{mthree}

SRE in Action

Site Reliability Engineering

Overview

- DevOps vs SRE
- SRE Activities
- Advantages of SRE
- Error Budget & Calculations
- 7 Principles in detail

{mthree}

DevOps vs. SRE

DevOps

- The intersection of two key disciplines
  - Software Development (Dev) and IT Operations (Ops).

SRE

- The application of software engineering to operational problems
  - Teach application developers how to build reliable services

Typical Activities

- Develop and manage scalable, secure and stable systems
- Conduct Incident analysis
- Analyse performance and create improvement plans
- Monitor system efficiency
- Manage risks
- Automate manual tasks within the SDLC
- Ease workload through automated tools, logs and testing environments
- Implement new features
- Select infrastructure tools
- Adapt environments to increasing or decreasing numbers of users

{mthree}

SLIs versus SLOs

SLIs (Service Level Indicators)

- Real metrics obtained from monitoring systems and other systems that can provide metrics
- These metrics are based on values that will impact end-user experience

SLOs (Service Level Objectives)

- Agreed goals by the project team (Devs, Ops, Management)
- SLIs determine the basis for the SLO goals
- A tangible set of values to demonstrate to the client our commitment

{mthree}

The Error Budget

- A tool used to balance service reliability with pace of innovation
- A control mechanism for diverting attention to stability as needed

Google

Changes are a major source of instability, representing roughly 70% of our outages, and development work for features competes with development work for stability

{mthree}

Error Budget Calculations

- Embracing risk and managing risk requires measurement

- Time based risk

$$availability = \frac{uptime}{(uptime + downtime)}$$

- Aggregate availability

$$availability = \frac{successful\ requests}{total\ requests}$$

{mthree}

An Error Budget

1 minus the SLO of the service

100% reliability is never the right target!

- A 99.9% SLO service has a 0.1% error budget
- Example:
  - If our service receives 1,000,000 requests in four weeks,
  - A 99.9% availability SLO gives us a budget of 1,000 errors over that period
- Managing service reliability is largely about managing risk
- An error budget aligns incentives and emphasizes joint ownership between SRE and product development
- Error budgets help decide rates of releases and effectively defuse talks about outages with stakeholders
- Allows multiple teams to reach the same conclusion about production risk without rancour

{mthree}

Activity: Calculate an Error Budget

- The app devs state that to keep a customer happy, they need the DBAs to ensure that the databases will
  - Serve 10,000,000 requests per working day (Mon - Fri)
  - Time to service requests 1ms
  - Over a 4-week period, we can cope with a maximum delay of 5 seconds
- What is our Error Budget as a percentage?
- What is our reliability as a percentage?

Activity: Calculate an Error Budget (Answer)

- The app devs state that to keep a customer happy, they need the DBAs to ensure that the databases will
  - Serve 10,000,000 requests per working day (Mon - Fri)
    - 200,000,000 requests in a 4-week period
  - Time to service requests 1ms
    - 200,000,000 ms in a 4-week period
  - Over a 4-week period, we can cope with a maximum delay of 5 seconds
    - 5*1000 = 5000
- (10,000,000 x 20) / (5*1000) = 40,000 ms in a 4-week period
- Error budget 0.02% = 99.98% reliability

