{mthree}

Service Level Indicators

Site Reliability Engineering



Objectives

In this module, we will look at the concept of service level indicator, and how it is different than a simple metric.

Learning Objectives

- Define service level indicator (SLI)
- Explain the importance of service level indicators
 - Business
 - Personnel



- Key to evaluating the performance of the business
- Quantitative measure of service
 - >>> Based on Service Level Objectives
- Business focused
 - >>> Service results that matter to the user
 - >>> Rises and falls along with user happiness
 - >>> Shows obvious difference during service outage
 - >>> Oscillates in a narrow band during normal operations



Common SLI Attributes

- Aggregates or percentiles relevant and meaningful
 - >>> SLO target
 - >>> Ranges match expectations
- Indicate progress in continuous improvement
- Measures related to
 - >>> User experience
 - >>> Complex system performance
- Need strong correlation between SLI and user happiness
 - >>> Tune up your SLI metrics: CRE life lessons | Google Cloud Blog

Service Level Indicator Focus

- Application availability and uptime
 - >>> Percent
- Nequest rates / throughput
 - >>> Compared to baseline
- Error rates
- Satisfaction
 - >>> Latency
 - >>> Availability
 - >>> Response time



Selecting SLI

- Identify system boundary
 - >>> Identify user-facing expectations at boundary
 - ∼ What does availability look like
 - How to measure
 - ~ Capture baseline
 - >>> Based on baseline and expectations
 - ~ Track as time-based data
 - Confirm correlation of SLI with user expectations
- Fine tune SLI as business/user changes

SLI Specification Example

- User Transaction: Buying a Stock
- SLI Type: Availability
- SLI Specification: Proportion of buy page requests that are served successfully.
- SLI Implementations:
 - >>> Proportion of buy page requests that return a status code of 2xx.
 - → # buy page requests returning 2xx / total # buy page requests * 100
 - Should be greater than X
 - >>> Proportion of time that the OrderbookAPI container is accepting requests.
 - successful page requests processed by orderbookapi / successful page requests through web server * 100
 - Should be greater than Y

Actual Prometheus Metrics

Proportion of buy page requests that return a status code of 2xx

$$\left(\frac{count(http_server_requests_seconds_bucket\{container="orderbookapi",uri="/buy",status="200"\})}{count(http_server_requests_seconds_bucket\{container="orderbookapi",uri="/buy"\})} \times 100$$

Proportion of time that the OrderbookAPI container is accepting requests.

```
\left(\frac{count(http\_server\_requests\_seconds\_bucket\{container="orderbookapi",status="200"\})}{sum(nginx\_ingress\_controller\_requests\{exported\_namespace="orderbook-dev",status="200"\})}\right) \times 100
```

Example using Prometheus Metrics

- Success in using our sites ingress proxy
- Number of 200 successful requests in the last hour count_over_time(nginx_ingress_controller_requests{exported_namespace="orderbook-dev",status="200"}[1h])
- Number of total requests in the last hour $sum(count_over_time(nginx_ingress_controller_requests{exported_namespace="orderbook-dev"}[1h])$
- Total successful requests compared to total requests



Example - Calculation

Ideally, we express this as a percentage

```
 \begin{pmatrix} ("count\_over\_time(nginx\_ingress\_controller\_requests\{exported\_namespace = \\ "@"\"orderbook - dev\", status = \"200\"\}[1h])"" \\ \hline ("sum(count\_over\_time(nginx\_ingress\_controller\_requests\{exported\_namespace = \\ "@"\"orderbook - dev\"\}[1h]))"") \\ \end{pmatrix} \times 100
```

G

Frequently-Used KPIs

Nesponse Time (usually measured in minutes)

 $\frac{(When you \ respond \ to \ the \ alert) \ - (When \ the \ failure \ was \ detected)}{\# \ of \ Failures \ in \ a \ given \ time}$

Mean Time To Repair (MTTR)

 $\frac{Fixed\ DateTime-Time\ Detected}{\#\ of\ Failures\ in\ the\ given\ period}$

Average Cycle Time (measured for most important processes)

 $\frac{Finish\ Time\ - Start\ Time}{Total\ \#\ of\ Cycles}$

G

SLI for People

- SLI not restricted to technology monitoring
- Can be used to justify SRE efforts and watch for negative trends
- Positive metrics
 - >>> Mean Time to Restore decreasing
 - >>> Mean Time to Detect decreasing
 - >>> Number of backlog issues decreasing
 - >>> Velocity of updates increasing
- Negative metrics
 - >>> Burn out flagged by increasing Recovery Time
 - >>> Runaway toil flagged by increasing issue backlog



G

Activity: SLI calculation in Prometheus/Grafana

Create the SLI metric in Prometheus or Grafana for the following:

- SLI Type:
 - >>> Latency
- SLI Specification:
 - >>> Orderbook API **buy** requests served within **300ms**
 - >>> Time period for this check is per hour
- → Step-By-Step



Summary Q&A



{mthree}