Dynatrace AIOps Forum ‘22

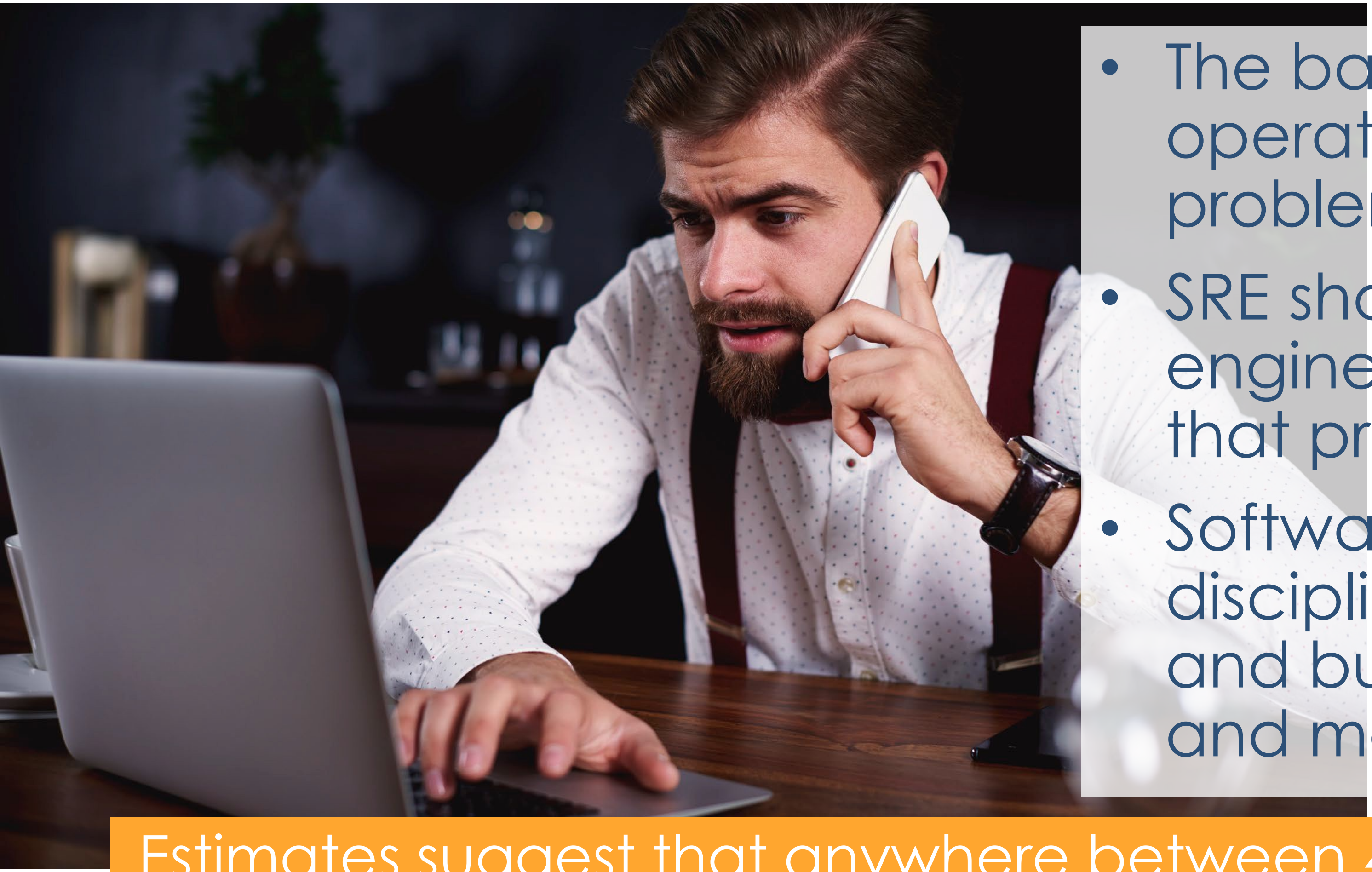
How to measure happiness?
Getting started with SLO’s

Lauri Humina, CDO
lauri.humina@wakaru.fi
0408310388

Wakaru Oy
www.wakaru.fi

SRE Principles & Practices

#1 Operations Is a Software Problem



- The basic tenet of SRE is that doing operations well is a software problem
- SRE should therefore use software engineering approaches to solve that problem
- Software engineering as a discipline focuses on designing and building rather than operating and maintaining

Estimates suggest that anywhere between 40% and 90% of the total cost of ownership are incurred after launch

Wakaru.

#2 Service Levels

- A Service Level Objective (SLO) is an availability target for a product or service (this is never 100%)
- In SRE services are managed to the SLO

SLOs need consequences if they are violated

Wakaru.

#3 Toil

- Any manual, mandated operational task is bad
- If a task can be automated then it should be automated
- Tasks can provide the "wisdom of production" that will inform better system design and behavior

SREs must have time to make tomorrow better than today

#4 Automation

- Automate what is currently done manually
- Decide what to automate, and how to automate it
- Take an engineering-based approach to problems rather than just toiling at them over and over
- This should dominate what an SRE does
- Don't automate a bad process – fix the process first

SRE teams have the ability to regulate their workload

#5 Reduce the Cost of Failure

- Late problem (defect) discovery is expensive so SRE looks for ways to avoid this
- Look to improve MTTR (Mean Time to Recover/Repair)
- Smaller changes help with this
- Canary deployments

Failure is an opportunity to improve

Wakaru.

