CLOUD CRE 生产成熟度评估指南

对生产环境 SRE 实践的评估指南

云客户可靠性工程师和客户共建 SRE

译者：刘征

【译者注】Google Cloud 的客户可靠性工程团队（Cloud CRE）是一支专业团队，致力于帮助客户提高其 Google Cloud 平台上的可靠性。Cloud CRE 团队的使命是帮助客户建立可靠的服务，以便他们可以专注于创新。为了实现这一目标，Cloud CRE 团队开发了一种生产成熟度评估方法，以帮助客户评估其生产环境的成熟度，并提供指导，以帮助他们提高其生产环境的可靠性。

Cloud CRE Production Maturity Assessment

From：*https://sre.google/resources/*

封面图片 *Photo by Loan Patru: https://www.pexels.com/photo/paon-27000643/*

目录

第一部分：监控和指标 - Monitoring & Metrics

确定期望的服务行为，度量服务的实际表现，并纠正差异。示例指标包括响应延迟、错误率或未答复查询率、资源的峰值利用率。

SLO 定义和度量 - SLO Definition and Measurement

拥有一个代表用户需求的，已发布的服务水平目标 (SLO)，并根据这些已发布的 SLO 进行评估、报表和总结。

- ✓ 您的服务/应用程序是否有明确定义的 SLO？ SLO 是对服务行为的度量目标，例如 95% 的查询请求延迟低于 500 毫秒。
- ✓ SLO 是否反映了客户的体验？即，满足 SLO 是否意味着客户体验是可以接受的？反之，未能满足 SLO 是否意味着客户体验是不可接受的？
- ✓ SLO 是否发布给用户？ 这并不常见，但有时如果您的服务有一个主要的大客户，您可能会发布您的 SLO，以便他们可以根据您的服务性能进行计划。
- ✓ 您是否有对 SLO 进行良好定义的度量？
  - ✦ 度量过程是否有文档记录？
  - ✦ 度量过程是否自动化？ 即，您是否能自动获得 SLO 的报告或图表，还是必须手动运行命令或填写电子表格，以查看您对 SLO 的满足情况？
- ✓ 您是否有修订/优化 SLO 的流程？例如，如果您一直未能满足 SLO，但用户似乎没有投诉，您如何决定是否修订 SLO 目标水平？

成熟度等级

1 - 未管理	服务没有 SLO 或 SLO 不代表用户需求。
2 - 已管理	服务有定义的 SLO，但 SLO 的度量是临时的或没有文档记录。
3 - 已定义	服务有定义的 SLO，并且度量过程有文档记录且有明确的负责人。
4 - 已度量	SLO 发布给用户，并且 SLO 自动度量，代表用户需求。
5 - 持续改进	与用户一起对 SLO 进行持续评估和改进。

仪表板与可视化 - Dashboards & Visualization

用清晰的数据展示来支持服务的管理、决策和行动事项调整。

- ✓ 您是否采集相关服务的数据?
  - ✦ 您使用了哪些采集方法?
    - 日志处理
    - 白盒监控：检查内部系统状态
    - 黑盒监控：模拟实际用户请求的人工流量事务/拨测
    - 从客户端设备采集指标
    - 其他： 您能简要描述您的采集方法吗?
  - ✦ 是否为自动化采集?

- ✓ 您管理的服务是否都有仪表板？仪表板通常是一个显示图表或其他关键监控信息的网页，使您可以一目了然地看到服务的性能状态。
- ✓ 仪表板是否包含关键服务指标？例如：QPS，延迟，容量，错误预算消耗情况
- ✓ 仪表板是否包含关键业务指标？例如：访问量，用户，浏览量，共享次数
- ✓ 仪表板是否可集中式的分享给相关业务部门使用？
- ✓ 仪表板在应用程序/服务/团队之间是否以统一的 UI / UX 共享？
- ✓ 企业是否有用于临时数据探索的工具？例如：可下钻分析式的仪表板，Splunk / BigQuery，数据仓库，以满足定制化分析需求。

