Опис завдання:

Створити docker-compose.yaml для формування макета системи централізованого логування з попереднього завдання. За бажанням контейнер чи контейнери, з яких збиратимуться логи, можна змінити на щось практичніше.

Результат завдання:

Завантажити на сервер протестований робочий docker-compose.yaml файл з коментарями

План

1. Готуємо все необхідне для створення іміджу Fluentd
    1.1 Створюємо теку fluentd-loki
    1.2 Створюємо Dockerfile для Fluentd
    1.3 Створюємо файл конфігурації fluentd.conf
2. Готуємо все необхідне для Grafana
    2.1 Створюємо теку grafana
    2.2 Створюємо файл datasources.yaml для Grafana
3. Створюємо файл docker-compose.yaml для Docker Compose
4. Створюємо файл .env
5. Перевірка, старт створених сервісів

Попередньо маємо налаштовану VM з встановленим Docker, та прописані у файлі hosts ip адреси нашої VM. Також для цього завдання створюємо окрему теку lesson7 і всі подальші дії відбуваються саме в цій теці

1.Готуємо все необхідне для створення іміджу Fluentd

1.1 Створюємо теку fluentd-loki

Команда:
    mkdir fluentd-loki
Відгук:
...

Входимо до створеної теки теки

Команда:
    cd fluentd-loki
Відгук:

1.2 Створюємо Dockerfile для Fluentd

Команда:
    nano Dockerfile
Відгук:

Заповнюємо файл наступним вмістом:

```
FROM fluent/fluentd:v1.16-debian
USER root
RUN gem install fluent-plugin-loki
USER fluent
```

Зберігаємо і закриваємо файл.
В ньому ми вказали яку саме версію `fluentd` ми хочемо використати, користувача для операційної системи на базі якої зібрано імідж, дію яку треба виконати в процесі створення іміджу і користувача для fluent

1.3 Створюємо файл конфігурації fluentd.conf

Команда:

```
nano fluentd.conf
```

Відгук:

Заповнюємо файл наступним вмістом:

```
<source>
  @type forward
</source>

<match **>
  @type loki
  endpoint_url "http://loki:3100"
  labels {"job":"docker-logs"}
</match>
```

Зберігаємо і закриваємо файл файл.
В ньому ми всказали кому будуть пересилатися логи, на яку адресу та порт, і позначку по який ми можемо фільтрувати наші логи у Grafana

2. Готуємо все необхідне для Grafana

2.1 Створюємо теку grafana

Команда:

```
mkdir grafana
```

Відгук:
...

Входимо до створеної теки теки

Команда:

```
 cd grafana
```

Відгук:

2.2 Створюємо файл datasources.yaml для Grafana

Команда:

```
nano  datasources.yaml
```

Відгук:

Заповнюємо файл наступним вмістом:

apiVersion: 1

datasources:
  - name: Loki
  - type: loki
  - access: proxy
  - url: http://loki:3100
  - isDefault: true

Зберігаємо і закриваємо файл.

В ньому ми вказали версію api, налаштування джерела даних - назву джерела даних, його тип, режим доступу, адресу звідки отримувати данні, підтверджуємо це джерело даних по замовченню для панелі Grafana

3. Створюємо файл docker-compose.yaml для Docker Compose

В кореневій теці цього ДЗ, створюємо файл docker-compose.yaml

Команда:
nano docker-compose.yaml
Відгук:

Заповнюємо файл наступним вмістом:

#Оскільки деякі директиви та секції повторюються, то шоб не робити довге простирадло їх опис буде або стислим або відсутнім

#Директива version визначає версію синтаксису Docker Compose, яка використовується у файлі docker-compose. Але чогось ця директива викликає помилку яку я поставив нижче
version: '3.8'
the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion

Ця секція описує контейнери (сервіси), їх налаштування і залежності. У нас це будуть - loki, fluentd-loki, log_container_fluentd та grafana
services:
Назва сервісу
  loki:
#Директива image вказує образ, який буде використовуватися для створення контейнера
    image: grafana/loki:latest
#Директива ports визначає порти, які мають бути відкриті і перенаправлені з хост-машини на контейнер
    ports:
- "3100:3100"

```
#Директива networks підключає сервіс до вказаної мережі або мереж
    networks:
      - bridge_network
#Назва сервісу
  fluentd-loki:
#Директива build використовується для вказівки шляху до Dockerfile, який має бути використаний для збірки образу. В нашому випадку це тека fluentd-loki у кореневій теці ДЗ
    build:
#Вказуємо теку
      context: ./fluentd-loki
#Вказуємо назву файлу
      dockerfile: Dockerfile
#Вказуємо порти на яких працюватиме сервіс
    ports:
      - "24224:24224"
- "24224:24224/udp"
#Директива volumes монтує томи в контейнери, що дозволяє зберігати дані між перезапусками контейнерів
    volumes:
      - ./fluentd-loki/fluentd.conf:/fluentd/etc/fluent.conf:ro
#Підключаємо до вказаної мережі
    networks:
      - bridge_network
#Директива depends_on вказує, що цей сервіс залежить від інших сервісів і має бути запущений після них. В нашому випадку цей сервіс чекає коли буде запущено після сервісу loki
    depends_on:
      - loki

  log_container_fluentd:
    image: busybox
#Директива command дозволяє перевизначити команду, яка буде виконана при запуску контейнера
    command: "sh -c 'while true; do echo \"Fluentd test log: $(date)\"; sleep 2; done'"
    networks:
        - bridge_network
#Директива logging яка визначає який драйвер логування треба використовувати
    logging:
#Драйвер логування, тут ми використовуємо fluentd
      driver: fluentd
#Встановлюємо параметри драйвера за допомогою параметрів які виглядають як пари ключ-значення
      options:
        fluentd-address: localhost:24224
        tag: log_container_fluentd.logs
#Запускаємо після fluentd-loki
    depends_on:
        - fluentd-loki
```

```
#Назва сервісу
  grafana:
#Образ шо використовуємо
    image: grafana/grafana
#Монтуємо том в контейнер
    volumes:
      - ./grafana/provisioning:/etc/grafana/provisioning:ro
#Використовуємо цю мережу
    networks:
       - bridge_network
#Використовуємо ці порти
    ports:
       - "3000:3000"
#Директива environment задає змінні середовища для контейнера
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=${GF_SECURITY_ADMIN_PASSWORD}
      - GF_DASHBOARD_DEFAULT_HOME_DASHBOARD_PATH=$
{GF_DASHBOARD_DEFAULT_HOME_DASHBOARD_PATH}
      - GF_SERVER_ROOT_URL=${GF_SERVER_ROOT_URL}
#Запускаємо після loki
    depends_on:
       - loki

#Розділ networks дозволяє визначити користувацькі мережі, в яких
працюватимуть контейнери
networks:
  bridge_network:
    driver: bridge
```

Зберігаємо і закриваємо цей файл.

4. Створюємо файл .env

В кореневій теці цього ДЗ, поруч з файлом docker-compose.yaml створюємо файл .env В цьому файлі зберігаються чутливі змінні середовища

Команда:

```
nano .env
```

Відгук:

Заповнюємо файл наступним вмістом:

```
GF_SECURITY_ADMIN_PASSWORD=admin
GF_DASHBOARD_DEFAULT_HOME_DASHBOARD_PATH=/etc/grafana/
dashboards/default-dashboard.json
GF_SERVER_ROOT_URL=http://localhost
```

Зберігаємо і закриваємо файл

5. Перевірка, старт створених сервісів

Ми маємо все необхідне для того шоб стартувати наші сервіси для системи логування.

