MailCow: kopia zapasowa i aktualizacja

◫◫ Poziom: operacyjny

◫◫ Dotyczy: MailCow Dockerized 2026-07a na docker01, skrypty backup_and_restore.sh i update.sh (sprawdzono w kodzie skryptów i w dokumentacji 3.10.2026)

⏱ Czas: ~45 min

MailCow ma własny skrypt do kopii zapasowej i własny skrypt do aktualizacji, i właśnie nimi się posługujemy, zamiast kopiować pliki ręcznie. Ta strona mówi, co dokładnie kopiuje skrypt, czego nie kopiuje, jak odtwarzamy pocztę na maszynie testowej oraz kiedy w Zielonej nie wolno aktualizować. Wracamy tu przed każdą aktualizacją i raz na kwartał przy teście odtwarzania.

◫◫ Skróty użyte na tej stronie (8)

Skrót	Co znaczy
ARC	Authenticated Received Chain: mechanizm, w którym serwer pośredniczący pieczętuje wyniki SPF i DKIM przy przekazywaniu wiadomości, żeby kolejny serwer mógł im zaufać; w panelu MailCow stoi obok kluczy DKIM
DKIM	DomainKeys Identified Mail: cyfrowy podpis wiadomości e-mail
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
GiB	gibibajt: 1024 MiB, czyli około 1,07 GB; tej jednostki używa dokumentacja MailCow przy wymaganiach pamięci
ICMP	Internet Control Message Protocol: protokół diagnostyczny sieci, którym działa między innymi ping
RAM	Random Access Memory: pamięć operacyjna
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Fiszka

Komendy wykonujemy na `docker01` z konta z uprawnieniami sudo. Katalog instalacji MailCow to `/docker/apps/mailcow`, a lokalne kopie trzymamy w `/docker/kopie/mailcow`.

◫ OSTROŻNIE · NIEODWRACALNE

Wykonanie kopii wszystkich składników: backup_and_restore.sh backup all

```
cd /docker/apps/mailcow
sudo env MAILCOW_BACKUP_LOCATION=/docker/kopie/mailcow THREADS=2 ./helper-
scripts/backup_and_restore.sh backup all --delete-days 3
```

Zmienna `MAILCOW_BACKUP_LOCATION` wskazuje katalog na kopie, a bez niej skrypt pyta o niego interaktywnie, co w zadaniu nocnym by go zawiesiło. `THREADS` to liczba wątków kompresji (dokumentacja radzi zostawić MailCow co najmniej dwa rdzenie wolne), a `--delete-days 3` usuwa lokalne kopie starsze niż trzy dni. Skryptu nie kopiujemy w inne miejsce, bo szuka plików względem swojego katalogu.

Przykład (początek wyniku):

```
Using 2 Thread(s) for this run.
Using /docker/kopie/mailcow as backup/restore location.

Found project name mailcowdockerized
Checking Docker image: ghcr.io/mailcow/backup:latest
Using SQL image mariadb:10.11, starting...
```

◫ **cd** = change directory: przejdź do katalogu · **env** = environment: uruchamia program z dodatkowymi zmiennymi środowiskowymi, na przykład sudo env ZMIENNA=wartość polecenie

◫ TYLKO ODCZYT

Sprawdzenie, co powstało: ls

```
ls -1A /docker/kopie/mailcow/mailcow-*/
```

Każda kopia to osobny katalog z datą w nazwie, w formacie `mailcow-RRRR-MM-DD-GG-MM-SS`. Nazw tych katalogów nie zmieniamy, bo po nich skrypt rozpoznaje kopie przy odtwarzaniu.

Przykład (zawartość jednego katalogu):

```
.x86_64
backup_crypt.tar.zst
backup_mariadb.tar.zst
```

```
backup_postfix.tar.zst
backup_redis.tar.zst
backup_rspamd.tar.zst
backup_vmail.tar.zst
mailcow.conf
```

ls = list: wypisz zawartość katalogu

TYLKO ODCZYT

Wysłanie kopii na bkp01: rsync

```
rsync -rt -e ssh /docker/kopie/mailcow/ nazwa_usera@bkp01:/backup/mailcow/
```

Wysyłamy lokalne kopie na `bkp01`. Bez przełącznika `--delete` kopie zostają tam dłużej niż lokalne trzy dni.

rsync = remote sync: kopiuje pliki i katalogi lokalnie albo przez SSH, przesyłając tylko to, co się zmieniło, tu migawki OpenSearch na nowy serwer

