



韧性生命周期框架

# AWS 规范性指导



# AWS 规范性指导: 韧性生命周期框架

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

# Table of Contents

简介 .....	1
术语和定义 .....	2
持续韧性 .....	2
第一阶段：设定目标 .....	3
映射关键应用程序 .....	3
映射用户故事 .....	4
定义衡量指标 .....	4
创建其他衡量指标 .....	5
第二阶段：设计与实施 .....	6
AWS Well-Architected 框架 .....	6
了解依赖关系 .....	6
灾难恢复策略 .....	7
定义 CI/CD 策略 .....	7
开展 ORR .....	8
了解 AWS 故障隔离边界 .....	8
选择响应 .....	8
韧性模拟 .....	9
安全失效 .....	9
第三阶段：评估与测试 .....	10
部署前活动 .....	10
环境设计 .....	10
集成测试 .....	10
自动化部署管线 .....	11
负载测试 .....	11
部署后活动 .....	11
进行韧性评估 .....	11
DR 测试 .....	12
偏差检测 .....	12
综合测试 .....	12
混沌工程 .....	12
第四阶段：运营 .....	14
可观测性 .....	14
事件管理 .....	14
持续韧性 .....	15

第五阶段：响应与学习 .....	16
创建事件分析报告 .....	16
进行运营回顾 .....	17
检查告警性能 .....	17
告警精度 .....	17
误报 .....	17
漏报 .....	18
重复提醒 .....	18
进行指标检查 .....	18
提供培训与赋能 .....	18
创建事件知识库 .....	18
深入实施韧性 .....	19
结论和资源 .....	20
贡献者 .....	21
文档历史记录 .....	22
术语表 .....	23
# .....	23
A .....	23
B .....	26
C .....	27
D .....	30
E .....	33
F .....	35
G .....	36
H .....	37
我 .....	38
L .....	40
M .....	41
O .....	45
P .....	47
Q .....	49
R .....	50
S .....	52
T .....	55
U .....	56
V .....	57

---

W .....	57
Z .....	58
.....	lix

# 韧性生命周期框架：持续提升韧性的方法

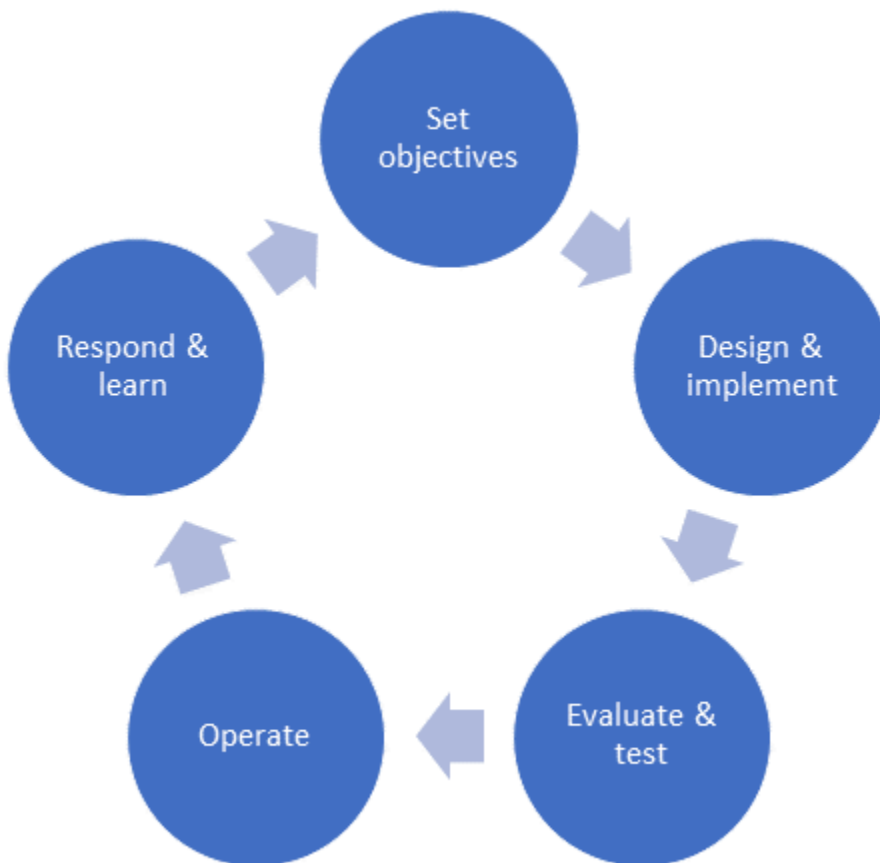
Amazon Web Services ( [贡献者](#) )

2023 年 10 月 ( [文档历史记录](#) )

如今的现代组织正面临着越来越多与韧性相关的挑战，尤其是在客户期望转向始终在线、始终可用的思维模式之际。远程团队与复杂的分布式应用程序相结合，再加上对频繁发布的需求不断增加。因此，组织及其应用程序需要比以往任何时候都更具韧性。

AWS 将弹性定义为应用程序抵御中断或从中恢复的能力，包括与基础架构、依赖服务、配置错误和临时网络问题相关的中断。（参见 [Well-Architected AWS d Framework 可靠性支柱文档中的弹性和可靠性组成部分](#)。）但是，为了达到所需的韧性水平，通常需要权衡取舍。需要评估运营复杂性、工程复杂性和成本并进行相应的调整。

基于与客户和内部团队多年的合作，开发 AWS 了一个可捕捉弹性学习和最佳实践的弹性生命周期框架。此框架概述了五个关键阶段，如下图所示。在每个阶段，您都可以使用策略、服务和机制来提升韧性状况。



本指南后文部分将介绍以下阶段：

- [第一阶段：设定目标](#)
- [第二阶段：设计与实施](#)
- [第三阶段：评估与测试](#)
- [第四阶段：运营](#)
- [第五阶段：响应与学习](#)

## 术语和定义

每个阶段的韧性概念应用于不同的层面，从单个组件到整个系统。实施这些概念需要明确定义几个术语：

- 组件是执行功能的元素，由软件和技术资源构成。组件的示例包括代码配置、网络等基础设施，甚至服务器、数据存储以及多重身份验证 ( MFA ) 设备等外部依赖关系。
- 应用程序是可提供商业价值的组件集合，如面向客户的网络店面或改进机器学习模型的后端流程。应用程序可能由单个 AWS 账户中的一部分组件组成，也可能是跨多个 AWS 账户 和区域的多个组件的集合。
- 系统是管理给定业务功能所需的应用程序、人员和流程的集合。它包括运行功能所需的应用程序；持续集成和持续交付 ( CI/CD )、可观测性、配置管理、事件响应和灾难恢复等运营流程；以及管理此类任务的运营人员。
- 中断是指阻止应用程序正常提供其业务功能的事件。
- 损害是指中断未得到缓解时对应用程序产生的影响。如果应用程序遭受一系列中断，则可能会造成损害。

## 持续韧性

韧性生命周期是一个持续的过程。即使在同一组织中，应用程序团队在每个阶段的执行完整程度也可能会有所不同，具体视应用程序的要求而定。但是，每个阶段越完整，应用程序的韧性水平就越高。

您应该将弹性生命周期视为组织可以实施的标准流程。AWS 故意将弹性生命周期建模为类似于软件开发生命周期 (SDLC)，目的是在开发和运行应用程序的同时，将规划、测试和学习纳入整个操作流程。与许多敏捷开发流程一样，弹性生命周期可以在开发过程的每次迭代中重复。我们建议您随着时间的推移，逐步深化生命周期每个阶段的实践。

## 第一阶段：设定目标

了解所需的韧性水平以及衡量方式是设定目标阶段的基础。如果您没有目标，也无法进行衡量，则很难改进。

并非所有应用程序都需要相同的韧性水平。在设定目标时，请考虑所需的水平，以便作出正确的投资和权衡取舍。一个很好的类比是汽车：一辆车有四个轮胎，但只携带一个备胎。行驶途中多个轮胎同时爆胎可能性很低，而携带多个备胎可能会影响载货空间或燃油效率等其他功能，因此这是一个合理的权衡取舍。

确定目标后，您需要在后续阶段（[第二阶段：设计与实施](#)和[第四阶段：运营](#)）中实施可观测性控制，以了解目标是否已实现。

## 映射关键应用程序

定义韧性目标不应仅仅是一场技术对话。相反，应从以业务为导向的重点入手，了解应用程序应提供的功能以及受损的后果。这种对业务目标的理解随后会延伸到架构、工程和运营等领域。您定义的任何韧性目标都可能适用于所有应用程序，但是衡量目标的方式通常会因应用程序的功能而异。您可能正在运行一个对业务至关重要的应用程序，如果此应用程序受到损害，您的组织可能会损失大量收入或遭受声誉损害。或者，您可能还有另一个不那么重要的应用程序，它可以容忍一些停机时间，而不会对组织的业务能力产生负面影响。

例如，设想一家零售公司的订单管理应用程序。如果订单管理应用程序的组件受损且无法正常运行，则新的销售将无法处理。这家零售公司还在某栋办公楼中为其员工开设了一家咖啡店。咖啡店有一个在线菜单，员工可以在静态网页上访问该菜单。如果该网页不可用，一些员工可能会抱怨，但这不一定会对该公司造成经济损失。基于此示例，企业可能会选择为订单管理应用程序设定更严格的韧性目标，但不会投入大量资源来确保 Web 应用程序的韧性。

确定最关键的应用程序、投入最多精力的领域以及进行权衡取舍的因素，与衡量应用程序在生产中的韧性同样重要。为了更好地了解损害的影响，您可以进行[业务影响分析 \(BIA\)](#)。BIA 提供了一种结构化和系统化的方法来识别关键业务应用程序并确定其优先级，评测潜在的风险和影响，并确定支持依赖关系。BIA 可帮助量化组织中最重要应用程序的停机时间成本。该指标有助于明确特定应用程序受损且无法完成其功能时的成本。在前面的示例中，如果订单管理应用程序受损，零售企业可能会损失大量收入。

## 映射用户故事

在 BIA 过程中，您可能会发现某个应用程序负责多项业务功能，或者某项业务功能需要多个应用程序。以之前的零售公司为例，订单管理功能可能需要单独的应用程序进行结账、促销和定价。如果一个应用程序发生故障，企业和与该公司互动的用户可能会受到影响。例如，公司可能无法添加新订单、提供对促销和折扣的访问或更新其产品的价格。订单管理功能所需的这些不同功能可能依赖于多个应用程序。这些功能还可能有多外部依赖关系，从而使实现纯粹以组件为中心的韧性的过程过于复杂。处理这种情况更好的方法是关注[用户故事](#)，这些故事概述了用户在与一个或一组应用程序交互时所期望的体验。

关注用户故事可以帮助您了解客户体验最重要的部分，这样您就可以建立机制来防范特定的威胁。在前面的示例中，一个用户故事可以是“结账”，其涉及结账应用程序，并且依赖于定价应用程序。另一个用户故事可以是“查看促销活动”，其涉及促销应用程序。在映射最关键的的应用程序及其用户故事之后，您可以开始定义用于衡量这些用户故事韧性的指标。这些指标可以应用于整个产品组合，也可以应用于单个用户故事。

## 定义衡量指标

[恢复点目标 \(RPO\)](#)、[恢复时间目标 \(RTO\)](#) 和 [服务水平目标 \(SLO\)](#) 是标准的行业衡量指标，用于评测给定系统的韧性。RPO 是指在发生故障时企业能够容忍的数据丢失量，而 RTO 则是应用程序在中断后必须恢复可用的速度衡量指标。这两个指标以时间单位衡量：秒、分钟和小时。您还可以衡量应用程序正常运行的时间；即按设计执行其功能并可供其用户访问的时间。这些 SLO 详细说明了客户将获得的预期服务水平，并通过在不到一秒的响应时间内无错误处理请求的百分比 (%) 等指标来衡量（例如，每月 99.99% 的请求将得到响应）。RPO 和 RTO 与灾难恢复策略有关，假设应用程序运行和恢复过程会中断，并且恢复过程范围从恢复备份到重定向用户流量。通过实施高可用性控制措施可解决 SLO，这通常能缩短应用程序的停机时间。

SLO 指标通常用于定义服务水平协议 (SLA)，即服务提供商与最终用户之间的合同。SLA 通常附带财务承诺，并规定当协议未得到满足时服务提供商需支付的罚款。但是，SLA 并非韧性状况的衡量指标，提高 SLA 也不能使您的应用程序更具韧性。

您可以基于 SLO、RPO 和 RTO 开始设定目标。在定义韧性目标并明确了解 RPO 和 RTO 目标之后，您可以使用 [AWS Resilience Hub](#) 对架构进行评估，以发现潜在的韧性相关弱点。AWS Resilience Hub 根据 AWS Well-Architected Framework 最佳实践评估应用程序架构，并针对需要改进的具体方面分享修复指导，以实现您确定的 RTO 和 RPO 目标。

## 创建其他衡量指标

RPO、RTO 和 SLO 是衡量韧性的良好指标，但您也可以从业务角度考虑目标，并围绕应用程序的功能定义目标。例如，您的目标可以是：如果前端和后端之间的延迟增加 40%，则每分钟的成功订单数将保持在 98% 以上。或：即使丢失了特定组件，每秒启动的流数仍将保持在平均值的标准偏差范围内。您还可以创建目标，以缩短已知故障类型的平均恢复时间（MTTR）；例如：如果发生任何已知问题，恢复时间将缩短 x%。创建与业务需求一致的目标可以帮助您预测应用程序应容忍的故障类型。它还可以帮助您确定降低应用程序受损可能性的方法。