#6 Shared Ownership

- SRE's share skill sets with product development teams
- Boundaries between “application development” and “production” (Dev & Ops) should be removed
- SRE's "shift left" and provide "wisdom of production" to development teams

Incentives across the organization are not currently aligned

SLO's – Service Level Objectives

What is an SLO?

- An SLO (“Service Level Objective”) is a goal for how well a product or service should operate
- SLO’s are tightly related to the user experience – if SLO’s are being met then the user will be happy
- Setting and measuring service level objectives is a key aspect of the SRE role
- The most widely tracked SLO is availability
- Products and services could (and should) have several SLO’s

SLO’s are about making the user experience better

Wakaru.

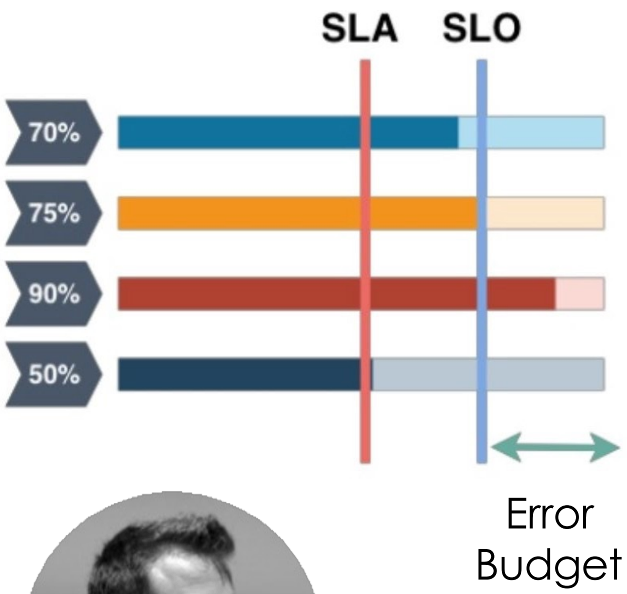
SLO's Are for Business

"Before getting into the technical details of a SLO, it is important to start the conversation from your customers' point of view: what promises are you trying to uphold?"

Ben McCormack, VP Operations, Evernote

Wakaru.

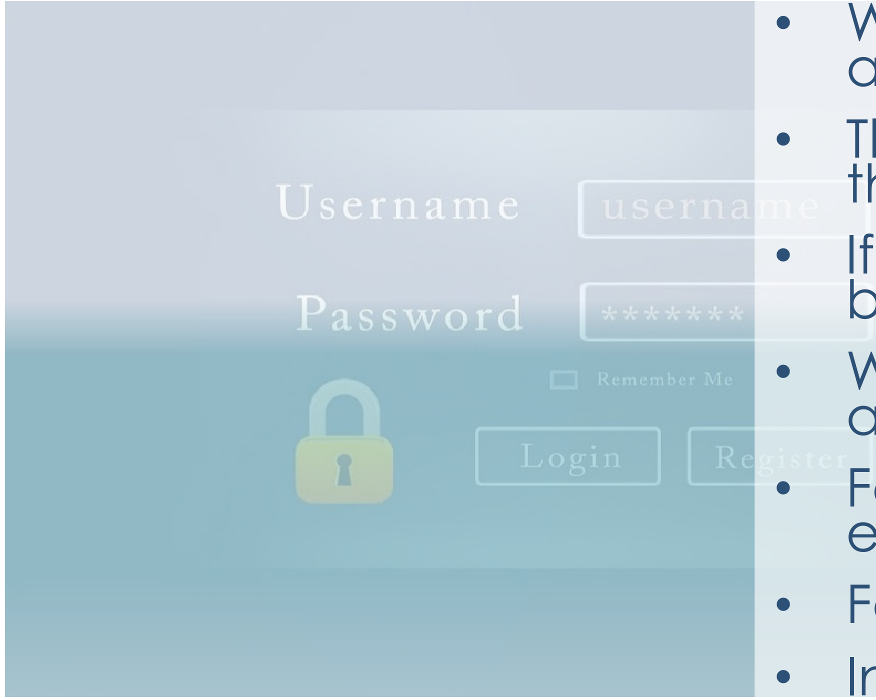
Example 1: SLO's & Error Budgets



Yaroslav Molochko
SRE Team Lead
AnchorFree

- We decide that 99.9% of web requests (www.....) per month should be successful – this is a "service level objective"
- If there are 1 million web requests in a particular month, then up to 1,000 of those are allowed to fail – this is an "error budget"
- Failure to hit an SLO <u>must</u> have consequences – if more than 1,000 web requests fail in a month then some remediation work must take place – this is an "error budget policies"

