# Operation

时长一个软件系统的绝大部分使用寿命都花在了使用上，而不是花在设计或实现上。一个系统的总成本的40-90%是在出生后发生的。

SRE(site reliability engineering)

WHY SRE?1.操作（Ops）需要与开发（Dev）明显不同的技能集：1.组装现有软件组件并部署它们一起工作以生成服务2.运行服务并在事件和更新发生时响应它们。3.负责可用性、延迟、性能、效率、变更管理、监控、应急响应和容量规划。2.系统管理方法和伴随的开发/操作拆分有许多缺点和缺陷：1.直接成本：规模/成本随服务的规模呈线性增长2.间接成本（更昂贵）：不同的背景、技能集、激励、词汇、对风险的假设、稳定性假设、沟通、目标、信任。3.大多数中断（中断）是由某种改变(新配置，一个新功能发布，一种新的流量)引起的。开发和行动因此从根本上在紧张（开发：我们想推出任何东西，任何时候，没有阻碍；行动：我们不想改变任何系统一旦工作网站可靠性工程（高级））

WHAT IS SRE?网站可靠性工程团队有能力实现运维自动化，目标“使服务自行运行和自我修复”

TENETS OF SRE： 1.确保持续关注工程：不超过50%的操作的时间；不超过两个事件和不少于一个事件每8-12小时轮班；事后分析应该写所有重大事件（紧急事件）2.追求最大变化速度不打扰服务的进行：业务/产品建立系统的非100%可用性目标，错误预算（错误预算）=1-可用性目标；中断（中断）不再是一件“坏事”——它是创新过程预期中的一部分3.监控：紧急情况：警报（立即行动）>票（工单）（行动）>日志（供参考）**Alerts (immediate actions) > Tickets (工单) (actions) > Logging (FYI)4.**应急响应：可靠性是平均故障时间（MTTF）和平均修复时间（MTTR）的函数；人可能增加延迟-尽可能地自动化；随叫随到（轮值）“剧本”（MTTR改进了3倍）和“灾难角色扮演”5.变更管理：70%的中断（中断）是由于实时系统的变化；从循环中移除人员；实现渐进式推出；快速、准确地检测问题；在问题出现时安全地回滚更改6.需求预测和能力规划：确保有足够的容量和冗余来满足所需的可用性的预计未来需求；三个强制性步骤：准确的有机需求预测，超出获取容量所需的提前时间；将无机需求源准确纳入需求预测；系统的定期负载测试将原始容量（服务器、磁盘等）与服务容量相互关联7.供应：结合变更管理和容量规划8.效率和性能：高效利用资源节省（大量）成本；应密切监控和保持性能（响应速度）

Principle

embracing risk：1.平衡不可用的风险与快速创新和高效服务操作的目标：优化用户的整体幸福感——功能、服务、性能、可靠性；用户通常不会注意到一个服务的高可靠性和极端可靠性之间的区别2.通过管理风险来管理服务的可靠性：减少系统故障的机会；确保可靠性的成本随着可靠性的增加而非线性地增加（冗余机器/计算资源/机会成本（做新产品特性））3.测量服务风险：低可靠性成本：用户不满、伤害，或失去信任；直接或间接收入损失；品牌或声誉影响；和不可取的新闻报道；基于时间的可用性：可用性=正常运行时间/（正常运行时间+停机时间）MTBF=uptime / num\_of\_failures ，time between failure；MTTR=downtime / num\_of\_failures, time to repair； **Availability= MTBF / (MTBF+MTTR)；**总可用性（合计可用性，谷歌首选）：可用性=成功请求/总请求，例如。99.99%的可用性意味着每250万个请求有250个错误4.管理服务的可靠性主要是为了管理风险，而管理风险可能成本很高。100%可能永远都不是正确的可靠性目标：它不仅不可能实现，而且通常比服务的用户想要或注意到的更有可靠性。将服务的概况与业务愿意承担的风险相匹配。错误预算（错误预算）协调激励措施，并强调SRE和产品开发之间的共同所有权。错误预算可以更容易地决定发布速率，并有效地化解与利益相关者的关于中断（中断）的讨论，并允许多个团队在不产生仇恨的情况下就生产风险得出相同的结论。

