Configurarea şi Lansarea Platformei de Laborator pe VPS (Ubuntu 24.04) cu Docker, Caddy şi Mosquitto

Context

49.13.76.182 Pass 12.08 Tprwd4PuL7gA

Documentul descrie paşii pentru configurarea unei platforme de laborator compusă dintr-un server web (**Caddy** cu TLS automat), un serviciu frontend, un serviciu backend demonstrativ (*echo-server*) şi un broker MQTT (**Mosquitto**), toate orchestrate prin **Docker Compose**. Infrastructura este implementată pe un VPS cu Ubuntu 24.04.

Cerinţe preliminare

- Înregistrări DNS de tip `A` către IP-ul VPS pentru subdomeniile: `app.domeniu.tld`, `api.domeniu.tld`, `mqtt.domeniu.tld` (Document VPS).
- Porturi deschise în firewall: 22, 80, 443, 1883, 9001 (Secțiunea 1).
- Docker şi Docker Compose instalate şi funcţionale (Document Instalare Docker si Docker Compose).

1 Configurarea firewall-ului (UFW) și deschiderea porturilor necesare

Pentru a proteja accesul la platformă, se utilizează **UFW** (*Uncomplicated Firewall*) pentru controlul traficului de reţea. Se vor permite doar porturile necesare pentru funcţionarea platformei:

- 22/tcp — SSH, administrare VPS
- 80/tcp — HTTP
- 443/tcp — HTTPS
- 1883/tcp — MQTT (protocol TCP)
- 9001/tcp — MQTT peste WebSockets

Configurarea se realizează astfel:

```
# Instalarea UFW (daca nu este deja instalat)
sudo apt install ufw -y

# Politici implicite: blocare intrari, permitere iesiri
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Permiterea porturilor necesare
sudo ufw allow 22/tcp    # SSH
sudo ufw allow 80/tcp    # HTTP
sudo ufw allow 443/tcp   # HTTPS
sudo ufw allow 1883/tcp  # MQTT TCP
sudo ufw allow 9001/tcp  # MQTT WebSockets

# Activarea firewall-ului
```

```
sudo ufw enable

# Verificarea regulilor active
sudo ufw status verbose
```

Notă: Portul 22/tcp trebuie permis înainte de activarea UFW pentru a evita pierderea conexiunii SSH.

2 Crearea structurii directoarelor

Se defineşte o variabilă de mediu pentru domeniu[1]. Se creează directoarele necesare pentru serviciile platformei.

```
BASE=exemplu.ro
aiot-xplorer.eu
sudo mkdir -p /opt/app/{caddy,mosquitto/{config,data,log}} /opt/app/{frontend,backend}
cd /opt/app
```

3 Fişierul de configurare `.env`

Fişierul `.env` centralizează valorile parametrilor utilizaţi de serviciile definite în Docker Compose şi Caddy.

```
cat > .env << EOF
DOMAIN=app.$BASE
API_DOMAIN=api.$BASE
MQTT_DOMAIN=mqtt.$BASE
ACME_EMAIL=admin@$BASE
MQTT_TCP_PORT=1883
MQTT_WS_PORT=9001
EOF
```

Parametrii includ:

- `DOMAIN`, `API_DOMAIN`, `MQTT_DOMAIN` — subdomeniile pentru serviciile frontend, backend şi MQTT.
- `ACME_EMAIL` — adresa de email utilizată pentru certificatele TLS (Let's Encrypt).
- `MQTT_TCP_PORT`, `MQTT_WS_PORT` — porturile externe pentru brokerul MQTT.

4 Verificarea porturilor

Pentru a verifica pe ce porturi ascultă fiecare serviciu:

```
docker exec -it <app> sh -lc 'apk add --no-cache curl >/dev/null 2>&1 || true; curl -I http://backend:80/'
```

5 Verificarea numelor serviciilor

```
docker compose config --services
```

[1] Utilizarea variabilei `BASE` permite reutilizarea rapidă a valorii domeniului în mai multe comenzi. Alternativ, domeniile pot fi introduse direct în fişierul `.env`.