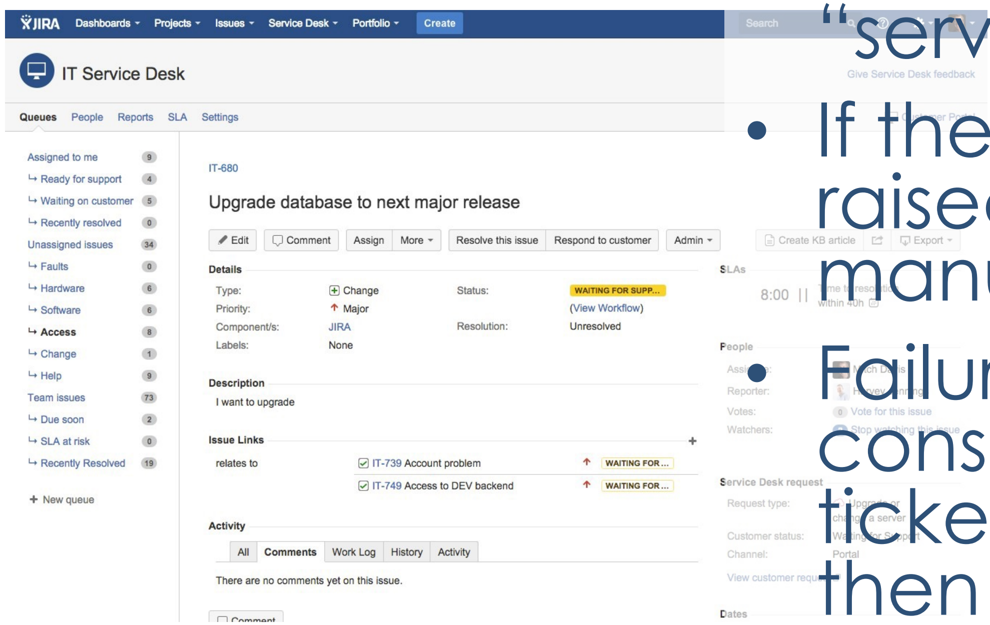
Example 2: SLO's & Error Budgets



- Our service has an average login rate of 1,000 per hour in a rolling 31-day period (month) – or 744,000 per month (31 * 24 * 1000)
- We want 99% of logins each month to be successful - this is a "service level objective"
- This equates to "losing" roughly 7,440 logins a month – this is the "error budget"
- If more 7,440 logins are lost in a month then we have breached the error budget.
- We use a service level indicator (SLI) to tell us how many actual logins we get in a month.
- For a particular month our actual logins were 726,560 - exceeding our error budget
- Failure to hit an SLO <u>must</u> have consequences
- In this case we instigated a business protection period preventing new releases – this is the "error budget policy"

Wakaru.

Example 3: SLO's & Error Budgets



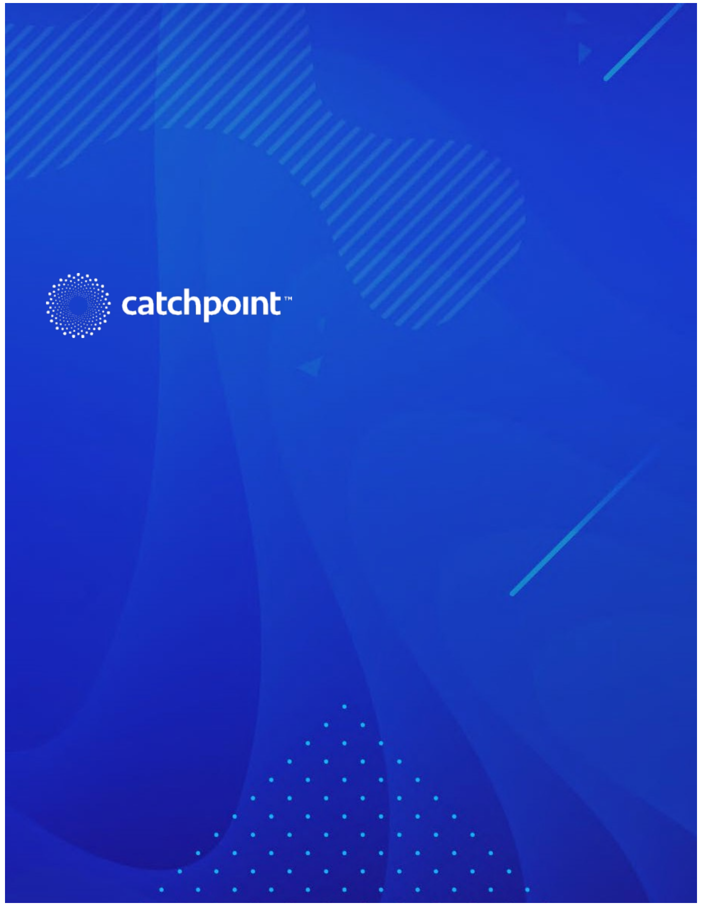
- We decide that 75% of support tickets must complete automatically – this is a "service level objective"
- If there are 1,000 new support tickets raised each month 250 can be handled manually – this is an "error budget"
- Failure to hit an SLO must have consequences – if more than 250 support tickets in a month require manual effort then some engineering work must take place – this is an "error budget policies"

Wakaru.


