Kopia zapasowa i aktualizacja Gitea

Poziom: operacyjny

Dotyczy: Gitea 1.27 w Dockerze, PostgreSQL 16 na pg01, Gitea Runner 1.0.0

Czas: ~40 min

Kopia zapasowa Gitei to dwie osobne rzeczy naraz: zrzut `gitea dump` z kontenera i osobny zrzut bazy danych z pg01, bo Gitea w naszym wdrożeniu trzyma metadane w zewnętrznym PostgreSQL, nie we wbudowanym SQLite. Aktualizacja to podmiana tagu obrazu, migracja bazy dzieje się sama, ale bez świeżej kopii sprzed niej wycofanie zmian jest trudne albo niemożliwe.

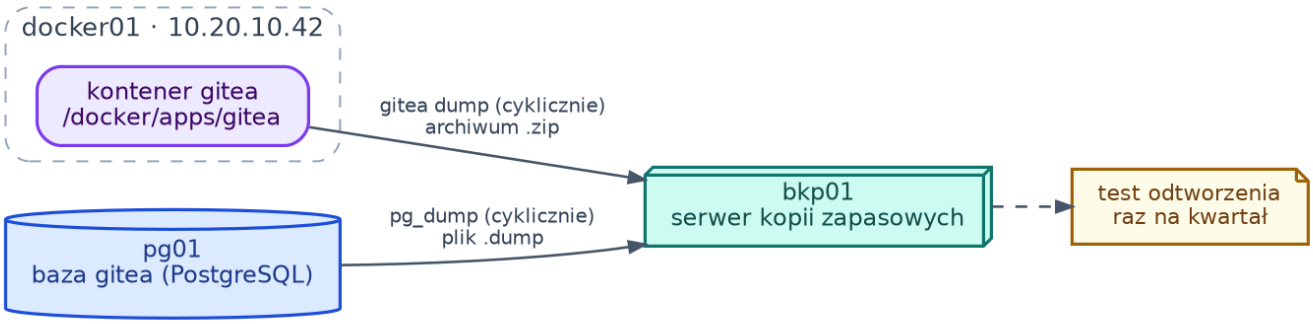
Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SQLite	mała baza danych zapisana w jednym pliku; Home Assistant trzyma w niej historię w pliku home-assistant_v2.db

Zanim zaczniemy

Wymaganie	Uwaga
Dostęp do `docker exec` na kontenerze `gitea` na docker01	konto z prawem uruchamiać kontenery Dockera
Dostęp do `pg_dump` na pg01 albo zdalnie do niego	konto bazy z prawem odczytu całej bazy `gitea`
Miejsce na `bkp01` na regularne kopie	patrz Świat przykładów: Zielona po adres i rolę serwera
Znajomość obecnego tagu obrazu Gitei	`docker compose images` w `/docker/compose/gitea/`

Cała droga danych, od kontenera do serwera kopii:



Kontener gitea i baza na pg01 wysyłają kopie do bkp01, cyklicznie i z okresowym testem odtworzenia (kliknięcie otwiera pełny rozmiar)

Uważaj: aktualizacja Gitei migruje schemat bazy danych przy pierwszym starcie nowej wersji. Migracja jest jednokierunkowa: cofnięcie samego tagu obrazu do starej wersji po tym, jak migracja już przeszła, zostawia starą Giteę wobec nowego schematu bazy, którego nie rozumie. Jedyna pewna droga wycofania po migracji to odtworzenie z kopii sprzed aktualizacji, nie sama podmiana tagu.

