Replikacja danych między hostami: DRBD, GlusterFS, Ceph, ZFS i replikacja w aplikacji

Poziom: średnio

Dotyczy: DRBD 9.3, GlusterFS 11, Ceph 20 (Tentacle), OpenZFS 2.4, rsync, replikacja baz, Rocky Linux 9

Czas: ~75 min

Kiedy dwa hosty mają trzymać te same dane, wybieramy, na jakim poziomie je kopiujemy: pliki, bloki dysku albo wewnątrz aplikacji. Na tej stronie porównujemy DRBD, GlusterFS, Ceph, replikację ZFS, rsync i replikację natywną baz, a na końcu uzasadniamy wybór dla Zielonej. Wracamy tu, gdy ktoś proponuje „dysk, który sam kopiuje się na drugi serwer”.

ZAŁOŻENIA SCENARIUSZA

replikacja idzie siecią serwerową VLAN 10 między dwiema maszynami `docker01` i `docker02` na `hv01` i `hv02`, w jednej serwerowni i bez drugiej lokalizacji. Trzeci głos (tiebreaker) ma stać na fizycznym `bkp01`, poza klastrem hostów.

Skróty użyte na tej stronie (21)

Skrót	Co znaczy
CDDL	Common Development and Distribution License: licencja ZFS, niezgodna z GPL, przez co moduł ZFS dla Linuksa żyje poza jądrem
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
DRBD	Distributed Replicated Block Device: replikowane urządzenie blokowe w jądrze Linuksa, w którym każdy zapis trafia na dwa węzły
ELRepo	Enterprise Linux Repository: repozytorium społeczności z dodatkowymi jądrami i sterownikami dla RHEL i jego pochodnych, na przykład modułami DRBD

Skrót	Co znaczy
etcd	etc distributed: rozproszony magazyn par klucz-wartość, w którym Kubernetes trzyma cały stan klastra, a zmiany uzgadnia protokołem Raft
GPL	General Public License: otwarta licencja oprogramowania, na której opierają się MySQL Community i MariaDB
HA	High Availability: wysoka dostępność, czyli działanie mimo awarii jednego elementu
NFS	Network File System: udostępnianie katalogów po sieci w świecie Linuksa
OSD	Object Storage Daemon: demon Ceph obsługujący jeden dysk z danymi
RBD	RADOS Block Device: urządzenie blokowe udostępniane przez Ceph
RPO	Recovery Point Objective: ile danych wolno stracić przy awarii
RTO	Recovery Time Objective: jak długo wolno nie działać przy awarii
S2D	Storage Spaces Direct: programowa macierz Windows Server z dysków wielu serwerów
SPOF	Single Point of Failure: element, którego awaria zatrzymuje całą usługę, bo nie ma dla niego zamiennika
SQLite	mała baza danych zapisana w jednym pliku; Home Assistant trzyma w niej historię w pliku home-assistant_v2.db
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia
VLAN	Virtual LAN: wydzielona logicznie sieć na tym samym sprzęcie
WAL	Write-Ahead Log: dziennik PostgreSQL, zapis zmian przed zapisem danych
XFS	system plików stworzony przez Silicon Graphics, domyślny w Rocky Linux i RHEL
ZFS	system plików z migawkami i replikacją przyrostową zbudowany w jednym narzędziu z menedżerem wolumenów (dawniej Zettabyte File System)

Fiszka

Komendy tylko odczytują stan i wykonujemy je z CLI bash z uprawnieniami root (sudo). Techniki z tej strony poza rsync nie działają dziś w Zielonej, więc przykłady podajemy po to, żeby rozpoznać stan systemu, gdy ktoś je wdroży albo gdy trafimy na taki system w innej firmie. Nazwy `NazwaZasobu`, `NazwaPuli` i `NazwaWoluminu` zastępujemy nazwami z danego systemu.

□□ TYLKO ODCZYT

Sprawdzenie stanu replikacji DRBD: drbdadm status

```
sudo drbdadm status
```

Pokazuje rolę każdego węzła (Primary albo Secondary) i stan dysku. Zdrowa replikacja to `UpToDate` po obu stronach, a `Inconsistent` oznacza trwającą synchronizację, `Outdated` albo `StandAlone` zasługuje na zatrzymanie i przyjrzenie się.

