Опис завдання:

Створити макет системи централізованого логування, яка повинна складатись з наступних елементів:

1. Контейнер, який генерує постійний потік логів та використовує драйвер fluentd
2. Контейнер з fluentd, який буде приймати лог з першого контейнера та передавати їх до Loki
3. Контейнер з Loki, який прийматиме та зберігатиме логи
4. Контейнер з Grafana, який буде мати Loki у якості Data source та призначатиметься для аналізу зібраних логів

Результат завдання:

1. Завантажити на сервер файл з командами та покроковими інструкціями для виконання завдання
2. завантажити всі файли використані для створення та налаштування контейнерів

План

1. Створити свою мережу
2. Запуск Fluentd для збору логів
3. Перенаправлення логів log_container у Fluentd
4. Запуск Loki для зберігання логів
5. Запуск Grafana
6. Перевірка логів за допомогою Grafana

1.Створюємо власну мережу типу bridge

Команда:

```
sudo docker network create --driver bridge bridge_network
```

Відгук:

```
dcb8e9a777d2738e7559ebbdf357a510c708ec3f94a1588c7ea5ac2f3bd2ea4b
```

Перевіряємо чи з’явилася наша нова мережа

Команда:

```
sudo docker network ls
```

Відгук:

```
NETWORK ID     NAME              DRIVER    SCOPE
c669528b28d2   bridge            bridge    local
dcb8e9a777d2   bridge_network    bridge    local
bdafb993de32   host              host      local
2695dbcfa34b   none              null      local
```

