Engineer

Бюджет помилок

частка збоїв, яку ще дозволяє ціль надійності

Що це таке

Бюджет помилок — це обсяг збоїв, який сервісу дозволено за визначений період; його виводять із цілі рівня обслуговування (SLO). Якщо ціль вимагає, щоб цього кварталу успішно завершилися 99,99% запитів, то решта 0,01% і є бюджетом: запити, які можуть завершитися помилкою, випадково чи через свідомо взятий ризик, і ніхто при цьому не порушить обіцянки. Англійською це називають error budget. Бюджет рахують у тих самих одиницях, що й ціль, найчастіше в невдалих запитах, і витрачається він тієї миті, коли користувач натрапляє на збій.

Ідея походить із книжки Google про Site Reliability Engineering. У розділі Embracing Risk ціль задає продуктове керівництво, система моніторингу виступає нейтральною третьою стороною, яка вимірює, що сталося насправді, а різниця між ними — це ненадійність, яка ще лишилася на квартал. Розробка продукту й SRE погоджують цей бюджет разом і заздалегідь. Приклад у книжці взято з вимогливого кінця шкали: сервіс, який мусить відповісти на 99,999% запитів, має бюджет 0,001%, тож одна проблема, через яку збоять 0,0002% запитів кварталу, з'їдає п'яту частину бюджету.

Бюджет не питає, чия це провина. Книжка прямо каже, що збій мережі чи відмова дата-центру витрачають його так само, як невдалий реліз, і що решта кварталу тоді може минути з меншою кількістю релізів. Користувачі відчули збій, отже, він рахується. Хто має на нього реагувати — окреме питання, і відповідь на нього належить до письмової політики, погодженої до того, як щось піде не так.

Застосуйте поняття до рішення

Застосуйте «Що таке бюджет помилок?» 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

Ті, хто випускає нові функції, і ті, хто тримає сервіс у робочому стані, хочуть різного, і без спільного числа суперечку між ними виграє той, хто краще торгується. Книжка про SRE каже, що такі рішення не мають спиратися ні на політику, ні на страх, ні на надію, і нагадує неофіційне гасло Google SRE: «Надія — не стратегія». Бюджет замінює суперечку числом, яке обидві сторони прийняли заздалегідь. Поки бюджет є, релізи виходять; коли його витрачено, вони стають на паузу, а зусилля переходять на тестування й стійкість. Книжка описує й м’якші варіанти того самого механізму, наприклад сповільнювати релізи або відкочувати їх у міру того, як бюджет наближається до нуля, а у виносці називає простий варіант «увімкнено або вимкнено» керуванням bang/bang.

Найцікавіший ефект стосується поведінки. Коли бюджет великий, розробники можуть ризикувати: частіше викладати зміни, пробувати експеримент. Коли він майже вичерпаний, зауважує книжка, розробники самі просять більше тестування й повільніших релізів, бо ніхто не хоче зірвати власний запуск. Команда починає стежити за собою сама. Книжка ставить одну умову: це працює, лише якщо SRE справді має повноваження зупиняти запуски, щойно ціль порушено.

Бюджет керує здебільшого релізами, бо здебільшого саме релізи щось ламають: приклад політики в SRE Workbook оцінює, що зміни стоять приблизно за 70% збоїв Google. Тож бюджет підказує, куди варто спрямувати зусилля. Розділ Workbook про впровадження SLO розбирає сервіс із ціллю 97% і бюджетом на чотири тижні в 109 897 невдалих запитів. Невдалий реліз API, через який чотири години збоїли всі запити, коштував 14 066 із них, приблизно 13%. Сервер бази даних, який вийшов з ладу й 20 годин відновлювався з резервної копії, коштував за оцінкою 72 000, тобто 65%. Відмова сервера виглядає значно гірше, але компанія з прикладу стикається

з такою раз на п’ять років, а невдалих релізів має два-три на рік, тож у середньому релізи обходяться приблизно вдвічі дорожче. Спершу окупається виправлення процесу релізів.

