Apache Solr w Dockerze: instalacja, rdzeń i pierwsze wyszukiwanie

Typ: procedura

Dotyczy: Apache Solr 10.0.0, obraz solr:10.0.0 (sprawdzono na Docker Hub i w dokumentacji projektu 3.10.2026)

Czas: ~30 min

Efekt: działający kontener Solr na `docker01`, z jednym rdzeniem `dokumenty`, kilkoma załadowanymi dokumentami i pierwszym udanym wyszukiwaniem. Utrzymanie zainstalowanego Solra (limity plików, hasła) opisujemy w Apache Solr: utrzymanie, tu nie dublujemy.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
CNET	CNET Networks: amerykańska firma medialna, w której w 2004 roku powstał Solr
HTTP	Hypertext Transfer Protocol: protokół stron WWW
HTTPS	HTTP zabezpieczony szyfrowaniem TLS
SolrCloud	tryb Solra jako klastra z wbudowanym albo zewnętrznym ZooKeeperem, w którym dane trzymamy w kolekcjach
ZooKeeper	Apache ZooKeeper: osobny program, który w klastrze trzyma wspólną konfigurację i pilnuje, kto jest kim

Zanim zaczniemy

Z ŻYCIA WZIĘTE

> Wtorek, 9:30. Marek ze Swarzędzu skarży się, że wyszukiwarka instrukcji magazynowych w aplikacji Oli szuka ponad dziesięć sekund, bo baza na `pg01` przegląda każdy wiersz tekstu po kolei. Ola prosi o prawdziwy silnik wyszukiwania. Tomek stawia Solra w kontenerze na `docker01`, zakłada rdzeń `dokumenty`, a po kilkunastu minutach ta sama fraza wraca w kilka milisekund. Dane zostają w bazie, Solr dostaje tylko ich kopię do szukania.

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo na maszynie z zainstalowanym Dockerem i wtyczką Compose. Nazwy, hasła i adresy podmieniamy na własne.

Skąd się wziął Solr

Doug Cutting napisał Lucene w 1999 roku, w Javie, jako bibliotekę do szybkiego przeszukiwania tekstu. W 2001 roku trafiła do Apache, a w 2005 stała się samodzielnym projektem. Nazwa Lucene to drugie imię żony Douga, które nosiła też jej babcia.

Lucene jest biblioteką, nie programem: żeby z niej skorzystać, trzeba napisać aplikację. W 2004 roku Yonik Seeley, pracujący w firmie CNET Networks, zrobił taką aplikację na własny użytek firmy, jako wyszukiwarkę dla serwisów CNET. Nazwał ją Solr i udostępnił Lucene przez zwykłe zapytania HTTP, więc korzystać z niej mógł każdy język programowania.

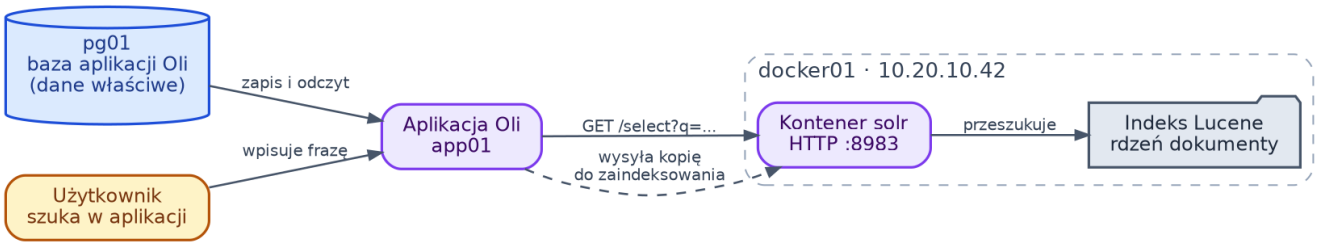
W styczniu 2006 roku CNET przekazał kod Fundacji Apache. W 2007 roku Solr dostał status samodzielnego projektu głównego, w 2010 roku zespoły Lucene i Solr połączyły siły, a w lutym 2021 roku Solr znów został osobnym projektem Apache. Wersja 10.0.0 ukazała się 3 marca 2026 roku.