Фінальний результат в теці ДЗ виглядає так:

Тека lesson7
  Тека grafana
    Тека provisioning
      Файл datasources.yaml
  Тека fluentd-loki
    Файл Dockerfile
    Файл fluentd.conf
  Файл docker-compose.yaml
  Файл .env

Стартуємо сервіси

Команда:
```
sudo docker compose up -d
```
Відгук:
```
WARN[0000] /home/daks/lesson7/docker-compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Building 1.2s (8/8) FINISHED
docker:default
 => [fluentd-loki internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 130B
0.0s
 => [fluentd-loki internal] load metadata for docker.io/fluent/fluentd:v1.16-debian                                                  1.1s
 => [fluentd-loki auth] fluent/fluentd:pull token for registry-1.docker.io
0.0s
 => [fluentd-loki internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [fluentd-loki 1/2] FROM docker.io/fluent/fluentd:v1.16-debian@sha256:6b8112e5e937722889e106970000fed331adcd3c7155b8986d9b5ad95574d769                  0.0s
 => CACHED [fluentd-loki 2/2] RUN gem install fluent-plugin-loki
0.0s
 => [fluentd-loki] exporting to image
0.0s
 => => exporting layers
0.0s
 => => writing image sha256:4898a14e13a01924bdbba6bb6dfb9f84f782c39a092473621c53f9f153db6440                                         0.0s
 => => naming to docker.io/library/lesson7-fluentd-loki
0.0s
 => [fluentd-loki] resolving provenance for metadata file
0.0s
```

[+] Running 6/6
 ✔ fluentd-loki Built
0.0s
 ✔ Network lesson7_bridge_network Created
0.1s
 ✔ Container lesson7-loki-1 Started
0.4s
 ✔ Container lesson7-fluentd-loki-1 Started
0.7s
 ✔ Container lesson7-grafana-1 Started
0.7s
 ✔ Container lesson7-log_container_fluentd-1 Started

Слід додати, шо деякі іміджі для цієї вправи в нас вже були скачані, того перший старт 4-х сервісів був такий швидкий. Наступні старти, будуть ще швидше, бо треба буде стартанути лише сервіси. Зараз приведу приклад
Зупиняємо сервіси

Команда:
sudo docker compose down
Відгук:
[+] Running 5/5
 ✔ Container lesson7-grafana-1 Removed
0.2s
 ✔ Container lesson7-log_container_fluentd-1 Removed
10.1s
 ✔ Container lesson7-fluentd-loki-1 Removed
1.1s
 ✔ Container lesson7-loki-1 Removed
2.0s
 ✔ Network lesson7_bridge_network Removed
0.1s