Przykład z hipotetycznego zasobu `dane` między `docker01` i `docker02`:

```
dane role:Primary
  disk:UpToDate
  docker02 role:Secondary
    peer-disk:UpToDate
```

DRBD = Distributed Replicated Block Device: replikowane urządzenie blokowe w jądrze Linuksa, w którym każdy zapis trafia na dwa węzły · **drbdadm** = administracja DRBD: pokazuje i zmienia stan zasobów replikacji

ZMIENIA SYSTEM

Sprawdzenie migawek ZFS i ich wieku: zfs list -t snapshot

```
sudo zfs list -t snapshot -o name,used,creation -r NazwaPuli/docker
```

Pokazuje migawki zbioru razem z datą utworzenia. Brak świeżej migawki oznacza, że replikacja nie ma z czego wysłać przyrostu, a po stronie odbiorcy ta sama komenda pokazuje, co dotarło.

ZFS = system plików z migawkami i replikacją przyrostową zbudowany w jednym narzędziu z menedżerem wolumenów (dawniej Zettabyte File System) · **zfs** = zarządzanie zbiorami danych ZFS: migawki, wysyłka (send) i odbiór (receive) replikacji

TYLKO ODCZYT

Sprawdzenie rozjazdu w GlusterFS: gluster volume heal info split-brain

```
sudo gluster volume heal NazwaWoluminu info split-brain
```

Wypisuje pliki, które na kopiach mają sprzeczną zawartość, i liczbę wpisów dla każdego brika. Zero wpisów to stan prawidłowy.

Przykład:

```
Brick docker01:/gluster/brick1
Status: Connected
Number of entries in split-brain: 0
```

gluster = administracja GlusterFS: woluminy, brik-i i leczenie rozjazdów (heal)

TYLKO ODCZYT

Sprawdzenie stanu klastra Ceph: ceph -s

```
sudo ceph -s
```

Pokazuje ogólny stan (`HEALTH_OK`, `HEALTH_WARN`, `HEALTH_ERR`), liczbę monitorów w kworum, liczbę OSD i stan grup umieszczania (PG). Szukamy `quorum` z większością monitorów i `active+clean` przy wszystkich PG.

OSD = Object Storage Daemon: demon Ceph obsługujący jeden dysk z danymi ·**ceph** = administracja klastrem Ceph, na przykład ceph -s pokazuje jego stan

TYLKO ODCZYT

Sprawdzenie wieku danych na rezerwie rsync: tail /var/log/ft-ha.log

```
sudo tail -n 5 /var/log/ft-ha.log
```

W Zielonej to jedyna komenda z tej listy, która opisuje działającą replikację: skrypt ze strony Przenoszenie kontenerów z docker01 na docker02: zimna i ciepła rezerwa, przełączenie wpisuje tam każdą synchronizację. Brak wpisu z ostatnich 15 minut oznacza, że timer nie działa albo host nie trzyma adresu wirtualnego.

tail = pokaż koniec pliku, z flagą -f śledzi plik na żywo

Z życia wzięte

Z ŻYCIA WZIĘTE

Środa, 14:00. Ola rysuje na tablicy dwa prostokąty, `docker01` i `docker02`, i strzałkę między nimi z podpisem „replikacja”. Tomek pyta, co się stanie, gdy ktoś na `docker01` wykona `rm -rf /docker/apps/gitea/data`. Ola kreśli drugi krzyżyk przy `docker02` i mówi, że to samo, tylko

> po kilku sekundach.

Replikacja kopiuje także pomyłki, więc nie zastępuje kopii zapasowej, tylko ją uzupełnia. Gdy Ola dopisuje na tablicy strzałkę „restic na bkp01”, zostaje jeszcze pytanie, w jaki sposób zapis ma trafiać na drugi host. Na to odpowiada ta strona.

Jak to działa

Rozwiń: jak to działa pod spodem

Trzy pytania przed wyborem

Technikę wybieramy po odpowiedzi na trzy pytania. Po pierwsze, co kopiujemy: pliki, bloki dysku czy rekordy w bazie. Po drugie, kiedy zapis uznajemy za zakończony: po zapisie lokalnym czy po potwierdzeniu z drugiego węzła. Po trzecie, kto rozstrzyga, który węzeł ma rację, gdy sieć między nimi zniknie.

> **Pojęcie: replikacja synchroniczna i asynchroniczna.** W synchronicznej aplikacja dostaje potwierdzenie zapisu dopiero po tym, jak dane trafiły na oba węzły, więc po awarii nie tracimy nic, ale każdy zapis jest wolniejszy o opóźnienie sieci. W asynchronicznej potwierdzenie przychodzi po zapisie lokalnym, a kopia dociera do drugiego węzła chwilę później, więc po awarii tracimy to, co nie zdążyło dotrzeć.

