



在 AWS 上使用生成式 AI 加速软件开发生命周期

AWS 规范性指导

AWS 规范性指导：借助生成式 AI 在 AWS 上加速软件开发生命周期

版权所有 © 2025 Amazon Web Services, Inc. 及/或其关联公司。保留所有权利。

未经许可，不得以任何可能导致客户混淆的方式，或以任何贬损或诋毁 Amazon 的方式，将 Amazon 的商标和商业外观用于与任何非 Amazon 的产品或服务相关的用途。除 Amazon 拥有的商标外，所有其他商标均为其各自所有者的财产，这些所有者可能与 Amazon 有或没有隶属关系、关联关系，或由 Amazon 赞助。

目录

引言	1
目标	1
目标受众	2
开发经验	3
使用生成式 AI	4
5-I 框架	5
框架概述	6
与 SDLC 集成	8
基础能力	9
项目管理	15
需求管理	18
架构与设计	19
协作	20
DevSecOps	21
运营与维护	29
AI 助手	31
分析与洞察	33
知识管理	36
可扩展性	37
最佳实践	39
集成式工具链	39
DevSecOps 流水线	40
协作工具和实践	40
任务自动化	40
审查与迭代	41
项目管理实践	41
知识管理	42
可扩展性与定制化	42
优化	42
数据驱动的洞察	43
基于平台的方法	43
衡量成功	44
部署速度	45
代码质量	45

运营效率	46
团队生产力与满意度	46
业务影响	47
结论	48
资源	48
文档历史记录	50
术语表	51
	51
A	52
B	55
C	57
D	60
E	64
F	66
G	68
H	69
I	70
L	73
M	74
O	78
P	81
Q	83
R	84
S	87
T	91
U	92
V	93
W	93
Z	94

在 AWS 上使用生成式 AI 加速软件开发生命周期

Chetan Makvana, Amazon Web Services

2025 年 4 月（文档历史）

对高质量软件日益增长的需求，正推动各组织不断寻求加速其软件开发生命周期（SDLC）的方法。随着组织努力保持竞争力，在维持或提升产品质量的同时缩短上市时间至关重要。为应对这些挑战，软件开发体验必须演进，采用前沿技术、方法论与实践，以简化流程并赋能软件开发团队提升生产力与创造力。下一代开发体验的出现，标志着软件在构思、构建、测试与部署方式上的重大转变。它整合了多种能力——包括云原生开发、AI 驱动的自动化、高级项目管理、协作工具以及 DevSecOps——共同提升 SDLC 的效率与有效性。

在这场变革的最前沿，是生成式 AI 在软件工程领域的崛起。根据 Gartner 的数据，到 2027 年，40% 的平台工程团队将使用 AI 来增强 SDLC 的每个阶段，而 2023 年这一比例仅为 5%。该报告还指出，软件工程领导者如今必须为在更广泛、对开发过程至关重要的领域采用生成式 AI 做好准备。另一份报告中，麦肯锡研究显示，开发者速度指数更高的公司收入增长快 4–5 倍，股东回报高 60%，创新能力高 55%。通过在不限于代码生成的层面拥抱生成式 AI，组织可以在软件开发工作流中释放全新水平的效率、生产力与创新。这可以减少人工工作量，挖掘洞见，并增强人类专业能力。

目标

本策略文档概述了一个框架、基础能力、用例、最佳实践和成功指标，可帮助你借助生成式 AI 加速 SDLC。它描述了如何在所有开发阶段有效集成生成式 AI，以提升产品质量和效率。

本策略文档可帮助您和您的组织实现以下目标：

- 实施一个框架、基础能力、用例、最佳实践和成功指标，以借助生成式 AI 加速你的 SDLC。
- 在所有开发阶段有效集成生成式 AI，以提升产品质量、发布速度和开发效率。
- 通过引入前沿的 AI 技术、方法论和实践来简化流程并赋能开发团队，从而适应下一代软件开发。

目标受众

本战略文档面向希望通过将生成式 AI 应用于开发实践来加速软件开发生命周期的 IT 领导者、工程经理、首席技术官以及软件开发团队。

理解软件开发体验

软件开发体验涵盖了你的开发团队在整个软件开发生命周期（SDLC）中使用的环境、工具和流程。它包括集成开发环境（IDE）、协作平台、测试框架、知识管理系统、部署流水线等。

精心设计的开发体验能够简化 workflow、减少人工操作，并使团队专注于高价值任务，从而最终加速你的 SDLC。例如，通过无缝集成 IDE、版本控制系统和部署工具，你可以让开发人员以更高的速度和效率编写、测试并部署代码，相比之下，碎片化的工具链需要人工交接和频繁的上下文切换。类似地，集成一个强大的知识管理框架有助于团队轻松访问和共享组织知识、最佳实践和文档，从而提升整体生产力与问题解决能力。

软件开发体验会直接影响软件开发团队的整体绩效与成功。次优的体验可能导致以下问题：

- 生产力降低——低效的工具、复杂的工作流以及缺乏自动化会阻碍团队生产力，从而减缓功能与更新的交付。
- 技术债务增加——集成不佳的工具和临时性的流程可能导致技术债务，使得随着时间推移维护并扩展软件系统更具挑战。
- 创新能力减弱——当被手动、重复性任务拖累时，团队探索新技术并推动创新的能力会受到限制。
- 质量受损——碎片化的测试与部署流程会增加软件缺陷和漏洞的风险，从而对交付软件的整体质量产生负面影响。

通过投资打造良好设计的软件开发体验，你可以获得显著收益，例如更快的上市时间、更高的软件质量、更强的软件开发团队满意度，以及更高的业务敏捷性。

用生成式 AI 赋能软件开发体验

将生成式 AI 集成到软件开发生命周期 (SDLC) 中，代表着整个软件开发团队在构思、设计、实现和维护软件解决方案方式上的范式转变。生成式 AI 有潜力革新 SDLC 的每个阶段，包括项目管理、需求收集、设计、编码、测试、部署和维护。

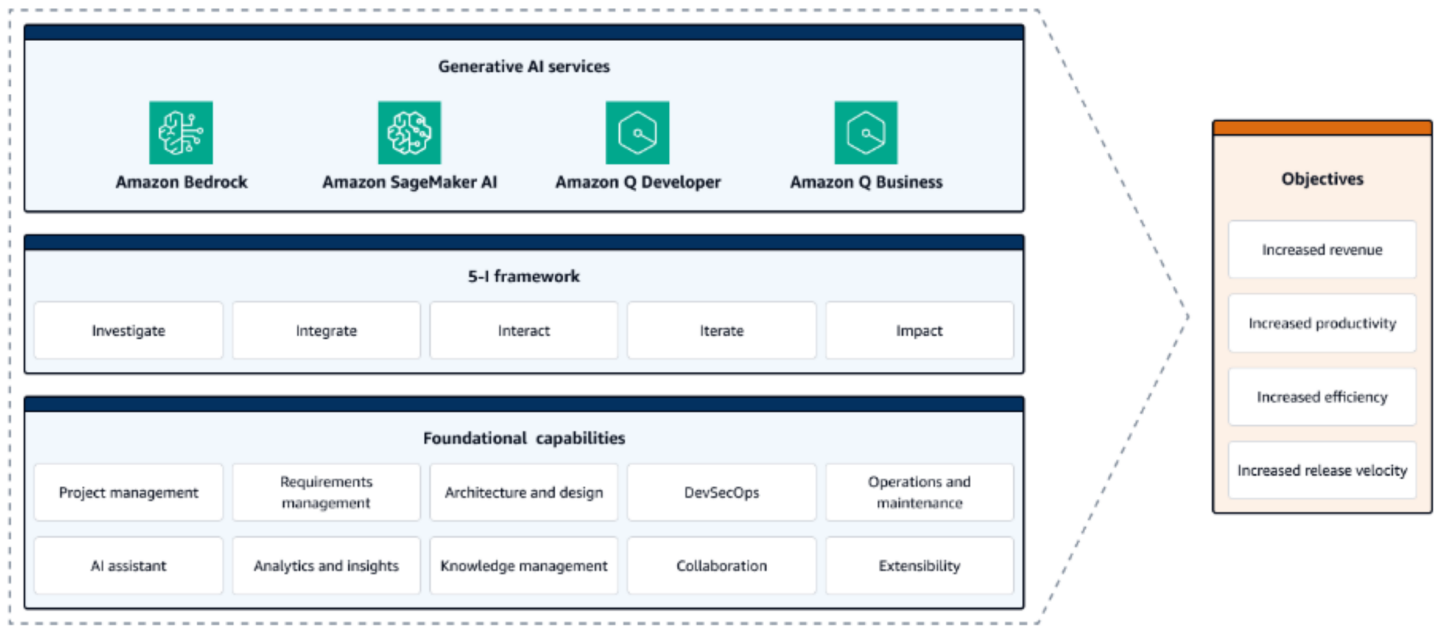
从本质上讲，由生成式 AI 驱动的开发体验充当整个软件开发团队的人工智能协作者，涵盖产品经理、设计师、解决方案架构师、开发人员、测试人员以及运维人员。它提供具备上下文感知的辅助，生成各类工件（如用户故事、设计原型、代码片段和测试用例），提供近乎实时的建议，甚至在问题出现之前就能预测潜在问题。这种 AI 增强的方法显著降低了团队成员的认知负荷，使他们能够专注于高层次的战略决策和复杂问题解决，而由生成式 AI 处理更为琐碎、重复的任务。