Co instalujemy

Pojęcie: Apache Solr. Serwer wyszukiwania pełnotekstowego zbudowany na Lucene, z którym rozmawiamy przez HTTP. Wysyłamy do niego dokumenty, a potem pytamy o słowa, które w nich występują, i dostajemy odpowiedź w kilka milisekund.

Pojęcie: indeks odwrócony. Struktura, w której dla każdego słowa pamiętamy listę dokumentów, w których ono występuje, jak skorowidz na końcu książki. Dzięki niej szukanie słowa nie wymaga czytania wszystkich dokumentów, tylko zajrzenia do jednego wpisu.

Pojęcie: rdzeń (core) i kolekcja. Rdzeń to jeden osobny indeks ze swoim zestawem pól, odpowiednik jednej bazy w bazie relacyjnej. Kolekcja to ta sama rzecz w trybie klastra SolrCloud, gdzie jedna kolekcja może być podzielona na kilka rdzeni na różnych serwerach.



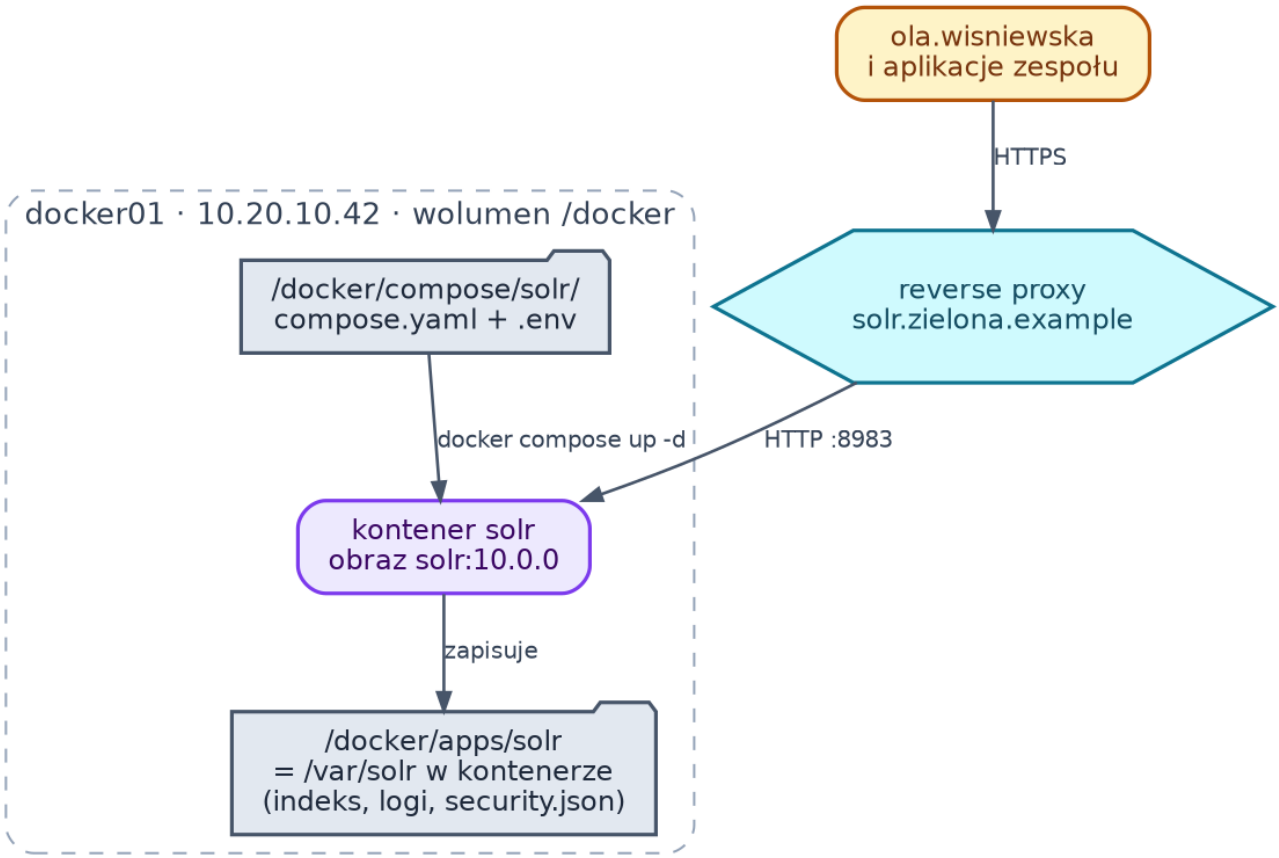
Solr w architekturze Zielonej: aplikacja Oli zapisuje dane w pg01, a kopię do szukania wysyła do Solra na docker01 (kliknięcie otwiera pełny rozmiar)

