O'REILLY®

Compliments of Google Cloud

企业 SRE 路线图
Enterprise Roadmap to SRE

How to Build and Sustain an SRE Function

如何构建和保持 SRE 职能

James Brookbank
& Steve McGhee

REPORT

译者：刘征

本文档非 Google 官方翻译，内容解释权归 Google 所有。

Enterprise Roadmap to SRE
by James Brookbank and Steve McGhee

Printed in the United States of America.
Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://oreilly.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.
Acquisitions Editor: John Devins
Development Editor: Virginia Wilson
Production Editor: Christopher Faucher
Copyeditor: Tom Sullivan
Proofreader: Stephanie English
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Kate Dullea
January 2022: First Edition
Revision History for the First Edition
2022-01-20: First Release

目录

前言

Google 的前两本 O'Reilly 书籍——《SRE Google 运维解密》和《Google SRE工作手册》——展示了为什么要承诺服务的整个生命周期，才可以使组织成功地构建、部署、监控和维护软件系统。前者由 Betsy Beyer、Chris Jones、Niall Richard Murphy 和 Jennifer Petoff 编辑，后者由 Betsy Beyer、Niall Richard Murphy、David K. Rensin、Kent Kawahara 和 Stephen Thorne 编辑。

本报告旨在在这些书籍的基础上，深入探讨在大型复杂组织（即企业）中采用 SRE 的挑战。尽管 SRE 在过去几年中非常流行，但参考我们从许多企业所获得的反馈表明，SRE 的热情与实际采用之间存在差距。

我们认为这是一个需要弥合的重要差距，因为可靠性正日益成为企业的重要竞争优势。上公有云和 COVID-19 大流行引发的技术变革速度和规模，通常需要不同的技术来应对增加的复杂性。

如果您参与生产系统的可靠性，或者依赖其可靠性，并且需要了解更多关于 SRE 采用的信息，这些主题将引起您的兴趣。这包括执行和领导角色，也包括个体贡献者（如云架构师、站点可靠性工程师 [SRE]、平台开发人员等）。无论您的角色是什么，如果您设计、实施或维护技术系统，这里都有适合您的内容。

第一章：初探企业 SRE

将 SRE 导入当前的各种传统企业，是一项看似可能艰巨的工作，因此 Google SRE 整理了一些建议，希望能帮助到更多企业。通过评估企业现有的环境、设定合理的预期，并确保企业朝着正确的方向迈出正确的步伐，企业可以从评估 SRE的原则和实践，从评估SRE在组织中的运作方式开始。

演进优于革命

企业的一个显著特点是：会始终拥有以前的 IT/管理信息系统（MIS）方法和原则的历史，Google SRE将详细讨论一些常见的方法。无论当前状态如何，Google SRE发现在采用 SRE 时，选择通过演进和补充现有框架的方式，而不是直接与其对抗会取得最大成功。此外，SRE 与其他任何技术的采用过程类似，都会受到历史遗留的影响（参见维基百科关于路径依赖的页面）。简而言之，这意味着：在像企业这样的复杂系统中，在不同地方应用相同的改变，也将产生出不一致而非收敛的结果。Google SRE将从一些成功的采用了不同的流行框架的示例开始讨论。

SRE 实践可与 ITSM 框架共存

信息技术基础架构库（ITIL）是一组详细的 IT 活动实践，如 IT 服务管理（ITSM）。并非每个企业都使用 ITIL，但如果企业的相关组织在一定程度上采用过 ITIL，则应准备好 SRE 和 ITIL 实践之间会存在着重叠。此外，由于 ITIL 是一个框架，企业的定制化实施可能与库中的内容有很大差异。

关键点： ITIL 有五本核心书籍，用数千页的篇幅涵盖了：关于如何构建和运行 IT 服务的内容，其中的很多主题与可靠性无关，也有很多内容故意未被 SRE 涵盖。ITIL 是一个框架，而 SRE 是一组实践，因此它们肯定是兼容的，但是在翻译术语（例如，“warranty”，“utility” 等）时可能会遇到挑战。此外，SRE 在变更管理和服务的拥有权等领域有着明确的观点，因此即使结果是一致的，也要做好调整的准备。

对于调和现存的一些常见 SRE 的反模式，可能也会比较有挑战。变更咨询委员会（CAB）是变更控制的常见模式。SRE 所秉持的持续交付的方式，意味着要让这个机构简化和战略化：企业可以在 Google 的 DevOps 研究和评估（DORA）的文章中，了解到更多有关简化变更审批的内容。类似地，对于网络运维中心（NOC，或者中国的 ECC）而言，则应该将其从事件驱动模型转变为更具前瞻性的方式，重点是对其进行自动化和赋能。在这两种情况下，重点是演进当前的运作模式，而非立即替换它们。

DEVOPS/敏捷/精益

DevOps 有多种定义。为了简单起见，Google SRE 假定它包括其他方法的相关部分，如敏捷（SAFe、DAD 和 LeSS）和精益（Six Sigma、看板）。Google 的 DORA 研究表明，SRE 和 DevOps 是互补的，因此如果企业的组织在一定程度上采用了 DevOps，则通常会有所裨益。与 ITIL 一样，我们要预见到 SRE 和 DevOps 实践存在着一些重叠，并且企业的定制化实施可能与《DevOps Handbook - DevOps 实践指南》存在着广泛的差异。Google SRE 将在后面更详细地介绍特定的 SRE 实践，但 SRE 与 DevOps 相关最大的许多能力（例如版本控制、同行评审等）也通常被视为采用 SRE 的先决条件。无论企业选择通过 DevOps 还是 SRE 倡议来构建这些能力，这些都由企业来决定，但为了确保采用 SRE 的成功，那些重要的 DevOps 能力仍然需要提前准备继续。

> **关键点：**当企业在调和 DevOps 和 SRE 的差异时，倡议务实的原则；想要完成大规模演进变革的成功，还是要通过迭代和循序渐进的方式来实现。重要的是：需要把特定的工作活动拆解出来，并专注于对人员的赋能，而不是花费不必要的时间和精力来获取一个完美的"空架子"。

尽管 DevOps 和 SRE 是互补的，但它们在一些领域还可能会令人难以调和。例如，企业可能已经决定将开发和运维报告层次结构替换为跨职能的 DevOps 团队。在这种情况下，重新引入像 SRE 这样的专门职能则需要进行认真的考虑。

千里之行，始于足下

无论您的企业正在使用着什么方法和框架，了解并诚实地对待企业今天的现状都很重要。正如《Google SRE 运维解密》一书所言，"希望不是一种策略！" 如果企业认为：当前的企业环境中即没有任何缺失，也不存在任何改进的机会，那么企业应该问自己：为什么要采用 SRE。同样，企业现有的一些技术或员工的想法，在刚开始的时候，看起来也可能与企业的 SRE 愿景并不一致。在做出任何改变以前，花时间来了解这些也很重要。

明确企业的期望和愿景

接下来，企业了解自己期望的结果很重要。SRE 会包含许多技术和文化的组成部分，但它们都指向一个相同的目标，即：实现可靠性的目标。企业应该提前预计到：企业需要花费大量的时间和精力，来定义SRE 的技术和文化与现有框架的交互方式。只是简单地说"提高可靠性"是行不通的。同样，如果企业期望的结果与可靠性无关（例如：成本、速度），那么就需要准备额外的工作成本，来让SRE 实践与企业的整体愿景进行适配。

启动 SRE 与人

随着时间的推移，流程和技术会潮起潮落，而人员和实践则能够接受和适应它们。如果，企业是从培训和招聘开始的，那么企业可以不断的添加或删除技术和流程。而建立 SRE 能力则是一个渐进的过程；所以，不要试图通过简单的招聘来取得成功。将招聘看做是培训方式的一种加强版，而不是取代培训。记住，SRE 需要一种 “生机文化”（与病态和官僚的企业文化并列） 才能取得成功，所以确保这一点至关重要。

拥抱自身的特殊性

在企业的某个特定组织内采用 SRE，其实并没有一个标准化的最佳实践的做法。企业能成功的方式才是它唯一正确的方式。Google SRE 现在已经对很多组织的工作成果进行了大量的研究，知道了一些行之有效的方式，还有一些无效的做法；然而，企业必然还是会犯一些新颖的错误。将这些组织的经验视为真正的学习工具，将各种有效的改进循环融入你当前的企业的组织中。

第二章：SRE 方法对可靠性的意义何在?

可靠性并非新鲜事物。企业一直将其视为业务系统需要不断提升的重要品质，无论是在服务质量、可靠性还是系统正常运行时间方面。那么，SRE方法又有何独特之处呢？为何当下引起如此关注？它与传统方法有何异同？对企业来说，又意味着什么？

将可靠性作为产品的关键竞争优势

为什么企业要建立 SRE 团队或者追求可靠性呢？他们希望实现什么样的结果？行业技术的流行趋势总是在不断的变化（技术、流程），但它们需要有实质性的商业价值才能被企业留存下来。考虑到可靠性和安全性，它们起初都不是产品明显的差异化因素，而是被视为必备的需求而已。只有在出现问题，并且期望产品具有高可靠性，或其依赖性时，它们才会变得更加突出。例如，多年前，安全漏洞和黑客攻击相对较少见，因此安全性是存在的，但并不会出现在面向消费者，或企业的产品营销材料中。现在，随着漏洞越来越普遍，并成为人们关注的焦点，我们将安全性视为产品的差异化因素之一。

可靠性（更常见的是可用性或正常运行时间）往往主要会在服务级别协议（SLA）和类似协议或期望设定的详细条款中提及。然而，我们在客户满意度（CSAT）评分、像 Downdetector 这样的第三方网站，以及将更多的生活和业务转移到互联网的整体趋势中，才会感受到可靠性的存在。在 COVID-19 疫情期间，许多软件即服务（SaaS）产品经历了业务的高速增长，并且不得不大幅提高对 SaaS 产品可靠性的期望。

除了“可用性”之外，作为可靠性特性常见的代名词，我们还可以想到：耐久性、数据持久性、在负载下的速度或性能、一致性和返回结果的质量等各种特性描述，与作为消费者级和互联网服务客户隐性的可靠性诉求相似的特性。

一旦我们理解到了：可靠性实际上是产品的高期待特性，我们甚至可以大胆地说，它是产品最需要必备的特性。因为，如果产品一旦不可用了，那么它的任何功能（增删改查...）也都无法发挥作用。如果产品在性能或质量方面的糟糕使用体验导致用户感到沮丧，那么产品服务将不会令用户满意。如果产品服务在高峰时段、关键业务时刻无法使用了，那么产品服务就会让用户感到不值得拥有，而离你而去。

谷歌搜索以"始终可用"的特点而闻名，以至于认为它是无处不在的服务（甚至用于测试网络是否能正常连接）。谷歌搜索的可用性是它与竞争对手进行对比时的关键差异化因素，与速度、质量、易用性和用户体验并列。这并非偶然，而是谷歌十多年来的一个刻意选择和投资的成果。

应该何时关注可靠性?