生成式 AI 还可作为知识放大器。它帮助团队成员从海量数据存储库中快速获取相关信息、最佳实践和模式。这能够有效地在组织内实现专业知识的普惠化。通过在整个开发工具链中无缝集成生成式 AI 能力，你可以为整个软件开发团队打造一个更直观、更高效、更具生产力的环境。这种增强的开发体验可加速 SDLC 并提升整体质量。它还能减少错误并促进创新，因为团队成员能够更快速地探索新想法和新方法。

要在组织中采用由生成式 AI 驱动的开发体验，请考虑以下关键要素：

- **5-I 框架**——由五个维度构成，5-I 框架为驾驭现代软件开发过程提供了一种全面的方法。它提供了一套结构化的方法论，帮助你在 SDLC 的所有阶段系统性地应用生成式 AI。
- **基础能力**——要在现代软件开发的各个维度充分发挥生成式 AI 的力量，你需要建立一套强健的基础能力。这些能力构成了 AI 驱动开发体验的支柱。这些能力可帮助你在整个 SDLC 中集成并使用生成式 AI。

5-I 框架与基础能力共同构成了一种重新构想软件开发体验的策略。这五个维度为应用生成式 AI 提供了战略框架，而基础能力则使你的组织能够为这种 AI 驱动的方法提供支持。AWS 服务（如 Amazon Bedrock、Amazon SageMaker AI、Amazon Q Developer 和 Amazon Q Business）提供了生成式 AI 能力和特性，你可以将其集成到你的软件开发体验中。



用于 AI 驱动软件开发体验的 5-I 框架

5-I 框架为软件开发团队提供了一种结构化方法，以便将生成式 AI 有效地集成到其开发实践中。它帮助你为在整个 SDLC 中使用生成式 AI 建立坚实的基础。它还帮助你建立合适的开发实践、工作流和思维方式，从而充分释放生成式 AI 的潜力。

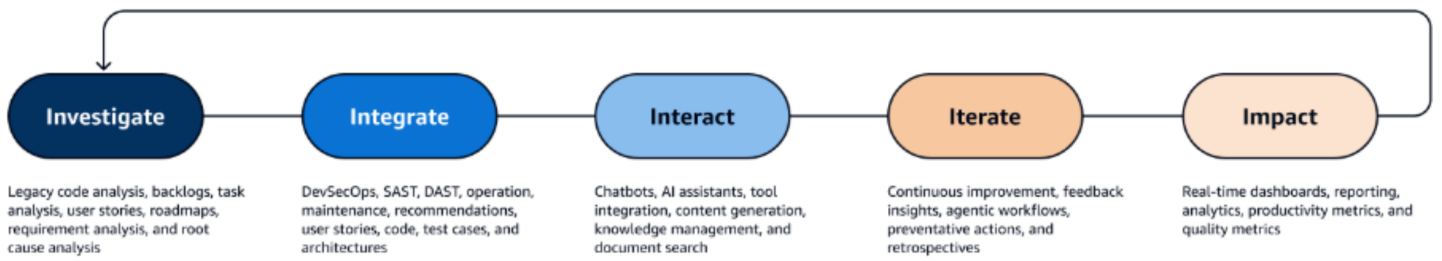
本节包含以下主题：

- 框架概述
- 与软件开发生命周期集成

框架概述

5-I 框架围绕五个关键维度构建：Investigate、Integrate、Interact、Iterate 和 Impact。每个维度都代表一个关键领域，在该领域中，生成式 AI 能显著增强软件开发过程。通过在这些维度上战略性地集成生成式 AI，该框架能够满足现代软件开发不断演进的需求。它可以降低认知负荷并放大创造潜力。它认识到，理想的开发体验不仅仅关乎工具——更在于打造一种环境，让 AI 在每个阶段都能无缝增强人类能力。

下图展示了 AI 驱动的软件开发的五个维度。对于每个维度，它都说明了你可以在何处集成生成式 AI，以推动效率与创新。



以下是该框架中的五个维度：

- **调研**——借助生成式 AI 强化软件开发流程中的每一项分析任务。使用生成式 AI 来理解需求、处理海量数据、识别模式，并生成可能超出人类能力或需要更长时间才能产出的洞察。这些洞察可帮助你做出更明智的决策，快速识别改进机会，并更高效地交付高质量软件。

生成式 AI 可以成为贯穿 SDLC 各类分析流程的智能伙伴。通过利用生成式 AI，你可以对关键领域进行深入分析，例如需求收集、遗留代码库审查以及产品待办事项优化。例如，产品负责人可以在创建用户故事之前，使用生成式 AI 分析用户旅程或需求。开发团队可以发现现有代码库中的低效之处并识别优化机会。DevOps 工程师可以运用根因分析快速诊断性能问题或安全漏洞，从而提升可靠性。

- **集成**——集成生成式 AI，以在整个 SDLC 中自动化广泛的任务和流程。这包括自动生成代码片段、测试用例、架构设计、用户故事和部署流水线。通过将这些通常需要手动完成的任务自动化，

团队可以将精力集中在更具战略性和创新性的工作上，从而加快产品上市速度并交付高质量应用。集成维度代表了软件开发中的范式转变，即 AI 成为开发过程不可或缺的一部分。它与您的软件开发团队并肩工作，以提升生产力、改进质量并推动创新。这将带来更快的上市速度。它促使您的软件开发团队定期评估其流程和工作流，在每一步都要问：“这能否实现自动化？”

- **交互**——使用由生成式 AI 驱动的助手，为你的团队在各种任务和问题上提供即时、具备上下文的支持。这些智能助手充当知识渊博的协作者，从海量信息库中汲取内容。它们可以回答编码问题、提供设计建议、解释标准操作流程，并帮助排查复杂问题。将这些 AI 助手集成到开发工作流中，能够提升生产力，并营造更具协作性、以解决问题为导向的环境。
- **迭代**——使用生成式 AI 在整个 SDLC 中实现快速、数据驱动的调整。你可以持续分析来自客户反馈、使用模式、市场趋势以及团队绩效指标等来源的数据，从而快速做出明智决策。这种适应性将你的软件开发从静态、预先定义的流程，优化为流动且响应迅速的方法。它会以多种方式体现出来，包括对待办事项列表的动态优先级排序、灵活的资源分配、自适应的测试策略、不断演进的文档，以及响应式的部署流程。例如，产品经理可以利用 AI 生成的洞察近乎实时地重新排序待办事项列表，整合新的客户需求和市场趋势。DevOps 工程师可以基于性能分析调整部署计划和基础设施配置，确保应用保持韧性并持续优化。开发团队可以将冲刺回顾中的反馈转化为下一次迭代的可执行改进，推动持续改进流程的文化。
- **影响**——应用生成式 AI 来评估你的软件开发流程的有效性和性能。通过使用 AI 驱动的分析与指标，你可以更深入地洞察开发效率、代码质量、用户参与度以及整体应用性能。这种数据驱动的方法有助于你做出明智决策、优化开发工作流，并持续提升应用的质量与用户体验。在评估软件团队生产力时，生成式 AI 会分析各种数据点，例如代码提交频率、问题解决时间、发布速度、功能交付率等。它还可以评估代码评审的质量、协作工具的有效性，以及不同开发实践对团队整体产出的影响。通过将这些指标与项目结果相关联，AI 能识别出人类分析师可能忽略的模式与趋势，并提供可执行的洞察以提升团队生产力。此外，生成式 AI 还可以

