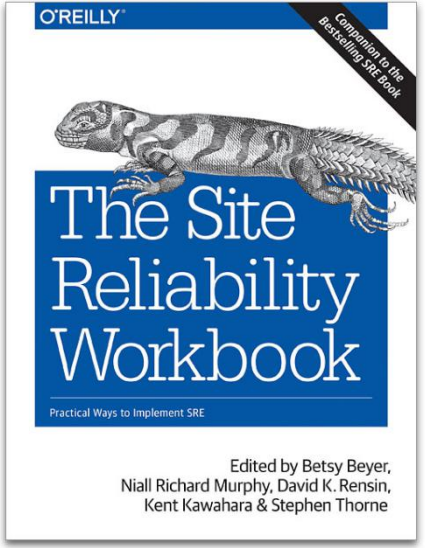
Google SRE 体系核心基础解读

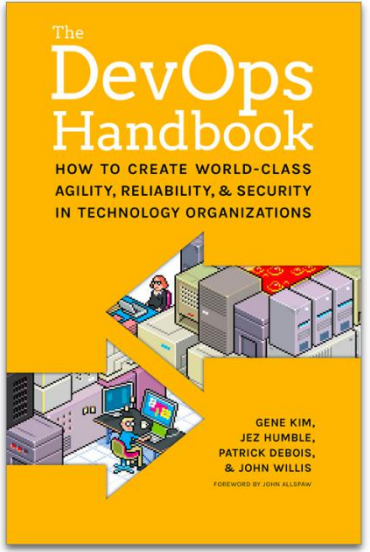
程序猿必备的运维理念

刘征
Elastic 技术布道师
中国 DevOps 社区组织者

Who am I?

```
{
    "name": "Liu Zheng - 刘征",
    "title": "evanglist",
    "company": "elastic",
    "community": "china devops",
    "translator": {
        "devops handbook": "done",
        "sre workbook": "doing"
    },
    "contact": {
        "wechat": "martinliu_cn",
        "blog": "martinliu.cn",
        "twitter": "@martinliu"
    }
}
```





SRE是什么？

- **SRE - Site Reliability Engineering *站点可靠性工程*** 起源于 **2003** 年。
- 一个可靠的运维大规模系统的框架（**tao lu**）。
- 就是让软件工程师来设计运维功能所发生的事情。
- 在运维层次上负责生产系统的运行。
- 构建和运行高可靠性系统的、普遍适用的最佳方式。

什么不是 SRE？

- **SRE** 原理听着不错，但是，它在我们这里水土不服，南橘北枳，它只能在特定文化里生长，只对超大规模才有意义。
- **SRE vs. DevOps**，它们之间是冲突的，谁更好？我们应该选择哪个？
- 可以把以前工程师和团队的名字改名为 **SRE** 工程师/团队/部门。

