Engineer

Prometheus

база даних, що збирає і зберігає метрики

Що це таке

Prometheus — це система моніторингу з відкритим кодом, побудована навколо власної бази даних часових рядів. Кожні кілька секунд вона звертається по HTTP до програм, за якими стежить, забирає їхні поточні числа, зберігає кожне значення з позначкою часу і дає змогу робити запити, будувати графіки та налаштовувати сповіщення за всім, що накопичилося. Цією назвою позначають і сам сервер, і невелику родину компонентів, що виросла навколо нього.

Проєкт створили в SoundCloud, де робота над ним почалася 2012 року, а нині він належить Cloud Native Computing Foundation. До фонду він приєднався 2016 року як другий проєкт після Kubernetes, а 2018 року отримав статус graduated. Prometheus 3.0, анонсований у листопаді 2024 року, став першим мажорним релізом за сім років. Код здебільшого написаний мовою Go і поширюється за ліцензією Apache 2.0.

Усе, що зберігає Prometheus, — це часові ряди. Ряд має назву метрики й набір міток, тобто пар «ключ — значення», які вказують, який саме екземпляр чогось виміряно, наприклад `api_http_requests_total{method="POST", handler="/messages"}`. Кожен відлік у такому ряді — це 64-бітне число з рухомою комою або нативна гістограма разом із позначкою часу в мілісекундах. Інше значення будь-якої мітки означає інший ряд. Саме тому дані легко розрізати за хостом, маршрутом чи кодом статусу, і саме тому їх так легко перевантажити.

Програми викладають свої числа на окремій HTTP-адресі, зазвичай `/metrics`, через клієнтську бібліотеку, вбудовану в код, або через експортер — невеликий адаптер, що перекладає показники системи, яку Prometheus не може інструментувати напряму, наприклад Linux-хоста чи бази даних.

Сервер знаходить свої цілі через service discovery або статичний список і опитує їх за розкладом; англійською це називають scrape. Пакетне завдання, яке завершується раніше за наступне опитування, може натомість надіслати свої числа в окремий push gateway. Правила сповіщень виконуються на сервері, а ті, що спрацювали, потрапляють до Alertmanager, який прибирає дублікати, групує їх, застосовує тимчасові вимкнення (silences) і надсилає далі — на електронну пошту, у PagerDuty, OpsGenie та подібні канали.

Його зазвичай порівнюють із push-збирачами в традиції Graphite і StatsD, із загальними базами часових рядів на кшталт InfluxDB та з хмарними сервісами метрик. Обличчям для нього здебільшого слугує Grafana. Prometheus зберігає числа, а не події, тож рядок журналу чи трасування мають жити деінде. Проєкт прямо каже, що надійність для нього важливіша за повноту: білінг за кожен запит, де кожен підрахунок мусить бути точним, — завдання для іншої системи.

Застосуйте поняття до рішення

Застосуйте «Що таке Prometheus?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Моніторинг і сповіщення, які попереджають, поки збій ще малий, і цифри, що показують, як часто так і залишається.

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. Запустила перший дашборд спостережуваності — перший огляд власних систем компанії

Запустила перший дашборд спостережуваності Зв'язатися з нами →

Чому це важливо

За Prometheus стоїть рішення: тримати власну базу метрик чи орендувати чужу, а якщо тримати власну, то якого розміру.

Ліцензійної плати немає. Витрати — це машина й години людини, яка стежить за її станом. Хмарний сервіс знімає цю роботу, але натомість виставляє рахунок, що росте разом з обсягом надісланих даних, тож кожна нова мітка стає рядком у рахунку. Для кількох десятків машин власний сервер зазвичай дешевший і простіший. Розрахунок змінюється, коли кількість рядів сягає мільйонів, а команда воліє платити, ніж виділяти на це людину.

Арифметика розміру м’якша, ніж зазвичай очікують. Документація проєкту про зберігання оцінює середні витрати в 1–2 байти на відлік. Тисяча рядів, що опитуються кожні 15 секунд і зберігаються стандартні 15 днів, — це 86,4 мільйона відліків, приблизно 86–173 МБ на диску. Диск закінчується рідко. Закінчується пам’ять: блок, який зараз заповнюється, живе в оперативній пам’яті, від збою його захищає журнал попереднього запису (write-ahead log), і кожен активний ряд займає там місце незалежно від того, чи хтось колись його запитає.

Кожен сервер навмисно самодостатній. Він не спирається на мережеве сховище, тож продовжує відповідати, коли інфраструктура, за якою він стежить, розвалюється, — саме тоді, коли він потрібен найбільше. Ціна в тому, що локальне сховище не кластеризоване й не реплікується. Висока доступність означає два однакові сервери, що опитують ті самі цілі, а історія, довша за кілька тижнів, — копію, надіслану через `remote_write` у сховище, створене для довгого зберігання.