如果考量在丢失 5% 为应用程序提供支持的实例时仍继续运行的目标，您可能会认为应用程序应预先扩展，或者能够以足够快的速度进行扩展，以支持在该事件期间产生的额外流量。或者，您可以决定应使用不同的架构模式，如[第二阶段：设计与实施](#)部分中所述。

您还应该针对特定业务目标实施可观测性措施。例如，您可以跟踪平均订单率、平均订单价格、平均订阅读数量或根据应用程序行为提供业务健康状况洞察的其他指标。通过为应用程序实施可观测性功能，您可以创建告警，并在这些指标超出定义的边界时采取措施。[第四阶段：运营](#)部分将更详细地介绍可观测性。

## 第二阶段：设计与实施

在上一阶段，您设定了韧性目标。现在，在设计与实施阶段，您将尝试按照在上一阶段设定的目标来预测故障模式并确定设计选择。您还需要定义变更管理策略，并开发软件代码和基础设施配置。以下各节重点介绍在权衡取舍成本、复杂性和运营开销等因素时应考虑的 AWS 最佳实践。

### AWS Well-Architected 框架

在根据所需的韧性目标架构应用程序时，您需要评估多个因素，并对最优架构进行权衡取舍。要构建高韧性的应用程序，必须考虑设计、构建和部署、安全以及运营等方面。[AWS Well-Architected Framework](#) 提供一组最佳实践、设计原则和架构模式，可帮助您在 AWS 上设计具有韧性的应用程序。AWS Well-Architected Framework 的六大支柱提供了设计和运行具有韧性、安全、高效、经济实惠且可持续系统的最佳实践。该框架提供了一种方法，可以根据最佳实践持续衡量架构，从而确定需要改进的方面。

以下是 AWS Well-Architected Framework 如何帮助您设计和实施满足韧性目标的应用程序的示例：

- **可靠性支柱**：[可靠性支柱](#) 强调构建即使在故障或中断期间也能正确持续运行的应用程序的重要性。例如，AWS Well-Architected Framework 建议您使用微服务架构来缩小和简化应用程序，这样就可以区分应用程序中不同组件的可用性需求。您还可以通过使用节流、指数回退重试、快速失效机制（卸除负载）、幂等性、持续工作、断路器和静态稳定性，找出构建应用程序最佳实践的详细说明。
- **全面审查**：AWS Well-Architected Framework 鼓励根据最佳实践和设计原则对您的架构进行全面审查。它提供了一种持续衡量架构并识别改进领域的方法。
- **风险管理**：AWS Well-Architected Framework 可帮助您识别和管理可能影响应用程序可靠性的风险。通过主动解决潜在的故障场景，您可以降低其发生的可能性或由此造成的损害。
- **持续改进**：韧性是一个持续的过程，AWS Well-Architected Framework 强调持续改进。通过根据 AWS Well-Architected Framework 的指导定期审查和完善您的架构和流程，可以确保您的系统在面对不断变化的挑战和要求时保持韧性。

### 了解依赖关系

了解系统的依赖关系是实现韧性的关键。依赖关系包括应用程序内各组件之间的连接，以及与应用程序外部组件（例如第三方 API 和企业拥有的共享服务）的连接。了解这些连接可帮助您隔离和管理中断，因为一个组件受损可能会影响其他组件。这些知识可以帮助工程师评测损害的影响并进行相应的规划，确保资源得到有效利用。了解依赖关系可以帮助您制定替代策略和协调恢复过程。它还可以帮助您确定哪些情况可以将硬依赖关系替换为软依赖关系，以便在依赖关系受损时，您的应用程序可以继续履

行其业务功能。依赖关系还会影响有关负载均衡和应用程序扩展的决策。对应用程序进行更改时，了解依赖关系至关重要，因为这可以帮助您确定潜在的风险和影响。这些知识可以帮助您创建稳定、具有韧性的应用程序，协助进行故障管理、影响评测、损害恢复、负载均衡、扩展和变更管理。您可以手动跟踪依赖关系，也可以使用工具和服务（例如 [AWS X-Ray](#)）了解分布式应用程序的依赖关系。

## 灾难恢复策略

灾难恢复（DR）策略在构建和运行韧性应用程序中起着关键作用，其主要作用是确保业务连续性。该策略保证即使在灾难性事件期间，关键业务运营也能以尽可能少的损害持续下去，从而最大限度地减少停机时间和潜在的收入损失。DR 策略对于数据保护至关重要，因为其通常包含定期的数据备份和跨多个位置的数据复制，这有助于保护宝贵的业务信息，并有助于防止灾难期间发生完全丢失。此外，许多行业都受到策略的监管，要求企业制定 DR 策略，以保护敏感数据并确保灾难期间服务保持可用。通过确保最大限度地减少服务损害，DR 策略还可以提升客户信任和满意度。实施得当且经常练习的 DR 策略可以缩短灾难发生后的恢复时间，并有助于确保应用程序快速恢复在线状态。此外，灾难可能造成巨额成本，不仅包括由于停机时间造成的收入损失，还包括恢复应用程序和数据的费用。精心设计的 DR 策略有助于防范这些经济损失。

您选择的策略取决于应用程序的特定需求、RTO 和 RPO 以及您的预算。[AWS 弹性灾难恢复](#)是一项专门构建的韧性服务，可用于帮助为本地和基于云的应用程序实施 DR 策略。

有关更多信息，请参阅 AWS 网站上的 [Disaster Recovery of Workloads on AWS](#) 和 [AWS Multi-Region Fundamentals](#)。

## 定义 CI/CD 策略

应用程序受损的常见原因之一是代码或其他变更导致应用程序从之前的已知工作状态发生变化。如果您没有谨慎处理变更管理，则可能会导致频繁的损害。变更频率越高，产生影响的可能性越大。但是，降低变更频率会导致变更集更大，而由于变更集的复杂性较高，因此更有可能导致损害。持续集成和持续交付（CI/CD）实践旨在保持变更小而频繁（从而提高工作效率），同时通过自动化对每项变更进行高层级的检查。一些基本策略包括：

- **完全自动化**：CI/CD 的基本概念是尽可能实现构建和部署流程的自动化。这包括构建、测试、部署甚至监控。自动化管线有助于减少人为错误的可能性，确保一致性，并使流程更加可靠和高效。
- **测试驱动开发（TDD）**：在编写应用程序代码之前编写测试。这种做法可确保所有代码都有相关的测试，从而提高代码的可靠性和自动检查的质量。这些测试在 CI 管线中运行，以验证变更。
- **频繁提交和集成**：鼓励开发人员频繁提交代码并经常进行集成。小而频繁的变更更易于测试和调试，从而降低出现重大问题的风险。自动化降低了每次提交和部署的成本，从而使频繁的集成变为可能。

- **不可变基础设施**：将您的服务器和其他基础设施组件视为静态、不可变的实体。尽可能更换基础设施而不是对其进行修改，使用经过测试并通过管线部署的[代码](#)来构建新的基础设施。
- **回滚机制**：如果出现问题，始终用一种简单、可靠且经常测试的方法来回滚更改。能够快速恢复到之前已知的良好状态对于部署安全至关重要。此功能可以是恢复到之前状态的简单按钮，也可以完全自动并通过告警启动。
- **版本控制**：将所有应用程序代码、配置甚至基础设施作为代码保存在版本控制的存储库中。这种做法有助于确保您可以轻松跟踪变更并在需要时进行恢复。
- **金丝雀部署和蓝绿部署**：首先将应用程序的新版本部署到基础设施的子集，或者维护两个环境（蓝/绿），使您能够在生产环境中验证变更的行为，并在必要时快速回滚。

CI/CD 不仅关乎工具，也关乎文化。创造一种重视自动化、测试和从故障中吸取教训的文化与实施正确的工具和流程同样重要。如果回滚能够快速完成且影响最小，则不应将其视为故障，而应视为一次学习经验。

## 开展 ORR

运营准备情况审查 ( ORR ) 有助于确定运营和程序方面的差距。在 Amazon，我们创建了 ORR，将数十年来运营大规模服务的经验提炼成精心策划的问题，并提供最佳实践指导。ORR 总结了以往的经验教训，并要求新团队确保在其应用中已考虑这些经验教训。ORR 可以提供故障模式或故障原因的列表，用于执行下文韧性模拟部分所述的韧性模拟活动。有关更多信息，请参阅 AWS Well-Architected Framework 网站上的 [Operational Readiness Reviews \( ORR \)](#)。

## 了解 AWS 故障隔离边界

AWS 提供多个故障隔离边界，以帮助您实现韧性目标。您可以使用这些边界，利用其提供的可预测的影响控制范围。您应该熟悉 AWS 服务如何利用这些边界进行设计，以便能够根据应用程序有意选择合适的依赖关系。要了解如何在应用程序中使用边界，请参阅 AWS 网站上的 [AWS Fault Isolation Boundaries](#)。

## 选择响应

系统可以通过多种方式对告警作出响应。有些告警可能需要运营团队作出响应，而另一些告警可能会触发应用程序内的自我修复机制。为了控制自动化成本或管理工程限制，您可决定将一些可以自动化的响应保留为手动操作。对告警响应类型的选择很可能基于实施响应的成本、告警的预期频率、告警的准确性以及根本不响应告警所造成的潜在业务损失等多个因素。

例如，当服务器进程崩溃时，操作系统可能会重新启动该进程，或者可能预调配新服务器并终止旧服务器，亦或可能会指示运营人员远程连接到该服务器并重新启动它。这些响应的结果相同，即重新启动应用程序服务器进程，但实施和维护成本各不相同。

### Note

您可以选择多个响应，以采取深入的韧性方法。例如，在前面的场景中，应用程序团队可能会选择实施所有三种响应，每个响应之间设置时间延迟。如果故障服务器的进程指示器在 30 秒后仍处于告警状态，则团队可以假设操作系统未能重新启动应用程序服务器。因此，他们可能会创建一个自动扩缩组，用于创建新的虚拟服务器并恢复应用程序服务器进程。如果指示器在 300 秒后仍处于告警状态，则可能会向运营人员发送提醒，要求其连接到原始服务器并尝试恢复进程。

应用程序团队和业务部门选择的响应方式应反映出企业通过前期投入工程时间来抵消运营开销的风险偏好。您应该通过仔细考虑每个响应选项的限制和预期维护来选择一种响应：静态稳定性等架构模式、断路器软件模式或操作程序。可能已存在一些标准响应可指导应用程序团队，因此您可以使用集中式架构功能管理的库和模式作为此考量的依据。

## 韧性模拟

韧性模拟记录了应用程序将如何响应不同的预期中断。通过预测中断情况，您的团队可以实施可观测性、自动化控制和恢复流程，以缓解或防止出现中断时的损害。AWS 已使用[韧性分析框架](#)创建了开发韧性模型的指导。该框架可以帮助您预测中断及其对应用程序的影响。通过预测中断，您可以确定构建具有韧性且可靠的应用程序所需的缓解措施。我们建议您在应用程序生命周期的每次迭代中使用韧性分析框架更新韧性模型。在每次迭代中使用此框架可以预测设计阶段的中断，并在生产部署前和部署后测试应用程序，从而有助于减少事故。使用此框架开发韧性模型可帮助您确保实现韧性目标。

## 安全失效

如果您无法避免中断，请安全失效。考虑使用默认的安全失效运行模式创建应用程序，在这种模式下，不会造成重大业务损失。数据库安全失效状态的一个例子是默认为只读操作，即不允许用户创建或更改任何数据。根据数据的敏感性，您甚至可能希望应用程序默认为关闭状态，甚至不执行只读查询。考虑一下应用程序应采用哪种安全失效状态，并在极端条件下默认采用这种运行模式。

## 第三阶段：评估与测试

在生命周期的评估与测试阶段，应用程序或对现有应用程序的变更已完成设计，但尚未发布到生产环境。在此阶段，您将实施活动来测试在前几个阶段已执行的实践并评估结果。该应用程序可能仍在积极开发中，也可能主要开发已经完成，且该应用程序正在进行发布到生产环境前的测试。在此阶段，重点在于开发和运行测试，以验证或否定应用程序是否能满足对既定韧性目标的期望。此外，您还可以制定和测试系统的操作程序。您在[第二阶段：设计与实施](#)阶段制定的部署程序将付诸实践，并对结果进行评估。尽管这些测试和评估活动在生命周期的这一部分开始，但并非到此结束。测试和评估将延续到[第四阶段：运营阶段](#)。

评估与测试阶段分为两个分阶段：[部署前活动](#)和[部署后活动](#)。部署前活动包含在将应用程序部署到任何环境之前应完成的任务，包括部署新版本的软件以及初次将其部署到测试环境中。部署后活动是在将软件部署到测试或生产环境中之后进行的。以下各节将更详细讨论这些分阶段。

### 部署前活动

#### 环境设计

您测试和评估应用程序的环境会影响您对其进行测试的彻底程度，以及您对这些结果是否能准确反映生产环境实际情况的信心。您可以使用 Amazon DynamoDB 等服务，在开发者计算机上本地执行一些集成测试（请参阅 DynamoDB 文档中的[设置 DynamoDB local](#)）。但是，在某些时候，您需要在复制生产环境的环境中进行测试，以获得最高可信度的测试结果。这种环境会产生费用，因此我们建议您对环境采取分阶段或管线化方法，将类似生产环境的测试环境放在管线的后期阶段。