Створена нами мережа bridge_network типу bridge з’явилася

2.Запуск Fluentd для збору логів

Створюємо нову теку для цього ДЗ

*** далі суто команди ***

```
$ mkdir lesson6
$ cd lesson6
```

Створюємо Dockerfile для іміджу Fluentd з плагіном Loki

Команда:

```
nano Dockerfile
```

Відгук:

```
...
```

Заповнюємо новостворений файл цім вмістом

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

Зберігаємо файл

Збираємо імідж Fluentd з плагіном Loki

Команда:

```
sudo docker build -t fluentd-loki .
```

Відгук:

```
[+] Building 21.0s (7/7) FINISHED
docker:default
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 130B
0.0s
 => [internal] load metadata for docker.io/fluent/fluentd:v1.16-
debian
2.2s
 => [auth] fluent/fluentd:pull token for registry-1.docker.io
0.0s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [1/2] FROM docker.io/fluent/fluentd:v1.16-
debian@sha256:6b8112e5e937722889e106970000fed331adcd3c7155b8986d9b
5ad95574d769                                      15.1s
```

```
 => => resolve docker.io/fluent/fluentd:v1.16-
debian@sha256:6b8112e5e937722889e106970000fed331adcd3c7155b8986d9b
5ad95574d769                                       0.0s
 => =>
sha256:afe44aabe7c35e08999092b873f0f92280866aba6a23baa7b0a0bda14ac
0ea08 10.95kB / 10.95kB
0.0s
 => =>
sha256:861a678977945e19b284da49bf4778f03e8e9862b4c300f591c0a8904f0
8f0f8 13.86MB / 13.86MB
7.6s
 => =>
sha256:5e2ffead034607804494635b1a4487f595b774567fa61941369bbd01435
ba37c 196B / 196B
0.8s
 => =>
sha256:6b8112e5e937722889e106970000fed331adcd3c7155b8986d9b5ad9557
4d769 2.20kB / 2.20kB
0.0s
 => =>
sha256:e4fff0779e6ddd22366469f08626c3ab1884b5cbe1719b26da238c95f24
7b305 29.13MB / 29.13MB
10.6s
 => =>
sha256:a8fa68e86ee9f815cf3654150433fc2e7f4a01327d5f6bf421e3cfd115e
2a94e 34.90MB / 34.90MB
9.8s
 => =>
sha256:948e7b65e90227b278163731371929b1f6b819091c4f12aaa8f5eec53d9
545a8 143B / 143B
8.0s
 => =>
sha256:7024a380c5b4540aa6cfaa697ffae6bc47dd87fa27226387537c8145503
10254 14.57MB / 14.57MB
12.1s
 => =>
sha256:46a518ea7a06cafe22ea2859654690e84a8ccaf5684f7533e4aaab268c9
9891e 1.23kB / 1.23kB
10.2s
 => =>
sha256:bb542afa2f6c96794ae92f4c259661e96e524d240d1c3ab7112d0991cfc
91ba5 405B / 405B
10.7s
 => =>
sha256:4717ed4817906760b0665dc5a0050a4a67edc6cfa9976a6401248aead8f
c619a 481B / 481B
11.2s
 => => extracting
sha256:e4fff0779e6ddd22366469f08626c3ab1884b5cbe1719b26da238c95f24
7b305
1.8s
```

```
 => => extracting
sha256:861a678977945e19b284da49bf4778f03e8e9862b4c300f591c0a8904f0
8f0f8
0.6s
 => => extracting
sha256:5e2ffead034607804494635b1a4487f595b774567fa61941369bbd01435
ba37c
0.0s
 => => extracting
sha256:a8fa68e86ee9f815cf3654150433fc2e7f4a01327d5f6bf421e3cfd115e
2a94e
0.8s
 => => extracting
sha256:948e7b65e90227b278163731371929b1f6b819091c4f12aaa8f5eec53d9
545a8
0.0s
 => => extracting
sha256:7024a380c5b4540aa6cfaa697ffae6bc47dd87fa27226387537c8145503
10254
0.7s
 => => extracting
sha256:46a518ea7a06cafe22ea2859654690e84a8ccaf5684f7533e4aaab268c9
9891e
0.0s
 => => extracting
sha256:bb542afa2f6c96794ae92f4c259661e96e524d240d1c3ab7112d0991cfc
91ba5
0.0s
 => => extracting
sha256:4717ed4817906760b0665dc5a0050a4a67edc6cfa9976a6401248aead8f
c619a
0.0s
 => [2/2] RUN gem install fluent-plugin-loki
3.4s
 => exporting to image
0.0s
 => => exporting layers
0.0s
 => => writing image
sha256:2d191476fbb2713e66c24e271cafc933b57eddc413afbea39bbc6058ad5
ed42b
0.0s
 => => naming to docker.io/library/fluentd-loki
```

Перевіряємо, чи з’явився у нас новий імідж

Команда:

```
sudo docker image ls
```

Відгук:

```
REPOSITORY                  TAG       IMAGE ID       CREATED
SIZE
```

```
fluentd-loki                latest    2d191476fbb2   2 minutes
ago   213MB
```

Новий імідж fluentd-loki з’явився

Створимо конфігураційний файл fluentd.conf для запуску Fluentd

Команда:

```
nano fluentd.conf
```

Відгук:

...

Заповнюємо файл цім змістом

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

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

Зберігаємо файл

Тепер запустимо Fluentd:

Команда:

```
sudo docker run -d --name fluentd --network bridge_network -v
$(pwd)/fluentd.conf:/fluentd/etc/fluent.conf -p 24224:24224 -
p 24224:24224/udp fluentd-loki
```

Відгук:

```
c8acaab8352ce2bb8c8b663dcb4f1161727390f6711d95d46cdc94f83bee05b6
```

Контейнер запущено і він має приймати логи на порту 24224
Перевірмо це

Команда:

```
sudo docker ps -a
```

Відгук:

```
CONTAINER ID   IMAGE                  COMMAND
CREATED        STATUS        PORTS
NAMES
c8acaab8352c   fluentd-loki           "tini -- /bin/entryp…"   4
minutes ago   Up 4 minutes   5140/tcp, 0.0.0.0:24224->24224/tcp,
0.0.0.0:24224->24224/udp, :::24224->24224/tcp, :::24224->24224/udp
fluentd
```

Бачимо шо контейнер fluentd-loki стартував і працює добре на призначених йому портах

