Graylog: przeprowadzka z docker01 na log01

Typ: procedura

Dotyczy: Graylog 7.1.9, MongoDB 8.0, OpenSearch 2.19.6 (wersje te same na obu hostach), Rocky Linux 9

Czas: ~2 godziny okna zmian plus przygotowanie

Graylog stoi dziś na `docker01` razem z Gitea, NetBoxem i MailCow, a docelowo ma mieć własny serwer `log01` (10.20.10.52). Ta strona planuje przeniesienie tak, żeby hosty nie musiały nic zmieniać u siebie: wysyłają logi na nazwę `logi.zielona.example`, a my przestawiamy tylko jeden rekord DNS.

ZAŁOŻENIA SCENARIUSZA

`docker01` i `log01` (10.20.10.52) stoją w VLAN 10 w jednej serwerowni, więc hosty z tej sieci wysyłają logi bez przechodzenia przez zaporę. Hosty z innych sieci (VLAN 20, 30 i 98) docierają do `log01` przez pojedynczy `fw01`, a magazyn jest w sieci centrali; wariant z tunelem do `rtr-mag01`: IPsec site-to-site: Pamiątkowo i Swarzędz między FortiGate a routerem innej firmy.

Skróty użyte na tej stronie (13)

Skrót	Co znaczy
CA	Certificate Authority: urząd certyfikacji, który podpisuje certyfikaty
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
GB	Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej
GELF	Graylog Extended Log Format: format logów Grayloga
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
S2D	Storage Spaces Direct: programowa macierz Windows Server z dysków wielu serwerów

Skrót	Co znaczy
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia
TTL	Time To Live: czas życia rekordu lub danych
UDP	User Datagram Protocol: szybki protokół sieciowy bez potwierdzeń
UUID	Universally Unique Identifier: długi, praktycznie niepowtarzalny identyfikator
VLAN	Virtual LAN: wydzielona logicznie sieć na tym samym sprzęcie
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Fiszka

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo (hosty linuksowe) albo w PowerShellu na `dc01` z uprawnieniami administratora DNS. Polecenia do OpenSearch wołamy z wnętrza kontenera `graylog-opensearch`, bo port `9200` nie jest wystawiony na hosta. W komendach stoją polskie placeholdery, w przykładach nazwy z Zielonej.

□□ TYLKO ODCZYT

Sprawdzenie rozmiaru indeksów: _cat/indices

```
docker exec graylog-opensearch curl -s
"localhost:9200/_cat/indices?v&s=store.size:desc&h=index,docs.count,store.size"
```

Pierwszy krok planowania: ile naprawdę zajmują poszczególne zestawy indeksów i ile przybywa na dobę.

Przykład:

```
index           docs.count store.size
ad-audit_3        18420511      9.8gb
fw_12             40118377     21.4gb
graylog_41        10211844      5.1gb
```

□□ **docker** = narzędzie do kontenerów

□□ TYLKO ODCZYT

Sprawdzenie wolnego miejsca: df i _cat/allocation

```
df -h /docker
docker exec graylog-opensearch curl -s "localhost:9200/_cat/allocation?v"
```

Przykład:

```
Filesystem                    Size  Used Avail Use% Mounted on
/dev/mapper/vgDocker-lvDocker 400G  352G   48G  89% /docker
```

df = disk free: ile miejsca wolnego na dyskach · **docker** = narzędzie do kontenerów

ZMIENIA SYSTEM

Obniżenie TTL rekordu DNS: Set-DnsServerResourceRecord

```
$stary = Get-DnsServerResourceRecord -ZoneName "NazwaStrefy" -Name "NazwaRekordu" -
RRType A
$nowy = $stary.Clone()
$nowy.TimeToLive = [TimeSpan]::FromSeconds(300)
Set-DnsServerResourceRecord -ZoneName "NazwaStrefy" -OldInputObject $stary -
NewInputObject $nowy
```

> (gdzie `NazwaStrefy` to strefa, w której stoi rekord, u nas `zielona.example`, a `NazwaRekordu` to `logi`)

Robimy to kilka dni przed oknem, żeby w dniu zmiany pamięć podręczna hostów trzymała stary adres najwyżej 5 minut. Zakładamy, że wewnętrzną strefę `zielona.example` obsługuje DNS na `dc01`. Jeśli rekordy stoją gdzie indziej, zmiana jest ta sama, tylko w innym miejscu.

TTL = Time To Live: czas życia rekordu lub danych · **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP

ZMIENIA SYSTEM

Zmiana adresu rekordu DNS: Set-DnsServerResourceRecord

```
$stary = Get-DnsServerResourceRecord -ZoneName "NazwaStrefy" -Name "NazwaRekordu" -
RRType A
```

```
$nowy = $stary.Clone()
$nowy.RecordData.IPv4Address = [System.Net.IPAddress]::Parse("NowyAdresIP")
Set-DnsServerResourceRecord -ZoneName "NazwaStrefy" -OldInputObject $stary -
NewInputObject $nowy
```

To jest moment przełączenia: od teraz hosty, które zapytają o nazwę, dostaną adres `log01`.