#### 集成测试

集成测试是测试应用程序中明确定义的组件在使用外部依赖关系运行时能否正确执行其功能的过程。这些外部依赖项可能是其他自定义开发的组件、用于应用程序的 AWS 服务、第三方依赖项和本地依赖项。本指南重点介绍验证应用程序韧性的集成测试。其假设已经存在可以验证软件功能准确性的单元测试和集成测试。

我们建议您设计专门测试已实施韧性模式的集成测试，例如断路器模式或卸除负载（请参阅[第二阶段：设计与实施](#)）。面向韧性的集成测试通常涉及向应用程序施加特定的负载，或者使用 [AWS Fault Injection Service \(AWS FIS\)](#) 等功能故意向环境中引入中断。理想情况下，应将所有集成测试作为 CI/CD 管道的一部分运行，并确保每次提交代码时都运行测试。这可以帮助您快速检测导致违反韧性目标的任何代码或配置变更并作出反应。大规模分布式应用程序非常复杂，即使是微小的变更也可能会严重影响应用程序中看似无关部分的韧性。尝试在每次提交时运行测试。AWS 为操作 CI/CD 管道

和其他工具提供了一套出色的 DevOps 工具。有关更多信息，请参阅 AWS 网站上的 [DevOps “简介”](#)。

## AWS

### 自动化部署管线

部署到预生产环境并在其中进行测试是一项重复且复杂的任务，最好留给自动化处理。该过程的自动化可以腾出人力资源，并减少出错的机会。自动化此过程的机制通常称为管线。在创建管线时，我们建议您设置一系列越来越接近生产配置的测试环境。您可以使用这一系列的环境来反复测试应用程序。第一个环境提供的功能集比生产环境更为有限，但成本要低得多。后续环境应添加服务并扩展，以更接近地反映生产环境。

首先在第一个环境中进行测试。在您的部署通过第一个测试环境中的所有测试后，让应用程序在一定负载下运行一段时间，以观察是否会随着时间的推移出现任何问题。确认您已正确配置可观测性（请参阅本指南后面的告警精度），以便能够检测到出现的任何问题。此观测期成功完成后，将您的应用程序部署到下一个测试环境并重复该过程，在环境支持的范围内添加其他测试或负载。以这种方式对应用程序进行充分的测试后，您可以使用之前设置的部署方法将应用程序部署到生产环境中（请参阅本指南前面的定义 CI/CD 策略）。Amazon Builders Library 中的 [Automating safe, hands-off deployments](#) 一文是一个优秀的资源，其中介绍了 Amazon 如何自动化代码部署。生产部署之前的环境数量会有所不同，具体视应用程序的复杂性及其依赖关系的类型而定。

### 负载测试

从表面上看，负载测试与集成测试类似。您可以测试应用程序的某项独立功能及其外部依赖关系，以验证其是否按预期运行。负载测试不仅仅是集成测试，其侧重于应用程序在明确定义的负载下如何运行。负载测试需要验证功能是否正确，因此必须在成功完成集成测试之后进行。重要的是要了解应用程序在预期负载下的响应情况，以及负载超出预期时应用程序的行为。这可以帮助您验证是否已经实施了必要的机制，以确保您的应用程序在极端负载下仍能保持韧性。有关负载测试的综合指南 AWS，请参阅 AWS 解决方案库 AWS 中的 [分布式负载测试](#)。

### 部署前活动

韧性是一个持续的过程，在部署应用程序之后，必须继续评估应用程序的韧性。部署后活动（例如持续的韧性评估）的结果可能需要您重新评估并更新在韧性生命周期早期执行的一些韧性活动。

### 进行韧性评估

将应用程序部署到生产环境后，评估韧性不会停止。即使您有明确定义的自动化部署管线，有时也可能直接在生产环境中进行更改。此外，在部署前韧性验证中，可能还有一些因素尚未考虑在内。 [AWS](#)

[Resilience Hub](#)提供了一个中心位置，可用于评测已部署的架构是否满足您定义的 RPO 和 RTO 需求。您可以使用此服务对应用程序的弹性进行按需评估，自动进行评估，甚至可以将其集成到您的 CI/CD 工具中，如 AWS 博客文章“使用和[持续评估应用程序弹性](#)”中所述。AWS Resilience Hub AWS CodePipeline将这些评估自动化是一种最佳实践，因为它有助于确保您在生产中持续评估自己的韧性状况。

## DR 测试

在[第二阶段：设计与实施](#)中，您制定了灾难恢复 (DR) 策略作为系统的一部分。在第四阶段，您应该测试 DR 程序，以确保您的团队为事件做好充分准备，并且您的程序按预期运行。您应定期测试所有 DR 程序（包括失效转移和失效自动恢复），并查看每项练习的结果，以确定是否以及如何更新系统的程序以获得最佳结果。在最初制定 DR 测试时，请提前安排好测试，并确保整个团队都了解预期结果、如何衡量结果，以及将使用哪种反馈机制根据结果更新程序。在您熟练运行计划的 DR 测试后，可以考虑运行无预告 DR 测试。真实的灾难不会按计划发生，因此您需要做好随时执行计划的准备。但是，无预告并不意味着计划外。主要利益相关者仍需要对事件进行规划，以确保进行适当的监控，并确保客户和关键应用程序不会受到不良影响。

## 偏差检测

即使采用了自动化和明确定义的程序，生产应用程序中的配置也可能发生意外的变更。要检测应用程序配置的变更，您应该有检测偏差的机制，后者是指与基准配置的偏差。要了解如何检测 AWS CloudFormation 堆栈中的偏差，请参阅文档中的[检测堆栈和资源的非托管配置更改](#)。AWS CloudFormation 要检测应用程序 AWS 环境中的偏差，请参阅 AWS Control Tower 文档[AWS Control Tower中的检测和解决偏差](#)。

## 综合测试

[综合测试](#)是创建可配置软件的过程，该软件按计划在生产环境 APIs 中运行，以模拟最终用户体验的方式测试您的应用程序。这些测试有时称为金丝雀，指的是该术语最初在煤炭开采中的使用。综合测试通常可以在应用程序出现中断时提供早期警告，即使损害是部分或间歇性的，这种情况通常称为[灰色故障](#)。

## 混沌工程

混沌工程是一个系统化的过程，它通过以风险可控的方式，故意让应用程序经历各种中断事件，密切监控其响应，并实施必要的改进。其目的是验证或挑战有关应用程序处理此类中断能力的假设。混沌工程并非任由这些事件随机发生，而是使工程师能够在受控的环境中编排实验，通常是在流量较低的时期，并提供随时可用的工程支持来有效缓解。

混沌工程首先要了解所考虑应用程序的正常运行条件（称为稳态）。在此基础上，您可以提出一个假设，详细说明在存在中断的情况下应用程序的成功行为。您可运行实验，其中涉及故意注入中断，包括但不限于网络延迟、服务器故障、硬盘驱动器错误以及外部依赖关系损害。然后，您可以分析实验结果，并根据经验教训增强应用程序的韧性。该实验是改善应用程序各个方面（包括其性能）的宝贵工具，可发现原本可能隐藏的潜在问题。此外，混沌工程有助于揭示可观测性和告警工具中的缺陷，并帮助您对其进行完善。它还有助于缩短恢复时间和提升运营技能。混沌工程加速了最佳实践的采用，培养了持续改进的心态。最终，它使团队能够通过定期练习和重复来构建和磨练其运营技能。

AWS 建议您在非生产环境中开始混沌工程工作。您可以使用 [AWS Fault Injection Service \(AWS FIS\)](#) 对通用故障以及 AWS 特有的故障进行混沌工程实验。这项完全托管的服务包括停止状态告警和完全权限控制，因此您可以安全放心地轻松采用混沌工程。

## 第四阶段：运营

完成[第三阶段：评估与测试](#)后，即可将应用程序部署到生产环境。在运营阶段，您可以将应用程序部署到生产环境并管理客户的体验。应用程序的设计与实施决定了其许多韧性成果，但本阶段的重点是您的系统用来维持和提升韧性的运营实践。建立卓越运营文化有助于在这些实践中建立标准并保持一致性。

### 可观测性

了解客户体验最重要的部分是通过监控和告警。您需要对应用程序进行检测以了解其状态，并且需要不同的视角，这意味着您需要从服务器端和客户端（通常使用金丝雀）进行衡量。您的指标应包括有关应用程序与其依赖关系交互的数据，以及[与故障隔离边界一致的维度](#)。您还应该生成日志，以提供有关应用程序执行的每个工作单元的其他详细信息。您可以考虑使用 [Amazon CloudWatch 嵌入式指标格式](#) 等解决方案来合并指标和日志。您可能会发现对更高可观测性的需求永无止境，因此请考虑实施所需检测级别所需的成本、工作量和复杂性权衡取舍。

以下链接提供了检测应用程序和创建告警的最佳实践：

- [Monitoring production services at Amazon](#) ( AWS re:Invent 2020 演讲 )
- [Amazon Builders' Library: Operational Excellence at Amazon](#) ( AWS re:Invent 2021 演讲 )
- [Observability best practices at Amazon](#) ( AWS re:Invent 2022 演讲 )
- [检测分布式系统的运营可见性](#) ( Amazon Builders' Library 文章 )
- [Building dashboards for operational visibility](#) ( Amazon Builders' Library 文章 )

### 事件管理

当您的告警（或者更糟糕的是，您的客户）告知您出现问题时，您应该有一个事件管理流程来处理损害。该流程应包括：安排值班的运营人员、上报问题，以及制定运行手册，以采用一致的排查方法，帮助消除人为错误。但是，损害通常并非孤立发生；一个应用程序可能会影响依赖它的多个其他应用程序。通过了解所有受影响的应用程序并将召集多个团队的运营人员一起参加一次电话会议，您可以快速解决问题。但是，根据您的组织的规模和结构，此流程可能需要一个集中式运营团队。

除了设置事件管理流程外，您还应该通过控制面板定期查看指标。定期查看可帮助您了解客户体验和应用程序性能的长期趋势，从而在造成重大生产影响之前识别问题和瓶颈。以一致、标准化的方式查看指标可以带来显著收益，但需要获得自上而下的支持和投入时间。

以下链接提供有关构建控制面板和运营指标查看的最佳实践：

- [Building dashboards for operational visibility](#) ( Amazon Builders' Library 文章 )
- [Amazon's approach to failing successfully](#) ( AWS re:Invent 2019 演讲 )

## 持续韧性

在[第二阶段：设计与实施](#)和[第三阶段：评估与测试](#)期间，您在将应用程序部署到生产环境之前启动了审查和测试活动。在运营阶段，您应该继续迭代生产中的这些活动。您应该通过 [AWS Well-Architected Framework 审查](#)、[运营准备情况审查 \( ORR \)](#) 和[韧性分析框架](#)，定期审查应用程序的韧性状况。这有助于确保您的应用程序不会偏离既定的基准和标准，并使您随时了解新的或更新的指引。这些持续的韧性活动可以帮助您发现以前未预见的中断，并帮助您制定新的缓解措施。

在预生产环境中成功运行[实际试用](#)和[混沌工程](#)实验之后，您可能还需要考虑在生产环境中运行这些实验。实际试用模拟您已经建立韧性机制来缓解的已知事件。例如，实际试用可模拟 AWS 区域服务损害，并实施多区域失效转移。尽管实施这些活动可能需要投入大量精力，但这两种做法都能增强您对系统韧性的信心，确保其能承受设计时预期的故障模式。

通过运行应用程序、应对运营事件、查看指标和测试应用程序，您将获得大量响应与学习的机会。

## 第五阶段：响应与学习

您的应用程序响应中断事件的方式会影响其可靠性。从经验中吸取教训，并了解您的应用程序如何应对过去的中断，对于提高其可靠性也至关重要。

响应与学习阶段重点介绍您可以实施以更好地应对应用程序中断事件的实践。它还包括一些实践，可帮助您从运营团队和工程师的经验中汲取最大价值。

### 创建事件分析报告

事件发生时，首要行动是尽快阻止对客户和业务的进一步损害。应用程序恢复后，下一步是了解所发生的事件并制定防止再次发生的措施。这种事件后分析通常以报告的形式收集，其记录了导致应用程序受损的一系列事件，以及中断对应用程序、客户和业务的影响。此类报告将成为宝贵的学习构件，应在整个企业内广泛共享。

#### Note

进行事件分析时，切忌归咎责任。假设所有运营人员都根据所掌握的信息采取了最好且最恰当的行动方案。不要在报告中提及运营人员或工程师的姓名。将人为错误作为损害原因可能会导致团队成员为了自保而有所顾虑，从而使收集到的信息不准确或不完整。

一份优秀的事件分析报告，如 [Amazon 更正错误 \(COE\) 流程](#) 中所记录，应遵循标准化格式，并尝试尽可能详细地记录导致应用程序受损的情况。该报告详细介绍了一系列带有时间戳的事件，并收集了定量数据（通常是监控控制面板中的指标和屏幕截图），这些数据描述了应用程序在时间线上的可衡量状态。该报告应收集采取行动的运营人员和工程师的思考过程，以及促使他们得出结论的信息。报告还应详细说明不同指标的性能：例如，发出了哪些告警、这些告警是否准确反映了应用程序的状态、事件与生成的告警之间的时间延迟，以及解决事件的时间。时间线还记录了已启动的运行手册或自动化，以及它们如何帮助应用程序恢复有用状态。这些时间线要素可帮助您的团队了解自动化响应和运营人员响应的有效性，包括他们解决问题的速度以及缓解中断的有效性。