Użytkownik wpisuje frazę w aplikacji, aplikacja pyta Solra przez HTTP i dostaje listę identyfikatorów. Prawdziwe dane zostają w `pg01`, bo indeks można w każdej chwili odbudować z bazy, a z indeksu bazy nie odbudujemy.

> **Pojęcie: tryb SolrCloud i tryb user-managed.** Od wersji 10 polecenie `bin/solr start` domyślnie uruchamia Solra jako SolrCloud, z wbudowanym ZooKeeperem (osobnym programem, który w klastrze trzyma konfigurację) i z kolekcjami. Tryb user-managed, uruchamiany przełącznikiem `--user-managed`, jest klasycznym, samodzielnym Solrem z rdzeniami. Sposób, w jaki obraz Dockera 10.0.0 wybiera tryb, sprawdzamy po pierwszym starcie w poniższym kroku, bo od tego zależy kilka poleceń (tworzenie, kopia).

> 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
Docker Engine i wtyczka Compose	sprawdzamy `docker compose version`
Wolny port `8983` na hoście	domyślny port Solra
Około 2 GB wolnej pamięci dla kontenera	w pliku Compose ustawiamy stertę Javy na `1g`
Katalog `/docker/apps/solr` należący do użytkownika o numerze 8983	tym numerem pracuje proces Solra w kontenerze, patrz krok 1



Układ plików i ruchu: compose.yaml na docker01, kontener solr, katalog danych /docker/apps/solr jako /var/solr i reverse proxy przed Solrem (kliknięcie otwiera pełny rozmiar)

Obraz przypinamy do konkretnej wersji (`10.0.0`), nigdy do `latest`, a aktualną wersję sprawdzamy na Docker Hub przed każdą instalacją. Plik Compose nie zawiera haseł, więc osobny plik `.env` nie jest potrzebny: hasło użytkownika Solra ustawiamy poleceniem w kroku 11.

> Po instalacji Solr odpowiada każdemu, kto dotrze do portu `8983`, bez pytania o hasło. Zanim udostępnimy go aplikacji z `app01`, włączamy uwierzytelnianie (krok 11) i stawiamy przed nim reverse proxy z HTTPS, jak w Reverse proxy i równoważenie obciążenia.

Przebieg

1. Utworzenie katalogów i nadanie właściciela

```
sudo mkdir -p /docker/apps/solr /docker/compose/solr
sudo chown 8983:8983 /docker/apps/solr
```

Katalog danych musi należeć do użytkownika, którym pracuje Solr w kontenerze, inaczej kontener nie zapisze indeksu. Numer 8983 sprawdzamy poleceniem:

```
docker run --rm --entrypoint id solr:10.0.0
```

Przykład:

```
uid=8983(solr) gid=8983(solr) groups=8983(solr)
```

2. Plik Compose: compose.yaml

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

```
services:
  solr:
    image: solr:10.0.0
    container_name: solr
    restart: always
    ports:
      - "8983:8983"
    environment:
      - SOLR_HEAP=1g
    volumes:
      - /docker/apps/solr:/var/solr
    ulimits:
      nofile:
        soft: 65000
        hard: 65000
```

Katalog `/docker/apps/solr` na hoście to `/var/solr` w kontenerze: tam leży indeks, logi i konfiguracja. Sekcja `ulimits` robi to samo co `LimitNOFILE` w systemd, o którym piszemy w Apache Solr: utrzymanie.