Бюджет, який ніколи не витрачається, теж про щось свідчить. Книжка зазначає, що він показує, чого коштує надто висока ціль у негнучкості й повільних інноваціях, і що команда, якій важко випустити продукт, може послабити ціль і тим самим збільшити бюджет.

Як це виглядає на практиці

Витрата має не лише суму, а й швидкість. Розділ Workbook про сповіщення називає цю швидкість burn rate, тобто темпом спалювання бюджету: наскільки швидко, відносно цілі, сервіс витрачає свій бюджет. За burn rate 1 бюджет закінчується рівно наприкінці періоду. Для цілі 99,9% за 30 днів таблиця з Workbook виглядає так:

Burn rate	Частка помилок	Бюджет вичерпано через
1	0,1%	30 днів
2	0,2%	15 днів
10	1%	3 дні
1 000	100%	43 хвилини

Сповіщати за burn rate краще, ніж піднімати чергового щоразу, коли частка помилок перевищує 0,1%: тоді його будив би кожен короткий сплеск. Workbook радить стежити за кількома значеннями burn rate водночас, кожне — у довгому й короткому вікні. Його відправна точка для цілі 99,9% піднімає чергового, коли за годину витрачено 2% 30-денного бюджету (burn rate 14,4) або за шість годин — 5% (burn rate 6), і відкриває тікет, коли за три дні витрачено 10% (burn rate 1). Коротке вікно, що становить дванадцяту частину довгого, змушує тривогу вщухнути невдовзі після того, як минула проблема. У прикладі з Workbook середнє в короткому вікні опускається нижче порогу через п’ять хвилин після того, як помилки припинилися, а середнє за годину лишається вищим за поріг ще цілу годину.

У файлі правил Prometheus дві умови для виклику чергового виглядають так. Частки — це recording rules, названі так само, як у Workbook, і кожна містить частку невдалих запитів за вікно, вказане в її назві:

```
groups:
  - name: orders-error-budget
    rules:
      - alert: OrdersErrorBudgetBurn
        expr: |
          (
            job:slo_errors_per_request:ratio_rate1h{job="orders"}
              > (14.4 * 0.001)
            and
            job:slo_errors_per_request:ratio_rate5m{job="orders"}
              > (14.4 * 0.001)
          )
          or
          (
            job:slo_errors_per_request:ratio_rate6h{job="orders"}
              > (6 * 0.001)
            and
            job:slo_errors_per_request:ratio_rate30m{job="orders"}
              > (6 * 0.001)
          )
        labels:
          severity: page
```

Простежмо через нього одне невдале розгортання. О 14:00 реліз починає зривати 2% запитів на замовлення, а це burn rate 20. П’ятихвилинна частка перетинає межу 1,44% протягом чотирьох хвилин, але годинна досягає її лише тоді, коли помилки тривають уже близько 43 хвилин, тож чергового піднімають приблизно о 14:43. На той момент витрачено рівно 2% місячного бюджету, і в цьому компроміс, на який іде правило: короткий сплеск нікого не будить, а тривале спалювання не сховається. Відкотіть реліз о 14:50, і п’ятихвилинна частка опуститься нижче межі протягом двох хвилин, тож тривога припиниться; сама лише годинна частка тримала б її приблизно до 15:07.

Тримати сповіщення співмірними з проблемою — та сама турбота, що стоїть за моніторингом помилок і трасуванням 2025 року, які компанія створила власними силами. Вони виявляють сплески й регресії з паузою між сповіщеннями, тож одне невдале розгортання не піднімає чергового сорок разів, правила сповіщень підігнано саме під цю платформу, а не під абстрактну, і одне з 11 операторських подань відведено цілям рівня обслуговування.