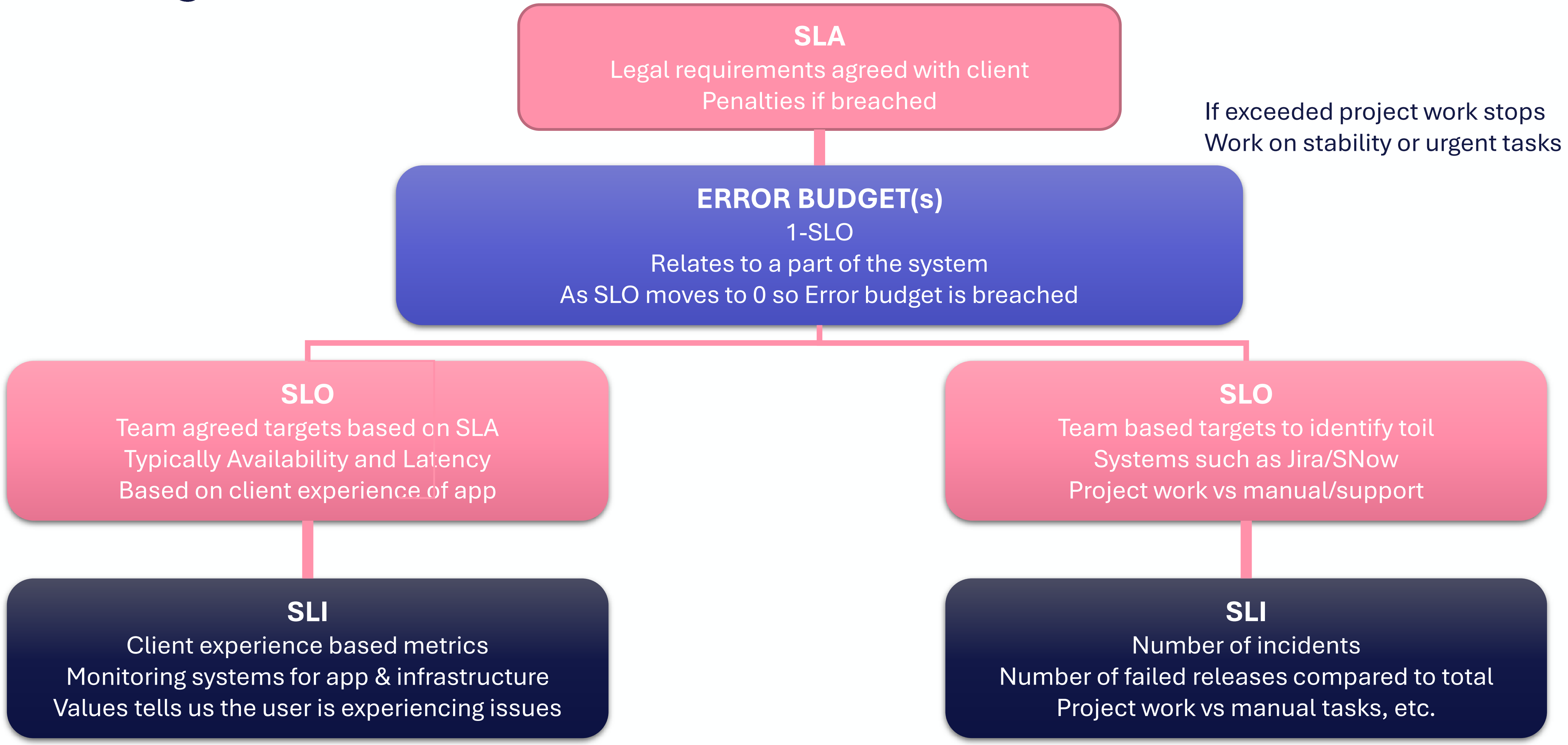
{mthree}

SRE 7 Principles in Detail

1. Embracing risk
   - >>> Manage, be open and learn
   - >>> Every release comes with risk
   - >>> Be prepared
2. Service level objectives
   - >>> Decide as a team = SRE, DevOps, Product Owners
3. Eliminating toil
   - >>> If it doesn't need human decisions, automate it
   - >>> If it costs more to automate than to perform the task, leave it
4. Monitoring distributed systems
   - >>> Reduce fatigue
   - >>> What broke, when and why
   - >>> Fix root causes to prevent repeats
5. Automation
   - >>> The move to self-healing systems
   - >>> Anything that doesn't need human decision or interaction
   - >>> When it doesn't exceed the error budget or the total time to fix manually
6. Release engineering
   - >>> Consistent deployment
   - >>> Solid understanding of SCM, Testing, CI, Post deployment monitoring
7. Simplicity
   - >>> "At the end of the day, our job is to keep agility and stability in balance"
   - >>> Smaller releases, easier measurements

{mthree}

SRE Diagram


SLA
Legal requirements agreed with client
Penalties if breached
If exceeded project work stops
Work on stability or urgent tasks
ERROR BUDGET(s)
1-SLO
Relates to a part of the system
As SLO moves to 0 so Error budget is breached
SLO
Team agreed targets based on SLA
Typically Availability and Latency
Based on client experience of app
SLO
Team based targets to identify toil
Systems such as Jira/SNow
Project work vs manual/support
SLI
Client experience based metrics
Monitoring systems for app & infrastructure
Values tells us the user is experiencing issues
SLI
Number of incidents
Number of failed releases compared to total
Project work vs manual tasks, etc.

{mthree}

Summary
Q & A

{mthree}

Resources

- Alvidrez, M. (2016). Chapter 3: Embracing Risk. In Site Reliability Engineering: How Google Runs Production Systems, Eds. Beyer, B., Jones, C., Petoff, J, & Murphy, Niall. From https://sre.google/sre-book/embracing-risk
- Beyer, B. et al. (n.d.) Google SRE Workbook. From https://sre.google/workbook/table-of-contents/
- Grams, C. (15 Oct 2019). How Much Time Do Developers Spend Actually Writing Code? From https://thenewstack.io/how-much-time-do-developers-spend-actually-writing-code/
- Vizard, M. (16 Feb 2021). Survey: Fixing Bugs Stealing Time from Development. From https://devops.com/survey-fixing-bugs-stealing-time-from-development/

{mthree}