Kopia zapasowa i aktualizacja Grayloga

Poziom: operacyjny **Dotyczy: Graylog 7.1 w Dockerze, MongoDB 8.0, OpenSearch 2.19**

Czas: ~45 min

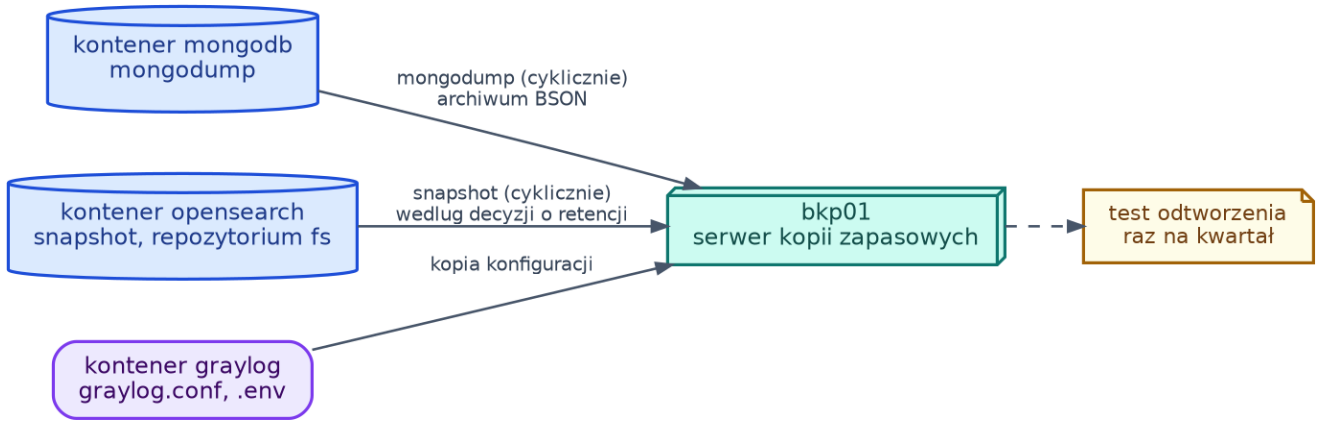
Kopia zapasowa Grayloga to w praktyce trzy osobne rzeczy naraz: zrzut `mongodump` z konfiguracji, świadoma decyzja o snapshotach OpenSearch dla samych logów, i kopia plików `compose.yaml` oraz `.env`. Aktualizacja to podmiana trzech tagów obrazów w konkretnej kolejności, nie jednego naraz.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
BSON	Binary JSON: binarny format zapisu dokumentów używany wewnętrznie przez MongoDB, w tym w archiwach mongodump

Zanim zaczniemy

Wymaganie	Uwaga
Dostęp do `docker exec` na kontenerach `graylog-mongodb` i `graylog-opensearch` na docker01	konto z prawem uruchamiać kontenery Dockera
Miejsce na `bkp01` na regularne kopie	patrz Świat przykładów: Zielona po adres i rolę serwera
Znajomość obecnych tagów obrazów	`docker compose images` w `/docker/compose/graylog/`
Podjęta decyzja o zakresie kopii logów	patrz sekcja „Snapshoty OpenSearch albo świadoma rezygnacja" niżej



Kontenery mongodb, opensearch i graylog wysyłają kopie do bkp01, cyklicznie i z okresowym testem odtworzenia (kliknięcie otwiera pełny rozmiar)

Uważaj: aktualizacja Grayloga migruje schemat konfiguracji w MongoDB przy pierwszym starcie nowej wersji, a nowsza wersja Grayloga bywa zbudowana pod nowszy OpenSearch. Dokumentacja producenta jest tu jednoznaczna: przy własnym, samodzielnie zarządzanym OpenSearch aktualizujemy najpierw MongoDB, potem OpenSearch, dopiero na końcu sam Graylog. Odwrotna kolejność zostawia Graylog wobec silnika, którego jeszcze nie obsługuje, albo wobec bazy konfiguracji w formacie sprzed migracji.