帮助你将团队绩效与行业标准或历史数据进行对标，并提供个性化的改进建议。它还可以预测开发过程中的潜在瓶颈或风险，以便你采取主动措施。

与软件开发生命周期集成

SDLC 由多个阶段组成，不同组织之间可能有所不同。通常，这些阶段包括：需求与规划、设计与架构、实现、测试、部署，以及运营与维护。

下表将 5-I 框架的各个维度映射到 SDLC 各阶段，并提供每个维度的集成级别。

框架 维度	需求 以及 规划	设计 和 架构 ure	实施 ation	测试 部署	t	运营 以及 维护 ce
调查 te	高	低	低	低	低	中
集成 中 中 高				中高		高
交互 高		高	高	中 中 高		
迭代 中 低			低	低	低	中
影响：高		中等偏高		低	高	高

集成程度从高到低不等。该映射揭示了每个维度的关键关注领域。例如，Investigate 在需求与规划阶段表现出高强度。Integrate 在实现、部署以及运维阶段表现出高强度。

通过使用该映射，你可以有效地确定工作的优先级。我们建议你先关注高强度，其次是中等强度，最后是低强度。确保采用一种平衡且富有成效的方法，通过生成式 AI 提升软件开发体验。

面向 AI 驱动的软件开发体验的基础能力

要成功实施由生成式 AI 驱动的软件开发体验，你需要建立一套覆盖组织内多类角色的基础能力。这些能力体现了你在 AI 驱动的软件开发背景下，有效部署资源、落实流程并实现预期成果的能力。通过培育这些能力，你将打造坚实的基础，帮助你在 SDLC 的各个阶段无缝集成生成式 AI。

AWS 提供了关键服务来帮助你实现这些能力。例如，Amazon Q Developer 作为 AI 驱动的助手，帮助加速软件开发。Amazon Q Business 帮助你快速获得与紧迫问题相关的答案、解决问题并生成内容。它还可以通过集成与软件开发相关的工具来代表你执行操作。Amazon Bedrock 提供对基础模型的访问，以及广泛的能力，用于定制特定的开发工作流和需求。

通过借助 AWS 服务培养这些能力，你可以打造坚实的基础，帮助你在 SDLC 的各个阶段无缝集成生成式 AI。

以下是你应重点关注的基础能力：

- 项目管理
- 需求管理
- 架构与设计
- 协作
- DevSecOps
- 运营与维护
- AI 助手
- 分析与洞察
- 知识管理
- 可扩展性

每项基础能力都与框架维度以及 SDLC 的不同阶段相集成。这种集成有助于你在整个过程中有效地使用 AI 能力。

软件开发过程。它在每一步都提升效率、质量和创新。这些基础能力、框架与 SDLC 各阶段之间的协同作用，构建了一个用于 AI 驱动软件开发的全面生态系统。这有助于你充分发挥生成式 AI 的全部潜力，推动持续改进，加速开发周期，并交付高质量的软件产品。

下表展示了基础能力及其子能力如何映射到框架维度和 SDLC 阶段。

能力： 子能力 lity	调研 集成 交互 迭代				影响
项目 管理 ：问题 管理	需求 以及 规划	无	无	无	无
项目 管理 ：冲刺 和任务 管理	需求 以及 规划	需求 以及 规划	无	无	无
项目 管理 t：产品 待办事项列表 管理	需求 以及 规划	无	无	需求 以及 规划	无
项目 管理 ：用户 故事 映射	需求 以及 规划	无	无	无	无

能力： subcapability	调研 集成 交互 迭代				影响
项目管理： 报告和分析	需求以及规划	无	无	无	需求以及规划
项目管理： 产品路线图管理	需求以及规划	无	需求以及规划	无	无
项目管理： 反馈循环	无	无	无	需求以及规划	无
项目管理： 回顾会议	无	无	无	需求以及规划	无
需求管理	需求以及规划	需求以及规划	无	无	无
架构以及设计： 解决方案设计	设计与架构	设计与架构	无	无	无

能力： 子能力 力	调查 集成 交互 迭代				影响
Collabora tion： 文档化 tion 管理	所有 SDLC 阶段	无	所有 SDLC 阶段	无	无
Collabora tion： 知识 共享	所有 SDLC 阶段	无	所有 SDLC 阶段	无	无
Collabora tion：项目 资产 管理	无	所有 SDLC 阶段	所有 SDLC 阶段	无	无
DevSecOps： CI/CD	测试， 部署	实施 ation， 测试， 部署	部署 无		无
DevSecOps ： DevOps 安全	实施 ation	实施 行动， 测试， 运营 以及 维护	无	实施 ation， 测试， 运营 以及 维护	无
DevSecOps ： 应用程序 性能 监控	无	操作 以及 维护	无	无	无

能力： 子能力 力	调查 集成 交互 迭代				影响
DevSecOps ：日志 聚合 和分析	操作 以及 维护	操作 以及 维护	无	无	无
DevSecOps： AIOps	操作 以及 维护	无	无	操作 以及 维护	无
DevSecOps ：持续 改进	无	无	无	操作 以及 维护	无
DevSecOps ：仪表板 监控	无	操作 以及 维护	无	无	无
DevSecOps： 性能 洞察	操作 以及 维护	无	无	操作 以及 维护	无
操作 以及 维护 ce：事件 管理	无	无	无	操作 以及 维护	无
操作 以及 维护 ce：代码 升级	无	操作 以及 维护	无	无	无

能力： 子能力 力	调查 集成 交互 迭代				影响
操作 以及 维护 ce: 代码 优化	操作 以及 维护	操作 以及 维护	无	无	无
操作 以及 维护 ce: 技术 债务 管理	无	操作 以及 维护	操作 以及 维护	无	无
操作 以及 维护 ce: 变更 管理	无	实施 ation, 部署	无	无	无
操作 以及 维护 ce: 逆向 工程	操作 以及 维护	无	无	无	无
操作 以及 维护 ce: 代码 moderniza tion	无	实施 ation	无	无	无

能力： 子能力 力	调查 集成 交互 迭代				影响
操作 以及 维护 ce: 性能 优化	无	操作 以及 维护	无	操作 以及 维护	无
分析 和洞察	无	需求 以及 规划	无	无	所有 SDLC 阶段
AI 助手 无		无	所有 SDLC 阶段	无	无
知识 管理 t	无	无	所有 SDLC 阶段	无	无
可扩展性：无		部署 无		无	无

用于项目管理的生成式 AI 用例

有效的项目管理是成功软件开发的核​​心。在生成式 AI 的背景下，项目管理呈现出新的维度：它可以变得更具预测性、更具适应性，并以数据为驱动。由 AI 驱动的项目管理工具会分析历史项目数据，从而生成更准确的时间与资源估算。它们可以根据业务目标和团队产能自动对任务进行优先级排序，甚至能在潜在障碍发生之前进行预测。例如，项目经理可以使用生成式 AI 基于项目需求以及类似项目的历史数据来创建初步的项目计划。随后，AI 还能建议最优的团队构成，综合考虑技能、工作负载和项目需求。在整个项目过程中，AI 驱动的仪表板会通过自动生成报告并突出需要关注的领域，提供近乎实时的项目状态洞察。