3. Pierwsze uruchomienie: docker compose up

```
cd /docker/compose/solr
docker compose up -d
docker compose logs -f solr
```

Pobranie obrazu zajmuje chwilę. Log przeglądamy, aż pojawi się informacja, że Solr nasłuchuje, i przerywamy go klawiszami Ctrl+C.

4. Sprawdzenie trybu pracy: admin/info/system

```
curl -s "http://localhost:8983/solr/admin/info/system?wt=json" | grep -o '"mode":"[a-z]*"'
```

Wartość `solrcloud` oznacza tryb klastra (kolekcje, ZooKeeper), a `std` tryb samodzielny (rdzenie). Zapamiętujemy wynik, bo wraca w kroku 5 i w kopii zapasowej.

Przykład:

```
"mode":"solrcloud"
```

5. Utworzenie rdzenia albo kolekcji: solr create

```
docker exec solr solr create -c dokumenty
```

W trybie samodzielnym polecenie tworzy rdzeń, a w SolrCloud kolekcję, w obu przypadkach o nazwie `dokumenty`.

Przykład (tryb samodzielny):

```
Created new core 'dokumenty'
```

Przykład (SolrCloud):

```
Created collection 'dokumenty' with 1 shard(s), 1 replica(s) with config-set 'dokumenty'
```

6. Zdefiniowanie pól: Schema API

```
curl -X POST -H 'Content-type:application/json' "http://localhost:8983/solr/dokumenty/schema" -d '{"add-field":[{"name":"tytul","type":"text_general","stored":true},{"name":"tresc","type":"text_general","stored":true},{"name":"dzial","type":"string","stored":true}]}'
```

Mówimy Solrowi, jakie pola mają nasze dokumenty. Pola `tytul` i `tresc` są tekstem dzielonym na słowa (`text_general`), a `dzial` jednym niepodzielnym napisem (`string`), po którym będziemy filtrować.

Przykład:

```
{
  "responseHeader":{
    "status":0,
    "QTime":87}}
```

7. Przygotowanie przykładowych dokumentów: dokumenty.json

Plik `/docker/compose/solr/dokumenty.json`:

```
[
  {"id": "1", "tytul": "Regulamin magazynu w Swarzędzu", "tresc": "Zmiana zaczyna się o 6:00. Każdy pracownik loguje skaner przed wydaniem towaru.", "dzial": "magazyn"},
  {"id": "2", "tytul": "Instrukcja logowania skanera", "tresc": "Skaner loguje się do systemu WMS. Gdy skaner nie widzi serwera, zgłaszamy to kierownikowi zmiany.", "dzial": "magazyn"},
  {"id": "3", "tytul": "Procedura zamknięcia miesiąca", "tresc": "Księgowość kończy ewidencję faktur do trzeciego dnia roboczego i wysyła raport do prezesa.", "dzial": "ksiegowosc"},
  {"id": "4", "tytul": "Zasady pracy zdalnej handlowców", "tresc": "Handlowiec łączy się z siecią firmy przez VPN. Faktury drukujemy w centrali.", "dzial": "handel"}
]
```

Każdy dokument ma unikalne `id`. Wysłanie dokumentu z już istniejącym `id` zastępuje stary, więc ładowanie można bezpiecznie powtarzać.