这种对历史事件的详细记录是极具价值的教育工具。团队应将这些报告存储在可供整个企业使用的中央存储库中，以便其他人能够查看事件并从中学习。这可以提升团队对生产环境中可能出现问题的直觉。

详细事件报告存储库也成为运营人员培训材料的来源。团队可以使用事件报告来激发桌面或现场实际试用的灵感，团队将从中获得重现报告中所记录时间线的信息。运营人员可以使用时间线中的部分信息演练场景，并描述其将要采取的行动。然后，实际试用的主持人可以根据运营人员的行动，指导应用程序应如何响应。这可以培养运营人员的故障排除技能，因此他们可以更轻松地预测问题并对其进行排查。

负责应用程序可靠性的集中式团队应将这些报告保留在一个可供整个组织访问的集中式库中。该团队还应负责维护报告模板和培训团队如何完成事件分析报告。可靠性团队应定期查看报告，以检测可以通过软件库、架构模式或团队流程变更改进的业务趋势。

## 进行运营回顾

正如[第四阶段：运营](#)中所述，运营回顾是回顾最近发布的功能、事件和运营指标的机会。运营回顾同时也是一个与组织内更广泛工程群体分享功能发布与事件经验教训的契机。在运营回顾期间，这些团队会回顾已回滚的功能部署、发生的事件以及这些事件的处理方式。这为整个组织的工程师提供了学习他人的经验并提出问题的机会。

向公司的工程群体开放运营回顾，以便他们可以更多地了解运营业务的 IT 应用程序以及他们可能遇到的问题类型。设计、实施和部署其他业务应用程序时，他们会运用这些知识。

## 检查告警性能

如运营阶段所述，告警可能会导致控制面板提醒、创建工单、发送电子邮件或寻呼运营人员。应用程序将配置许多告警，以监控其运行的各个方面。随着时间的推移，应检查这些告警的准确性和有效性，以提高告警精度、减少误报并整合重复的提醒。

### 告警精度

告警应尽可能具体，以减少您解读或诊断导致告警的特定中断所花费的时间。当应对应用程序受损而发出告警时，接收告警并作出响应的运营人员必须首先解读告警传达的信息。这些信息可能是一个简单的错误代码，对应某个操作流程（例如恢复过程）；也可能包括应用程序日志中的几行，您必须查看其内容才能了解告警发出的原因。当您的团队学会更有效地运行应用程序后，他们应该完善这些告警，使其尽可能清晰简洁。

您无法预见应用程序可能出现的所有中断，因此总会有需要运营人员进行分析和诊断的一般告警。您的团队应努力减少一般告警的数量，以缩短响应时间并缩短平均维修时间（MTTR）。理想情况下，告警与自动化响应或人工响应之间应存在一对一的关系。

### 误报

随着时间的推移，运营人员会忽略不需要其采取任何操作，但会生成电子邮件、寻呼或工单的提醒。定期（或作为事件分析的一部分）检查这些告警，以确定经常被忽略或不需要运营人员采取任何行动的告警（误报）。您应该努力消除告警或改进告警，使其向运营人员发出可操作的提醒。

## 漏报

在事件发生期间，配置为在事件发生期间发出提醒的告警可能会失效，这可能是由于某个事件以意想不到的方式影响了应用程序。作为事件分析的一部分，您应该查看本应发出但未发出的告警。应该努力改进这些告警，使其更好地反映事件可能引发的各种情况。或者，您可能必须创建其他告警，这些告警虽然对应相同的中断事件，但由不同的中断症状引发。

## 重复提醒

影响应用程序的中断可能会导致多种症状，并可能产生多个告警。您应定期（或作为事件分析的一部分）查看已发出的告警和提醒。如果运营人员收到重复的提醒，请创建汇总告警，将其合并为一条提醒消息。

## 进行指标检查

您的团队应收集有关应用程序的运营指标，例如按严重性划分的每月事件数量、检测事件的时间、确定原因的时间、修复的时间以及创建的工单数量、发送的提醒数和发出的寻呼数。至少每月检查一次这些指标，以了解运营人员的负担、其处理的信噪比（例如，信息警报与可操作提醒），以及团队是否提升了其控制下的应用程序的运营能力。利用此检查了解运营团队可衡量方面的趋势。向团队征求有关如何改进这些指标的建议。

## 提供培训与赋能

详细记录导致事件或异常行为的应用程序及其运行环境并非易事。此外，模拟应用程序的韧性以预测此类场景也并不简单。您的组织应投入资源，为运营团队和开发人员提供培训与赋能资料，帮助他们参与韧性模拟、事件分析、实际试用和混沌工程实验等活动。这将提升团队生成报告的准确性和所掌握知识的深度。这些团队也将能够更好地预测故障，而无需依赖少数经验丰富的工程师团队通过定期检查来提供其洞察。

## 创建事件知识库

事件报告是事件分析的标准输出。即使应用程序没有受到损害，也应使用相同或类似的报告来记录检测到异常应用程序行为的场景。使用相同的标准化报告结构来收集混沌实验和实际试用的结果。该报告显示了导致事件或其他意外行为的应用程序及其环境的快照。您应该将这些标准化报告存储在可供企业内所有工程师访问的中央存储库中。

然后，运营团队和开发人员可以搜索此知识库，以了解过去哪些原因中断了应用程序、哪些类型的场景可能导致中断，以及哪些因素可以防止应用程序受损。该知识库成为提升运营团队和开发人员技能的加

速器，使他们能够分享知识和经验。此外，您可以将这些报告用作实际试用或混沌实验的培训材料或场景，以提升运营团队的直觉和排查中断的能力。

#### Note

标准化报告格式还能让读者感到熟悉，并帮助他们更快地找到所需的信息。

## 深入实施韧性

如前所述，高成熟度的组织会针对一次告警实施多重响应。由于无法保证响应一定有效，因此通过分层响应，应用程序能够更好地实现优雅降级。我们建议您为每个指标至少实施两种响应，以确保单一响应不会成为可能导致 DR 场景的单点故障。这些响应层应按顺序创建，这样仅在上一个响应无效时，才会执行后续响应。您不应该对单个告警运行多个分层的响应。相反地，应使用告警来指示响应是否失败；如果是，则启动下一层响应。

## 结论和资源

本指南展示了一个生命周期，通过在以下五个阶段实施最佳实践，帮助您持续提升应用程序的韧性：设定目标、设计与实施、评估与测试、运营以及响应与学习。

有关本指南中所讨论服务和概念的更多信息，请参阅以下资源。

AWS 服务：

- [AWS Backup](#)
- [AWS 弹性灾难恢复](#)
- [AWS Fault Injection Service \(AWS FIS\)](#)
- [AWS Resilience Hub](#)
- [Amazon 应用程序恢复控制器 \(ARC\)](#)
- [AWS X-Ray](#)

博文和文章：

- [Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS](#)
- [AWS Fault Isolation Boundaries](#)
- [AWS Multi-Region Fundamentals](#)
- [Chaos Engineering in the cloud](#)
- [Continually assessing application resilience with AWS Resilience Hub and AWS CodePipeline](#)
- [Disaster Recovery of On-Premises Applications to AWS](#)
- [可靠性支柱 – AWS Well-Architected Framework](#)
- [韧性分析框架](#)

# 贡献者

本指南的贡献者包括：

- Bruno Emer , AWS 首席解决方案架构师
- Clark Richey , AWS 首席解决方案架构师
- Elaine Harvey , AWS 可靠性服务总经理
- Jason Barto , AWS 首席解决方案架构师
- John Formento , AWS 首席解决方案架构师
- Lisi Lewis , AWS 高级产品营销经理
- Michael Haken , AWS 首席解决方案架构师
- Neeraj Kumar , AWS 首席解决方案架构师
- Wangechi Doble , AWS 首席解决方案架构师

# 文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 [RSS 源](#)。

变更	说明	日期
<a href="#">初次发布</a>	—	2023 年 10 月 6 日

# AWS 规范性指导词汇表

以下是 AWS 规范性指导提供的策略、指南和模式中的常用术语。若要推荐词条，请使用术语表末尾的提供反馈链接。

## 数字

### 7 R

将应用程序迁移到云中的 7 种常见迁移策略。这些策略以 Gartner 于 2011 年确定的 5 R 为基础，包括以下内容：

- **重构/重新架构**：充分利用云原生功能来提高敏捷性、性能和可扩展性，以迁移应用程序并修改其架构。这通常涉及到移植操作系统和数据库。示例：将本地 Oracle 数据库迁移到 Amazon Aurora PostgreSQL 兼容版。
- **更换平台**：将应用程序迁移到云中，并进行一定程度的优化，以利用云功能。示例：将本地 Oracle 数据库迁移到 AWS Cloud 中的 Amazon Relational Database Service ( Amazon RDS ) for Oracle。
- **重新购买**：转换到其他产品，通常是从传统许可转向 SaaS 模式。示例：将客户关系管理 ( CRM ) 系统迁移到 Salesforce.com。
- **重新托管 ( 直接迁移 )**：将应用程序迁移到云中，无需进行任何更改即可利用云功能。示例：将本地 Oracle 数据库迁移到 AWS Cloud 中 EC2 实例上的 Oracle。
- **重新放置 ( 虚拟机监控器级直接迁移 )**：将基础设施迁移到云中，无需购买新硬件、重写应用程序或修改现有操作。您将服务器从本地平台迁移到同一平台的云服务中。示例：将 Microsoft Hyper-V 应用程序迁移到 AWS。
- **保留 ( 重访 )**：将应用程序保留在源环境中。其中可能包括需要进行重大重构的应用程序，并且您希望将工作推迟到以后，以及您希望保留的遗留应用程序，因为迁移它们没有商业上的理由。
- **停用**：停用或删除源环境中不再需要的应用程序。

## A

### ABAC

请参阅[基于属性的访问控制](#)。

## 抽象服务

请参阅[托管服务](#)。

## ACID

请参阅[原子性、一致性、隔离性、持久性](#)。

## 主动-主动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步（通过使用双向复制工具或双写操作），两个数据库都在迁移期间处理来自连接应用程序的事务。这种方法支持小批量、可控的迁移，而不需要一次性割接。它比[主动-被动迁移](#)更灵活，但工作量更大。

## 主动-被动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步，但在将数据复制到目标数据库时，只有源数据库处理来自连接应用程序的事务。目标数据库在迁移期间不接受任何事务。

## 聚合函数

一种 SQL 函数，它对一组行进行操作并计算该组的单个返回值。聚合函数的示例包括 SUM 和 MAX。

## AI

请参阅[人工智能](#)。

## AIOps

请参阅[人工智能运营](#)。

## 匿名化

永久删除数据集中个人信息的过程。匿名化可以帮助保护个人隐私。匿名化数据不再被视为个人数据。

## 反模式

一种用于解决反复出现的问题的常用解决方案，而在这类问题中，此解决方案适得其反、无效或不如替代方案有效。

## 应用程序控制

一种安全方法，仅允许使用经批准的应用程序，以帮助保护系统免受恶意软件的侵害。

## 应用程序组合

有关组织使用的每个应用程序的详细信息的集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是[产品组合发现和分析过程](#)的关键，有助于识别需要进行迁移、现代化和优化的应用程序并确定其优先级。

## 人工智能 ( AI )

计算机科学领域致力于使用计算技术执行通常与人类相关的认知功能，例如学习、解决问题和识别模式。有关更多信息，请参阅[什么是人工智能？](#)

## 人工智能操作 (AIOps)

使用机器学习技术解决运营问题、减少运营事故和人为干预以及提高服务质量的过程。有关如何在 AIOps AWS 迁移策略中使用的更多信息，请参阅[操作集成指南](#)。

## 非对称加密

一种加密算法，使用一对密钥，一个公钥用于加密，一个私钥用于解密。您可以共享公钥，因为它不用于解密，但对私钥的访问应受到严格限制。

## 原子性、一致性、隔离性、持久性 ( ACID )

一组软件属性，即使在出现错误、电源故障或其他问题的情况下，也能保证数据库的数据有效性和操作可靠性。

## 基于属性的访问权限控制 ( ABAC )

根据用户属性（如部门、工作角色和团队名称）创建精细访问权限的做法。有关更多信息，请参阅 AWS Identity and Access Management (IAM) [文档](#) [AWS 中的 AB AC](#)。

## 权威数据来源

存储主要数据版本的位置，被认为是最可靠的信息源。您可以将数据从权威数据来源复制到其他位置，以便处理或修改数据，例如对数据进行匿名化、编辑或假名化。

## 可用区

中的一个不同位置 AWS 区域，不受其他可用区域故障的影响，并向同一区域中的其他可用区提供低成本、低延迟的网络连接。

## AWS 云采用框架 (AWS CAF)

该框架包含指导方针和最佳实践 AWS，可帮助组织制定高效且有效的计划，以成功迁移到云端。AWS CAF 将指导分为六个重点领域，称为视角：业务、人员、治理、平台、安全和运营。业务、人员和治理角度侧重于业务技能和流程；平台、安全和运营角度侧重于技术技能和流程。例如，人

员角度针对的是负责人力资源 ( HR )、人员配置职能和人员管理的利益相关者。从这个角度来看，AWS CAF 为人员发展、培训和沟通提供了指导，以帮助组织为成功采用云做好准备。有关更多信息，请参阅 [AWS CAF 网站](#) 和 [AWS CAF 白皮书](#)。