"SLO's are the most important component of SRE."
Lyon Wong, co-founder, Blameless - the SRE Platform

Adoption of SLO’s

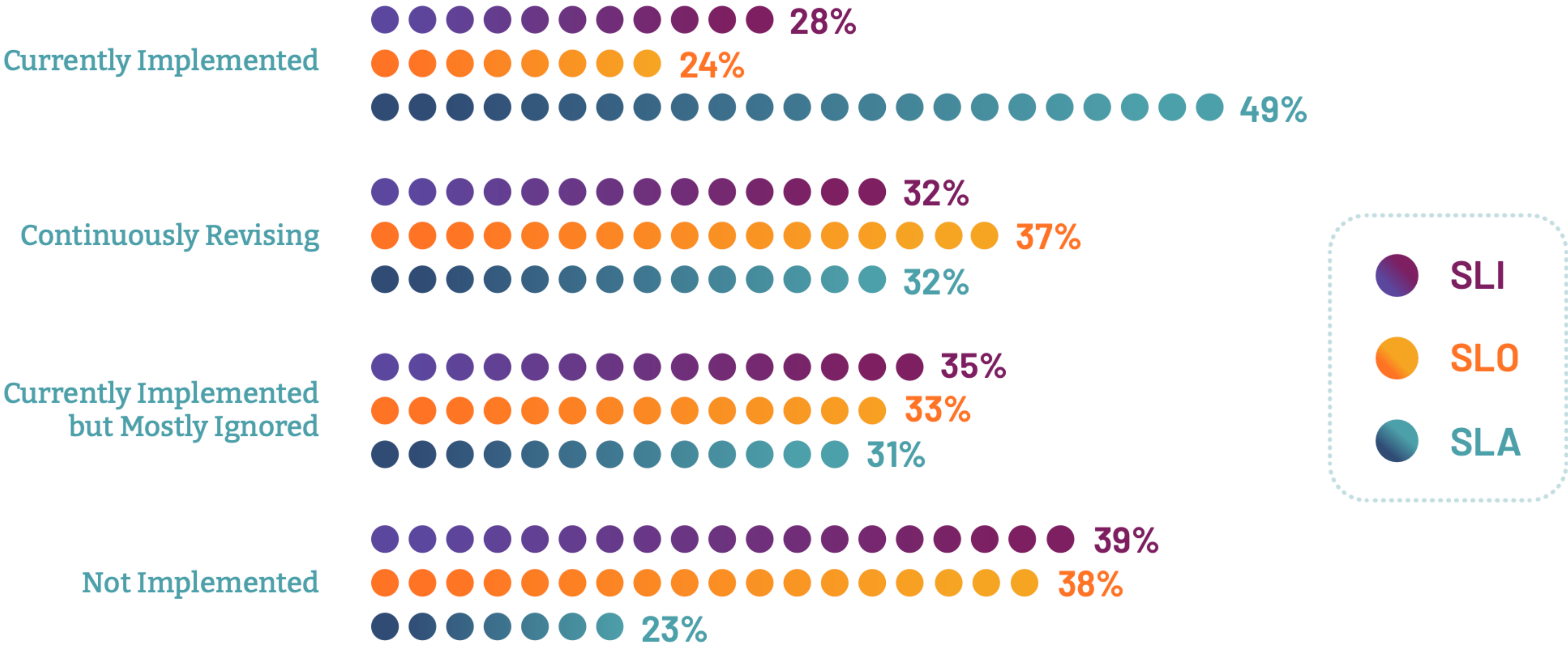
According to the 2019 Catchpoint SRE Survey the most popular SLO’s are:	
Availability	**72%**
Response time	**47%**
Latency	**46%**
We don’t have SLOs	**27%**



Global SRE Pulse 2022

SLA, SLO, SLI ... DIFFERENT USAGE LEVELS

Which tools or mechanisms do you use today within your SRE practice or team?



Dynatrace state of SRE 2022

Data overload stands in the way of setting SLOs

Despite the growing use of SLOs 99% of SREs say there are challenges to defining and creating them. However, these challenges are mostly tactical, and therefore are relatively easy to solve with the right solutions in place.

For their more strategic challenges, SREs should invest time in keeping up to date with industry best practices through sources such as Google's SRE Handbook. Continually reviewing what competitors and peers are using as their benchmarks can help to develop a deeper understanding of SLOs.

99% of SREs say they encounter challenges when defining and creating SLOs.

What are the biggest challenges your teams experience to **define and create SLOs?**

- **64%** Too many data sources, difficulty in synthesizing disparate data
- **54%** Too many metrics, difficulty in finding the most relevant ones for a particular service
- **36%** Monitoring tool doesn't allow me to easily define and track SLO performance history
- **22%** Don't know what makes a good SLO
- **18%** Don't know what metrics to track
- **18%** Don't know how to evaluate an SLO
- **16%** Don't know how/where to get started

