Welcome!
Agenda

/ Terminology
/ Why your services need SLOs
/ Spending your error budget
/ Choosing a good SLI
/ Developing SLOs and SLIs
Service Level Indicator

A quantifiable measure of service reliability
Service Level Objectives

Set a **reliability target** for an SLI
Users? Customers?

Customers are users who directly pay for a service.
Services Need SLOs
The most important feature of any system is its **reliability**.
A principled way to argue about the desirable reliability of a service
What is "reliable"?

Think about Netflix, Google Search, Gmail, Twitter... how do you tell if they are ‘working’?
Services need target SLOs that capture the performance and availability levels that, if barely met, would keep the typical customer happy.

“meets target SLO” ⇒ “happy customers”
“sad customers” ⇒ “misses target SLO”
"HTTP GET / ..."
When do we need to make a service **more reliable**?
“100% is the wrong reliability target for basically everything.”

Benjamin Treynor Sloss, Vice President of 24x7 Engineering, Google
Measure SLO achieved & try to be *slightly* over target...

...but don’t be too much better or users will depend on it!
Error Budgets

An SLO implies an acceptable level of unreliability

This is a budget that can be allocated
Implementation Mechanics

Evaluate SLO performance over a set window, e.g. 28 days
Remaining budget drives prioritization of engineering effort
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**
Benefits of error budgets

- Common incentive for devs and SREs
  Find the right balance between innovation and reliability

- Dev team can manage the risk themselves
  They decide how to spend their error budget

- Unrealistic reliability goals become unattractive
  These goals dampen the velocity of innovation

- Dev team becomes self-policing
  The error budget is a valuable resource for them

- Shared responsibility for system uptime
  Infrastructure failures eat into the devs’ error budget
Activity

Reliability Principles
Dear Colleagues,

The negative press from our recent outage has convinced me that we all need to take the reliability of our services more seriously. In this open letter, I want to lay down three reliability principles to guide your future decision making.
The first principle concerns our users. We let them down, and they deserve better. They deserve to be happy when using our services!

Our business must...

1. ... compensate our users for future outages.
2. ... find ways to help our users tolerate or enjoy future outages.
3. ... strive to exceed our users expectations of reliability.
4. ... build the features that make our users happy faster.
5. ... never fail our users again.
The second principle concerns the way we build our services. We have to change our development process to incorporate reliability.

**1.** ... choose to fail fast and catch errors early through rapid iteration.

**2.** ... have ops engage in the design of new features to reduce risk.

**3.** ... only release new features publicly when they are shown to be reliable.

**4.** ... build and release software in small, controlled steps.

**5.** ... reduce development velocity when our systems are unreliable.

Our business must...
The third principle concerns our operational practices. What we're doing today isn't working; we have to do things differently to improve!

Our business must...

1. ... share responsibility for reliability between ops and dev teams.
2. ... tie operational response and prioritization to a reliability goal.
3. ... make our systems more resilient to failure to cut operational load.
4. ... give ops a veto on all releases to prevent failures reaching our users.
5. ... route negative complaints on twitter directly to ops pagers.
To put these principles into practice, we are going to borrow some ideas from Google! The next step is to define some SLOs for our services and begin tracking our performance against them.

Thanks for reading!

*Eleanor Exec, CEO*
Break!
Choosing a Good SLI
SLI provides poor signal-to-noise ratio. Variance in metric obscures deterioration.

SLI provides good signal-to-noise ratio. Metric deterioration correlates with outage.
<table>
<thead>
<tr>
<th>SLI Menu</th>
</tr>
</thead>
</table>
| Request / Response | Availability  
Latency  
Quality |
| Data Processing | Coverage  
Correctness  
Freshness  
Latency |
| Storage | Durability |
SLI: \( \left( \frac{\text{good events}}{\text{all events}} \right) \times 100\% \)
1–3 SLIs* per user journey
what performance does the business need?
user expectations are strongly tied to past performance
continuous improvement
Developing SLOs and SLIs
Our Game: Tribal Thunder

- Website
- API Server
- Leader Boards
- User Profiles
- Game Servers
- Leaderboard Generation
- Load Balancer
**SomeUser's Profile**

**Tribe Name:** Tribe of Frog

**Leader Name:** SomeUser

**Email Address:** user@example.com

---

**Tribe Score:** 31337

1. Tri-Bool 65535
2. Tri Repetae 61995
3. Triassic Five 52391
4. Tricksy Hobbits 37164
5. Tribe of Frog 31337
6. Trite Examples 29243

---

[https://tribalthundergame.com/profile/someuser](https://tribalthundergame.com/profile/someuser)
Loading a Profile Page

Website

API Server

Load Balancer

Leader Boards

User Profiles

Game Servers

Leaderboard Generation
<table>
<thead>
<tr>
<th>SLI Menu</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request / Response</td>
<td>Availability</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
</tr>
<tr>
<td></td>
<td>Quality</td>
</tr>
<tr>
<td>Data Processing</td>
<td>Coverage</td>
</tr>
<tr>
<td></td>
<td>Correctness</td>
</tr>
<tr>
<td></td>
<td>Freshness</td>
</tr>
<tr>
<td></td>
<td>Throughput</td>
</tr>
<tr>
<td>Storage</td>
<td>Durability</td>
</tr>
</tbody>
</table>
Measurement Strategies

Application Level Metrics
Server-side Logs
Frontend Infra Metrics
Synthetic Clients/Data
Client-side Instrumentation
Availability

The profile page should load successfully

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

Percentage of HTTP GET requests for /profile/{user} or /profile/{user}/avatar that have 2XX, 3XX or 4XX (excl. 429) status measured at the load balancer

Latency

The profile page should load fast

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

Percentage of HTTP GET requests for /profile/{user} that send their entire response within Xms measured at the load balancer
Activity

Postmortem Analysis
**Availability**

Percentage of **HTTP GET** requests for /profile/{user} or /profile/{user}/avatar that have **2XX, 3XX** or **4XX (excl. 429)** status measured at the load balancer.

**and**

Percentage of **HTTP GET** requests for /profile/prober_user and all linked resources that have **200** status and "ProberUser" in HTML measured by a **black-box prober** every 5s.

**Latency**

Percentage of **HTTP GET** requests for /profile/{user} that send their **entire response within Xms** measured at the load balancer.
Do the SLIs cover the failure modes?

- Black Box Prober
- Availability
- Latency
- Load Balancer
- Website
- API Server
- User Profiles
- Leader Boards
- Game Servers
- Leaderboard Generation
Activity

Define SLO Targets
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>SLO Type</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web: User Profile</td>
<td>Availability</td>
<td><strong>99.95% successful</strong> in <strong>previous 28d</strong></td>
</tr>
<tr>
<td>Web: User Profile</td>
<td>Latency</td>
<td><strong>90% of requests &lt; 500ms</strong> in <strong>previous 28d</strong></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Break!
Workshop: let's develop some more SLIs and SLOs!

For each critical user journey, stack-ranked by business impact

1. Choose an SLI specification from the menu
2. Refine the specification into a detailed SLI implementation
3. Walk through the user journey and look for coverage gaps
4. Set aspirational SLOs based on business needs
Thanks!

Please fill in the feedback form

https://goo.gl/HGtrSc