webPlatform.zip and asyncController.zip inside the folder.

The installation is the same for both applications (WebPlatform and AsyncController):

1. Open a terminal and navigate into the folder of the application (asyncController-docker or webPlatform-docker)
2. In the terminal, enter the following command: docker load -i webPlatform.tar (or asyncController.tar)
3. Once completed, check that the images have been loaded with the command: *docker images*. The command should return a list similar to this:

```
PS D:\asyncController-Docker> docker images
REPOSITORY                TAG          IMAGE ID       CREATED          SIZE
b-middleware              latest       9b383099b30f   33 hours ago     372MB
b-datamanager             latest       0dbe09f54395   4 days ago       404MB
nginx                     latest       048ac9066b58   4 days ago       188MB
f-frontend                latest       9c47e4fdfa40   5 days ago       1.62GB
b-applicationgateway      latest       8c6677e7d059   5 days ago       372MB
b-authenticationmanager   latest       efaa0dd1da77   6 days ago       373MB
b-assetmanager            latest       518c6e5f4f13   6 days ago       372MB
grafana/grafana-oss       latest       f9095e2f0444   3 weeks ago      443MB
influxdb                  2.6-alpine   f293d68a70a7   14 months ago    158MB
```

4. Create an internal network in Docker named "int-net" with the command: *ù*
   docker network create int-net
   This step is necessary only if both servers are running on the same machine in localhost
5. At this point, you can run the container with the command:
   docker-compose -f docker-compose.prod.yaml up -d
   This command should be executed in the folder corresponding to the container you want to run
6. Once the container has finished initializing, simply open a browser (except Safari), and the application will be available at http://localhost:3050

Final Notes

Currently, both applications only work if run on the same machine, as the URLs they use to communicate with each other are saved as localhost. If you want to run them on different machines, you need to change these URLs, recreate the images, and generate the .tar files for transfer.

To modify the URLs:

1. Clone the GitHub repository.
2. Open the project and navigate to "EdgeAnalytics/AsyncController/Async-Backend/Middleware/src/main/resources" and open application.properties. Here, change gateway.address and gateway.port.
3. Navigate to "EdgeAnalytics/WebPlatform/Backend/ApplicationGateway/src/main/resources" and open application.properties. Here, change asyncController.address and asyncController.port.

At this point, you need to recreate the images. To do this, navigate to the project folder in a terminal and:

- Open the /AsyncController folder and execute the command
  docker-compose -f docker-compose.prod.yaml --build
- Open the /WebPlatform folder and execute the command
  docker-compose -f docker-compose.prod.yaml --build

To run the two applications, execute the command *docker-compose -f docker-compose.prod.yaml up* (with the --build option if you also want to create the images).

Finally, to create the .tar files for transferring the containers:

- Check that the images have been correctly created and saved with the command *docker images*
- Open a terminal and execute the command
  docker save -o asyncController.tar b-middleware:latest
- Then execute the command
  docker save -o webPlatform.tar nginx:latest b-applicationgateway:latest grafana/grafana-oss b-assetmanager:latest b-authenticationmanager:latest influxdb:2.6-alpine b-datamanager:latest f-frontend:latest

These commands will create two .tar files corresponding to the two containers. Simply replace these files in the asyncController-Docker and webPlatform-Docker folders.

Try the application with Python scripts

In the folder, there are two Python scripts to "imitate" the presence of real devices.
To use both the scripts you should install the package asyncCommunication with the command
pip install asyncCommunication
In the folder is also present another folder “assets” that contains some files to try the application.
The scripts have these functions:

- registerDevice: This script is used only to register the device. It needs to be run only once for each device (with a different name) and it just declares its existence to the main application.
- deviceLoop: This script simulates a potential loop of a device. It is important to change the device name within the script to simulate a different device. The script asks the application if there is a new request to execute, and if so, it executes it.