当初创公司考虑是否要在可靠性方面进行投资时，可能会认为可靠性还是为时尚早。特别是当他们考虑到像谷歌这样的大型组织所采取的全面措施时。这是可以理解的，因为：初创公司首要的工作是构建一个最小可行产品（MVP），而不是一个耐用的、有韧性的服务。然而，一旦产品的可行性确定了，那就应尽快将可靠性纳入产品路线图，与安全性和其他"横向"工作（国际化、可访问性等）一起开展起来。

在这些初创公司或早期企业中，关于可靠性的高成本定制化开发投资可能尚早，但与安全管理类似，在可靠性管理领域中，也有许多产品通过开源软件，以及通过第三方提供的服务和工具变得更加通用化。可以尽早的开始利用这些通用的工具和服务，从而避免在后期，不得不在已经发展壮大的复杂系统中进行痛苦的集成工作，或是被动响应可靠性问题。积极的前瞻性考虑是可靠性，及其相关准备工作的关键。另外，值得注意的是，尽管像谷歌这样的公司在内部构建了许多可靠性管理系统，但这绝不是最具成本效益的方法。利用外部的服务和工具是很早就经过验证的最佳实践。外部采购胜过内部自研是值得推荐的做法，特别是在诸如安全性（"永远不要自己编写加密算法"）和可靠性等领域，因为，自研可能会产生大量边缘场景和副作用。虽然，目前可靠性管理供应商领域的成熟程度和规模尚不如安全性，但它正在增长中，并且会对不断发展的公司产生重大影响。

在规划稳定性投资时，麦肯锡的“增长的三阶段”模型（见图 2-1）可能会对你有所帮助。它描述了公司未来发展的三种思考方式：

- 阶段 1 是当前已经很重要的工作领域。
- 阶段 2 是预期中新的增长领域。
- 阶段 3 是未来潜在的长期增长领域，目前处于研发阶段。

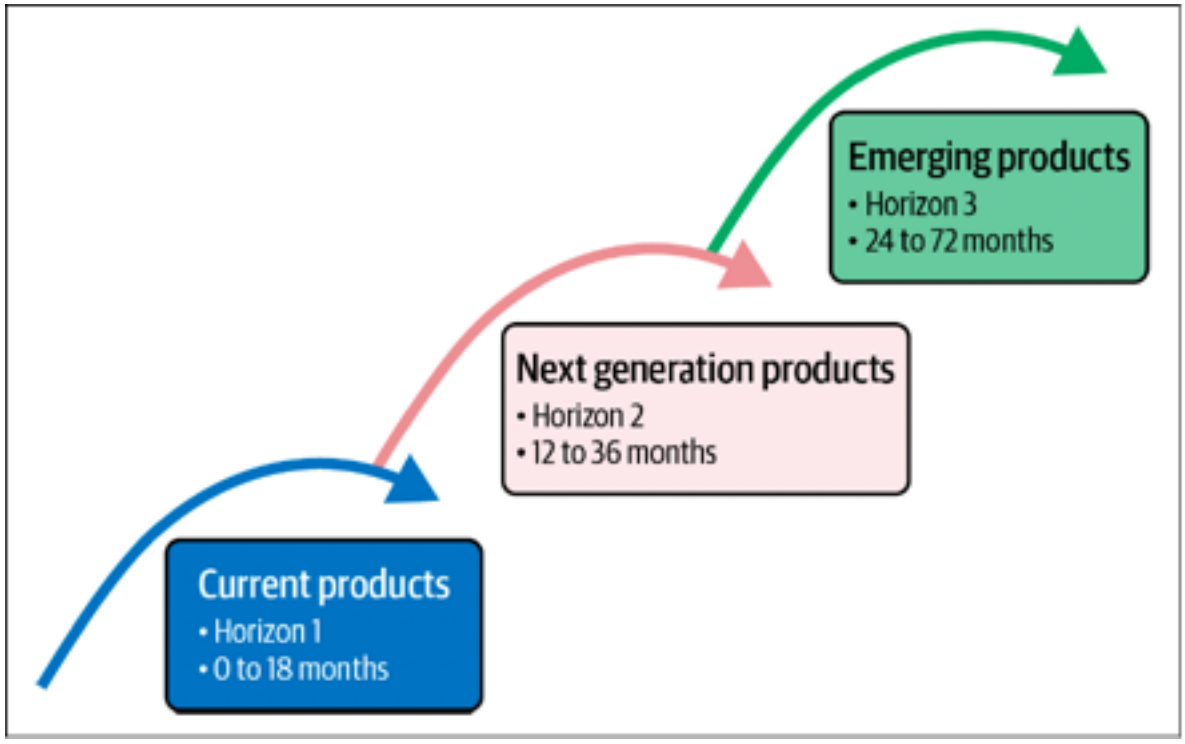

图 2-1. 增长的三个阶段。来源：Mehrdad Baghai、Stephen Coley 和 David White 的《增长的炼金术》（Basic Books 出版社）。

通过考虑对每个阶段 的不同投资水平，各个团队就可以拆解可靠性领域中不断涌现的工作内容。

首先，让我们从阶段 1 模型的产品开始，我们应该专注于确保：让可靠性工作能够为公司当前的业务模式保驾护航，同时助力短期需求的持续创新。这相关的工作包括：对于传统的运维工作，可以通过 SRE 实践进行自动化。其他工作内容还包括：服务监控（服务质量目标[SLO]）、事故响应和持续集成/持续交付（CI/CD）等等。

阶段 2 模型产品考虑的是：将核心业务扩展到新的市场和面向新客户。将现有的可靠性功能进行扩展，从而支持到更广泛的消费者，并可能在必要的情况下在全球范围内扩展基础设施。这些情况都将带来新的可靠性挑战，例如：分布式团队（7x24覆盖）、针对多个客户群体的容量规划、多区域部署，以及传统的维护窗口应用就不太现实了，这类工作事项本质上仅适用于本地化产品，而不适用于全球化产品（不是"所有用户都在深夜中"）。

最后，与阶段 3 模型产品相关的可靠性工作包括：公司可能扩展其业务提供的方式。为了应对未来颠覆性机遇或应对竞争威胁，公司应该用新的能力和新的商业模式来实现。投资于阶段 3 的公司将确保其平台和架构不会被绑定在单一的商业模式上，而是允许各种形态的系统的生成和演变，同时保持控制和质量标准。在这里，系统需要是可靠的，但不能僵化。诸如集中式批准委员会和自上而下的架构标准等工作会扼杀阶段 3 模型产品所需的创新。

因此，将 SRE 应用于阶段 1 可以对您当前重要的业务产生立竿见影的影响。将 SRE 作为阶段 2 的核心基础可以保障未来的成功。然而，阶段 3 并不是开展 SRE 的最佳领域，因为在那里做投资的可行性还不明朗。

为什么 SRE 在现在才开始流行?

为什么 SRE 不是在 20 世纪 70 年代或者 2010 年被发明并流行起来？显然，基于互联网的服务的复杂性近年来已经明显增长，尤其值得注意的是伴随着云计算的崛起。从商业上讲，我们将云视为分布式系统的后浪，而分布式系统是计算机科学中的一个深入研究的领域。只有在过去的十多年里，这个计算机科学的分支才对个人消费者（例如，Google，Facebook，

Apple）和企业（如 Salesforce）产生了重要影响，或者说，它的原则早已经被服务提供商（如 Akamai，Stripe）广泛有效地用于提供可扩展的互联网系统，更不用说云服务提供商了。

“数据仓库级计算”（这是 Google 的一个概念，将数据中心比喻为一个超级的数据仓库计算机的模式）的引入改变了企业构建、交付、运营和扩展服务的方式。这些新模式明显改变了企业对待成本（CapEx）的方式，从租赁或建造空间和购买计算机系统的资本支出模型转向了按需租赁计算服务片段的运营支出模型（OpEx）。然而，这还涉及到系统设计、架构和应对不断变化的故障模式的问题。

传统的基础设施遵循类似建筑行业或金字塔的模型：从底部向上构建的大而坚固的基座。如果基座出现了问题，对其上面的所有东西都会是灾难。我们将这种模型称为基于组件的可靠性模型，或者联合模型，即需要使系统中的所有组件都可用，系统才能正常运行；如图 2-2 所示，

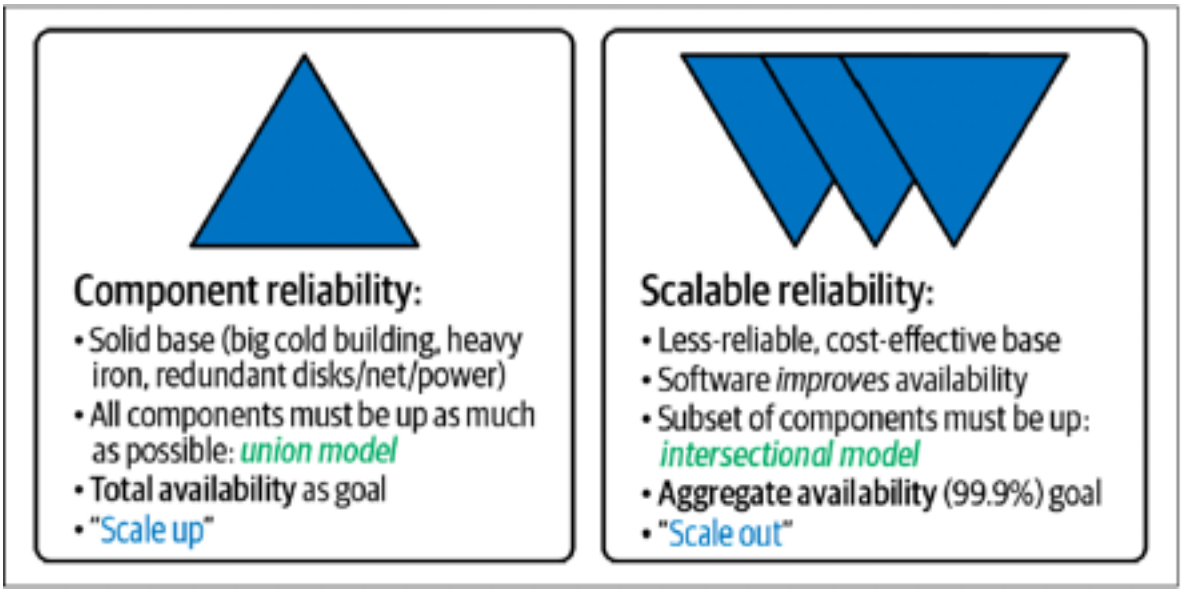

图2-2 , 可靠性金字塔

在云计算中使用的新模型是概率可靠性或交集模型，由于架构选择是期待故障的，所以只要系统的一部分子集可用，系统就可以正常工作。

虽然这个概念并不新颖或难以理解，但对于云服务的使用者来说，特别是当我们提出"迁移上云"的建议时，这并不显而易见，因为他们认为旧模型和新模型之间存在这等同性。虽然在新平台上运行旧模型是完全可能的，但必须考虑许多其他因素，这经常会让那些没有做好准备的人感到困惑。例如，传统的 IT 部门可能会为任何特定的虚拟机（VM）的正常运行时间而感到自豪，而对云 VM 则预期其实生命更为短暂：它们会被任意地创建和销毁，而且这个过程通常会非常快。

企业在像 Google，Facebook 和 Apple 这样的现代公司中看到了成功的案例，发现了两个主要的优势：（1）大规模创新，以及（2）大规模可靠系统。这些公司不仅可以构建新系统，而且可以保持它们的可用性，敏捷性和正确性。这个组合对于企业来说非常有吸引力，因为这使企业也能够快速响应市场需求，并向整个市场提供广泛的解决方案。

超越 GOOGLE 的光环

当然，Google 并不总是如此庞大。实际上，在早期，Google 以更传统的方式管理服务器群。使 Google 与众不同的是，它早期从垂直扩展转向了水平扩展其服务器群，也就是说，从购买更大更强大的计算机转向了购买更多更便宜的计算机。

你可以通过走过早期托管设施的走道看到这种变化。虽然在第一次互联网热潮期间，许多租户的机架里都有看起来很酷、昂贵的硬件，但 Google 却使用的是大量标准商用硬件，预期任何时候某个机器都可能会发生故障，并在软件设计中也考虑到这种故障。

引发这个转变的一个重要因素是，这是一个有意为之的选择：在进行水平扩展的过程中，避免相关的运营成也线性增长。也就是说，从财务的角度看，当水平扩展时，也持续雇佣更多人来维护新增的机器是没不合理的。

在这种技术和财务选择的双重推动下，Google做出了自己的选择——这就是 SRE 部门的诞生。Google 只是在大多数其他公司之前做出了这个选择，因为它是一个非常早期的互联网规模化公司。

我们相信，许多公司现在面临的扩展挑战与 Google 当时面临的挑战相似。只不过现在这些公司有公共云的优势，而不必在自己的数据中心中填充大量的标准商用硬件。我们相信，Google 能够通过发展 SRE 职能部门来克服这种方法的变化，这意味着 SRE 也可以帮助在其他公司克服同样的难关。

我们在 Google 学到的一个关于 SRE 与传统 IT 运维人员配备水平的重要观点是，那就是亚线性扩展。我们的意思是，运维一个系统的团队的规模不应该与系统本身的增长速率相同。如果你的系统的使用量翻倍了，那么你也不应该需要两倍的运维团队。Google 选择不按机器数量进行扩展，而是按其他更高级的指标进行扩展：集群、服务或平台。通过关注更高的抽象层级，团队可以做更多的事情，而开销更小。 这些抽象层级往往是由那些以前运维过这些系统的人来构建和扩展的。

复杂性可以增加对 SRE 的需求，但你的SRE人员配备应该比服务的接入增长得更慢，这个概念被称为亚线性扩展。这实际上直接与减少团队中的重复劳动琐事的原则有关。随着系统的增长和衍生，团队需要做更多的重复的任务来保持系统的健康。SRE 管理必须积极防止并跟踪这一点。如果团队允许过多的重复劳动琐事发生，不主动察觉，这可能会是一个下滑的开始，会导致团队无法维持，同时宕机时间也会增加。

为什么不保持传统的运维方式?

