SISTEME DE PROGRAME PENTRU REȚELE DE CALCULATOARE
TEMA 3
PLATFORMĂ INTERNET OF THINGS FOLOSIND MICROSERVICII

Cătălin-Alexandru Rîpanu, 341C3
Facultatea de Automatică și Calculatoare
Universitatea Națională de Știință și Tehnologie Politehnica București
catalin.ripanu@stud.acs.upb.ro

9 Ianuarie 2024

ABSTRACT

Această temă are drept obiectiv implementarea unei arhitecturi **IoT de monitorizare** ce folosește tehnologia **Docker Swarm** pentru construirea unei **stive de servicii** ce servesc la manipularea logicii de **colectare**, **stocare** și **vizualizare** a **datelor numerice** provenite de la un număr semnificativ de **dispozitive conectate** prin rețea.

1 Implementare

Stiva soluției conține 4 servicii importante, și anume: **broker-ul de MQTT** ce folosește ultima versiune a imaginii aferente, **baza de date de tipul InfluxDB** cu versiunea 1.8 ce **nu necesită vreo logică de autentificare**, **adapter-ul de MQTT** implementat în python, respectiv descris într-un fișier Dockerfile în directorul său și **instanța de tip Grafana** ce permite vizualizarea datelor trimise de adapter prin intermediul unor dashboard-uri scrise în **JSON** și adăugate în directorul **grafana_db/grafana_provisioning/grafana_dashboards**.

Soluția se poate **executa** folosind doar **o comandă**, și anume **./run.sh** (acesta este un script care realizează **întregul setup**). De asemenea, **ștergerea tuturor imaginilor / volumelor locale** create de run.sh se poate face folosind comanda **./clean.sh** (acesta este un script care elimină toate lucrurile **adăugate de run.sh**).

1.1 Broker-ul de MQTT

Acest **broker** așteaptă eventualele conexiuni din partea **dispozitivelor IoT** pe **port-ul 1883**. Mai mult, el folosește fișierul de configurație **mosquitto_mqtt_vol/mosquitto.conf** astfel încât să permită (sau nu) conexiuni din partea unor clienți **neautorizați** (care nu folosesc vreun cont **valid**, de asemenea, în script-ul **run.sh** există un **prompt** pentru utilizatorul soluției care modifică fișierul de configurație al broker-ului în funcție de ceea ce se tastează).

Dacă se alege varianta cu 'da', atunci broker-ul citește **alte 2 fișiere** ce se ocupă de configurarea următoarelor 2 conturi: **username = writer** & **password = writer_sprc**, respectiv **username = reader** & **password = reader_sprc**. Adaptorul se conectează la broker folosind credențialele **contului cu username = reader** indiferent de opțiunea utilizatorului.

De asemenea, acesta se află în rețeaua **broker_adaptor_network** împreună cu adapter-ul astfel încât să comunice doar cu acesta, și, nu în ultimul rând, s-a luat decizia ca broker-ul să își **salveze toate fișierele** în directorul menționat anterior astfel încât să nu se piardă log-urile ce ar putea ajuta într-o eventuală **situație de cădere**.

1.2 Baza de Date InfluxDB

Acest **serviciu** a fost configurat folosind script-ul bash din directorul **database/init_influx_db.sh** ce folosește comenzi **CLI** astfel încât să creeze **baza de date** numită **IoT_Devices** cu retenția datelor nelimitată. La fel ca mai sus, se păstrează toate informațiile trimise în timpul execuției în directorul **database/influx_db** prin copierea tuturor fișierelor din **/var/lib/influxdb** (cale ce se află în **container-ul Docker**). Evident, pentru a limita comunicarea cu celelalte containere, acest serviciu se află în rețelele **adaptor_influx_network** și **influx_grafana_network** astfel încât să poată comunica doar cu **adapter-ul** și **instanța de Grafana**.

1.3 Adapter-ul de MQTT

Acest serviciu a fost implementat în **Python** folosind pachetul **paho.mqtt.client** și se află în directorul **adapter**. De asemenea, imaginea aferentă **adapter-ului** se poate construi folosind fișierul **Dockerfile** ce se află în același director. Pentru conceperea **acestei entități** s-a preferat o implementare **OOP** codată în sursa **adapter.py** ce folosește un **client MQTT** pentru a se conecta la **broker** și un **client InfluxDB** pentru a comunica cu **baza de date** (adapter-ul este cel care adaugă informațiile procesate în această bază, nu clienții ce transmit date către broker). Pentru **eliminarea mesajelor** cu topicul **formatat greșit** s-a folosit o **expresie regulată** ce ajută la **respingerea lor** (folosind, evident, funcția **match** din pachetul **re**).

1.4 Instanța de tip Grafana

Acest serviciu este accesibil **pe port-ul 80** și reprezintă o **interfață grafică** ce se poate activa cu ajutorul unui **browser web** și care conține **2 dashboard-uri predefinite** ce ilustrează datele provenite din **baza de date a stack-ului**. Dashboard-urile au fost scrise în **JSON** și se află în directorul **grafana_db/grafana_provisioning/grafana_dashboards** (în **grafana_db/grafana_provisioning** se află și **câteva fișiere .yml** folosite de Grafana pentru încărcarea acestor **dashboard-uri** și **setarea sursei de date**).

Folosind **expresii regulate** în sursele **JSON**, cele **2 dashboard-uri** sunt **capabile** să respecte **restricțiile din enunț** (cum ar fi adăugarea **noilor dispozitive care se conectează** și **noilor metrici** care apar).