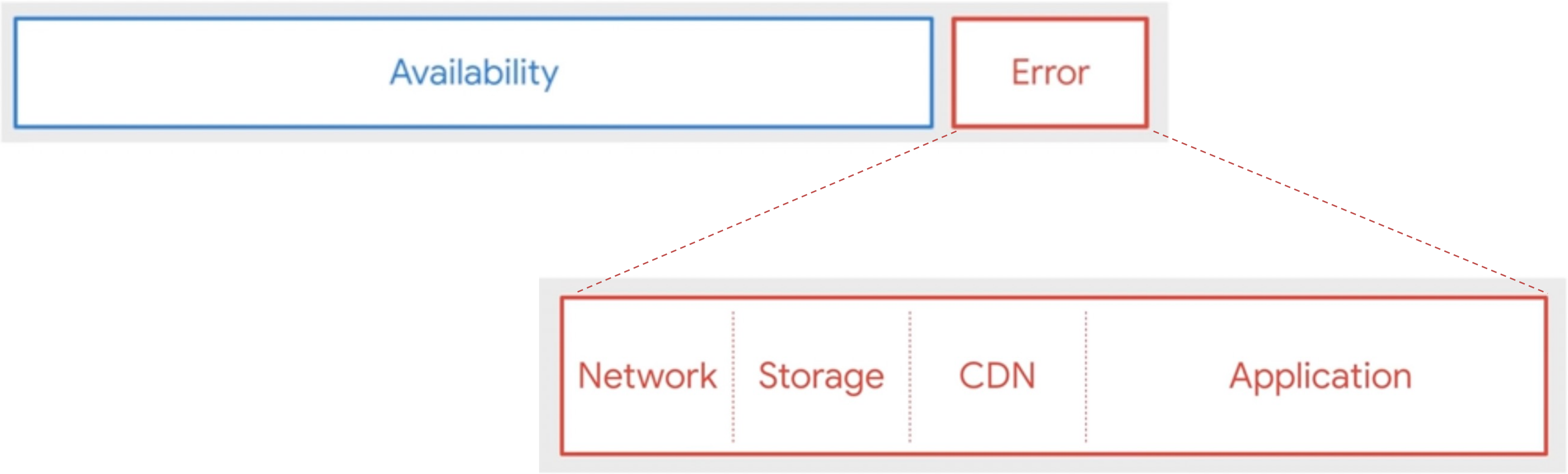
Error Budgets

“100% is the wrong reliability target for basically everything.”

Benjamin Treynor Sloss, VP 24x7 Engineer at Google

Error Budgets

How much risk a service is willing to tolerate.



Wakaru.

Error Budgets

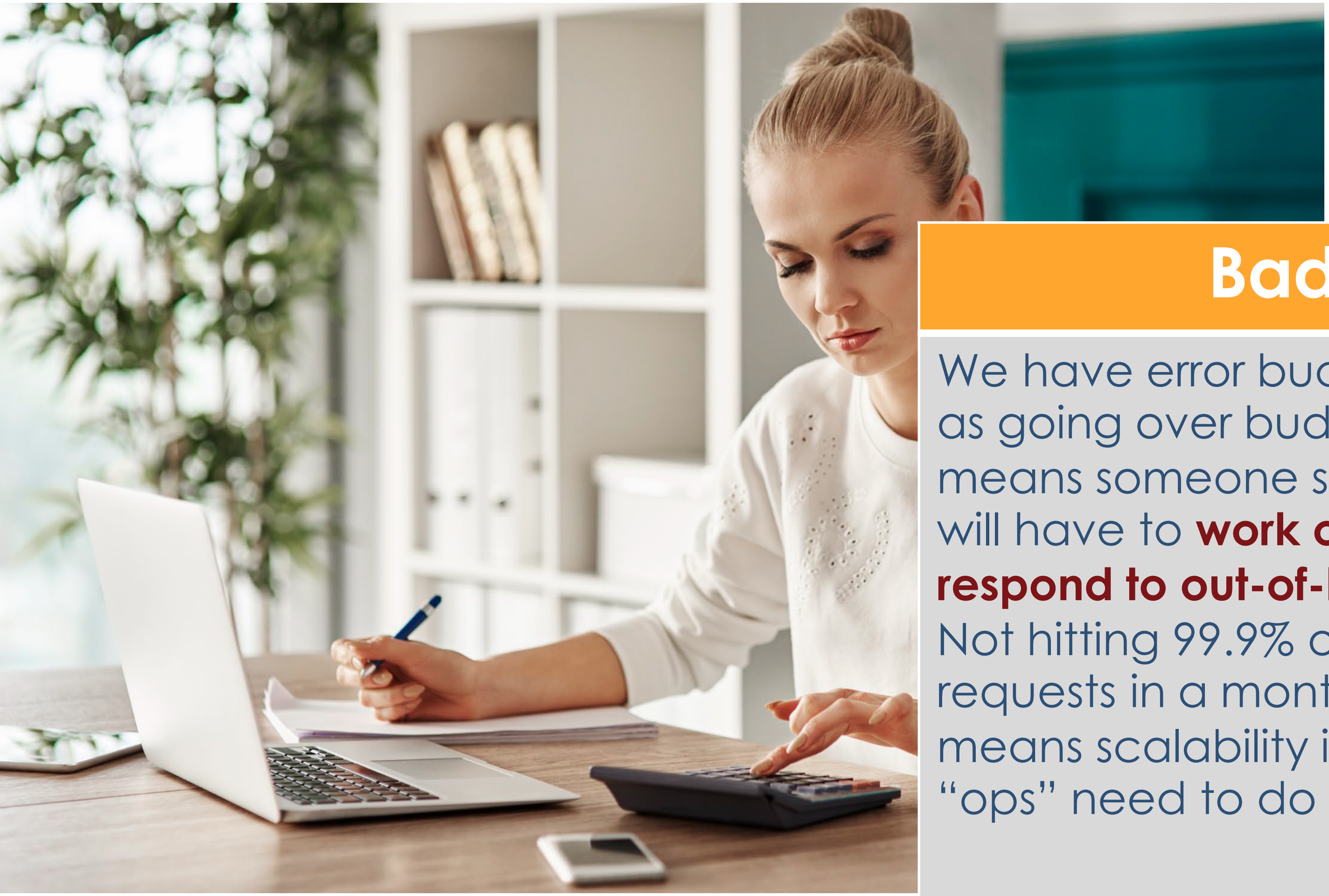
How much down time is allowed for service issues?