你的组织可以通过利用 Google 从垂直扩展转向水平扩展的经验，以及通过发展 SRE 而产生的相关变化，更早地享受到规模的优势，同时也可以节省资金。想一下另一种方式：为了降低扩展团队的成本，而随着复杂性的增加，团队的责任也在增加，人们也可能会从外部寻找更便宜的劳动力（例如，“外包员工”或“正确的离岸外包”）。这是处理规模和复杂性时一个太常见的方法。这通常会导致系统增长受阻，导致发生停机事故，实际上随着时间的推移，增加的成本会更高。这些意外的成本可能不仅来自停机或对品牌的损害，还可能来自于执行或扩展速度降低、新产品的上市时间延长，甚至最终被竞争对手超越等其他形式的业务收入损失。在选择如何优化运营和可靠性投资的成本时，组织需要考虑全局。

因此，配备合适的 SRE 团队是很重要的。其实你不需要都去雇佣博士学历，但你也不能吝啬。尽量不要只关注运维人员的单位成本，而是要关注整个系统的综合成本。做个类比，工业食品包装使用着价值百万的机器设备，将桃子装入罐头盒中。你可能会想，“为什么不雇佣非熟练工人？那会比百万美元的机器便宜得多。”乍一看，这听起来更便宜。然而，当你考虑到雇佣非熟练工人的整个系统成本时，实际上更贵。因此，使用价值百万的机器的整个系统最终比雇佣非熟练工人更好、更便宜。如果你让他们，SRE 和平台工程师来为你建造你的罐头机器人。不要因为你过去常做的事情，就强迫他们手动填装桃罐头。

如果没有高效能的员工，采用像 SRE 这样的高性能实践会更困难。那么现有的团队是不是就没有希望了？一点也不。完全有可能，并且强烈建议开发现有的人才。团队可能会试图雇佣一个外部专家，甚至是一个拥有 SRE 经验的外部团队，但这可能是一个错误。同样，期望通过外派员工或离岸外包获得（长期的）SRE 能力，也不太可能得到你期望的结果。SRE 人员的单位成本往往比传统的运维团队高，试图削减人员预算，或者想在在配备 SRE 团队时，以低成本获得高价值的方式往往都会失败。如果你的组织重视可靠性，你应该能够合理化这个成本，我们将在后面的部分探讨如何做到这一点。

将运维只看做成本中心是一个常见的错误。你应该考虑收入和总拥有成本的全局，避免局部优化成本，并认识到，只关注短期削减成本最终可能会让你的公司付出更大的代价。例如，

通过评估发生事故的情况，估计事故将对收入或品牌的影响，投资一个 SRE 团队的定位，就可以视为一种长期的保险，包括一系列的事故缓解和预防策略。理想情况下，这个团队不仅仅是“保险”，实际上还是驱动改善（可靠性！）向客户交付创新的驱动力。考虑以你的阶段 2 模型产品为目标，并统筹规划你的平台。不要只解决今天的问题；还要为将来做计划。

考虑转型现有员工的好处。只要给予正确的激励、机会和足够的时间，一个组织就可以改变其常态，并优雅地接受其人员的现代化角色和责任，同时也尽可能地减少了不必要的人员流动。因为，毫无疑问，一个组织最宝贵的资产永远是它的人员。在评估员工的技能集时，不应低估了他们对公司核心业务的真正理解。

第三章：SRE 原则

在我们探讨具体的实践方法之前，首先要明白的是原则的重要性，这就像法律中遵守法律的字面含义和精神含义。单单实践方法本身并不足够；SRE 的核心精神在于它的原则中。实践方法也面面俱到——它们只是原则的外在体现，也会随着时间和组织的不同而随机应变。

原则是你转型基础的基本真理，它们在你的决策过程中提供帮助和指导。实现业务目标通常有多种方式，因此鼓励人们充分理解和执行核心原则，要优于：设定一套面子工程的详尽的规则，那只让人按照字面意义去机械的执行，而忽视了核心精神。以 Google 的原则为例，虽然我们有多种：关于如何设计和构建新服务的内部政策，但我们始终坚持的核心原则是："以用户为中心，其他一切都会随之而来"。

你的重点应该是：激发每个层次的人，使其都展示领导力，而不是从各种方面剥夺他们的个人意识，不是用指令束缚他们。特别是，业务部门和经理需要认可转型的原因和动机，并必须愿意在他们的专业领域的范围内调整和进行详细的指导。一旦说服了这些有影响力的人，他们就会成为你最大的资产（资源），如果没有被说服，他们就会成为你最大的障碍。

与原则类似，良好的政策关注的是产出，而不是任务的完成；然而，它们更像观察视角方面的指导。它们是你用于牵引业务的抓手，而不是对抗业务流程的工具。政策和政策框架应该让人们在明确的界限内安全地运行。同时，它们也应该包含合理的默认设置，以便引导行为朝着正确的方向发展。

反模式：关于如何实施SRE，预先规划一个大而全的计划或设计。

本质上，你需要花费大量的时间来学习，我们建议：你构建一个由一致的原则指导的反馈循环（也就是说，通过反馈改进，形成一个良性的循环）。

我们将简要介绍 SRE 书中的每一个原则，以及如何在你的组织中应用它们。若想要获得更多细节，我们建议你阅读《Google SRE运维解密》一书中的相关章节。

拥抱风险

详见《GOOGLE SRE运维解密》第三章。

这是在初始阶段最难迈出的一步。我们通常把这个问题描述为：可靠性与速度之间的权衡；然而，这并不一定是真的。对企业来说，对于理解可靠性，最有帮助的方式，是将其与指数级的运维成本联系起来。大约每提高一个"9"（例如，从 99.9% 提高到 99.99%）都会导致成本增加一个数量级，无论是软件、硬件还是人力资源。考虑是否能从这种投资中获得良好的回报，有助于根据业务需求进行调整。故障的类型也非常重要。例如，需要全天候运行的服务会更适合实施 SRE（相比那些公司内部的一周只运行 8 小时，每周 5 天的系统）。另外，对于那些没有得到积极维护的服务，SRE 的作用将会大打折扣，因为这些服务在持续改进方面的机会本来就会很少。尤其是在你故意不进行太频繁的版本发布，或不在编写新的代码时，这点尤为明显。

反模式：服务 100% 可靠性目标。

对于几乎所有事情来说，100% 都不是正确的目标。

反模式：在"常规"的运维中实现了 99.999% 的可靠性

月度指标或维护窗口可能会掩盖灾难带来的巨大影响。

服务质量目标

详见《GOOGLE SRE运维解密》第四章。

在你开始考虑实施 SLO 和 SLA 之前，先从服务质量指标（SLI）开始，并根据你系统的实际观测数据，来制定和校准的 SLI，然后用来支持 SLO/SLA 协商（利益干系人间的对齐）。不要让你现有的业务承诺影响你对 SLO/SLI 准确性和相关性的判断----你可以选择使用指标来驱动工作的变化，或者反之亦然。总之，不能进行粉饰，或者挑选优点展示。而要，花时间理解你的客户想要的是什么，而不能，为了支持&证明你理论，使用方便的数据点。简单来说，让事实证据（SLI/SLO）来驱动你的结论（SLA）。尝试关注那些 >99.9% 的服务，对于 <99.9% 的服务而言，SRE 可以先不参与相关维护工作（直到它们产生了需求）。我们反复强调：如果一个服务不从 SLO/SLI 中受益，那么它可能也不会从 SRE 中受益。最后，如果在 SLO 不违规的情况下，你就不能对软件或流程做任何变更了，那么 SRE 对你的收益也会甚微。

反模式： SLO = SLA

你应该总是将 SLO 设置得比 SLA（例如，SLO：99.95%，SLA：99.9%）更严格。

反模式： SLI = OKR（目标和关键结果）/KPI（关键绩效指标）

Goodhart 的法则在这里适用：当一个度量成为了目标，它就不再是一个好的度量。

消除琐事

详见：《GOOGLE SRE运维解密》第五章。

这可能是最重要的原则之一，因为它与 SRE 成功所需的创新文化密切相关。大多数时候，企业领导层希望加快进度，并通过确保将所有资源都 100% 的利用来实现这一点。如果你真的希望能确保：你的团队正在做正确的事情，而不是快速地做错误的事情，那么你的目标应该

是少于 50% 的繁琐工作（或者我们所说的琐事）。这是可靠性（和速度）在大规模下的秘密。不要将其等同于技术债务，即可以把它们都攒起来，以后一起偿还，或者以“琐事周”的形式，每个季度解决一次这个问题。一旦琐事压制住了你的团队，那么所有 SRE 的所有其他活动都会停滞不前。你必须为组织来定义：什么是琐事，并且这必须由 SRE 实践者来决定，而不是自上而下的指令。琐事的定义也会随着时间的推移而改变（定义再次由实践者更新）。

反模式：将琐事作为一个可有可无的原则

忽视消减琐事会对应用 SRE 产生很大的影响。如果你没有时间减少琐事，那么你就没有时间实施 SRE。

反模式：琐事是某个人/某个团队的工作，而不是每个人的工作

最接近工作的人需要是修复它的人。如果你试图把这个工作转嫁出去，它会驱动错误的行为。

反模式：琐事清除周

每季度举行一次“琐事清除周”是常见的诱人的做法，但这是这样是行不通的。你需要采取更系统、更持续的消除琐事的方法。

监控分布式系统

详见《GOOGLE SRE运维解密》第六章。

可观测性是一门独立的专门学科，它需要像其他开发实践一样受到同等的重视和思考。实际上，大多数企业应当预期投资于多种系统，这些系统将帮助团队更高效地工作。单一的监控平台（大而全的统一控制台）并不能很好地运作；同样，使用数百种功能叠加在一起的工具也不可行。通过理解你独特的 SRE 用户路径，以及他们需要使用多种工具来诊断和解决系统

间的逻辑关系，找到适合你的平衡点。将可观测性系统视为：需要投资和精心设计的内部产品，强调工具的实用性，而不是"完美"的仪表板，因为系统总是在变化。记住，告警过度和告警不足同样的糟糕：告警不应直接发送给人类，除非需要他们采取行动。构建这种告警学习循环是加速学习的常见方法；弄的不合理，会迅速使 SRE 精疲力竭。

