Instalacja Grayloga w Dockerze: Graylog, MongoDB i OpenSearch

Typ: procedura

Dotyczy: Graylog 7.1.9, MongoDB 8.0, OpenSearch 2.19.6 (sprawdzone na go2docs.graylog.org i hub.docker.com 28.09.2026)

Czas: ~40 min

Efekt: trzy działające kontenery, Graylog, MongoDB i OpenSearch, połączone w jeden system dostępny po WWW pod `logi.zielona.example`, gotowy na pierwsze wejścia i pierwsze logowanie.

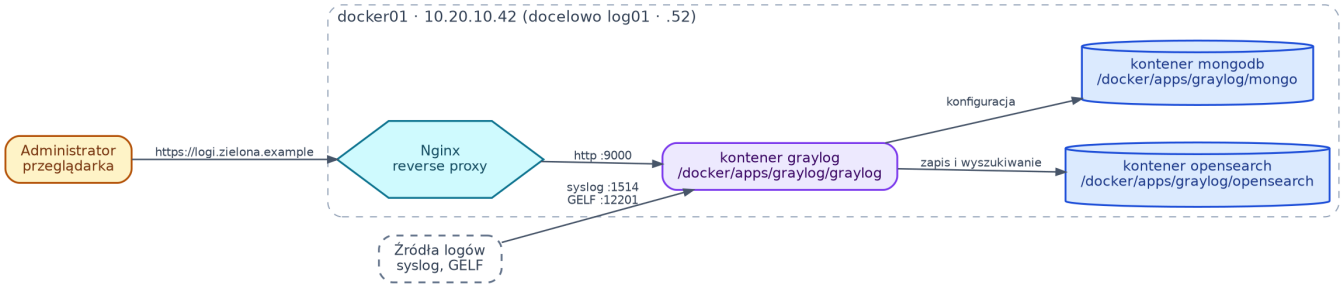
Skróty użyte na tej stronie (6)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
SHA256	funkcja skrótu kryptograficznego dająca 256-bitowy odcisk danych
WebSocket	stałe, dwukierunkowe połączenie przeglądarki z serwerem, przez które Home Assistant na żywo odświeża panele; serwer pośredniczący musi przekazywać nagłówki Upgrade i Connection
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Zanim zaczniemy

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root, na maszynie z zainstalowanym Dockerem i wtyczką Compose. Graylog sam w sobie to tylko jeden z trzech kontenerów: potrzebuje MongoDB na konfigurację i OpenSearch na same logi, opisane szerzej na stronie Graylog: wejścia, strumienie, wyszukiwanie.

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/<nazwa>`, a plik Compose w `/docker/compose/<nazwa>/compose.yaml`. To konwencja, nie wymóg Dockera: można ją zmienić, byle w całej firmie była jedna. Katalog `/docker` to osobny wolumen `vgDocker/lvDocker`, żeby pełny dysk aplikacji nie zatrzymał systemu.



Graylog, MongoDB i OpenSearch jako trzy kontenery na docker01, za Nginksem, ze źródłami logów wchodzącymi na porty 1514 i 12201 (kliknięcie otwiera pełny rozmiar)

Wymaganie	Uwaga
Docker Engine i wtyczka Compose	sprawdzamy `docker compose version`
`vm.max_map_count` od `262144` w górę	wymóg OpenSearch, ustawiamy w kroku 1
Wolny port `9000` na hoście	interfejs i API Graylogu, zwykle za Nginksem, nie wystawiany wprost na świat
Porty `1514` i `12201` wolne, tylko jeśli faktycznie ich użyjemy	pierwsze wejście syslog i GELF, patrz Graylog: wejścia, strumienie, wyszukiwanie
Wpis DNS `logi.zielona.example` na `docker01`	patrz Świat przykładów: Zielona po adres
Certyfikat dla `logi.zielona.example`	z CA Zielonej albo Let's Encrypt, do Nginksa w kroku 5

Uważaj: obrazy zawsze przypinamy do konkretnej wersji, nigdy do `latest`. OpenSearch dodatkowo ma górną granicę: Graylog 7.1.x wspiera OpenSearch tylko do wersji 2.19.x, a wersje 3.0 i nowsze nie są przez niego obsługiwane, więc podmiana tagu na `latest` przy aktualizacji potrafi po cichu wystawić kombinację, której Graylog w ogóle nie rozumie.

