Kopia zapasowa aplikacji dockerowych

Typ: procedura **Dotyczy: wszystkie aplikacje na docker01** **Czas: ~40 min**

Kopia zapasowa aplikacji dockerowej to zawsze dwie osobne rzeczy: pliki spod `/docker/apps` i `/docker/compose` oraz osobny zrzut bazy danych, jeśli aplikacja jej używa, zamiast kopiowania plików bazy w trakcie jej działania. Ta strona opisuje wspólny mechanizm dla wszystkich aplikacji na `docker01`, nie backup jednej konkretnej z nich.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Zanim zaczniemy

Z ŻYCIA WZIĘTE

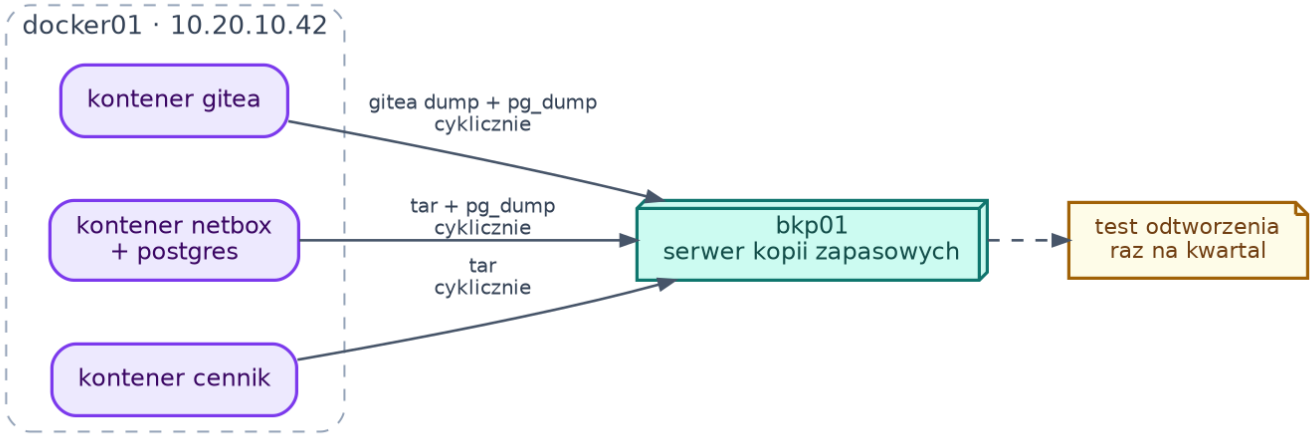
W czwartek Ola prosi o pilne odtworzenie NetBoksa na testowym hoście po nieudanej migracji danych. Zespół sięga po najświeższą kopię z `bkp01`, ale nikt wcześniej nie sprawdzał, czy zrzut bazy faktycznie się odtwarza. Test na czystym hoście kończy się sukcesem dopiero za trzecim razem, po poprawieniu uprawnień do pliku `.env` skopiowanego osobno od `compose.yaml`.

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.

Wymaganie	Uwaga
Dostęp SSH z `docker01` na `bkp01` kluczem, bez hasła	konto dedykowane kopiom, nie konto administratora

Wymaganie	Uwaga
Miejsce na `bkp01` pod kopie z `docker01`	patrz Świat przykładów: Zielona po adres i rolę serwera
Znajomość, które aplikacje trzymają dane w zewnętrznej bazie, a które w plikach	tabela w kroku 1

Uważaj: kopiowanie plików działającej bazy danych, na przykład katalogu `postgres` bind mountowanego pod kontener, w trakcie jej pracy daje niespójny obraz, bo baza w tym samym czasie zapisuje zmiany. Zamiast tego zawsze wykonujemy logiczny zrzut narzędziem samej bazy (`pg_dump` , `mysqldump`) i kopiujemy dopiero gotowy plik zrzutu.