反模式：告警信息过载

告警的电子邮件充斥了你的收件箱，我们会忽略所有信息，这意味着高优先级的告警也无法得到必要的响应，因为告警噪音太多。

反模式："NoOps"工具会替代 SRE

工具可以增强 SRE 的能力，但还不能替代他们。完全消除运维是不可能的，这样会迅速的让你的 SRE 团队渐行渐远。

反模式：告警即是原因

你可以记录很多事情，但告警总是针对症状而不是原因发出。

GOOGLE 自动化的演变

详见《GOOGLE SRE运维解密》第七章。

当涉及到极高的可靠性水平 (99.99% 或更高) 时，自动化是最重要的，因为在这个时候，如果需要人工介入，你几乎总是会经历的的是：服务水平目标 (SLO) 违约。随着系统错误预算的逐渐缩减，干预的平衡点也随之变化，最终转变为主动维护，会采用的技术手段包括优雅降级、重试等。自动化本身也会成为一个常见的问题，花时间修复不良流程是非常重要的，但很难融入团队文化。自动化也需要和系统其他部分一样的容易维护。

反模式：不管流程的质量或适用性，默认一切自动化

最好的代码是并不写的代码！对于不很频繁的流程，操作手册（Playbook）是一个很好的中间解决方案。

反模式：对于"非常重要"的部分也不要人工介入

只有当你真正需要又人来做决定，并且他们有权这样做时，才让人工介入。

发布工程

详见《GOOGLE SRE运维解密》第八章。

发布工程与你的 DevOps 团队可能已经在进行的持续集成/持续交付 (CI/CD) 实践有广泛重叠。要充分利用这些现有工作，而不试图自上而下的强加另一套实践。强调结果和流程指标以对齐团队，并确保你在一个平台团队 (或根据规模的不同而有多个团队) 上有足够的投资。

尽可能提前发布相关的工作，即尽早让测试团队也参与进来，并在所有阶段考虑测试。

不要让开发人员负担过重，但确保发布周期的每个部分都被视为有价值的，并与其他部分保持一致。对于 SRE 来说，发布流水线是导致大多数问题的原因 (因此也是解决问题的关键)。与值班和维护人员也需要紧密的配合。

反模式：DevOps/SRE 团队负责所有的发布

那是让不同职位的人都来干运维的活。

反模式：发布工程必须引入 CI/CD

持续交付本身就是一门学科，你的平台和开发团队需要在这方面打好基础 (SRE 可以提供帮助)。

简单性

详见《GOOGLE SRE运维解密》第九章。

团队的认知负荷很重要，并且会随着团队职责的扩展或缩减而变化。确保允许团队合并或拆分，从而让认知负荷匹配。基本上，复杂性意味着很多事情都会很难理解；因此，要尽可能的激励：减少不必要的复杂性，并将复杂的事情拆分成更小、更易管理的部分，例如领域驱动设计 (DDD)。另一个从 DevOps 中重用的重要概念是：高上下文 (High Context) 与低上下文 (Low Context)，以及 SRE 的一些概念如操作手册（Playbook）、文档、灾难恢复测试 (DiRT) 演习等，这些都是使事情变成低上下文的重要部分。拥有更少的代码和更少的产品特性，可能会与大多数产品的激励相悖，因此：要考虑其可靠性影响时，请确保对此进行控制。

反模式：简单意味着我能理解它

用一个高管专用的仪表盘，并不可能有意义地显示所有内容。更不要试图强行的实现它。

反模式：基于年度评估的静态团队

动态团队的形成需要一年多次。

如何导入这些原则?

如何将这些原则映射到你组织？将这些原则完全与您的组织对齐的可能性很小，但这没关系！你的 SRE 版本并不需要完全和 Google 的相同，只需要原则一致。但是，请确保你特意地选择将要追求的目标，检查与现存原则之间的差异，并利用这段时间仔细检查面子指标 (参见 Eric Ries 在《精益创业》中解释的"成功剧场")。在不稳定的基础上进行变革可能很难，因此如果你没有信心，请假设你需要检查和改变。尽量不要在原则上犹豫不决，如果你认为某事无法完成，那么推迟它的实施，要比假装工作更好。

防止组织破坏性错误

变更可能会潜在的产生非常不同影响。采纳新原则时，进行的一些变更不一定总是有效。变更所带来影响，通常比能否恢复原样更不重要，这意味着最难逆转的变更，通常也会造成最大的痛苦。专注于更容易逆转的改变，即使这些变更是错误的，它们仍然会带来经验教训。例如，如果第一次重组不成功，你可以随时进行另一场重组，但你不能让解雇的人重新回到公司。

反模式：所解雇的运维人员都不会编程

除了显而易见的道德或法律影响，你根本无法逆转这个决定。

反模式：给所有开发人员生产的 root 访问权限

良好的安全和运维实践包括：与过往的任何时候相比，最小可用权限最适用于自动化。

反模式：选择业务中最关键的系统作为 SRE 的试点

你不会在马拉松训练计划的第一天就跑 26 英里。

建立安全失败的环境

为你的采纳之旅创建一个安全失败的环境，期望失败会发生，但确保你从中学习并长进。在做复杂的事情时，请确保有主题专家（SME）的参与，但在做复杂的事情时，请确保你要么奖励失败，要么有失败预算。在大多数组织中，真正奖励失败很难，因此有时失败预算更合适。这意味着：你根据成功的前 n% 进行衡量，而不是平均/中位数。领导团队中这些行为的榜样作用至关重要，否则他们将无法在整个组织中融入这些行为。

反模式：我们会支持你的任何冒险，只要结果成功就行

真正的风险预算意味着接受一系列的失败。

当心优先级分歧

整个领导团队完全支持你的可能性很小。更可能的是，人们想要可靠性，但对变更和成本有合理的担忧。我们建议承认变更的 J 型曲线，如图 3-1 所示，这意味着在一开始的几个相对容易的胜利之后，实现有影响的变更的曲线变得困难。例如，采用自己的新自动化可能感觉像是一个倒退，然后才会实现显著的收益。通过进行屋顶射击而不是月球发射来确保成功。你仍然可以追求显著的改进，但一开始要保守一些。

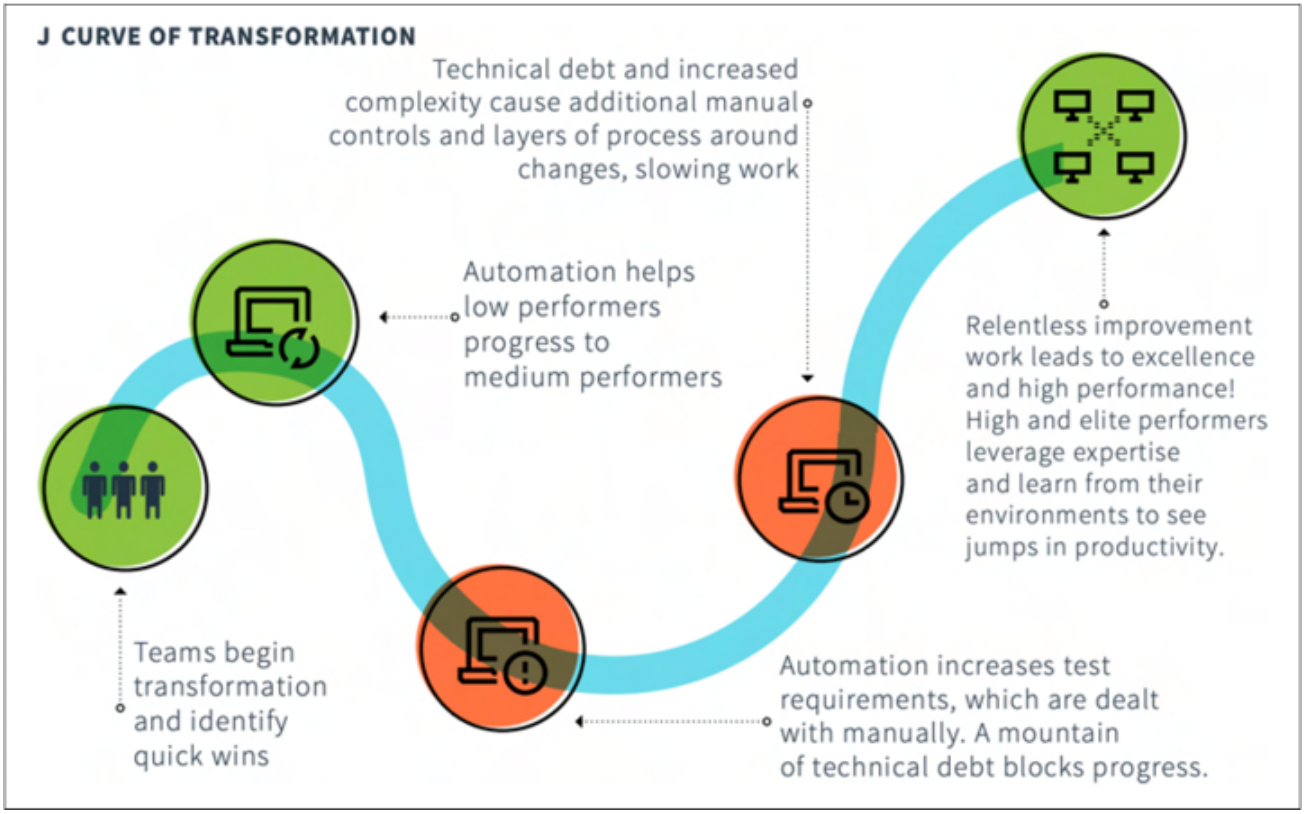

图 3-1. 变革的 J 曲线，来自 DORA 2018 年 DevOps 状态报告

反模式：过早放弃。例如，尝试 SRE 六个月，然后在还没有好转的情况下就喊停了

这并不意味着你需要立即完成所有事情，但在几个季度之后，必须有一个明确的方向感，表明在朝着正确的方向前进。

如何取得支持

如何获得这些原则的支持，获得你需要的关键批准和支持？通过考虑 John Kotter 或 BJ Fogg 提到的一般企业变革原则，确保你为 SRE 成功做好准备。即使你的领导团队不完全相信你正在尝试的事情，也没关系，但你需要确保至少有足够的紧迫感来进行变革，并有动机去实施它。

在技术领域，我们经常奖励解决问题，而不是防止问题的发生，SRE 原则和实践的采用，可能会成为这种操作模式的牺牲品。通过找到适合你组织的指标来确保 SRE 采纳的持续价值是显而易见的。例如，在零售业，你可能专注于在黑色星期五期间最大化销售额，而在医疗保健领域，你可能专注于持续合规性和可用性，在金融领域，可能是关于交易系统的吞吐量或完成分析流水线处理的速度。

反模式：如果你构建 SRE，他们就会来

实践不能孤立存在。你必须开始实际的工作，才能做出真正的改进。

反模式：稳步上升的进展

现实世界的变革有起有落。如果你没有失败，那么你就没有在学习。

第四章： SRE 实践

一旦你建立了第一个 SRE 团队并掌握了这些原则，就该制定一套实践了。团队的实践由以下组成部分：成员能做什么、他们知道什么、他们拥有的工具，以及他们对这些工具的使用舒适度。

团队的任务和环境最初决定了他们的工作内容。通常情况下，这意味着那些“开发团队未做的一切”，可能是一些列危险的稳定性缺陷。通过将团队聚焦于部分运维职责，他们可以产生一个能力循环，随着时间的推移相互增强。如果将他们投入到一个未定义范围的复杂环境中，那么结果必然是琐事和沮丧。另一个常见的反模式是将 SRE 工作添加到已经超负荷的团队中。

团队的知识可以通过教育来扩展，无论是通过自我组织的还是企业集中组织的形式。应该鼓励团队定期举行团的内（间）的交叉培训会议，例如，每周一小时的会议，欢迎研讨任何关于生产环境的问题，无论是新成员还是资深成员都参与其中。如果问题有人能回答，则可以进行经验学习。如果没有人知道答案，可以转变为协作调查。根据我们的经验，这些会议对团队中的每个人都非常有价值。初级团队成员可以学习到新事物，资深成员也有机会传播他们的知识，通常还会发现一些大家都不知道的新事物。类似地，可以开展厄运之轮“Wheel of Misfortunes” 或桌面演练，在非正式的环境中，让团队成员讨论：他们在紧急情况下的角色和响应方式，对于让人们在无压力的环境下，更加舒适地接触生产环境非常有帮助。重现最近的所发生的故障是一个容易开始的起点。如果一个团队成员可以扮演指挥者的角色（会议主持人），并按照现实生产工作中的情况展示证据，其他团队成员就可以讨论：他们会怎么做和/或直接使用什么工具来调查事故发生期间的系统。

还应该鼓励团队：从开发团队那里获得更多，有关他们正在运维的系统的知识。这不仅是更好地理解现有系统的很好的练习，也是直接引入新监控工具、讨论和规划系统变更（如性能改进）或解决可伸缩性或一致性问题的机会。这些对话在建立团队之间的信任方面通常非常有价值。

团队的能力也可以通过引入新的第三方工具、开源软件工具或团队编写自己的工具来扩展。

从哪里开始?