6 Verificarea comunicației dintre frontend și backend

```
    docker exec -it app_stack-frontend-1 sh -lc 'apk add --no-cache curl bind-tools >/dev
    /null 2>&1 || true; \
  echo "resolv.conf:"; cat /etc/resolv.conf; \
  echo "--- nslookup backend:"; nslookup backend 127.0.0.11 || true; \
  echo "--- curl:"; curl -I http://backend:80/ || true'
```

```
    # 1) API prin Caddy (HTTPS)
docker exec -it app_stack-frontend-1 sh -lc '
apk add --no-cache curl bind-tools >/dev/null 2>&1 || true;
CADDY_IP=$(getent hosts caddy | awk "{print \$1}");
curl -svL --resolve api.aiot-xplorer.eu:443:$CADDY_IP https://api.aiot-xplorer.eu/ -o /
    dev/null
'

# 2) Frontend prin Caddy (HTTPS)
docker exec -it app_stack-frontend-1 sh -lc '
CADDY_IP=$(getent hosts caddy | awk "{print \$1}");
curl -svL --resolve app.aiot-xplorer.eu:443:$CADDY_IP https://app.aiot-xplorer.eu/ -o /
    dev/null
'
```

7 Caddyfile: rol şi utilizare

7.1 Definiţie

Caddyfile este fişierul de configurare al serverului web *Caddy*. Acesta specifică domeniile deservite, modul de rutare al traficului către alte servicii, configuraţia TLS şi opţiuni suplimentare precum logarea sau compresia conţinutului.

7.2 Roluri principale

- **Definirea domeniilor:** se indică pe ce domenii să asculte Caddy.
- **Rutare prin reverse proxy:** cererile HTTP/S sunt redirecţionate către servicii backend (de exemplu, containere Docker interne).
- **Gestionarea certificatelor TLS:** Caddy obţine şi reînnoieşte automat certificate HTTPS de la *Let's Encrypt*.
- **Opţiuni suplimentare:** se pot adăuga reguli de compresie, redirecţionări, autentificare şi logare.

7.3 Exemplu de configurare

```
# Frontend (SPA) pe portul 3000
app.domeniu.tld {
    reverse_proxy frontend:3000
}

# API backend pe portul 8080
api.domeniu.tld {
    reverse_proxy api:8080
}

# MQTT peste WebSockets securizat
```

```
mqtt.domeniu.tld {
    reverse_proxy mqtt:9001
}
```

7.4 Comenzi utile

- **Verificarea sintaxei Caddyfile:**

  ```
  docker run --rm -v $(pwd)/Caddyfile:/etc/caddy/Caddyfile caddy validate
  ```

- **Repornirea Caddy după modificări:**

  ```
  docker compose restart caddy
  ```

- **Vizualizarea logurilor Caddy:**

  ```
  docker logs -f caddy
  ```

7.5 Observaţie

Blocurile de configurare din Caddyfile trebuie scrise cu acolade pe linii separate:

```
app.domeniu.tld {
    reverse_proxy frontend:3000
}
```

şi nu pe aceeaşi linie cu instrucţiunea `reverse_proxy`.

7.6 Fişierul `Caddyfile`

Caddy este configurat pentru a acţiona ca reverse proxy şi pentru a obţine automat certificate TLS pentru fiecare subdomeniu.

```
cat > caddy/Caddyfile << 'EOF'
{
  email {$ACME_EMAIL}
}
{$DOMAIN} {
  encode zstd gzip
  header { Strict-Transport-Security "max-age=31536000;␣includeSubDomains;␣preload" }
  reverse_proxy frontend:3000
}
{$API_DOMAIN} {
  encode zstd gzip
  header { Strict-Transport-Security "max-age=31536000;
␣␣includeSubDomains;␣preload" }
  reverse_proxy backend:80
}
{$MQTT_DOMAIN} {
  header { Strict-Transport-Security "max-age=31536000;␣includeSubDomains;␣preload" }
  reverse_proxy http://mosquitto:9001
}
EOF
```