Skrypt na docker01 kopiuje pliki spod /docker/apps i zrzuty baz danych na bkp01, cyklicznie i z okresowym testem odtworzenia (kliknięcie otwiera pełny rozmiar)

Przebieg

1. Rozpoznanie, co kopiujemy dla każdej aplikacji

Aplikacja	Dane w plikach	Baza danych	Co zrzucamy
`gitea`	repozytoria, załączniki	PostgreSQL na `pg01`	`gitea dump` plus `pg_dump`
`netbox`	pliki multimedialne	PostgreSQL w kontenerze na `docker01`	`pg_dump` z kontenera bazy

Aplikacja	Dane w plikach	Baza danych	Co zrzucamy
`cennik`	cały katalog danych	brak, tylko pliki	sam katalog, bez zrzutu bazy

Rozpoznanie robimy raz, przy wdrożeniu każdej aplikacji, i zapisujemy w skrypcie kopii, nie sprawdzamy tego ręcznie przy każdym uruchomieniu.

2. Skrypt kopii dla aplikacji z bazą PostgreSQL w kontenerze

```
sudo vim /docker/skrypty/kopia-netbox.sh
```

```
#!/usr/bin/env bash
set -euo pipefail

# Kopia zapasowa NetBoksa: pliki aplikacji plus zrzut bazy z kontenera postgres.
DATA=2026-09-27
CEL=nazwa_usera@bkp01:/backup/netbox/

# Zrzut bazy z kontenera, bez zatrzymywania samej bazy.
docker exec netbox-postgres pg_dump -U netbox netbox > /tmp/netbox_${DATA}.sql

# Archiwum plików aplikacji (media, załączniki), bez katalogu bazy.
tar czf /tmp/netbox-apps_${DATA}.tar.gz -C /docker/apps/netbox --exclude=postgres .

# Wysyłka obu plików na bkp01.
scp /tmp/netbox_${DATA}.sql /tmp/netbox-apps_${DATA}.tar.gz "${CEL}"

# Sprzątanie plików tymczasowych na docker01.
rm -f /tmp/netbox_${DATA}.sql /tmp/netbox-apps_${DATA}.tar.gz
```

```
sudo chmod +x /docker/skrypty/kopia-netbox.sh
```

`set -euo pipefail` na początku skryptu zatrzymuje go od razu przy pierwszym błędzie, zamiast kontynuować z niepełną kopią i cichym wpisem w logu: brak bazy, pełny dysk na `bkp01` albo zerwane SSH przerywa skrypt, zamiast wysyłać częściowe lub puste archiwum.

3. Skrypt kopii dla aplikacji bez bazy danych

```
sudo vim /docker/skrypty/kopia-cennik.sh
```

```
#!/usr/bin/env bash
set -euo pipefail

DATA=2026-09-27
CEL=nazwa_usera@bkp01:/backup/cennik/

tar czf /tmp/cennik_${DATA}.tar.gz -C /docker/apps/cennik .
scp /tmp/cennik_${DATA}.tar.gz "${CEL}"
rm -f /tmp/cennik_${DATA}.tar.gz
```

Bez bazy danych skrypt jest krótszy o cały krok zrzutu: samo archiwum katalogu `/docker/apps/cennik` wystarcza, bo to jedyne miejsce, gdzie ta aplikacja trzyma cokolwiek trwałego.

4. Timer systemd zamiast crona

Wybieramy systemd timery: OnCalendar zamiast crona zamiast wpisu w `crontab`, żeby mieć w jednym miejscu log wykonania i status ostatniego uruchomienia.

```
sudo vim /etc/systemd/system/kopia-docker.service
```

```
[Unit]
Description=Kopia zapasowa aplikacji dockerowych na docker01

[Service]
Type=oneshot
ExecStart=/docker/skrypty/kopia-netbox.sh
ExecStart=/docker/skrypty/kopia-cennik.sh
```

```
sudo vim /etc/systemd/system/kopia-docker.timer
```

```
[Unit]
Description=Harmonogram kopii zapasowej aplikacji dockerowych
```

```
[Timer]
OnCalendar=*-*-* 02:30:00
Persistent=true

[Install]
WantedBy=timers.target
```

```
sudo systemctl enable --now kopia-docker.timer
```