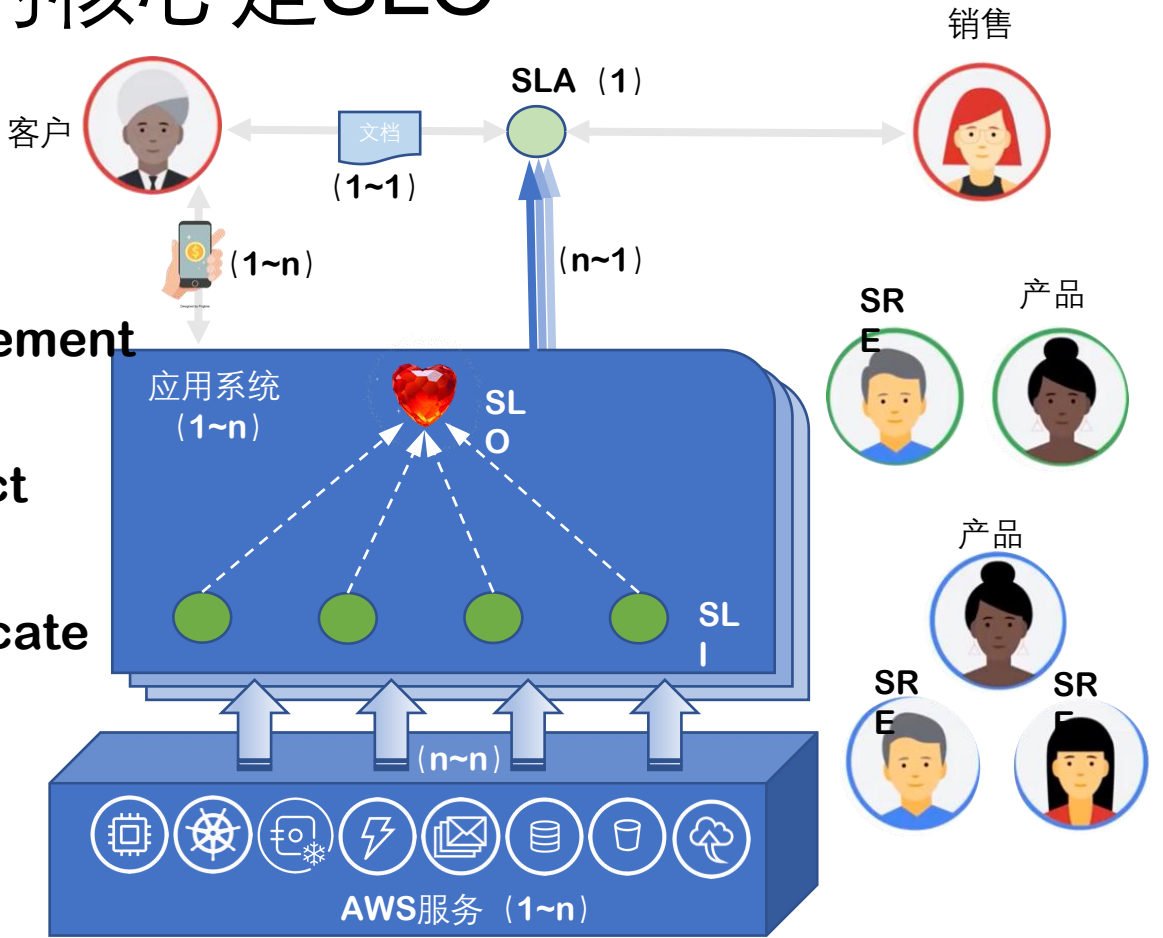
SRE 的五大根基

SLO 是什么？

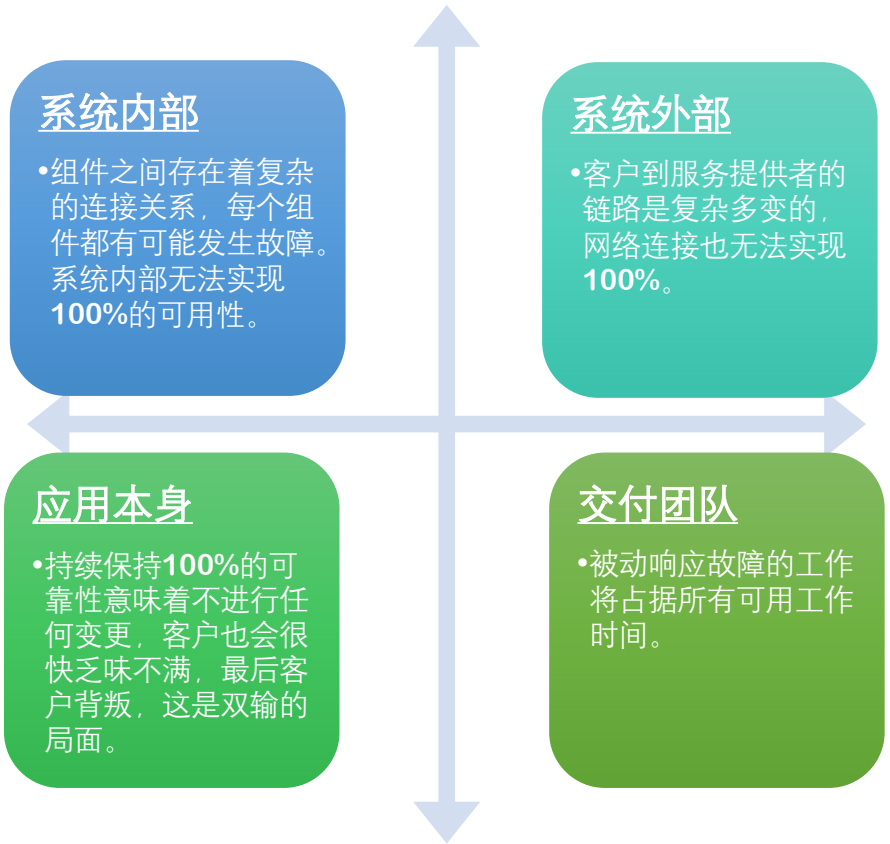
- 系统的服务质量目标定义了系统应当具有的正常表现。
- 专注于跟踪客户（人/机）的使用体验。
- 假如客户是满意的，那么 **SLO** 就达标了。