## AWS 工作负载资格框架 (AWS WQF)

一种评估数据库迁移工作负载、推荐迁移策略和提供工作估算的工具。AWS WQF 包含在 AWS Schema Conversion Tool (AWS SCT) 中。它用来分析数据库架构和代码对象、应用程序代码、依赖关系和性能特征，并提供评测报告。

## B

### 恶意机器人

一种旨在扰乱或伤害个人或组织的[机器人](#)。

### BCP

请参阅[业务连续性计划](#)。

### 行为图

一段时间内资源行为和交互的统一交互式视图。您可以使用 Amazon Detective 的行为图来检查失败的登录尝试、可疑的 API 调用和类似的操作。有关更多信息，请参阅 Detective 文档中的[行为图中的数据](#)。

### 大端序系统

一个先存储最高有效字节的系统。另请参阅[字节顺序](#)。

### 二进制分类

一种预测二进制结果 ( 两个可能的类别之一 ) 的过程。例如，您的 ML 模型可能需要预测诸如“该电子邮件是否为垃圾邮件？”或“这个产品是书还是汽车？”之类的问题

### bloom 筛选条件

一种概率性、内存高效的数据结构，用于测试元素是否为集合的成员。

### 蓝/绿部署

一种部署策略，您可以创建两个独立但完全相同的环境。在一个环境中运行当前应用程序版本 ( 蓝色 )，在另一个环境中运行新应用程序版本 ( 绿色 )。此策略可帮助您在影响最小的情况下快速回滚。

## 自动程序

一种通过互联网运行自动任务并模拟人类活动或交互的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的 Web 爬网程序。还有一些被称为恶意机器人的机器人，其目的是扰乱或伤害个人或组织。

## 僵尸网络

被[恶意软件](#)感染并受单方（称为僵尸网络控制者或僵尸网络操作者）控制的[僵尸网络](#)。僵尸网络是最著名的扩展机器人及其影响力的机制。

## 分支

代码存储库的一个包含区域。在存储库中创建的第一个分支是主分支。您可以从现有分支创建新分支，然后在新分支中开发功能或修复错误。为构建功能而创建的分支通常称为功能分支。当功能可以发布时，将功能分支合并回主分支。有关更多信息，请参阅[关于分支](#)（GitHub 文档）。

## 紧急（break-glass）访问

在特殊情况下，通过批准的流程，用户 AWS 账户 可以快速访问他们通常没有访问权限的内容。有关更多信息，请参阅 AWS Well-Architected Guidance 中的 [Implement break-glass procedures](#) 指示器。

## 棕地策略

您环境中的现有基础设施。在为系统架构采用棕地策略时，您需要围绕当前系统和基础设施的限制来设计架构。如果您正在扩展现有基础设施，则可以将棕地策略和[全新](#)策略混合。

## 缓冲区缓存

存储最常访问的数据的内存区域。

## 业务能力

企业如何创造价值（例如，销售、客户服务或营销）。微服务架构和开发决策可以由业务能力驱动。有关更多信息，请参阅[在 AWS 上运行容器化微服务](#)白皮书中的[围绕业务能力进行组织](#)部分。

## 业务连续性计划（BCP）

一项计划，旨在应对大规模迁移等破坏性事件对运营的潜在影响，并使企业能够快速恢复运营。

# C

## CAF

请参阅 [AWS 云采用框架](#)。

## 金丝雀部署

缓慢而渐进地向最终用户发布版本。当您确信无误后，即可部署新版本，并完全替换当前版本。

## CCoE

请参阅[云卓越中心](#)。

## CDC

请参阅[更改数据捕获](#)。

## 更改数据捕获 (CDC)

跟踪数据来源（如数据库表）的更改并记录有关更改的元数据的过程。您可以将 CDC 用于各种目的，例如审计或复制目标系统中的更改以保持同步。

## 混沌工程

故意引入故障或破坏性事件来测试系统的韧性。您可以使用 [AWS Fault Injection Service \(AWS FIS\)](#) 来执行实验，对您的 AWS 工作负载施加压力并评估其响应。

## CI/CD

请参阅[持续集成和持续交付](#)。

## 分类

一种有助于生成预测的分类流程。分类问题的 ML 模型预测离散值。离散值始终彼此不同。例如，一个模型可能需要评估图像中是否有汽车。

## 客户端加密

在目标 AWS 服务 收到数据之前，对数据进行本地加密。

## 云卓越中心 (CCoE)

一个多学科团队，负责推动整个组织的云采用工作，包括开发云最佳实践、调动资源、制定迁移时间表、领导组织完成大规模转型。有关更多信息，请参阅 AWS Cloud 企业战略博客上的 [CCoE 帖子](#)。

## 云计算

通常用于远程数据存储和 IoT 设备管理的云技术。云计算通常连接到[边缘计算](#)技术。

## 云运营模型

在 IT 组织中，一种用于构建、完善和优化一个或多个云环境的运营模型。有关更多信息，请参阅[构建您的云运营模型](#)。

## 云采用阶段

组织迁移到 AWS Cloud 中时通常会经历四个阶段：

- 项目 - 出于概念验证和学习目的，开展一些与云相关的项目
- 基础 — 进行基础投资以扩大云采用率（例如，创建着陆区、定义 CCo E、建立运营模型）
- 迁移 - 迁移单个应用程序
- 重塑 - 优化产品和服务，在云中创新

Stephen Orban 在 AWS Cloud 企业战略博客的博客文章 [《云优先之旅和采用阶段》](#) 中定义了这些阶段。有关它们与 AWS 迁移策略的关系的信息，请参阅 [迁移准备指南](#)。

## CMDB

请参阅 [配置管理数据库](#)。

## 代码存储库

通过版本控制过程存储和更新源代码和其他资产（如文档、示例和脚本）的位置。常见的云存储库包括 GitHub 或 Bitbucket Cloud。每个版本的代码都称为一个分支。在微服务结构中，每个存储库都专门用于一个功能。单个 CI/CD 管线可以使用多个存储库。

## 冷缓存

一种空的、填充不足或包含过时或不相关数据的缓冲区缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取，这比从缓冲区缓存读取要慢。

## 冷数据

很少访问的数据，且通常是历史数据。查询此类数据时，通常可以接受慢速查询。将这些数据转移到性能较低且成本更低的存储层或类别可以降低成本。

## 计算机视觉 ( CV )

一种 [AI](#) 领域，它使用机器学习来分析和提取数字图像和视频等视觉格式中的信息。例如，Amazon SageMaker AI 为 CV 提供了图像处理算法。

## 配置偏移

对于工作负载而言，一种偏离预期状态的配置更改。这可能会导致工作负载变得不合规，且通常是渐进的，不是故意的。

## 配置管理数据库 ( CMDB )

一种存储库，用于存储和管理有关数据库及其 IT 环境的信息，包括硬件和软件组件及其配置。您通常在迁移的产品组合发现和分析阶段使用来自 CMDB 的数据。

## 合规性包

一系列 AWS Config 规则和补救措施，您可以汇编这些规则和补救措施，以自定义您的合规性和安全性检查。您可以使用 YAML 模板将一致性包作为单个实体部署在 AWS 账户 和区域或整个组织中。有关更多信息，请参阅 AWS Config 文档中的 [一致性包](#)。

## 持续集成和持续交付 (CI/CD)

自动执行软件发布过程的源代码、构建、测试、暂存和生产阶段的过程。CI/CD 通常被描述为管道。CI/CD 可以帮助您实现流程自动化、提高生产力、提高代码质量和更快地交付。有关更多信息，请参阅[持续交付的优势](#)。CD 也可以表示持续部署。有关更多信息，请参阅[持续交付与持续部署](#)。

## CV

请参阅[计算机视觉](#)。

## D

### 静态数据

网络中静止的数据，例如存储中的数据。

### 数据分类

根据网络中数据的关键性和敏感性对其进行识别和分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它可以帮助您确定对数据的适当保护和保留控制。数据分类是 Well-Architected AWS d Framework 中安全支柱的一个组成部分。有关详细信息，请参阅[数据分类](#)。

### 数据漂移

生产数据与用来训练机器学习模型的数据之间的有意义差异，或者输入数据随时间推移的有意义变化。数据漂移可能降低机器学习模型预测的整体质量、准确性和公平性。

### 传输中数据

在网络中主动移动的数据，例如在网络资源之间移动的数据。

### 数据网格

一种架构框架，可提供分布式、去中心化的数据所有权以及集中式管理和治理。

### 数据最少化

仅收集并处理绝对必要数据的原则。在中进行数据最小化 AWS Cloud 可以降低隐私风险、成本和分析碳足迹。

## 数据边界

AWS 环境中的一组预防性防护措施，可帮助确保只有可信身份才能访问来自预期网络的可信资源。有关更多信息，请参阅在[上构建数据边界](#)。AWS

## 数据预处理

将原始数据转换为 ML 模型易于解析的格式。预处理数据可能意味着删除某些列或行，并处理缺失、不一致或重复的值。

## 数据溯源

在数据的整个生命周期跟踪其来源和历史的过程，例如数据如何生成、传输和存储。

## 数据主体

正在收集和处理其数据的人。

## 数据仓库

一种支持商业智能（例如分析）的数据管理系统。数据仓库通常包含大量历史数据，通常用于查询和分析。

## 数据库定义语言（DDL）

在数据库中创建或修改表和对象结构的语句或命令。

## 数据库操作语言（DML）

在数据库中修改（插入、更新和删除）信息的语句或命令。

## DDL

请参阅[数据库定义语言](#)。

## 深度融合

组合多个深度学习模型进行预测。您可以使用深度融合来获得更准确的预测或估算预测中的不确定性。

## 深度学习

一个 ML 子字段使用多层神经网络来识别输入数据和感兴趣的目标变量之间的映射。

## defense-in-depth

一种信息安全方法，经过深思熟虑，在整个计算机网络中分层实施一系列安全机制和控制措施，以保护网络及其中数据的机密性、完整性和可用性。当你采用这种策略时 AWS，你会在 AWS

Organizations 结构的不同层面添加多个控件来帮助保护资源。例如，一种 defense-in-depth 方法可以结合多因素身份验证、网络分段和加密。

## 委派管理员

在中 AWS Organizations，兼容的服务可以注册 AWS 成员帐户来管理组织的帐户并管理该服务的权限。此帐户被称为该服务的委托管理员。有关更多信息和兼容服务列表，请参阅 AWS Organizations 文档中[使用 AWS Organizations 的服务](#)。

## 部署

使应用程序、新功能或代码修复在目标环境中可用的过程。部署涉及在代码库中实现更改，然后在应用程序的环境中构建和运行该代码库。

## 开发环境

请参阅[环境](#)。

## 侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出提醒。这些控制是第二道防线，提醒您注意绕过现有预防性控制的安全事件。有关更多信息，请参阅在 AWS 上实施安全控制中的[侦测性控制](#)。

## 开发价值流映射 ( DVSM )

用于识别对软件开发生命周期中的速度和质量产生不利影响的限制因素并确定其优先级的流程。DVSM 扩展了最初为精益生产实践设计的价值流映射流程。其重点关注在软件开发过程中创造和转移价值所需的步骤和团队。

## 数字孪生

真实世界系统的虚拟再现，如建筑物、工厂、工业设备或生产线。数字孪生支持预测性维护、远程监控和生产优化。

## 维度表

[星型架构](#)中的一种较小的表，其中包含事实表中定量数据的数据属性。维度表属性通常是文本字段或行为类似于文本的离散数字。这些属性通常用于查询约束、筛选和结果集标注。

## 灾难

阻止工作负载或系统在其主要部署位置实现其业务目标的事件。这些事件可能是自然灾害、技术故障或人为操作的结果，例如无意的配置错误或恶意软件攻击。

## 灾难恢复 ( DR )

您用来最大程度地减少由[灾难](#)造成的停机时间和数据丢失的策略和流程。有关更多信息，请参阅 Well-Architected Framework AWS work 中的“[工作负载灾难恢复：云端 AWS 恢复](#)”。

## DML

请参阅[数据库操作语言](#)。

## 领域驱动设计

一种开发复杂软件系统的方法，通过将其组件连接到每个组件所服务的不断发展的领域或核心业务目标。Eric Evans 在其著作[领域驱动设计：软件核心复杂性应对之道](#) ( Boston: Addison-Wesley Professional, 2003 ) 中介绍了这一概念。有关如何将领域驱动设计与 strangler fig 模式结合使用的信息，请参阅[使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET \( ASMX \) Web 服务现代化](#)。

## DR

请参阅[灾难恢复](#)。

## 偏差检测

跟踪与基准配置的偏差。例如，您可以使用 AWS CloudFormation 来[检测系统资源中的偏差](#)，也可以使用 AWS Control Tower 来[检测着陆区中可能影响监管要求合规性的变化](#)。

## DVSM

请参阅[开发价值流映射](#)。

## E

### EDA

请参阅[探索性数据分析](#)。

### EDI

请参阅[电子数据交换](#)。

## 边缘计算

该技术可提高位于 IoT 网络边缘的智能设备的计算能力。与[云计算](#)比较时，边缘计算可以减少通信延迟并缩短响应时间。

## 电子数据交换 ( EDI )

组织之间业务文件的自动交换。有关更多信息，请参阅[什么是电子数据交换](#)。

## 加密

一种将人类可读的纯文本数据转换为加密文字的计算流程。

## 加密密钥

由加密算法生成的随机位的加密字符串。密钥的长度可能有所不同，而且每个密钥都设计为不可预测且唯一。

## 字节顺序

字节在计算机内存中的存储顺序。大端序系统先存储最高有效字节。小端序系统先存储最低有效字节。

## 端点

请参阅[服务端点](#)。

## 端点服务

一种可以在虚拟私有云 ( VPC ) 中托管，与其他用户共享的服务。您可以使用其他 AWS 账户 或 AWS Identity and Access Management (IAM) 委托人创建终端节点服务，AWS PrivateLink 并向其授予权限。这些账户或主体可通过创建接口 VPC 端点来私密地连接到您的端点服务。有关更多信息，请参阅 Amazon Virtual Private Cloud ( Amazon VPC ) 文档中的[创建端点服务](#)。