> Dziś Graylog stoi na `docker01` razem z Gitea i resztą aplikacji Zielonej. Docelowo przenosimy go na dedykowany `log01` (10.20.10.52): przenosiny to kopia katalogów `/docker/apps/graylog` i `/docker/compose/graylog` na nowy host, zmiana wpisu DNS `logi.zielona.example` i przepięcie Nginksa, nie osobna instalacja od zera.

Przebieg

1. Ustawienie vm.max_map_count na hoście

OpenSearch odmawia startu, jeśli system nie pozwala procesowi zmapować odpowiednio dużej liczby obszarów pamięci. Ustawiamy to trwale, żeby przetrwało restart hosta:

```
sudo tee /etc/sysctl.d/99-opensearch.conf <<< "vm.max_map_count=262144"
sudo sysctl --system
```

Przykład:

```
* Applying /etc/sysctl.d/99-opensearch.conf ...
vm.max_map_count = 262144
```

2. Katalogi na dane, zgodnie z konwencją ścieżek

```
sudo mkdir -p /docker/apps/graylog/mongo /docker/apps/graylog/opensearch
/docker/apps/graylog/graylog /docker/apps/graylog/snapshots
sudo mkdir -p /docker/compose/graylog
```

Kontenery OpenSearch i Grayloga działają jako zwykli użytkownicy, nie jako root, więc katalog założony przez `mkdir` (właściciel root) blokuje im zapis i kontener nie wstaje. Numery użytkowników sprawdzamy w obrazach, nie przepisujemy z przykładu:

```
docker run --rm --entrypoint id opensearchproject/opensearch:2.19.6
docker run --rm --entrypoint id graylog/graylog:7.1.9
```

```
sudo chown -R NumerUID:NumerGID /docker/apps/graylog/opensearch /docker/apps/graylog/snapshots
sudo chown -R NumerUID:NumerGID /docker/apps/graylog/graylog
```

> (gdzie `NumerUID:NumerGID` to numery z wyniku poprzedniej komendy, osobne dla OpenSearch i osobne dla Grayloga)

MongoDB nie wymaga tego kroku: obraz sam nadaje właściciela katalogowi `/data/db`. Pełny opis z przykładowym wynikiem jest na stronie Graylog: przeprowadzka z docker01 na log01.

3. Wygenerowanie sekretów Grayloga

> **Pojęcie: GRAYLOG_PASSWORD_SECRET.** Losowy ciąg znaków, którym Graylog szyfruje hasła i sesje przechowywane w MongoDB. Musi mieć przynajmniej 16 znaków, w praktyce generujemy dłuższy i nie zmieniamy go już później, bo zmiana unieważnia wszystkie zapisane hasła.

```
< /dev/urandom tr -dc 'A-Za-z0-9' | head -c96; echo
```

Przykład:

```
qT8fRZmN2vLxKp...(96 znaków)
```

> **Pojęcie: GRAYLOG_ROOT_PASSWORD_SHA2.** Skrót SHA256 hasła konta `admin`, jedynego konta zapisanego wprost w konfiguracji, a nie w bazie. Graylog nigdy nie przechowuje tego hasła jawnym tekstem, więc wpisujemy tu wynik komendy, nie samo hasło:

```
echo -n 'HasloAdministratora' | sha256sum | cut -d' ' -f1
```