ZMIENIA SYSTEM

Odtworzenie z kopii: backup_and_restore.sh restore

```
cd /docker/apps/mailcow
sudo env MAILCOW_BACKUP_LOCATION=/docker/kopie/mailcow THREADS=2 ./helper-
scripts/backup_and_restore.sh restore
```

Skrypt wypisuje kopie z katalogu do wyboru („Select a restore point"), potem listę składników („Select a dataset to restore"), z pozycją `0` oznaczającą wszystko. Za jednym razem odtwarzamy jeden wybór, więc kilka składników odtwarzamy kolejnymi uruchomieniami. Odtwarzanie bazy zatrzymuje cały MailCow i zmienia wpisy bazy w bieżącym `mailcow.conf` na te z kopii, a odtwarzanie skrzynek zatrzymuje Dovecota. Tej komendy nie uruchamiamy na działającym środowisku produkcyjnym bez decyzji i bez świeżej kopii tego, co zastępujemy.

cd = change directory: przejdź do katalogu · **env** = environment: uruchamia program z dodatkowymi zmiennymi środowiskowymi, na przykład sudo env ZMIENNA=wartość polecenie

TYLKO ODCZYT

Wymuszenie ponownego odczytu skrzynek po odtworzeniu: doveadm force-resync

```
cd /docker/apps/mailcow
sudo docker compose exec dovecot-mailcow doveadm force-resync -A '*'
```

Po odtworzeniu skrzynek skrypt sam pyta, czy to zrobić („Force a resync now?"). Zwykle nie jest to potrzebne, ale wykonujemy to, gdy po odtworzeniu któraś skrzynka pokazuje błędy albo brakujące wiadomości.

cd = change directory: przejdź do katalogu · **docker** = narzędzie do kontenerów

ZMIENIA SYSTEM

Sprawdzenie, czy jest nowe wydanie: update.sh --check

```
cd /docker/apps/mailcow
sudo ./update.sh --check
```

Skrypt porównuje nasz kod z repozytorium i niczego nie zmienia. Kod wyjścia `0` oznacza, że aktualizacja jest dostępna, a `3`, że jej nie ma, co przydaje się w skryptach. Panel pokazuje to samo w `System → Information`, w komunikacie „There is an update available".

Przykład:

```
Checking remote code for updates...
Updated code is available.
The changes can be found here: https://github.com/mailcow/mailcow-
dockerized/commits/master
```

cd = change directory: przejdź do katalogu

ZMIENIA SYSTEM

Pobranie obrazów z wyprzedzeniem: update.sh --prefetch

```
cd /docker/apps/mailcow
sudo ./update.sh --prefetch
```

Pobiera nowe obrazy kontenerów i kończy pracę, nie zatrzymując poczty. Dzięki temu w oknie zmian nie czekamy na pobieranie, a przerwa w pracy poczty jest krótsza.

cd = change directory: przejdź do katalogu

ZMIENIA SYSTEM

Aktualizacja: update.sh

```
cd /docker/apps/mailcow
sudo ./update.sh
```

Skrypt pyta „Are you sure you want to update mailcow: dockerized? All containers will be stopped. [y/N]", więc przez czas aktualizacji poczta nie działa. Nie używamy przełącznika `--force` (`-f`), który wyłącza wszystkie pytania, ani `--ours`, który przy konflikcie faworyzuje nasze zmiany w plikach repozytorium. Przełącznika `--skip-ping-check` używamy tylko wtedy, gdy zapora blokuje ICMP z serwera, bo skrypt na początku sprawdza łączność pingiem do publicznych serwerów DNS.

ICMP = Internet Control Message Protocol: protokół diagnostyczny sieci, którym działa między innymi ping · **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **cd** = change directory: przejdź do katalogu

ZMIENIA SYSTEM

Sprzątnięcie starych obrazów po aktualizacji: update.sh --gc

```
cd /docker/apps/mailcow
sudo ./update.sh --gc
```

Usuwa stare znaczniki obrazów, które po aktualizacji zajmują miejsce na `/docker`. Wykonujemy to po sprawdzeniu, że nowa wersja działa, bo bez starych obrazów szybki powrót do poprzedniej wersji wymaga ponownego pobrania.

cd = change directory: przejdź do katalogu

ZMIENIA SYSTEM

Sprawdzenie punktów powrotu sprzed aktualizacji: git log

```
cd /docker/apps/mailcow
git log --pretty=oneline | grep -i "before update"
```

Każda aktualizacja zapisuje w repozytorium commit „Before update on RRRR-MM-DD_GG_MM_SS". Identyfikator takiego commita jest potrzebny do wycofania, patrz sekcja „Wycofanie aktualizacji".

