Zmiana nazwy klastra i datacenter

Typ: procedura **Dotyczy: Apache Cassandra 3.x / 4.x** **Czas: ~90 min**

Efekt: klaster działający pod nową nazwą i/lub z nowym oznaczeniem datacenter, bez utraty danych — kosztem chwilowej niedostępności i utraty historii napraw w Cassandra Reaper, jeśli jest używany.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
CQL	Cassandra Query Language: język zapytań Cassandry, podobny do SQL
DC	Domain Controller: serwer, który trzyma Active Directory i loguje użytkowników
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy

Zanim zaczniemy

Z ŻYCIA WZIĘTE

Po fuzji dwóch zespołów IT pan Zbigniew chce, żeby nazwa klastra Cassandry odzwierciedlała nową strukturę firmy, zamiast starej nazwy roboczej sprzed lat. Operacja wymaga zmiany cluster_name na cass01, cass02 i cass03 jednocześnie, bo węzeł uruchomiony z nową nazwą, podczas gdy reszta wciąż działa pod starą, zostałby odrzucony przez gossip jako niezgodny. Wykonujemy ją w niedzielę wieczorem, poza godzinami szczytu magazynu, z pełnym backupem konfiguracji i danych na wszelki wypadek. Po restarcie wszystkich trzech węzłów naraz klaster wraca online pod nową nazwą, bez utraty ani jednego rekordu telemetrii.

Zmianę nazwy klastra i datacenter wykonujemy tylko wtedy, gdy jest to naprawdę konieczne — wykonana na działającym środowisku produkcyjnym generuje realne ryzyko problemów, i to zarówno przy samej zmianie, jak i przy sprzątaniu po niej.

Wymaganie	Jak sprawdzić

Dostęp roota (sudo) na każdym węźle klastra	`sudo -l` na każdym z węzłów
Aktualny backup / snapshot danych	`nodetool listsnapshots`, opisane na stronie Backup i odtwarzanie
Kopia plików konfiguracyjnych	patrz krok 1
Zidentyfikowane keyspace ze strategią `NetworkTopologyStrategy` — jeśli zmieniamy nazwę datacenter	`DESCRIBE KEYSPACES` w `cqlsh`, opisane na stronie cqlsh: klient wiersza poleceń

Uważaj: wszystkie kroki od 3 do 7 wykonujemy równolegle, na każdym węźle klastra naraz — nie węzeł po węźle. Węzeł uruchomiony z nową nazwą klastra, podczas gdy reszta klastra wciąż działa pod starą, zostanie odrzucony przez gossip jako niezgodny. Operacja niesie realne ryzyko uszkodzenia spójności klastra przy błędzie w którymkolwiek z kroków — wykonujemy ją poza godzinami szczytu, z pełną świadomością, że w razie problemu wracamy do backupu z kroku 1.

Przebieg

1. Backup konfiguracji i danych

```
sudo cp /etc/cassandra/conf/cassandra.yaml /etc/cassandra/conf/cassandra_backup.yaml
```

Kopiujemy plik konfiguracyjny przed jakąkolwiek zmianą — na każdym węźle z osobna.

Wykonujemy też snapshot danych opisany na stronie Backup i odtwarzanie, jeśli zmiana dotyczy również datacenter i wiąże się z późniejszą modyfikacją definicji keyspace.

2. Jeśli w klastrze działa Cassandra Reaper — zabezpieczenie definicji jego keyspace

```
DESCRIBE reaper_db;
```

Wykonujemy to w `cqlsh` przed dalszymi krokami — polecenie jako pierwszy element wyniku zwraca gotową instrukcję `CREATE KEYSPACE`, którą ten keyspace powstał. Zapisujemy ją — będzie potrzebna w kroku 9, jeśli po zmianie nazwy datacenter trzeba będzie odtworzyć ten keyspace od zera.

3. Zmiana nazwy klastra w cassandra.yaml (równolegle na każdym węźle)

```
cluster_name: 'NazwaKlastra'
```

Edytujemy wartość `cluster_name` w `/etc/cassandra/conf/cassandra.yaml` na każdym węźle klastra jednocześnie, na nową, docelową nazwę.