service level objectives：1.服务水平术语：服务水平指标（SLI）：精心定义的定量测量某些方面的服务水平的数量指标，例如，请求延迟、错误率、系统吞吐量、可用性，通常聚集在一个时间窗口：速率、平均或百分位；服务水平目标（SLO）：由SLI衡量的服务水平的目标值或范围，应该是优先工作的主要驱动因素，例如，每个请求的平均延迟应该低于100毫秒，例如谷歌计算引擎99.95%可用；服务级别协议（SLA）与用户的显式或隐式合同，包括满足（或丢失）它们所包含的SLO的后果2.您和您的用户关心什么？通常一些代表性指标：不太多，不少；面向用户的服务系统（如前端）：可用性、延迟和吞吐量；存储系统：延迟、可用性和耐久性；大数据系统：吞吐量和端到端延迟3.选择目标:1.不根据当前性能选择目标2.保持简单：没有复杂的聚合3.有尽可能少的SLOs4.达到完美需要时间，可以等待4.SLOs设置预期:1.保持安全边际范围2.不会过度实现

eliminating toil：琐事是一种与运行生产服务相关的工作，该生产服务往往是手动的、重复的、可自动化的、战术的（战术性的），缺乏持久的价值，并且随着服务的增长而呈线性增多,SRE管理组织有一个公开宣传的目标，即将操作工作（即辛劳）保持在每个SRE时间的50%以下。每个SRE至少50%的时间应该花在工程项目工作上，这些工作将减少未来的辛劳或增加服务功能，谷歌SRE的平均水平是33%,工程工作是新颖的且本质上需要人类的判断

典型的SRE活动分类：1.软件工程：编写或修改代码，以及任何相关的设计和文档化工作2.系统工程：配置生产系统，修改配置等3.Toil：直接与运行服务相关的重复的手动的工作4.开销：不直接和运行服务相关的管理工作

monitoring distributed systems：1.概念：监控：例如SLI2.白盒监控：内部指标，如日志，JVM日志，HTTP日志3.黑盒监控：测试外部可见行为4.仪表板：监视服务的核心指标5.警报：通知旨在被人类读取6.根本原因：软件或人类系统的缺陷，如果修复，灌输让人们认为这些事件不会再次发生以同样的方式的信心7.节点/机器：物理服务器、虚拟机或容器

监控的目的：1.分析长期趋势2.随着时间的推移或实验组别的不同进行比较3.警报：损坏的东西，有人现在需要修复（有效警报系统有良好的信号和非常低的噪音；寻呼一个人是一种相当昂贵的使用员工的时间的方式；太频繁的寻呼导致“寻呼疲劳（紧急警报疲劳）”，这可以延长中断）4.构建仪表板（四个黄金信号，避免要求有人盯着屏幕观察问题）进行特别的回顾性（回溯）分析（即调试）

黑盒和白盒：黑盒：面向症状，表示活动的-非预测的问题“系统现在不能正常工作。”；白盒：依赖于检查系统内部的能力，允许检测迫在眉睫的（即将来临的）问题，被重试掩盖的故障，等等。

监控系统的四个黄金信号（黑盒）：1.延迟：服务所需的时间请求2.流量：衡量多少需求被放置在你的系统，在高级系统特定度量3.错误：失败的请求率4.饱和：你的系统有多“满”。一个对系统分数的衡量标准，强调最受限制的资源，在一些小窗口（例如，一分钟）上测量你的第99百分位响应时间可以给出一个非常早期的饱和信号。

担心你的尾巴：百分位数而不是平均，如果你运行一个web服务平均延迟100 ms 1000个请求每秒，1%的请求很可能需要5秒

选择一个合适的测量精度：有时需要精细的监控，但不总是；高分辨率不总是需要极低的延迟：聚合本地

尽可能简单：1.最能捕捉真实事件（紧急事件）的规则应该尽可能简单、可预测和可靠。2.很少被使用的数据收集、聚合和警报配置应该被删除。3.收集但未在任何仪表盘中暴露或被任何警报使用的信号是被删除的候选信号。

在创建监视和警报的规则时，询问以下问题可以帮助您避免误报和寻呼机耗尽：1.此规则是否检测到紧急的、主动的或即时用户可见的情况？2.我能忽略这个警报，知道它是良性的吗？我什么时候以及为什么能够忽略这个警报，我如何避免这种情况？3.此警报是否明确表明用户受到了负面影响？是否存在可检测到的用户没有受到负面影响的情况，如流量耗尽或测试部署，这种应该被过滤掉的情况？4.我可以采取行动来响应这个警报吗？这是一种紧急的行动，还是可以等到明天早上再说？该操作可以安全地自动化吗？这种行动是一个长期的解决方案，还是只是一个短期的解决方案？5.其他人是否因为这个问题而被分页，因此使至少有一个页面不必要？