这种由 AI 增强的项目管理方法可以提升效率。它帮助项目经理将精力集中在战略决策和团队领导上，而不是陷入日常行政事务等例行任务中。

下表展示了你可以借助生成式 AI 加以增强的项目管理用例，以及负责这些用例的角色。

子能力：用例	角色
问题管理：创建并分配问题	项目经理
问题管理：在测试期间发现问题并记录	测试工程师
问题管理：根据严重程度对问题进行优先级排序，并将其分配给开发人员	项目经理
问题管理：识别并合并重复问题	项目经理
问题管理：跟踪并生成有关关键问题、指标以及项目整体健康状况的报告	项目经理
冲刺与任务管理：估算工作量 根据团队产能为任务分配故事点	Scrum Master
冲刺与任务管理：分配 在团队成员之间分配任务，确保整个冲刺期间工作量均衡	Scrum Master
冲刺与任务管理：促进冲刺 规划会议，使团队工作与冲刺目标保持一致	Scrum Master
产品待办事项管理：重新排序 根据业务价值、紧迫性和用户反馈对待办事项进行排序	产品负责人

子能力：用例	角色
产品待办事项管理：整合 将新的客户反馈和市场洞察纳入产品待办事项，以便进行近乎实时的优先级排序	产品负责人
产品待办事项管理：识别并 管理待办事项之间的依赖关系，以简化开发	产品经理
用户故事映射：创建用户的地图 梳理用户旅程，以识别所有必要功能及其对应的用户故事	产品负责人
用户故事映射：识别差距或 用户流程中缺失的步骤	业务分析师
用户故事映射：根据用户故事对业务价值的影响对其进行优先级排序	产品经理
报告与分析：生成近乎实时的报表与分析 实时仪表板，用于可视化关键项目指标，例如冲刺速度和问题解决率	项目经理
报告与分析：分析历史 数据并预测未来项目结果，例如潜在的延误或瓶颈	项目经理
报告与分析：创建自定义 报告，例如团队绩效或项目状态报告，以满足不同利益相关者的需求	项目经理
产品路线图管理：创建并 维护产品路线图，概述主要里程碑和发布日期	项目经理

子能力：用例	角色
产品路线图管理：根据项目优先级或时间线的变化更新路线图，根据项目优先级或时间线的变化	产品经理
产品路线图管理：分享该路线图与利益相关者制定路线图，以便清晰了解产品的发展方向	产品经理
反馈闭环：在每个冲刺结束后收集团队反馈，并识别需要改进的领域	Scrum Master
回顾会议：将反馈转化为下一次冲刺的可执行事项，推动持续改进	Scrum Master
回顾会议：跟踪从以往回顾会议中实施的变更所产生的影响，以衡量其有效性	Scrum Master

用于需求管理的生成式 AI 用例

需求管理是一个关键流程，与项目管理紧密相关。设想一位产品负责人使用 AI 工具来分析客户反馈、市场趋势和利益相关方输入。AI 工具可以生成一套全面的用户故事和需求，自动对其进行分类，检测潜在冲突或缺口，甚至基于业务价值和实现复杂度提出优先级建议。随着项目推进、需求不断演进，AI 还能持续更新和完善需求，确保它们始终与不断变化的业务需求和技术约束保持一致。这种动态的、AI 驱动的需求管理方法，有助于确保在整个项目生命周期内，开发工作始终与用户需求和业务目标紧密对齐。

下表展示了你可以使用生成式 AI 增强的需求管理用例，以及负责这些用例的角色。

用例	角色
创建业务需求	业务分析师
从功能创建史诗	产品负责人
通过监控其关联用户故事的完成情况来跟踪史诗的进度	产品经理
创建用户故事	产品负责人
估算每个用户故事所需的工作量并分配故事点	Scrum Master
为每个用户故事定义验收标准	产品负责人

用于架构与设计的生成式 AI 用例

在具备扎实的项目管理基础和明确的需求之后，下一个关键能力是架构与设计。在这一领域，生成式 AI 正在为创建稳健、可扩展且高效的软件架构开辟新的可能性。AI 驱动的设计工具可以分析需求和约束条件，提出最优的架构模式和设计方法建议。它们会生成多个设计备选方案，并且每个方案都会针对不同的优先级进行优化，例如性能、可扩展性或可维护性。例如，解决方案架构师可以使用 AI 助手根据项目需求快速生成多个高层次的架构设计。这种 AI 增强的方法加速了设计过程，并帮助架构师做出更明智的决策，从而带来更稳健、更具前瞻性的软件设计。

下表展示了可通过生成式 AI 增强的架构与设计用例，以及负责这些用例的角色画像。

用例	角色
创建架构文档	解决方案架构师
创建一份详细的设计文档	技术负责人

用例	角色
了解现有架构和设计标准	解决方案架构师
开发用户界面的详细模型和原型	UX/UI 设计师

用于协作的生成式 AI 用例

软件开发本质上是一项协作性工作。你可以使用生成式 AI 来增强软件开发团队的协作。由 AI 驱动的协作工具不止于简单的消息传递和文件共享。它们通过总结冗长的讨论线程、突出关键决策，甚至根据团队成员的日程安排和生产力模式建议最佳会议时间，来促进更高效的沟通。AI 还能在代码审查中提供帮助，自动识别潜在问题、提出改进建议，甚至向审查者解释复杂的变更。在头脑风暴会议中，AI 可以充当引导者，生成想法、帮助梳理思路，甚至调解讨论，以确保所有人的声音都能被听到。对于分布式团队，AI 可以帮助弥合文化和语言障碍。它可以在聊天和视频通话中提供近乎实时的语言翻译，并提供文化背景信息以帮助避免误解。通过用 AI 增强人类协作，这一能力可帮助团队更高效、更有效地工作，从而促进创新并改善整体项目成果。

下表展示了如何使用生成式 AI 来增强协作类用例。

子能力：用例	角色
文档管理：创建并维护一个集中式文档存储库	技术文档撰写人员
文档管理：允许多个团队成员实时协作编写文档	开发团队
知识共享：使用讨论论坛作为开发人员提问的平台	开发团队

子能力：用例	角色
分享知识，并协作排查问题	
知识共享：使用讨论论坛记录并跟踪项目讨论中做出的决策，确保关键决策背后的理由被记录下来，并可供未来参考	产品经理
项目资产管理：促进轻松共享与项目相关的资源	开发团队
项目资产管理：实施对共享内容进行版本控制，使团队成员能够跟踪更改、回滚到先前版本，并协作更新内容	开发团队

用于 DevSecOps 的生成式 AI 用例

AI 驱动的 DevSecOps 工具可自动化软件交付流水线的许多环节。例如，在开发人员编写代码时，它们几乎可以实时执行智能代码审查、检测潜在缺陷、发现安全漏洞并识别性能问题。AI 会生成并运行全面的测试套件，并随着代码库的演进自动更新这些测试套件。这种由 AI 增强的 DevSecOps 方法可加速交付流水线，并显著提升所交付软件的安全性与可靠性。

下表展示了可通过生成式 AI 增强的 DevSecOps 用例，以及负责这些用例的角色画像。

子能力：用例	角色
DevOps 与持续交付：自动化整个部署流水线	DevOps 工程师

子能力：用例	角色
DevOps 和持续交付：接收 近乎实时的代码质量反馈以及潜在问题	软件开发人员
DevOps 和持续交付：接收 近乎实时的安全问题与修复建议	软件开发人员
DevOps 和持续交付：接收 近乎实时的代码和最佳实践建议	软件开发人员
DevOps 和持续交付：自动化 重复性任务，并将命令集成到脚本中	DevOps 工程师
DevOps 和持续交付：构建代码 并在每次代码提交后自动生成构件	软件开发人员
DevOps 和持续交付：构建代码 符合组织的标准和框架	软件开发人员
DevOps 和持续交付：自动 在每次提交时自动运行单元测试，以便在开发过程的早期发现错误	软件开发人员
DevOps 和持续交付：分析该 单元测试覆盖率，确保所有关键代码路径都经过测试	软件开发人员
DevOps 和持续交付：管理 分支并合并更改	软件开发人员