成熟度等级

1 - 未管理	没有仪表板。数据采集是临时的、不一致的，没有文档记录。
2 - 已管理	可能有关键指标的仪表板，但仅以静态形式存在（不可定制）。仪表板没有集中管理，用途没有标准化，所有权不明确。
3 - 已定义	仪表板存在并支持常见的技术用例，所有权明确。支持临时查询的工具存在，但使用起来有些复杂（需要培训或长时间的试验才能得到结果）。
4 - 已度量	仪表板支持技术和业务用例。临时数据探索工具是可定制的，支持常见用例，无需培训（例如，按照手册文档操作）或通过直观的界面使用。
5 - 持续改进	仪表板在各业务单元之间标准化。

用户关注 - User Focus

采集准确反映用户体验的数据，并使用这些数据来维护服务质量。可区分人工流量指标和关键用户旅程的度量；例如，如果服务器的“正常运行”，但用户仍然无法使用产品，则该指标无法提供用户体验的认知。

- ✓ 服务是否采集任何未从服务器导出的数据？通常是探针或关键用户旅程探测脚本。
- ✓ 探针覆盖是否涵盖复杂的用户旅程？ 例如，对于电商业务：主页 => 搜索产品 => 产品列表页 => 产品详情页 => “加入购物车” => 结账
- ✓ 是否为特定的用户旅程/流程提供 SLO?
- ✓ 是否有针对用户旅程度量性能回归的既定响应流程？例如，每当用户完成帐户创建流程的监控度量时间超过 5 分钟时，既定响应是：先暂停功能发布工作，并优先进行次问题的补救工作。
  - ✦ 如有，响应是什么?
    - 回滚发布
    - 触发告警
    - 开发临时修复补丁
    - 其他：您能简要描述您的响应方法吗?
- ✓ 是否评估用户旅程 SLO 的状态以推动服务改进?

成熟度等级

1 - 未管理	没有采集用户体验数据，或仅有服务器端指标数据（如服务响应延迟）。

2 - 已管理	存在探针来测试特定用户端点（如站点主页、登录），但缺少对复杂用户旅程/流程的探测覆盖。探针通常是单步的，而非多步旅程。
3 - 已定义	复杂用户旅程得到度量（通过复杂探测或客户端实时流量报告）。服务改进是被动的，而非主动的。
4 - 已度量	SLO 明确涵盖特定的用户旅程。不只是“服务器是否正常运行”，而是产品是否正常工作，能够让用户执行特定的产品功能。用户旅程的 SLO 违反会导致发布回滚。
5 - 持续改进	持续评估服务延迟和可用性，并用于推动服务改进。

第二部分：容量规划 - Capacity Planning

预测未来需求，并确保服务在适当资源水位工作，并有足够的容量来满足这些需求。

业务指标预测 - Business Metric Forecasting

预测服务关键业务指标的增长。业务指标的示例包括用户数量、销售数据、产品采用率等。预测需要准确且长期，以便有意义地指导容量规划。

- ✓ 您的应用程序是否有关键业务指标？例如，用户数量、图片数量、交易数量等。
- ✓ 您是否有关键业务指标的历史趋势数据?
  - ✦ 保留时间是多少个月?
- ✓ 您是否预测关键业务指标的未来增长?
  - ✦ 您预测的未来时间范围是多少个月?
- ✓ 您是否通过将预测与度量期间的实际值进行比较分析，来衡量预测准确性?
  - ✦ 6 个月预测的数字与观测值的误差百分比是多少？（0-100+）
- ✓ 您的预测是否考虑了可能导致业务指标快速变化的发布或其他非有机事件?

成熟度等级

1 - 未管理	没有定义容量单位，或定义了单位但没有历史趋势数据。
2 - 已管理	定义了业务单位，有单位的历史趋势数据，但预测能力有限。预测要么不准确，要么是短期的（<=1 个季度）。
3 - 已定义	有可靠的历史趋势数据，有 4-6 个季度的预测。有一个复杂的预测模型。
4 - 已度量	已衡量过去预测的准确性。使用 6-8 个季度的预测。
5 - 持续改进	定义并度量且准确的预测，知道如何准确预测并不断提高准确性记录；实际需求的准确性在 6 个月预测内误差在 5% 以内。

供给指标建模 - Supply Metric Modeling

通过经验转换模型计算一组业务指标的容量需求。一个基本的示例模型：服务 1000 名用户需要 1 台虚拟机。预测显示用户每季度增长 10%，因此虚拟机也必须每季度增长 10%。

