EDGE ANALYTICS FOR BROADCASTING NETWORK MONITORING SYSTEMS

Requisiti

- **Docker**
- **Docker-compose**
- **Docker Desktop (Opzionale)**

Installazione con la repo su github

Dopo aver clonato e aperto la repo di github, troverete due cartelle principali che corrispondono alle due applicazioni WebPlatform e AsyncController.
Entrambe le applicazioni hanno al loro interno due file docker-compose:

- docker-compose.dev.yaml: è il file per i developer da usare durante lo sviluppo, permette di poter interagire con tutti i componenti all'interno del docker tramite localhost.
- docker-compose.prod.yaml: è il file per la distribuzione finale, permette di poter interagire solo con i componenti che devono essere contattati dall'esterno.

Prima di eseguire le due applicazioni, dovete creare una rete di docker. Per farlo, eseguite questo comando in un terminale: *docker network create int-net* .

AsyncController:
Per creare e far eseguire l' asyncController, si deve andare nella cartella in cui si trova il docker-compose file "/EdgeAnalytics/AsyncController" . Qui, si deve aprire un terminale ed eseguire il comando

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

per creare il container e il comando

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

per eseguirlo. Si può pure fare il comando

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

per fare entrambe le operazioni insieme.

WebPlatform:
Per creare e far eseguire la webPlatform, si deve andare nella cartella in cui si trova il docker-compose file "/EdgeAnalytics/WebPlatform" . Qui, si deve aprire un terminale ed eseguire il comando

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

per creare il container e il comando

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

per eseguirlo. Si può pure fare il comando

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

per fare entrambe le operazioni insieme. Quando il container è stato eseguito, basta aprire il browser (non Safari) e inserire l'url localhost:3050 per utilizzare l'applicazione.

Installazione con i container .tar

Dato il file finalDelivery.zip , per creare i container dovete decomprimere il file e poi decomprimere anche i file webPlatform.zip e asyncController.zip all'interno della cartella finalDelivery.

L'installazione è la uguale per entrambe le applicazioni (WebPlatform e AsyncController):

1. Apri un terminale e naviga all'interno della cartella dell'applicazione (asyncController-docker o webPlatform-docker)
2. Nel terminale scrivere il seguente comando: *docker load -i webPlatform.tar* (oppure asyncController.tar)
3. Al termine, controllare che le immagini siano state caricate con il comando: *docker images* . Il comando dovrebbe tornare una lista simile a questa:

```
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. Creare una rete interna in docker col nome "int-net" con il comando: *docker network create int-net* . Questo passaggio serve solo se entrambi i server vengono fatti eseguire nella stessa macchina in localhost.
5. A questo punto si può eseguire il container con il comando: *docker-compose -f docker-compose.prod.yaml up -d* . Questo comando va eseguito nella cartella corrispondente al container che si vuole eseguire.
6. Quando il container ha finito l'inizializzazione, basta aprire un browser (tranne Safari), e l'applicazione sarà utilizzabile in http://localhost:3050

Note finali

Attualmente entrambe le applicazioni funzionano solo se eseguite nella stessa macchina, dato che gli url con i quali comunicano fra di loro sono salvati come localhost.
Nel caso si volessero eseguire in macchine diverse, si dovrebbero cambiare questi url e ricreare le immagini e i file .tar per poterli trasferire.
Per modificare gli url:

1. Clonare la repo di github
2. Aprire il progetto e navigare in "EdgeAnalytics/AsyncController/Async-Backend/Middleware/src/main/resources" e aprire application.properties, qui modificare gateway.address e gateway.port
3. Navigare in "EdgeAnalytics/WebPlatform/Backend/ApplicationGateway/src/main/resources" e aprire application.properties, qui modificare asyncController.address e asyncController.port

A questo punto, si devono ricreare le immagini. Per farlo basta posizionarsi nella cartella del progetto con un terminale e:

- Aprire la cartella /AsyncController e eseguire il comando *docker-compose -f docker-compose.prod.yaml –build*
- Aprire la cartella /WebPlatform e eseguire il comando *docker-compose -f docker-compose.prod.yaml –build*

Se si vuole far eseguire le due applicazioni, basta runnare il comando *docker-compose -f docker-compose.prod.yaml up* (con l'opzione –build se si vuole anche creare le immagini).

Infine, per creare i file .tar per trasferire i container si deve:

1. Controllare che le immagini siano state correttamente create e salvate con il comando *docker images*
2. Aprire un terminale ed eseguire il comando *docker save -o asyncController.tar b-middleware:latest*
3. Eseguire poi il comando *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*

Questi comandi creeranno due file .tar corrispondenti ai due container, basta quindi sostituire questi file nelle cartelle asyncController-Docker e webPlatform-Docker.

Provare l'applicazione con gli script di Python

Nella cartella sono presenti due script in python per "imitare" la presenza di veri Device.
Per usare entrambi gli script devi installare la libreria asyncCommunication con il comando
pip install asyncCommunication
Nella cartella è anche presente un'altra cartella "assets" con all'interno alcuni file già pronti con cui provare l'applicazione.
Gli script, in particolare, hanno queste funzioni:

- registerDevice: è uno script che serve solo a registrare il device. Deve essere eseguito una sola volta per ogni device (differente nome) e non fa altro che dichiarare la propria esistenza all'applicazione principale.
- deviceLoop: è lo script che simula un eventuale loop di un device. E' importante modificare il nome del device all'interno dello script per poter cambiare il device da simulare. Lo script richiede all'applicazione se c'è un nuova richiesta da eseguire e in caso affermativo la esegue.