子能力：用例	角色
DevOps 和持续交付：管理代码和工件版本控制	软件开发人员
DevOps 和持续交付：存储并管理构建工件和依赖项	DevOps 工程师
DevOps 和持续交付：解决并在构建过程中获取依赖项	软件开发人员
DevOps 和持续交付：生成并运行集成测试，以确保各组件能够按预期协同工作	测试工程师
DevOps 和持续交付：在集成测试期间使用模拟服务来模拟对外部系统的交互服务，在集成测试期间用于模拟与外部系统的交互	
DevOps 和持续交付：对应用程序在不同负载下的性能进行基准测试	性能工程师
DevOps 和持续交付：模拟高流量场景，以测试应用程序的可扩展性和响应时间	性能工程师
DevOps 和持续交付：测试系统从故障（如服务器崩溃或网络中断）中恢复的能力	站点可靠性工程师
DevOps 和持续交付：执行混沌工程	站点可靠性工程师
DevOps 和持续交付：运行测试以验证应用程序满足业务需求	QA 工程师
DevOps 和持续交付：执行用户验收测试	产品负责人

子能力：用例	角色
DevOps 和持续交付：扫描 依赖项中的漏洞和许可证合规性问题	安全工程师
DevOps 和持续交付：监控 并管理开源依赖项，确保它们保持最新且安全	安全工程师
DevOps 和持续交付：生成 并维护软件物料清单（SBOM），以跟踪所有 组件和依赖项	安全工程师
DevOps 和持续交付：使用 SBOM 开展监管合规审计	合规官
DevOps 和持续交付：创建 发布说明	发布经理
DevOps 和持续交付：规划并 协调发布	发布经理
DevOps 和持续交付：实施 用于回滚和发布管理的标准操作程序	发布经理
DevOps 和持续交付：使用功能标志在生产环境中启用或禁用功能，而无需部署新代码 功能标志，以便在生产环境中启用或禁用功能，而无需部署新代码	
DevOps 和持续交付：运行 A/ B 测试，使用功能标志来衡量不同功能对用户行为的影响	产品经理

子能力：用例	角色
DevOps 和持续交付：分析并监控流水线故障	DevOps 工程师
DevOps 和持续交付：创建并管理基础设施资源	DevOps 工程师
DevOps 与安全：扫描代码仓库以查找硬编码的密钥	DevOps 工程师
DevOps 与安全：实施近乎实时的检测，以便在机密信息被提交到代码仓库时立即提醒开发人员	DevOps 工程师
DevOps 与安全：实施近乎实时的检测，以便在机密信息被提交到代码仓库时立即提醒开发人员	DevOps 工程师
DevOps 与安全：强制持续执行代码质量监控	软件开发人员
DevOps 和安全：检测并标记代码中潜在安全漏洞的指标	软件开发人员
DevOps 和安全：实施自动化针对 Open Worldwide Application Security Project (OWASP) 十大安全风险的测试，以确保应用程序遵循行业标准的安全实践	安全工程师
DevOps 与安全：定期更新并将检查集成到开发流程中，教育开发人员了解 OWASP 风险	安全工程师
DevOps 与安全：扫描第三方已知安全漏洞的库和依赖项	DevOps 工程师

子能力：用例	角色
DevOps 与安全：扫描应用程序代码和基础设施以检测漏洞	DevOps 工程师
DevOps 与安全：分析代码以在部署前发现漏洞	安全工程师
DevOps 与安全：通过阻止包含关键漏洞的代码被合并来强制执行安全策略	安全工程师
DevOps 与安全：实施基于角色的使用基于角色的访问控制（RBAC）来限制对敏感系统和数据的访问，并确保只有经授权的人员才能访问关键资源	安全工程师
DevOps 与安全：根据角色和职责调整访问控制，并通过适应团队结构的变化来进行调整	DevOps 工程师
DevOps 与安全：通过在接近实时的情况下模拟对生产环境的攻击，测试正在运行的应用程序是否存在安全漏洞	安全工程师
DevOps 与安全：持续监控已部署的应用程序是否存在安全漏洞	DevOps 工程师
DevOps 与安全：定期安排在所有环境中进行漏洞扫描，以识别并解决安全弱点	安全工程师
DevOps 和安全：根据漏洞扫描结果应用补丁和更新，以帮助维护安全的系统	DevOps 工程师

子能力：用例	角色
应用程序性能监控： 以近乎实时的方式持续监控应用程序性能，在性能问题影响用户之前进行检测和诊断	站点可靠性工程师
应用程序性能监控：检测 性能异常，例如响应时间突然飙升或错误率增加，并触发告警	DevOps 工程师
应用性能监控：跟踪 请求在分布式系统中传播的过程，以识别性能瓶颈和延迟问题	DevOps 工程师
应用程序性能监控：使用 分布式追踪来精确定位导致故障或性能下降的具体服务或组件	DevOps 工程师
日志聚合与分析：聚合 将来自多个来源的日志汇总到一个集中式系统中，便于搜索和分析，以识别趋势和问题	站点可靠性工程师
日志聚合与分析：实施 自动化日志解析，以提取相关信息并检测可能表明存在问题的模式或异常	DevOps 工程师
日志聚合与分析：收集并 可视化关键性能指标	站点可靠性工程师
日志聚合与分析：监控 指标是否符合预定义的服务级别协议（SLA）	产品经理

子能力：用例	角色
AI 运维：在无需人工干预的情况下检测事件、分析根本原因并启动纠正措施	DevOps 工程师
AI 运维：预测未来资源需求并优化容量规划，以避免中断	站点可靠性工程师
持续改进：监控真实用户与应用程序的交互，以收集有关性能的洞察并识别需要改进的领域	UX 设计师
持续改进：跟踪应用程序在不同地理区域的性能，以确保全球一致的用户体验	产品经理
仪表板监控：创建可自定义的仪表板，以近乎实时的方式可视化关键指标、日志和追踪信息，从而提供系统健康状况的全面视图	站点可靠性工程师
仪表板监控：创建仪表板为不同团队（例如开发、运维和产品团队）创建仪表板，以便根据其关注领域提供相关洞察	DevOps 工程师
性能洞察：开展详细的应用性能分析，以识别低效之处并优化代码或基础设施	软件开发人员

子能力：用例	角色
性能洞察：使用性能洞察来迭代改进应用程序性能，并随着时间推移优化用户体验	产品经理

用于运维的生成式 AI 用例

软件部署后，重点会转向运行与维护。生成式 AI 可以通过提供更主动、更高效的系统管理来增强传统方法。由 AI 驱动的运维工具会持续监控系统性能，并在潜在问题影响用户之前预测这些问题。当问题发生时，它们会执行自动化根因分析，从而显著缩短平均解决时间。AI 还可以近乎实时地优化系统性能。它会根据不断变化的负载模式和用户行为自动调整配置。例如，运维团队可能会使用 AI 助手生成预测性维护计划，自动识别可能发生故障的组件，并建议预防性措施。AI 还可以通过分析使用趋势并高精度预测未来资源需求来帮助进行容量规划。

下表展示了你可以借助生成式 AI 加以增强的运维用例，以及负责这些用例的角色。

子能力：用例	角色
事件管理：管理事件于通过将监控工具与聊天平台集成，实现近乎实时的响应，使团队能够直接在聊天环境中发现、讨论并解决问题	站点可靠性工程师
事件管理：允许团队去从聊天界面直接发起部署、运行脚本并执行命令，从而简化运维操作	DevOps 工程师