3.Перенаправлення логів log_container у Fluentd

Запустимо контейнер з busybox якому вкажемо мережу в якій він буде працювати, драйвер логування це буде fluentd, вкажемо адресу і порт де саме у нас знаходиться fluentd, а також задамо команду для генерації логів

Команда:

```
sudo docker run -d --name log_container_fluentd --network
bridge_network --log-driver=fluentd --log-opt fluentd-
address=localhost:24224 busybox sh -c 'while true; do echo
"Fluentd test log: $(date)"; sleep 2; done'
```

Відгук:

```
840cf8b16e2851ae2505d8907b734e67060471c76afd551750223bf0bb6335f1
```

Перевіримо чи добре працює наш контейнер з busybox

Команда:

```
sudo docker ps -a
```

Відгук:

```
CONTAINER ID   IMAGE                  COMMAND
CREATED        STATUS        PORTS
NAMES
840cf8b16e28   busybox                "sh -c 'while true; …"   2
minutes ago   Up 2 minutes hours
log_container_fluentd
c8acaab8352c   fluentd-loki           "tini -- /bin/entryp…"   19
minutes ago   Up 19 minutes   5140/tcp, 0.0.0.0:24224->24224/tcp,
0.0.0.0:24224->24224/udp, :::24224->24224/tcp, :::24224->24224/udp
fluentd
```

Бачимо шо контейнер з busybox стартував, працює добре і тепер у нас два запущених контейнера. Новий log_container_fluentd має передавати свої логи у контейнер з Fluentd

4.Запуск Loki для зберігання логів

Тепер для зберігання лог-файлів які генерує контейнер з busybox нам треба запустити Loki. Який має бути у мережі bridge_network для того шоб співпрацювати з іншими контейнерами нашої системи логування

Команда:

```
sudo docker run -d --name loki --network bridge_network -p
3100:3100 grafana/loki:latest
```

Відгук:

```
Unable to find image 'grafana/loki:latest' locally
latest: Pulling from grafana/loki
38f4a9ccb8d6: Pull complete
2e4cf50eeb92: Pull complete
d44d440ac96b: Pull complete
0f8b424aa0b9: Pull complete
```

```
d557676654e5: Pull complete
d82bc7a76a83: Pull complete
d858cbc252ad: Pull complete
1069fc2daed1: Pull complete
b40161cd83fc: Pull complete
3f4e2c586348: Pull complete
80a8c047508a: Pull complete
5391af8231d5: Pull complete
380288218376: Pull complete
1d91cb518276: Pull complete
e85d29cd1ed2: Pull complete
b1d8ed633678: Pull complete
Digest:
sha256:58a6c186ce78ba04d58bfe2a927eff296ba733a430df09645d56cdc158f
3ba08
Status: Downloaded newer image for grafana/loki:latest
0533343bafcbf796433390dfbdbe4668c3b7786fbcf01ffe09ab3ed2167fcfcb
```

Контейнер з Loki стартував і має бути доступний за адресою http://localhost:3100 і також має приймати логи від Fluentd

Перевірмо чи працює контейнер і на яких саме портах, а збір логів підтвердимо пізніше, коли піднімемо контейнер з Grafana

Команда:

```
sudo docker ps -a
```

Відгук:

```
CONTAINER ID   IMAGE                  COMMAND
CREATED        STATUS        PORTS
NAMES
0533343bafcb   grafana/loki:latest   "/usr/bin/loki -conf…"   2
minutes ago   Up 2 minutes   0.0.0.0:3100->3100/tcp, :::3100-
>3100/tcp
loki
840cf8b16e28   busybox               "sh -c 'while true; …"   9
minutes ago   Up 9 minutes hours
log_container_fluentd
c8acaab8352c   fluentd-loki          "tini -- /bin/entryp…"   26
minutes ago   Up 26 minutes   5140/tcp, 0.0.0.0:24224->24224/tcp,
0.0.0.0:24224->24224/udp, :::24224->24224/tcp, :::24224->24224/udp
fluentd
```

Бачимо всі три наші контейнери які працюють добре на призначений ним портах

5.Запуск Grafana

Створимо конфігураційний файл для підключення Loki

*** далі суто команди ***

```
$ mkdir -p provisioning/datasources
```

```
$ nano provisioning/datasources/datasources.yaml
```

