Engineer

エラーバジェット

信頼性の目標がまだ許容する失敗の量

どのようなものか

エラーバジェットは、定めた期間にサービスが許容する失敗の量です。サービスレベル目標から求めます。今四半期のリクエストの99.99%が成功するという目標なら、残る0.01%がバジェットです。偶然の事故でも、意図して取ったリスクでも、約束を破らずに失敗できるリクエストです。目標と同じ単位、通常は失敗したリクエスト数で数え、利用者が失敗に遭遇した時点で消費します。

考え方の出典は、Googleの_Site Reliability Engineering_です。リスクを受け入れる章では、プロダクト管理側が目標を決め、監視システムが中立の第三者として、実際に起きたことを測ります。両者の差が、その四半期にまだ許容される失敗の余地です。製品開発とSREは、事前に一緒に、そのバジェットに合意します。同書の例は、厳しい側の目標です。クエリーの99.999%に応答するサービスには、0.001%のバジェットがあります。四半期のクエリーの0.0002%を失敗させる問題が一つ起きると、五分の一を使います。

誰の責任かは、バジェットに関係ありません。同書は、ネットワーク停止やデータセンターの障害も、不具合のあるリリースと同じようにバジェットを使い、その後、四半期の残りのデプロイ回数を減らす場合があると明記します。利用者が失敗の影響を受けたので、数えます。誰が対応するかは別の問いで、問題が起きる前に合意した、文書の方針で決めます。

概念を判断に生かす

「エラーバジェットとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 障害が小さいうちに知らせる監視とアラートを整え、安定した状態をどれだけ維持できているかを数値で示します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 社内初のオブザーバビリティダッシュボードを構築 — 社内システムの状態を初めて可視化

社内初のオブザーバビリティダッシュボードを構築 お問い合わせ →

なぜ重要なのか

機能を届ける人と、サービスを動かし続ける人では、求めるものが違います。共通の数字がなければ、交渉のうまい側が議論に勝ちます。SREの本は、その判断を、政治、恐れ、希望のいずれにも頼るべきではないとし、Google SREの非公式の標語を挙げます。「希望は戦略ではない」。バジェットは、議論を、双方が事前に受け入れた数字へ置き換えます。残っている間はリリースし、使い切ったら止め、試験や障害への強さを高める仕事へ力を移します。同書は、残りが少なくなったらリリースの頻度を下げたり、ロールバックしたりする、より緩やかな運用の形も説明します。脚注では、単純なオン・オフ型を、バンバン制御と呼びます。

より興味深いのは、行動への影響です。余裕が大きければ、開発者は、頻繁にデプロイしたり、実験を試したりと、リスクを取れます。同書によると、ほぼ使い切ると、開発者自身が、試験を増やし、デプロイの頻度を落とすよう求めます。自分たちのリリースを止めたい人はいないためです。チームが、自分たちで歯止めをかけます。ただし、同書は条件を付けます。目標を破ったときに、SREが本当にリリースを止める権限を持っていなければ、機能しません。

バジェットが主に制御するのは、リリースです。障害の主因が、リリースだからです。_SRE Workbook_の方針の例は、Googleの停止の約70%を、変更によるものとします。バジェットは、どこへ力を注ぐかの指針になります。WorkbookのSLO実装の章は、目標97%、四週間の許容失敗数が109,897件のバジェットを持つサービスを検討します。不具合のあるAPIリリースで、四時間、全リクエストが失敗したことで、14,066件、約13%を使いました。データベースサーバーが故障し、バックアップからの復元に20時間かかったことで、推定72,000件、65%を使いました。サーバー障害の方が、はるかに悪く見えます。しかし、例の会社では、サーバー障害は五年に一回なのに対し、不具合のあるリリースは年に二、三回あります。平均すると、リリースが約二倍のバジェットを使います。先に効果が出るのは、リリース手順の改善です。

一度も使わないバジェットにも、意味があります。同書によると、高すぎる目標がもたらす、柔軟性の不足と革新の遅れという代償を示します。リリースに苦労するチームは、目標を緩め、バジェットを増やす選択もできます。

実際の活用例

消費には、合計だけでなく速度があります。Workbookのアラートの章は、その速度をバーンレートと呼びます。目標に対して、どの速さでバジェットを消費するかです。バーンレート1なら、対象期間の終わりに、ちょうど使い切ります。30日間で99.9%の目標なら、同書の表は次のようになります。

```
バーンレート   エラー率   使い切るまでの時間
1             0.1%       30日
2             0.2%       15日
10            1%         3日
1,000         100%       43分
```

エラーが0.1%を超えるたびに当番を呼ぶと、短い急増でも毎回、人を起こします。それより、バーンレートで通知する方が適切です。Workbookは、複数のバーンレートを、それぞれ長短二つの期間で、同時に調べるよう勧めます。99.9%の目標に対する出発点は、30日間のバジェットの2%を一時間で使ったとき、バーンレート14.4、または5%を六時間で使ったとき、バーンレート6で当番へ通知します。10%を三日で使ったとき、バーンレート1では、チケットを起票します。長い期間の十二分の一である短い期間があるため、問題が終わって間もなく、アラートも終わります。Workbookの例では、エラーが止まると、五分後には短い期間の平均がしきい値を下回ります。一時間の平均は、一時間ずっと上回ったままです。