子能力：用例	角色
代码升级：升级代码依赖项和库，以减少人工工作量，并确保代码库始终与最新版本保持同步	软件开发人员
代码优化：审查代码以寻找优化机会	软件开发人员
代码优化：识别代码中的瓶颈，并重构或优化代码以提升性能	软件开发人员
技术债务管理：记录技术 将技术债务作为开发流程的一部分进行记录	产品经理
技术债务管理：优先排序并 根据影响、风险和成本处理技术债务，并将其纳入常规的冲刺规划流程中	软件开发人员
技术债务管理：减少 现有应用程序代码中的技术债务	软件开发人员
变更管理：实施变更 审批流程，确保在部署前所有代码变更都经过必要相关方的审查、测试和批准	变更经理
变更管理：执行影响评估 拟议变更分析	DevOps 工程师
逆向工程：分析并理解遗留代码的结构和行为	解决方案架构师
逆向工程：解释现有代码并生成文档	软件开发人员

子能力：用例	角色
代码现代化：将代码从一种编程语言翻译为另一种编程语言	软件开发人员
代码现代化：将遗留代码现代化为最新的编程语言	软件开发人员
性能优化：通过优化资源分配、负载均衡以及重新配置应用程序，持续监控并调优系统性能	站点可靠性工程师
性能优化：识别并重构导致性能下降的代码，以提升速度和系统响应能力	软件开发人员

生成式 AI 助手在软件开发中的用例

AI 助手能力是由生成式 AI 驱动的开发体验的核心。这个智能、具备上下文感知能力的系统在整个 SDLC 中为所有团队成员充当虚拟协作者。设想一名开发人员正在处理一段复杂代码。他们只需向 AI 助手寻求帮助，AI 助手就能提供相关代码片段、解释复杂算法，甚至基于当前上下文和最佳实践提出优化建议。AI 助手还可以帮助 ITOps 经理基于内部文档理解标准操作流程。通过提供即时且与上下文相关的支持，AI 助手显著降低团队成员的认知负荷。这有助于他们将精力集中在更高层次的问题解决和创造性任务上。这项能力如同一个“倍增器”，在软件开发的各个阶段提升生产力与质量。

下表展示了你可以通过 AI 助手增强的用例以及受益的角色。

用例	角色
通过回答有关需求、架构和标准操作程序等问题，为开发团队提供即时协助	软件开发团队
使用自然语言查询从大量文档中搜索或检索摘录，或生成摘要	软件开发团队
总结冗长的技术文档，例如需求文档、架构设计文档和内部流程	软件开发团队
维护一个提示词库，供团队用于常见任务	软件开发团队
将生成式 AI 无缝集成到现有工具和系统中	软件开发团队
跨各种平台、工具和内部系统自动化任务	软件开发团队
创建一个集中式知识库，包含最佳实践、项目特定信息和团队知识，并确保所有团队成员都可访问	软件开发团队
根据任务上下文从知识库中检索相关知识	软件开发团队
执行自动化代码审查、根因分析、提出改进建议、检测潜在缺陷，并进行故障排查	软件开发人员、DevOps 工程师和站点可靠性工程师

用例	角色
分析性能数据，识别可为性能优化决策提供依据的趋势和模式	站点可靠性工程师
提供关于提高效率、降低复杂性和增强安全性的建议	软件开发人员
就云资源使用提出优化建议，例如扩缩容建议或成本节约策略	软件开发人员、DevOps 工程师、站点可靠性工程师和解决方案架构师
生成新内容，例如基于代码的文档、用户指南或产品功能发布说明	软件开发团队

用于分析与洞察的生成式 AI 用例

分析与洞察能力有助于将海量数据转化为可执行的洞察，从而推动决策制定与持续改进。通过使用生成式 AI，该能力可处理来自多种来源的数据，包括代码仓库、项目管理工具和团队协作平台，从而提供对开发过程与团队生产力的整体视图。生成式 AI 超越了传统指标，能够提供预测性与处方性分析。它可以预测潜在问题并提出有针对性的改进建议。例如，它可以分析代码提交模式、缺陷解决率以及功能交付速度等，以识别高绩效团队、定位瓶颈并建议流程优化。此外，它还可以提供关于团队动态与个人绩效的洞察。这些洞察可帮助领导者就工作负载分配、培训需求和团队构成做出数据驱动的决策。通过交互式仪表板呈现这些洞察，该能力使各级利益相关者能够做出明智决策、优化流程并持续提升团队生产力，从而更快交付高质量软件。

下表展示了可通过生成式 AI 增强的分析用例，以及负责这些用例的角色。

用例	角色
监控个人和团队生产力	开发经理
分析生产力趋势以发现潜在的倦怠，从而采取主动措施，维护团队福祉和生产力	开发经理
跟踪代码变更部署到生产环境的频率，以衡量开发过程的速度和敏捷性	产品经理
分析部署频率数据，以识别部署活动较低的时期，这可能表明流程效率低下或资源受限	产品经理
衡量从代码提交到部署之间的时间，以识别简化开发和部署流程的机会	开发经理
跟踪导致失败并需要立即修复的部署百分比，以评估发布流程的可靠性	站点可靠性工程师
使用变更失败率指标识别经常引发问题的代码区域，以指导有针对性的重构和测试工作	软件开发人员
监控在发生宕机或事件后恢复服务所需的时间，以便减少停机时间并提升整体系统韧性	站点可靠性工程师

用例	角色
分析恢复时间趋势，以改进事件响应流程，并推动从系统故障中更快恢复	DevOps 工程师
创建一个自定义仪表板，汇总部署频率、交付周期和变更失败率等关键指标，以提供对开发与运营健康状况的全面视图	产品经理
创建针对不同团队需求定制的仪表板，以便为其各自的职责领域（如开发、运营或业务）提供聚焦洞察	产品经理
跟踪业务关键绩效指标（KPI），例如收入影响、客户满意度和市场份额，以使开发工作与更广泛的业务目标保持一致	产品经理
分析新功能对业务 KPI 的影响，以评估其成功程度并指导未来的产品开发	业务分析师
监控代码质量指标，例如代码复杂度、测试覆盖率和缺陷密度，以确保代码库保持可维护性和安全性	软件开发人员
识别代码库中需要重构的领域，以推动长期可持续性并减少技术债务	解决方案架构师

用于知识管理的生成式 AI 用例

在任何软件开发组织中，知识都是一项关键资产。由生成式 AI 驱动的知识管理能力增强了对这一资产的捕获、组织和使用方式。传统的知识管理系统往往信息过多、包含过时内容，或难以搜索，从而无法快速找到相关信息。

生成式 AI 直面这些挑战。它会根据代码变更、对话和项目工件自动生成并更新文档。这确保知识库始终保持最新，而无需团队成员投入手动工作。更重要的是，AI 以直观的方式让这些知识触手可及。团队成员可以用自然语言提问，AI 则能提供相关答案。AI 可以从多种来源汲取信息，例如官方文档、代码注释、讨论线程，甚至外部资源。例如，一位新成员想要理解某个特定组件时，可以问 AI：“认证模块是如何工作的？”AI 随后会给出简明解释，并提供指向相关代码段、架构图以及近期变更的链接。它

甚至可以根据该成员的角色和专业水平对这些信息进行定制。

这种能力加速入职培训，减少重复性问题，并促进组织内的知识共享。它有助于保留机构知识，使团队更容易随着时间推移维护并演进复杂系统。

下表展示了可通过生成式 AI 增强的知识管理用例，以及负责这些用例的角色。

用例	角色
创建一个统一的平台，便于访问所有与项目相关的知识	软件开发团队
从各种开发活动中捕获知识	软件开发团队
提供高级搜索功能，以便在存储库中快速找到相关知识	软件开发团队

用例	角色
为团队个性化学习模块和学习路径	软件开发团队

用于可扩展性的生成式 AI 用例

可扩展性支持与现有工具和工作流的无缝集成，同时使组织能够根据自身特定需求定制 AI 系统。该能力提供强大的 API、SDK 和可自定义的界面，便于将 AI 功能集成到常用的开发与项目管理工具中。例如，组织可以为 Jira 增强由 AI 驱动的功能，用于自动化工单优先级排序、工作量估算和冲刺规划。你还可以在 Jenkins 流水线中引入 AI，实现智能构建优化和预测性测试选择。