寻呼哲学：1.每次寻呼机响起，我应该能有一种紧迫感。在我疲劳之前，我只能有几次紧迫感。2.每一页都应该是可操作的。3.每个页面的响应都应该需要人类的智能。如果一个页面仅仅值得一个机器人的响应，那么它就不应该是一个页面。4.页面应该是关于一个新奇的问题或一个以前从未见过的事件。5.应该有人找到并消除问题的根本原因；如果不可能解决这个问题，警报响应就应该被完全自动化。

the evolution of automation at google

Practice

Practical Alerting from Time-Series Data：1.最基本的要求：作为一个服务发挥功能2.高级要求：允许自实现（自我实现）；主动控制服务的方向，而不是被动灭火3.成功地运维一项服务需要广泛的活动

：开发监控系统；规划能力；响应事件（紧急事件）；确保中断的根本原因（中断）被解决4.监控是运行稳定服务的基础。它使服务所有者做出合理的决定；影响服务的变化；应用科学方法响应事件；测量服务与业务目标5.监控一个非常大的系统是具有挑战性的几个原因有几个：被分析的组件的绝对数量；需要为负责该系统的工程师保持一个合理的较低的维护负担6.我们需要监控系统，允许我们警报高级服务目标，但有需要的话也保留检查单个组件的粒度。7.规则计算：规则是一种简单的代数表达式，它从其他时间序列数据中计算其他时间序列数据。E.g.：错误率=（#错误/#请求）；在并行线程池中运行；聚合需要从作业中的任务中获取一组时间序列数据的和，以便将作业作为一个整体来处理，例如，每秒查询在数据中心=sum（每秒查询在所有集群）；结果时间序列也存储在时间序列领域待查询8.警报：当警报规则评估，结果是真的，在这种情况下，触发警报，或假的；规则允许最小持续时间（2个周期），警报规则必须在发送警报之前（处理故障）已经正确；警报发送到中央警报管理器，负责将警报通知路由到正确的收件人，警报抑制和补偿

Being On-Call (轮值)：1.轮值：在工作时间和非工作时间均可打访问，以保持其服务的可靠和可用2.轮值工程师（生产系统的监护人）：一旦收到并确认页面，工程师将对问题进行分类并努力解决问题，可能涉及其他团队成员，并根据需要进行升级扩展，99.99%可用性->季度错误预算13分钟->在5分钟内启动操作，30分钟对时间不敏感的事件，非分页事件在工作时间以较低的优先级完成3.平衡轮值：数量平衡：我们坚信“SRE”中的“E（工程）”。至少50%的SRE时间投入工程，其余的，不超过25%的时间可以花在轮值；质量平衡：对于每个轮值的班次，工程师应该有足够的时间来处理任何事件（紧急事件）和后续活动，如撰写事后报告，每个事件需要6小时，这是与同一根本原因相关的一系列事件和警报序列，将作为同一事后分析的一部分进行讨论。4.相关的理性，专注，和有意识的（深思熟虑的）认知功能中断（中断）：避免不反思和未考虑的（但立即）行动，导致潜在的启发探索式滥用，遵循直觉会导致工程师浪费时间追随从一开始就不正确的一系列推理5.事件（紧急事件）管理中最理想的方法论是，在有足够的数据可以做出合理的决定时，同时严格检查你的假设时，以期望的速度采取步骤的完美平衡6.依赖于轮值资源：明确升级（升级）路径；定义良好的事件管理过程和相应的工具；一个无可指责的事后文化，但SRE必须在重大事件后进行事后分析，并详细说明所发生事件的完整时间表7.避免不适当的操作负载（>50%）：错误配置监控是运维过载的常见原因；控制轮值的工程师为单个事件（紧急事件）接收到的警报的数量：抑制和压缩；与应用程序开发人员一起制定共同的目标来改进系统是合适的；将寻呼消息返回给开发人员，直到系统满足SRE标准为止