Przykład:

```
8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a91
```

Oba wyniki wklejamy do pliku `/docker/compose/graylog/.env`, obok pliku `compose.yaml`, nie w nim samym:

```
GRAYLOG_PASSWORD_SECRET=qT8fRZmN2vLxKp...(96 znaków)
GRAYLOG_ROOT_PASSWORD_SHA2=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a91
```

4. Plik compose.yaml

Plik `/docker/compose/graylog/compose.yaml`:

```
services:
  mongodb:
    image: mongo:8.0
    container_name: graylog-mongodb
    restart: always
    volumes:
      - /docker/apps/graylog/mongo:/data/db

  opensearch:
    image: opensearchproject/opensearch:2.19.6
    container_name: graylog-opensearch
    restart: always
    environment:
      - discovery.type=single-node
      - plugins.security.disabled=true
      - bootstrap.memory_lock=true
      - OPENSEARCH_JAVA_OPTS=-Xms1g -Xmx1g
      - path.repo=/usr/share/opensearch/snapshots
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    volumes:
      - /docker/apps/graylog/opensearch:/usr/share/opensearch/data
      - /docker/apps/graylog/snapshots:/usr/share/opensearch/snapshots

  graylog:
    image: graylog/graylog:7.1.9
    container_name: graylog
    restart: always
    depends_on:
      - mongodb
      - opensearch
    environment:
      - GRAYLOG_PASSWORD_SECRET=${GRAYLOG_PASSWORD_SECRET}
```

```
      - GRAYLOG_ROOT_PASSWORD_SHA2=${GRAYLOG_ROOT_PASSWORD_SHA2}
      - GRAYLOG_HTTP_BIND_ADDRESS=0.0.0.0:9000
      - GRAYLOG_HTTP_EXTERNAL_URI=https://logi.zielona.example/
      - GRAYLOG_ELASTICSEARCH_HOSTS=http://opensearch:9200
      - GRAYLOG_MONGODB_URI=mongodb://mongodb:27017/graylog
    volumes:
      - /docker/apps/graylog/graylog:/usr/share/graylog/data
    ports:
      - "9000:9000/tcp"
      - "1514:1514/tcp"
      - "1514:1514/udp"
      - "12201:12201/tcp"
      - "12201:12201/udp"
```

Ustawienie `path.repo` i wolumen `snapshots` przygotowują miejsce na migawki OpenSearch. Bez nich późniejsza rejestracja repozytorium kopii (opisana na stronie Kopia zapasowa i aktualizacja Grayloga) zwróci błąd, a dodanie ich po fakcie wymaga odtworzenia kontenera. Jeśli migawek nie planujemy, możemy te dwie linie pominąć.

Port `9200` OpenSearch celowo nie trafia na listę `ports`: Graylog rozmawia z nim po wewnętrznej sieci Dockera, a `plugins.security.disabled=true` wyłącza wbudowane uwierzytelnianie tego portu, więc nie wystawiamy go na hosta, tym bardziej na świat. Porty `1514` i `12201` zostawiamy zmapowane tylko wtedy, gdy rzeczywiście zamierzamy z nich korzystać: dokładamy albo usuwamy pojedynczą parę linii, zależnie od tego, które wejście skonfigurujemy w interfejsie Graylogu.

5. Pierwsze uruchomienie

```
cd /docker/compose/graylog
docker compose up -d
```

Przykład:

```
[+] Running 4/4
 ✔ Network graylog_default       Created
 ✔ Container graylog-mongodb     Started
 ✔ Container graylog-opensearch  Started
 ✔ Container graylog             Started
```

Pierwszy start trwa dłużej niż zwykły restart: Graylog czeka, aż OpenSearch faktycznie odpowie, i dopiero potem zakłada własne indeksy. Sprawdzamy log, dopóki nie zobaczymy potwierdzenia gotowości:

```
docker compose logs -f graylog
```

Przykład:

```
graylog  | 2026-09-28T10:02:41.331Z INFO [InputSetupService] Triggering launching persisted
inputs
graylog  | 2026-09-28T10:02:41.402Z INFO [ServerBootstrap] Graylog server up and running.
```