此外，可扩展性支持与集成开发环境（IDE）、版本控制系统和代码审查平台进行深度集成。AI 可以帮助编写代码、自动化代码审查，并生成具备上下文的文档。

该能力还支持基于组织特定数据对 AI 模型进行训练和微调。这有助于 AI 理解公司特有的编码模式、架构偏好和领域知识。其结果是在所有已集成工具中提供更相关、更具上下文感知的辅助。通过提供这种程度的灵活性与集成，可扩展性确保 AI 驱动的开发体验能够随组织共同演进。它可以适应不断变化的技术与业务需求，同时无缝增强现有工具链与工作流。

下表展示了可通过生成式 AI 增强的可扩展性用例，以及负责这些用例的角色画像。

用例	角色
将第三方工具集成到开发环境中	DevOps 工程师
创建定制化自动化工作流，以适配团队独特的开发流程	DevOps 工程师

用例	角色
连接到各种 API 和服务	DevOps 工程师
为跨平台工具创建连接器 DevOps 工程师	

在软件开发中使用生成式 AI 的最佳实践

本节介绍将生成式 AI 集成到软件开发生命周期（SDLC）中的最佳实践。从实施无缝的工具链和 DevSecOps 流水线，到促进协作并自动化重复性任务，这些指南可帮助你利用 AI 的力量来增强开发流程与体验。遵循这些最佳实践，软件开发团队能够在工作中解锁更高水平的效率、创新与质量。

本节讨论以下最佳实践：

- 实施无缝的端到端集成工具链
- 为 DevSecOps 实施端到端 CI/CD 流水线
- 采用协作工具与实践
- 将重复性任务自动化
- 定期审查并迭代开发体验
- 采用有效的项目管理实践
- 实施知识管理
- 提供可扩展性和自定义能力
- 针对运营进行优化
- 利用数据驱动的洞察
- 采用平台化方法

实施无缝的端到端集成工具链

实施无缝的端到端集成工具链，是打造由生成式 AI 驱动的开发体验的一项基础性最佳实践。其核心理念是建立一个连贯一致的工具与平台生态系统，供你的软件团队在整个 SDLC 中使用。团队可以使用该工具链进行规划、构思、编码、构建、测试、部署以及管理持续运营。通过将生成式 AI 能力集成到该工具链中，你可以确保在每个阶段都能获得 AI 辅助。这种集成减少或消除人工交接，降低上下文切换，并帮助数据与工件在不同开发阶段之间顺畅流转。例如，来自你集成开发环境的 AI 生成代码片段

集成开发环境（IDE）中的工作可以无缝流入你的版本控制系统，而来自部署平台的 AI 驱动分析可以为你的项目管理工具提供信息。这会形成一个持续的反馈闭环，从而改进你的开发流程。

为 DevSecOps 实施端到端 CI/CD 流水线

在这一集成工具链的基础上，为 DevSecOps 实施端到端的持续集成与持续部署（CI/CD）流水线。这条由 AI 驱动的流水线是关键组件，可简化你的软件交付流程。它能帮助你更快速、更可靠地发布新应用和更新。通过在整个软件开发生命周期（SDLC）中嵌入安全实践，你可以更早识别并解决漏洞，从而降低总体成本与风险。该流水线应在每个阶段都引入 AI，从持续集成与测试到安全检查与部署。例如，你可以使用 AI 近乎实时地分析代码提交，以便在潜在的集成问题发生之前进行预测。在 CI/CD 流水线中，你还可以使用生成式 AI 根据最新的威胁情报自动更新安全策略。

采用协作工具与实践

在提升开发基础设施的同时，别忘了人的因素。软件开发本质上是一项协作性工作。它涉及由开发人员、设计师、产品经理、Scrum Master、业务分析师以及其他利益相关者组成的跨职能团队。这些人共同努力，将想法变为现实。通过使用现代协作工具并培育开放沟通与知识共享的文化，你可以显著提升软件开发团队的生产力和效率。在由 AI 驱动的软件开发体验中，这些工具将呈现新的维度。你可以将 AI 集成到协作平台中，促进团队成员之间更高效的沟通与知识共享。AI 助手可以回答常见问题、总结讨论，甚至调解冲突。生成式 AI 还能通过自动提出改进建议或识别潜在问题来增强代码审查流程。此外，你可以使用 AI 来创建动态的，随着项目演进而近乎实时更新的上下文感知文档，使所有团队成员都能获取最新且最相关的信息。

将重复性任务自动化

通过使用生成式 AI 处理日常且耗时的活动，你可以让软件团队专注于高价值的创造性工作，从而推动创新并带来业务影响。示例

重复性任务的例子包括生成样板代码、创建测试数据、编写文档，甚至起草初步的项目计划。通过将这些任务交给 AI，团队成员可以专注于更具创造性和战略性的工作。例如，AI 驱动的代码补全工具可以根据上下文和编码模式建议相关代码片段，从而显著加快编码过程。同样，生成式 AI 可以随着代码变更自动创建并更新技术文档。这能使文档保持最新，并减少通常为此任务所需的人工投入。在测试方面，AI 可以基于需求和代码分析生成全面的测试用例，从而提升测试覆盖率，并降低遗漏边界情况的可能性。通过智能地自动化这些重复性任务，生成式 AI 加速开发周期、提高一致性并减少人为错误。其结果是产出更高质量的软件。

定期审查并迭代开发体验

你的软件开发体验本身应被视为一款需要持续打磨的产品。这包括建立一套系统化流程，定期审查并迭代开发生命周期、工具和实践的各个方面。对整个工具链、工作流和流程进行周期性评估。收集来自不同角色的所有团队成员的反馈，包括产品经理、设计师、架构师、开发人员、测试人员和运维人员。请他们识别痛点、瓶颈以及改进机会。例如，团队可能会按季度审查其 CI/CD 流水线的性能，并

分析构建时间、部署频率和错误率等指标，以识别可优化的领域。由于生成式 AI 能力仍在快速演进，持续评估新的 AI 驱动工具和功能至关重要，这些工具和功能可能会进一步简化工作流，或在 SDLC 的所有角色中增强能力。

采用有效的项目管理实践

为了有效协调复杂的软件开发工作，应采用 AI 增强的项目管理实践。在此背景下，有效的项目管理超越了传统方法论。它采用 AI 增强的方法，在整个 SDLC 中提升规划、执行和监控。敏捷框架促进灵活性、协作和快速迭代，你可以使用生成式 AI 来优化这些流程。例如，生成式 AI 可以分析历史项目数据以获得更准确的估算，根据业务目标和客户反馈自动生成并确定用户故事的优先级，并提供智能

深入了解团队绩效。由 AI 驱动的项目管理工具可以预测潜在障碍，并根据团队成员的技能和工作负载建议最优的任务分配。通过将 AI 驱动的能力集成到项目管理实践中，你可以获得更高的可视化程度，更快地做出数据驱动的决策，并确保团队成员保持一致、协同高效地朝着共同目标推进。

实施知识管理

随着由 AI 驱动的软件开发体验日趋成熟，请实施一个强大的知识管理系统。强大的知识管理系统有助于你捕获、组织并授予对宝贵洞见、最佳实践和解决方案的访问权限。SDLC 全流程中的所有团队成员都应能轻松访问该系统。使用生成式 AI 创建动态、智能的知识库，使其随组织共同演进。例如，AI 可以根据代码变更、对话和项目工件自动生成并更新文档，从而使信息无需人工干预也能保持最新。生成式 AI 还可以提供智能搜索能力，帮助团队成员使用自然语言查询快速找到相关信息，即使他们不知道确切术语也可以。此外，生成式 AI 还能根据团队成员当前的任务或挑战，主动推送相关信息。它充当虚拟导师，增强所有角色的决策制定与问题解决能力。通过实施由 AI 驱动的知识管理系统，你可以打破信息孤岛，加速入职，减少重复性工作，并在整个软件开发团队中培育持续学习与创新的文化。