> **Pojęcie: replikacja a kopia zapasowa.** Replika odzwierciedla bieżący stan źródła, więc razem z danymi przenosi skasowanie, ransomware i błąd logiczny. Kopia zapasowa trzyma stan z przeszłości, do którego można wrócić.

Zestawienie technik

Technika	Na jakim poziomie	Minimum	RPO	Koszt i złożoność	Główna pułapka

rsync z timerem	pliki	dwa hosty	okres timera plus czas kopiowania	niski, niska	`--delete` powiela skasowanie, baza w trakcie kopiowania jest niespójna
ZFS send/receive	migawki zbioru	dwa hosty, dyski pod ZFS	częstość migawek (minuty)	średni, średnia	moduł jądra spoza dystrybucji, `receive -F` kasuje to, czego nie ma w nadawcy
DRBD	bloki urządzenia	dwa węzły, trzeci głos dla kworum	zero (protokół C) albo sekundy (A)	średni, wysoka	split brain, system plików tylko na jednym węźle naraz, pakiety spoza dystrybucji
GlusterFS	pliki w rozproszonym systemie	trzy węzły albo dwa i arbiter	zero (replica 3)	wysoki, wysoka	projekt wygasa, rozjazd kopii przy replica 2
Ceph	obiekty, bloki (RBD), pliki (CephFS)	trzy węzły, sieć 10 Gb	zero	bardzo wysoki, bardzo wysoka	wymagania sprzętowe, brak Rocky 9 w tabelach wsparcia Tentacle
Replikacja w aplikacji (bazy)	rekordy lub strumień logu	dwa węzły i zasady przełączenia	sekundy lub zero	średni, średnia	awans repliki bez kworum albo arbitra grozi dwoma serwerami głównymi

rsync z timerem systemd

Rsync porównuje pliki na obu hostach i kopiuje różnice. W Zielonej jest podstawą ciepłej rezerwy z poprzedniej strony, a jego atutem jest to, że znamy go wszyscy i nie wymaga niczego poza SSH. Ceną jest rozmiar okna: dane na `docker02` są tak stare jak ostatnia synchronizacja, a baza zapisująca pliki w trakcie kopii daje niespójny obraz.

Dlatego aplikacje z bazą w plikach (SQLite, katalog danych PostgreSQL w kontenerze) kopiujemy w trybie z zatrzymaniem albo bazę wyprowadzamy z kontenera. Pełny przykład z kontrolą kierunku i zabezpieczeniem przed pustym `/docker` jest na stronie Przenoszenie

kontenerów z docker01 na docker02: zimna i ciepła rezerwa, przełączenie.

ZFS send/receive

> **Pojęcie: migawka (snapshot).** Niezmienny obraz zbioru danych z jednej chwili. W ZFS powstaje w ułamku sekundy i na początku nie zajmuje miejsca, bo zapamiętuje tylko późniejsze zmiany.

Replikacja ZFS wysyła migawkę przez `zfs send` i odbiera ją przez `zfs receive`. Pierwsza wysyłka to cały zbiór, kolejne mogą być przyrostowe: opcja `-i` wysyła różnicę między dwiema migawkami, a `-I` także wszystkie pośrednie [1]. Odbiorca dostaje spójny obraz z jednej chwili, więc baza zapisana w tej samej migawce jest odtwarzalna tak samo jak po nagłym wyłączeniu zasilania.

```
sudo zfs snapshot NazwaPuli/docker@ha-$(date +%F-%H%M)
sudo zfs send -i NazwaPuli/docker@PoprzedniaMigawka NazwaPuli/docker@NowaMigawka \
  | ssh nazwa_usera@docker02 sudo zfs receive -s -u NazwaPuli/docker
```

Opcji `-F` po stronie odbiorcy świadomie nie używamy: wycofuje ona zbiór odbiorcy do jego ostatniej migawki przed odbiorem, a przy strumieniu replikacyjnym usuwa to, czego nie ma w nadawcy [1]. Opcja `-s` pozwala wznowić przerwany transfer z tokenu, a `-u` nie montuje odebranego zbioru, żeby pozostał nieaktywny do chwili przełączenia [1].

Cena ZFS jest organizacyjna. Dyski rezerwy i aktywnego hosta muszą być pod ZFS, więc to zmiana układu `vgDocker/lvDocker` na pulę ZFS. Moduł jądra pochodzi spoza dystrybucji (repozytorium `zfs-release-3-0` obejmuje EL8 do EL10) i po aktualizacji jądra musimy sprawdzić, czy się zbudował [2]. Licencja CDDL nie jest zgodna z GPL, dlatego ZFS na Linuksie żyje poza jądrem, co sprawdzamy w dokumentacji OpenZFS przed wdrożeniem [2].