SRE实施策略的核心是SLO

- 服务质量协议： **SLA**
  - **Service Level Agreement**
- 服务质量目标： **SLO**
  - **Service Level Object**
- 服务质量指标： **SLI**
  - **Services Level Indicate**



100%的 SLO 正确么？


系统内部
•组件之间存在着复杂的连接关系，每个组件都有可能发生故障。系统内部无法实现100%的可用性。
系统外部
•客户到服务提供者的链路是复杂多变的，网络连接也无法实现100%。
应用本身
•持续保持100%的可靠性意味着不进行任何变更，客户也会很快乏味不满，最后客户背叛，这是双输的局面。
交付团队
•被动响应故障的工作将占据所有可用工作时间。

以 SLO 为支点的制衡机制

特性交付速度

系统可靠性

系统需要的 Uptime 是几个 9？

- 2 个 9 是：99%
- 3 个 9 是：99.9%
- 4 个 9 是：99.99%
- 5 个 9 是：99.999%
- 6 个 9 是：99.9999%
- 7 个 9 是：99.99999%
- SLA Uptime 计算器

  https://www.xarg.org/tools/sla-uptime-calculator/

SLO 在那一级能够用？

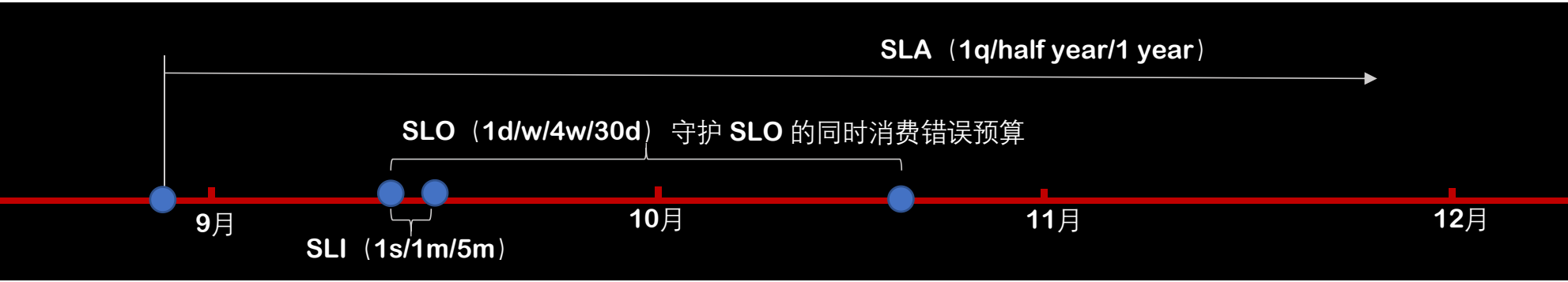
宕机时间 **= 1** 天/周/月/年的秒数 **x (100% -** 百分数）

SLO百分位	一天	一周	一月	一年
2：99%	14m12s	1h40m48s	7h18m17s	3d15h39m29s
3：99.9%	1m26	10m5s	43m50s	8h45m57s
4：99.99%	9s	1m	4m23	52m36s
5：99.999%	<1s	6s	26s	5m16s
6：99.9999%	<1s	<1s	3s	32s
7：99.99999%	<1s	<1s	<1s	3s
8：99.999999%	<1s	<1s	<1s	<1s

将 SLI 度量数值转换成 SLO 百分位

- 面向**SLI** 的度量，监控指标的单位本来就是不一致的：
  - 网卡流量 **MB/s**、 磁盘写入 **write/s**、**HTTP** 响应 **ms**、主页打开时间 **s** 等等
- 持续度量 **SLI** 的数值，并将监控到的 **SLI** 数值换算为，在不同百分位上的数值：
  - 在最近的 **10** 分钟里，**SLI-**主页打开时间，**P90**（**90%**）均值为 **259ms**
  - 在最近的 **10** 分钟里，**SLI-**主页打开时间，**P99**（**99%**）均值为 **589ms**
  - 在最近的 **10** 分钟里，**SLI-**磁盘写入，**P90**（**90%**）均值为 **45 write/s**
  - 在最近的 **10** 分钟里，**SLI-**磁盘写入，**P99**（**99%**）均值为 **12 write/s**
- 思考任一 **SLI** 度量数值在 **P90** 和 **P99** 的分布状态，客户满意么？