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP

TYLKO ODCZYT

Sprawdzenie, na co wskazuje nazwa: dig

```
dig +short NazwaFQDN
```

Przykład:

```
10.20.10.52
```

dig = domain information groper: odpytywanie DNS

OSTROŻNIE · NIEODWRACALNE

Utworzenie wolumenu na dane OpenSearch: vgcreate i lvcreate

```
sudo pvcreate /dev/sdX
sudo vgcreate vgOpenSearch /dev/sdX
sudo lvcreate -L 300G -n lvOpenSearch vgOpenSearch
sudo mkfs.xfs /dev/mapper/vgOpenSearch-lvOpenSearch
```

> (gdzie `/dev/sdX` to czwarty, jeszcze pusty dysk z wyniku `lsblk`)

Komendy można wykonywać całościowo jako blok tekstu. Wpis do `/etc/fstab` i montowanie opisujemy w kroku 2 przebiegu.

pvcreate = physical volume create: oddaj dysk pod LVM · **vgcreate** = volume group create: utwórz grupę wolumenów · **lvcreate** = logical volume create: utwórz wolumen logiczny · **mkfs.xfs** = załóż system plików XFS na wskazanym urządzeniu

⚠ OSTROŻNIE · NIEODWRACALNE

Sprawdzenie numerów użytkowników w obrazach: docker run --entrypoint id

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

Przykład (numery sprawdzamy sami, nie przepisujemy):

```
uid=1000(opensearch) gid=1000(opensearch) groups=1000(opensearch)
uid=1100(graylog) gid=1100(graylog) groups=1100(graylog)
```

docker = narzędzie do kontenerów

⚠ ZMIENIA SYSTEM

Nadanie właściciela katalogom danych: chown

```
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)

Kontener działa jako zwykły użytkownik, więc katalog należący do roota blokuje zapis i kontener nie wstaje.

chown = change owner: zmień właściciela

PLIK KONFIGURACYJNY

Włączenie repozytorium migawek w Compose: path.repo

Plik `/docker/compose/graylog/compose.yaml`, usługa `opensearch` (na `docker01` i na `log01`):

```
  opensearch:
    environment:
```

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

Fragment dopisujemy do dotychczasowych list `environment` i `volumes`, nie zastępujemy ich. Bez `path.repo` OpenSearch odrzuci rejestrację repozytorium, nawet gdy katalog jest zamontowany. Zmiana wymaga odtworzenia kontenera `graylog-opensearch`, więc na `docker01` robimy ją poza oknem zmian, w spokojnej chwili.

□□ TYLKO ODCZYT

Rejestracja repozytorium migawek: _snapshot

```
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}
```

Ta sama komenda, ta sama nazwa repozytorium `graylog_backup` jak na stronie Kopia zapasowa i aktualizacja Grayloga, wykonywana na obu hostach.

□□ **docker** = narzędzie do kontenerów

□□ TYLKO ODCZYT

Wykonanie migawki wybranego zestawu: _snapshot PUT

```
docker exec graylog-opensearch curl -s -X PUT
"localhost:9200/_snapshot/graylog_backup/NazwaMigawki?wait_for_completion=true" \
  -H 'Content-Type: application/json' \
  -d '{"indices": "NazwaPrefiksu_*", "include_global_state": false}'
```

> (gdzie `NazwaPrefiksu` to prefiks zestawu indeksów, u nas `ad-audit`)

Przykład:

```
{"snapshot":{"snapshot":"przenosiny-ad-
audit","state":"SUCCESS","shards":{"total":4,"failed":0,"successful":4}}}
```

docker = narzędzie do kontenerów

TYLKO ODCZYT

Skopiowanie migawek na log01: rsync

```
sudo rsync -a --numeric-ids --rsync-path="sudo rsync" /docker/apps/graylog/snapshots/
nazwa_usera@log01:/docker/apps/graylog/snapshots/
```

Zachowujemy numery właściciela, bo OpenSearch na `log01` musi móc te pliki czytać. Zdalne `sudo` bez terminala wymaga wpisu `NOPASSWD` dla `rsync` albo kopiowania na konto z prawem zapisu do tego katalogu.

NOPASSWD = wpis w sudoers pozwalający wykonać wskazane polecenie przez sudo bez podawania hasła · **rsync** = remote sync: kopiuje pliki i katalogi lokalnie albo przez SSH, przesyłając tylko to, co się zmieniło, tu migawki OpenSearch na nowy serwer

ZMIENIA SYSTEM

Zrzut konfiguracji Grayloga: mongodump do pliku na hoście

