ERROR BUDGETS EXPLAINED: RISK & RELIABILITY IN ONE METRIC

The goal of most IT pursuits is optimization. Yet, wise professionals are well aware that perfect optimization isn't possible. No service, solution, or system will ever be completely optimized. That doesn't mean optimization isn't worth pursuing—it just means optimization goals need to take reality into account.

With systems growing increasingly complex, failure at some point, of some kind, is inevitable. This fact is the reason why so many resources are invested in failure mitigation and recovery.

As developers add new features and make changes to the system, unforeseen complications will arise. Change is daunting because of the possibility for unknown factors to cause mayhem in your system. But you can't just stop making changes. Changes are a necessity in today's competitive business world. After all, you can't win a race by standing still.

But how do you know…

- The right time for making changes?
- How much room for error the system has?
- How much testing is enough?
- When new pushes should be made?

As with most business decisions, the right metrics can make all the difference. One of the best methods for deciding the timing of system modifications is error budgeting.

What is an error budget?

Error budgets are an allowance for downtime. They account for the fact that pursuing perfection—100% uptime—is a fruitless labor that is all but impossible, not worth the expense, and stifling to the overall quality of the service going forward.

Instead, error budgets aim to take advantage of inevitable downtime or failures during normal operations. Error budgets are a tool in site reliability engineering, but you can use the concept in non-SRE environments, too.

(Compare SRE & DevOps environments.)

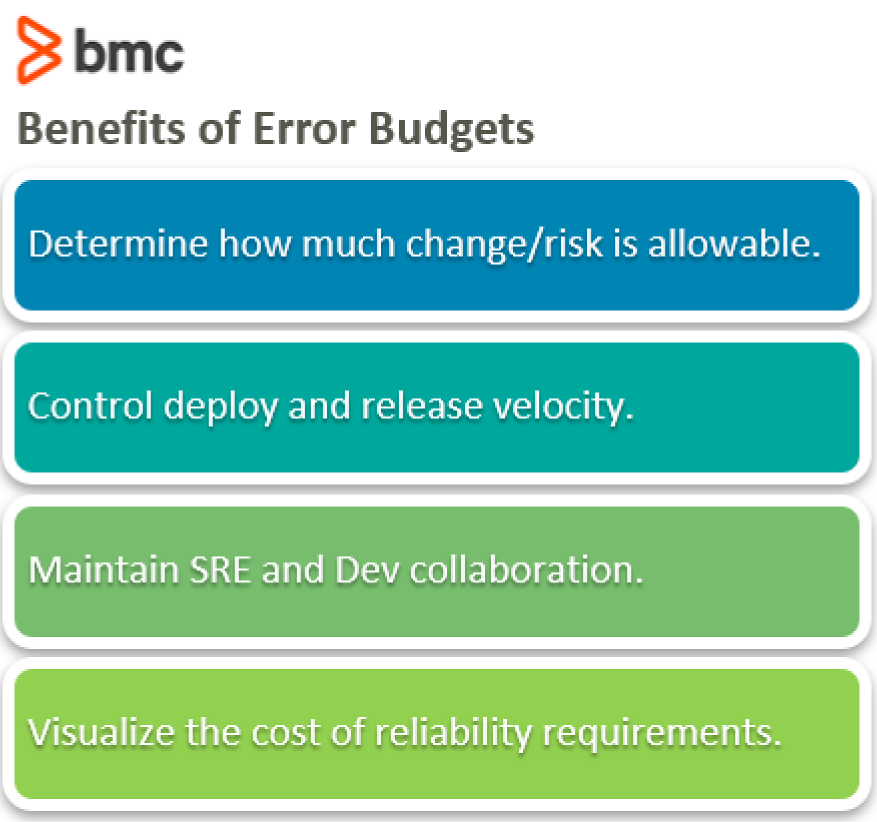
If a service's service level objectve (SLO) states that it will have a query success rate of 99.99%, then the error budget allows for failure to occur .01% of the time. By acknowledging that perfection isn't realistic, error budgets:

- Leverage failure to make room for service improvements that have a slightly reduced requirement of success.
- Ensure too much liberty isn't taken—that the overall performance of the system is maintained at an acceptable level.

An error budget can play dual roles by acting as a performance indicator and as a guideline for directing system development efforts. Here's how this works:

1. Measure and compare actual uptime against the error budget.
2. The difference determines how much downtime the service is permitted to have throughout the rest of the quarter.
3. Until the error budget is met, you can continue pushing new releases.

Benefits of error budgeting


bmc

Benefits of Error Budgets

Determine how much change/risk is allowable.

Control deploy and release velocity.

Maintain SRE and Dev collaboration.

Visualize the cost of reliability requirements.

Google uses error budgets as a way for their SRE teams and product developer teams to define in their SLO "how unreliable the service is allowed to be within a single quarter." By using error budgeting in this way, Google's teams have a clear and measurable guideline for exactly how much risk is permissible. This gives the development team leeway to:

- Make changes
- Take some risks without disadvantaging the SRE team

Error budgeting helps to control the release velocity by ensuring SLOs are met before new releases are given the go ahead. If/when SLO violations completely spend down the error budget, you'll put new releases on hold while the team performs more testing and performance improvements.

Of course, error budgets don't have to be used in such a simplistic and binary manner.

The remaining error budget functions as a cushioning for development teams to use as they see fit. By keeping tabs on how you consume the error budget, you can decide the rate of deployment—slowing down or speeding up in response to the remaining error budget.

Error budgets help keep development and SRE teams working in tandem towards the overall improvement of the service. Product development has room to take risks and increase their push velocity while there is plenty of error budget remaining. Once the error budget dries up, development is incentivized to slow down their push velocity to make time for more testing. Otherwise, they'll exceed the budget and risk delaying the release.

An error budget provides hard data for when these two teams are at an impasse regarding decisions for push velocity. The remaining error budget indicates how much risk can be safely undertaken, giving either team the ammunition they need to back up their requests for speeding up or slowing down.

Error budgets also function as a means of illustrating the costs of excessively high reliability requirements that can result in stemmed innovation and flexibility. The error budget may need to be adjusted to allow for more room for the development team to introduce new features or it may need to be tightened if its impact on overall service availability proves to be too high.

Error budgeting is risk management

With error budgeting, IT teams can leverage failure by turning it into an advantage for the quality of the product they are producing.

Error budgeting helps teams appropriately manage risk to:

- Make the most of their resources
- Have the best chance of improving the quality of the service without sacrificing too much reliability

Business leaders make informed decisions about every aspect of their company's direction by leveraging metrics that help them gain insight into daily operations and how everything fits into the bigger picture. Error budgets provide indisputable evidence regarding the impact of new pushes. They're also a metric for judging whether your development cadence is sustainable.

Error budgets are a balance between releasing new builds and maintaining an acceptable level of performance. This balance maintains the service at high rates of performance without impeding the goal of continual improvement.

Related reading

- BMC DevOps Blog
- DevOps Guide
- Lewin's 3 Stage Model of Change Explained