- SLO's introduce a constraint on the amount of time available
- A service with a "three-nines" availability SLO requires all issues in a month to be fixed inside 43 minutes
- This time includes issue identification, alerting, messaging, triage and fix

	per year	per quarter	per month	per week	per day	per hour
90%	36.5 days	9 days	3 days	16.8 hours	2.4 hours	6 minutes
95%	18.25 days	4.5 days	1.5 days	8.4 hours	1.2 hours	3 minutes
99%	3.65 days	21.6 hours	7.2 hours	1.68 hours	14.4 minutes	36 seconds
99.5%	1.83 days	10.8 hours	3.6 hours	50.4 minutes	7.20 minutes	18 seconds
99.9%	8.76 hours	2.16 hours	43.2 minutes	10.1 minutes	1.44 minutes	3.6 seconds
99.95%	4.38 hours	1.08 hours	21.6 minutes	5.04 minutes	43.2 seconds	1.8 seconds
99.99%	52.6 minutes	12.96 minutes	4.32 minutes	60.5 seconds	8.64 seconds	0.36 seconds

You can see why appropriate SLO's for services are so important

Error Budgets – Good and Bad



Bad	Good
We have error budgets in SRE as going over budget usually means someone somewhere will have to **work over-time or respond to out-of-hours issues**. Not hitting 99.9% of HTTP requests in a month usually means scalability issues so "ops" need to do something	On the other hand SRE practices encourage you to strategically **burn the budget to zero** every month, whether it's for feature launches or architectural changes. This way you know you are running as fast as you can (velocity) without compromising availability

Wakaru.

Error Budgets – Fixed?

- But watch out – high-risk deployments or large "big-bang" changes have more likelihood of issues and therefore more chance of the error budget being blown
- This should encourage the Lean preference for small changes ("smaller batch size") to stay within the error budget.
- In some cases the error budget may need to change to accommodate complex releases but this needs to be agreed between Dev and Ops and the Business

Error Budget Policies

Consequences

"There will be <u>no</u> new feature launches allowed. Sprint planning may only pull post-mortem action items from the backlog. Software Development Team must meet with SRE Team daily to outline their improvements."
Jennifer Petoff, Google

Wakaru.

CASE STORY: Evernote

"We wanted to ensure we initially focused on the most important and common customer need: the availability of the Evernote service for users to access and sync their content across multiple clients. Our SLO journey started from that goal."

"We needed to ensure the move to GCP (Google Cloud Platform) did not dilute or mask our commitment to our users."

Ben McCormack
VP Operations

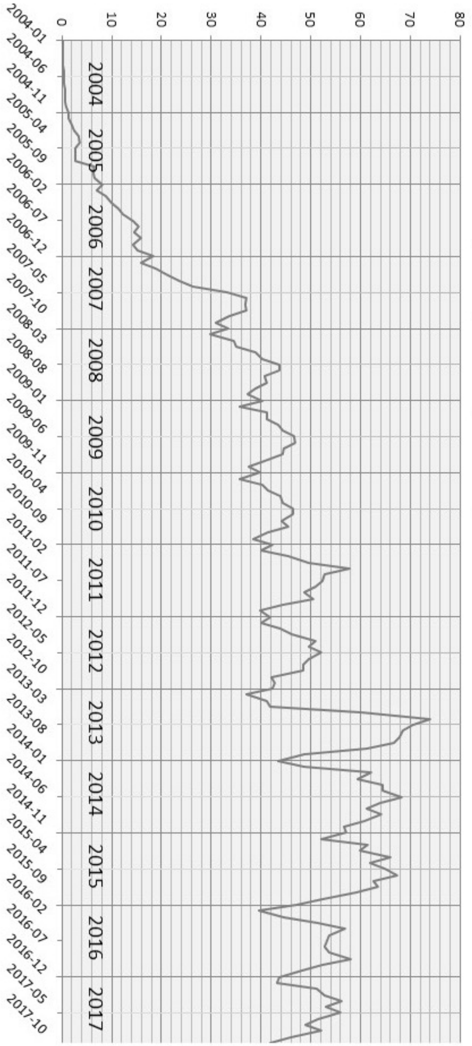
Benefits

- Consistent focus on the user experience, whilst obtaining the benefits of cloud adoption
- Join clarity around service availability and downtime
- Monitoring the right things, from a user perspective

SLI's – Service Level Indicators

Wakaru.

Let's Revisit an Earlier Example



- We decided that 99.9% of web requests (www.....) per month should be successful – this was the "service level objective"
- If there were 1 million web requests in a particular month, then up to 1,000 of those were allowed to fail – this was the "error budget"
- **In this example the "service level indicator" (SLI) is "web requests" so we need a way to track and record this data**

SLI Measurement

While many numbers can function as an SLI, it is generally recommended to treat the SLI as the ratio of two numbers: the number of good events divided by the total number of events.