Де виникають проблеми

Облік збою плутають із пошуком винних. Усе, що відчувають користувачі, витрачає бюджет, але не все має зупиняти власні релізи команди. Приклад політики з Workbook дозволяє продовжувати роботу над функціями, коли збій спричинила проблема з мережею в масштабі всієї компанії або сервіс іншої команди, якщо та команда сама заморозила свої релізи; коли ж бюджет витратила власна помилка чи процедура команди, надійність стає першочерговою. Для збою, спричиненого залежністю від іншої команди, розділ Implementing SLOs описує дві школи, працювати далі або все одно заморозити релізи, і каже, що заморожування робить користувачів задоволенішими. Виберіть заздалегідь, запишіть вибір у політику й визначте, хто розв'язує суперечки щодо підрахунку; приклад політики передає їх CTO.

Помилки потрапляють не в ту категорію. Невдалий навантажувальний тест чи відхилені запити пентестера витрачають бюджет, хоча жоден користувач не постраждав, і приклад політики називає неправильно класифіковані помилки одним зі способів марно втратити бюджет. Зворотне ще гірше: справжні збої, записані як щось нешкідливе, тримають сервіс у межах цілі на папері, тоді як користувачі страждають, і та сама політика робить роботу над надійністю обов'язковою, коли таке виявлено. Не пускайте в індикатор трафік, якого ціль не охоплює, і перевіряйте класифікацію після кожного інциденту, вартість якого виглядала дивно.

Великий інцидент оплачують і забувають. Заморожування дає час, але не розуміння. Приклад політики пов'язує одне з іншим: окремий інцидент, що витратив понад 20% чотиритижневого бюджету, вимагає постмортему щонайменше з одним завданням найвищого пріоритету, а клас збоїв, що за квартал витратив понад 20%, має потрапити до плану наступного кварталу.

Тривога ніколи не спрацює. Пороги, скопійовані з таблиці для 99,9%, не переносяться на інші цілі. За доступності 90%, зауважує Workbook, повний збій витрачає за годину лише 1,4% бюджету, тож правило, яке піднімає чергового на 2% за годину, мовчатиме весь цей час. На іншому кінці шкали місячну ціль 99,999% повний збій вичерпує за 26 секунд, швидше, ніж багато систем моніторингу встигають зібрати хоча б одне вимірювання. Налаштуйте вікна й пороги під ціль, а для дуже високих цілей прийміть висновок Workbook, що захистити їх може лише проєктування системи: спершу розгорніть зміну на 1% користувачів, і той самий збій лишить 43 хвилини до вичерпання бюджету.

Через низький трафік одна помилка піднімає чергового. Якщо на годину приходить десять запитів, один невдалий запит дає 10% помилок за цю годину. Для цілі 99,9%, зазначає Workbook, цей єдиний запит одразу піднімає чергового, бо витрачає 13,9% 30-денного бюджету: за весь місяць дозволено лише сім збоїв. Workbook пропонує генерувати штучний трафік, об'єднати кілька малих сервісів в один для моніторингу або змінити продукт так, щоб один невдалий запит важив менше.

Ціль підвищують раніше, ніж сервіс здатен її виконати. Команда, якій потрібна суворіша ціль і яка запроваджує її одразу, живе в постійному аварійному режимі, а бюджет у неї завжди витрачено. Workbook натомість пропонує амбітну ціль: її вимірюють і відстежують поряд із поточною, але в політиці зазначають, що вона не вимагає жодних дій, доки сервіс не буде до неї готовий.

Що почитати далі

- Embracing Risk — розділ 3 книжки Google про Site Reliability Engineering, де бюджет помилок запроваджено як спосіб ухвалювати рішення про релізи на основі даних.
- Alerting on SLOs — SRE Workbook про burn rate, сповіщення з кількома вікнами й сервіси з малим трафіком.
- Example Error Budget Policy — повна політика: коли релізи зупиняються, які збої є винятками і коли потрібен постмортем.
- Implementing SLOs — як вивести бюджет із цілі й використати його, щоб вибрати, у що вкладати зусилля з надійності.
- Alerting rules — документація Prometheus про формат файлу правил, використаний вище.

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 бюджет помилок?» 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/error-budget/