Заповнимо файл datasources.yaml таким вмістом

```
apiVersion: 1

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

Зберігаємо файл

Тепер стартуємо контейнер з grafana який має таки опції:

--network bridge_network нашу мережу в якій у нас працює вся система логування
-p 3000:3000 порти на яких саме у нас моє працювати grafana
-e "GF_SECURITY_ADMIN_PASSWORD=admin" - вкажімо пароль за замовченням для адміна
-e "GF_DASHBOARD_DEFAULT_HOME_DASHBOARD_PATH=/etc/grafana/dashboards/default-dashboard.json" - шлях до файлу з налаштуваннями по замовченню для дашборда
-e "GF_SERVER_ROOT_URL=http://localhost:3000" - адреса та порт на якому має працювати grafana
-v $(pwd)"/provisioning:/etc/grafana/provisioning" - файл для grafana з якого вона має підхопити налаштування для отримання логів з Loki

Стартуємо контейнер

Команда:

```
sudo docker run -d --name grafana --network bridge_network -p 3000:3000 -e "GF_SECURITY_ADMIN_PASSWORD=admin" -e "GF_DASHBOARD_DEFAULT_HOME_DASHBOARD_PATH=/etc/grafana/dashboards/default-dashboard.json" -e "GF_SERVER_ROOT_URL=http://localhost:3000" -v $(pwd)"/provisioning:/etc/grafana/provisioning" grafana/grafana
```

Відгук:

```
Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
66a3d608f3fa: Pull complete
b1aef3685777: Pull complete
4ccbda8be621: Pull complete
b1297553a5ad: Pull complete
a7831deda8b6: Pull complete
6a05763a1f35: Pull complete
```

```
ee9115344252: Pull complete
55b26e77bbee: Pull complete
e19370890b86: Pull complete
48c6994f6952: Pull complete
Digest:
sha256:8b37a2f028f164ce7b9889e1765b9d6ee23fec80f871d156fbf436d6198
d32b7
Status: Downloaded newer image for grafana/grafana:latest
ffc4433b3278f51d3972b0279c2ffc76bb9b348e843ab299fd30bcc7d521aead
```

Команда відпрацювала, контейнер стартував, подивимося як він працює

Команда:

```
sudo docker ps -a
```

Відгук:

```
CONTAINER ID   IMAGE                  COMMAND
CREATED              STATUS                          PORTS
NAMES
ffc4433b3278   grafana/grafana        "/run.sh"                30
seconds ago   Up 29 seconds   0.0.0.0:3000->3000/tcp, :::3000-
>3000/tcp
```

Контейнер з Grafana працює добре

Перевірка роботи контейнерів

Команда:

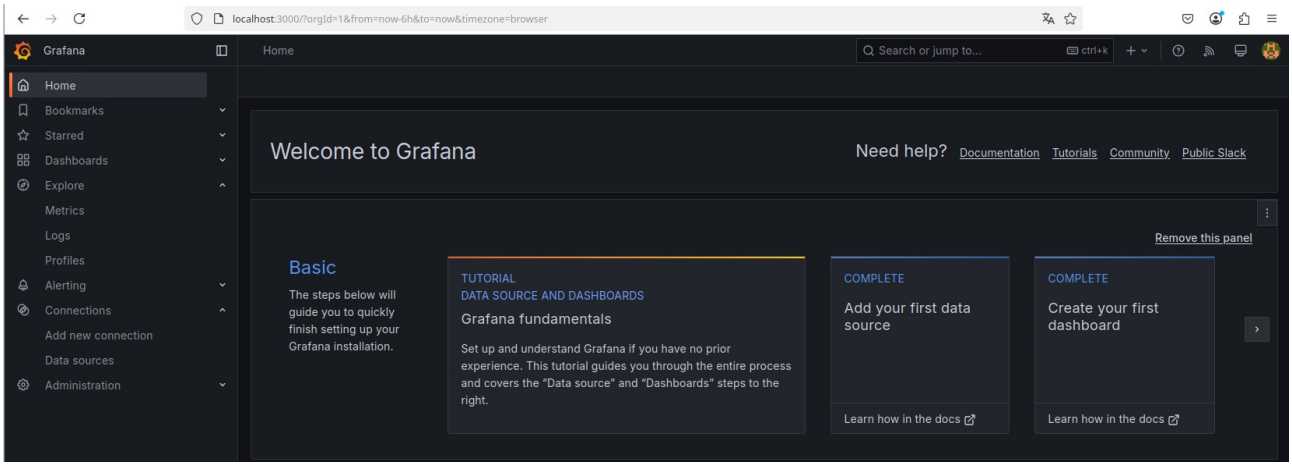
```
sudo docker ps -a
```

Відгук:

```
CONTAINER ID   IMAGE                  COMMAND
CREATED              STATUS              PORTS
NAMES
ffc4433b3278   grafana/grafana        "/run.sh"                54
seconds ago       Up 53 seconds
0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
grafana
0533343bafcb   grafana/loki:latest    "/usr/bin/loki -conf…"
About an hour ago   Up About an hour
0.0.0.0:3100->3100/tcp, :::3100->3100/tcp
loki
840cf8b16e28   busybox                "sh -c 'while true; …"
About an hour ago   Up About an hour
log_container_fluentd
c8acaab8352c   fluentd-loki           "tini -- /bin/entryp…" About
an hour ago   Up About an hour   5140/tcp,
0.0.0.0:24224->24224/tcp, 0.0.0.0:24224->24224/udp, :::24224-
>24224/tcp, :::24224->24224/udp   fluentd
```

Успіх))

Тепер перейдемо до вебінтерфейсу grafana, за адресою - http://localhost:3000 - перевіримо як вона працює і чи працює загалом. І тут успіх, вебінтерфейс працює. Змінюємо пароль за замовченням для адміна і входимо до системи аналізу логів Grafana



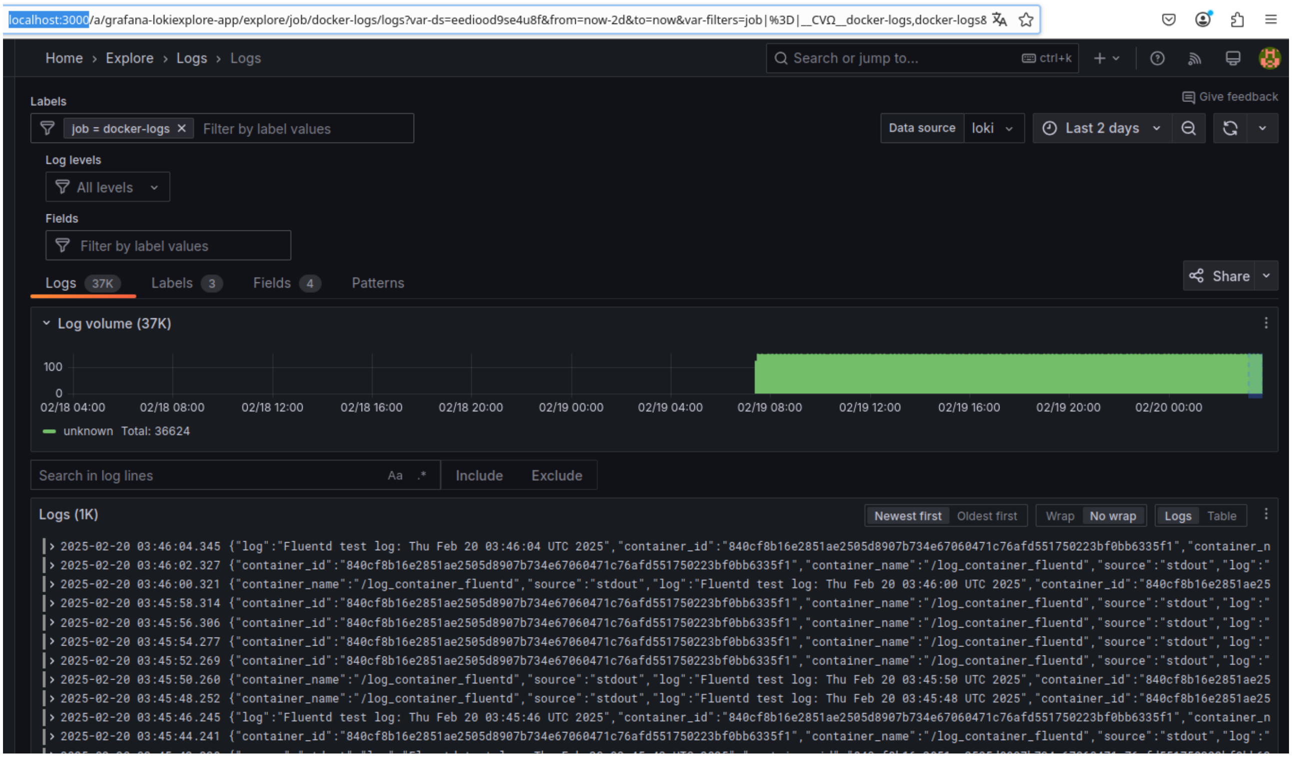
6.Перевірка логів за допомогою Grafana

Йдемо на вебінтерфейс Grafana за посиланням - http://localhost:3000 – вводимо наш логін і пароль (поки шо він за замовченням – admin:admin), заходимо на дашборд.
Переходимо в розділ Explore де обираємо джерелом Loki.
Наступним кроком виконуємо запит {job="docker-logs"} і бачимо наші логи які Fluentd отримав від log_container_fluentd та передав у Loki

Шось там логи купкою не дуже копіюються, на зображені у pdf версії все видно. А у текстовий варіант надам пару записів

{"container_id":"840cf8b16e2851ae2505d8907b734e67060471c76afd551750223bf0bb6335f1","container_name":"/log_container_fluentd","source":"stdout","log":"Fluentd test log: Thu Feb 20 10:29:00 UTC 2025"}

{"container_id":"840cf8b16e2851ae2505d8907b734e67060471c76afd551750223bf0bb6335f1","container_name":"/log_container_fluentd","source":"stdout","log":"Fluentd test log: Thu Feb 20 10:28:00 UTC 2025"}



Підсумок
За підсумком контейнер з Grafana пропрацював без зауважень біля 20 годин, весь цей час йшли логи, які можна було аналізувати через вебінтерфейс Grafana, тобто вся система працювала добре

