SRE and the art of SLOs
Nathen Harvey

Developer Advocate
@nathenharvey
(He/him)
Most important feature of any system is its reliability
Monitoring does not decide our reliability

Our Users Do
Well engineered

- Software
- Operations
- Business
How do we incentivize reliability?
100% is the **wrong** reliability target for basically **everything**

— *Benjamin Treynor Sloss*, VP 24x7, Google; Site Reliability Engineering, Introduction
A principled way to agree on the desired reliability of a service.
Error Budgets

An acceptable level of unreliability

This is a budget that can be allocated
Error budgets capture the performance and availability levels that, if barely met, would keep the typical customer happy.

“meets target error budget” ⇒ “happy customers”
“sad customers” ⇒ “misses target error budget”
Implementation Mechanics

Evaluate **performance** over a set **window**, e.g. 28 days
Remaining budget **drives prioritization** of engineering effort
Implementation Mechanics

Evaluate performance over a set window, e.g. 28 days
Remaining budget drives prioritization of engineering effort
What are the consequences when we exhaust or overspend our error budget?
Consequences may include

/ freeze **feature releases**

/ prioritize **post mortem items**

/ **automate** deployment pipelines *

/ improve **monitoring and observability**

/ require SRE **consultation**
What should we **spend** our error budget on?
Error budgets can accommodate

/ releasing new **features**

/ expected system **changes**

/ inevitable **failure** in hardware, networks, etc.

/ planned **downtime**

/ risky **experiments**
Service Level Indicator

A quantifiable measure of service reliability
Every SLI is a metric...Not every metric is an SLI

- CPU Utilization
- Number of backend hosts available
- Free memory on a node
- Uptime of a container
- Disk space available
Four Golden Signals

- Latency
- Traffic
- Errors
- Saturation
<table>
<thead>
<tr>
<th>SLI Menu</th>
<th>Request / Response</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Latency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quality</td>
</tr>
<tr>
<td></td>
<td>Data Processing</td>
<td>Coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Correctness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Freshness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Throughput</td>
</tr>
<tr>
<td></td>
<td>Storage</td>
<td>Throughput</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Latency</td>
</tr>
</tbody>
</table>

Google Cloud

@nathenharvey
2019-09-11
Remember Carmel's talk?

Provisioning customer environments
Availability

The customer environment should be provisioned successfully
Availability

The **customer environment** should be provisioned **successfully**

- How do we define **success**?
- Where is the success / failure **recorded**?
Availability

The **customer environment** should be provisioned **successfully**

- How do we define **success**?
- Where is the success / failure **recorded**?

The proportion of **valid** requests served **successfully**.
Availability

The **customer environment** should be provisioned **successfully**

- How do we define **success**?
- Where is the success / failure **recorded**?

The proportion of **valid** requests served **successfully**.
Availability

The **customer environment** should be provisioned **successfully**

- How do we define **success**?
- Where is the success / failure **recorded**?

The proportion of **HTTP POST** requests for `/environment/new` served **successfully**.
Availability

The customer environment should be provisioned successfully.

- How do we define success?
- Where is the success / failure recorded?

The proportion of HTTP POST requests for /environment/new served successfully.
Availability

The **customer environment** should be provisioned **successfully**

- How do we define **success**?
- Where is the success / failure **recorded**?

The proportion of **HTTP POST** requests for **/environment/new** that have **2XX** or **3XX** status
Availability

The customer environment should be provisioned successfully.

- How do we define success?
- Where is the success / failure recorded?

The proportion of HTTP POST requests for /environment/new that have 2XX or 3XX status measured at the load balancer.
Availability

The customer environment should be provisioned successfully

- How do we define success?
- Where is the success / failure recorded?

The proportion of HTTP POST requests for /environment/new that have 2XX or 3XX status measured at the load balancer.

Latency

The customer environment should provisioned quickly

- How do we define quickly?
- When does the timer start / stop?

The proportion of HTTP POST requests for /environment/new that send their entire response within Xms measured at the load balancer.
SLI: \( \left( \frac{\text{good events}}{\text{valid events}} \right) \times 100\% \)
Services Need SLOs
Service Level Objectives

Set a reliability target for an SLI
What goals should we set for the reliability of our journey?

Your objectives should have both a **target** and a **measurement window**

<table>
<thead>
<tr>
<th>Service</th>
<th>SLI Type</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provision</td>
<td>Availability</td>
<td>99.95% successful in previous 28d</td>
</tr>
<tr>
<td>Provision</td>
<td>Latency</td>
<td>90% of requests &lt; 5000ms in previous 28d</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
What goals should we set for the reliability of our journey?

Your objectives should have both a **target** and a **measurement window**

<table>
<thead>
<tr>
<th>Service</th>
<th>SLI Type</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provision</td>
<td>Availability</td>
<td>99.95% successful in previous 28d</td>
</tr>
<tr>
<td>Provision</td>
<td>Latency</td>
<td>90% of requests &lt; 5000ms in previous 28d</td>
</tr>
<tr>
<td>Provision</td>
<td>Latency</td>
<td>50% of requests &lt; 2000ms in previous 28d</td>
</tr>
</tbody>
</table>
Summary

#1 Define your problem space - SLOs and SLIs

#2 Make your system as reliable as it must be but no more

#3 Error budgets are your primary basis of communication

#4 SLOs are not set in stone forever

#5 The team relationship has to be strong to make this practice work
https://landing.google.com/sre/books/
SOFTWARE DEVELOPMENT

Lead Time

Deployment Frequency

SOFTWARE DEPLOYMENT

Change Fail

Time to Restore

SERVICE OPERATION

Availability

FOUR KEY METRICS
The industry continues to improve, particularly among the elite performers.
Delivering software quickly, reliably, and safely is at the heart of technology transformation and organizational performance.
The best strategies for scaling DevOps in organizations focus on structural solutions that build community.
Cloud continues to be a differentiator for elite performers and drives high performance.
Productivity can drive improvements in work/life balance and reductions in burnout, and organizations can make smart investments to support it.
There’s a right way to handle the change approval process, and it leads to improvements in speed and stability and reductions in burnout.
cloud.google.com/devops
Focus resources on what is currently holding you back, then iterate: Identify constraints and choose the next target.
Nathen Harvey

Developer Advocate
@nathenharvey
(He/him)