

**ЗВІТ**  
**ЗАВДАННЯ 1**  
**ВАРІАНТ 1**  
**ХУТКА ХРИСТИНА ПМП–22**

**ЗАВДАННЯ:**

-Написати bash скрипт, який буде перевіряти код відповіді веб сервера. Якщо код відповіді не 2xx або 3xx–то робити запис в лог файл, який користувач зможе продивитися. (додаткове завдання: замість лог файла–відправляти повідомлення на пошту).

-Описати Dockerfile, який буде запускати ваш bash скрипт за розкладом.

-Написати docker-compose, який буде створювати два контейнера, веб сервер(apache or nginx)та контейнер с bash скриптом, який повинен за розкладом(наприклад кожні 30 секунд)запускати ваш скрипт та перевіряти статус.

**ХІД ВИКОНАННЯ**

- 1) Пишемо bash script, який прийматиме код відповіді сервера:
  - Визначаємо URL-адресу, яку потрібно перевірити.
  - Виконуємо запит за допомогою утиліти curl та отримуємо код відповіді.
  - Перевіряємо, чи належить код відповіді діапазону 2xx або 3xx.
  - Якщо так, виводимо повідомлення на екран, у протилежному випадку робимо запис у лог файл error.log. Крім цього надсилаємо імейл про помилку, у якому міститься отриманий код відповіді. Робимо це за допомогою сервісу MailGun.

```

$ my_script.sh
1  #!/bin/bash
2  url="http://localhost:8080"
3  response=$(curl --write-out %{http_code} --silent --output /dev/null "$url")
4
5  if [[ "$response" =~ ^(2|3)[0-9]{2}$ ]]; then
6      echo "Response code is $response. OK"
7  else
8      curl -s --user 'api:e833fca264caf6c1adcf065921d944a3-181449aa-fb5a0dfc' \
9          https://api.mailgun.net/v3/sandboxbe80319ed83d4a2ab36ae43fb081abfd.mailgun.org/messages \
10         -F from='Excited User <mailgun@sandboxbe80319ed83d4a2ab36ae43fb081abfd.mailgun.org>' \
11         -F to=kkhutka@gmail.com \
12         -F subject='Server Response' \
13         -F text='Response code is not 2xx or 3xx'
14      echo "Response code is $response. Error"
15      echo "$(date): $url returned $response" >> error.log
16  fi
17

```

## 2) Створюємо Docker file:

- Оновлюємо список пакетів та встановлюємо утиліту curl.
- Копіюємо файл my\_script.sh в директорію /usr/local/bin/.
- Надаємо права на виконання файлу зі скриптом.
- Пишемо команду CMD, яка буде виконуватися при запуску контейнера. Команда запускається в безкінечному циклі, виконуючи скрипт скрипт кожні 30 секунд.

```

Dockerfile > ...
1  FROM ubuntu:latest
2
3  RUN apt-get update && apt-get -y install curl
4
5  COPY my_script.sh /usr/local/bin/my_script.sh
6  RUN chmod +x /usr/local/bin/my_script.sh
7  RUN chmod 777 /usr/local/bin/error.log
8
9
10 CMD while true; do /usr/local/bin/my_script.sh; sleep 30; done
11

```

## Створюємо docker-compose.yml:

- Створюємо дві послуги : webserver та script.
- script використовує офіційний образ Nginx та розгортається на порту 80.
- обираємо назву контейнера, а також створюємо директорію volume

- При запуску docker-compose up буде створено два контейнери: один з Nginx в ролі веб-сервера, а другий з контейнером з bash скриптом, який запускається кожні 30 секунд та перевіряє статус веб-сервера.

```
🚢 docker-compose.yml
1  version: '3'
2
3  services:
4    webserver:
5      image: nginx:latest
6      ports:
7        - "80:80"
8
9      script:
10       build: .
11       container_name: my_compose
12       volumes:
13         - ./my_script.sh:/usr/local/bin/my_script.sh
14       depends_on:
15         - webserver
```

У результаті, запустивши контейнери, отримуємо:



≡ error.log

```
1 Sat Apr 29 01:56:48 UTC 2023: http://localhost:8080 returned 000
2 Sat Apr 29 01:57:18 UTC 2023: http://localhost:8080 returned 000
3 Sat Apr 29 01:58:04 UTC 2023: http://localhost:8080 returned 000
4 Sat Apr 29 01:58:34 UTC 2023: http://localhost:8080 returned 000
5 Sat Apr 29 01:59:59 UTC 2023: http://localhost:8080 returned 000
6 Sat Apr 29 02:00:29 UTC 2023: http://localhost:8080 returned 000
7 Sat Apr 29 02:00:59 UTC 2023: http://localhost:8080 returned 000
8 Sat Apr 29 02:01:30 UTC 2023: http://localhost:8080 returned 000
9 Sat Apr 29 02:02:00 UTC 2023: Follow link (cmd + click) returned 000
10 Sat Apr 29 02:02:26 UTC 2023: http://localhost:8080 returned 000
11 Sat Apr 29 02:02:57 UTC 2023: http://localhost:8080 returned 000
12 Sat Apr 29 02:03:27 UTC 2023: http://localhost:8080 returned 000
13 Sat Apr 29 02:03:51 UTC 2023: http://localhost:8080 returned 000
14 Sat Apr 29 02:04:21 UTC 2023: http://localhost:8080 returned 000
15 Sat Apr 29 02:04:51 UTC 2023: http://localhost:8080 returned 000
16
```