```
CONTAINER ID   IMAGE                  COMMAND
CREATED        STATUS        PORTS
NAMES
ffc4433b3278   grafana/grafana        "/run.sh"                 19
hours ago   Up 19 hours   0.0.0.0:3000->3000/tcp,
:::3000->3000/tcp
grafana
0533343bafcb   grafana/loki:latest    "/usr/bin/loki -conf…"   20
hours ago   Up 20 hours   0.0.0.0:3100->3100/tcp,
:::3100->3100/tcp
loki
840cf8b16e28   busybox                "sh -c 'while true; …"   20
hours ago   Up 20 hours
log_container_fluentd
c8acaab8352c   fluentd-loki           "tini -- /bin/entryp…"   20
hours ago   Up 20 hours   5140/tcp, 0.0.0.0:24224->24224/tcp,
0.0.0.0:24224->24224/udp, :::24224->24224/tcp, :::24224->24224/udp
fluentd
```

Домашнє завдання до уроку 6:

Створити макет системи централізованого логування, яка повинна складатись з наступних елементів:

5. Контейнер, який генерує постійний потік логів та використовує драйвер fluentd
6. Контейнер з fluentd, який буде приймати лог з першого контейнера та передавати їх до Loki
7. Контейнер з Loki, який прийматиме та зберігатиме логи
8. Контейнер з Grafana, який буде мати Loki у якості Data source та призначатиметься для аналізу зібраних логів

Студент: Олександр Болотов

Дата виконання завдання: 20.02.2025