- ✓ 您是否有一个将业务指标转换为逻辑容量指标（虚拟机、pods、分片等）的模型？一个简单的模型示例可能是："每增长 1000 名新用户我们需要增加一个 VM-large-1。"
  - ✦ 该模型是否有文档记录？
- ✓ 您是否验证容量模型的各个方面？对于上述示例模型：我们怎么知道 1000 名用户放在一台 VM-large-1 上是合适的？我们有压力测试吗？
  - ✦ 验证是否是自动化的？
- ✓ 该模型是否考虑了资源规格或成本的变化？例如，不同地点的 CPU 成本效率或可用虚拟机规格的差异。

✓ 实际需求的持续度量是否反馈到容量模型的修订中?

成熟度等级

1 - 未管理	没有将业务指标转化为逻辑供给的模型。服务是资源过度配置的（购买并希望使用模型），没有花费大量时间进行容量规划。
2 - 已管理	使用简化的经验规则/指导方案，例如每 1000 用户/虚拟机。
3 - 已定义	有压力测试，知道可能的扩容时间点。例如，压力测试的数据表明，1200 名用户确实适合在一个虚拟机上。使用经过验证的经验法则。模型验证可能是临时的（未自动化）。
4 - 已度量	有一个记录在案的服务模型，并定期验证（例如，通过了在发布验证期间的压力测试）。模型简单，维度上存在不足（例如，涵盖 CPU，但不包括带宽）。模型没有考虑不同地点的资源成本/规格差异，例如 CPU 类型、资源成本、虚拟机规格等。
5 - 持续改进	有一个定义良好的模型，将需求单位转化为供给单位组合，并使用实时系统的反馈。制定了改进关键维度的计划。理解并在模型中考虑CPU平台/区域/网络之间的差异。如果存在非线性，则理解非线性。

容量获取 - Acquiring Capacity

为服务配置额外资源：知道何时何地需要什么资源，了解必须满足的约束条件，并有满足这些需求的流程。

✓ 团队每月在资源管理上花费多少总时间?

✓ 获取更多资源的流程是否有明确定义?

- ✦ 流程是否有文档记录?
- ✦ 流程是否自动化?

✓ 资源请求的发起是否由预测程序自动生成?

✓ 您是否知道哪些约束条件决定了服务的位置？例如，您的服务器是否必须位于特定的云区域（az）或地区（region）？如果是，为什么？您的服务、处理流程和数据存储是否必须共址或在同一区域（az）或地区（region），为什么？您是否有数据区域化要求?

- ✦ 这些约束条件是否有文档记录?

✓ 您是否监控对这些服务约束的遵守情况?

成熟度等级

1 - 未管理	资源是手动且临时部署的。系统组件之间的关系没有文档记录。不确定部署的资源是否足够。
2 - 已管理	存在服务约束的经验法则（例如数据库和 Web 服务器需靠近），但这些约束没有被度量（未验证）。有定义的资源获取流程，但流程是手动的。
3 - 已定义	获取资源的过程大部分是自动化的，但何时获取资源仍是手动过程。
4 - 已度量	系统约束被建模，但缺乏评估。随着时间推移，服务变动可能导致模型中的约束与现实脱节。何时获取资源由预测驱动。
5 - 持续改进	模型不断评估和维护，即使在服务变动的情况下也是如此。获取容量的位置由模型驱动，获取容量的时间由预测驱动。

容量利用率 - Capacity Utilization

监控服务资源的使用情况，理解容量和利用率之间紧密而复杂的关系，设定有意义的利用率目标，并实现这些目标。

- ✓ 您是否为服务定义了任何利用率指标？例如，现在有多少总资源在使用？
- ✓ 您是否持续度量利用率？
- ✓ 您是否有公认的利用率目标？即，您是否有特定的利用率指标阈值，这些阈值被记录，并认可为所需的最低值，未达到该目标会触发响应处理流程？
- ✓ 利用率目标是否在驱动服务改进？
- ✓ 您是否定期审查利用率目标，及服务对该目标的符合情况？例如，每季度审查一次。
- ✓ 您是否了解服务的资源瓶颈是什么？例如（每台虚拟机支持 1000 用户）：如果我们将 1200 用户放在一台虚拟机上，哪个资源会首先耗尽？（CPU、RAM、线程、磁盘等）
  - ✦ 您能简要描述一下吗？
- ✓ 您是否测试新版本的利用率回归情况？
- ✓ 显著的利用率回归是否会阻止发布？