8. Załadowanie dokumentów do Solra: curl update

```
curl -X POST -H 'Content-type:application/json' --data-binary @dokumenty.json
"http://localhost:8983/solr/dokumenty/update?commit=true"
```

Parametr `commit=true` każe Solrowi od razu zatwierdzić zmiany, żeby dokumenty były widoczne w wynikach. Przy dużych ładowaniach commit robimy raz na końcu, nie przy każdej porcji.

Przykład:

```
{
  "responseHeader":{
    "status":0,
    "QTime":112}}
```

Dokumenty można też załadować narzędziem dołączonym do Solra, które samo rozpoznaje typ pliku:

```
docker exec solr solr post -c dokumenty /var/solr/dokumenty.json
```

Ta wersja wymaga, żeby plik leżał wewnątrz kontenera, więc najpierw kopiujemy go do `/docker/apps/solr` na hoście. Opis opcji narzędzia `post` sprawdzamy w dokumentacji Solra dla naszej wersji.

9. Pierwsze wyszukiwanie: select

```
curl "http://localhost:8983/solr/dokumenty/select?q=tresc:skaner&fl=id,tytul&rows=5"
```

(gdzie `q` to zapytanie, `fl` lista pól w odpowiedzi, a `rows` maksymalna liczba wyników)

Przykład:

```
{
  "responseHeader":{
    "status":0,
    "QTime":4,
    "params":{
      "q":"tresc:skaner",
      "fl":"id,tytul",
      "rows":"5"}},
  "response":{"numFound":2,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"1",
```

```
        "tytul":"Regulamin magazynu w Swarzędzu"},
      {
        "id":"2",
        "tytul":"Instrukcja logowania skanera"}]
  }}
```

Oba dokumenty ze słowem „skaner" w treści wracają w kilka milisekund. Wielkość liter nie ma znaczenia, bo Solr normalizuje tekst przy indeksowaniu.

10. Zawężenie wyników do jednego działu: fq

```
curl "http://localhost:8983/solr/dokumenty/select?q=*:*&fq=dzial:handel&fl=id,tytul"
```

Parametr `fq` to filtr, który zawęża wyniki, ale nie wpływa na ich kolejność. Zwraca tu jeden dokument, numer 4. Filtry po niepodzielnych polach, jak `dzial`, są szybkie, bo Solr zapamiętuje ich wyniki.

11. Włączenie uwierzytelniania Basic Auth: solr auth enable

```
docker exec solr solr auth enable --type basicAuth --credentials nazwa_usera:HasloDoSolra
```

Od tej chwili każde zapytanie bez hasła kończy się odpowiedzią `401`, więc do kolejnych poleceń `curl` dodajemy `--user nazwa_usera`. Jeśli polecenie zgłosi brak adresu ZooKeepera albo pliku `solr.in.sh`, sprawdzamy w dokumentacji rozdział Basic Authentication Plugin, jakich opcji wymaga nasz tryb. Przełączanie, zmianę hasła przez API i diagnozę błędu `access denied` opisujemy w Apache Solr: utrzymanie.

Sprawdzamy, że zamek działa:

```
curl -i "http://localhost:8983/solr/dokumenty/select?q=*:*"
```

Przykład:

```
HTTP/1.1 401 Unauthorized
```

12. Katalog na kopie zapasowe

```
sudo install -d -o 8983 -g 8983 /docker/apps/solr/data/kopie
```

Kopie z API zapisujemy pod `/var/solr/data/kopie` w kontenerze, czyli w katalogu domowym Solra. Solr domyślnie pozwala pisać i czytać tylko pod swoim katalogiem domowym, więc ścieżka spoza niego kończy się błędem. Z hosta ten katalog zabiera potem NetWorker na `bkp01`.