为系统设置一套合理的SLO



- uptime > 99.9%
- HTTP 200 > 99.99%
- HTTP 300ms 内响应 > 50%
- 日志 5min 内处理 > 99%

正向 SLO

- downtime < 43 分钟
- 非 200 返回码 < 0.01%
- HTTP响应超300ms < 50%
- 日志处理超5min < 1%

反向 错误预算

错误预算的逻辑

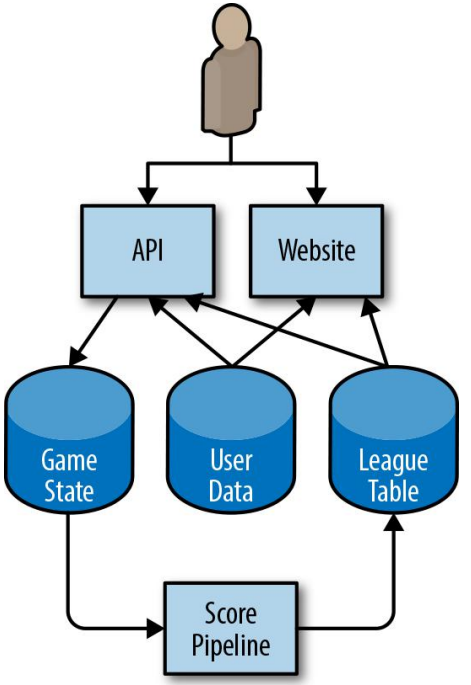
实时的监控　　理智的消费

SLO+错误预算=100%

手机游戏案例分析

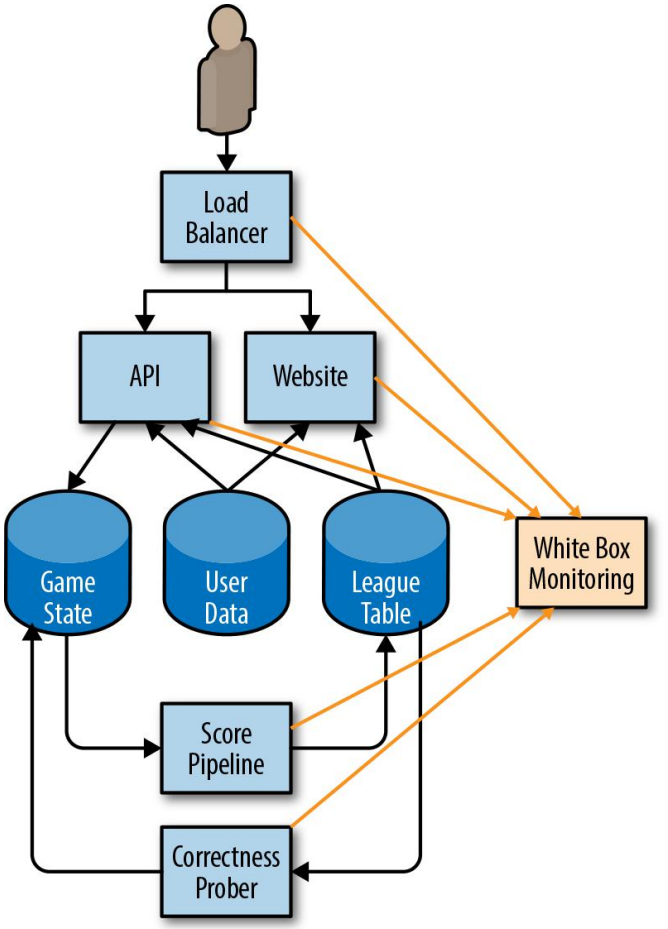
系统架构图

服务	AWS	类型	SLI 类型
API	EC2, ELB	请求驱动	可用性、延迟、质量
Website	EC2, ELB	请求驱动	可用性、延迟、质量
Game State	ElastiCache	存储	持久性
User Data	RDS	存储	持久性
League Table	DynamoDB	存储	持久性
Score Pipeline	EC2	流水线	时效性、正确率、覆盖率