4. Zmiana nazwy datacenter i/lub racku (opcjonalnie, jeśli dotyczy)

```
dc=NazwaDatacenter
rack=NazwaRacku
```

Edytujemy `/etc/cassandra/conf/cassandra-rackdc.properties` na każdym węźle, jeśli zmiana obejmuje też datacenter albo rack — nie tylko samą nazwę klastra. Ten krok pomijamy, jeśli zmieniamy wyłącznie `cluster_name`.

5. Dodanie tymczasowej dyrektywy JVM ignorującej niezgodność racku i DC

```
JVM_OPTS="$JVM_OPTS -Dcassandra.ignore_rack=true -Dcassandra.ignore_dc=true"
```

Dopisujemy tę linię w `/etc/cassandra/conf/cassandra-env.sh` na każdym węźle, jeśli zmieniamy nazwę racku lub datacenter. Bez niej węzeł odmówi startu, widząc niezgodność między tym, co ma zapisane lokalnie z poprzedniego uruchomienia, a nową wartością w `cassandra-rackdc.properties` — dyrektywa pozwala mu wystartować mimo tej niezgodności, zamiast wymuszać pełny `nodetool rebuild`. Ten krok pomijamy, jeśli zmieniamy wyłącznie `cluster_name` bez ruszania racku i DC.

6. Aktualizacja nazwy klastra w lokalnym stanie węzła

```
UPDATE system.local SET cluster_name = 'NazwaKlastra' WHERE key = 'local';
```

Wykonujemy to poleceniem CQL, połączeni przez `cqlsh` bezpośrednio do danego węzła — z uwagi na to, że tabela `system.local` nie jest replikowana i zawsze odzwierciedla stan tego konkretnego węzła, wykonujemy je osobno na każdym węźle, łącząc się za każdym razem z nim samym. Bez tego kroku restart usługi zakończy się błędem niezgodności między nową nazwą w `cassandra.yaml` a starą nazwą zapisaną lokalnie przez węzeł przy poprzednim uruchomieniu.

7. Zatrzymanie i ponowne uruchomienie usługi (równolegle na każdym węźle)

```
sudo systemctl stop cassandra.service
```

```
sudo systemctl start cassandra.service
```

Restartujemy usługę na wszystkich węzłach mniej więcej jednocześnie, żeby okno, w którym część klastra działa pod starą nazwą, a część pod nową, było jak najkrótsze.

8. Weryfikacja i sprzątanie tymczasowej dyrektywy JVM

```
sudo nodetool status
```

Sprawdzamy, czy wszystkie węzły wróciły do stanu `UN`. Jeśli usługa wstała poprawnie, usuwamy dyrektywę dodaną w kroku 5 z `cassandra-env.sh` i restartujemy usługę jeszcze raz na każdym węźle, żeby potwierdzić, że start jest stabilny bez tego tymczasowego obejścia. Jeśli usługa nie wstanie bez tej dyrektywy, cofamy tę zmianę — pozostawiamy linię z `-Dcassandra.ignore_rack=true -Dcassandra.ignore_dc=true` odkomentowaną do czasu wyjaśnienia przyczyny.

9. Aktualizacja definicji keyspace przy zmianie nazwy datacenter

```
ALTER KEYSPACE NazwaKeyspace WITH replication = {'class': 'NetworkTopologyStrategy',
'NazwaDatacenter': 3};
```

Jeśli usługa wstała poprawnie i zmienialiśmy nazwę datacenter, sprawdzamy w `cqlsh`, które keyspace używają strategii `NetworkTopologyStrategy` opisanej na stronie Architektura Cassandry: pierścień, węzeł, replikacja, i dla każdego z nich wykonujemy `ALTER KEYSPACE` z nową nazwą datacenter, zachowując tę samą wartość replication factor, jaką miał wcześniej.

