9 Things you should Do When starting to use SLOs

Sal Furino
Customer Reliability Engineer
@sfurino
He/Him
/in/salvatore-furino/
Who has read these?
The SRE books are directionly correct.

How to Achieve “SRE” still left up to the reader.
SLOs are Hard to get Right
SLOs are Hard to get Right

Common questions

● Which metrics best instrument the service?
● What is a reasonable objective for this service?
● How should the error budget be used?
● Who else needs to understand these signals?
What are the 9 things?

Understanding the systems
● Measure meaningful things
● Success rates > Error rates
● Raw data for SLOs

Understanding expectations of system behavior
● Examine historical performance
● Define and take actions
● Different time windows for different folks

Recognizing when a system doesn’t match those expectations
● Enrich your dashboards with context
● Document your SLOs
● Adopt open standards
Understanding Systems
Measure Meaningful Things

Shine Light On...

Don’t Measure

CPU Utilization

Disk IO

What Matters to Users

Customer Journeys

Interactions
Measure Meaningful Things

“Fortress Bank”

Industry:
Finance - Card Processor

Motivation:
Contractual and regulatory impacts

Goal:
Insure reliability for quicker person to person payments
Journey:
As a user, I want to send money to quickly to a 3rd party

Expectation:
I should receive notification that the payment is being processed

Dependencies:
3rd party application gateway, clearing house, screening
Measure Meaningful Things

“Fortress Bank”
Measure Meaningful Things
“Fortress Bank” - Level 1 Ack

Legend:
- External Service
- Internal Service
- Clients

API Interface

API 2

Start

Service Connect API

Direct API

Manage ment API

Clients

Synthetic Clients
Measure Meaningful Things
"Fortress Bank" - Level 2 ack
Don’t Define Bad Behavior

- Maybe Easier
- Has a limited scope
- Known undesired problems

Define Success

- Measure Success Rate
- Everything else is uncertain
Measure Success Rate
“Fortress Bank” - Level 1 Ack

AVAILABILITY

The proportion of requests to Service Connect API that have 2XX, 3XX or 4XX status measured at the Load Balancer
Provide Raw Data for SLOs

Don’t Down Sample

avg(query)
percentile(95, query)
sum(query)

Simple Queries

Provide High Fidelity Information to Your SLOs
Provide Raw Data for SLOs

Streams “R” Us

Industry:
   Entertainment - Streaming Video

Motivation:
   Contractual requirement
   Competitive requirements

Goal:
   Insure low latency streaming video to consumers
Streams “R” Us
Streams “R” Us
Understanding Expectations of Systems Behavior
Look Back to Look Forward

Don’t say

“All Our Services Need
To Have 99.999%
Or Better Reliability”
Look Back to Look Forward

Ask your teams:

- How reliable does the service need to be to be competitive?
- How much un-reliability can we accept with the current design?
- How much are we willing to spend to achieve those goals?
Don’t say

“All Our Services Need To Have 99.999% Or Better Reliability”

Do

Benchmark Achievable Reliability targets
Capture SLI Activity and Calculate Error Budget After a recent incident
Look Back to Look Forward
“Fortress Bank” - Level 1 Ack

AVAILABILITY

The proportion of requests to Service Connect API that have 2XX, 3XX or 4XX status measured at the Load Balancer

<table>
<thead>
<tr>
<th>Percentile</th>
<th>ms</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.9</td>
<td>2000</td>
<td>SLA</td>
</tr>
<tr>
<td>99</td>
<td>1700</td>
<td>Something is wrong</td>
</tr>
<tr>
<td>95</td>
<td>1500</td>
<td>Starting to lag</td>
</tr>
<tr>
<td>90</td>
<td>1000</td>
<td>Okay</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>lightning</td>
</tr>
</tbody>
</table>
Look Back to Look Forward
“Fortress Bank” - Level 2 ack

<table>
<thead>
<tr>
<th>Percentile</th>
<th>minutes</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>95</td>
<td>30</td>
<td>SLA</td>
</tr>
<tr>
<td>92</td>
<td>25</td>
<td>Something is wrong</td>
</tr>
<tr>
<td>90</td>
<td>15</td>
<td>Okay</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>lightning</td>
</tr>
</tbody>
</table>
Observability Without Action is Just Storage

Don’t
Set and forget Dashboards for the sake of Dashboards

Do
Codify Your Reliability Targets
Set Automated Action
Alert when necessary
Look Back to Look Forward
“Fortress Bank” - Level 1 Ack

