Upgrade silnika bazy danych

□□ Typ: procedura

□□ Dotyczy: MongoDB 5.0–8.0, wdrożenie Docker Compose (wariant baremetal opisany przy każdym kroku)

□ Czas: ~60 min na jeden skok wersji

Efekt: silnik MongoDB działa na kolejnej wersji głównej, z podniesionym poziomem kompatybilności funkcji, z zachowanymi danymi i możliwością wycofania się do chwili, aż ten poziom zostanie faktycznie podniesiony.

□□ Skróty użyte na tej stronie (2)

Skrót	Co znaczy
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
FCV	featureCompatibilityVersion: parametr MongoDB pilnujący, z którą wersją formatu danych baza aktualnie pracuje, niezależnie od wersji zainstalowanych binariów

Zanim zaczniemy

□□ Z ŻYCIA WZIĘTE

Przed weekendową promocją w sklepie B2B Tomek planuje skok replica setu mongo01 z wersji 5.0 na 6.0, żeby zdążyć przed zakazem zmian obowiązującym tuż przed dużymi wydarzeniami sprzedażowymi. Aktualizację zaczyna od węzła secondary, zgodnie z procedurą, dzięki czemu przez cały czas trwania operacji katalog produktów odpowiada klientom bez przerwy z węzła primary działającego na sprawdzonej wersji. Dopiero gdy secondary działa już stabilnie na nowym silniku, primary świadomie ustępuje przez rs.stepDown() i przejmuje po nim rolę zaktualizowany węzeł, więc żaden klient sklepu nie widzi ani sekundy niedostępności.

Pojęcie: zasada pojedynczego skoku. MongoDB pozwala aktualizować silnik tylko o jedną wersję główną naraz — nie da się przeskoczyć na przykład z wersji 5.0 od razu do wersji 7.0. Trzeba przejść przez każdą wersję pośrednią, jedna po drugiej: `5.0 → 6.0 → 7.0 → 8.0`. Każdy taki skok to osobne, pełne powtórzenie procedury opisanej niżej, nie jeden

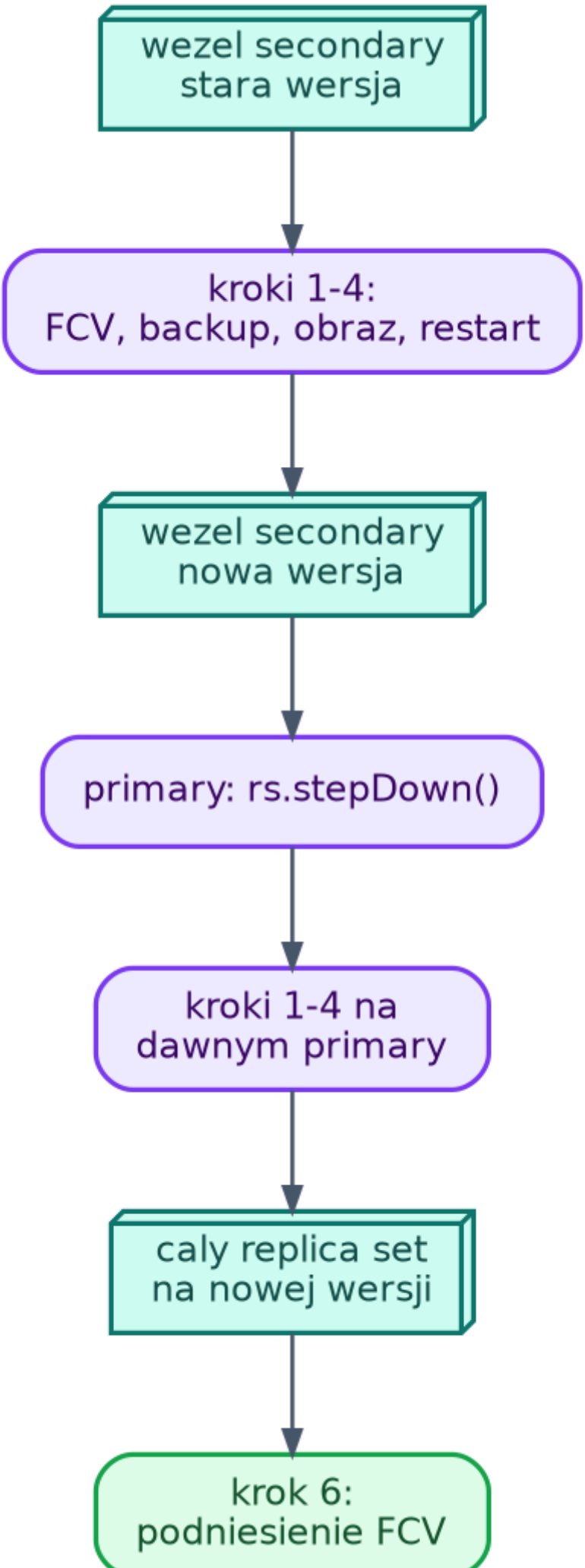
większy skok wykonany raz.