cd = change directory: przejdź do katalogu · **git** = system kontroli wersji (nazwa to brytyjski slang: nieznośny typ)

Z życia wzięte

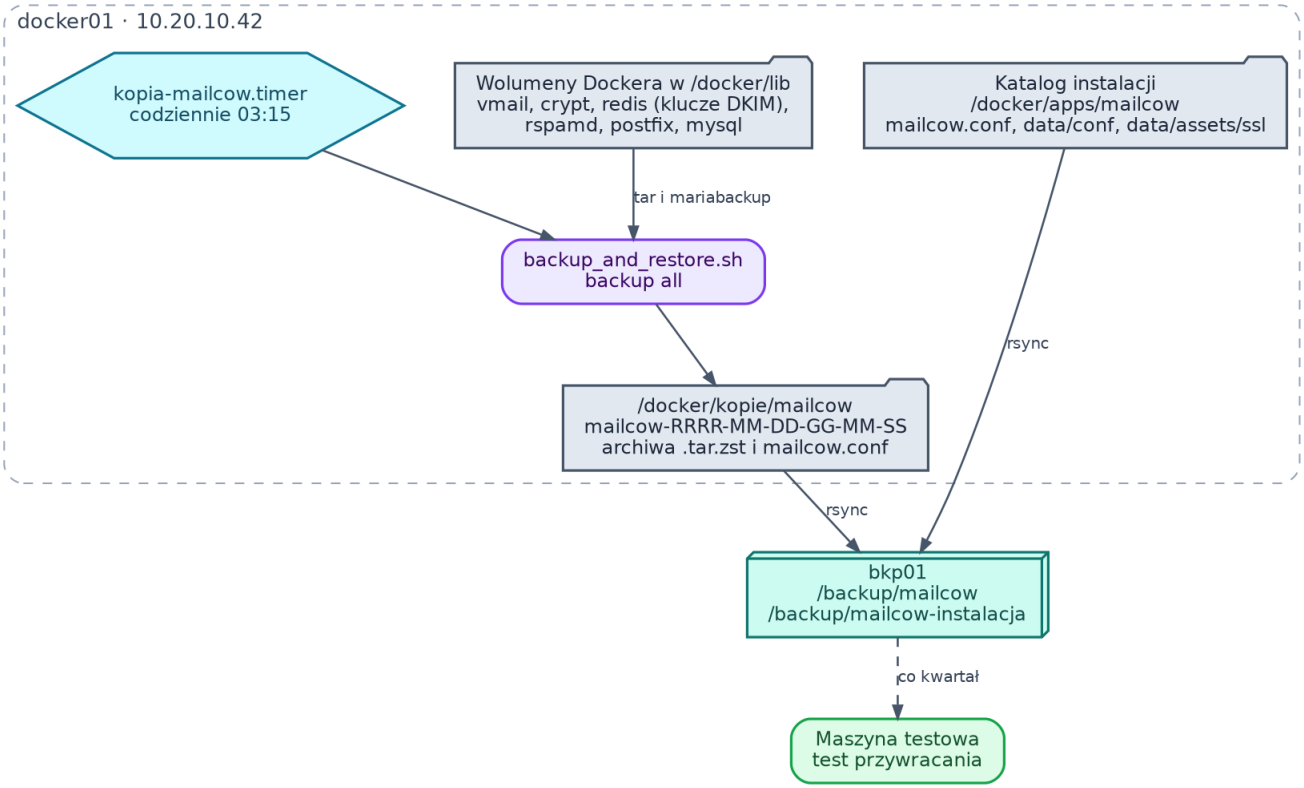
Z ŻYCIA WZIĘTE

Piątek 13 listopada, 15:40. Tomek widzi w panelu komunikat „There is an update available" i chce uruchomić `update.sh` jeszcze przed weekendem, bo zaczynają się święta sprzedażowe. Ola zatrzymuje go przy biurku: za dwadzieścia minut zaczyna się piątek po piętnastej, a od soboty obowiązuje zakaz zmian do 6 stycznia. Tomek uruchamia tylko `update.sh --check` i `update.sh --prefetch`, zapisuje w zgłoszeniu numer wydania i prosi pana Zbigniewa o decyzję, czy poprawka wymaga wyjątku od zakazu.