Wariant ZFS dla Zielonej opisujemy jako szkic na poprzedniej stronie.

DRBD

> **Pojęcie: DRBD (Distributed Replicated Block Device).** Replikowane urządzenie blokowe w jądrze Linuksa. Każdy zapis na urządzenie na jednym węźle trafia też do drugiego, więc oba mają te same bloki, a system plików montujemy tylko na jednym, aktywnym (Primary).

DRBD ma trzy protokoły. W protokole A zapis jest potwierdzony po zapisie lokalnym i wysłaniu do bufora TCP, w B po dotarciu pakietu do partnera, a w C po zapisie na obu dyskach, czyli synchronicznie [3]. Dla danych, których strata boli, wybieramy C i płacimy opóźnieniem sieci przy każdym zapisie.

Dwa fakty decydują o przydatności DRBD w Zielonej. Pierwszy to kworum: dwa węzły nie wystarczą, więc DRBD 9 pozwala dodać trzeci, bezdyskowy węzeł (tiebreaker), który tylko głosuje [3]. Drugi to tryb dual-primary: DRBD samo nie zamienia zwykłego XFS albo ext4 w system aktywny-aktywny, bo wymaga to klastrowego systemu plików z rozproszonym menedżerem blokad [3].

Wersja 9.3.4 pochodzi z 15 września 2026, a pakiety wstępnie zbudowane dostajemy w subskrypcji LINBIT; ścieżkę bez subskrypcji (ELRepo albo kompilacja) trzeba sprawdzić, bo w tej sesji nie zdołaliśmy jej potwierdzić [3]. Moduł jądra jest poza dystrybucją, tak jak przy ZFS, więc aktualizacje jądra wymagają planu. Replikacja bloków przenosi także uszkodzony system plików, a przełączenie to dwie czynności więcej niż przy rsync: promocja (Secondary na Primary) i montowanie.

GlusterFS

> **Pojęcie: GlusterFS.** Rozproszony system plików, w którym pliki są kopiowane na kilka serwerów (brików) i widoczne jako jeden wolumin. Zwykle używa się replica 3 albo replica 2 z arbitrem, który trzyma tylko metadane.

Dla nowych wdrożeń go nie rekomendujemy. Red Hat zakończył wsparcie dla Gluster Storage 31 grudnia 2024 roku, liczba zmian w repozytorium spadła z 78 w 2023 roku do 31 w 2024 roku, a CentOS Storage SIG nie buduje pakietów dla Stream 10 [4]. Ostatnie wydanie, które znaleźliśmy, to 11.2, a jego rok trzeba sprawdzić. Replica 2 bez arbitra jest podatna na split brain, a przy zaostrzeniu kworum traci dostępność [4].

Ceph

> **Pojęcie: Ceph.** Rozproszony magazyn obiektów, bloków (RBD) i plików (CephFS), który sam wykrywa awarie i odtwarza brakujące kopie. Składa się z monitorów, które trzymają mapę klastra i głosują, oraz z demonów OSD, po jednym na każdy dysk.

Ceph jest najlepszy z opisanych technik i najdroższy w utrzymaniu. Potrzebuje co najmniej trzech węzłów i trzech monitorów, a dokumentacja zaleca sieć 10 Gb i osobny dysk na każdy

OSD [5]. Gałąź Tentacle (20.2.4) ma wsparcie do 1 czerwca 2027, Squid (19.2.6) do 31 października 2026, czyli za miesiąc, a tabele systemów dla Tentacle wymieniają CentOS 9 i Rocky 10, ale nie Rocky 9 [5]. Dla firmy z dwoma hostami Hyper-V, na których trzy maszyny wirtualne i tak tracą kworum przy awarii hosta, Ceph jest więc przerostem.

Współdzielony NFS

Dwa hosty montujące jeden udział NFS wyglądają jak najprostsza gorąca rezerwa, ale serwer NFS sam staje się punktem awarii (SPOF). Bazy i SQLite na NFS są ryzykowne ze względu na blokady plików. Używamy NFS do danych, które i tak są tylko do odczytu albo do kopii, a nie jako podkładu pod aplikacje z zapisem.