У того, що сервер сам питає, а не чекає, є й практична вигода. Він помічає, коли ціль перестає відповідати, і записує це як `up`, що впав до нуля. А інженер може відкрити ту саму сторінку `/metrics` у браузері й побачити рівно те, що бачить сервер, тож суперечку між графіком і застосунком вдається розв’язати швидко.

Як це виглядає на практиці

У роботі засновниці Prometheus забезпечує збирання даних для дашборда: в одному випадку для групи серверів баз даних, в іншому — для другої версії дашборда, за яким стежив увесь офіс.

Випадок із базами даних — десять серверів із PostgreSQL 14 та MS SQL Server 2019 на віртуальних серверах з Ubuntu 20.04 LTS. Prometheus і Grafana відстежували CPU, пам’ять, дискове введення-виведення і продуктивність запитів у реальному часі, поряд з автоматичними резервними копіями, правилами брандмауера та власними скриптами для аналізу журналів. На всіх десяти серверах доступність трималася на рівні 99,9%. Хости баз даних добре підходять для опитування: числа, що віщують проблеми, — диск, який заповнюється, чи запит, що сповільнюється, — змінюються годинами, а відлік раз на кілька секунд помічає їх задовго до користувача.

Офісний дашборд показує крок від саморобного збирача до Prometheus. На одній із попередніх посад засновниця створила перший в організації дашборд спостережуваності. Він спершу зчитував CPU, RAM, диск, температуру та інше з віддалених Linux-серверів через SSH і так само стежив за Docker-контейнерами. Другу версію збудували на Prometheus і Grafana навколо метрик, обраних разом із командами DevOps та розробки: завантаження CPU, використання пам'яті, час безвідмовної роботи сервісів і затримка API, зі сповіщеннями в разі перетину порогів. З дашбордом команди виявляли й усували інциденти на 40% швидше. Опитування через SSH — непоганий початок для кількох машин, а збирач із мітками й мовою запитів — те, що дає тому самому дашборду рости далі.

Де виникають проблеми

Пам'ять росте після релізу, у якому змінилися лише дрібниці. Хтось додав мітку з необмеженою кількістю значень: ідентифікатор користувача, адресу електронної пошти, повний шлях URL, ідентифікатор запиту. Кожне нове значення стало новим рядом, і сервер тепер тримає мільйони рядів там, де тримав тисячі. Настанови щодо іменування застерігають саме від цього. Тримайте значення міток у невеликих відомих межах, деталі окремих запитів надсилайте в журнали чи трасування, а найгірших порушників знайдіть таким запитом:

```
topk(10, count by (__name__) ({__name__=~".+"}))
```

Графік показує обрив чи сплеск, якого не було. Лічильник лише зростає, а після перезапуску процесу падає до нуля. Якщо малювати сире значення або віднімати вручну, кожен перезапуск перетворюється на аварію чи наплив трафіку. `rate()` розуміє скидання, тож спершу беріть швидкість, а вже потім додавайте, і вікно робіть завширшки кілька інтервалів опитування:

```
sum by (job) (rate(http_requests_total{status=~"5.."}[5m]))
```

Правило запису (recording rule), яке підсумовує лічильник до обчислення швидкості, ховає скидання всередині суми й повертає фальшиві обриви.

Сервіс падає, а його сповіщення про помилки лишається зеленим. Правило на кшталт «частка помилок вища за поріг» обчислюється над рядами, які існують. Коли ціль зникає, зникають і її ряди, вираз повертає порожній результат, а сповіщення закривається саме в найгірший момент. Ставте сповіщення на `up == 0` для кожної цілі, використовуйте `absent()` для рядів, що мають бути завжди, і нехай щось поза сервером перевіряє, що сам сервер працює.

Дані за минулий квартал зникають саме тоді, коли про них питають. Локальне сховище за замовчуванням тримає 15 днів і є однією копією на одному диску. Як вікно в недавню поведінку системи воно працює, а як архів підводить. Збільшуйте строк зберігання обережно, те, що має прожити довше, надсилайте в довгострокове сховище через `remote_write`, а цифри, від яких залежать гроші, тримайте в базі даних обліку.

Що почитати далі

- Prometheus overview — власний опис проєкту: що це, з яких компонентів воно складається і коли це невдалий вибір.
- Data model — назви метрик, мітки й відліки, і чому кожне значення мітки починає новий ряд.
- Metric types — counter, gauge, histogram і summary та на які питання відповідає кожен із них.
- Querying basics — селектори PromQL, діапазонні вектори й типи, які може повернути вираз.
- Storage — двогодинні блоки, журнал попереднього запису, строк зберігання і remote write.
- Metric and label naming — базові одиниці, суфікс `_total` і застереження щодо міток із високою кардинальністю.

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 Prometheus?» 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/prometheus/