在为我们的团队赋予新的能力时，应该从哪里入手呢？可靠性和 SRE 的领域非常广泛，并不是所有 SRE 都能适合于一个全新的团队。我们建议从一套能够帮助团队学习成长的，并有具体的下一步工作内容的实践开始。抽象地说，我们推荐使用所谓的"计划－执行－检查－行动（PDCA）"模型。通过根据系统当前的工作状态来决定下一步计划，你的每一步都将需要是能落地的。我们将在本章后面的内容里，解释如何构建这些能力的平台，以及建议从哪里开始。这些初期的能力将形成一个良性循环，团队就不必猜测：接下来应该构建或采用什么，他们将能够根据对系统的观察自然而然地得出下个迭代的工作内容的结论。

你的目标是什么?

设定适当的目标非常重要。并不是所有系统都需要达到"五个九"（99.999%）的超高可靠性。我们建议根据你的服务和应用的可靠性需求进行分类，并相应地设定投入水平。如我们之前提到的，记住每提升一个"九"成本可能是前一个"九"的十倍，即99.99%的成本可能是99.9%的十倍。虽然这个说法很难精确证明，但这个原则是成立的。因此，如果在没有深思熟虑的情况下，就盲目或过于广泛地设定目标，可能会让你的投入变得非常昂贵，并在投入中陷入新的困境。过分追求不必要的高可靠性目标也可能导致团队顶尖人才的流失。如果你的目的只是到达近地轨道，那就没有必要设定登月的目标。

确保你成功的路径是一个可循序渐进实现的目标，而不是期望通过一次性的大型项目或革命性变革来实现。在这里，渐进式改进才是关键。

当你的团队开始尝试新的实践时，确保记录下所取得的成果，并在团队内部及对外宣传这些成果。同行间的认可非常重要，可以通过在团队站立会议中表扬成员、在会议上让人们分享他们是如何避免灾难的、在内部通讯中发布近失事件，以及向整个组织展示如果没有采取预防措施会发生什么等方式来进行。庆祝这类工作成果非常重要，尤其是在过去没有这样做习惯的环境中。口头表扬和书面表扬可以与奖金或礼物相结合。即使是小礼物，也能产生很大的鼓励作用。

如何到达目标?

不要制定一个长期且过于详细的计划，例如三年的详细规划。相反，你应该专注于确定前进的大方向。了解你的长期目标，但根据当前完成的任务来决定下一步。确立了方向之后，不必急于全面改变现有的团队和流程来适应新模式。相反，应该逐步引导团队步入正确的方向发展。

我们将这种方法称为"战争迷雾"策略，意味着你清楚最终的目的地，但也为途中可能遇到的任何小问题做好了准备。在这个过程中，短期规划和灵活性至关重要，尤其是在初期，迅速取得成效，并立即展示其影响力，对于一个刚刚起步的项目和团队的士气有着极大的积极作用。设定可达成的短期目标来解决当下问题，同时开始构建可通用、可复用的长期能力，让多个团队都能受益。通过构建一个能够提供这些能力的基础设施，你可以放大投资的影响力。我们将在本章后面进一步详细阐述这个平台和能力的概念。

组织内不同的产品开发团队在需求和现有能力方面都有所不同。在引入 SRE 时，你应当在参与模式（engagement model）上保持灵活，以适应各个团队的具体情况。通过理解产品团队的现状，你可以解决当前的问题，同时引入全组织范围的标准和最佳实践。当 SRE 团队刚开始运作时，如果有很多团队都想寻求他们的帮助，他们可能会感到负担过重。通过制定明确的参与"菜单"，你可以避免一次性的合作，或其他不可持续的合作模式。参与模式有几种类型，包括嵌入式、咨询式、基础设施支持等，这些在谷歌客户可靠性工程（CRE）团队的博客文章以及《Googel SRE 运维解密》的第32章中有很好的描述。

对于 SRE 的采用，明确的汇报结构在早期就很关键。我们建议建立一个独立的组织，并且 SRE 领导应在管理团队中拥有一席之地。通过将 SRE 的领导层与产品开发部门分开，SRE 团队将更容易专注于可靠性这个核心目标，而不会受到那些更关注速度和功能交付的团队的直接压力。然而，在这样做时，要避免形成一个孤立的"运维"部门，因为 SRE 与企业其他部分的紧密合作至关重要。开发团队应该与这些共享的 SRE 团队开展紧密合作的投入，从而确保从 SRE 团队获得的价值大于自行构建 SRE 功能。

SRE 成功的关键

SRE 的成功不仅仅取决于实践方法，如服务水平目标（SLO）和事故回顾（postmortem）。这些实际上是创造 SRE 工作文化起步阶段的产物。因此，成功地采用 SRE 不仅需要模仿这些实践，还必须采纳一种兼容的文化才能取得成功。

这种文化建立在团队自身的信任和安全感之上。当团队负责控制重要系统时，他们必须感到在心理上是安全的。他们必须能够在不担心惩罚的情况下对同事和领导说"不"。他们必须感到自己的时间被重视，他们的意见受到倾听，他们的贡献得到认可。最重要的是，SRE 不应该感到自己比开发部门的同事更为"另类"或"次等"。基于历史上的原因 Dev 与 Ops 对立模型是一种常见陷阱。

无责事故回顾就是一个著名的例子。通过记录"出了什么问题"，团队可以协作地确定导致故障的各种因素，无论是技术问题还是程序问题。经常，当人为错误发生时，将错误归咎于"人的因素"可能很有诱惑力，但这已被证明是没有意义的，也不是改进系统的有效方式。相反，SRE 倡导无责文化。可以这样理解：系统应该让人很难犯错。应当有自动化和检查措施来验证操作者的输入，并且鼓励用同行审查（peer review）来促进双方的共识和协作。当人们在报告中自由地提及自己的名字，并且知道不会因为可能发生在任何人身上的简单错误而受到羞辱、降级或负面绩效评估时，这表明你已经实现了无责事故回顾。如果你看到事故回顾中使用了"工程师"或用"人员1"这样的表述，你可能认为这是一种良好的无责实践，但这实际上

可能反映了潜在的文化问题，必须直接解决。如果文件中的名字被隐去并替换为“工程师”或“人员1”，但在事故回顾之外仍然对工程师进行指责，那么责任文化问题就没有得到解决。你绝对不应该自动化地从记录或文件中删除任何人的名字——由于这并不能解决根本的文化问题，只会使文件更加难以阅读和理解。与其表面上删除名字，不如直接解决潜在的文化问题，从而实现真正的无责文化。

一个不良文化的标志是 **西瓜指标**： 外表看似绿油油，实则内部问题重重。这类指标反映了团队的努力，它们被精心设计，看上去很美好，但实际上隐藏了真正的缺陷。这与古德哈特法则类似，即任何成为目标的指标就不再是好的衡量指标。例如，过分关注支持工单数量，或整体平均解决时间（MTTR）往往会被滥用，不管是故意的还是出于好意，但都不主动认知自己的错误。通过度量团队的活动，我们把这些活动变成了目标，而不是客户的成果。相反，团队应该定义自己的成功指标，这些指标直接反映了客户满意度、系统稳定性和开发速度等因素。

SRE 不应仅仅被视为“20%的时间”角色，而应该是组织内一个明确的职位和头衔。应该有一个公开的职业晋升路径，包括转岗要求和晋升期望。团队间的级别和薪酬应该公平对等。转岗不应该有任何显著的影响。

判断一个成熟的 SRE 团队是否成功的好方法是：观察人员转入和转出SRE的情况。确保人员转移是常规的且没有任何官僚主义或限制，这样你可以快速了解人们是否感到在 SRE 团队中“被困”，或者它是否是一个理想的角色。通过观察从开发转移到 SRE 的自愿转移率，你可以判断这一角色是否吸引人。

SRE 必须知道他们的时间是被重视的，尤其是当他们的工作需求超出了“正常工作时间”。例如，在谷歌，当 SRE 需要在正常工作时间以外值班时（即“on-call”），他们应该获得加班费补偿。谷歌的一些团队允许值班工程师在金钱补偿和休假之间二选一，按值班时间的一定比例看，通常设置一个值班时间的上限。不应该对一个团队提出超出其交付能力的需求，因此，确保值班池足够大是很重要的。常见的错误是：仅将值班池限定为 SRE 人员，这是不必

要的限制。值班池也应该是基于自愿加入的。一旦团队感觉他们的时间被滥用，就会导致士气的迅速下降。

另一个文化抓手是规划和目标设定。由于 SRE 最接近生产问题，他们通常很清楚最重要的是什么，哪些问题最紧急，哪些问题造成了最大的痛苦。允许 SR E团队设定自己的优先级和路线图，你就赋予了团队权力，他们将会更加有效率和愉快的工作。管理层应遵循共同制定，并达成对预期成果的共识的做法。业务需要加速开发吗？用户需要更快获得发布结果（新版本）吗？一个常见的反模式是泰勒主义，即领导者专权设定和优先考虑详细的计划和任务，然后将它们分配给下属。

构建能力平台

SRE 团队可以构建一个平台，向合作团队提供能力，理想情况下，随着时间的推移，他们的贡献将扩展到整个组织。通过在共享服务、实践、规范和代码中引入弹性机制，这些团队可以开发出由：自动化、代码、共享库、管道、流程、规范、文档、手册组成的共享平台，甚至包括那些只存在于人们头脑中的特殊未记录知识。与其让每个团队都试图创建自己的最佳实践，不如将这些实践融入到平台中。产品可以从头在平台上构建（所谓的“数字原住民”），也可以移植到平台上。随着平台能力的增长，团队对其操作特性越来越有信心和体感舒适，逐渐将更加关键的工作负载都可以迁移过来。通过采纳这种将能力编码到平台中的模型，SRE 团队可以通过将能力同时应用到多个服务中来放大他们的影响力。平台是一个内部产品，应该像产品一样进行管理，将服务团队视为客户，接受功能请求，并跟踪缺陷（见图 4-1）。