实施面向 SLO 的系统监控

- 使用白盒监控模式
- 实施**API/HTTP**服务的两类 **SLI**：
  - 可用性：收集 **ELB** 的访问请求中 **5XX** 的总数
  - 延迟：收集 **ELB** 访问请求的响应时间
- 实施**Score Pipeline** 的三类 **SLI**：
  - 时效性：**App** 客户端检查积分榜时间戳
  - 覆盖率：收集流水线处理总数和成功总数
  - 正确性：使用外部正确性评判探针程序



采集负载均衡器的指标

- **CloudWatch** 可以所提供需要的数据
- 用 https://github.com/prometheus/cloudwatch_exporter 采集数据
- 用 **Prometheus** 的 **notation** 表示法 **SLI**，部分示例代码如下：

```
18  http_requests_total {host =“api”, status =“500”}
19
20  http_request_duration_seconds {host =“api”, le =“0.1”}
21  http_request_duration_seconds {host =“api”, le =“0.2”}
22  http_request_duration_seconds {host =“api”, le =“0.4”}
```

计算负载均衡器的指标

SLI	计算方法

API服务可用性	`sum(rate(http_requests_total{host="api", status!~"5.."}[7d]))`
	`/`
	`sum(rate(http_requests_total{host="api"}[7d]))`

API服务延迟	`histogram_quantile(0.9, rate(http_request_duration_seconds_bucket[7d]))` `histogram_quantile(0.99, rate(http_request_duration_seconds_bucket[7d]))`

从此进入 **P90, P95, P99, PXX Style**

使用 4 周的数据计算初始 SLO

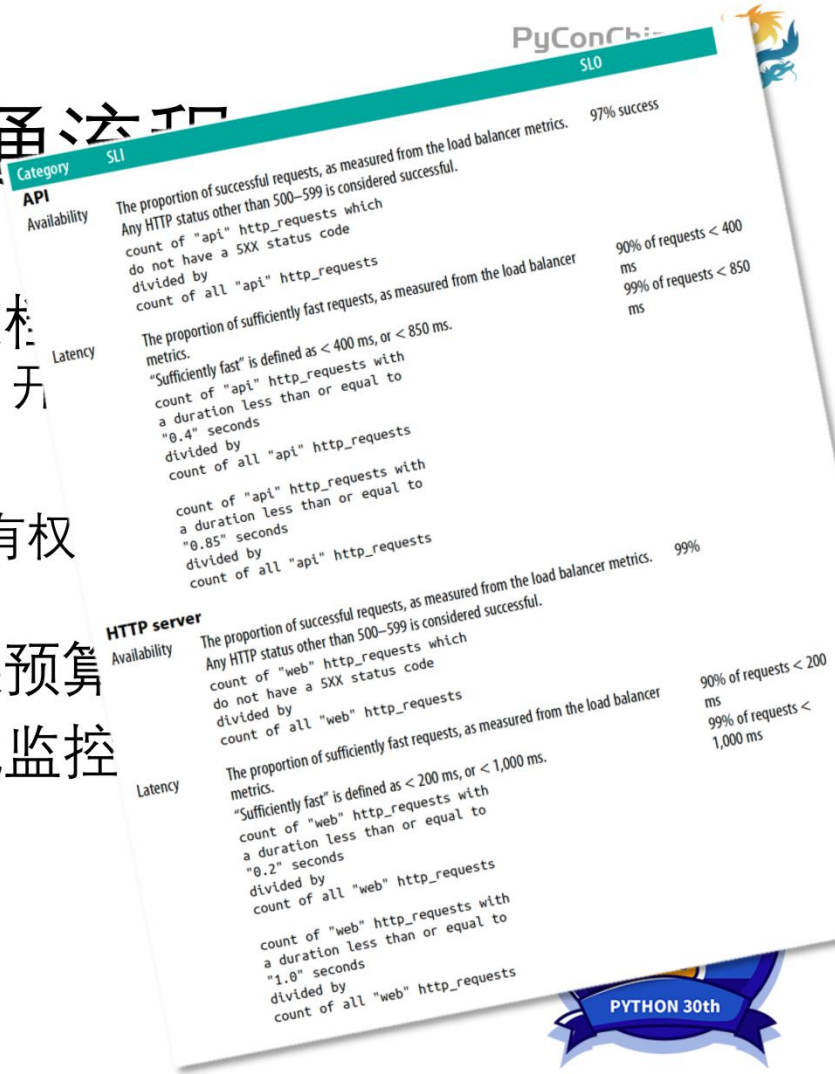
- **API** 服务 **4** 周的监控结果：
  - 请求总数：**3,663,253**
  - 成功请求总数：**3,557,865**
  - 成功请求百分比：**97.123%**
  - **90**百分位延迟：**432ms**
  - **99**百分位延迟：**891ms**