Przebieg

1. Kopia konfiguracji: compose.yaml i .env

W naszej instalacji dockerowej nie edytujemy osobnego pliku `server.conf` (dawniej `graylog.conf`) tak jak przy instalacji pakietem: cała konfiguracja leży w dwóch plikach tekstowych, które kopiujemy jak każdy inny plik.

```
mkdir -p /docker/apps/graylog/kopie
cp /docker/compose/graylog/compose.yaml /docker/compose/graylog/.env
/docker/apps/graylog/kopie/
```

2. Kopia konfiguracji Grayloga z MongoDB: mongodump

```
docker exec graylog-mongodb mongodump --archive --db graylog >
/docker/apps/graylog/kopie/graylog-mongo-2026-09-28.archive
```

Przykład:

```
2026-09-28T02:00:03.104+0000□writing graylog.users to archive on stdout
2026-09-28T02:00:03.221+0000□writing graylog.streams to archive on stdout
2026-09-28T02:00:03.340+0000□done dumping graylog.streams (14 documents)
```

> **Pojęcie: mongodump.** Narzędzie eksportujące zawartość bazy MongoDB do binarnego archiwum BSON, tu jedną komendą pakujące całą bazę `graylog` (użytkowników, wejścia, strumienie, definicje alertów) do pojedynczego pliku, gotowego do skopiowania na `bkp01`.

Opcja `--archive` bez nazwy pliku wysyła archiwum na standardowe wyjście, a przekierowanie `>` zapisuje je od razu w katalogu na hoście. Dzięki temu zrzut nie zajmuje miejsca w kontenerze i nie wymaga osobnego kopiowania `docker cp`. Komunikaty `mongodump` idą na standardowy strumień błędów, więc w pliku jest tylko archiwum. Z hosta kopiujemy je na `bkp01`:

```
scp /docker/apps/graylog/kopie/graylog-mongo-2026-09-28.archive
nazwa_usera@bkp01:/backup/graylog/
```

3. Snapshoty OpenSearch albo świadoma rezygnacja

> **Pojęcie: repozytorium snapshotów.** Miejsce, do którego OpenSearch zapisuje kopie swoich indeksów, zarejestrowane pod nazwą przez wywołanie API, najczęściej typu `fs` (zwykły katalog na dysku, zamontowany też jako wolumen kontenera) albo `s3`. Bez zarejestrowanego repozytorium OpenSearch w ogóle nie przyjmie żądania snapshotu.

Dokumentacja producenta uczciwie dopuszcza dwie strategie naraz, nie tylko pełny backup: `Minimal`, czyli kopia samej konfiguracji i MongoDB bez indeksów logów, oraz pełny backup ze snapshotami OpenSearch. W Zielonej to świadoma decyzja, nie zaniedbanie: jeżeli logi starsze niż nasza retencja i tak nie mają wartości dochodzeniowej, koszt utrzymywania osobnego repozytorium snapshotów i miejsca na `bkp01` bywa wyższy niż ryzyko utraty ich historii przy poważnej awarii `docker01`.

Gdy decydujemy się na pełny backup, najpierw przygotowujemy katalog i usługę OpenSearch. Bez dwóch zmian w `compose.yaml` rejestracja repozytorium typu `fs` się nie uda. Pierwsza to ustawienie `path.repo` w środowisku kontenera, druga to wolumen `snapshots`. Katalog musi też należeć do

użytkownika z obrazu OpenSearch (numery sprawdzamy komendą `docker run --rm --entrypoint id` i nadajemy przez `chown`), jak opisano w Fiszce na stronie Graylog: przeprowadzka z docker01 na log01.

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

Plik `/docker/compose/graylog/compose.yaml`, usługa `opensearch`. Fragment dopisujemy do istniejących list `environment` i `volumes`, nie zastępujemy ich:

```
  opensearch:
    environment:
      - path.repo=/usr/share/opensearch/snapshots
    volumes:
      - /docker/apps/graylog/snapshots:/usr/share/opensearch/snapshots
```

Zmiana wymaga odtworzenia kontenera (`docker compose up -d opensearch`). Dopiero potem rejestrujemy repozytorium, wykonując zapytanie z wnętrza kontenera, bo port `9200` nie jest wystawiony na hosta:

```
docker exec graylog-opensearch curl -s -X PUT "localhost:9200/_snapshot/graylog_backup" \
  -H 'Content-Type: application/json' \
  -d '{"type": "fs", "settings": {"location": "/usr/share/opensearch/snapshots"}}'
```

Przykład:

```
{"acknowledged":true}
```

Samo wykonanie snapshotu, uruchamiane cyklicznie razem z krokiem 2:

```
docker exec graylog-opensearch curl -s -X PUT
"localhost:9200/_snapshot/graylog_backup/snapshot-2026-09-28?wait_for_completion=true"
```