10. Reset Cassandra Reaper, jeśli był używany

Zmiana nazwy klastra i/lub datacenter zostawia Cassandra Reaper w stanie niespójnym z rzeczywistą topologią — klaster i job'y napraw wciąż odwołują się do starej nazwy. Najprostszym i jednocześnie najbardziej drastycznym rozwiązaniem jest reset stanu narzędzia:

```
sudo systemctl stop cassandra-reaper.service
```

```
DROP KEYSPACE reaper_db;
```

```
CREATE KEYSPACE reaper_db WITH replication = {'class': 'NetworkTopologyStrategy',
'NazwaDatacenter': 3};
```

Odtwarzamy pusty keyspace dokładnie tą definicją, którą zapisaliśmy w kroku 2 (z uwzględnieniem nowej nazwy datacenter, jeśli się zmieniła). Następnie edytujemy `clusterName` w `/etc/cassandra-reaper/cassandra-reaper.yaml`:

```
cassandra:
  clusterName: "NazwaKlastra"
```

```
sudo systemctl restart cassandra-reaper.service
```

Efekt jest równoważny świeżej instalacji narzędzia — historia napraw i skonfigurowane wcześniej zadania (job'y) zostają utracone. Po restarcie logujemy się do panelu administracyjnego Cassandra Reaper, dodajemy klaster ponownie i konfigurujemy zadania naprawy od nowa.

Weryfikacja

```
sudo nodetool describecluster
```

```
Cluster Information:
    Name: NazwaKlastra
    Snitch: org.apache.cassandra.locator.GossipingPropertyFileSnitch
    Partitioner: org.apache.cassandra.dht.Murmur3Partitioner
    Schema versions:
        a1b2c3d4-e5f6-3789-a123-456789abcdef: [10.20.10.36, 10.20.10.37, 10.20.10.38]
```

Jedna wersja schematu widziana identycznie przez wszystkie węzły i nazwa klastra zgodna z oczekiwaną potwierdzają, że zmiana się rozpropagowała poprawnie do całego klastra.

Wycofanie zmian

Do kroku 6 włącznie wycofanie polega na przywróceniu skopiowanego w kroku 1 pliku `cassandra_backup.yaml` (oraz `cassandra-rackdc.properties`, jeśli był zmieniany) i restarcie usługi ze starą konfiguracją — węzły nie zdążyły jeszcze wystartować z nową nazwą, więc lokalny stan pozostaje zgodny ze starą. Po kroku 7, jeśli klaster wystartował niestabilnie, powrót do poprzedniego stanu wymaga powtórzenia kroków 3–7 z odwróconymi wartościami (starą nazwą klastra/DC) — z tego samego powodu co przy zmianie docelowej, róbmy to równolegle na wszystkich węzłach.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Węzeł nie startuje, błąd niezgodności nazwy klastra	pominięty krok 6 — `system.local` wciąż ma starą nazwę	wykonujemy `UPDATE system.local SET cluster_name = ...` i uruchamiamy usługę ponownie
Węzeł nie startuje, błąd niezgodności racku/DC	brak dyrektywy z kroku 5 albo usunięta zbyt wcześnie	przywracamy dyrektywę `JVM_OPTS` z krokiem ignorowania racku/DC, restartujemy usługę
Węzły widzą różne wersje schematu długo po restarcie	restart wykonany zbyt rozłącznie w czasie między węzłami	sprawdzamy `nodetool describecluster` na każdym węźle, w razie potrzeby restartujemy opóźnione węzły
Cassandra Reaper nie widzi klastra po restarcie	`clusterName` w `cassandra-reaper.yaml` nie zgadza się z nową nazwą	poprawiamy plik i restartujemy `cassandra-reaper.service`

Powiązane: Architektura Cassandry: pierścień, węzeł, replikacja · Konfiguracja i diagnostyka · Backup i odtwarzanie · Utrzymanie klastra: repair, compaction, czyszczenie

Źródła: dokumentacja Apache Cassandra, rozdz. „Configuration" · dokumentacja Cassandra Reaper · notatki własne z eksploatacji klastrów produkcyjnych

---

Revision #42
Created 2026-09-24 23:35:01 UTC by Tomasz Konieczny
Updated 2026-10-26 14:13:23 UTC by Tomasz Konieczny