## 企业资源规划 ( ERP )

一种自动化和管理企业关键业务流程 ( 例如会计、[MES](#) 和项目管理 ) 的系统。

## 信封加密

用另一个加密密钥对加密密钥进行加密的过程。有关更多信息，请参阅 AWS Key Management Service (AWS KMS) 文档中的[信封加密](#)。

## 环境

正在运行的应用程序的实例。以下是云计算中常见的环境类型：

- 开发环境 — 正在运行的应用程序的实例，只有负责维护应用程序的核心团队才能使用。开发环境用于测试更改，然后再将其提升到上层环境。这类环境有时称为测试环境。
- 下层环境 — 应用程序的所有开发环境，比如用于初始构建和测试的环境。

- 生产环境 — 最终用户可以访问的正在运行的应用程序的实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上层环境 — 除核心开发团队以外的用户可以访问的所有环境。这可能包括生产环境、预生产环境和用户验收测试环境。

## epic

在敏捷方法学中，有助于组织工作和确定优先级的功能类别。epics 提供了对需求和实施任务的总体描述。例如，AWS CAF 安全史诗包括身份和访问管理、侦探控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中 epics 的更多信息，请参阅[计划实施指南](#)。

## ERP

请参阅[企业资源规划](#)。

## 探索性数据分析 (EDA)

分析数据集以了解其主要特征的过程。您收集或汇总数据，并进行初步调查，以发现模式、检测异常并检查假定情况。EDA 通过计算汇总统计数据 and 创建数据可视化得以执行。

# F

## 事实表

[星型架构](#)中的中心表。它存储有关业务运营的定量数据。通常，事实表包含两种类型的列：包含度量的列和包含维度表外键的列。

## 快速失效机制

一种使用频繁且增量式的测试来缩短开发生命周期的理念。这是敏捷方法的关键部分。

## 故障隔离边界

在中 AWS Cloud，诸如可用区 AWS 区域、控制平面或数据平面之类的边界，它限制了故障的影响并有助于提高工作负载的弹性。有关更多信息，请参阅[AWS 故障隔离边界](#)。

## 功能分支

请参阅[分支](#)。

## 特征

您用来进行预测的输入数据。例如，在制造环境中，特征可能是定期从生产线捕获的图像。

## 特征重要性

特征对于模型预测的重要性。这通常表示为数值分数，可以通过各种技术进行计算，例如 Shapley 加法解释 ( SHAP ) 和积分梯度。有关更多信息，请参阅使用[机器学习模型的可解释性 AWS](#)。

## 功能转换

为 ML 流程优化数据，包括使用其他来源丰富数据、扩展值或从单个数据字段中提取多组信息。这使得 ML 模型能从数据中获益。例如，如果您将“2021-05-27 00:15:37”日期分解为“2021”、“五月”、“星期四”和“15”，则可以帮助学习与不同数据成分相关的算法学习精细模式。

## 少样本提示

在要求 [LLM](#) 执行类似任务之前，先向其提供少量示例，以演示任务和预期输出。此技术是上下文内学习的一种应用，其中模型可以从提示中嵌入的示例 ( 样本 ) 中学习。对于需要特定格式、推理或领域知识的任务，少样本提示可能非常有效。另请参阅[零样本提示](#)。

## FGAC

请参阅[精细访问控制](#)。

### 精细访问控制 ( FGAC )

使用多个条件允许或拒绝访问请求。

## 快闪迁移

一种数据库迁移方法，通过[更改数据捕获](#)使用连续数据复制，在极短的时间内迁移数据，而非使用分阶段方法。目标是将停机时间降至最低。

## FM

请参阅[基础模型](#)。

### 基础模型 ( FM )

一个大型深度学习神经网络，一直在广义和未标记数据的大量数据集上进行训练。FMs 能够执行各种各样的一般任务，例如理解语言、生成文本和图像以及用自然语言进行对话。有关更多信息，请参阅[什么是基础模型](#)。

## G

### 生成式人工智能

[AI](#) 模型的一个子集，这些模型已经过大量数据训练，可以使用简单的文本提示来创建新的内容和构件，例如图像、视频、文本和音频。有关更多信息，请参阅[什么是生成式人工智能](#)。

## 地理阻止

请参阅[地理限制](#)。

### 地理限制 ( 地理阻止 )

在 Amazon 中 CloudFront，一种阻止特定国家/地区的用户访问内容分发的选项。您可以使用允许列表或阻止列表来指定已批准和已禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档[中的限制内容的地理分布](#)。

### GitFlow 工作流程

一种方法，在这种方法中，下层和上层环境在源代码存储库中使用不同的分支。Gitflow 工作流程被认为是传统的工作流程，而[基于中继的工作流程](#)则是现代的、首选的方法。

### 黄金映像

系统或软件的快照，用作部署该系统或软件的新实例的模板。例如，在制造业中，黄金映像可用于在多个设备上预调配软件，并有助于提高设备制造操作的速度、可扩展性和生产效率。

### 全新策略

在新环境中缺少现有基础设施。在对系统架构采用全新策略时，您可以选择所有新技术，而不受对现有基础设施 ( 也称为[棕地](#) ) 兼容性的限制。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

### 防护机制

帮助管理各组织单位的资源、策略和合规性的高级规则 (OUs)。预防性防护机制会执行策略以确保符合合规性标准。它们是使用服务控制策略和 IAM 权限边界实现的。侦测性护栏会检测策略违规和合规性问题，并生成提醒以进行修复。它们通过使用 AWS Config、Amazon、AWS Security Hub CSPM GuardDuty AWS Trusted Advisor、Amazon Inspector 和自定义 AWS Lambda 支票来实现。

## H

### HA

请参阅[高可用性](#)。

### 异构数据库迁移

将源数据库迁移到使用不同数据库引擎的目标数据库 ( 例如，从 Oracle 迁移到 Amazon Aurora )。异构迁移通常是重新架构工作的一部分，而转换架构可能是一项复杂的任务。[AWS 提供了 AWS SCT](#) 来帮助实现架构转换。

## 高可用性 ( HA )

在遇到挑战或灾难时，工作负载无需干预即可连续运行的能力。HA 系统旨在自动进行故障转移、持续提供良好性能，并以最小的性能影响处理不同负载和故障。

## 历史数据库现代化

一种用于实现运营技术 ( OT ) 系统现代化和升级以更好满足制造业需求的方法。历史数据库是一种用于收集和存储工厂中各种来源数据的数据库。

## 保留数据

从用于训练[机器学习](#)模型的数据集中保留的一部分标注的历史数据。通过将模型预测与保留数据进行比较，您可以使用保留数据来评估模型性能。

## 同构数据库迁移

将源数据库迁移到共享同一数据库引擎的目标数据库 ( 例如，从 Microsoft SQL Server 迁移到 Amazon RDS for SQL Server )。同构迁移通常是更换主机或更换平台工作的一部分。您可以使用本机数据库实用程序来迁移架构。

## 热数据

经常访问的数据，例如实时数据或近期的转化数据。这些数据通常需要高性能存储层或存储类别才能提供快速的查询响应。

## 修补程序

针对生产环境中关键问题的紧急修复。由于其紧迫性，修补程序通常是在典型的 DevOps 发布工作流程之外进行的。

## hypercure 周期

割接之后，迁移团队立即管理和监控云中迁移的应用程序以解决任何问题的时间段。通常，这个周期持续 1-4 天。在 hypercure 周期结束时，迁移团队通常会将应用程序的责任移交给云运营团队。

# 我

## laC

请参阅[基础设施即代码](#)。

## 基于身份的策略

附加到一个或多个 IAM 委托人的策略，用于定义他们在 AWS Cloud 环境中的权限。

## 空闲应用程序

90 天内平均 CPU 和内存使用率在 5% 到 20% 之间的应用程序。在迁移项目中，通常会停用这些应用程序或将其保留在本地。

## IloT

请参阅[工业物联网](#)。

## 不可变基础设施

一种模型，可为生产工作负载部署新的基础设施，而不是更新、修补或修改现有基础设施。不可变基础设施本质上比[可变基础设施](#)更一致、更可靠、更可预测。有关更多信息，请参阅 AWS Well-Architected Framework 中的[使用不可变基础设施进行部署](#)最佳实践。

## 入站 ( 入口 ) VPC

在 AWS 多账户架构中，一种接受、检查和路由来自应用程序外部的网络连接的 VPC。[AWS 安全参考架构](#)建议设置您的网络帐户，包括入站、出站和检查，VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

## 增量迁移

一种割接策略，在这种策略中，您可以将应用程序分成小部分进行迁移，而不是一次性完整割接。例如，您最初可能只将几个微服务或用户迁移到新系统。在确认一切正常后，您可以逐步迁移其他微服务或用户，直到停用遗留系统。这种策略降低了大规模迁移带来的风险。

## 工业 4.0

该术语由 [Klaus Schwab](#) 在 2016 年提出，指的是通过连接、实时数据、自动化、分析和 AI/ML 的进步来实现制造流程的现代化。

## 基础设施

应用程序环境中包含的所有资源和资产。

## 基础设施即代码 ( IaC )

通过一组配置文件预调配和管理应用程序基础设施的过程。IaC 旨在帮助您集中管理基础设施、实现资源标准化和快速扩展，使新环境具有可重复性、可靠性和一致性。

## 工业物联网 (IloT)

在工业领域使用联网的传感器和设备，例如制造业、能源、汽车、医疗保健、生命科学和农业。有关更多信息，请参阅[制定工业物联网 \(IloT\) 数字化转型战略](#)。

## 检查 VPC

在 AWS 多账户架构中，一种集中式 VPC，用于管理对 VPCs（相同或不同 AWS 区域）、互联网和本地网络之间的网络流量的检查。[AWS 安全参考架构](#)建议设置您的网络帐户，包括入站、出站和检查，VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

## 物联网 ( IoT )

由带有嵌入式传感器或处理器的连接物理对象组成的网络，这些传感器或处理器通过互联网或本地通信网络与其他设备和系统进行通信。有关更多信息，请参阅[什么是 IoT ?](#)

## 可解释性

它是机器学习模型的一种特征，描述了人类可以理解模型的预测如何取决于其输入的程度。有关更多信息，请参阅使用[机器学习模型的可解释性 AWS](#)。

## 物联网

请参阅[物联网](#)。

## IT 信息库 ( ITIL )

提供 IT 服务并使这些服务符合业务要求的一套最佳实践。ITIL 是 ITSM 的基础。

## IT 服务管理 ( ITSM )

为组织设计、实施、管理和支持 IT 服务的相关活动。有关将云运营与 ITSM 工具集成的信息，请参阅[运营集成指南](#)。

## ITIL

请参阅[IT 信息库](#)。

## ITSM

请参阅[IT 服务管理](#)。

## L

## 基于标签的访问控制 ( LBAC )

强制访问控制 ( MAC ) 的一种实施方式，其中明确为用户和数据本身分配了安全标签值。用户安全标签和数据安全标签之间的交集决定了用户可以看到哪些行和列。

## 登录区

landing zone 是一个架构精良的多账户 AWS 环境，具有可扩展性和安全性。这是一个起点，您的组织可以从这里放心地在安全和基础设施环境中快速启动和部署工作负载和应用程序。有关登录区的更多信息，请参阅[设置安全且可扩展的多账户 AWS 环境](#)。

## 大语言模型 ( LLM )

一种基于大量数据进行预训练的深度学习 [AI](#) 模型。LLM 可以执行多项任务，例如回答问题、总结文档、将文本翻译成其他语言以及完成句子。有关更多信息，请参阅[什么是 LLMs](#)。

## 大规模迁移

迁移 300 台或更多服务器。

## LBAC

请参阅[基于标签的访问控制](#)。

## 最低权限

授予执行任务所需的最低权限的最佳安全实践。有关更多信息，请参阅 IAM 文档中的[应用最低权限许可](#)。

## 直接迁移

请参阅 [7 R](#)。

## 小端序系统

一个先存储最低有效字节的系统。另请参阅[字节顺序](#)。

## LLM

请参阅[大型语言模型](#)。

## 下层环境

请参阅[环境](#)。

# M

## 机器学习 ( ML )

一种使用算法和技术进行模式识别和学习的人工智能。ML 对记录的数据 ( 例如物联网 ( IoT ) 数据 ) 进行分析和学习，以生成基于模式的统计模型。有关更多信息，请参阅[机器学习](#)。

## 主分支

请参阅[分支](#)。

## 恶意软件

旨在危害计算机安全或隐私的软件。恶意软件可能会破坏计算机系统、泄露敏感信息或获得未经授权的访问权限。恶意软件的示例包括病毒、蠕虫、勒索软件、木马、间谍软件和键盘记录器。

## 托管式服务

AWS 服务 它 AWS 运行基础设施层、操作系统和平台，您可以访问端点来存储和检索数据。Amazon Simple Storage Service ( Amazon S3 ) 和 Amazon DynamoDB 就是托管服务的示例。这些服务也称为抽象服务。