```
docker exec graylog-mongodb mongodump --archive --db graylog >
/docker/apps/graylog/kopie/graylog-mongo-przenosiny.archive
```

Tym razem archiwum idzie strumieniem na hosta, a nie do `/tmp` w kontenerze, więc nie znika razem z kontenerem. Plik kopiujemy na `log01` przez `scp` do `/docker/apps/graylog/kopie/`.

docker = narzędzie do kontenerów

TYLKO ODCZYT

Przywrócenie konfiguracji na log01: mongorestore

```
docker exec -i graylog-mongodb mongorestore --archive --drop <
/docker/apps/graylog/kopie/graylog-mongo-przenosiny.archive
```

Wykonujemy przy uruchomionym kontenerze `graylog-mongodb` i jeszcze nieuruchomionym Graylogu. Flaga `--drop` kasuje kolekcje przed wgraniem, więc nigdy nie używamy jej na bazie z danymi, które chcemy zachować.

docker = narzędzie do kontenerów

TYLKO ODCZYT

Przywrócenie migawki na log01: _restore

```
docker exec graylog-opensearch curl -s -X POST
"localhost:9200/_snapshot/graylog_backup/NazwaMigawki/_restore?wait_for_completion=true"
\
  -H 'Content-Type: application/json' \
  -d '{"indices": "NazwaPrefiksu_*", "include_aliases": true, "include_global_state":
false}'
```

Wykonujemy przed pierwszym startem Grayloga na `log01`: gdy Graylog wystartuje pierwszy, sam założy indeksy i alias o tych samych nazwach, a OpenSearch odmówi przywrócenia indeksu, który już istnieje.

docker = narzędzie do kontenerów

TYLKO ODCZYT

Porównanie liczby dokumentów: _cat/count

```
docker exec graylog-opensearch curl -s "localhost:9200/_cat/count/NazwaPrefiksu_*?v"
```

Wynik z `docker01` i z `log01` powinien być taki sam, o ile zatrzymaliśmy Graylog przed migawką.

docker = narzędzie do kontenerów

KLIKAMY W INTERFEJSIE

Przeliczenie zakresów indeksów: System → Indices

```
System → Indices → NazwaZestawu → Maintenance → Recalculate index ranges
```

Zakresy mówią Graylogowi, który indeks zawiera jakie daty. Po przywróceniu przeliczamy je dla każdego zestawu, żeby wyszukiwanie nie szukało w indeksach, których już nie ma.

TYLKO ODCZYT

Test wysyłki na adres IP nowego hosta: logger -n

```
logger -n NowyAdresIP -P 1514 -T -p local0.info "test przeprowadzki $(hostname)"
```

Wysyłamy jedną wiadomość na adres IP `log01` jeszcze przed zmianą DNS, z TCP (`-T`) i portem wejścia. Szukamy jej potem w Graylogu po treści.

IP = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection) · **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **TCP** = Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia · **logger** = Logger: zapisuje komunikat w dzienniku systemowym

TYLKO ODCZYT

Sprawdzenie portu wejścia: ncat

```
ncat -zv NazwaFQDN 1514
```

Przykład:

```
Ncat: Connected to 10.20.10.52:1514.
```

ncat = network cat: narzędzie z pakietu nmap-ncat do łączenia się z portem TCP albo UDP, tu do sprawdzenia, czy wejście Grayloga jest osiągalne

TYLKO ODCZYT

Zatrzymanie Grayloga na docker01: docker compose stop

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

Zatrzymujemy tylko kontener `graylog`: MongoDB i OpenSearch zostają włączone, bo z nich robimy kopie. Hosty z `rsyslog` buforują w kolejkach opisanych na stronie Wysyłka logów z hostów, a po powrocie odbiorcy dosyłają zaległe wiadomości.

docker = narzędzie do kontenerów

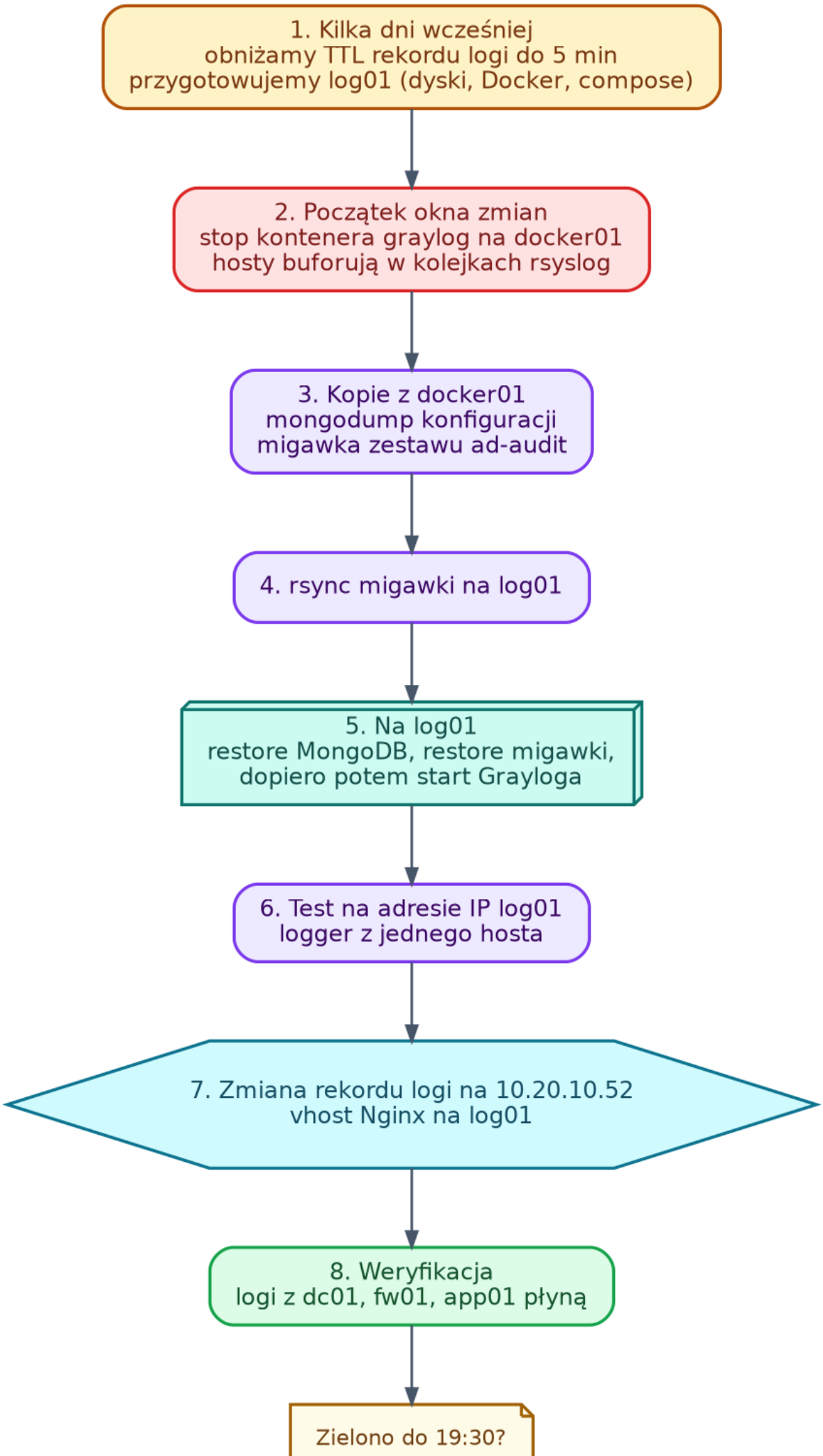
Zanim zaczniemy

Z ŻYCIA WZIĘTE

Wtorek, 8:20. Tomek zgłasza, że `/docker` na `docker01` ma 89 procent zajętości i rośnie o kilka gigabajtów na dobę. Winowajcą jest OpenSearch Grayloga, który z domyślną retencją zajął ponad 200 GB z 400. Pełny wolumen `vgDocker/lvDocker` nie zatrzymał systemu operacyjnego, ale Gitea odmówiła przyjęcia pusha Oli. To jest dokładnie ten powód, dla którego logi dostają własny serwer: ich apetyt nie powinien odbierać miejsca reszcie aplikacji.

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.

Plan opiera się na trzech osobnych sprawach, bo Graylog to trzy kontenery z trzema rodzajami danych. Konfigurację (MongoDB) przenosimy zawsze, jest mała i bez niej nowy Graylog nie wie, jakie ma wejścia, strumienie, konta i alerty. Logi (OpenSearch) przenosimy albo nie, o czym decyzja niżej. Adres, na który wysyłają hosty, zmieniamy rekordem DNS.


1. Kilka dni wcześniej
obniżamy TTL rekordu logi do 5 min
przygotowujemy log01 (dyski, Docker, compose)
2. Początek okna zmian
stop kontenera graylog na docker01
hosty buforują w kolejkach rsyslog
3. Kopie z docker01
mongodump konfiguracji
migawka zestawu ad-audit
4. rsync migawki na log01
5. Na log01
restore MongoDB, restore migawki,
dopiero potem start Grayloga
6. Test na adresie IP log01
logger z jednego hosta
7. Zmiana rekordu logi na 10.20.10.52
vhost Nginx na log01
8. Weryfikacja
logi z dc01, fw01, app01 płyną
Zielono do 19:30?

Kolejność kroków przeprowadzki: obniżenie TTL, przygotowanie log01, zatrzymanie Grayloga na docker01, kopie, przywrócenie na log01, test na adresie IP, zmiana DNS, weryfikacja i decyzja o wycofaniu (kliknięcie otwiera pełny rozmiar)

Decyzja: przenosimy logi czy zaczynamy z czystym indeksem

Wariant	Co robimy	Zaleta	Wada
A. Migawka	`_snapshot` wybranych zestawów, `rsync`, `_restore` na `log01`	historia jest od razu na nowym hoście	wymaga `path.repo` na obu hostach, miejsca na kopię i dłuższego okna
B. Czysty indeks	nowy OpenSearch zaczyna od zera, stary Graylog na `docker01` zostaje do odczytu, aż wygaśnie retencja	najprostsza procedura, brak ryzyka błędu przy przywracaniu	dwa miejsca do przeszukiwania przez czas retencji, `docker01` trzyma zajęte miejsce
C. Kopia katalogu	stop wszystkich kontenerów, `rsync` całego `/docker/apps/graylog`	zachowuje wszystko razem z konfiguracją	okno zależne od rozmiaru danych, wymaga tych samych wersji i tej samej architektury

W Zielonej proponujemy wariant mieszany. Zestaw `ad-audit` z retencją 180 dni przenosimy migawką (wariant A), bo to dowód dla audytora i nie czekamy pół roku, aż wygaśnie na `docker01`. Zestawy `fw` i `app` z retencją 30 dni zaczynają od czystego indeksu (wariant B), a stary Graylog na `docker01` pozostaje do odczytu przez 30 dni. Domyślny zestaw i zestawy zdarzeń (historia alertów) nie jadą na `log01`, co jest świadomą stratą.

Ostateczna decyzja należy do autora: jeśli pan Wojtek wymaga ciągłej historii zapory, dokładamy `fw` do migawki.