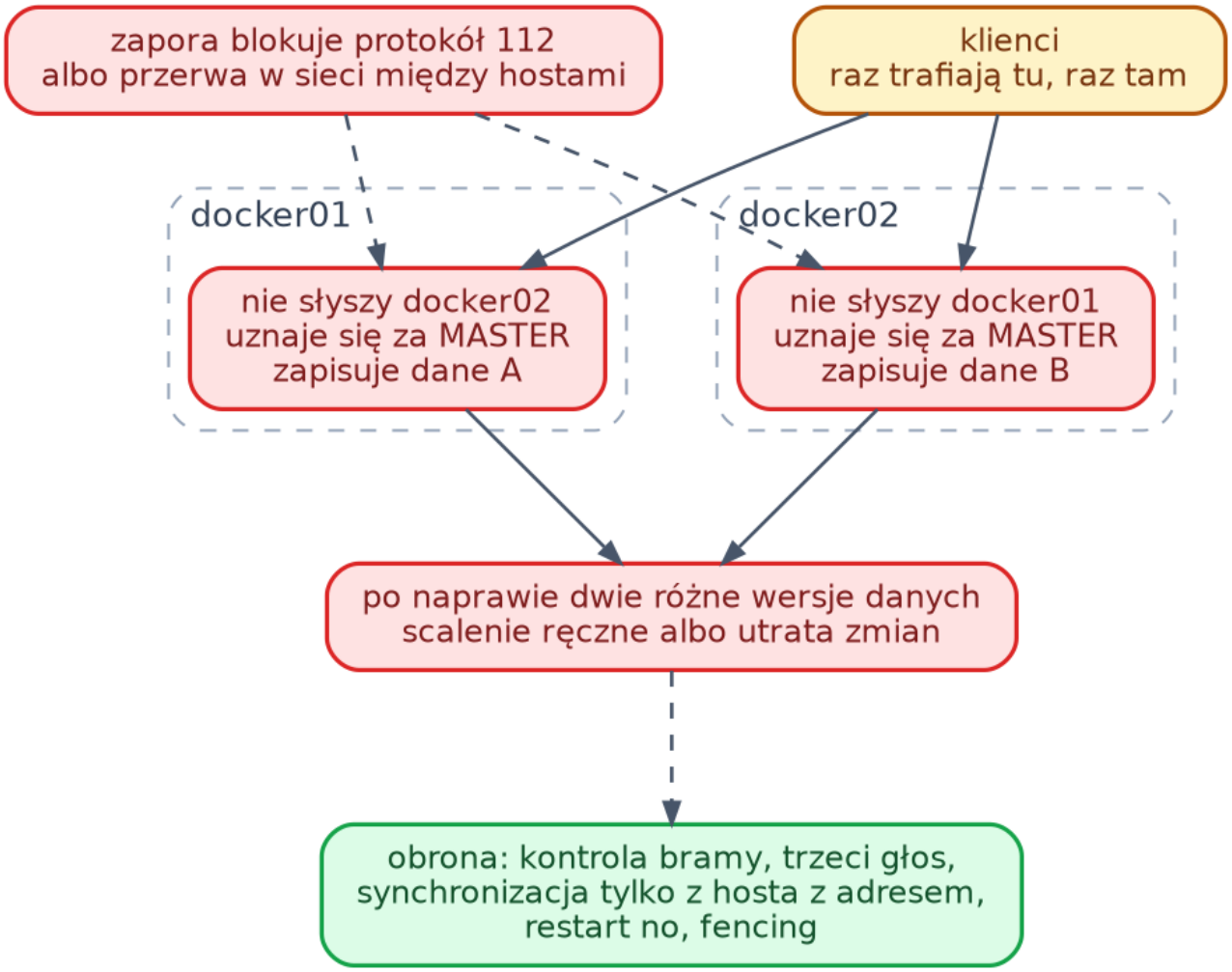
Replikacja w samej aplikacji

Najlepszy mechanizm gorącej rezerwy dla danych strukturalnych to ten wbudowany w silnik. PostgreSQL replikuje strumień dziennika WAL do repliki (piszemy o tym w Replikacja strumieniowa), baza MongoDB ma zestawy replik, Cassandra trzyma kilka kopii (u nas współczynnik 3 na `cass01` do `cass03`), a Active Directory replikuje się natywnie między `dc01` i `dc02`. Silnik wie, które zapisy są potwierdzone, więc nie ma problemu niespójnego obrazu pliku.

Dla Zielonej to oznacza, że gorąca rezerwa danych jest przede wszystkim sprawą baz, a nie systemu plików. Gitea trzyma repozytoria w plikach i metadane w PostgreSQL na `pg01`, więc replika `pg01` załatwia drugą połowę.