6. Reverse proxy: Nginx przed Graylogiem

Fragment vhostu `/etc/nginx/conf.d/logi.zielona.example.conf`, z nagłówkami wymaganymi przez WebSocket, bo interfejs Graylogu odświeża wyniki wyszukiwania w czasie rzeczywistym właśnie po WebSocket, opisanym na stronie Reverse proxy i równoważenie obciążenia:

```
server {
    listen 443 ssl;
    server_name logi.zielona.example;

    ssl_certificate     /etc/nginx/certy/logi.zielona.example.crt;
    ssl_certificate_key /etc/nginx/certy/logi.zielona.example.key;

    location / {
        proxy_pass http://10.20.10.42:9000;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

Weryfikacja

Sprawdzamy, że wszystkie trzy kontenery działają:

```
docker compose ps
```

Przykład:

```
NAME                IMAGE                                    STATUS
graylog             graylog/graylog:7.1.9                    Up 3 minutes
graylog-mongodb     mongo:8.0                                 Up 3 minutes
graylog-opensearch  opensearchproject/opensearch:2.19.6      Up 3 minutes
```

Sprawdzamy stan zdrowia Grayloga przez wbudowany punkt kontrolny API, ten sam, którego używa load balancer:

```
curl -s http://localhost:9000/api/system/lbstatus
```

Przykład:

```
ALIVE
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: pierwszy ekran logowania pod `https://logi.zielona.example`, z widocznym polem „Username" i „Password" oraz numerem wersji Graylogu w stopce.

Wycofanie zmian

Zatrzymanie i usunięcie kontenerów, bez kasowania danych z wolumenów:

```
docker compose down
```

Pełne usunięcie razem z danymi, tylko gdy naprawdę chcemy zacząć od zera:

```
docker compose down -v
sudo rm -rf /docker/apps/graylog
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Kontener `graylog-opensearch` restartuje się w pętli, w logu `max virtual memory areas vm.max_map_count too low`	krok 1 pominięty albo nie przetrwał restartu hosta	`sysctl vm.max_map_count`, w razie potrzeby powtarzamy krok 1

Objaw	Przyczyna	Naprawa
Kontener `graylog` bez końca czeka, w logu brak `Graylog server up and running`	OpenSearch jeszcze nie odpowiada albo `GRAYLOG_ELASTICSEARCH_HOSTS` wskazuje zły adres	`docker compose logs opensearch`, sprawdzamy nazwę usługi w `compose.yaml`
`curl http://localhost:9000/api/system/lbstatus` zwraca błąd połączenia	kontener `graylog` jeszcze się uruchamia albo port `9000` zajęty przez inną usługę	`docker compose logs graylog`, `ss -tlnp \`
Logowanie kontem `admin` odrzucone mimo poprawnego hasła	`GRAYLOG_ROOT_PASSWORD_SHA2` policzony z innego hasła niż wpisywane w formularzu	liczymy skrót ponownie krokiem 3, poprawiamy `.env`, `docker compose up -d`
Strona `https://logi.zielona.example` wisi przy ładowaniu wyników wyszukiwania	brak nagłówków `Upgrade` / `Connection` w Nginksie	dokładamy blok WebSocket z kroku 6

Powiązane: Po co centralne logowanie i jak to działa · Graylog: wejścia, strumienie, wyszukiwanie · Konwencja Dockera w Zielonej · Reverse proxy i równoważenie obciążenia

Źródła: go2docs.graylog.org, rozdział „Docker Installation with Self-Managed OpenSearch" · go2docs.graylog.org, „Compatibility Matrix" · opensearch.org, dokumentacja instalacji Docker · endoflife.date/graylog i endoflife.date/opensearch (sprawdzone 28.09.2026)

Revision #23
Created 2026-09-28 16:57:54 UTC by Tomasz Konieczny
Updated 2026-10-26 14:15:31 UTC by Tomasz Konieczny