成熟度等级

1 - 未管理	没有定义系统利用率指标。
2 - 已管理	定义了利用率指标和目标，但度量是临时的（非持续）。每季度/每年进行。
3 - 已定义	持续度量利用率，并定期审查（每月/每季度）。

4 - 已度量	利用率是发布健康状况的指标（利用率显著回归的新版本会被阻止发布）。
5 - 持续改进	持续评估利用率指标，并用于推动服务改进（例如，保持服务成本的可持续性）。

第三部分：变更管理 - Change Management

在保持所需的服务运行状态的同时变更服务的行为。示例：金丝雀发布、1% 实验、滚动升级、快速失败回滚、季度错误预算。

发布过程 - Release Process

标志、数据和二进制程序文件的变更流程。

术语澄清

不同上下文中的发布术语可能有所不同，因此在本次评估中我们将使用以下定义。

- ✓ **服务版本**：一组代码、二进制文件和/或配置，封装一个定义明确的服务状态。
- ✓ **候选版本**：一个用于部署到生产环境的新服务版本。
- ✓ **发布**：准备和部署一个新的发布候选版本到生产环境。
- ✓ **候选版本准备**：创建一个可行的候选发布版本。这包括编译和所有测试、分级或在与生产环境接触之前必须进行的其他验证活动。
- ✓ **部署**：转换到新服务版本的过程。例如，将二进制文件推送到其生产位置，并重新启动相关进程。
- ✓ **增量部署**：以一系列连续逻辑划分的方式结构化地部署到多个生产位置。例如，一次部署到多个不同站点中的一个站点，或者在总共 1000 台虚拟机的环境中，每次在其中的 100 台虚拟机子集进行部署。

- ✓ **金丝雀发布**：战略性地部署到生产环境的有限子集，以测试版本的可行性，必须满足明确的健康检查才能继续更广泛的部署。

- ✓ 您是否有定期的发布节奏？即，您是否有固定频率执行发布流程？
  - ✦ 以下哪项最接近您的发布节奏？
    - 构建成功即发布（Push-on-green）
    - 每日
    - 每周
    - 每两周
    - 每月
- ✓ 一次典型发布需要多少工程师工时（小时）？
- ✓ 从发布候版本选准备到部署完监控配置结束，总共需要多少人力时间？
- ✓ 发布的成功率是多少？成功的发布通常是指达到完全部署，且不需要回滚或产品补丁的发布。
- ✓ 发布成功率是否满足您的业务需求？例如，发布是否足够可靠，不会阻碍开发进度，或削弱对功能交付的信心？
- ✓ 您是否有回滚部署的流程？
- ✓ 您是不是进行的增量部署？例如，将新版本先暴露给 1% 的用户访问，然后 10%，然后 50%...（参见上面的定义。）
- ✓ 对版本发布结果，您是否有明确的失败条件检查？例如，由人类或机器人评估的指标，以查看发布结果是否存在问题。

- ✓ 您是否使用金丝雀发布（或一系列金丝雀发布）来验证每次部署？即，少量真实请求被发送到新版部署，以查看其是否能正确处理，然后再接收所有请求。（参见上面的定义。）
- ✓ 发布过程的所有部分是否都是自动化的？
  - ✦ 哪些部分？
    - 发布候选版本准备
    - 部署
    - 失败检测
    - 部署回滚
    - 金丝雀发布
- ✓ 发布过程的所有部分是否都有文档记录？即，如果您让一个新团队成员执行发布，他们是否能找到，并遵循该过程的说明文档？
  - ✦ 哪些部分？
    - 发布候选版本准备
    - 部署
    - 失败检测
    - 部署回滚
    - 金丝雀发布