Pliki idą rsync-iem. Awans repliki PostgreSQL (`pg_ctl promote`) wykonuje ręcznie człowiek albo narzędzie z zewnątrz, a różnicę między planowanym przełączeniem i awaryjnym opisuje strona o replikacji strumieniowej [6]. Dla automatycznego przełączenia potrzebujemy kworum albo arbitra, o co pytamy w przebiegu z macierzą HA.

Split brain w każdej technice



Rysunek 1. Split brain w dwóch hostach. Po zerwaniu łączności oba uznają się za aktywne i przyjmują zapisy, a po jej przywróceniu każdy ma zmiany, których nie ma drugi. Ratuje trzeci głos albo ręczna decyzja, który zbiór danych zostaje. (kliknięcie otwiera pełny rozmiar)

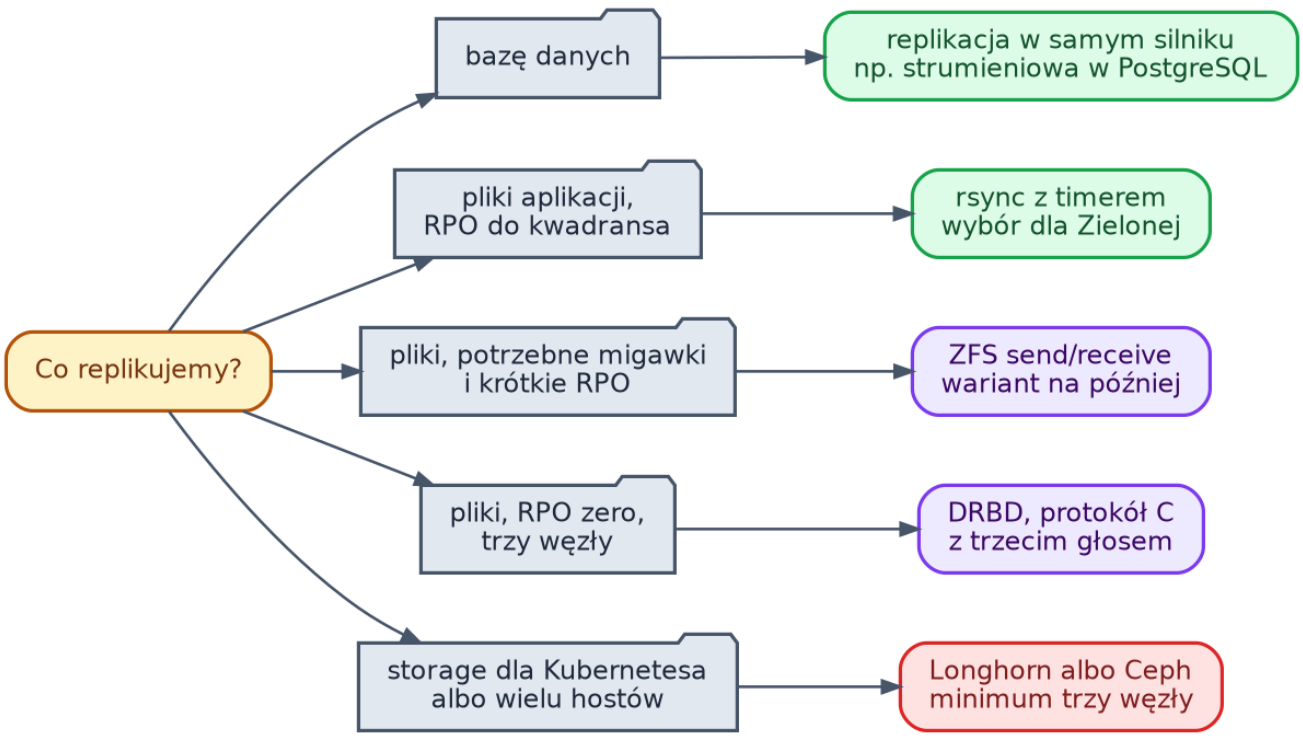
Każda technika ma swój sposób na rozjazd i swój koszt jego naprawy. Tabela pokazuje, co dzieje się po zerwaniu sieci i jak zwykle wracamy do jednego zbioru.