Przebieg

1. Wykonanie kopii wewnątrz kontenera: gitea dump

```
docker exec -u git gitea gitea dump -c /data/gitea/conf/app.ini -w /tmp
```

Przykład:

```
2026/09/27 03:00:02 ...s/setting/setting.go:320:NewContext() [I] Custom config:
/data/gitea/conf/app.ini
Dumping local repositories...git-repos.zip
Dumping database...
Packing dump files...
Removing tmp work dir: /tmp/gitea-dump-1758943202
Finish dumping in file /tmp/gitea-dump-1758943202.zip
```

> **Pojęcie: gitea dump.** Wbudowane narzędzie pakujące w jeden plik zip repozytoria, konfigurację, dane załączników i (jeśli używamy wbudowanego SQLite) samą bazę. Przy zewnętrznej bazie, tak jak u nas, dump obejmuje repozytoria i konfigurację, a bazę zabezpieczamy osobno krokiem 3, bo `gitea dump` nie ma uprawnień administratora PostgreSQL, żeby zrobić to za nas.

2. Skopiowanie zrzutu z kontenera na hosta i na bkp01

```
docker cp gitea:/tmp/gitea-dump-1758943202.zip /docker/apps/gitea/kopie/
```

```
scp /docker/apps/gitea/kopie/gitea-dump-1758943202.zip nazwa_usera@bkp01:/backup/gitea/
```

Nazwę pliku wklejamy dokładnie taką, jaką pokazał krok 1 w ostatniej linii `Finish dumping in file`: znacznik czasu w nazwie jest unikalny dla każdego uruchomienia.

3. Kopia bazy danych z pg01: pg_dump

```
pg_dump -h pg01 -U nazwa_usera -Fc -f /backup/gitea/gitea_baza_2026-09-27.dump gitea
```

Format `custom` (`-Fc`) pozwala odtworzyć bazę `pg_restore`m niezależnie od tego, ile miejsca zajmuje pełny zrzut tekstowy, i pozwala przy odtwarzaniu wybrać tylko część obiektów, jeśli kiedyś to wystarczy. Więcej o formatach zrzutu na stronie pg_dump, pg_dumpall, pg_restore.

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. Przykładowy wpis crona na koncie z dostępem do `docker exec` i `pg_dump`:

```
crontab -e
```

Przykład:

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

5. Miejsce przechowywania kopii

Kopie z kroków 1 do 3 trafiają na `bkp01`, do katalogu `/backup/gitea/`, 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`, nie osobno dla Gitei.

6. Test odtworzenia

Raz na kwartał odtwarzamy najnowszą parę zrzutów (`gitea dump` plus `pg_dump`) na osobnej, testowej instalacji tej samej wersji Gitei: rozpakowanie archiwum z `gitea dump`, `pg_restore` zrzutu bazy do pustej bazy testowej, wskazanie tej bazy w `app.ini` testowej instancji, pierwszy start. Kopia, której nikt nigdy nie spróbował odtworzyć, jest tylko przypuszczeniem, że działa.

Aktualizacja

1. Przeczytanie informacji o wydaniu

Przed zmianą tagu sprawdzamy release notes na `docs.gitea.com` (sekcja Release Notes) pod kątem zmian wymagających ręcznej ingerencji: zmian w `app.ini`, usuniętych ustawień, ostrzeżeń o migracji, która potrwa długo na dużej instancji.

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. Zmiana tagu obrazu

`/docker/compose/gitea/.env`:

```
TAG_GITEA=1.28.0
```

```
docker compose -f /docker/compose/gitea/compose.yaml pull
docker compose -f /docker/compose/gitea/compose.yaml up -d
```

Przykład:

```
[+] Running 2/2
 ✔ Container gitea      Started
 ✔ Container gitea-db   Running
```

Migracja schematu bazy wykonuje się sama przy starcie nowego obrazu, w tym samym momencie co pierwsze uruchomienie po podmianie: nie ma osobnej komendy migracji do wywołania ręcznie.

4. Weryfikacja po aktualizacji

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

Sprawdzamy w logu potwierdzenie zakończonej migracji i brak błędów przy starcie, a w stopce strony logowania w przeglądarce nowy numer wersji.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: stopka dowolnej strony Gitei po zalogowaniu, z widocznym numerem wersji zgodnym z nowym tagiem obrazu (na przykład „Gitea Version: 1.28.0").