По часу контейнери зупиняються довше ніж стартують

Стартуємо сервіси

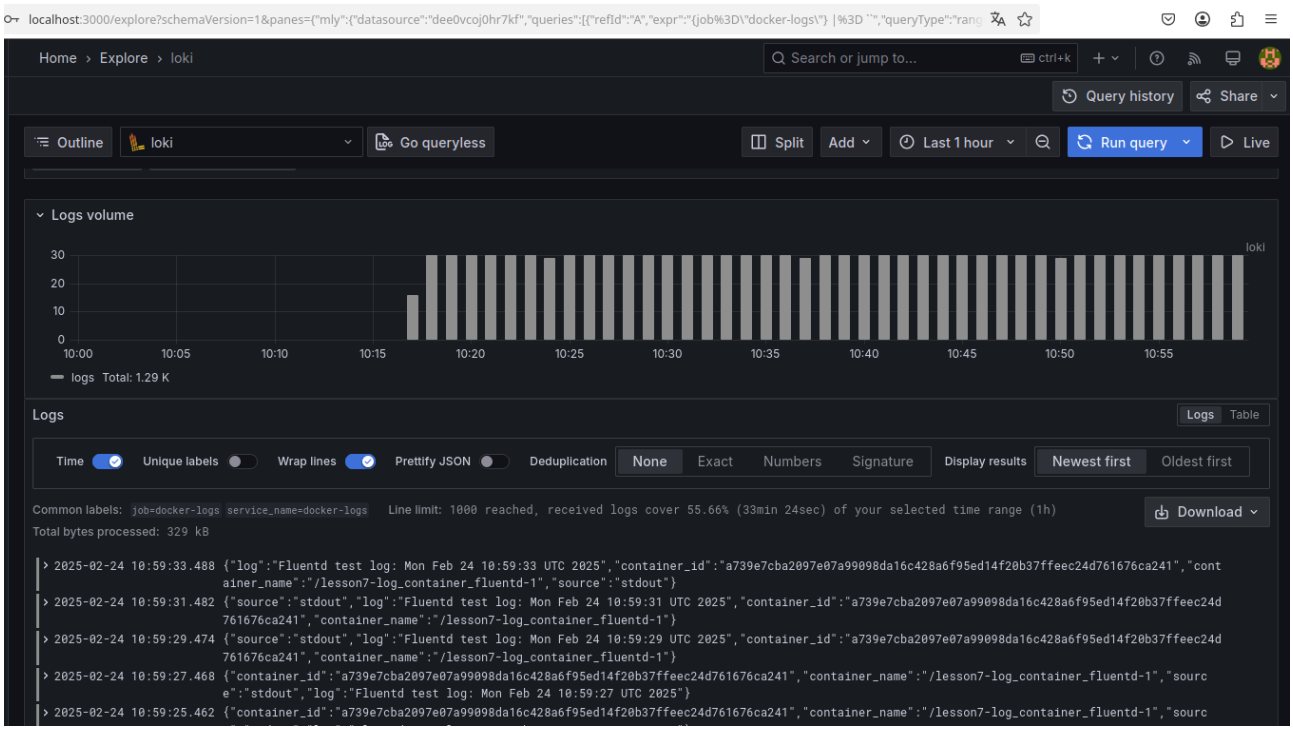
Команда:
sudo docker compose up -d
Відгук:
[+] Running 5/5
 ✔ Network lesson7_bridge_network Created
0.1s
 ✔ Container lesson7-loki-1 Started
0.3s
 ✔ Container lesson7-grafana-1 Started
0.6s
 ✔ Container lesson7-fluentd-loki-1 Started
0.7s
 ✔ Container lesson7-log_container_fluentd-1 Started
1.0s

Бачимо шо для старту 4-х сервісів треба лічені секунди, це безумовна перевага

Останній тест

Команда:
sudo docker ps -a
Відгук:
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
a739e7cba209 busybox "sh -c 'while true; ..." 12 minutes ago Up 12 minutes lesson7-log_container_fluentd-1
beb67503cb26 lesson7-fluentd-loki "tini -- /bin/entryp..." 12 minutes ago Up 12 minutes 5140/tcp, 0.0.0.0:24224->24224/tcp, 0.0.0.0:24224->24224/udp, :::24224->24224/tcp, :::24224->24224/udp lesson7-fluentd-loki-1
b3c70c86a8a9 grafana/grafana "/run.sh" 12 minutes ago Up 12 minutes 0.0.0.0:3000->3000/tcp, :::3000->3000/tcp lesson7-grafana-1
7e7c87b6c770 grafana/loki:latest "/usr/bin/loki -conf..." 12 minutes ago Up 12 minutes 0.0.0.0:3100->3100/tcp, :::3100->3100/tcp lesson7-loki-1

Бачимо шо наші сервіси працюють
Тепер лишилася Grafana, вводимо в браузері адресу http://localhost:3000 і переходимо на неї. Маємо запрошення логіну і паролю, вводимо admin:admin (пароль той, шо ми вказали у файлі .env)
Далі налаштовуємо dashboard для обробки логів від Loki та обираємо це джерело в Explore, далі виконуємо запит {job="docker-logs"} і бачимо наші логи



Домашнє завдання до уроку 7:

Створити docker-compose.yaml для формування макета системи централізованого логування з попереднього завдання. За бажанням контейнер чи контейнери, з яких збиратимуться логи, можна змінити на щось практичніше.

Студент: Олександр Болотов

Дата виконання завдання: 24.02.2025