For our example this is:

- Number of successful (HTTP) web requests / total (HTTP) requests (success rate)

SLO's & SLI's

We use monitoring tools to measure SLI's constantly, aggregating across suitable time periods

Our SLO's are what we expect - monitoring our SLI's will tell us if we are meeting a SLO or not – they also tell us how much of our error budget is left (if any)

SLO’s, SLI’s & Observability

- SLO’s are from a user perspective and help identify what is important
  - E.g. 90% of users should complete the full payment transaction in less than one elapsed minute
- SLI’s give detail on how we are currently performing
  - E.g. 98% of users in a month complete a payment transaction in less than one minute
- Observability gives use the normal state of the service
  - 38 seconds is the “normal” time it takes users to complete a payment transaction when all monitors are healthy

Wakaru.

CASE STORY: Kudos Engineering

"Choosing an SLI normally involves a bit of a discussion around where you should measure your objective. For example, we could choose to measure our SLI from the web server logs. However if do that we will be missing requests that do not get to the application, like failures on the load balancer or uncaught exceptions in the application"

....

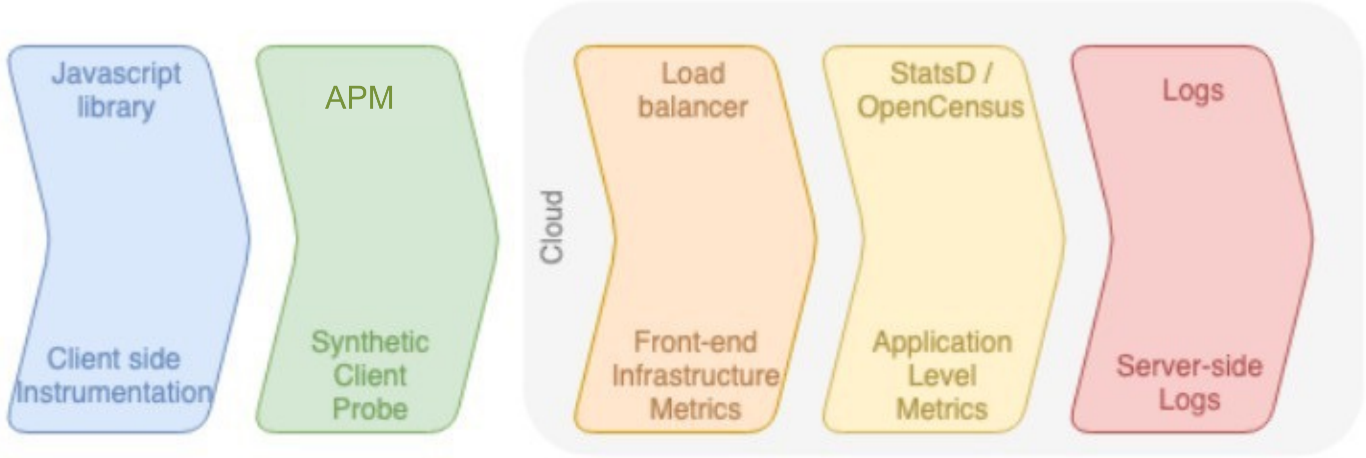
Tim Little

We started by auditing all the services that were in production, what they are being used for and most importantly how resilient and reliable they need to be. We also created a pre-launch checklist to help consider SLOs, alerting and disaster recover at inception of a new service.

Benefits

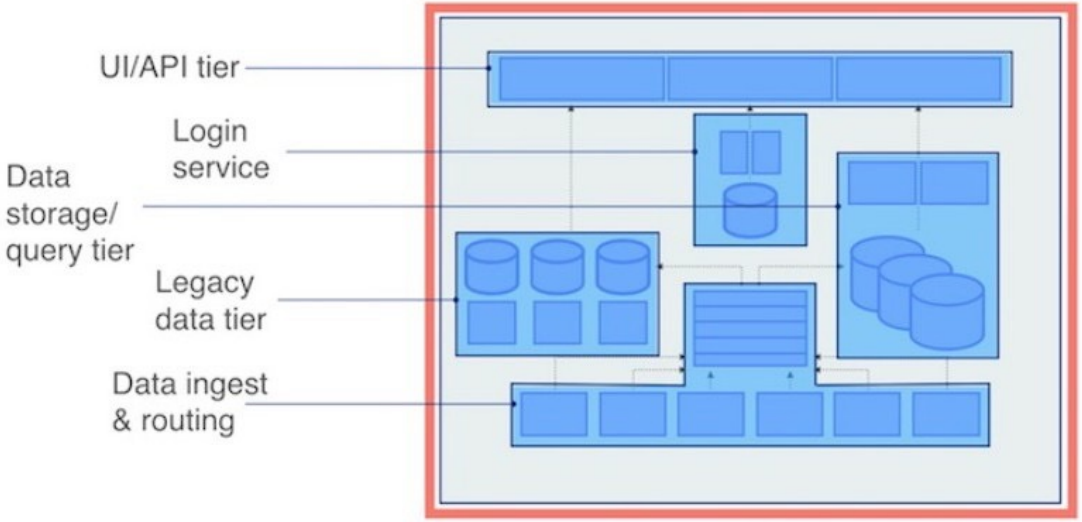
- We now have SLOs and an Error Budget, all of our request-based service with dashboards and alerting in place to inform us if we consume too much of our Error Budget.
- We then use Elasticsearch Heartbeat to monitor our internal only APIs and services. The SLOs for the internal services are also stored in Google Datastore and updated by the same Kubernetes cronjob to keep consistent dashboards.