API 服务初始的**SLO** 建议值和错误预算

SLO 类型	SLO 目标	容忍的错误
可用性	97%请求成功	109,897 次 5XX
延迟	90%的请求 < 450ms	366,325 次长于 450ms
延迟	99%的请求 < 900ms	36,632 次长于 900ms

建立 SLO 相关文档和沟通流程

- 为手游应用系统建立正式的《**SLO** 文档
  - 获得所有利息干系者的认可：产品经理、开
- 建立《出错预算策略》文档
  - 面向后果的，得到管理层的授权，**SRE** 有权
    统的运维工作退回给开发团队
- 建立 **SLO** 的监控仪表板、报表和错误预算
- 持续优化 **SLO** 目标的设置，持续优化监控



Category	SLI	SLO
API		
Availability	The proportion of successful requests, as measured from the load balancer metrics. Any HTTP status other than 500–599 is considered successful. count of "api" http_requests which do not have a 5XX status code divided by count of all "api" http_requests	97% success
Latency	The proportion of sufficiently fast requests, as measured from the load balancer metrics. "Sufficiently fast" is defined as < 400 ms, or < 850 ms. count of "api" http_requests with a duration less than or equal to "0.4" seconds divided by count of all "api" http_requests count of "api" http_requests with a duration less than or equal to "0.85" seconds divided by count of all "api" http_requests	90% of requests < 400 ms 99% of requests < 850 ms
HTTP server		
Availability	The proportion of successful requests, as measured from the load balancer metrics. Any HTTP status other than 500–599 is considered successful. count of "web" http_requests which do not have a 5XX status code divided by count of all "web" http_requests	99%
Latency	The proportion of sufficiently fast requests, as measured from the load balancer metrics. "Sufficiently fast" is defined as < 200 ms, or < 1,000 ms. count of "web" http_requests with a duration less than or equal to "0.2" seconds divided by count of all "web" http_requests count of "web" http_requests with a duration less than or equal to "1.0" seconds divided by count of all "web" http_requests	90% of requests < 200 ms 99% of requests < 1,000 ms

基于 SLO 和错误预算的决策

SLO	琐事	客户满意度	行动
达标	低	高	a) 放心且加速的执行发布和部署工作； b) **SRE**回撤，把工程时间转移到其它更需要可靠性的服务上。
达标	低	低	收紧**SLO**。
达标	高	高	如果有假阳性警报，就降低敏感度；否则就放宽**SLO**（或消除琐事）并且修补产品的 **bug**，或者改善系统的自动化故障迁移机制。
达标	高	低	收紧**SLO**。
未达	低	高	放宽**SLO**。
未达	低	低	增加警报敏感度。
未达	高	高	放宽**SLO**。
未达	高	低	消除琐事，或者改善系统的自动化故障迁移机制。

SRE 的五大根基

SRE 工作原则 #1

- **SRE** 需要设计和实施面向后果的 **SLO**。
- 任何组织，甚至连一个 **SRE**都不用雇，就能够设计错误预算策略。
- 这意味着识别和利用任何能防范客户遭遇痛苦使用体验的抓手。
- 你今天就能开始实施：度量、负责、行动

SRE 工作原则 #2

- **SRE** 需要有时间进行优化改善。
- 一旦 **SRE** 人员就绪：就要确保他们知道，他们的工作不是继续遭运维工作的罪，而是每天都优化运维工作。
- “更聪明的工作”可能意味着做不同的事情：这要看你的 **SRE** 能找到什么最有用的工作事项。

SRE 工作原则 #3

- **SRE** 需要有能力调控他们的工作负载。
- **SRE** 团队需要能够安排优先级并开展工作。
- 每一个新系统的维护都需要人力成本。
- 必须能够压制不可靠的工作实践，推回不靠谱的系统。

THANK YOU

刘征
email: zheng.liu@elastic.co
blog: https://martinliu.cn
wechat: maritnliu_cn
twitter: @martinliu

- 关注微信公众号
- 回复 **SRE**
- 下载本 **PPT**