Jak to działa

Rozwiń: jak to działa pod spodem

Skrypt `backup_and_restore.sh` nie kopiuje katalogów, tylko wolumeny Dockera, w których MailCow trzyma dane. Na `docker01` leżą one pod `/docker/lib/volumes`, czyli na osobnym wolumenie `/docker`. Skrypt uruchamia pomocniczy kontener, wchodzi do niego z wolumenem tylko do odczytu i pakuje zawartość programem `tar` z kompresją zstd. Wyjątkiem jest baza: ta jest kopiowana narzędziem `mariabackup`, które daje spójny obraz bazy mimo pracy serwera.



Skrypt kopii pakuje wolumeny Dockera do katalogu z datą, a rsync wysyła je na bkp01 razem z katalogiem instalacji (kliknięcie otwiera pełny rozmiar)

Składnik (argument)	Co zawiera	Plik w kopii
`vmail`	wiadomości wszystkich skrzynek	`backup_vmail.tar.zst`
`crypt`	dane szyfrowania skrzynek (o ile włączone)	`backup_crypt.tar.zst`
`redis`	klucze DKIM, nauczone wzorce spamu, limity	`backup_redis.tar.zst`
`rspamd`	dane i ustawienia filtra antyspamowego	`backup_rspamd.tar.zst`
`postfix`	dane Postfixa, między innymi kolejka wiadomości	`backup_postfix.tar.zst`
`mysql`	baza MariaDB: domeny, skrzynki, aliasy, kwarantanna, kalendarze i kontakty SOGo	`backup_mariadb.tar.zst`
(zawsze)	bieżący plik `mailcow.conf`, w tym hasła do bazy	`mailcow.conf`

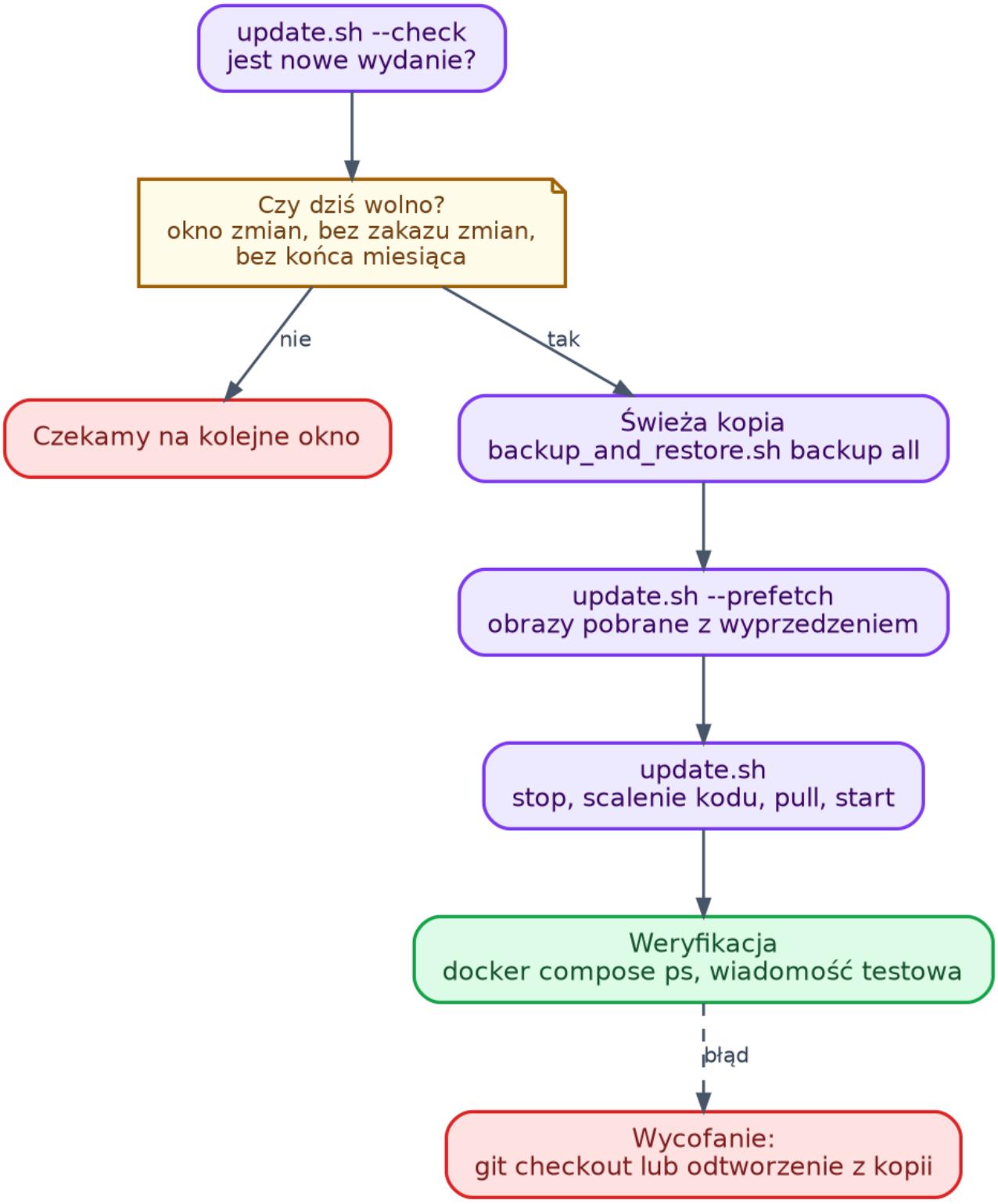
Są rzeczy, których skrypt nie kopiuje: plik `docker-compose.yml`, nasze zmiany w katalogu `data/conf`, certyfikaty z `data/assets/ssl` i plik `docker-compose.override.yml`, jeśli kiedyś go