Technika	Co się psuje po zerwaniu sieci	Jak wracamy do porządku
rsync z VIP	oba hosty trzymają adres wirtualny i oba zapisują	decyzja człowieka, którą stronę zostawiamy, potem synchronizacja z wybranej strony
ZFS	zbiór po stronie odbiorcy ma migawki, których nie ma nadawca	rozstrzygamy, którą historię zostawiamy, a `zfs receive -F` wycofuje zbiór odbiorcy [1]

Technika	Co się psuje po zerwaniu sieci	Jak wracamy do porządku
DRBD	dwa węzły Primary z rozbieżnymi blokami	polityki automatycznej naprawy albo ręczny wybór ofiary, której zmiany tracimy [3]
GlusterFS	pliki z różną zawartością na kopiach	`gluster volume heal` z polityką wyboru źródła [4]
Ceph	mniejszość monitorów traci kworum i przestaje działać	po powrocie łączności mniejszość dołącza do większości, bo przy braku kworum nie przyjmowała zapisów [5]
Baza natywna	dwa węzły uznają się za główne	arbiter, kworum, ręczne `pg_rewind` albo odbudowa repliki

Najlepszą obroną jest trzeci głos albo bezpiecznik w skryptach. W Zielonej drugi host zabezpieczamy skryptem synchronizacji, który nie nadpisuje danych, gdy partner też trzyma adres wirtualny, a decyzję o powrocie starego hosta podejmuje człowiek.

Wybór dla Zielonej



Rysunek 2. Drzewo decyzji: od typu danych (baza czy pliki), przez liczbę węzłów i wymagane RPO, do techniki. Dla Zielonej kończy się na bazach z replikacją natywną i na plikach z rsync albo ZFS. (kliknięcie otwiera pełny rozmiar)

Rekomendacja dla Zielonej składa się z czterech decyzji i jednej zasady. Zasada brzmi: replikacja zawsze obok kopii, nigdy zamiast niej.

1. Bazy replikujemy natywnie. Dla `pg01` oznacza to repliki strumieniowe i sprawdzenie, czy potrzebujemy automatycznego przełączenia. Baza w kontenerze na `docker01` (NetBox) trafia na `pg01` albo zostaje na zimnej rezerwie.
2. Pliki aplikacji z `docker01` kopiujemy rsync-iem z timerem co 15 minut na `docker02` i dodatkowo restic o 02:30 na `bkp01`. Daje to RPO rzędu kwadransa i RTO rzędu kilku minut.
3. ZFS send/receive jest wariantem, który wybieramy, jeśli RPO ma spaść do kilku minut albo jeśli zależy nam na spójnych migawkach. Wymaga przebudowy układu dysków i modułu jądra, więc nie wdrażamy go na początek.
4. DRBD bierzemy pod uwagę tylko z trzecim głosem, na przykład tiebreakerem na `bkp01`, co wymaga sprawdzenia możliwości instalacji tam modułu. GlusterFS odrzucamy, Ceph zostawiamy na sytuację z co najmniej trzema fizycznymi hostami.

Uzasadnienie ma cztery części.

Zielona ma dwa hosty Hyper-V i dwóch administratorów, a każda technika z trzema węzłami zaczyna od tego, że kworum nie przeżyje awarii hosta z dwoma węzłami. S2D pod spodem już wykonuje replikację dysków (dwie kopie), więc replikacja bloków nad nim dubluje ochronę, a nie dodaje nowej. Zakładamy, że ilość danych aplikacji na `docker01` jest niewielka (sprawdzamy to niżej), a RPO rzędu kwadransa dla Gitea, Vaultwardena i poczty jest do zaakceptowania, co potwierdzamy z właścicielami usług. Ostatnia część to prostota: rsync i restic znamy, a przy stresie w piątek o 20:00 działa to, co znamy.

Wielkość danych sprawdzamy komendą `sudo du -sh /docker/apps/*` na `docker01`, bo od niej zależy czas synchronizacji. Jeśli któraś aplikacja urośnie tak, że synchronizacja nie mieści się w kwadransie, przesuwamy ją na ZFS albo wynosimy dane na `pg01`.

Kiedy zmieniamy rekomendację

Rekomendację zmieniamy, gdy zmieni się któryś z założeń. RPO poniżej minuty dla plików wskazuje na DRBD z trzecim głosem albo ZFS z częstymi migawkami. Trzeci fizyczny host pozwala na Ceph albo k3s z Longhornem, czemu poświęcimy osobne strony. Potrzeba aktywnego-aktywnego dla aplikacji przerzuca sprawę do samej aplikacji, bo żaden system plików tego za nią nie załatwi.

