Engineer

Резервна копія

копія даних та її створення на випадок втрати

Що це таке

Резервна копія — це те, з чого відновлюють дані, коли оригінал видалено, пошкоджено або втрачено разом із машиною, на якій він зберігався, тож вона мусить лежати там, куди ці події не дістануть. Глосарій NIST визначає її як копію файлів і програм, зроблену для полегшення відновлення. У розмовній мові вживають слово бекап, а саму дію називають резервним копіюванням: нічне завдання виконує резервне копіювання бази даних і залишає по собі резервну копію.

Із нею плутають сусідні речі. Репліка отримує кожну зміну за лічені секунди: це захищає від мертвого диска, але так само сумлінно копіює помилковий `DELETE`. Знімок, або snapshot, заморожує том в одну мить і стає резервною копією лише тоді, коли зберігається там, куди збої оригіналу не дістануть. Резервну копію визначають історія та відокремленість: вона повертає дані такими, якими вони були раніше, з місця, яке пережило те, що забрало оригінал.

Бази даних роблять це конкретним. Документація PostgreSQL описує три принципово різні підходи. SQL-дамп, створений через `pg_dump`, записує команди, які відтворюють базу, як узгоджений знімок того моменту, коли дамп почався. Резервна копія на рівні файлової системи копіює сам каталог даних. Безперервне архівування поєднує таку базову копію з журналом попереднього запису (write-ahead log, WAL), який сервер створює після неї, і відтворення цього журналу дає змогу відновити базу на будь-який момент після базової копії. Цю можливість називають відновленням на момент часу, point-in-time recovery.

Застосуйте поняття до рішення

Застосуйте «Що таке резервна копія?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Резервні копії, які перевірено, а не просто вважають робочими, і записаний план на той ранок, коли сервер не ввімкнеться.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Автоматизувала 100 критичних резервних копій — 100 критичних резервних копій автоматизовано в усіх базах

Автоматизувала 100 критичних резервних копій Зв'язатися з нами →

Чому це важливо

Два числа перетворюють загальне бажання вберегти дані на проєкт. NIST визначає цільову точку відновлення, RPO, як момент часу, до якого дані мають бути відновлені після збою: на практиці це те, скільки останньої роботи бізнес може дозволити собі втратити. Цільовий час відновлення, RTO, — це те, як довго система може перебувати у відновленні, перш ніж збій зашкодить тому, заради чого існує організація.

Ці два числа тягнуть різні частини проєкту. RPO задає частота копій: нічний дамп може втратити день записів, а архівування WAL зменшує втрату до останніх миттєвостей журналу, які ще в дорозі до архіву. RTO задає тривалість відновлення. Документація PostgreSQL зазначає, що сервер відтворює кожен сегмент WAL від останньої базової копії, і це може тривати довго, якщо копія стара. Тож частота базових копій — таке саме рішення про RTO, як і про місце на диску.

Обидва числа — твердження про відновлення, і тому резервну копію оцінюють, відновлюючи її. Завдання, яке завершилося зі статусом нуль, показує лише, що десь записано байти. Чи відтворять вони робочу систему в межах RTO, показує відновлення за розкладом, поза продакшеном, з порівнянням того, що повернулося.

Відокремленість — друга половина. Копія в тому самому обліковому записі, з тими самими обліковими даними, поділяє збої, які мала б пережити: закритий обліковий запис, викрадений ключ, скрипт, запущений проти хибної цілі. Корисне запитання до будь-якої копії: яка одна подія знищила б її разом з оригіналом?

Як це виглядає на практиці

На найменшому масштабі реалізацію резервного копіювання та тестування відновлення компанії спроєктовано, щоб забезпечити основу для відновлення, ширшу за знімок провайдера. Роль Ansible встановлює скрипти restic і юніти systemd для дампа бази даних там, де його налаштовано, зашифрованих дедуплікованих знімків, видалення за правилами зберігання та перевірок цілісності. Окремий щомісячний процес відновлює відомий файл у тимчасове місце та порівнює його байти. Копіювання й відновлення мають окремі сигнали успіху, а таймери потребують налаштованого сховища. Парольну фразу читають із захищеного файла, а не зі значення змінної середовища, яке розбирає systemd. Це реалізовані засоби контролю та передумови активації: у записі про реалізацію від 2026-10-27 зазначено, що таймери були вимкнені без сховища, і не зафіксовано жодного завершеного відновлення в реальних умовах. Запис підтверджує роботу над автоматизацією, а не твердження про успішне відновлення чи про те, що хост уже можна безпечно втратити.