Pojęcie: featureCompatibilityVersion (FCV). Parametr określający, z którą wersją głównego formatu funkcji i danych bieżąco pracuje serwer, niezależnie od wersji zainstalowanych binariów. Nowa wersja binariów potrafi uruchomić się i pracować przy FCV ustawionym na wersję starszą — dopóki FCV nie zostanie jawnie podniesiony, nowe funkcje danej wersji pozostają wyłączone, ale za to zachowana jest możliwość powrotu do poprzednich binariów bez utraty spójności danych. Dlatego kolejność w tej procedurze jest zawsze taka sama: najpierw upewniamy się, że FCV odpowiada obecnej wersji, potem podmieniamy silnik, dopiero po sprawdzeniu, że nowa wersja działa poprawnie, podnosimy FCV do wersji docelowej.

Wymaganie	Jak sprawdzić
Aktualna wersja FCV zgodna z obecną wersją binariów	`db.adminCommand({ getParameter: 1, featureCompatibilityVersion: 1 })`
Świeży backup przed jakąkolwiek zmianą	`mongodump`, opisany na stronie Backup i odtwarzanie: mongodump i mongorestore
Kolejność węzłów w replica secie: secondary, potem primary	`rs.status()`, opisane na stronie mongosh: powłoka administracyjna

Uważaj: aktualizacja zmienia format danych na dysku. Bez wykonanego wcześniej backupu wycofanie się do poprzedniej wersji po nieudanej aktualizacji bywa niemożliwe, nie tylko utrudnione.

W replica secie aktualizację prowadzimy zawsze od węzłów `secondary`, dopiero na końcu od `primary` po jego świadomym ustąpieniu (`step down`) — dzięki temu w każdej chwili trwania procedury co najmniej jeden węzeł obsługuje ruch aplikacji na nowszej albo na sprawdzonej starszej wersji, a nigdy nie zatrzymujemy całego replica setu naraz.


wezel secondary
stara wersja
kroki 1-4:
FCV, backup, obraz, restart
wezel secondary
nowa wersja
primary: rs.stepDown()
kroki 1-4 na
dawnym primary
caly replica set
na nowej wersji
krok 6:
podniesienie FCV

Kolejność aktualizacji w replica secie: najpierw secondary, potem stepDown primary, na końcu podniesienie FCV (kliknięcie otwiera pełny rozmiar)

Przebieg

Instrukcja opisuje jeden skok między sąsiednimi wersjami głównymi (na przykładzie `5.0 → 6.0`) dla wdrożenia opartego na Docker Compose. Wariant baremetal różni się wyłącznie krokiem podmiany silnika — zaznaczono to przy nim osobno.