Prometheusのルールファイルでは、当番へ通知する二つの条件は次のように書けます。比率は、Workbookと同じ名前の記録ルールです。名前に含まれる期間について、失敗したリクエストの割合を保持します。

```
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
```

不具合のあるデプロイを、このルールで追います。14:00にリリースが、注文リクエストの2%を失敗させ始めます。バーンレートは20です。五分間の比率は、四分以内に1.44%のしきい値を超えますが、一時間の比率が届くのは、エラーが約43分続いてからです。そのため、当番への通知は14:43ごろに出ます。その時点で、月間バジェットをちょうど2%使っています。短い乱れでは誰も起こさず、持続する消費は見逃さないという、ルールのトレードオフです。14:50にロールバックすると、五分間の比率は二分以内にしきい値を下回り、通知が終わります。一時間の比率だけなら、15:07ごろまで通知を続けることになりました。

必要に見合った頻度で当番を呼ぶことは、2025年からの職務で手がけた自社製のエラー監視とトレースでも、同じ課題です。急増やリグレッションを検出し、通知にはクールダウン期間を設けるため、不具合のある一回のデプロイで、四十回も呼び出しません。通知ルールは、汎用のものではなく、そのプラットフォームに合わせました。11ある運用担当者向けビューの一つは、サービスレベル目標を扱います。

問題が起きるところ

失敗を数えることと、責任を追及することを混同します。 利用者が感じる失敗は、すべてバジェットを使いますが、すべてが、自分たちのリリースを止める理由になるとは限りません。Workbookの方針の例は、全社のネットワーク障害が原因の場合や、別チームのサービスが原因で、そのチームが自分たちのリリースを止めている場合、機能の仕事を続けられるとします。自分たちのバグや手順で使い切ったなら、信頼性を優先します。別チームが運用する依存先が原因の停止について、SLO実装の章は、続ける考え方と、それでも止める考え方の二つを挙げ、止める方が利用者の満足につながるとします。事前に選び、方針へ書き、計算に異論が出たときの判断者も決めます。方針の例では、CTOへ判断を委ねます。

エラーを、違う分類に入れます。 負荷テストの失敗や、侵入テスト担当者の拒否されたリクエストが、利用者へ影響していないのにバジェットを使います。方針の例は、誤分類を、無駄にバジェットが消える原因に挙げます。逆は、さらに悪い状態です。本物の失敗を無害なものへ分類すると、利用者が困っていても、

書類上は目標内に収まります。同じ方針は、それが判明したら、信頼性の改善を必須とします。目標の対象でない通信を指標へ含めず、消費量がおかしく見えたインシデントの後には、分類を確かめます。

大きなインシデントの代価を払い、忘れます。 リリースの凍結で買えるのは時間であり、理解ではありません。方針の例は、両者を結び付けます。単独のインシデントが四週間のバジェットの20%を超えて使うと、最優先の対応項目を少なくとも一つ含む事後検証が必要です。同じ種類の停止が、四半期で20%を超えて使うなら、次の四半期の計画へ載せる必要があります。

アラートが、決して発火しません。 99.9%の表から写したしきい値は、別の目標にそのまま使えません。Workbookによると、可用性90%では、全面停止でも、一時間にバジェットの1.4%しか使いません。一時間に2%で呼び出すルールは、停止中ずっと沈黙します。反対の極端な例として、月間目標99.999%では、全面停止が26秒続くとバジェットを使い切ります。多くの監視システムが、一つのサンプルを集めるより早い時間です。期間としきい値を目標へ合わせます。非常に高い目標には、設計だけが防御になるという、Workbookの結論を受け入れます。変更をまず利用者の1%へ出せば、同じ失敗でも、使い切るまで43分残ります。

通信量が少なく、一件の失敗で当番を呼びます。 一時間に十件のリクエストなら、一件の失敗が、その一時間のエラー率10%になります。Workbookによると、目標99.9%では、その一件で直ちに通知します。30日間のバジェットの13.9%を使うためです。一か月全体でも、七件の失敗しか許しません。Workbookの対処は、人工的な通信を作ること、監視上は複数の小さいサービスを一つにまとめること、一件の失敗が持つ意味を小さくするよう製品を変えることです。

サービスが満たせる前に、目標を上げます。 厳しい目標が必要だからと、すぐ採用すると、チームは常に緊急事態で、バジェットも常に使い切ったままです。Workbookは、代わりに、将来目指す目標を提案します。現在の目標と並べて測り、追跡しますが、サービスの準備が整うまでは対応を求めないと、方針へ明記します。

参考資料

- リスクを受け入れる — Googleの_Site Reliability Engineering_の第3章です。データに基づいてリリースを判断する方法として、エラーバジェットを紹介します。

- SLOに基づくアラート — バーンレート、複数の期間を使うアラート、通信量の少ないサービスについての、SRE Workbookの説明です。
- エラーバジェット方針の例 — リリースを止める条件、除外する停止、事後検証を求める条件を含む、完全な方針です。
- SLOの実装 — 目標からバジェットを計算し、信頼性への投資先を選ぶ方法です。
- アラートルール — 上で使うルールファイルの形式についての、Prometheusの文書です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「エラーバジェットとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く 連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/error-budget/