Rozmiar dysków na log01

Wolumeny według konwencji `vg<Przeznaczenie>` i `lv<Przeznaczenie>`, każde przeznaczenie na osobnym dysku wirtualnym. Wolumen na dane OpenSearch jest osobny celowo: pełny dysk indeksów nie może zatrzymać Dockera, MongoDB ani dziennika Grayloga, bo to one pozwalają mu przeżyć awarię indeksu, o czym piszemy w Graylog: częste problemy.

Przeznaczenie	Grupa	Wolumen	Rozmiar LV	Dysk wirtualny	Montowanie
system	`vgOS`	`lvOS`	40 GB	60 GB	`/`
`/var`	`vgVAR`	`lvVAR`	30 GB	50 GB	`/var`

Przeznaczenie	Grupa	Wolumen	Rozmiar LV	Dysk wirtualny	Montowanie
Docker, Compose, MongoDB, dziennik Grayloga, migawki na czas przeprowadzki	`vgDocker`	`lvDocker`	150 GB	200 GB	`/docker`
indeksy OpenSearch	`vgOpenSearch`	`lvOpenSearch`	300 GB	400 GB	`/docker/apps/graylog/opensearch`

Nie zajmujemy całych grup (nie używamy `100%FREE`): wolne miejsce w grupie to zapas na `lvextend`. Rozmiar `lvOpenSearch` wynika z rachunku, w którym liczby dobowe są założeniami Zielonej, do zmierzenia na `docker01` komendą `_cat/indices` z Fiszki.

Zestaw indeksów	Przyrost na dobę (założenie)	Retencja	Miejsce
`ad-audit`	0,4 GB	180 dni	72 GB
`fw`	3 GB	30 dni	90 GB
`app`	0,8 GB	30 dni	24 GB
domyślny	0,5 GB	14 dni	7 GB
Razem			193 GB

OpenSearch domyślnie przestaje przydzielać nowe shardy przy 85 procentach zajętości dysku, a przy 95 oznacza indeksy jako tylko do odczytu. Dlatego wolumen liczymy tak, żeby normalna praca kończyła się przy około 65 procentach: 193 GB podzielone przez 0,65 to około 300 GB. Pamięć na `log01` proponujemy 16 GB i 4 vCPU, z heap OpenSearch 4 GB i heap Grayloga 2 GB, a resztę zostawiamy systemowi i MongoDB (zasada producenta: heap najwyżej połowa pamięci i nie więcej niż 31 GB).

> Dyski wirtualne leżą na klastrze Hyper-V z S2D, a przy lustrzanym układzie każdy gigabajt zajmuje w puli dwa razy więcej. Zamawiamy 710 GB wirtualnych, więc sprawdzamy wolne miejsce w puli, zanim poprosimy o dyski.

Co musi być gotowe przed oknem

Wymaganie	Uwaga
`log01` zainstalowany według Instalacja Rocky Linux 9 z własnym nazewnictwem VG i LV	cztery wolumeny z tabeli wyżej

Wymaganie	Uwaga
Docker z `data-root` w `/docker/lib`	Instalacja Dockera na Rocky Linux 9 z osobnym wolumenem /docker
`vm.max_map_count=262144` na `log01`	ten sam krok 1 co na stronie Instalacja Grayloga w Dockerze: Graylog, MongoDB i OpenSearch
Plik `.env` skopiowany z `docker01` bez zmian	`GRAYLOG_PASSWORD_SECRET` musi być ten sam, inaczej konfiguracja z MongoDB, w tym zaszyfrowane hasła, jest nieczytelna
Nginx na `log01` z certyfikatem `logi.zielona.example`	certyfikat i klucz kopiujemy z `docker01`, vhost opisujemy w kroku 6
Dostęp z `log01` do `dc01` na 636 i do `docker01` na 587	LDAPS do logowania, SMTP do powiadomień MailCow
Dostęp z hostów do `log01` na 1514 i 12201	ten sam VLAN serwerowy, z innych VLAN-ów sprawdzamy reguły na `fw01`
TTL rekordu `logi` obniżony do 300 sekund	kilka dni przed oknem

Okno zmian proponujemy na środę 14 października 2026, od 18:00 do 20:00: po Patch Tuesday z 13 października, przed zamknięciem miesiąca (28 do 30 października) i dalekie od zakazu zmian od 15 listopada. Środa wieczorem, nie piątek po 16:00. Termin zatwierdza autor.

Przebieg

1. Kilka dni wcześniej: przygotowanie log01

Zakładamy wolumeny według tabeli, instalujemy Docker i przenosimy pliki konfiguracyjne. Na `log01` tworzymy katalogi i nadajemy właściciela komendami z Fiszki:

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

Katalog `/docker/apps/graylog/opensearch` powstaje w następnym kroku, jako punkt montowania własnego wolumenu.

2. Montowanie wolumenu OpenSearch