5. Weryfikacja ostatniego uruchomienia

```
systemctl status kopia-docker.timer
```

Przykład:

```
● kopia-docker.timer - Harmonogram kopii zapasowej aplikacji dockerowych
     Loaded: loaded (/etc/systemd/system/kopia-docker.timer; enabled)
     Active: active (waiting) since Sun 2026-09-27 02:30:12 UTC
    Trigger: Mon 2026-09-28 02:30:00 UTC
```

```
journalctl -u kopia-docker.service --since today
```

6. Test odtworzenia na czystym hoście

Raz na kwartał odtwarzamy najświeższą kopię jednej aplikacji na osobnym, testowym hoście z tą samą wersją Dockera, niezależnie od `docker01`. Kolejność jest zawsze taka sama: rozpakowanie archiwum plikowego do `/docker/apps/<nazwa>`, odtworzenie zrzutu bazy do pustej bazy (`psql` albo `pg_restore`, zależnie od formatu zrzutu), skopiowanie `compose.yaml` i `.env` z `/docker/compose/<nazwa>`, `docker compose up -d`, sprawdzenie w przeglądarce, że aplikacja wygląda tak jak na `docker01`. Kopia, której nikt nigdy nie spróbował odtworzyć na osobnym hoście, jest tylko przypuszczeniem, że działa.

Weryfikacja

```
ssh nazwa_usera@bkp01 "ls -la /backup/netbox/ /backup/cennik/"
```

Przykład:

```
-rw-r--r--. 1 nazwa_usera nazwa_usera 4823112 wrz 27 02:30 netbox_2026-09-27.sql
-rw-r--r--. 1 nazwa_usera nazwa_usera 118293440 wrz 27 02:31 netbox-apps_2026-09-27.tar.gz
-rw-r--r--. 1 nazwa_usera nazwa_usera 2048512 wrz 27 02:30 cennik_2026-09-27.tar.gz
```

Wycofanie zmian

Wyłączenie harmonogramu bez usuwania już wykonanych kopii:

```
sudo systemctl disable --now kopia-docker.timer
```

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
Skrypt kończy się natychmiast bez żadnego pliku na `bkp01`	`set -euo pipefail` zatrzymał go na pierwszym błędzie	`journalctl -u kopia-docker.service`, sprawdzamy dokładny komunikat i krok, w którym padło
`pg_dump` z kontenera zwraca `role does not exist`	nazwa użytkownika w skrypcie inna niż w kontenerze bazy	sprawdzamy zmienne `POSTGRES_USER` w `compose.yaml` aplikacji
Archiwum `.tar.gz` dużo mniejsze niż spodziewany rozmiar katalogu	katalog bazy nieodfiltrowany przez `--exclude`, ale akurat pusty w danym momencie	`du -sh` na katalogu źródłowym przed archiwizacją, porównanie z rozmiarem archiwum
Test odtworzenia na czystym hoście nie startuje kontenera	inna wersja Dockera albo brakujący plik `.env` skopiowany osobno od `compose.yaml`	kopiujemy `.env` razem z `compose.yaml`, sprawdzamy `docker version` na obu hostach

Powiązane: Jak trzymamy aplikacje w Dockerze: konwencje Zielonej · Utrzymanie hosta dockerowego · systemd timery: OnCalendar zamiast crona · pg_dump, pg_dumpall, pg_restore

Źródła: man 1 systemd.timer · dokumentacja PostgreSQL, rozdział „pg_dump"

Revision #31
Created 2026-09-27 11:19:59 UTC by Tomasz Konieczny
Updated 2026-10-26 14:09:29 UTC by Tomasz Konieczny