8 Configurarea brokerului MQTT (Mosquitto)

Brokerul este configurat să accepte conexiuni pe protocolul MQTT clasic (TCP) şi pe MQTT peste WebSockets.

```
cat > mosquitto/config/mosquitto.conf << 'EOF'
listener 1883 0.0.0.0
protocol mqtt
listener 9001 0.0.0.0
protocol websockets
persistence true
persistence_location /mosquitto/data/
log_dest file /mosquitto/log/mosquitto.log
allow_anonymous true
EOF
```

Notă de securitate: setarea `allow_anonymous true` este destinată exclusiv mediilor de testare. În medii de producţie se recomandă:

1. Crearea unui fişier de parole cu `mosquitto_passwd`.
2. Setarea `allow_anonymous false` şi `password_file` în configuraţie.
3. Repornirea serviciului Mosquitto.

Securizarea brokerului MQTT în medii de producţie

Pentru a restricţiona accesul la broker şi a permite conectarea doar utilizatorilor autorizaţi, se recomandă configurarea autentificării cu nume de utilizator şi parolă.

1. Crearea unui fişier de parole:

```
docker exec -it mosquitto mosquitto_passwd -c /mosquitto/config/passwd utilizator1
```

2. Modificarea fişierului `mosquitto.conf` pentru a dezactiva accesul anonim şi a utiliza fişierul de parole:

```
allow_anonymous false
password_file /mosquitto/config/passwd
```

3. Repornirea serviciului Mosquitto pentru aplicarea modificărilor:

```
docker compose restart mosquitto
```

După aplicarea acestei configuraţii, toate conexiunile la broker vor necesita autentificare cu nume de utilizator şi parolă.

Restricţionarea accesului pe echipe (ACL – Access Control List)

Pentru a separa accesul la topicuri între diferite echipe sau grupuri, se utilizează un fişier ACL care defineşte permisiunile de publicare (`write`) şi abonare (`read`).

1. Crearea fişierului ACL:

```
cat > /opt/app/mosquitto/config/acl << 'EOF'
# Echipa 1 - poate publica si citi pe topicurile "team1/#"
user echipa1
topic readwrite team1/#

# Echipa 2 - poate publica si citi pe topicurile "team2/#"
user echipa2
topic readwrite team2/#

# Acces read-only pentru topicurile publice
pattern read public/#
EOF
```

2. Modificarea fişierului `mosquitto.conf` pentru a utiliza ACL-ul:

```
allow_anonymous false
password_file /mosquitto/config/passwd
acl_file /mosquitto/config/acl
```

3. Crearea utilizatorilor şi setarea parolelor:

```
docker exec -it mosquitto mosquitto_passwd -c /mosquitto/config/passwd echipa1
docker exec -it mosquitto mosquitto_passwd /mosquitto/config/passwd echipa2
```

4. Repornirea brokerului Mosquitto:

```
docker compose restart mosquitto
```

Notă: În exemplul de mai sus, fiecare echipă are acces complet doar la propriile topicuri (`team1/#` sau `team2/#`), iar topicurile `public/#` pot fi citite de toţi utilizatorii autentificaţi.