Przykład:

```
{"snapshot":{"snapshot":"snapshot-2026-09-
28","state":"SUCCESS","shards":{"total":6,"failed":0,"successful":6}}}
```

4. Harmonogram: cron albo timer systemd

Wybieramy jeden z dwóch mechanizmów opisanych osobno, cron: zadania cykliczne albo systemd timery: OnCalendar zamiast crona, i spinamy nim kroki 1 do 3 w jeden skrypt uruchamiany co noc.

```
crontab -e
```

Przykład:

```
0 2 * * * /docker/apps/graylog/skrypty/kopia-nocna.sh >> /var/log/graylog-kopia.log 2>&1
```

5. Miejsce przechowywania kopii

Kopie z kroków 1 do 3 trafiają na `bkp01`, do katalogu `/backup/graylog/`, tym samym mechanizmem co reszta serwerów opisanych w Świat przykładów: Zielona. Liczbę przechowywanych kopii i rotację starszych plików ustalamy tak samo jak dla pozostałych aplikacji na `bkp01`.

6. Test odtworzenia

Raz na kwartał odtwarzamy najnowszy zrzut `mongodump` na osobnej, testowej instalacji tej samej wersji Grayloga: `mongorestore` do pustej bazy testowej, wskazanie jej w `GRAYLOG_MONGODB_URI` testowej instancji, pierwszy start. Jeśli w danym kwartale mamy też snapshot OpenSearch, dokładamy jego odtworzenie tym samym testem. Kopia, której nikt nigdy nie spróbował odtworzyć, jest tylko przypuszczeniem, że działa.

Aktualizacja

1. Przeczytanie informacji o wydaniu

Przed zmianą tagów sprawdzamy changelog na `go2docs.graylog.org` i macierz kompatybilności wersji MongoDB i OpenSearch dla docelowej wersji Grayloga, opisaną na stronie Instalacja Grayloga w Dockerze: Graylog, MongoDB i OpenSearch. Wersje docelowe w przykładach poniżej to te same, które stoją w całej książce: Graylog 7.1.9, OpenSearch 2.19.6, MongoDB 8.0 (sprawdzone 03.10.2026). Macierz dla Grayloga 7.1 dopuszcza OpenSearch najwyżej do 2.19.6 i MongoDB do 8.2.x. Nowszej poprawki Grayloga niż 7.1.9 nie udało się potwierdzić, więc przy kolejnej aktualizacji sprawdzamy ją na stronie wydań i podmieniamy tag w przykładach. Skok o więcej niż jedną wersję główną Grayloga naraz zwykle wymaga przejścia przez wersje pośrednie, nie bezpośredniego skoku.

2. Wykonanie kopii sprzed aktualizacji

Powtarzamy kroki 1 do 3 z sekcji wyżej, poza harmonogramem, bezpośrednio przed aktualizacją: to jest kopia, do której wracamy w razie problemu, świeższa niż ostatnia nocna.

3. Aktualizacja MongoDB

```
sed -i 's/mongo:.*/mongo:8.0/' /docker/compose/graylog/compose.yaml
docker compose -f /docker/compose/graylog/compose.yaml pull mongodb
docker compose -f /docker/compose/graylog/compose.yaml up -d mongodb
```

Sprawdzamy, że MongoDB wstała poprawnie, zanim przejdziemy dalej:

```
docker compose -f /docker/compose/graylog/compose.yaml logs mongodb | tail -n 20
```

4. Aktualizacja OpenSearch

```
sed -i 's/opensearchproject\/opensearch:.*/opensearchproject\/opensearch:2.19.6/'
/docker/compose/graylog/compose.yaml
docker compose -f /docker/compose/graylog/compose.yaml pull opensearch
docker compose -f /docker/compose/graylog/compose.yaml up -d opensearch
```

Uważaj: trzymamy się gałęzi 2.x, a według macierzy kompatybilności najwyższy potwierdzony OpenSearch dla Grayloga 7.1 to 2.19.6. Graylog 7.1.x nie obsługuje OpenSearch 3.0 i nowszych, więc `docker compose pull` bez przypiętego tagu (albo z tagiem `latest`) potrafi po cichu ściągnąć wersję, z którą Graylog w ogóle nie wystartuje.