założymy. Dlatego kopiujemy też katalog instalacji `/docker/apps/mailcow`, co jest zalecane w dokumentacji MailCow. Kopia całego `docker01` opisana na stronie Kopia całego docker01: /docker/apps, /docker/compose i bazy w kontenerach nie wymienia katalogu `/docker/lib`, więc nie liczymy, że obejmuje skrzynki.

Redis jest w tej kopii ważniejszy, niż wygląda. Tam leżą klucze prywatne DKIM, więc kopia bez składnika `redis` oznacza po awarii nowe klucze i nowe rekordy w DNS. Skrzynki kopiowane są z działającego serwera, więc wiadomość, która przyszła w trakcie pakowania, może się nie zmieścić w kopii. Pełna kopia jest zatem spójna co do bazy, a co do skrzynek jest dobra z dokładnością do ostatnich minut.

> □□ Nasze własne zmiany w konfiguracji MailCow trzymamy w plikach dodatkowych (`docker-compose.override.yml`, pliki w `data/conf/...` przewidziane na rozszerzenia), a nie w plikach, które aktualizacja nadpisuje. `update.sh` scala kod z repozytorium, a przy konflikcie domyślnie wygrywa wersja z repozytorium.

Aktualizacja wykonuje po kolei kilka rzeczy. Sprawdza łączność z internetem i to, czy ma nowszą wersję samego siebie, pyta o potwierdzenie, zapisuje różnice w naszych plikach do katalogu `update_diffs`, pobiera obrazy, zatrzymuje wszystkie kontenery, robi commit „Before update", scala kod z repozytorium, pobiera pozostałe obrazy i uruchamia stos od nowa. Na końcu porządkuje stare obrazy.



Kolejność przed aktualizacją: sprawdzenie, czy wolno, świeża kopia, pobranie obrazów, aktualizacja, weryfikacja i ścieżka wycofania (kliknięcie otwiera pełny rozmiar)

Wydania MailCow wychodzą mniej więcej raz w miesiącu i mają numery `RRRR-MM`, z literą dla poprawek (`2026-07a`). Stabilna gałąź to `master`, a gałąź `nightly` jest testowa i nie nadaje się do produkcji. Wydanie 2026-07a zawiera poprawki bezpieczeństwa i autorzy zalecają szybką aktualizację, dlatego zakaz zmian trzeba umieć obejść w uzasadnionym przypadku, patrz niżej.

Skrypt kopii i timer na docker01

Kopię uruchamiamy codziennie w nocy, osobną usługą systemd, żeby długie pakowanie skrzynek nie blokowało kopii innych aplikacji z `kopia-docker.service`, opisanej na stronie Kopia zapasowa aplikacji dockerowych. Konto `nazwa_usera` na `bkp01` i klucz SSH dla roota na `docker01` zakładamy według tamtej strony.

Katalog na lokalne kopie tworzymy raz. Skrypt wymaga, żeby był zapisywalny dla innych, dlatego `755`:

```
sudo mkdir -p /docker/kopie/mailcow
sudo chmod 755 /docker/kopie/mailcow
```