**Availability**

The proportion of requests to Service Connect API that have 2XX, 3XX or 4XX status measured at the Load Balancer

<table>
<thead>
<tr>
<th>Percentile</th>
<th>ms</th>
<th>label</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.9</td>
<td>2000</td>
<td>SLA</td>
<td>Full Outage Crisis Mode</td>
</tr>
<tr>
<td>99</td>
<td>1700</td>
<td>Something is wrong</td>
<td>Page on Call</td>
</tr>
<tr>
<td>95</td>
<td>1500</td>
<td>Starting to lag</td>
<td>Service Now Ticket</td>
</tr>
<tr>
<td>90</td>
<td>1000</td>
<td>Okay</td>
<td>Warn / Send Slack message</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>lightning</td>
<td>Steady State N/A</td>
</tr>
</tbody>
</table>
Different Time Windows for Different Folks

Don’t say
“SLOs Are Only For Operations and Site Reliability Engineers”

Do
Different Groups Take Different Actions to Reliability
Same SLIs But Different Time Windows
Different Time Windows for Different Folks

Don’t say

“SLOs Are Only For Operations and Site Reliability Engineers”

Do

24h/48h – SRE/OPs
14D/2W – Dev teams
Monthly / Quarterly – Arch / Leadership
Recognizing when a system doesn’t match those expectations
Enrich Dashboards with Contextual Information

Don’t

Do

Annotations
Badges
Dependency Mapping
Document your SLOs

Don’t say

“What is This Service?”
“What’s Responsible For it?”
“Where is it Deployed?”
“How Do We Fix it?”
“Why is This Metric Important?”

Do

Document the WWWWHW For the SLOs Existence
Track and Version With Your SLO Definitions
SLODLC

- **Prepare Business Case**
  - Prioritize User Journeys & Expectations
  - Analyze Dependencies
  - Observe System Behavior

- **Identify Stakeholders**

- **Define Desired Outcomes**
  - Define Meaningful SLI
  - Define Achievable SLO
  - Establish Error Budgets
  - Define Aspirational SLO

- **DISCOVER**
  - Collect SLI
  - Publish SLO Goals
  - Enforce Error Budget Policy

- **DESIGN**
  - Gain SLO Insights

- **IMPLEMENT**
  - Respond to Error Budget Events & Alerts
  - Ensure SLI Data Cleanliness
  - Adjust SLO Targets
  - Gain SLO Insights

- **OPERATE**

- **INITIATE**

- **Review Periodically**

- **Align with Service & Software Development and Business Life Cycles**

- **Listen and Engage Users and Stakeholders**

- **Share Learnings from SLO Journey to Align Practices and Standards**
SLODLC Discovery Worksheet

Service Name:
SLODLC Adoption:
SLO Adoption Leader:
Worksheet Owner:
Document Status:
Related Docs:

Discovery Worksheet Scope

- Service
- Prioritize User Journeys & Expectations
- Analyze Dependencies
- Observe System Behavior

How to work with Discovery Worksheet

1. Please walk through each point in the table
Your Problems Aren’t Unique

Don’t
Operate in a Silo

Do
Embrace open standards
Your Problems Aren’t Unique

**Standards:** Open Telemetry & Open SLO

**Frameworks:** DORA, R9Y.dev, SLODLC

**Communities:** LFI, Art of SLOs

**Slacks:** DevOpsDays, SRECon, HangOps, Rands Leadership, Reliability.org

**Socials:** /r/sre, /r/devops, Tech Twitter?, Mastodon, bluesky?, etc
Thank you!

Salvatore Furino
CRE - Customer Reliability Engineer
I HAVE A QUESTION.
WELL, LESS OF A QUESTION
AND MORE OF A COMMENT.
I GUESS IT'S LESS OF A COMMENT
AND MORE OF AN UTTERANCE.
REALLY IT'S LESS AN UTTERANCE,
MORE AN AIR PRESSURE WAVE.
IT'S LESS AN AIR PRESSURE WAVE
AND MORE A FRIENDLY HAND WAVE.
I GUESS IT'S LESS A FRIENDLY
WAVE THAN IT IS A FRIENDLY BUG.
I FOUND THIS BUG AND NOW WE'RE
FRIENDS. DO YOU WANT TO MEET IT?

Source: xkcd
Your Problems Aren’t Unique