## 制造执行系统 ( MES )

一种软件系统，用于跟踪、监控、记录和控制将原材料转化为成品的生产过程。

## MAP

请参阅[迁移加速计划](#)。

## 机制

一个完整的过程，您可以在其中创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运作过程中自我强化和改善的循环。有关更多信息，请参阅在 Well-Architect AWS ed 框架中[构建机制](#)。

## 成员账户

AWS 账户 除属于组织中的管理账户之外的所有账户 AWS Organizations。一个账户一次只能是一个组织的成员。

## MES

请参阅[制造执行系统](#)。

## 消息队列遥测传输 ( MQTT )

[一种基于发布/订阅模式的轻量级 machine-to-machine \(M2M\) 通信协议，适用于资源受限的物联网设备。](#)

## 微服务

一种小型的独立服务，通过明确的定义进行通信 APIs ，通常由小型的独立团队拥有。例如，保险系统可能包括映射到业务能力（如销售或营销）或子域（如购买、理赔或分析）的微服务。微服务

的好处包括敏捷、灵活扩展、易于部署、可重复使用的代码和恢复能力。有关更多信息，请参阅[使用 AWS 无服务器服务集成微服务](#)。

## 微服务架构

一种使用独立组件构建应用程序的方法，这些组件将每个应用程序进程作为微服务运行。这些微服务使用轻量级通过定义明确的接口进行通信。APIs 该架构中的每个微服务都可以更新、部署和扩展，以满足对应用程序特定功能的需求。有关更多信息，请参阅[在上实现微服务](#)。AWS

## 迁移加速计划 ( MAP )

AWS 该计划提供咨询支持、培训和服务，以帮助组织为迁移到云奠定坚实的运营基础，并帮助抵消迁移的初始成本。MAP 提供了一种以系统的方式执行遗留迁移的迁移方法，以及一套用于自动执行和加速常见迁移场景的工具。

## 大规模迁移

将大部分应用程序组合分波迁移到云中的过程，在每一波中以更快的速度迁移更多应用程序。本阶段使用从早期阶段获得的最佳实践和经验教训，实施由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付简化工作负载的迁移。这是[AWS 迁移策略](#)的第三阶段。

## 迁移工厂

跨职能团队，通过自动化、敏捷的方法简化工作负载迁移。迁移工厂团队通常包括运营、业务分析师和所有者、迁移工程师、开发人员和冲刺 DevOps 领域的专业人员。20% 到 50% 的企业应用程序组合由可通过工厂方法优化的重复模式组成。有关更多信息，请参阅本内容集中[有关迁移工厂的讨论](#)和[云迁移工厂指南](#)。

## 迁移元数据

有关完成迁移所需的应用程序和服务器器的信息。每种迁移模式都需要一套不同的迁移元数据。迁移元数据的示例包括目标子网、安全组和 AWS 账户。

## 迁移模式

一种可重复的迁移任务，详细列出了迁移策略、迁移目标以及所使用的迁移应用程序或服务。示例：使用 AWS 应用程序迁移服务重新托管向 Amazon EC2 的迁移。

## 迁移组合评测 ( MPA )

一种在线工具，提供了用于验证迁移到 AWS Cloud 的业务案例的信息。MPA 提供了详细的组合评测（服务器规模调整、定价、TCO 比较、迁移成本分析）以及迁移计划（应用程序数据分析和数据收集、应用程序分组、迁移优先级排序和波次规划）。所有 AWS 顾问和 APN 合作伙伴顾问均可免费使用[MPA 工具](#)（需要登录）。

## 迁移准备情况评测 ( MRA )

使用 AWS CAF 深入了解组织的云就绪状态、确定优势和劣势以及制定行动计划以缩小已发现差距的过程。有关更多信息，请参阅[迁移准备指南](#)。MRA 是 [AWS 迁移策略](#) 的第一阶段。

## 迁移策略

将工作负载迁移到 AWS Cloud 的方法。有关更多信息，请参见术语表中的 [7 R](#) 词条，以及[动员您的组织以加快大规模迁移](#)。

## ML

请参阅[机器学习](#)。

## 现代化

将过时的（原有的或单体）应用程序及其基础设施转变为云中敏捷、弹性和高度可用的系统，以降低成本、提高效率和利用创新。有关更多信息，请参阅[在 AWS Cloud 中实现应用程序现代化的策略](#)。

## 现代化准备情况评估

一种评估方式，有助于确定组织应用程序的现代化准备情况；确定收益、风险和依赖关系；确定组织能够在多大程度上支持这些应用程序的未来状态。评估结果是目标架构的蓝图、详细说明现代化进程发展阶段和里程碑的路线图以及解决已发现差距的行动计划。有关更多信息，请参阅[在 AWS Cloud 中评估应用程序的现代化准备情况](#)。

## 单体应用程序 ( 单体式 )

作为具有紧密耦合进程的单个服务运行的应用程序。单体应用程序有几个缺点。如果某个应用程序功能的需求激增，则必须扩展整个架构。随着代码库的增长，添加或改进单体应用程序的功能也会变得更加复杂。若要解决这些问题，可以使用微服务架构。有关更多信息，请参阅[将单体分解为微服务](#)。

## MPA

请参阅[迁移组合评测](#)。

## MQTT

请参阅[消息队列遥测传输](#)。

## 多分类器

一种帮助为多个类别生成预测（预测两个以上结果之一）的过程。例如，ML 模型可能会询问“这个产品是书、汽车还是手机？”或“此客户最感兴趣什么类别的产品？”

## 可变基础设施

一种用于更新和修改生产工作负载的现有基础设施的模型。为了提高一致性、可靠性和可预测性，Well-Architect AWS ed Framework 建议使用[不可变基础设施](#)作为最佳实践。

## O

### OAC

请参阅[来源访问控制](#)。

### OAI

请参阅[来源访问身份](#)。

### OCM

请参阅[组织变革管理](#)。

## 离线迁移

一种迁移方法，在这种方法中，源工作负载会在迁移过程中停止运行。这种方法会延长停机时间，通常用于小型非关键工作负载。

## OI

请参阅[运营集成](#)。

### OLA

请参阅[运营级别协议](#)。

## 在线迁移

一种迁移方法，在这种方法中，源工作负载无需离线即可复制到目标系统。在迁移过程中，连接工作负载的应用程序可以继续运行。这种方法的停机时间为零或最短，通常用于关键生产工作负载。

### OPC-UA

请参阅[开放流程通信 – 统一架构](#)。

## 开放流程通信 – 统一架构 ( OPC-UA )

一种用于工业自动化的 machine-to-machine ( M2M ) 通信协议。OPC-UA 提供了一个包含数据加密、身份验证和授权方案的互操作性标准。

## 运营级别协议 ( OLA )

一项协议，阐明了 IT 职能部门承诺相互交付的内容，以支持服务水平协议 ( SLA )。

## 运营准备情况审查 ( ORR )

一份问题核对清单和关联的最佳实践，可帮助您了解、评估、预防或缩小事件和可能的故障的范围。有关更多信息，请参阅 [AWS Well-Architected Framework 中的运营准备情况审查 \( ORR \)](#)。

## 运营技术 ( OT )

与物理环境配合使用以控制工业运营、设备和基础设施的硬件和软件系统。在制造业中，OT 和信息技术 ( IT ) 系统的集成是[工业 4.0](#) 转型的关键重点。

## 运营整合 ( OI )

在云中实现运营现代化的过程，包括就绪计划、自动化和集成。有关更多信息，请参阅[运营整合指南](#)。

## 组织跟踪

由 AWS CloudTrail 此创建的跟踪记录组织 AWS 账户 中所有人的所有事件 AWS Organizations。该跟踪是在每个 AWS 账户 中创建的，属于组织的一部分，并跟踪每个账户的活动。有关更多信息，请参阅 CloudTrail 文档中的[为组织创建跟踪](#)。

## 组织变革管理 ( OCM )

一个从人员、文化和领导力角度管理重大、颠覆性业务转型的框架。OCM 通过加快变革采用、解决过渡问题以及推动文化和组织变革，帮助组织为新系统和战略做好准备和过渡。在 AWS 迁移策略中，该框架被称为人员加速，因为云采用项目需要变更的速度。有关更多信息，请参阅 [OCM 指南](#)。

## 来源访问控制 ( OAC )

在中 CloudFront，一个增强的选项，用于限制访问以保护您的亚马逊简单存储服务 (Amazon S3) 内容。OAC 全部支持所有 S3 存储桶 AWS 区域、使用 AWS KMS (SSE-KMS) 进行服务器端加密，以及对 S3 存储桶的动态PUT和DELETE请求。

## 来源访问身份 ( OAI )

在中 CloudFront，一个用于限制访问权限以保护您的 Amazon S3 内容的选项。当您使用 OAI 时，CloudFront 会创建一个 Amazon S3 可以对其进行身份验证的委托人。经过身份验证的委托人只能通过特定 CloudFront 分配访问 S3 存储桶中的内容。另请参阅 [OAC](#)，其中提供了更精细和增强的访问控制。

## ORR

请参阅[运营准备情况审查](#)。

## OT

请参阅[运营技术](#)。

## 出站 ( 出口 ) VPC

在 AWS 多账户架构中，一种处理从应用程序内部启动的网络连接的 VPC。[AWS 安全参考架构](#)建议设置您的网络帐户，包括入站、出站和检查，VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

## P

### 权限边界

附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可以拥有的最大权限。有关更多信息，请参阅 IAM 文档中的[权限边界](#)。

### 个人身份信息 ( PII )

直接查看其他相关数据或与之配对时可用于合理推断个人身份的信息。PII 的示例包括姓名、地址和联系信息。

## PII

请参阅[个人身份信息](#)。

## playbook

一套预定义的步骤，用于捕获与迁移相关的工作，例如在云中交付核心运营功能。playbook 可以采用脚本、自动化运行手册的形式，也可以是操作现代化环境所需的流程或步骤的摘要。

## PLC

请参阅[可编程逻辑控制器](#)。

## PLM

请参阅[产品生命周期管理](#)。

## policy

一个对象，可以定义权限（请参阅[基于身份的策略](#)）、指定访问条件（请参阅[基于资源的策略](#)）或定义 AWS Organizations 的组织中所有账户的最大权限（请参阅[服务控制策略](#)）。

## 多语言持久性

根据数据访问模式和其他要求，独立选择微服务的数据存储技术。如果您的微服务采用相同的数据存储技术，它们可能会遇到实现难题或性能不佳。如果微服务使用最适合其需求的数据存储，则可以更轻松地实现微服务，并获得更好的性能和可扩展性。

## 组合评测

一个发现、分析和确定应用程序组合优先级以规划迁移的过程。有关更多信息，请参阅[评估迁移准备情况](#)。

## 谓词

返回 true 或 false 的查询条件，通常位于 WHERE 子句中。

## 谓词下推

一种数据库查询优化技术，可在传输之前筛选查询中的数据。这将减少从关系数据库检索和处理的数据量，并提高查询性能。

## 预防性控制

一种安全控制，旨在防止事件发生。这些控制是第一道防线，帮助防止未经授权的访问或对网络的意外更改。有关更多信息，请参阅在 AWS 上实施安全控制中的[预防性控制](#)。

## 主体

中 AWS 可以执行操作和访问资源的实体。此实体通常是 IAM 角色的根用户或用户。AWS 账户有关更多信息，请参阅 IAM 文档中的[角色术语和概念](#)中的主体。

## 隐私设计

一种在整个开发过程中都考虑隐私的系统工程方法。

## 私有托管区

一个容器，其中包含有关您希望 Amazon Route 53 如何响应针对一个或多个 VPCs 域名及其子域名的 DNS 查询的信息。有关更多信息，请参阅 Route 53 文档中的[私有托管区的使用](#)。

## 主动控制

一种[安全控制](#)，旨在防止部署不合规资源。这些控制会在资源预置之前对其进行扫描。如果资源与控制不兼容，则不会预置它。有关更多信息，请参阅 AWS Control Tower 文档中的[控制参考指南](#)，并参见在上实施安全[控制中的主动控制](#) AWS。

## 产品生命周期管理 ( PLM )

对产品在其整个生命周期内的数据和流程的管理，从设计、开发和发布，到增长和成熟，再到衰退和淘汰。

### 生产环境

请参阅[环境](#)。

## 可编程逻辑控制器 ( PLC )

在制造业中，一种高度可靠、适应性强的计算机，用于监控机器并实现制造过程自动化。

### 提示串接

使用一个 [LLM](#) 提示的输出作为下一个提示的输入，以生成更好的响应。该技术用于将复杂的任务分解为子任务，或者迭代地完善或扩展初步响应。它有助于提高模型响应的准确性和相关性，并允许获得更精细的个性化结果。

### 假名化

用占位符值替换数据集中个人标识符的过程。假名化可以帮助保护个人隐私。假名化数据仍被视为个人数据。

## publish/subscribe (pub/sub)

一种支持微服务间异步通信的模式，可提高可扩展性和响应能力。例如，在基于微服务的 [MES](#) 中，微服务可以将事件消息发布到其他微服务可以订阅的频道。系统可以在不更改发布服务的情况下添加新的微服务。