У більшому масштабі сто критичних резервних копій даних автоматизовано за допомогою Barman і Google Cloud для продукту, який живе своїми GIS-даними. Barman відповідав за PostgreSQL, Google Cloud тримав офсайт-копії, а Bash і Python оркестрували й перевіряли запуски. Перевірки цілісності підтверджували, що кожна копія справді придатна, а не просто наявна.

Ще в одному продукті позицію відновлення треба було визначити, а не припускати. Політика зберігання резервних копій живе в шаблоні Вісер поруч із базою даних, тож відновлення на момент часу є властивістю шаблону, а не налаштуванням, яке хтось колись клікнув у порталі, і середовище довкола теж описане кодом. Аудит записав прогалини: геонадлишковість вимкнено, а RTO запропоновано, а не виміряно, і тестове відновлення, яке його виміряє, описане й чекає на запуск.

Де виникають проблеми

Проблеми з резервними копіями зазвичай лишаються прихованими, доки не знадобиться відновлення, тож кожну з них варто пошукати в спокійний день.

Файлова копія робочої бази відновлюється як зламаний кластер. Якщо копіювати каталог даних PostgreSQL через `tar`, поки сервер працює, файли збираються з різних моментів, і документація вимагає зупинити сервер, щоб отримати придатну копію таким способом. Знімок файлової системи, зроблений в одну атомарну мить, спрацює, якщо містить WAL: сервер відтворить його під час запуску, як після аварії. Якщо дані й WAL лежать на різних файлових системах, одночасний знімок може виявитися неможливим. `pg_dump` або базова копія для безперервного архівування дає узгоджену копію, поки сервер працює далі.

Прогалина в архіві WAL обриває відновлення на момент часу саме на ній. Для відновлення потрібна неперервна послідовність заархівованих файлів WAL аж до початку базової копії. Якщо команда архівування раз у раз завершується помилкою, PostgreSQL повторює спроби, а WAL накопичується в `pg_wal/`, тож першим симптомом часто стає диск, що заповнюється. Документація радить налаштувати й перевірити архівування до першої базової копії та стежити за затримками.

Завдання зупиняється, і тиша схожа на успіх. Завдання таймера, яке зламалося, пише помилку в журнал, якого ніхто не читає, і кожен тихий день здається добрим. Моніторинг має сприймати тишу як тривогу: сигнал, який надсилається лише після успішного запуску до служби, що сповіщає про його запізнення, ловить і завдання, яке зупинилося, і те, що падає гучно.

Ключ губиться або надходить зміненим. Шифрування робить доступ залежним від ключів. restic підтримує кілька паролів для одного репозиторію; застереження про безповоротну втрату даних стосується ситуації, коли не лишається жодного придатного ключа репозиторію. Ключ має зберігатися там, де він переживе машину. У налаштуванні restic вище парольна фраза читається з файлу, а не з оточення, бо диспетчер процесів обробляє екранування у значеннях змінних оточення, і парольна фраза зі зворотною похилою рискою мовчки надійшла б іншою, ніж та, що створила репозиторій.

Політика зберігання видаляє саме ту копію, яка була потрібна. Пошкодження часто помічають через тижні, і політика, що тримає сім щоденних копій, на той час уже замінила кожну чисту копію пошкодженою. Багаторівнева політика зберігає довгий хвіст за невелику ціну, а `--dry-run` у restic показує, що політика видалила б, ще до того, як щось зникне:

```
restic forget --keep-daily 7 --keep-weekly 5 --keep-monthly 12 \
  --prune --dry-run
```

`forget` видаляє знімки, а `prune` — дані, на які посилалися лише вони, тож після prune старі копії зникають назавжди.

Перевірка цілісності проходить, а відновлення все одно ламається. Типово `restic check` перевіряє структуру репозиторію, не читаючи кожен pack-файл, бо для цього довелося б завантажити їх усі; `--read-data` читає все, а `--read-data-subset=1/5` читає п'яту частину, тож п'ять запусків від `1/5` до `5/5` покривають усе. Повне читання доводить лише, що збережені байти цілі. Чи відтворять вони робочу базу з правильними ролями й налаштуваннями, покаже відновлення.

Що почитати далі

- NIST glossary: backup — визначення, на яке спирається ця стаття, і публікації, з яких воно походить.
- NIST SP 800-34 Rev. 1 — посібник із планування дій на випадок збоїв, який визначає RPO і RTO.
- PostgreSQL: Backup and Restore — три підходи і ціна кожного з них.
- PostgreSQL: Continuous Archiving and Point-in-Time Recovery — базові копії, архівування WAL і відновлення на обраний момент.
- restic: Preparing a new repository — паролі, файли паролів і попередження про їхню втрату.
- restic: Removing backup snapshots — політики зберігання, `forget`, `prune` і `--dry-run`.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке резервна копія?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/backup/