成熟度等级

1 - 未管理	发布没有固定的节奏。发布过程是手动且无文档记录的。
2 - 已管理	发布过程有文档记录。尝试定期发布。有应对不良发布的流程。
3 - 已定义	拥有自动化的持续集成/持续交付 (CI/CD) 流水线。发布是可预测的，成功率满足业务需求。发布过程设计允许在预先与业务约定的参数范围内，尽量降低对客户的不良影响。
4 - 已度量	完全自动化的发布过程，包括自动化测试、金丝雀发布过程和自动回滚。发布满足业务需求。展示出可在不显著消耗错误预算的情况下，进行回滚的能力。
5 - 持续改进	标准的质量保证（Q/A）和金丝雀发布过程。发布快速、可预测，且需要最少的人工监督。过程完全按标准操作落地，并可发布完成。自动化的发布验证和测试，全面的与监控集成。健全的回滚和异常处理程序。发布和发布频率符合业务需求。发布过程符合产品需求。

设计与发布 - Design & Launch

通过早期参与、设计审查、引入最佳实践等方式，设计一个成功的服务。

- ✓ 团队是否有重大代码变更的评审流程？例如，设计评审或发布评审。
  - ✦ 评审是否自助进行？
  - ✦ 评审是否有特定的批准人？
- ✓ 团队是否有执行重大变更的最佳实践？例如，如果您知道您正在发布一个新功能，会添加一个新组件，是否有关于如何以标准方式配置、监控、部署和操作该组件的协议？
  - ✦ 这些实践是否有文档记录？
- ✓ 最佳实践是否是通过代码和/或自动化方式强制执行的？

- ✓ 是否可以因未遵循最佳实践而拦截发布?
- ✓ 团队是否有高风险发布的强制性流程?

成熟度等级

1 - 未管理	团队没有新代码或新服务的设计或评审流程。
2 - 已管理	团队有一个轻量级的引入新组件或服务的评审流程。团队有一套最佳实践，但它们是临时的且记录不完整。
3 - 已定义	团队有高风险发布的强制性流程。该流程涉及应用发布的记录最佳实践，可能包括设计/发布审查。
4 - 已度量	大多数发布的评审是自助进行的（例如，通过简短的调查，可以触发对边缘情况的审查）。最佳实践通过共享代码模块和调优自动化来应用。团队有记录的重大服务变更流程，这些设计有指定的评审者/批准人。
5 - 持续改进	最佳实践通过自动化强制执行。允许通过特例申请，来请求忽略最佳实践。包含1-4中的所有内容。

变更流程自动化 - Change Process Automation

自动化了与服务运维相关的手动工作。

- ✓ 您是否有变更流程自动化？即，当需要执行生产操作（如重启一组作业或变更一组虚拟机的配置）时，是否有脚本或其他工具或服务来帮助更安全和轻松地进行变更?
- ✓ 您的自动化是否使用了“基础设施即代码 (Infrastructure as code)”？与命令执行（命令式）自动化流程相对的。

- ✓ 您的自动化是否支持"差异"（diffing）或"干运行 (dry-run)"功能，以便操作员可以看到操作的效果?
- ✓ 您的自动化是否幂等?
- ✓ 您的自动化是否具有弹性？即，很少因服务、政策或基础设施变化而中断。
- ✓ 您的常规流程是否有文档记录？例如，如果让新团队成员升级所有虚拟机上的 Linux 版本，他们是否能找到并遵循操作说明?
- ✓ 您的常规任务中有多少百分比是自动化的?
- ✓ 您是否有启动或关闭服务的流程？例如，在新的公有云（ GCP）区域部署。
  - ✦ 该流程是否有文档记录?
  - ✦ 该流程是否自动化?
  - ✦ 需要多少工程师工时（小时）？

成熟度等级

1 - 未管理	团队花费大量时间在操作上，或有一个"运维团队"花费大量时间在操作上。任务记录不足（例如，流程仅特定个人知道，但没有文档记录）。
2 - 已管理	手动任务有文档记录。高负担任务部分为自动化，并且了解操作时间的花费。
3 - 已定义	许多高负担任务已自动化。团队在操作上花费的时间不到一半。自动化不灵活，易受政策或基础设施变化的影响。
4 - 已度量	所有高负担任务均已自动化。团队在操作工作上花费的时间少于 20%。
5 - 持续改进	所有可自动化的任务均已自动化，团队专注于更高层次的自动化和可扩展性。现有自动化可以快速适应服务或政策变化。