1. Ustawienie FCV na wersję obecną

Wchodzimy do powłoki `mongosh` w działającym kontenerze:

```
docker exec -it NazwaKontenera mongosh
```

Sprawdzamy bieżący poziom kompatybilności:

```
db.adminCommand({ getParameter: 1, featureCompatibilityVersion: 1 })
```

Jawnie ustawiamy FCV na wersję, na której obecnie pracujemy, żeby mieć pewność, że dane są w pełni zgodne z bieżącą wersją, zanim podmienimy silnik:

```
db.adminCommand({ setFeatureCompatibilityVersion: "5.0" })
```

```
{ ok: 1 }
```

Wariant baremetal: kroki FCV są identyczne, różni się tylko sposób wejścia do powłoki — samym poleceniem `mongosh`, bez `docker exec`.

2. Wykonanie kopii zapasowej

```
mongodump --db NazwaBazy --gzip --archive > "backup_przed_upgrade.gz"
```

Wykonujemy pełny backup przed jakąkolwiek dalszą zmianą, zgodnie z procedurą opisaną na stronie Backup i odtwarzanie: mongodump i mongorestore. Ten krok jest tak samo obowiązkowy przy węźle `secondary`, jak przy `primary`.

3. Zmiana wersji obrazu w docker-compose.yaml

```
NazwaUslugi:
  image: mongo:6.0
  container_name: NazwaKontenera
```

Podnosimy tag obrazu usługi bazy o jedną wersję główną w pliku `docker-compose.yaml`, zmieniając wyłącznie wersję w polu `image` — reszta konfiguracji usługi pozostaje bez zmian.

Wariant baremetal: odpowiednikiem tego kroku i kroku 4 jest przełączenie repozytorium pakietów MongoDB na nową wersję główną, zatrzymanie usługi `mongod`, instalacja nowego pakietu i ponowne uruchomienie usługi. Dane i konfiguracja na dysku pozostają nietknięte — podmieniają się wyłącznie binaria.

4. Restart kontenerów na nowym obrazie

```
docker compose down
docker compose pull
docker compose up -d
```

`down` zatrzymuje i usuwa kontenery, przy czym dane pozostają na przypisanym woluminie i nie są kasowane. `pull` pobiera obraz w nowo zadeklarowanej wersji, a `up -d` uruchamia kontenery w tle już na nowym silniku, startującym z dotychczasową zawartością wolumenu danych.

```
[+] Running 2/2
 ✔ Container mongo-secondary  Started
 ✔ Container mongo-primary    Started
```

Kolejność w replica secie: ten sam zestaw czterech kroków wykonujemy najpierw na każdym węźle `secondary` z osobna. Dopiero gdy wszystkie `secondary` działają już na nowej wersji, wykonujemy na `primary` `rs.stepDown()`, żeby świadomie oddał rolę primary jednemu z już zaktualizowanych węzłów `secondary`, i dopiero wtedy powtarzamy kroki 3–4 na nim.

5. Weryfikacja uruchomienia

```
docker compose ps
```

```
NAME             IMAGE        STATUS
mongo-primary    mongo:6.0    Up 42 seconds
mongo-secondary  mongo:6.0    Up 3 minutes
```

Wariant baremetal: stan usługi sprawdzamy menedżerem usług systemu (na przykład `systemctl status mongod`) zamiast `docker compose ps`.

Potwierdzamy wersję silnika, na której faktycznie działa kontener:

```
docker exec -it NazwaKontenera mongosh --quiet --eval 'db.version()'
```

```
6.0.19
```

Na tym etapie warto pozwolić bazie popracować przez jakiś czas w tak zwanym trybie „burn-in", zanim przejdziemy do kroku 6 — dopóki FCV nie zostanie podniesiony, nadal istnieje droga powrotu do poprzedniego silnika bez utraty spójności danych.