Częste błędy

Błąd	Skutek	Jak unikamy
Replikacja zamiast kopii zapasowej	skasowanie lub ransomware dociera do drugiego hosta	restic na `bkp01` zostaje na każdym poziomie
DRBD na dwóch węzłach bez trzeciego głosu	po zerwaniu sieci oba węzły mogą stać się Primary	tiebreaker, kworum i polityki, nie wybór „na oko”
Zwykły XFS albo ext4 zamontowany na obu węzłach DRBD	uszkodzenie systemu plików	system plików tylko na Primary, dual-primary tylko z klastrowym systemem plików
`zfs receive -F` jako domyślny nawyk	kasuje po stronie odbiorcy dane, których nie ma w nadawcy	pierwsza wysyłka bez `-F`, `-F` dopiero po decyzji i sprawdzeniu
Replikacja żywej bazy rsync-iem	niespójny obraz bazy po awarii	replikacja natywna albo zrzut, albo rsync z zatrzymaniem aplikacji
GlusterFS replica 2 bez arbitra	rozjazd kopii przy zerwaniu sieci	jeśli już działa, replica 3 albo arbiter, a w nowym wdrożeniu inna technika
Ceph z trzema węzłami na dwóch hostach Hyper-V	awaria hosta zabiera kworum monitorów	trzy fizyczne hosty albo inna technika
Brak alarmu na wiek danych rezerwy	po awarii okazuje się, że rezerwa ma dane sprzed tygodnia	Zabbix sprawdza ostatni wpis w `/var/log/ft-ha.log` albo stan timera

□□ Ćwiczenia

1. Dlaczego przy DRBD w protokole C po awarii nie tracimy danych, a przy protokole A możemy?

Odpowiedź

W protokole C potwierdzenie zapisu wraca do aplikacji dopiero po zapisie na dysku lokalnym i zdalnym, więc każdy potwierdzony zapis jest na obu węzłach. W protokole A potwierdzenie przychodzi po zapisie lokalnym i wysłaniu do bufora, więc przy awarii węzła główne dane, które nie zdążyły dotrzeć do partnera, giną. C płaci za to opóźnieniem sieci przy każdym zapisie.

1. Na `docker02` po awarii sieci okazuje się, że zbiór ZFS ma migawki, których nie ma `docker01`. Co sprawdzamy przed `zfs receive -F`?

Odpowiedź

Sprawdzamy, które migawki są nowsze i czy zawierają zapisy, które chcemy zachować, bo `-F` zniszczy po stronie odbiorcy to, czego nie ma w nadawcy. Jeśli prawdziwe zmiany użytkowników trafiły na `docker02`, kierunek replikacji odwracamy albo zachowujemy migawki jako klony przed

odbiorem.

1. Zielona ma dwa hosty Hyper-V. Dlaczego trzy maszyny wirtualne z Cepha albo etcd nie dają odporności na awarię hosta?

Odpowiedź

Trzy węzły na dwóch hostach oznaczają, że jeden host trzyma co najmniej dwa węzły. Awaria tego hosta zabiera większość głosów (dwa z trzech), więc klaster traci kworum i staje. Odporność na awarię hosta dają trzy węzły na trzech niezależnych hostach.

1. Dla której z aplikacji z `docker01` mamy najlepszą gorącą rezerwę danych, jeśli replikujemy tylko `pg01`?

Odpowiedź

Dla tych, których dane leżą wyłącznie w PostgreSQL na `pg01`. Gitea ma tam metadane, ale repozytoria leżą w plikach w `/docker/apps/gitea/data`, więc replika `pg01` załatwia tylko połowę. Pozostałą część chronimy rsync-iem i restic, a pełna gorąca rezerwa wymagałaby także replikacji plików.

Powiązane: Wysoka dostępność: usługa, która nie ma jednego serwera · Wysoka dostępność: pojęcia, liczenie dostępności i poziomy rezerwy · Przenoszenie kontenerów z docker01 na docker02: zimna i ciepła rezerwa, przełączenie · Replikacja: synchroniczna, asynchroniczna, logiczna, fizyczna · Replikacja strumieniowa · Kopia całego docker01: /docker/apps, /docker/compose i bazy w kontenerach

Źródła:

1. openzfs.github.io: zfs-send(8) i zfs-receive(8)
2. openzfs.github.io: Getting Started, RHEL-based distro; wydania na GitHubie
3. LINBIT: DRBD 9 User Guide, repozytorium DRBD i strona pobierania
4. docs.gluster.org: Arbiter volumes and quorum; Phoronix o wycofaniu GlusterFS z Fedory; Red Hat o zakończeniu wsparcia
5. docs.ceph.com: Releases, Add or remove monitors, OS recommendations, Hardware recommendations
6. Ta baza wiedzy: Replikacja strumieniowa, sekcje o promocji repliki, failover i split-brain

Revision #15
Created 2026-10-03 12:08:45 UTC by Tomasz Konieczny
Updated 2026-10-26 14:21:08 UTC by Tomasz Konieczny