第四部分：紧急响应 - Emergency Response

注意并有效响应服务故障以保持服务符合SLO。示例：值班轮换、主要/次要/升级、操作手册、不幸之轮、告警审查。

组织值班轮换 - Organize Oncall Rotation

响应告警 - Responding to Alerts

- ✓ 您是否有应对重大事件的事件管理协议（流程）？例如，指定事件经理、启动跨公司通信/沟通、定义其他管理危机的角色。
- ✓ 您的服务是否有特定情况的操作手册？操作手册是描述如何响应特定类型故障的文档，例如将服务从一个区域切换到另一个区域。
    - ✦ 大多数操作手册条目是否提供明确且有效的响应操作？即，能够减轻问题并帮助诊断或解决原因的操作。
- ✓ 您的服务是否有例行遵循的升级流程？即，如果无法解决问题，是否有流程可以找到并联系能够解决问题的人？
- ✓ 升级流程是否有文档记录？
- ✓ 一线响应者能否在不求助的情况下解决90%的事件？例如，事件在不升级到主题专家或同事的情况下解决？

值班轮换结构 - Oncall Rotation Structure

关于值班责任的程序和期望，明确识别受过训练的工程师团队，能够可持续地快速响应事件以维护服务的SLO。

- ✓ 您的服务是否有值班/通知触达的轮换?
- ✓ 您的值班轮换是否有文档记录的响应时间？响应时间是“到键盘时间”而不是“解决时间”。
  - ✦ 响应时间是多少分钟?
- ✓ 您的值班团队成员是否在轮班时进行交接?
- ✓ 您的团队是否有将新响应者加入值班轮换的流程？例如，培训过程、指导、跟班主要响应者。
  - ✦ 培训材料是否有文档记录?
  - ✦ 您的团队是否有“值班配对”计划，即由有经验的响应者与新响应者配对进行在岗培训?
- ✓ 您的团队是否进行“厄运之轮”或其他值班培训演习？**厄运之轮**：生产灾难角色扮演，用于培训和审查及维护文档。

成熟度等级

1 - 未管理	临时支持：尽力而为，仅限白天，无实际轮换，手动告警，无定义的升级流程。
2 - 已管理	有文档记录的轮换和响应时间。告警自动化并与监控集成。培训是临时的。

3 - 已定义	存在培训流程制度（幸运之轮、值班跟班在岗培训等）。轮换人员充足，但有些告警需要升级到高级团队成员才能解决。
4 - 已度量	度量事件的平均修复时间 (MTTR) 和平均检测时间 (MTTD)。轮班之间有交接。大多数事件需要最小的升级（可以由值班人员单独处理）。
5 - 持续改进	每周回顾审查事件并改进策略、交接、班次间的沟通，大多数问题无需升级即可解决，回顾审查轮换的价值和规模，评估范围。建立了事件响应协议。

告警分析 - Alert Analysis

回顾审查实际接收到的告警，覆盖现有系统，管理告警的实践和流程，可操作告警的数量，按原因、位置和条件对事件进行分类的能力。

- ✓ 您的告警是否基于监控数据自动化?
- ✓ 您的团队收到的告警量是否一致？如果答案为"是"表示大多数值班轮班的告警数量相当。
- ✓ 您的团队是否有维持告警增长低于线性增长的流程？即，随着服务数量的增长，是否采取措施来防止告警量成比例的增加?
- ✓ 您的团队是否有自动化的告警抑制规则或依赖关系，以减少告警数量？例如，当负载均衡服务器被排空时，该服务器的告警会自动被抑制；或者，细粒度的告警（如特定延迟阈值）可能依赖于粗粒度的"无法访问"告警的静音状态。
- ✓ 您的团队是否经常忽略或手动抑制任何告警，因为它们噪音大、垃圾多、假告警或不可/无需操作？例如，是否有某个告警每几天都会触发一次，但通常无需采取任何行动就关闭了？短期内有垃圾告警是可以的，只要在持续修复它们。
- ✓ 您的团队是否在值班轮班之外花费大量时间处理轮班期间发生的事件?