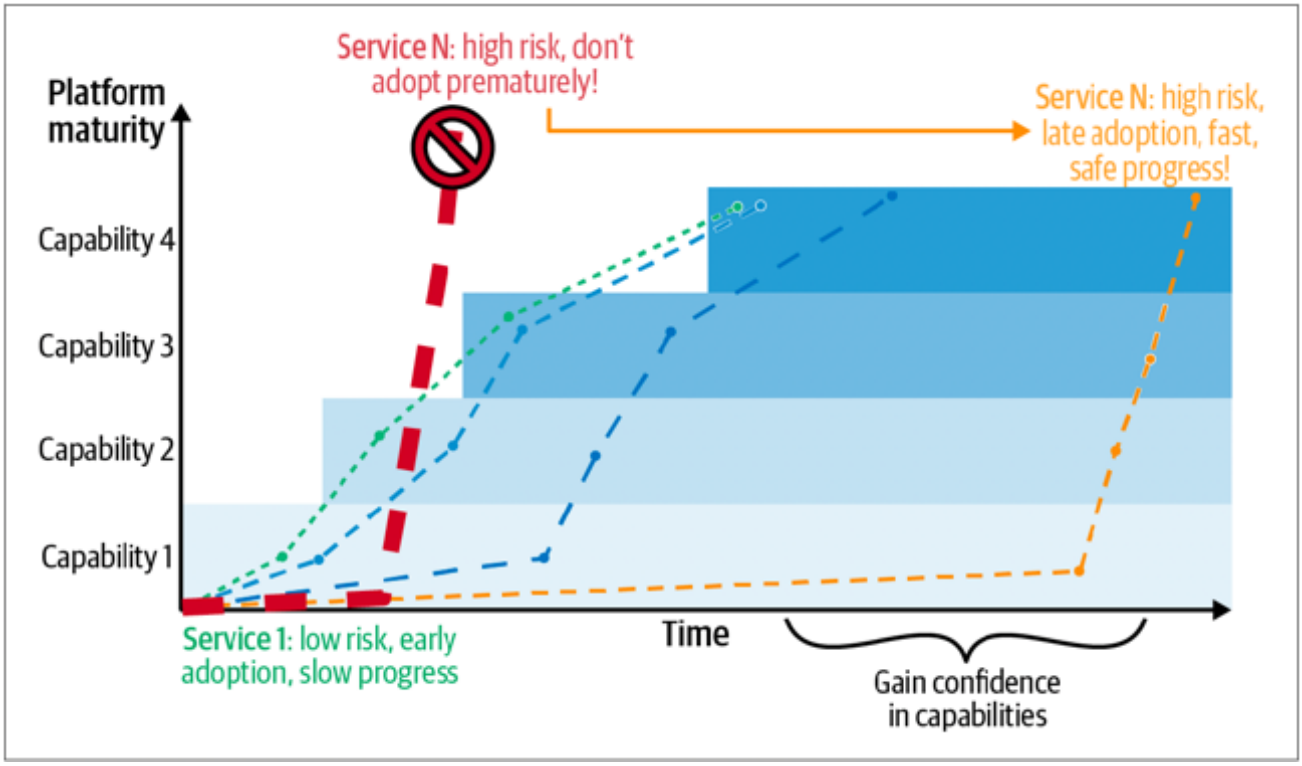

图 4-1. 能力平台示意图

当一个 SRE 团队开始来构建这个平台时，他们会面临一个问题："首先应该构建什么能力？"通过分析前期采用 SRE 的低风险服务，你就可以将需求列表最小化为一个最小可行产品（MVP）。随着时间的推移，你会添加更多的能力。但下一步是什么呢？有两个来源：你的开发人员和你的环境。也就是说，根据他们的需求来构建，例如，"我们需要一个消息总线！"以及根据你所知道他们将需要的来构建，例如，"必须有一个可扩展的服务发现系统，否则这将无法工作。"

对于环境所需的能力，这些通常包括：

- 对 DevOps 进行优化改进，例如：增强软件开发生命周期（SDLC），更快、更安全地发布更多代码；
- 可靠性工程的改进：最小化那些已经潜伏于系统中的错误带来的风险。

为了增强可靠性工程，我们推荐在团队中培养一种持续改进的机制。如果你不确定应该从哪里开始，可以从分析已经发生的系统中断事故入手，采取以下措施：

- 制定服务水平目标（SLO），明确可接受的系统性能标准。
- 建立一套正式的事故响应流程，确保在发生故障时迅速有效地响应。
- 开展无责任的事故回顾和审查，鼓励团队成员坦诚地分析问题，而不担心受到指责。
- 利用风险建模来确定改进的优先顺序，确保优先处理最关键的风险点。
- 使用错误预算或其他风险管理方法，来消化待解决的可靠性问题，以保持系统的整体稳定性。

让这种持续改进成为推动团队不断创新的动力。例如，如果有一次的部署导致了整个服务器群的宕机，你可能需要探索减少此类风险的方法，比如通过限制影响范围、实施金丝雀发布，或使用其他灰度（渐进式）部署策略。如果发现了内存泄露问题，你可以在部署的前置流程中加入新的压力测试方法。这些新能力将被集成到你的基础设施平台中，为平台上的每项服务提供额外的保护和收益。随着这些通用的缓解策略逐步的证明了它们的价值，单次修复的情况将会大幅降低。

领导力

想要构建一个这样的技术平台，你就需要投入宝贵的工程资源，这些资源本可以用于开发新的业务功能。这就要求从基层到高层都需要施必要的加影响力。在功能开发和系统稳定性之间进行权衡时，决策者必须具有全局视野并得到适当的激励。我们越来越多地看到“首席可靠性官”这一职位，他们在组织内应当占有一席之地，参与制定战略性的可靠性决策（这一概念在马克·施瓦茨的《在桌边》一书中有所讨论）。虽然这是 SRE 成功实施的关键角色，但并不是一个常见的职位名称，通常是现有高管的额外职责。

了解效果

一个运作良好、重视可靠性的组织会展现出几个明显的特征。首先是面对可靠性问题时，能够及时缓解或暂停新功能的发布。如果唯一目标是快速发布，可靠性和其他非功能性需求就会被牺牲。你的组织是否总是将可靠性工作位居新功能开发之后？是否有因"时间不足"而永远无法完成的项目？重要的是，这并不意味着：你应该放慢代码交付流程的速度，而是保持持续的推进。

另一个成功的迹象是，个体英雄主义不再受到赞扬，反而积极的劝阻。当系统的成功依赖于少数人的担当时，团队就会形成一种不可持续的英雄主义文化，这种文化终将崩塌。英雄们会被鼓励独占特定领域知识，而不愿意去系统地预防问题的发生。这与《凤凰项目》一书中的布伦特角色类似。依赖个体英雄不仅效率低下，而且可能非常危险。团队必须积极阻止个体英雄行为的发生，同时保持团队责任感，因为英雄主义在短期内看似合理，但实际上不可取。

一个表现良好的团队还会在出现中断前就对可靠性工作进行投资，并作为主动规划的一部分。在表现不佳的团队中，我们看到仅在发生一系列中断之后，才开始对可靠性进行投资。虽然这可能是必要的增加，但这种投资需要长期维持，而不是被视为一次性的应对措施或在情况好转后就被撤销。

进一步说明，设想你的组织对于可靠性有两种不同的态度：和平时期和战时。它们分别对应于"一切正常"和"所有人都知道问题即将爆发"。通过区分这两种状态，你可以做出关于投资的决策。在战时，你会在平台的隐性特性、基础设施、流程和培训上投入更多的时间和金钱。而在和平时期，你也不会放弃这些工作，但你肯定会减少投资。

然而，谁决定公司何时进入战时？这个决定是如何做出的？它如何在整个公司传达，而不造成恐慌或人才流失？一种方式是使用优先级代码，比如黄色警报或红色警报。这些是帮助团

队确定工作优先级的组织实践。黄色警报意味着在一个季度内，当前的技术问题可能会变成一定的业务紧迫性。红色警报则表示问题可能在几天内发生，或者是已经存在的稳定性威胁。这些警报应该有明确的标准，所有领导团队成员都必须理解并同意这些标准。宣布这些警报必须得到领导层的批准才能产生预期效果。这些警报的结果应该是改变团队当前的工作优先级，可能会暂停现有的工作（如红色警报的情况），批准大批量投入，并能够直接协调其他团队来提供帮助。优先级警报对企业来说是一项代价昂贵的操作，因此你应该确保它们有明确的预期结果。这些结果应该在一开始就定义好退出标准，并在完成时清晰地传达出结束的信号。否则，团队会经历信号疲劳，而不再进行适当的响应。

选择投资于可靠性

那么，作为一个可靠性领导者可能会做出哪些较不戏剧性的改变呢？这将涉及政策和投入。当政策是从基层向上的推动时，那么设置的全组织范围政策往往会是不一致的。如果存在一个领导的角色，由他来进行审核、消除重复、批准和传播这些政策，那将更为有效。同样，公司的资金支出，包括人员、硬件、软件、差旅和服务，通常是以分层的方式进行的。

在构建前面提到的结构之前，必须考虑组织内可靠性的价值。为了使其有意义，组织必须将可靠性视为一项投资，甚至是业务产品差异化的竞争因素，而不是成本中心。应该明确可靠性是最重要的隐性产品特性。一个不可用、卡顿或充满 bug 的产品，无论具有多么丰富的功能，对客户的价值都会大打折扣。想要设定这个方向，你必须在高层进行操作，以确保持一致性，特别是：如果这是一个新的方向。

一个简单的论点是，可靠性还可以作为更容易理解的概念的代名词，例如代码质量。如果系统出现了用户可见的问题，那么诸如灰度变更等可靠性实践的应用，就可以使系统在直接解决代码质量问题之前，让最终用户感觉更加稳定。例如，通过只对1%的用户发布带有缺陷的版本，那么99%的用户就不会发现问题。这使得系统看起来比实际上好100倍，并显著的降低了支持成本和声誉的损害。

做出决策

在你将可靠性设定为对更强大产品的投资后，你就可以制定更长远的计划了，并会产生更大的影响。传统模式将 IT 视为成本中心，完全倾向于随着时间的推移逐渐降低成本投入。但最终，如果服务不可用了，无论它的后续运维成本是多么的廉价都将失去意义。你仍然可以实施成本的削减，但应该是在实现了可靠性目标之后再予以考虑。如果你发现维持所设定的可靠性目标的成本过高，你可以明确重新定义这些目标——比如，降低一个"9"的标准——并评估由此产生的权衡。

要实现这些目标，你可能需要说服某个管理委员会、决策者群体或高层管理。你需要他们的支持，以便随着时间推移为供给和维护团队，提供必要的资源，并培训及进一步发展团队成员。这应该被视为长期投资，并明确得到相应的资金支持，而不是隐藏在其他预算项里。

反模式：忽略奥德修斯

在涉及可靠性时，一个常见的反模式是让停机或其他"坏消息"影响你的计划周期，即使这些情况是预期的。领导层在面对坏消息时，往往会感到需要"做点什么"，而"坚持计划"通常看起来没有影响力。然而，如果计划已经考虑到了停机，除非对系统的理解发生重大变化，否则"坚持计划"正是正确的做法。"奥德修斯契约 (Ulysses pact)"这个术语在这里是一个有用的例子。领导者奥德修斯告诉他的团队在他被绑在桅杆上时，坚持计划，驶过塞壬。当他的团队坚持计划（尽管他在挣扎和乞求停下），他表扬了他们。他们没有被短期思维所诱惑。他们的计划考虑了长期影响，并在混乱开始前制定了一个清晰的计划。

如果允许团队在当下做出决定，你往往会选择忽略一个好的计划，而做出情绪化或以自我为中心的选择。一个经典的例子是领导者介入故障处理中，而没有全面了解情况，尽管一个有能力的团队已经在控制局势。这通常是公司文化的结果。一种高薪人员的意见 (HiPPO) 文化可能对事件管理和整体可靠性产生极其不利的影响。相反，听从奥德修斯，坚持计划，不要弃船。这不仅适用于事件响应，还适用于错误预算耗尽或在面对"非常糟糕"的事件时跟踪 SLO。如果你的计划是在错误预算耗尽时停止功能发布，但你每次都为"这个重要功能"做例外，你的领导力将受到严重削弱。一个有效的改善措施是引入"银弹"，即领导者被授予三颗

银弹，用于在必要时覆盖预期计划。通过引入这种人为的稀缺性，领导者必须做出明确的权衡。同样，如果单一的坏事件消耗了一个 SLO，不要忽略它。召集团队分析这如何改变你们对系统的共同理解。这种类型的故障以前从未被考虑过吗？对故障的响应是否不足？

反模式：同时采用

另一个反模式是尝试在不进行修改的情况下混合旧模型和新模型。这会使团队偏离正确的方向，应避免这种情况。例如，在 ITIL 问题管理中，通常期望一个中心团队通过问题经理来减少问题的原因并缩短解决时间。相比之下，SRE 期望嵌入的工程师通过事后总结和评审来推动他们自己的问题解决。虽然结果仍然一致（更少且更短的停机时间），但方法和角色大不相同。尝试同时做这两件事，你最终会陷入混乱，并且这两种方法的预期结果相互冲突，效果不佳。