Skrypt kopii zapisujemy w `/docker/skrypty/kopia-mailcow.sh`:

```
#!/usr/bin/env bash
set -euo pipefail

# Kopia MailCow: skrypt z repozytorium MailCow, potem wysyłka na bkp01.
KATALOG_MAILCOW=/docker/apps/mailcow
export MAILCOW_BACKUP_LOCATION=/docker/kopie/mailcow
export THREADS=2
CEL=nazwa_usera@bkp01:/backup

cd "${KATALOG_MAILCOW}"

# Wszystkie składniki, lokalnie zostają kopie z trzech ostatnich dni.
./helper-scripts/backup_and_restore.sh backup all --delete-days 3

# Archiwa na bkp01, bez usuwania starszych (rotację ustawiamy po stronie bkp01).
rsync -rt -e ssh "${MAILCOW_BACKUP_LOCATION}/" "${CEL}/mailcow/"

# Katalog instalacji: konfiguracja, certyfikaty, nasze dodatki.
```

```
rsync -rlptH --delete -e ssh "${KATALOG_MAILCOW}/" "${CEL}/mailcow-instalacja/"
```

```
sudo chmod 700 /docker/skrypty/kopia-mailcow.sh
```

Prawa `700` są celowe, bo w kopiach leży `mailcow.conf` z hasłami do bazy. Przełączniki `-rt` i `-rlptH` pomijają właściciela i grupę plików, bo zwykłe konto na `bkp01` nie może ich ustawiać. Przy odtwarzaniu katalogu instalacji z `bkp01` przywracamy właściciela `root` poleceniem `chown -R`.

Usługa i timer (pliki konfiguracyjne systemd):

```
# /etc/systemd/system/kopia-mailcow.service
[Unit]
Description=Kopia zapasowa MailCow na docker01

[Service]
Type=oneshot
ExecStart=/docker/skrypty/kopia-mailcow.sh
```

```
# /etc/systemd/system/kopia-mailcow.timer
[Unit]
Description=Harmonogram kopii zapasowej MailCow

[Timer]
OnCalendar=*-*-* 03:15:00
Persistent=true

[Install]
WantedBy=timers.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable --now kopia-mailcow.timer
```

Po pierwszym przebiegu sprawdzamy `df -h /docker`, ile miejsca zajmują trzy lokalne kopie, i zapisujemy ten rozmiar w zgłoszeniu. Dokumentacja MailCow nie podaje potrzebnej ilości wolnego miejsca, więc liczymy ją sami z pierwszego przebiegu. Rotację na `bkp01` (ile dni trzymamy) ustalamy tam osobno.

Kiedy nie aktualizujemy

Aktualizacja zatrzymuje całą pocztę na kilka minut, ale w tym czasie nadawcy z zewnątrz ponawiają próby dostarczenia, więc przychodząca poczta nie ginie. Przerwa uderza w ludzi, którzy akurat potrzebują poczty: księgowość przy zamknięciu miesiąca i magazyn przed świętami.

Kiedy	Aktualizujemy?	Dlaczego
Okno zmian: sobota rano, 7:00 do 10:00	tak	najmniej ludzi pracuje, a do poniedziałku zostaje zapas na naprawę (założenie, do potwierdzenia przez autora)
Ostatnie trzy dni robocze miesiąca	nie	zamknięcie miesiąca: pani Halina wysyła pliki do biura rachunkowego, każda przerwa boli
Od 15 listopada do 6 stycznia	nie	zakaz zmian w infrastrukturze, magazyn pracuje na trzy zmiany
Piątek po 15:00	nie	piątek 16:00 to najlepsza pora, żeby coś przestało działać, a naprawiać trzeba przez weekend
Brak kopii z ostatniej doby	nie	przy problemie jedyną drogą powrotu jest kopia
Brak testu odtwarzania w bieżącym kwartale	nie	kopia, której nikt nie odtworzył, jest tylko nadzieją

Wydanie z poprawkami bezpieczeństwa w czasie zakazu zmian wymaga osobnej decyzji (do uzgodnienia z autorem: kto ją podejmuje i w jakim trybie). Do tego czasu robimy to, co wolno: `update.sh --check` i `update.sh --prefetch`.

Wycofanie aktualizacji

Wycofanie ma sens tylko wtedy, gdy nowa wersja nie zmieniła jeszcze schematu bazy albo gdy mamy kopię sprzed aktualizacji. Aktualizacja potrafi zmienić schemat bazy i tego nie da się cofnąć samym przełączeniem kodu, więc przy podejrzeniu zmiany bazy sięgamy po odtworzenie z kopii.

Powrót do kodu sprzed aktualizacji, według dokumentacji MailCow:

```
cd /docker/apps/mailcow
git log --pretty=oneline | grep -i "before update"
sudo docker compose down
sudo git checkout NumerCommitu
sudo docker compose pull
sudo docker compose up -d
```

(gdzie `NumerCommitu` to identyfikator z wiersza „Before update on …" z dnia aktualizacji)

Przed następną aktualizacją wracamy na gałąź poleceniem `sudo git checkout master`. Przy odtwarzaniu z kopii kod ma być w wersji, z której kopia pochodzi, bo kopia bazy ze starszego schematu nie zadziała z nowszym kodem.