9 `docker-compose.yml`

Fişierul Docker Compose defineşte serviciile platformei: Caddy, frontend-ul demonstrativ (Nginx static), backend-ul demonstrativ (echo-server) şi brokerul MQTT.

```
cat > docker-compose.yml << 'EOF'
version: "3.9"
name: app_stack
services:
  caddy:
    image: caddy:2.8-alpine
    restart: unless-stopped
    ports: ["80:80","443:443"]
    environment:
      - DOMAIN=${DOMAIN}
      - API_DOMAIN=${API_DOMAIN}
      - MQTT_DOMAIN=${MQTT_DOMAIN}
      - ACME_EMAIL=${ACME_EMAIL}
    volumes:
      - ./caddy/Caddyfile:/etc/caddy/Caddyfile:ro
      - caddy_data:/data
      - caddy_config:/config
    depends_on: [frontend, backend, mosquitto]
    networks: [app_net]

  frontend:
    image: nginx:alpine
    restart: unless-stopped
    expose: ["3000"]
    command: ["/bin/sh","-c","mkdir -p /usr/share/nginx/html && echo '<h1>Frontend OK</h1
    >' > /usr/share/nginx/html/index.html && nginx -g 'daemon off;'"]
    networks: [app_net]

  backend:
    image: ealen/echo-server:latest
    restart: unless-stopped
    expose: ["80"]
    networks: [app_net]

  mosquitto:
    image: eclipse-mosquitto:2
    restart: unless-stopped
    ports:
      - "${MQTT_TCP_PORT}:1883"
      - "${MQTT_WS_PORT}:9001"
    volumes:
      - ./mosquitto/config/mosquitto.conf:/mosquitto/config/mosquitto.conf:ro
      - ./mosquitto/data:/mosquitto/data
      - ./mosquitto/log:/mosquitto/log
    networks: [app_net]

networks: { app_net: { driver: bridge } }
volumes: { caddy_data: {}, caddy_config: {} }
EOF
\end
```

Pentru verificarea volumelor:

```
docker inspect <nume_container> --format '{{json␣.Mounts}}' | jq
```

Acestea trebuie să fie montate astfel: config, data și (opțional) log montate persistent (bind sau named volume):

```
/mosquitto/config -> catre /opt/app/mosquitto/config
/mosquitto/data -> catre /opt/app/mosquitto/data
```

```
/mosquitto/log -> catre /opt/app/mosquitto/log
```

Dacă jq nu este instalat, se poate instala folosind apt install jq.

Se recomandă efectuarea unui backup cu:

```
sudo tar -C /opt/app/mosquitto -czf ~/mosquitto-backup-$(date +%F).tgz config data
```

Mai departe, se dau permisiunile corecte pe host:

```
sudo chown -R 1883:1883 /opt/app/mosquitto/{config,data,log}
sudo chmod -R u+rwX,go-rwx /opt/app/mosquitto/{config,data,log}
```

Configurația se poate testa, mai departe, cu un container temporar, pentru a nu afecteze instanța curentă. Acest container temporar va încerca să pornească cu aceleași fișiere (pe alt port), doar ca test.

```
docker run --rm -it \
  -p 1884:1883 \
  -v /opt/app/mosquitto/config:/mosquitto/config \
  -v /opt/app/mosquitto/data:/mosquitto/data \
  -v /opt/app/mosquitto/log:/mosquitto/log \
  eclipse-mosquitto:2 \
  mosquitto -c /mosquitto/config/mosquitto.conf -v -p 1883
```

Mai departe, vom avea nevoie de două terminale. În acest scop, instalăm tmux cu apt install tmux.

Pentru a avea două terminale, unul in care sa vedem logurile mosquitto și altul în care să dăm comenzi în continuare, folosim comanda:

```
tmux split-window -h \; send-keys -t 0 'docker compose logs -f mosquitto' C-m \;
select-pane -R
```

Pentru a naviga in terminalul unde vom introduce comenzile, folosim Ctrl+b și tasta pentru săgeata dreapta.

Vom instala aplicația pentru clienți MQTT mosquitto-clients si mosquitto

```
sudo apt install mosquitto-clients
sudo apt install mosquitto
```

10 Verificarea autentificării şi a regulilor ACL în Mosquitto

După configurarea fişierului `passwd` şi a fişierului `acl`, este necesară verificarea funcţionalităţii acestora prin teste de conectare şi publicare.

10.1 Testarea utilizatorului `admin`

Utilizatorul `admin` trebuie să aibă acces complet la toate topic-urile:

```
mosquitto_sub -h 127.0.0.1 -p 1883 -u admin -P 'ADMIN@IoT' -t '#' -v
mosquitto_pub -h 127.0.0.1 -p 1883 -u admin -P 'ADMIN@IoT' \
  -t 'test/topic' -m 'mesaj test'
```

Rezultatul aşteptat: mesajul publicat trebuie să apară imediat în sesiunea de `subscribe`.