5. Aktualizacja runnera

Runner aktualizujemy osobno, niezależnie od samej Gitei, bo to osobny obraz i osobny plik Compose:

```
sed -i 's/gitea\/runner:.*/gitea\/runner:1.0.1/' /docker/compose/gitea-runner/compose.yaml
docker compose -f /docker/compose/gitea-runner/compose.yaml pull
docker compose -f /docker/compose/gitea-runner/compose.yaml up -d
```

Plik `.runner` w `/docker/apps/gitea-runner` przeżywa aktualizację (leży na trwałym wolumenie), więc runner nie rejestruje się ponownie, tylko wraca do pracy pod starym wpisem.

Weryfikacja

```
docker compose -f /docker/compose/gitea/compose.yaml ps
docker compose -f /docker/compose/gitea-runner/compose.yaml ps
```

Przykład:

```
NAME    IMAGE                   STATUS
gitea   gitea/gitea:1.28.0      Up 5 minutes (healthy)
```

```
curl -s -o /dev/null -w "%{http_code}\n" https://git.zielona.example/api/v1/version
```

Przykład:

```
200
```

Wycofanie zmian

Przed migracją (kopia z kroku 2 aktualizacji jeszcze nie była potrzebna): przerywamy, wracamy do poprzedniego tagu w `.env`, `docker compose up -d` uruchamia starą wersję na niezmienionej jeszcze bazie.

Po migracji, gdy nowa wersja działa niepoprawnie: samo cofnięcie tagu obrazu nie wystarcza, bo baza ma już nowy schemat, którego stara wersja Gitei nie rozumie, i próba jej uruchomienia kończy się błędem albo, gorzej, uszkodzeniem danych. Odtwarzamy zamiast tego kopię z kroku 2 aktualizacji: bazę przez `pg_restore` do bazy `gitea` na pg01, repozytoria i konfigurację z archiwum `gitea dump`, dopiero potem uruchamiamy Gitea ze starym tagiem obrazu wskazanym w `.env`.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
`gitea dump` kończy się błędem braku miejsca	katalog tymczasowy (`-w`) na tym samym, zapchanym wolumenie co dane aplikacji	wskazujemy `-w` na katalog z wolnym miejscem, na przykład `/tmp` na osobnym punkcie montowania
`pg_dump` pyta o hasło mimo poprawnego użytkownika	brak wpisu w `.pgpass` albo `PGPASSWORD` na koncie uruchamiającym harmonogram	uzupełniamy `~/.pgpass` dla konta z crona, zgodnie z regułami uwierzytelniania opisanymi na stronie Uwierzytelnianie i bezpieczeństwo połączeń
Po aktualizacji kontener `gitea` bez końca się restartuje	migracja przerwana w połowie (np. restart hosta w trakcie)	odtwarzamy z kopii sprzed aktualizacji, nie próbujemy uruchamiać ponownie tego samego obrazu na uszkodzonej w połowie bazie

Objaw	Przyczyna	Co robimy
Runner po aktualizacji obrazu rejestruje się jako nowy wpis zamiast wrócić do starego	katalog `/data` z plikiem `.runner` nie był zamontowany jako trwały wolumen	dokładamy wolumen `/docker/apps/gitea-runner:/data`, usuwamy zduplikowany wpis w Site Administration → Actions → Runners

Powiązane: Gitea Actions: runner i pierwszy workflow · Instalacja Gitea w Dockerze · pg_dump, pg_dumpall, pg_restore · cron: zadania cykliczne · systemd timery: OnCalendar zamiast crona

Źródła: docs.gitea.com, rozdział „Administration → Backup and Restore" · dokumentacja PostgreSQL, rozdział „pg_dump"

Revision #33
Created 2026-09-27 10:32:13 UTC by Tomasz Konieczny
Updated 2026-10-26 14:17:37 UTC by Tomasz Konieczny