6. Podniesienie FCV na wersję docelową

Gdy nowa wersja działa stabilnie na wszystkich węzłach replica setu, wchodzimy ponownie do powłoki i podnosimy FCV do wersji docelowej — dopiero ta operacja faktycznie odblokowuje nowe funkcje nowej wersji:

```
db.adminCommand({ setFeatureCompatibilityVersion: "6.0" })
```

Od MongoDB 7.0 wymagane jest dodatkowo pole `confirm: true` — bez niego komenda kończy się błędem. Dla skoków do wersji 7.0 i wyższych używamy formy:

```
db.adminCommand({ setFeatureCompatibilityVersion: "7.0", confirm: true })
```

Przy skokach do wersji 5.0 i 6.0 parametr `confirm` nie jest potrzebny, a na starszych binariach bywa odrzucony jako nieznane pole.

Jeśli planujemy kolejny skok w tej samej sesji administracyjnej (na przykład zaraz dalej do 7.0), wracamy do kroku 1 z nowymi wartościami wersji — cała procedura powtarza się od nowa dla każdego kolejnego skoku.

Weryfikacja

Po zakończeniu skoku sprawdzamy jeszcze raz stan całego replica setu:

```
rs.status()
```

```
{
  set: 'rsAppdb',
  members: [
    { _id: 0, name: 'mongo01:27017', stateStr: 'PRIMARY', health: 1 },
    { _id: 1, name: 'mongo02:27017', stateStr: 'SECONDARY', health: 1 }
  ],
  ok: 1
}
```

I potwierdzamy FCV faktycznie ustawiony na wartość docelową:

```
db.adminCommand({ getParameter: 1, featureCompatibilityVersion: 1 })
```

```
{ featureCompatibilityVersion: { version: '6.0' }, ok: 1 }
```

Wycofanie zmian

Dopóki FCV nie został podniesiony do wersji docelowej (koniec kroku 6), cofnięcie się do poprzedniego silnika polega na powtórzeniu kroków 3–4 ze starszym tagiem obrazu — dane pozostają w formacie zgodnym z tą wersją, bo FCV nadal na nią wskazuje.

Po podniesieniu FCV cofnięcie wymaga dodatkowego kroku: najpierw obniżamy FCV z powrotem do wersji wcześniejszej,

```
db.adminCommand({ setFeatureCompatibilityVersion: "5.0" })
```

a dopiero potem podmieniamy obraz albo pakiet z powrotem na starszą wersję. Starsza wersja binariów nie uruchomi się poprawnie, jeśli FCV nadal wskazuje wersję nowszą, niż te binaria potrafią obsłużyć.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Kontener na nowej wersji nie startuje po `docker compose up -d`	FCV nie został ustawiony na wersję obecną przed podmianą obrazu (pominięty krok 1)	powrót do poprzedniego obrazu, wykonanie kroku 1, ponowna próba

Objaw	Przyczyna	Naprawa
`setFeatureCompatibilityVersion` do wersji 7.0+ zwraca błąd o brakującym polu	brak `confirm: true`, wymaganego od MongoDB 7.0	powtórzenie komendy z dodanym `confirm: true`
Węzeł `primary` nie chce ustąpić przy `rs.stepDown()`	trwająca długa operacja zapisu blokująca ustąpienie	odczekanie na zakończenie operacji albo zwiększenie limitu czasu w `rs.stepDown(sekundy)`
Aplikacja zgłasza błędy po skoku, mimo poprawnego `db.version()`	sterownik (driver) aplikacji nieaktualny względem nowej wersji serwera	sprawdzenie macierzy kompatybilności sterownika z nową wersją MongoDB przed kolejnym skokiem
Próba przeskoczenia bezpośrednio o dwie wersje główne	złamana zasada pojedynczego skoku	wykonanie procedury osobno dla każdej wersji pośredniej, po kolei

Powiązane: Architektura MongoDB: dokument, kolekcja, baza, replica set · mongosh: powłoka administracyjna · Backup i odtwarzanie: mongodump i mongorestore

Źródła: notatki własne administratora MongoDB · dokumentacja MongoDB, rozdział „Upgrade a Replica Set" i „setFeatureCompatibilityVersion"

Revision #44
Created 2026-06-09 09:50:33 UTC by Tomasz Konieczny
Updated 2026-10-26 14:10:05 UTC by Tomasz Konieczny