10.2 Testarea utilizatorului `echipa1`

Utilizatorul `echipa1` are acces doar la topic-urile `team1/#`:

```
# Abonare pe topic permis
mosquitto_sub -h 127.0.0.1 -p 1883 -u echipa1 -P 'PAROLA1' \
  -t 'team1/#' -v

# Publicare pe topic permis
mosquitto_pub -h 127.0.0.1 -p 1883 -u echipa1 -P 'PAROLA1' \
  -t 'team1/test' -m 'ok'

# Publicare pe topic interzis
mosquitto_pub -h 127.0.0.1 -p 1883 -u echipa1 -P 'PAROLA1' \
  -t 'team2/test' -m 'nu'
```

Rezultatul aşteptat: publicarea pe `team1/#` funcţionează, publicarea pe `team2/#` este blocată de ACL (nu ajunge la abonaţi, dar se înregistrează în logurile brokerului ca `DENIED`).

10.3 Testarea utilizatorului `echipa2`

Utilizatorul `echipa2` are acces doar la topic-urile `team2/#`:

```
# Publicare pe topic permis
mosquitto_pub -h 127.0.0.1 -p 1883 -u echipa2 -P 'PAROLA2' \
  -t 'team2/test' -m 'ok'

# Publicare pe topic interzis
mosquitto_pub -h 127.0.0.1 -p 1883 -u echipa2 -P 'PAROLA2' \
  -t 'team1/test' -m 'nu'
```

Rezultatul aşteptat: publicarea pe `team2/#` funcţionează, publicarea pe `team1/#` este blocată de ACL.

10.4 Observaţie

Protocolul MQTT nu transmite clientului un mesaj de eroare atunci când un `PUBLISH` este refuzat de ACL (în special pentru QoS 0). Singura confirmare a refuzului este vizibilă în logurile brokerului, unde apare o intrare de tip:

`Denied PUBLISH from <utilizator> ...`

Pentru diagnosticare, este recomandată monitorizarea logurilor în paralel cu testele.

11 Pornirea serviciilor şi verificarea stării

Pentru a porni serviciile definite în `docker-compose.yml` şi a ne asigura că rulează corect, se utilizează următoarele comenzi:

```
docker compose pull
docker compose up -d
docker compose ps
```

- `docker compose pull` – descarcă cele mai recente versiuni ale imaginilor Docker specificate în fişierul `docker-compose.yml`.
- `docker compose up -d` – porneşte containerele în mod *detached* (în fundal), conform configuraţiei. Flagul up creează și pornește containerele conform fișierului docker-compose.yml, iar -d = detached mode, rulează containerele fără să le atașeze la terminal. Astfel, promptul revine imediat și se poate continua scrierea de comenzi.

- `docker compose ps` – afişează lista containerelor gestionate de Docker Compose, cu starea curentă, porturile mapate şi eventualele reporniri.

Este recomandată verificarea ca toate containerele să fie în starea `running` înainte de a continua testele de autentificare şi ACL.

12 Verificări funcţionale

- Jurnalul Caddy trebuie să indice emiterea certificatelor TLS pentru subdomenii: `docker logs -f caddy`.
- Accesarea `https://app.domeniu.tld` trebuie să afişeze pagina *Frontend OK*.
- Accesarea `https://api.domeniu.tld` trebuie să returneze răspuns JSON de la *echo-server*.
- Conexiune MQTT TCP: host `mqtt.domeniu.tld`, port `1883`.
- Conexiune MQTT peste WebSockets (WSS): `wss://mqtt.domeniu.tld` (port 443 extern către WS 9001 intern).

13 Recomandări pentru medii de producţie

- Securizarea brokerului MQTT prin utilizatori/parole şi ACL-uri.
- Adăugarea de *healthchecks* pentru monitorizarea serviciilor.
- Implementarea rotaţiei logurilor.
- Actualizarea periodică a imaginilor şi serviciilor.