- ✓ 您的团队是否有定义的超负荷处理流程，以应对告警触发率达到不可持续水平的时期?
- ✓ 您的服务是否经历过未被告警初期检测到的重大故障？即，没有触发任何告警的故障，或只有在通过其他来源发现事件后才触发的告警。
- ✓ 您的团队是否收集告警统计数据（原因、采取的行动、告警解决方案）以推动改进?
- ✓ 您是否有定期审查评估会议，审查评估告警或事件以进行模式匹配、发生率、操作手册改进等?

成熟度等级

1 - 未管理	Pager 过载，告警被忽略，告警长时间静音，无操作手册，告警量增加或不可预测，事件发生时无告警（手动检测事件）。
2 - 已管理	事件率不可持续（值班成员报告过劳）。虚假告警被静音。有操作手册，但指导操作较少。许多告警是"信息性"的，无明确操作。告警量不可预测。
3 - 已定义	大多数告警有明确人类操作的操作手册。告警量在轮班/周之间无显著变化。告警量在可持续水平（由值班团队测量），有处理告警量过载的流程（如开发人员停止开发并致力于可靠性）。
4 - 已度量	大多数告警有有用的操作手册条目。几乎所有告警都需深思熟虑的人工反应。定期分析并处理告警主要原因。积极使用告警抑制以消除重复告警和已提醒其他团队的告警。维持服务增长下的低于线性增长的告警量。
5 - 持续改进	识别告警模式，定期审查故障率，修剪告警，审查基本服务故障模式，所有告警需深思熟虑的人工干预，告警操作手册提供适当的调试入口。

事后复盘分析 - Postmortems

撰写事后复盘分析的制度，格式和行动项目，以及后续行动的期望。通过根本原因分析和发现结果推动服务可靠性改进的实践。

- ✓ 您的团队是否有事后复盘分析流程？事后复盘分析也称为事件回顾、回顾。
- ✓ 事后复盘分析流程是否仅针对大型/重大事件？
- ✓ 您的事后复盘分析流程是否会为团队生成行动项目？
  - ✦ 您的团队是否有优先处理并完成事后复盘分析行动项目的流程？
- ✓ 大多数事件是否进行了彻底的根本原因分析，并有明确的结果？
- ✓ 您的服务是否有优先修复或减轻已识别根本原因的流程？
- ✓ 以下哪些是您的事后复盘分析流程的一部分？
  - ✦ 检测时间
  - ✦ 从事件发生到检测到的时间。
  - ✦ 修复时间
  - ✦ 从事件发生到解决的时间。
  - ✦ 以上均无
- ✓ 您的公司是否有在组织/团队之间共享事后复盘分析的流程？
- ✓ 您的公司是否收集事后复盘分析元数据（如根本原因分类，MTTR，MTTD）？
  - ✦ MTTD：平均检测时间。事件从发生到检测的平均时间。
  - ✦ MTTR：平均修复时间。事件从发生到解决的平均时间。

- 您的公司是否有使用这些数据识别问题区域的流程？例如，是否对数据进行分类和/或汇总以识别故障模式或高风险服务方面，以指导风险缓解投资？

✓ 您的团队或公司是否有事后复盘分析流程的所有权或审核周期？即，是否有人标准化格式并定期评估其有效性和价值？

成熟度等级

1 - 未管理	无跟进或系统错误识别，无根本原因分析，事件得到控制但未分析，同类事件不断发生，未识别长期趋势。
2 - 已管理	有定义的事后复盘分析流程和行动项目，但行动项目跟进差（仅处理 P0 项目）。根本原因分析（RCA）不足（“为什么”问得不够多）。由于行动项目跟进不力或 RCA 不充分，类似事件重现。
3 - 已定义	事后复盘分析流程应用于所有重大事件（包含行动项目）。优先处理 P0 级别的行动项目。RCA 广泛且正确归因于大多数事件的根本原因。
4 - 已度量	事后复盘分析附有注释元数据以促进分析。事后复盘分析结果在受影响团队之间广泛分享，从错误中学习。
5 - 持续改进	所有行动项目及时完成，由相关团队和其他团队审查以便学习，识别问题区域，有流程确保行动项目完成，标准化格式，审查事后复盘分析流程以评估其价值。

关于译者

刘征：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

推荐阅读

Google SRE 白皮书：

1. 《事故管理剖析》
2. 《SRE 容量管理最佳实践》
3. 《产品导向的 SRE 可靠性》
4. 《企业 SRE 路线图》