13. Kopia rdzenia przez API replikacji: command=backup

Dla trybu samodzielnego (`"mode":"std"`):

```
curl --user nazwa_usera
"http://localhost:8983/solr/dokumenty/replication?command=backup&location=/var/solr/data/kopie
&name=dokumenty-2026-10-03&numberToKeep=7"
```

(gdzie `name` to nazwa kopii, a `numberToKeep` liczba kopii, które zostawiamy, starsze znikają)

Przebieg kopii sprawdzamy poleceniem:

```
curl --user nazwa_usera
"http://localhost:8983/solr/dokumenty/replication?command=details&wt=json"
```

W odpowiedzi szukamy w sekcji `backup` statusu `success`.

14. Kopia kolekcji w SolrCloud: action=BACKUP

Dla trybu klastra (`"mode":"solrcloud"`) kopię robi Collections API, a nie replikacja:

```
curl --user nazwa_usera
"http://localhost:8983/solr/admin/collections?action=BACKUP&name=dokumenty-2026-10-
03&collection=dokumenty&location=/var/solr/data/kopie"
```

Każdy katalog kopii może zawierać kopię tylko jednej kolekcji, więc dla kolejnych kolekcji zakładamy osobne podkatalogi.

15. Znacznik przed ryzykowną zmianą: CREATESNAPSHOT

```
curl --user nazwa_usera
"http://localhost:8983/solr/admin/cores?action=CREATESNAPSHOT&core=dokumenty&commitName=przed-
aktualizacja"
```

Snapshot to nazwany znacznik na konkretnym stanie indeksu, a nie osobna kopia plików: chroni ten stan przed usunięciem przy scalaniu segmentów. Samą kopię robimy z niego potem poleceniem `command=backup` z dodanym `commitName=przed-aktualizacja`. W SolrCloud analogiczne polecenie wywołujemy w Collections API.

16. Odtworzenie rdzenia z kopii: command=restore

```
curl --user nazwa_usera
"http://localhost:8983/solr/dokumenty/replication?command=restore&name=dokumenty-2026-10-
03&location=/var/solr/data/kopie"
```

Postęp sprawdzamy poleceniem `command=restorestatus`, a wynik `success` oznacza koniec. Odtwarzamy tylko po zatrzymaniu aplikacji, która pisze do indeksu, bo w trakcie odtwarzania Solr odpowiada niespójnie.

17. Aktualizacja do nowszej wersji: docker compose pull

Najpierw robimy kopię (kroki 13 albo 14), potem zmieniamy numer wersji w `compose.yaml` i wykonujemy:

```
cd /docker/compose/solr
docker compose pull
docker compose up -d
```

Przy aktualizacji w obrębie tej samej wersji głównej (np. 10.0.0 do 10.0.1) zmiana jest zwykle bezbolesna. Przy skoku między wersjami głównymi czytamy najpierw Upgrade Notes: Lucene czyta indeksy z poprzedniej wersji głównej, ale nie z dwóch wstecz, więc stary indeks trzeba wtedy

zbudować od nowa z bazy.

Weryfikacja

Sprawdzamy, że kontener działa:

```
docker compose ps
```

Przykład:

```
NAME      IMAGE          STATUS          PORTS
solr      solr:10.0.0    Up 5 minutes    0.0.0.0:8983->8983/tcp
```

Sprawdzamy, że rdzeń odpowiada (przed włączeniem uwierzytelniania, albo z `--user nazwa_usera` po nim):

```
curl -s "http://localhost:8983/solr/dokumenty/admin/ping"
```

Przykład:

```
{
  "responseHeader":{
    "status":0,
    "QTime":1},
  "status":"OK"}
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: panel administracyjny Solra pod `http://docker01:8983/solr/`, wybrany rdzeń `dokumenty` z lewego menu, zakładka „Query" z wykonanym zapytaniem `q=tresc:skaner` i widocznym wynikiem `numFound: 2`.

Wycofanie zmian

Zatrzymanie i usunięcie kontenera, bez kasowania indeksu:

```
cd /docker/compose/solr
docker compose down
```

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