Effective Troubleshooting：1.problem report->triage->examine->diagnose->test/treat->cure2.故障排除是任何操作分布式计算系统的人的关键技能，特别是SREs3.有效故障排除所需的两个因素:理解如何一般故障排除（即，没有任何特定的系统知识）;系统的扎实知识4.故障排除过程的一般模型—假设-演绎方法,给定一组关于系统的观察结果和理解系统行为的理论基础，我们迭代地假设失败的潜在原因，并试图检验这些假设,比较理论与证据；处理修复系统，观察，直到确定根本原因。5.查看陷阱:查看与之不相关的症状或误解系统指标意义;误解如何改变系统、输入或环境，以安全有效地测试假设;提出了关于什么是错误的极不可能的理论，或者抓住了过去问题的原因，认为既然它发生过一次，它一定会再次发生,在所有条件相同的情况下，更喜欢更简单的解释。（Ocamm剃须刀）;寻找实际上是巧合或与共同原因相关的虚假相关性,相关性不是因果关系6.问题报告:一个有效的报告应该告诉您预期的行为、实际的行为，以及，如果可能的话，如何重现该行为;请记录调查和补救活动，以供日后参考7.分类:评估问题的严重程度;停止出血应该是你的首要任务：让系统在这种情况下尽可能地正常工作,将流量从中断的集群转移到其他仍在工作的集群;批量减少流量以防止级联故障;禁用子系统以减轻负载8.检查:检查系统中的每个组件;监控系统：所有指标,系统和应用程序日志：开源日志存储和搜索工具：弹性搜索;调用跟踪;更改日志;工作流日志9.简化和减少:查看组件之间的连接—或者同等地，查看组件之间的数据流动—以确定给定的组件是否正常工作;系统地从堆栈的一端开始，并向另一端工作，依次检查每个组件10.问“什么”，“在哪里”和“为什么”:找出故障系统在做什么;然后问它为什么要这样;它的资源在哪里被使用或输出去哪里11.上次触及它的是:配置更改或所服务的负载类型的更改;将系统性能和行为的变化与系统和环境中的其他事件关联起来，也有助于构建监控仪表板;注释一个图表，显示新版本部署的开始和结束时间12.结论:使用白盒度量和结构化日志从头开始构建每个组件的可观察性;记录工作流和更改;设计组件之间具有良好理解和可观察的接口的系统;确保信息在整个系统中以一致的方式可用，从而加快进行诊断和恢复的时间;采用系统的方法来排除故障——而不是依赖运气或经验——可以帮助限制您的服务恢复时间，为您的用户带来更好的体验。

Emergency Response:1.当系统破坏时该怎么办？不要惊慌！你并不孤单，天空也没有落下;如果你感到不知所措，请吸引更多的人2.三例：测试（受控故障）诱发紧急//变化诱发紧急:谷歌病例▪//过程诱发紧急:阿里云病例3.所有的问题都有解决问题的办法:时间和经验表明，系统不仅会崩溃，而且会以人们以前无法想象的方式崩溃;参与更多的队友，寻求帮助，做任何你必须做的事情，但要很快;非常重要的是，一旦紧急情况得到缓解，不要忘记留出时间来清理，写下事件4.从过去学习，不要重复它:保存停机历史（中断），写下预防策略和策略;问一个大的，甚至不可能的问题：如果……？;鼓励进行主动式测试

Managing Incidents (紧急事件):1.有效的事件管理是限制事件造成的中断和尽快恢复正常业务运营的关键2.精心设计的事件管理过程的要素:递归的责任分离(事件指挥官,运维工作：在事件期间修改系统的唯一小组,通信：事件响应工作组的公众面孔,计划：支持运维)3.一个公认的指挥所：作战室或IRC4.实时事件状态文件5.事件指挥官清楚的生动的实时移交。

Postmortem Culture: Learning from Failure:1.除非我们从这些事件中学习一些正式的过程，否则它们可能会无限出现。如果不加以控制，事件可能会增加复杂性，甚至是级联（连续传递），淹没一个系统及其运维员，并最终影响到我们的用户2.事后分析是对事件、其影响、为减轻或解决事件所采取的行动、根本原因以及防止事件再次发生的后续行动的书面记录3.死后目标:事件被记录下来;所有的根本原因都被很好地理解;采取有效的预防措施，以减少复发的可能性和/或影响

Conclusion:1.一个系统的总成本的40-90%是在出生后产生的2.软件工程：专注于设计和构建软件系统3.站点可靠性工程：重点关注软件对象的整个生命周期，从一开始，到部署和运维，改进，并最终和平退役4.SRE：首先，SREs是工程师//SREs主要关注系统的可靠性//SREs专注于建立在我们的分布式计算系统之上的运维服务5.SRE取得巨大成功的关键是它运作的原则（在第2部分中）的本质，例如，包括风险、服务水平目标、消除辛劳、监控等，SRE在实践中已经成长起来。