Test przywracania na maszynie testowej

Kopię uznajemy za dobrą dopiero wtedy, gdy ktoś z niej poczty odtworzył. Test robimy raz na kwartał (rytm do potwierdzenia przez autora) i po każdej większej zmianie w MailCow. Zapisujemy w nim datę kopii i czas, jaki odtworzenie zajęło.

> Maszyna testowa nie może mieć wyjścia na porty pocztowe ani na `zimbra01`. Odtworzona baza zawiera zadania synchronizacji, powiadomienia o kwarantannie i, jeśli odtworzymy `postfix`, kolejkę wiadomości, więc maszyna z dostępem do sieci wysłałaby prawdziwym ludziom stare wiadomości albo ściągała pocztę z innych serwerów. Odcinamy ruch wychodzący na zaporze `fw01` albo w ustawieniach sieci maszyny wirtualnej.

1. Przygotowanie maszyny

Maszyna wirtualna z Rocky Linux 9 na klastrze Hyper-V, z Dockerem i osobnym wolumenem `/docker` według strony Instalacja Dockera na Rocky Linux 9 z osobnym wolumenem /docker, z co najmniej 6 GiB RAM (minimum z dokumentacji MailCow). Na `docker01` zapisujemy, z jakiego commita działa produkcja:

```
cd /docker/apps/mailcow
git rev-parse HEAD
```

2. Instalacja tej samej wersji MailCow

```
sudo mkdir -p /docker/apps
cd /docker/apps
sudo git clone https://github.com/mailcow/mailcow-dockerized mailcow
cd mailcow
sudo ./generate_config.sh
sudo git checkout NumerCommitu
```

(gdzie `NumerCommitu` to wynik `git rev-parse HEAD` z `docker01`)

Jako nazwę hosta podajemy `poczta-test.zielona.example`, a w `mailcow.conf` ustawiamy `SKIP_LETS_ENCRYPT=y`, bo maszyna testowa nie ma wystawiać certyfikatu. Odtwarzanie wymaga działającego, czystego MailCow w tej samej wersji, więc startujemy stos:

```
sudo docker compose pull
sudo docker compose up -d
```

3. Skopiowanie kopii z bkp01

```
sudo mkdir -p /docker/kopie/mailcow
sudo chmod 755 /docker/kopie/mailcow
sudo rsync -rt -e ssh nazwa_usera@bkp01:/backup/mailcow/ /docker/kopie/mailcow/
```

4. Odtworzenie składników

Uruchamiamy odtwarzanie kilka razy, za każdym razem wybierając jeden składnik z listy. Odtwarzamy kolejno: bazę (`SQL DB`), Redis, `Crypt data`, Rspamd i skrzynki (`Mail directory`). Składnika `Postfix data` nie odtwarzamy, bo zawiera kolejkę wiadomości. Na pytanie o ponowny odczyt skrzynek odpowiadamy `y`.

```
cd /docker/apps/mailcow
sudo env MAILCOW_BACKUP_LOCATION=/docker/kopie/mailcow THREADS=2 ./helper-
scripts/backup_and_restore.sh restore
```

Skrypt sam przepisuje do bieżącego `mailcow.conf` parametry bazy z kopii i uruchamia stos od nowa po odtworzeniu bazy.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: terminal z listą składników po wybraniu punktu odtwarzania, widoczne pozycje „[0] - all", „Crypt data", „Rspamd data", „Mail directory (/var/vmail)", „Redis DB", „Postfix data" i „SQL DB".

5. Sprawdzenie, czy poczta jest na miejscu

Do nazwy `poczta-test.zielona.example` wchodzimy z komputera, na którym dopisaliśmy ją do pliku hosts. Logujemy się w panelu kontem administratora (hasło takie jak na produkcji, bo baza wróciła

z kopii) i sprawdzamy trzy rzeczy: listę domen i skrzynek, klucz DKIM domeny `zielona.example` w zakładce ARC/DKIM keys oraz liczbę wiadomości w skrzynce testowej. Liczbę wiadomości porównujemy z produkcją:

```
cd /docker/apps/mailcow
sudo docker compose exec dovecot-mailcow doveadm mailbox status -u
halina.kowalska@zielona.example messages '*'
```

Porównujemy folder po folderze. Różnica rzędu ostatnich minut przed kopią jest normalna.