我们称这些 SRE 和非 SRE 原则的糟糕混合为“有毒组合”，类似于医学术语中指的不良药物混合。单独使用时每种原则可能是有益的，但两者结合在一起会导致意外的坏结果。我们经常发现使用两者的初衷是好的，通常是为了让现有员工参与进来，或者为了报告的连续性。然而，这种做法的吸引力远不及其带来的更糟的结果：更长的停机时间，更多的琐事和更低的可靠性。

人员配置和留存

在人员配置和角色定义上也可能出现反模式。在建立 SRE 团队时，很容易会选择从外部聘请 SRE 来对现有团队进行整顿。但这实际上可能导致精力的浪费，通常新聘请的 SRE 无法理解团队或现有技术的细微差别，回归到应用以前使用的方法，而不知道这些方法在新工作中是否合理。

我们建议将现有团队发展成 SRE 团队。仅仅重新命名是不够的，但提供一个结构化的学习路径和一个成长和发展的环境肯定是有效的。当然，有些情况下过渡可能会失败。如果个人没

有被设置在一个成功的环境中，而是被期望仅通过“阅读书籍”立即成为高级 SRE，他们可能会感到沮丧并寻找其他工作。同样，一些工程师看不到变革的理由，没有激励机制，或者非常抵制接受新角色。通过提供带薪教育、时间和学习的空间，并提供背景信息帮助团队理解变革的必要性，你可以成功地将团队过渡到 SRE 角色。这需要时间、精力和耐心。在过渡不成功的情况下，进行离职面谈是很重要的，特别是要解决过渡的问题，个人的感受和效果。你可能会发现你的计划中的缺陷，或发现它没有按你预期的方式执行。最后，当你要求团队做更复杂且影响更大的工作时，请注意这确实是更高价值的工作，团队应该为此获得相应的报酬。也就是说，当你的团队开始像 SRE 那样运作时，你应该支付他们 SRE 的薪酬，否则他们会转到能这样做的地方去。如果你提供团队学习高价值技能的机会，而他们离开去别处使用这些技能，你只能责怪自己。

技能提升

在培养和过渡现有员工成为 SRE 时，制定一个技能提升计划至关重要。这包括“需要哪些技能”和“如何获得这些技能”——即角色需要的技能以及如何使员工掌握这些技能。技能差距分析和调查等工具在这方面非常有用，用来核实对工作所需基础技能的假设。这些技能在 SRE 文献中往往没有具体提及，但它们对于 SRE 在全组织范围内扩大贡献至关重要。例如，传统运维团队对软件工程基础（如版本控制、单元测试和软件设计模式）不熟悉并不罕见。确保这些基本技能包含在你的技能提升计划中，并针对每个学习者的特点进行调整至关重要，这不仅是为了在团队中建立足够的技能基础，还为了为个人提供一个顺利过渡到新角色预期的路径（从而减少团队成员的流失）。

第五章：积极培育成功

一旦你决定 SRE 值得在你的组织中推行并决心投资于它，确保你的投资取得成功至关重要。在系统中引入变革总是困难的，但要让这种变革持续下去则更难。以下是一些关于如何在你的组织中保持 SRE 运作的建议。

着眼大处，行动小处

“如果你不能衡量它，你就不能管理它”这句话经常与 Edwards Deming 联系在一起。然而，完整的引用是“认为你不能衡量它就不能管理它，这是一个代价高昂的谬论。”SRE 的核心是一种以指标驱动的方法。然而，无论有多少 SLO 或 SLI，都无法帮助你理解你的 SRE 实践是否有效并与企业战略一致。你必须通过持续的实验和学习来发现这一点。

在前面的章节中，我们要求你“思考要大”，但在培养成功方面，你应该“行动要小”。任何形式的大规模变革都需要通过迭代和渐进的方式实现，SRE 也不例外。但有一个明显的警告——如果你的时间线太短，你将无法取得有意义的改变，所以要准备好找到平衡。

Google 内部使用目标和关键结果 (OKR) 来共享目标和对齐团队，即使在实现这些目标的方法不总是明确时。你的组织可能有自己的流程来实现这一点，但必须扩展到包括明确的迭代和定期审查 SRE 团队的各项指标（如琐事、警报、软件工程影响、容量计划等）。由于采用的非线性特性，你的进展总会有挫折，因此这也应该视为过程中的正常部分。

文化比战略更重要

Google 的一个假设，即 SRE 故事中的一个关键未书写部分，是内在的创新性 Google 文化。Google 还分享了我们进行的研究来描述这些属性。事实证明，团队成员是谁远不如团队成员如何互动、安排工作和看待他们的贡献重要。

我们了解到，有五个关键动态使成功团队在 Google 中与其他团队区分开来：

- 心理安全 : 我们是否能够在团队中冒险而不感到不安全或尴尬?
- 可靠性 : 我们是否可以指望彼此按时完成高质量的工作?
- 结构和清晰度 : 我们团队的目标、角色和执行计划是否清晰?
- 工作的意义 : 我们是否在做对每个人来说都非常重要的事情?
- 工作的影响 : 我们是否从根本上相信我们正在做的工作很重要?

我们看到的许多关于 SRE 采用的典型问题，如成本影响、特定行业关注、技术债务等。然而，这一发现的最好之处在于，像所有好的事物一样，这五个动态基本上是免费的！无论你的行业或情况如何，都可以优先考虑这些因素。Google 的高绩效团队依赖这些文化规范使 SRE 成功，使 SRE 成为这种文化基础上的自然行为。

忽视文化不会有帮助；等待也无济于事

听我们谈论文化对成功采用 SRE 至关重要，通常令人沮丧，这暗示你应该等到文化达到一定程度后才能采用 SRE。套用一句流行的谚语，最好的改变文化的时间可能是 20 年前，但第二好的时间是现在。除了可靠性问题，不让你的文化对可靠性反馈做出响应还有其他重大后果。

培养 SRE 意味着什么?

要培养和发展 SRE，需要考虑一些关键活动。

1. 亚线性扩展

我们之前提到过这一点，但需要澄清，这不是“用更少的资源做更多的事”，而是通过自动化和持续改进的文化来改变我们处理可靠性问题的方式。SRE 明确设计不是通过增加人数来扩展的，因此要抵制在现有软件流水线中增加更多人的诱惑，而是用 SRE 来自动化或省略这些步骤。

2. 建立和保持可持续的、快乐的团队

尽管科技行业已经从项目导向转向产品导向，但仍然很常见的是将个体视为可以随意在不同活动之间调动的可替换资源。这直接冲突于我们的文化建议。不要指望这样做还能在 SRE 上取得成功。

3. 承认 SRE 不是静态的—它本质上是一个动态角色，会随着时间成长

减少琐事和实施自动化的演变过程的一部分意味着 SRE 会在你的组织中发展。你仍然可以为此预算和计划，但目标是结果而不是具体任务和固定的团队规模。这一开始会感觉很奇怪，因为它与很多自上而下的计划活动相冲突。然而，当 SRE 动态地重新组建团队时，这通常是你正在取得成功的标志。

4. 评估你在组织内的可靠性思维水平和目标

达到高水平的 SRE 采用需要比预期更长的时间。在 Google 内部，我们认为达到产品战略级别的可靠性可能需要 3 到 5 年的时间。鉴于保持这一水平需要持续努力，恢复旧习惯也很常见。因此，花时间和精力不断评估和调整这种新的思维方式。

SRE 的关怀与培育

一旦启动 SRE，你需要照顾并培育你新生的组织。随着 SRE 实践的发展，你需要考虑以下几点。

将一号种子团队发展成更大的组织

不要从你最大的难题或每个人都不敢碰触的核心巨型单体应用开始。你需要在一个支持性的环境中通过一些快速的成功来起步，建立你的团队、原则和实践。相反，也不要从一个玩具服务开始。SRE 只有在有重要可靠性需求的地方才有价值。一旦你有了一号种子团队，你需要不断学习，安全地扩展。处理大量不太重要的服务可能看起来很有吸引力，但要抵制这种诱惑。SRE 的价值在于高可靠性服务。其他服务应该遵循“你构建它，你运行它”的模型。

SRE 组织结构：独立的 SRE 组织与嵌入式团队

自成立以来，Google 一直有一个专门的 SRE 组织，我们认为这样做有很多好处，例如可靠性文化、发布优先级、招聘等等。我们经常被要求将其与 DevOps 的“打破孤岛”方法进行比较。理解独立的 SRE 管理链不应该成为孤岛是至关重要的。SRE 有多种与开发团队合作的方式，从嵌入个体到轻度咨询。尽管如此，没有专门的组织结构也可能成功地部署 SRE，但需要广泛的高级领导支持。

晋升、培训和补偿

SRE 是开发人员，应该期望获得至少与组织中其他开发人员相等的补偿和激励。晋升率也是衡量是否与其他团队平等的一个重要指标。你应该定期比较薪酬和晋升率，以消除任何差距。防止任何认为这种薪酬水平允许你虐待 SRE 的假设（例如，长时间工作）。注意，SRE 对进行有意义和有影响力的工作的期望会更高。

值班是一项令人恐惧和疲惫的活动，需要仔细的准备和培训。还必须以有意义的方式补偿值班团队。如果你对直接补偿有限制，那么可以通过采用创意的方式（例如调休）来实现。

沟通和社区建设

SRE 使能涵盖了各种活动，例如正式培训课程、技术讲座、读书小组等。大部分工作是间接完成的，通过提供时间和资源进行实验（例如 20% 工作时间）。自主权和授权是建设社区的关键，这需要通过积极的领导方式来实现。这意味着设定明确的领导愿景或北极星，并在组织内显著地树立授权的榜样。在任何形式的变革中，沟通的量很容易被低估，而 SRE 也特别擅长检测不真实的信息。

评估 SRE 采用的有效性

在采用 SRE 的过程中，获得大量的 SRE 工件是很常见的，例如 SLO、SLI、错误预算、仪表板等。这些都是组织的代理指标，但它们不会总是给你一个全面的可靠性变化的图景。为此，你可能需要考虑一些更非常规的视角。如果事情确实进展顺利，良性循环会随着时间的推移显得更加平静。与其仅仅在事故与事故之间救火，不如有一种主动预防火灾的感觉。这可能会让人不安，特别是如果你的组织习惯于通过忙碌来展示其价值。在这一点上，抵制进行战术优化以重新获得忙碌感觉的诱惑。你的 SRE 会在经历失败并提升能力时自然地改进 SLO 和错误预算。

保持航向

采用更主动的方法可以让你有更多时间专注于战略愿景。你会开始更清楚地了解组织中不同服务实际需要的可靠性水平。利用这些数据并设定新的预期结果来决定优化的重点。也许你的一些内部系统被标记为业务关键，但你的 SRE 们现在知道它们只需要 99.9% 的服务水平目标（SLO）。其他系统可能现在需要更高的可靠性水平，而判断你是否成功的一个可靠方法是当你开始看到其他团队对获得 SRE 好处的兴趣。

第六章：不仅限于谷歌

为完善本报告中的观点，我们与来自不同行业的三位 SRE 领导者进行了交谈，他们在过去几年中以各种形式采用了 SRE。每个人都有关于采用 SRE 的独特故事以及他们可能会做出不同选择的见解，此外还有关于在他们的行业或组织中使 SRE 工作的洞见。

医疗保健 // JOSEPH

Joseph Bironas 自从担任 Google SRE 领导者后，一直在多个医疗保健组织中领导 SRE 的采用。因此，他能够提供一个行业层面的观点，说明在这一领域实施 SRE 如何与其他技术和初创文化不同。由于其生命攸关的工作流程的性质，可靠性通常是首要考虑的问题。然而，医疗保健行业面临着组织模式、文化、预算和监管要求方面的特定挑战。

在与一家专注于利润空间极窄的医疗设备制造及 FDA 监管领域的公司合作后，Joseph 观察到，虽然可靠性被视为一种需求，但在该行业中 SRE 的成本效益却未被充分理解。因此，SRE 和基础设施团队可能会发现自己成为“全能工程”，被纳入 IT 成本中心，职责范围大幅增加。