```
sudo mkdir -p /docker/apps/graylog/opensearch
sudo blkid /dev/mapper/vgOpenSearch-lvOpenSearch
```

W `/etc/fstab` dopisujemy wiersz (identyfikator UUID z wyniku `blkid`):

```
UUID=IdentyfikatorUUID  /docker/apps/graylog/opensearch  xfs  defaults  0 0
```

```
sudo mount -a
df -h /docker/apps/graylog/opensearch
```

Przykład:

```
Filesystem                              Size  Used Avail Use% Mounted on
/dev/mapper/vgOpenSearch-lvOpenSearch 300G  2.1G  298G   1% /docker/apps/graylog/opensearch
```

Dopiero teraz nadajemy właściciela katalogom komendą `chown` z Fiszki.

3. Compose na log01: skopiowany z docker01 z trzema zmianami

Kopiujemy `/docker/compose/graylog/compose.yaml` i `.env` z `docker01`, a w pliku na `log01` zmieniamy:

Co	Na `docker01`	Na `log01`
Heap OpenSearch	`OPENSEARCH_JAVA_OPTS=-Xms1g -Xmx1g`	`OPENSEARCH_JAVA_OPTS=-Xms4g -Xmx4g`
Heap Grayloga	brak zmiennej	`GRAYLOG_SERVER_JAVA_OPTS=-Xms2g -Xmx2g -server -XX:+UseG1GC -XX:-OmitStackTraceInFastThrow`
Repozytorium migawek	brak	`path.repo` i wolumen `snapshots` z Fiszki

Port `9000` możemy dodatkowo zawęzić do pętli zwrotnej (`"127.0.0.1:9000:9000/tcp"`), bo Nginx stoi teraz na tym samym hoście i nie potrzebuje adresu sieciowego. Zmienne `GRAYLOG_TRANSPORT_EMAIL_*` z Graylog: codzienna praca z logami zostają w pliku bez zmian.

W próbie uruchamiamy tylko dwie usługi, żeby Graylog nie założył pustych indeksów:

```
cd /docker/compose/graylog
docker compose up -d mongodb opensearch
docker exec graylog-opensearch curl -s "localhost:9200/_cluster/health?pretty"
```

Przykład:

```
{
  "cluster_name" : "docker-cluster",
  "status" : "green",
  "number_of_nodes" : 1
}
```

4. Kilka dni wcześniej, na docker01: włączenie migawek

Dopisujemy `path.repo` i wolumen `snapshots` do `compose.yaml` na `docker01` i odtwarzamy kontener OpenSearch (`docker compose up -d opensearch`). Graylog w tym czasie buforuje wiadomości w dzienniku, więc robimy to w spokojnej chwili, nie w oknie. Potem rejestrujemy repozytorium `graylog_backup` komendą z Fiszki na obu hostach.

5. Okno zmian: zatrzymanie, kopie, przywrócenie

Kolejność w oknie ma znaczenie, bo przywracanie migawki musi poprzedzić pierwszy start Grayloga na `log01`.

1. Na `docker01` zatrzymujemy kontener `graylog` (Fiszka). Hosty zaczynają buforować, FortiGate wysyłający przez UDP nie ma bufora, więc logi zapory z okna przepadną. Zapisujemy godzinę przerwy.
2. Na `docker01` robimy zrzut `mongodump` i migawkę zestawu `ad-audit` (Fiszka), kopiujemy jedno i drugie na `log01`.
3. Na `log01` restore konfiguracji `mongorestore`, potem rejestracja repozytorium i `_restore` migawki.
4. Na `log01` uruchamiamy Graylog: `docker compose up -d graylog`.
5. Po starcie powtarzamy krok z `keytool` ze strony Graylog: logowanie kontem z Active Directory, bo certyfikat CA dodany do magazynu `cacerts` w starym kontenerze nie przeszedł na nowy host.
6. W System → Indices przeliczamy zakresy indeksów dla każdego zestawu.
7. Testujemy wejście na adresie IP `log01`: `logger -n 10.20.10.52 -P 1514 -T ...` (Fiszka) i szukamy wiadomości w Graylogu.

6. Nginx na log01 i zmiana DNS

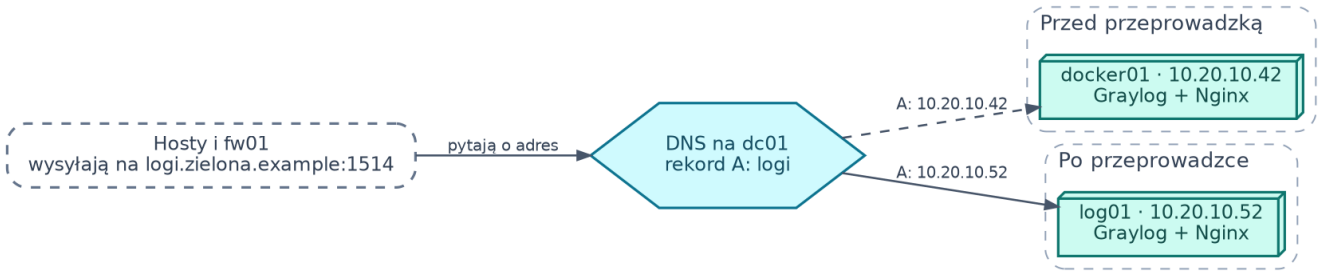
Vhost na `log01` jest kopią vhosta z `docker01`, z jedną zmianą: kontener Grayloga jest teraz na tym samym hoście.

```
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://127.0.0.1:9000;
        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";
    }
}
```

Zmieniamy rekord `logi` na `10.20.10.52` poleceniem z Fiszki i usuwamy vhost `logi.zielona.example` z Nginksa na `docker01`. Jedna nazwa służy tu dwóm celom naraz, stroną WWW i adresem wejść, więc musi wskazywać serwer, na którym stoją oba. Dlatego Nginx przechodzi razem z Graylogiem.



Hosty i fw01 pytają DNS na dc01 o rekord logi, który przed przeprowadzką wskazuje docker01, a po niej log01, bez żadnej zmiany na hostach (kliknięcie otwiera pełny rozmiar)

7. Po oknie: stary Graylog tylko do odczytu

Na `docker01` uruchamiamy Graylog ponownie (`docker compose up -d graylog`). Hosty już tam nie piszą, bo nazwa wskazuje `log01`, więc to jest przeglądarka starych logów zapory i aplikacji, dostępna pod `http://10.20.10.42:9000` z sieci administratorów. Retencja usuwa stare indeksy sama. Po 30 dniach zatrzymujemy go komendą `docker compose down` i zostawiamy katalog `/docker/apps/graylog` jeszcze na miesiąc jako zabezpieczenie, a potem usuwamy po uzgodnieniu z autorem.

Weryfikacja

Sprawdzamy, że nazwa wskazuje nowy host, z hosta linuksowego i z `dc01`:

```
dig +short logi.zielona.example
```

Przykład:

```
10.20.10.52
```

Na `log01` widzimy ruch z hostów, zanim zajrzymy do interfejsu:

```
sudo tcpdump -n -i any port 1514
```

W Graylogu na `log01` sprawdzamy po kolei:

1. System → Inputs: wejścia w stanie Running, z rosnącą liczbą wiadomości na sekundę.
2. Streams: trzy strumienie (`ad-logowania`, `fw-zapora`, `app01-aplikacje`) w stanie Running.
3. Search z zakresem 30 dni na strumieniu `ad-logowania`: widać logowania sprzed przeprowadzki, a `_cat/count` zgadza się z liczbą z `docker01`.
4. Alerts → Notifications → Execute Test Notification: mail dochodzi na `it@zielona.example`.
5. Logowanie kontem domenowym (`ola.wisniewska`) działa.
6. Logi z `dc01`, `fw01` i `app01` mają znaczniki czasu po godzinie przełączenia.

□□ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: System → Inputs na `log01` po przeprowadzce, z wejściami syslog i GELF w stanie Running i licznikiem wiadomości na sekundę.

Wycofanie zmian

Warunek decyzyjny ustalamy z góry: jeżeli o 19:30 nie ma logów z `dc01`, `fw01` i `app01` w Graylogu na `log01`, wracamy. Wycofanie jest tanie, bo na `docker01` niczego nie usunęliśmy, tylko

zatrzymaliśmy kontener.

1. Zmieniamy rekord `logi` z powrotem na `10.20.10.42` (komenda z Fiszki z adresem `docker01`). Przy TTL 300 sekund hosty wracają w ciągu kilku minut.
2. Na `docker01` uruchamiamy Graylog: `docker compose up -d graylog`, i przywracamy vhost `logi.zielona.example`.
3. Wiadomości, które w międzyczasie trafiły na `log01`, zostają tam. Nie scalamy ich ze starymi, tylko zapisujemy w notatce z okna lukę czasową.
4. Przyczynę błędu rozpracowujemy na `log01` bez presji czasu i wyznaczamy nowe okno.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
Kontener `graylog-opensearch` na `log01` restartuje się, w logu `AccessDeniedException`	katalog danych należy do roota, a nie do użytkownika z obrazu	`chown -R` na numery z `docker run --entrypoint id`, `docker compose up -d`
Rejestracja repozytorium zwraca `repository_exception` albo „location is not registered”	brak `path.repo` w środowisku kontenera albo kontener nie został odtworzony	dopisujemy `path.repo`, odtwarzamy kontener, powtarzamy rejestrację
`_restore` zgłasza, że indeks już istnieje	Graylog wystartował przed przywróceniem i założył własne indeksy o tych nazwach	zatrzymujemy Graylog, zamykamy albo usuwamy puste indeksy tego zestawu, powtarzamy `_restore`, dopiero potem startujemy Graylog
Graylog na `log01` startuje, ale nikt się nie loguje kontem domenowym	certyfikat CA nie jest w `cacerts` nowego kontenera	powtarzamy import `keytool` i `docker restart graylog`
Graylog po starcie prosi o konfigurację od zera albo nie przyjmuje haseł z konfiguracji	`GRAYLOG_PASSWORD_SECRET` w `.env` na `log01` inny niż na `docker01`	wracamy do oryginalnego `.env`, `docker compose up -d`
Część hostów dalej pisze na stary adres	pamięć podręczna DNS albo trwałe połączenie TCP do starego hosta	po zatrzymaniu Grayloga na `docker01` połączenie i tak się zrywa, a `rsyslog` przy ponownym łączeniu pyta DNS jeszcze raz, jeśli nie, restartujemy `rsyslog` na tym hoście
Logi z `fw01` nie docierają po zmianie DNS	FortiGate nie odświeżył adresu nazwy serwera syslog	sprawdzamy ustawienia serwera syslog na `fw01` i w razie potrzeby zapisujemy nazwę ponownie albo wpisujemy adres IP
Nowe wyszukiwania zwracają błędy o brakujących indeksach	nieprzeliczone zakresy indeksów po przywróceniu	System → Indices → zestaw → Maintenance → Recalculate index ranges

Powiązane: Kopia zapasowa i aktualizacja Grayloga · Instalacja Grayloga w Dockerze: Graylog, MongoDB i OpenSearch · Graylog: codzienna praca z logami · Graylog: częste problemy · Instalacja

Rocky Linux 9 z własnym nazewnictwem VG i LV

Źródła: go2docs.graylog.org: „Compatibility Matrix" (Graylog 7.1: OpenSearch do 2.19.6, MongoDB do 8.2), „Additional Configuration" (rozmiar heap) · docs.opensearch.org, „Snapshot and restore" (`path.repo`, `include_aliases`, konflikt istniejącego indeksu) · progi dysku OpenSearch 85, 90 i 95 procent według dokumentacji OpenSearch · go2docs.graylog.org, „Backup and Restore Best Practices": treści strony nie udało się pobrać, więc kolejność przywracania (konfiguracja, potem migawka, potem pierwszy start Grayloga) i przeliczenie zakresów indeksów są do potwierdzenia na próbie (sprawdzone 03.10.2026)

Revision #15
Created 2026-10-03 12:08:23 UTC by Tomasz Konieczny
Updated 2026-10-26 14:15:52 UTC by Tomasz Konieczny