## Q

### 查询计划

一系列用于访问 SQL 关系数据库系统中的数据的步骤，类似于指令。

### 查询计划回归

当数据库服务优化程序选择的最佳计划不如数据库环境发生特定变化之前时。这可能是由统计数据、约束、环境设置、查询参数绑定更改和数据库引擎更新造成的。

# R

## RACI 矩阵

请参阅[责任、问责、咨询和知情 \( RACI \)](#)。

## RAG

请参阅[检索增强生成](#)。

## 勒索软件

一种恶意软件，旨在阻止对计算机系统或数据的访问，直到付款为止。

## RASCI 矩阵

请参阅[责任、问责、咨询和知情 \( RACI \)](#)。

## RCAC

请参阅[行列访问控制](#)。

## 只读副本

用于只读目的的数据库副本。您可以将查询路由到只读副本，以减轻主数据库的负载。

## 重新架构

请参阅 [7 R](#)。

## 恢复点目标 ( RPO )

自上一个数据恢复点以来可接受的最长时间。这决定了从上一个恢复点到服务中断之间可接受的数据丢失情况。

## 恢复时间目标 ( RTO )

服务中断和服务恢复之间可接受的最大延迟。

## 重构

请参阅 [7 R](#)。

## Region

地理区域内的 AWS 资源集合。每一个 AWS 区域 都相互隔离，彼此独立，以提供容错、稳定性和弹性。有关更多信息，请参阅[指定您的账户可以使用的 AWS 区域](#)。

## 回归

一种预测数值的 ML 技术。例如，要解决“这套房子的售价是多少？”的问题 ML 模型可以使用线性回归模型，根据房屋的已知事实（如建筑面积）来预测房屋的销售价格。

## 重新托管

请参阅 [7 R](#)。

## 版本

在部署过程中，推动生产环境变更的行为。

## 重新放置

请参阅 [7 R](#)。

## 更换平台

请参阅 [7 R](#)。

## 重新购买

请参阅 [7 R](#)。

## 韧性

应用程序抵御中断或从中断中恢复的能力。在 AWS Cloud 中规划韧性时，[高可用性](#)和[灾难恢复](#)是常见的考虑因素。有关更多信息，请参阅 [AWS Cloud 韧性](#)。

## 基于资源的策略

一种附加到资源的策略，例如 AmazonS3 存储桶、端点或加密密钥。此类策略指定了允许哪些主体访问、支持的操作以及必须满足的任何其他条件。

## 责任、问责、咨询和知情 ( RACI ) 矩阵

定义参与迁移活动和云运营的所有各方的角色和责任的矩阵。矩阵名称源自矩阵中定义的责任类型：负责 ( R )、问责 ( A )、咨询 ( C ) 和知情 ( I )。支持 ( S ) 类型是可选的。如果包括支持，则该矩阵称为 RASCI 矩阵，如果将其排除在外，则称为 RACI 矩阵。

## 响应性控制

一种安全控制，旨在推动对不良事件或偏离安全基线的情况进行修复。有关更多信息，请参阅在 AWS 上实施安全控制中的[响应性控制](#)。

## 保留

请参阅 [7 R](#)。

## 停用

请参阅 [7 R](#)。

## 检索增强生成 ( RAG )

一种[生成式人工智能](#)技术，其中 [LLM](#) 在生成响应之前引用其训练数据来源之外的权威数据来源。例如，RAG 模型可以对组织的知识库或自定义数据执行语义搜索。有关更多信息，请参阅[什么是 RAG](#)。

## 轮换

定期更新[密钥](#)以使攻击者更难访问凭证的过程。

## 行列访问控制 ( RCAC )

使用已定义访问规则的基本、灵活的 SQL 表达式。RCAC 由行权限和列掩码组成。

## RPO

请参阅[恢复点目标](#)。

## RTO

请参阅[恢复时间目标](#)。

## 运行手册

执行特定任务所需的一套手动或自动程序。它们通常是为了简化重复性操作或高错误率的程序而设计的。

# S

## SAML 2.0

许多身份提供商 (IdPs) 使用的开放标准。此功能支持联合单点登录 (SSO)，因此用户无需在 IAM 中为组织中的所有人创建用户即可登录 AWS 管理控制台 或调用 AWS API 操作。有关基于 SAML 2.0 的联合身份验证的更多信息，请参阅 IAM 文档中的[关于基于 SAML 2.0 的联合身份验证](#)。

## SCADA

请参阅[监督控制和数据采集](#)。

## SCP

请参阅[服务控制策略](#)。

## 机密密钥

在中 AWS Secrets Manager，您以加密形式存储的机密或受限信息，例如密码或用户凭证。它由密钥值及其元数据组成。密钥值可以是二进制、单个字符串或多个字符串。有关更多信息，请参阅 Secrets Manager 文档中的[什么是 Amazon Secrets Manager 密钥？](#)。

## 安全设计

一种在整个开发过程中都考虑安全的系统工程方法。

## 安全控制

一种技术或管理防护机制，可防止、检测或降低威胁行为体利用安全漏洞的能力。安全控制有以下四种类型：[预防性](#)、[检测性](#)、[响应性](#)和[主动性](#)。

## 安全固化

缩小攻击面，使其更能抵御攻击的过程。这可能包括删除不再需要的资源、实施授予最低权限的最佳安全实践或停用配置文件中不必要的功能等操作。

## 安全信息和事件管理 ( SIEM ) 系统

结合了安全信息管理 ( SIM ) 和安全事件管理 ( SEM ) 系统的工具和服务。SIEM 系统会收集、监控和分析来自服务器、网络、设备和其他来源的数据，以检测威胁和安全漏洞，并生成警报。

## 安全响应自动化

一种预定义的程序化操作，旨在自动响应或修复安全事件。这些自动化可作为[侦探或响应式](#)安全控制措施，帮助您实施 AWS 安全最佳实践。自动响应操作的示例包括修改 VPC 安全组、修补 Amazon EC2 实例或轮换凭证。

## 服务器端加密

由接收数据的人在目的地对数据 AWS 服务 进行加密。

## 服务控制策略 ( SCP )

一种策略，用于集中控制组织中所有账户的权限 AWS Organizations。SCPs 定义防护措施或限制管理员可以委托给用户或角色的操作。您可以使用 SCPs 允许列表或拒绝列表来指定允许或禁止哪些服务或操作。有关更多信息，请参阅 AWS Organizations 文档中的[服务控制策略](#)。

## 服务端点

的入口点的 URL AWS 服务。您可以使用端点，通过编程方式连接到目标服务。有关更多信息，请参阅 AWS 一般参考 中的[AWS 服务 端点](#)。

## 服务水平协议 ( SLA )

一份协议，阐明了 IT 团队承诺向客户交付的内容，比如服务正常运行时间和性能。

## 服务水平指示器 ( SLI )

对服务性能方面的衡量，例如错误率、可用性或吞吐量。

## 服务水平目标 ( SLO )

代表服务运行状况的目标指标，由[服务水平指示器](#)衡量。

## 责任共担模式

描述您在云安全与合规方面共同承担 AWS 的责任的模型。AWS 负责云的安全，而您则负责云中的安全。有关更多信息，请参阅[责任共担模式](#)。

## SIEM

请参阅[安全信息和事件管理系统](#)。

## 单点故障 ( SPOF )

应用程序的单个关键组件出现故障，可能会中断系统。

## SLA

请参阅[服务水平协议](#)。

## SLI

请参阅[服务水平指示器](#)。

## SLO

请参阅[服务水平目标](#)。

## split-and-seed 模型

一种扩展和加速现代化项目的模式。随着新功能和产品发布的定义，核心团队会拆分以创建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的工作效率，支持快速创新。有关更多信息，请参阅[在 AWS Cloud 中实现应用程序现代化的分阶段方法](#)。

## SPOF

请参阅[单点故障](#)。

## 星型架构

一种数据库组织结构，它使用一个大型事实表来存储事务数据或测量数据，并使用一个或多个较小的维度表来存储数据属性。此结构专为在[数据仓库](#)中使用或用于商业智能目的而设计。

## strangler fig 模式

一种通过逐步重写和替换系统功能直至可以停用原有的系统来实现单体系统现代化的方法。这种模式用无花果藤作为类比，这种藤蔓成长为一棵树，最终战胜并取代了宿主。该模式是由 [Martin Fowler](#) 提出的，作为重写单体系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅[使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET \( ASMX \) Web 服务现代化](#)。

## 子网

您的 VPC 内的一个 IP 地址范围。子网必须位于单个可用区中。

## 监督控制和数据采集 ( SCADA )

在制造业中，一种使用硬件和软件来监控实物资产和生产操作的系统。

## 对称加密

一种加密算法，它使用相同的密钥来加密和解密数据。

## 综合测试

以模拟用户交互的方式测试系统，以检测潜在问题或监控性能。您可以使用 [Amazon S CloudWatch ynthetic](#) 来创建这些测试。

## 系统提示

一种为 [LLM](#) 提供上下文、说明或准则以指导其行为的技术。系统提示有助于设置上下文并制定与用户交互的规则。

# T

## 标签

键值对，用作组织资源的元数据。AWS 标签有助于您管理、识别、组织、搜索和筛选 资源。有关更多信息，请参阅[标记您的 AWS 资源](#)。

## 目标变量

您在监督式 ML 中尝试预测的值。这也被称为结果变量。例如，在制造环境中，目标变量可能是产品缺陷。

## 任务列表

一种通过运行手册用于跟踪进度的工具。任务列表包含运行手册的概述和要完成的常规任务列表。对于每项常规任务，它包括预计所需时间、所有者和进度。

## 测试环境

请参阅[环境](#)。

## 训练

为您的 ML 模型提供学习数据。训练数据必须包含正确答案。学习算法在训练数据中查找将输入数据属性映射到目标（您希望预测的答案）的模式。然后输出捕获这些模式的 ML 模型。然后，您可以使用 ML 模型对不知道目标的新数据进行预测。

## 中转网关

一个网络传输中心，可用于将您的网络 VPCs 和本地网络互连。有关更多信息，请参阅 AWS Transit Gateway 文档中的[什么是公交网关](#)。

## 基于中继的工作流程

一种方法，开发人员在功能分支中本地构建和测试功能，然后将这些更改合并到主分支中。然后，按顺序将主分支构建到开发、预生产和生产环境。

## 可信访问权限

向您指定的服务授予权限，该服务可代表您在其账户中执行任务。AWS Organizations 当需要服务相关的角色时，受信任的服务会在每个账户中创建一个角色，为您执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的[AWS Organizations 与其他 AWS 服务一起使用](#)。

## 优化

更改训练过程的各个方面，以提高 ML 模型的准确性。例如，您可以通过生成标签集、添加标签，并在不同的设置下多次重复这些步骤来优化模型，从而训练 ML 模型。

## 双披萨团队

一个小 DevOps 团队，你可以用两个披萨来喂食。双披萨团队的规模可确保在软件开发过程中充分协作。

# U

## 不确定性

这一概念指的是不精确、不完整或未知的信息，这些信息可能会破坏预测式 ML 模型的可靠性。不确定性有两种类型：认知不确定性是由有限的、不完整的数据造成的，而偶然不确定性是由数据中固有的噪声和随机性导致的。

## 无差别任务

也称为繁重工作，即创建和运行应用程序所必需的工作，但不能为最终用户提供直接价值或竞争优势。无差别任务的示例包括采购、维护和容量规划。

## 上层环境

请参阅[环境](#)。

# V

## vacuum 操作

一种数据库维护操作，包括在增量更新后进行清理，以回收存储空间并提高性能。

## 版本控制

跟踪更改的过程和工具，例如存储库中源代码的更改。

## VPC 对等连接

两者之间的连接 VPCs，允许您使用私有 IP 地址路由流量。有关更多信息，请参阅 Amazon VPC 文档中的[什么是 VPC 对等连接](#)。

## 漏洞

损害系统安全的软件缺陷或硬件缺陷。

# W

## 热缓存

一种包含经常访问的当前相关数据的缓冲区缓存。数据库实例可以从缓冲区缓存读取，这比从主内存或磁盘读取要快。

## 暖数据

不常访问的数据。查询此类数据时，通常可以接受中速查询。

## 窗口函数

一种对与当前记录有某种关联的一组行执行计算的 SQL 函数。窗口函数对于处理任务很有用，例如计算移动平均值或根据当前行的相对位置访问行的值。

## 工作负载

一系列资源和代码，它们可以提供商业价值，如面向客户的应用程序或后端过程。

## 工作流

迁移项目中负责一组特定任务的职能小组。每个工作流都是独立的，但支持项目中的其他工作流。例如，组合工作流负责确定应用程序的优先级、波次规划和收集迁移元数据。组合工作流将这些资产交付给迁移工作流，然后迁移服务器和应用程序。

## WORM

请参阅[一次写入多次读取](#)。

## WQF

请参阅[AWS 工作负载资格鉴定框架](#)。

## 一次写入多次读取 ( WORM )

一种存储模型，可一次写入数据并防止数据被删除或修改。授权用户可以根据需要多次读取数据，但无法对其进行更改。此数据存储基础设施被认为[不可变](#)。

# Z

## 零日漏洞利用

一种利用[零日漏洞](#)的攻击，通常为恶意软件。

## 零日漏洞

生产系统中不可避免的缺陷或漏洞。威胁主体可能利用这种类型的漏洞攻击系统。开发人员经常因攻击而意识到该漏洞。

## 零样本提示

为[LLM](#)提供执行任务的说明，但没有可以帮助指导的示例（样本）。LLM 必须使用预先训练的知识来处理任务。零样本提示的有效性取决于任务的复杂性和提示的质量。另请参阅[少样本提示](#)。

## 僵尸应用程序

平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常会停用这些应用程序。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。