6. Posprzątanie

```
cd /docker/apps/mailcow
sudo docker compose down -v
sudo rm -rf /docker/kopie/mailcow/mailcow-*
```

Polecenie `down -v` usuwa wolumeny maszyny testowej razem z odtworzoną pocztą. Po teście zostaje wpis z datą kopii, czasem odtwarzania i uwagami, na przykład „wszystkie skrzynki obecne, DKIM obecny".

Częste błędy

Objaw	Przyczyna	Co robimy
Zadanie nocne wisi i nie kończy się	skrypt pyta interaktywnie o katalog kopii, bo nie ustawiono `MAILCOW_BACKUP_LOCATION`	ustawiamy zmienną w skrypcie, jak wyżej
`... is not write-able for others, that's required for a backup`	katalog kopii ma za wąskie uprawnienia	`sudo chmod 755 /docker/kopie/mailcow`
`/docker` się zapełnia	trzy lokalne kopie plus stare obrazy	patrz MailCow: częste problemy, `update.sh --gc`, krótszy `--delete-days`
`update.sh` kończy się przy sprawdzaniu łączności	zapora blokuje ICMP z serwera	uruchamiamy z `--skip-ping-check`
`Oh no, what happened?` przy scalaniu kodu	w katalogu instalacji leży plik, który pojawił się w repozytorium MailCow	przenosimy własny plik w inne miejsce i ponawiamy
Po aktualizacji coś nie działa, a kopia jest sprzed niej	zmiana schematu bazy albo błąd w wydaniu	sprawdzamy logi, potem wycofanie albo odtworzenie z kopii
Po odtworzeniu brakuje klucza DKIM	kopia bez składnika `redis`	generujemy klucz w ARC/DKIM keys i publikujemy nowy rekord TXT

Objaw	Przyczyna	Co robimy
Odtworzenie składnika Rspamd odpada z komunikatem o architekturze	kopia pochodzi z innej architektury procesora	pomijamy Rspamd, uczenie zacznie się od nowa

□□ Ćwiczenia

1. Dlaczego sam rsync katalogu `/docker/apps/mailcow` na `bkp01` nie wystarcza jako kopia poczty?

Odpowiedź

Wiadomości, baza i klucze DKIM leżą w wolumenach Dockera pod `/docker/lib/volumes`, nie w katalogu instalacji. Katalog instalacji zawiera konfigurację i certyfikaty, ale bez skrzynek i bazy nie ma z czego odtworzyć poczty.

1. Dziś jest czwartek 26 listopada, `update.sh --check` pokazuje nowe wydanie. Aktualizujemy?

Odpowiedź

Nie, bo od 15 listopada do 6 stycznia obowiązuje zakaz zmian w infrastrukturze. Zostaje `update.sh --check` i `update.sh --prefetch` oraz decyzja, czy wydanie z poprawkami bezpieczeństwa uzasadnia wyjątek.

1. Dlaczego na maszynie testowej nie odtwarzamy składnika `postfix` i dlaczego odcinamy jej ruch wychodzący?

Odpowiedź

W składniku `postfix` jest między innymi kolejka wiadomości, a w odtworzonej bazie zadania synchronizacji i powiadomienia o kwarantannie. Maszyna z dostępem do sieci wysłałaby prawdziwym odbiorcom stare wiadomości albo ściągała pocztę z innych serwerów, więc test nie może mieć wyjścia na porty pocztowe.

1. Po aktualizacji panel nie działa, a `docker compose ps` pokazuje wszystkie kontenery zdrowe oprócz jednego. Od czego zaczynamy, od wycofania czy od logów?

Odpowiedź

Od logów tego jednego kontenera (`docker compose logs --tail 100 NazwaKontenera`). Wycofanie zmienia całą instalację, a problem dotyczy jednego kontenera. Dopiero gdy logi wskazują na zmianę schematu bazy albo błąd wydania, decydujemy o wycofaniu albo odtworzeniu z kopii.

Powiązane: MailCow: co to jest i z czego się składa · MailCow: częste problemy · Kopia zapasowa aplikacji dockerowych · Utrzymanie hosta dockerowego

Źródła: `helper-scripts/backup_and_restore.sh` i `update.sh` w github.com/mailcow/mailcow-dockerized (kod z 3.10.2026) · docs.mailcow.email: Backup, Restore, Update, Update hooks · dokumentacja `rsync`

Revision #14
Created 2026-10-03 12:08:20 UTC by Tomasz Konieczny
Updated 2026-10-26 14:20:39 UTC by Tomasz Konieczny