```
docker compose down
sudo rm -rf /docker/apps/solr
```

Powrót do poprzedniej wersji po nieudanej aktualizacji wymaga kopii: przywracamy stary numer wersji w `compose.yaml` i odtwarzamy rdzeń z kopii z kroku 13 albo 14.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
Kontener restartuje się, w logu `Permission denied` na `/var/solr`	katalog `/docker/apps/solr` nie należy do 8983 albo blokuje go SELinux	`sudo chown 8983:8983 /docker/apps/solr`, sprawdzamy `getenforce`, w razie potrzeby dopisujemy `:z` do wolumenu
`curl: (7) Connection refused` zaraz po `up -d`	Solr startuje kilkanaście sekund	`docker compose logs -f solr`, ponawiamy po komunikacie o nasłuchu
`unknown field 'tytul'` przy ładowaniu	pola nie zostały zdefiniowane	wykonujemy krok 6 przed krokiem 8
`401 Unauthorized` w każdym poleceniu	włączone uwierzytelnianie	dopisujemy `--user nazwa_usera`
`solr create` zgłasza, że rdzeń już istnieje	rdzeń o tej nazwie jest od poprzedniej próby	wybieramy inną nazwę albo usuwamy stary rdzeń
Kopia przez replikację nie działa w SolrCloud	tryb klastra używa Collections API	używamy kroku 14 zamiast 13
Błąd o niedozwolonej ścieżce przy kopii	`location` poza katalogiem domowym Solra	zapisujemy pod `/var/solr/data/kopie`
`Too many open files` w logu Solra	zbyt niski limit plików kontenera	podnosimy `nofile` w sekcji `ulimits` w `compose.yaml`

Objaw	Przyczyna	Co robimy
Port `8983` zajęty	inny proces na hoście	`ss -tlnp \

□□ Ćwiczenia

Ćwiczenie 1: dodanie dokumentu

Dopisujemy do `dokumenty.json` piąty dokument o dziale `handel`, ładujemy go ponownie i sprawdzamy, że `fq=dzial:handel` zwraca teraz dwa wyniki.

Odpowiedź

Dopisujemy obiekt z `"id": "5"` i `"dzial": "handel"`, a potem powtarzamy krok 8. Dokumenty o istniejących numerach zostają zastąpione identycznymi, a nowy pojawia się obok, więc zapytanie z krokiem 10 zwraca dokumenty 4 i 5.

Ćwiczenie 2: tryb pracy

Po co sprawdzamy tryb Solra w kroku 4, zanim zaczniemy robić kopie zapasowe?

Odpowiedź

Bo w trybie samodzielnym kopię rdzenia robimy przez API replikacji, a w SolrCloud kopię kolekcji przez Collections API. Użycie niewłaściwego polecenia kończy się błędem albo kopią, której nie da się odtworzyć.

Ćwiczenie 3: gdzie są dane

Ola pyta, czy może przestać trzymać treść instrukcji w bazie `pg01`, skoro wszystko jest już w Solrze. Co odpowiadamy?

Odpowiedź

Nie. Solr jest kopią do szukania, a nie miejscem na jedyny egzemplarz danych: indeks można odbudować z bazy, ale bazy z indeksu już nie. Prawdziwe dane zostają w `pg01`, a Solr dostaje ich kopię.

Powiązane: Apache Solr: utrzymanie · Generowanie certyfikatów: OpenSSL i urząd certyfikacji Windows · Reverse proxy i równoważenie obciążenia · Instalacja Gitea w Dockerze

Źródła: solr.apache.org/guide, rozdziały „Solr in Docker", „Solr Control Script Reference", „Basic Authentication Plugin", „Backup and Restore", „Collection Management", „Major Changes in Solr 10" · hub.docker.com/_/solr (tagi, sprawdzono 3.10.2026) · en.wikipedia.org, hasła „Apache Lucene" i „Apache Solr"

Revision #16
Created 2026-10-03 09:39:02 UTC by Tomasz Konieczny
Updated 2026-10-26 14:14:28 UTC by Tomasz Konieczny