你可能会问，将 SRE 团队归属在 IT 成本中心下有什么问题？当企业习惯于通过广泛的 IT 框架（如 ITIL）进行管理时，很难对 SRE 做出价值判断，而 SRE 只是 ITIL 的一部分——ITIL 还处理像硬件采购等 SRE 不涉及的事情。更重要的是，管理所有公司和生产 IT 的 CIO 并不是做出软件系统可靠性判断的最佳人选。相反，归属于专注于软件的领导者（例如工程高级副总裁（SVP）或 CTO）更为合理。

该领域的组织往往面临着希望采用 SRE 而尚未采用 DevOps 实践的陡峭曲线。例如，由于涉及法规和全组织合规控制的复杂性，他们每月只发布一次软件，并且几乎没有 CI/CD 自动化。许多医疗保健组织根本不愿快速部署：对于某些客户来说，快速部署意味着测试不足或安全性不足。

实施变革（如转向 SRE）的意愿在整个行业中差异很大，可能是由于领导优先事项和风格的不同。Joseph 描述了一个团队能够派遣设计师到现场收集需求，建立新工作流程，通过更好的产品革新护理的场景。在另一个场景中，一个不同的团队只被激励比现任者做得更好，这不需要同样的投资水平。在第三个场景中，一个团队被惯性困扰，在做出任何变革或投资之前等待自上而下的命令。根据 Joseph 的经验，更进步的领导往往对客户对可靠性的需求更敏感。

与初创文化相比，这些团队的一些变革非常缓慢。一个团队质疑他们是否能在 18 个月内完成"任何事情"（如采用 SRE）——对于初创企业来说，这段时间几乎是永恒的。在考虑这种节奏的组织中的重大变革时，你必须有模型来帮助理解计划的投资回报。了解 J 曲线（见屋顶射击与登月计划（roofshots versus moonshots））在这里很重要，以避免在低谷中放弃努力，错过真正的回报。Joseph 建议每季度与团队进行检查，以保持进展的稳定节奏。他建议在专注于 SLO 之前，从事件响应开始转向 SRE，并通过事件评审（例如，持续六个月）建立持续学习的循环。为了进行"真正的投资"而不是无声地失败，你可能需要寻求高管的赞助，实施顶层 OKR，或专注于使努力在你的组织中"真实"的任何事情。关键不仅是从这个循环中学习，还要将所学付诸实践。

另一个在医疗保健行业中常见的错误是忽视 SRE 的"软件方面"，当团队习惯于专注于传统运维工作时，认为"通过软件可以大幅减少 IT 支出"这种核心价值通常是领导者所陌生的概念，甚至可能被一些根深蒂固的系统管理员和操作人员故意抵制或破坏。忽视这一方面会使 SRE 显得非常无效。软件工程也很困难且昂贵。即使你购买了商业 SRE 相关工具（尽管有多年的大量贡献者，但这些工具仍不完美），你也无法逃避集成工作，这在很大程度上是一项软件工程工作。

为可靠性制定预算也可能是一个问题。Joseph 指出，“这个行业没有 [Google 在建立 SRE 时拥有的] 广告收入曲线。” 这影响了他们像 Google 那样专业化和投资的能力，导致他们更多地依赖商业解决方案。业务预算和规划通常仍然采用瀑布模式，这对 SRE 工作来说是一个挑战——探索、理解和设计新解决方案所需的时间不适合瀑布式工作方式。

从中得到的启示是，这些问题可以适用于所有行业。Joseph 分享了一个故事，说明即使是不完美的尝试也可以是一个有价值的起点。在他曾合作的一家公司中，领导层希望有一个极其简单的错误预算版本。他们没有选择适合其关键用户旅程 (CUJ) 的 SLO，而是为所有内容设定了一个单一的 SLO（99.95% 可用）。这个目标虽然简单易懂，但却削弱了整个工程团队对 SLO 概念的信心。状态和无状态应用程序、批处理和实时应用程序都采用相同的 SLO，这最终是无用的，并削弱了对该技术的信心。这也导致了毫无意义的错误预算，因此这些预算同样被削弱，任何试图使用这些错误预算的过程也被削弱。

零售业 // KIP 和 RANDY

The Home Depot（THD）的 Commodore “Kip” Primous 和 Randall Lee 分享了这家大型零售商如何采用 SRE 的经验、成功之处以及面临的一些挑战。THD 是 Google Cloud Platform（GCP）的早期大型零售客户之一，在采用云服务的过程中，他们也遵循最近出版的 SRE 书中的原则采用了 SRE。六年后，他们当初期望构建的与现在存在的截然不同。

Kip 最初是“点商”业务部门的可靠性工程（RE）经理，负责 THD 电子商务网站“浏览堆栈”的工作。Randy 比 Kip 早加入 THD，他们分享了一个共同的 SRE 目标：通过更好的平台提高弹性。他们最初考虑建立自己的云和数据中心，但后来评估了各种云服务提供商，并最终选择了 GCP。在向云迁移的过程中，唯一成功的方法是同时改变他们的工作方式，采用类似 SRE 或“DevOps 2.0”的方法。

最初，THD 的目标是摆脱庞大的单体商业服务。Aurora 项目由副总裁资助并推动，目的是实现规模经济，减少运营团队的规模，将团队从数百名承包商转变为显著减少的全职员工。还有一个普遍的意图是提高可靠性，减少对组织内部其他可能不完全一致的团队的依赖。点商团队希望能够以"互联网速度"运作。

对齐非常重要。在迁移到云之前，每次部署都"像发射航天飞机：需要多年努力的协调"。团队觉得目前的 DevOps 模式在 THD/点商内部已经走到了尽头。通过引入 RE，团队能够在新平台和新的职责下采用新的工作模式，并有能力解决任何与可靠性相关的问题。他们雇佣了很多云原生工程师，并尽可能地实现自动化。他们能够突破限制现有 DevOps 团队的边界。

从 2015 年到 2017 年，SRE 团队能够快速独立地行动，因为他们在新的云基础设施上使用现代工具和硬件工作。然后在 2018 年，企业团队赶上了步伐——SRE 不再是唯一在 GCP 上工作的团队。令人欣慰的是，双方在企业团队更新传统模型时实现了融合——例如，承认在新的短暂虚拟机环境中不应再跟踪单个机器的补丁。通过一系列建设性的对话将团队聚集在一起，点商 RE 团队能够与新成立的集中企业团队合作，帮助建立更符合企业需求的流程和更好的安全准则。此外，他们能够将 GCP 平台的大部分管理工作（如计费、权限、配额等）从 RE 团队转移到企业团队。

在 THD 进行 SRE 之旅的过程中，Kip 和 Randy 观察到了一些模式和经验，这些经验和模式可能适用于其他行业。让其他团队采用 SRE 概念的过程花了几年时间，并且是循环进行的：推动合规自动化、成本改进、访问控制和网络安全的改进。每次互动都需要大量讨论和教育。在故障或停机很少的平静时期，紧迫感可能来自外部事件。Equifax 故障或 Akamai 或 Facebook 的 DNS 问题可能会引发新一轮的可靠性改进。

高管的支持对 SRE 在 THD 内部的成功采用至关重要。在点商团队通过使用 SRE 模型成功迁移到云后，SRE 角色在公司内变得与高绩效同义。许多其他团队希望复制这一模型，有些团队即使没有明确的 SLO 要求也被迫实施 SRE。然而，并不是所有团队都像点商团队那样幸运，可以从零开始并采用云原生。这导致一些团队难以认识到 RE 团队所带来的价值，有时

会误解角色和责任的期望。当一个团队与组织中的“非官方” RE 互动时，这种模糊性可能会导致问题，因为这些 RE 可能无法在同一水平上工作或使用与原始团队相同的原则。例如，有些人“只是按按钮”，而没有真正的自动化琐事计划。这样的经历会让团队对未来与其他 RE 团队的合作失去兴趣。

Kip 还警告说，如果每隔几年没有新的 SRE 启发的努力，可靠性标准会退化。RE 团队打破了壁垒，但这些壁垒正在重新建立。团队认为，“可靠性不是我的问题，这是 RE 的问题！”这传递了错误的信号。Randy 补充说，一个运行良好的 RE 团队如果没有不断强化和教育 RE 的实践和原则，以及明确的角色和责任定义，就会倒退。

目前，THD 正在“加倍”投入 RE，但如果变革没有坚持 SRE 的原则，这实际上可能是一种反模式。SRE 不是解决所有问题的万能药，但当一个团队看到 SRE 的成功时，很难不想将 SRE 应用于各个方面。最近，Kip 被要求在分销中心和供应商支持的物理硬件上运行 RE，这并不适合 SRE。虽然总是有机会提高这些系统的可靠性，但在非云原生环境中应用许多 RE 实践更具挑战性。对于某些业务领域，可能更好的前进路径不是 SRE，而是价值流图（Value Stream Mapping）或精益（Lean）等实践。为了避免这些问题，更有意义的是将 SRE 作为“拉动”模型，而不是“推送”模型：不要强迫团队使用 SRE，而是将其作为一种服务提供，让团队自主选择。

Kip 和 Randy 的最大建议是专注于高管教育，并认识到拥护者的价值。如果没有自上而下的支持，很难为任何有意义的变革筹集资金。通过产品开发团队获得资金会导致这些团队“从不想支付这笔税”的现象。每当他们支付这笔税时，只希望 SRE 直接为他们的产品工作并朝着他们的产品目标努力。

在 THD，最初有一位高级领导倡导创建和发展点商 RE 团队以及其他领域的 RE 团队。THD 现在处于一个尴尬的境地，有许多 RE 团队在不同的项目上工作，应用 SRE 原则的能力各不相同。Randy 和 Kip 认为，拥有一个更高级的领导者可能会改善 THD 的状况。一个负责所

有 RE 角色的可靠性副总裁（VP of Reliability）可以提供规模经济。没有中央 RE 组织，SRE 角色可能会演变成不同组织中的 SRE 完全做着不同的事情，并遵循不同的标准和原则。

结论

我们希望这份报告能够为企业如何采用 SRE 以及可能面临的挑战提供一些见解。我们认为，如果您清晰地定义 SRE 原则，将这些原则映射到具体的实践和能力，并优先培养团队内的成长，成功的机会将更高。我们还展示了一些团队在企业内部启动 SRE 实践的过程中所经历的例子，以及他们所面临和克服的具体挑战。

我们相信这份报告将有助于您采用 SRE，带来更可靠的技术体验。希望通过这种形式的采用，运营团队可以变得更可持续，服务更具扩展性，开发速度得以提升。

“愿一切进展顺利，告警短信永不响起。”

关于作者

James Brookbank 是 Google 的一名云解决方案架构师。解决方案架构师通过解决复杂的技术问题并提供专业的架构指导，帮助 Google 的客户更轻松地使用云服务。在加入 Google 之前，James 曾在多家大型企业工作，专注于 IT 基础设施和金融服务。

Steve McGhee 是一位可靠性倡导者，帮助团队了解如何最佳地构建和运营世界级的可靠服务。在此之前，他在 Google 担任了超过 10 年的 SRE，学习如何在搜索、YouTube、Android 和云中扩展全球系统。他曾在加利福尼亚、日本和英国管理多个工程团队。Steve 还曾在一家加利福尼亚的企业工作，帮助他们向云迁移。

关于译者

中国DevOps社区核心组织者，前Elastic资深开发者布道师，《DevOps Handbook》《The Site Reliability Workbook》译者；精通DevOps/SRE/ITSM等理论体系和相关实践等落地实现。致力于在全国范围内通过社区来推广DevOps的理念、技术和实践。推动开源技术堆栈的应用，包括运维大数据分析平台、云原生服务治理、APM全链路监控和AIOps等使用场景。

博客：https://martinliu.cn

反馈： liuzh66@gmail.com

微信公众号：