Example – Kudos's SLO journey



- What's wrong with SLI on Webserver Logs?
  - You will be missing requests that do not get to the application, like failures on the load balancer or uncaught exceptions in the application
- How about Client-side SLI using client-side instrumentation such as analytics tools or frontend libraries?
  - You will be measuring the customer's internet connection and their phone/laptop/tablet reliability
- Example: Using synthetic client probes to check our ordering system response every 1 minute.
  - SLO can be 99.9% of remote probes sent every 1-minute return a good response over a 1-month period
  - Setup the remote probe to poll services every minute and alert if it is failing

System Boundaries come to the rescue



A System of System View

- Identify the system boundaries within your platform.
- Be logical to define a clear capability. If too broad, it would defeat the purpose of measurable SLI.
- Identify the customer-facing capabilities that exist at each system boundary.
- Articulate a plain-language definition of what it means for each capability to be available.
- Define SLIs for that definition and start measuring to get a baseline.
- Define an SLO for each capability and track how we perform against it.
- Iterate and refine our system and fine tune the SLOs over time.
- For example, if you set an SLI for query response times, do not look at averages because averages lie.
- Setting SLI @ 99.9th percentile may be edge-case scenario, hence check on the 95th or 99th percentile to get insights on the vast majority of queries.

Wakaru.

SLIs + SLOs Simple Recipe

Rinse & Repeat!

Identify System Boundaries

- Data Domain driven approach to identify business capabilities
- Use the capabilities to identify system boundaries

Define Capabilities for each system

- Address cross functional issues in a distributed system to keep single service responsibility

Define SLI for each capability

- Data Routing SLI (Time to deliver message to correct destination) is an example

Define SLO Targets (per SLI per capability)

- One SLO per capability goal
- Example:
  - Data Ingest SLO 99.9%
  - Data Routing SLI -> 99.5% of messages reached under 5 sec

Measure Baseline

- Assume that both your SLOs and SLIs will evolve over time.

Wakaru.

Large Scale Cultural Shift

Iterate!

SLO Buy-in

- Leadership Buy-in
- Senior Management Support
- Horizontal Role
- Standard Templates
- Training
- Working Examples

Become a SLO Expert

- Learn the context - what do Business users and Engineers care about?
- Learn and Practice
- Prepare your Sales Pitch

Create Standard Artifacts for Use

- Documentation
- Strategy
- Definition
- FAQ
- Implementation step by Step
- Use case example
- Training
- Hands-on workshop

SLO Pilot and Rollout

- First Service (Small or Most frequently used)
- Dependencies
- Instrumentation
- SLI visualizations

Communicate

- Internal Meetup/ Conferences
- Engineering Reviews
- Demo days
- Newsletter
- Distribution List
- Community of SLO experts
- Catalog of SLO case studies
- Scale out

Wakaru.

Bonus

Do SRE teams need Product Managers?

There's an increasing number of product owners and program managers in SRE and Platform teams because they have to:

- Build products for users
- **Prioritize which reliability investments have the highest impact on customers**
- Create the long-term reliability strategy for a company
- Make Build Vs. Buy decisions
- Liaise with several functional groups, including teams outside of engineering
- **Define reliability targets and report on performance from the perspective of customers' expectations**
- Manage relationships with new and existing vendors

An SREs plate is full already so the tasks listed above are arguably stealing time from reliability-ensuring activities.

Wakaru.

What do SRE Product Managers do?

Product Managers supporting SRE, and Platform teams are asked to bring traditional product management techniques, such as user research, roadmap prioritization, and stakeholder alignment into the reliability world. According to several job descriptions their responsibilities often include:

- Partnering with engineering and product leads to build **product roadmaps for SRE**
- Creating a long-term strategy for **observability and tooling** investments, including managing vendor relationships
- Implementing and maintaining **Service-Level Indicators (SLIs) and Service-Level Objectives (SLOs)**
- Creating **profiles of users** and ensuring SRE's products addresses their needs
- Championing **reliability ownership** across non-SRE teams and enabling them to account for & track reliability of the services they're responsible for
- Owning the vision and strategy for: incident management, disaster recovery, performance testing, chaos engineering, etc.

Note: Responsibilities will vary from one organization to another, as well as job titles — SRE Product Lead, Technical Program Manager, SRE Product Owner, etc.

SLOs are Product Managers best friend

SLO methodology allows Product Managers to:

- Agree with non-engineering functions on the **reliability goals needed to meet or exceed customer expectations**
- **Communicate about reliability** performance with SLIs/SLOs as a standardized language
- Prioritize roadmap according to **SLO historical performance**
- Design better alerting and incident management strategies with burn rate alerting
- Enable teams to **own reliability of their services** with out-of-the-box service SLIs
- Monitor data-driven KPIs/OKRs, allowing for weighted, justified and fast decision making

More product managers step into this area or, most likely, more engineers formally take on a technical product management role within reliability.

Wakaru.