5. Aktualizacja Grayloga

```
sed -i 's/graylog\/graylog:.*/graylog\/graylog:7.1.9/' /docker/compose/graylog/compose.yaml
docker compose -f /docker/compose/graylog/compose.yaml pull graylog
docker compose -f /docker/compose/graylog/compose.yaml up -d graylog
```

Przykład:

```
[+] Running 1/1
 ✔ Container graylog  Started
```

Migracja konfiguracji w MongoDB wykonuje się sama przy starcie nowego obrazu: nie ma osobnej komendy migracji do wywołania ręcznie.

6. Weryfikacja po aktualizacji

```
docker compose -f /docker/compose/graylog/compose.yaml logs graylog | tail -n 30
```

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

Przykład:

```
ALIVE
```

◫◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: stopka ekranu logowania Graylogu po zalogowaniu kontem `admin`, z widocznym numerem wersji zgodnym z nowym tagiem obrazu.

Weryfikacja

```
docker compose -f /docker/compose/graylog/compose.yaml ps
```

Przykład:

```
NAME                IMAGE                                  STATUS
graylog             graylog/graylog:7.1.9                  Up 4 minutes (healthy)
graylog-mongodb     mongo:8.0                               Up 12 minutes
graylog-opensearch  opensearchproject/opensearch:2.19.6     Up 8 minutes
```

Wycofanie zmian

Przed migracją (kopia z kroku 2 aktualizacji jeszcze nie była potrzebna): przerywamy, wracamy do poprzednich tagów w `compose.yaml`, `docker compose up -d` uruchamia stare wersje na niezmienionych jeszcze danych.

Po migracji, gdy nowa wersja działa niepoprawnie: samo cofnięcie tagów nie wystarcza, bo MongoDB ma już nowy schemat konfiguracji, którego stary Graylog może nie rozumieć. Odtwarzamy zamiast tego kopię z kroku 2: `mongorestore` archiwum do bazy `graylog`, przywrócenie `compose.yaml` i `.env` sprzed zmiany, dopiero potem uruchomienie starych tagów obrazów. Jeżeli podjęliśmy decyzję `Minimal` z sekcji o snapshotach, logi zapisane między ostatnią kopią a awarią przepadają: to jest właśnie koszt tej świadomej decyzji, nie efekt uboczny błędu w procedurze.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
`mongodump` kończy się błędem braku miejsca albo plik archiwum ma 0 bajtów	zapchany dysk hosta pod `/docker/apps/graylog/kopie`, albo komenda uruchomiona z `-t` (terminal psuje strumień binarny)	sprawdzamy `df -h /docker`, uruchamiamy `docker exec` bez `-t` i z przekierowaniem `>` do pliku na hoście
Rejestracja repozytorium snapshotów zwraca `repository_exception`	katalog `location` niezamontowany jako wolumen kontenera `graylog-opensearch`	dokładamy wolumen w `compose.yaml`, powtarzamy krok 3
Po aktualizacji kontener `graylog` bez końca się restartuje	MongoDB albo OpenSearch zaktualizowane w złej kolejności, albo pominięte	sprawdzamy kolejność z sekcji „Uważaj" wyżej, w razie wątpliwości odtwarzamy z kopii sprzed aktualizacji
`docker compose pull opensearch` ściąga wersję 3.x	tag w `compose.yaml` bez przypięcia albo ustawiony na `latest`	poprawiamy tag na konkretną wersję 2.x z macierzy kompatybilności, powtarzamy krok 4
Po odtworzeniu z kopii logi sprzed dnia awarii są niedostępne	świadoma decyzja `Minimal` bez snapshotów OpenSearch	zgodne z opisanym wyżej kompromisem, nie błąd procedury: przy kolejnej awarii rozważamy pełny backup ze snapshotami

Powiązane: Instalacja Grayloga w Dockerze: Graylog, MongoDB i OpenSearch · Graylog: przeprowadzka z docker01 na log01 · Graylog: wejścia, strumienie, wyszukiwanie · cron: zadania cykliczne · systemd timery: OnCalendar zamiast crona

Źródła: go2docs.graylog.org, rozdział „Backup and Restore Best Practices" · go2docs.graylog.org, rozdział „Upgrading Graylog" (kolejność aktualizacji) · opensearch.org, dokumentacja „Snapshot and restore" · dokumentacja MongoDB, rozdział „mongodump"

Revision #23
Created 2026-09-28 16:57:57 UTC by Tomasz Konieczny
Updated 2026-10-26 14:15:51 UTC by Tomasz Konieczny