提供可扩展性和自定义能力

为了最大化生成式 AI 在软件开发中的收益，请确保你的 AI 驱动工具和平台具备可扩展性与可定制性。这有助于你将 AI 能力按自身的特定需求、工作流和技术栈进行定制。例如，你可以在自己的代码库和文档上对 AI 模型进行微调，为特定任务创建自定义的 AI 驱动工具，或将 AI 能力集成到现有工具和流程中。这种可扩展性有助于你演进 AI 驱动的开发体验，以满足组织不断变化的需求。它也有助于你针对特定领域或项目类型优化体验。

针对运营进行优化

生成式 AI 在优化软件运营与维护方面发挥着关键作用。通过将 AI 能力集成到你的运营工具和流程中来优化运营。例如，使用

使用生成式 AI 近乎实时地分析日志数据，预测潜在的系统故障，并自动化日常维护任务。生成式 AI 还可以通过复杂的分布式系统中关联各类事件来帮助进行根因分析。这将提升系统可靠性、减少停机时间，并让你的运维团队能够将精力集中在更具战略性的举措上。

利用数据驱动的洞察

在你的 AI 赋能开发旅程中始终使用数据驱动的洞察。实施系统来收集、分析并对 SDLC 各阶段的数据采取行动。这包括代码度量指标、测试结果、部署数据、用户反馈以及运行性能。使用生成式 AI 挖掘人类观察者可能不易察觉的模式与洞察。然后，将这些洞察反馈到你的开发流程中，为从架构决策到功能优先级排序的各项工作提供依据。

采用平台化方法

要在软件开发中充分实现生成式 AI 的收益，应采用以平台为基础的方法。构建一个全面、集成的平台，将 AI 能力融入软件开发生命周期（SDLC）的各个方面。该平台应提供一致的用户体验、集中式的管理与数据，以及不同工具和流程之间的无缝集成。这将使 AI 的收益在整个组织内得到统一覆盖，降低管理多个分散且各不相同的 AI 工具所带来的开销，并为 AI 能力的持续改进与扩展奠定基础。

衡量生成式 AI 在软件开发中的成功

要有效衡量实施由生成式 AI 驱动的软件开发体验所带来的影响，你需要建立一套全面的指标体系，覆盖软件开发生命周期（SDLC）的各个维度。这些指标应既能捕捉效率与生产力的即时提升，也能反映软件质量、团队满意度和业务价值的长期收益。

要有效使用本节推荐的指标，请执行以下操作：

1. 建立基线——在着手实施由 AI 驱动的开发体验之前，花时间收集关于你在这些指标上的当前表现的全面数据。这将提供一个清晰的起点，并帮助你在后续进行有意义的对比。
2. 设定切实可行的目标——在掌握基线数据后，为每项指标设定可实现的改进目标。要有雄心，但也要现实。请记住，可持续的进步往往是循序渐进的。
3. 实施持续监控——使用自动化工具在你的环境中持续收集并分析这些指标的数据。近实时监控有助于你跟踪进展，并快速识别任何问题或机会。
4. 开展定期复盘——安排每季度或每半年一次的复盘会议，你和团队在会上全面评估相对于目标的进展。利用这些会议识别进一步改进的领域，并庆祝你们的成功。
5. 迭代与调整——基于你获得的洞察，持续优化你的生成式 AI 实施，并在必要时调整目标。

本节介绍以下几类指标：

- 部署速度
- 代码质量
- 运营效率
- 团队生产力和满意度
- 业务影响

部署速度

考虑衡量以下部署速度指标。

指标	描述
上市时间	衡量从创意构思到生产环境部署所需时间的缩短幅度
冲刺速度	跟踪团队每个冲刺完成的故事点数的增长情况
代码提交频率	监控代码提交次数的增加情况，这表明开发周期正在加速
拉取请求解决时间	评估在你的代码仓库中审查并合并代码变更所需时间的减少情况
发布速度	衡量每季度或每年发布次数的增长

代码质量

考虑衡量以下代码质量指标。

指标	描述
缺陷密度	衡量软件缺陷数量的减少情况
代码覆盖率	跟踪整个代码库中测试覆盖率百分比的提升
技术债务	监控已识别的技术债务随时间推移的减少情况
静态代码分析评分	根据你的自动化分析工具评估代码质量的改进情况

运营效率

考虑衡量以下运营效率指标。

指标	描述
部署频率	衡量成功部署次数的增长情况
平均恢复时间（MTTR）	跟踪从系统故障中恢复所需时间的减少情况
变更失败率	监控在你的部署中导致失败的变更百分比的下降情况

团队生产力和满意度

考虑衡量以下团队生产力和满意度指标。

指标	描述
生产力提升	监测每项任务的生产力提升百分比
满意度评分	定期开展调查，以评估团队士气和工作满意度的提升情况
知识共享效率	衡量团队在搜索信息或提出重复性问题上所花时间的减少幅度
入职培训时间	跟踪新团队成员达到高效产出所需时间的减少情况

业务影响

考虑衡量以下业务影响指标。

指标	描述
功能采用率	衡量用户对你已发布新功能的参与度提升情况
客户满意度评分	跟踪用户反馈和评分的改进情况
收入影响（直接和间接）	评估因发布速度提升或生产力提高而带来的收入增长

结论

本策略文档概述了由生成式 AI 驱动的软件开发体验。它探讨了 5-I 框架中的五个维度——Investigate、Integrate、Interact、Iterate 和 Impact。这些维度为在整个软件开发生命周期（SDLC）中集成生成式 AI 提供了战略路线图。文档还描述了成功实施该框架所需的基础能力。这些能力涵盖项目管理、DevSecOps、AI 助手、知识管理等领域。文档提供了在集成生成式 AI 时需要考虑的最佳实践，并帮助你使用指标来衡量生成式 AI 对你的软件开发体验所产生的影响。

将生成式 AI 集成到软件开发流程中代表了一种范式转变，具有加速创新、提升质量和增强生产力的潜力。然而，重要的是要认识到，这并非一次性实施。它是一种持续演进，需要长期投入与不断优化。

当你踏上这段旅程时，我们建议你先对组织当前的能力和准备情况进行全面评估。AWS Assessment Tool 是一款由 AI 驱动的软件开发评估工具，可帮助你识别优先领域并制定量身定制的实施路线图。

资源

在你确定关键优先领域后，以下资源可帮助你实施路线图：

AWS 文档

- 使用 Amazon Bedrock 自动化 AWS 基础设施运维（AWS Prescriptive Guidance）
- 使用 Amazon Q Developer 进行内联和助手式代码生成的最佳实践（AWS Prescriptive Guidance）
- 使用 Amazon Bedrock 代理和知识库开发一个完全自动化的基于聊天的助手（AWS Prescriptive Guidance）
- 使用生成式 AI 在 AWS 上转型应用程序开发与维护的运营模式（AWS Prescriptive Guidance）

- 使用 Amazon Q Developer 作为编码助手来提高你的生产力 (AWS 规范性指导)

AWS 博客文章和教程

- Amazon Q 博客文章
- 使用 Amazon Q 加速你的软件开发生命周期 (AWS 博客文章)
- 构建 AWS 解决方案架构师 AI 代理：利用 Amazon Bedrock 实现自动化架构与部署 (AWS 视频)
- 生成式 AI 驱动的技术运营 (AWS 博客文章)
- 使用 Amazon Q Developer 现代化改造你的 Java 应用程序 (AWS 博客文章)
- 在软件开发流水线中使用 Amazon Bedrock 生成、评估并理解代码 (AWS 博客文章)

文档历史记录

下表描述了本指南的重大变更。如果您希望收到未来更新的通知，可以订阅 RSS 源。

变更	说明	日期
首次发布	不适用	2025 年 4 月 18 日

AWS Prescriptive Guidance 术语表

以下是 AWS Prescriptive Guidance 提供的策略、指南和模式中常用的术语。要建议添加条目，请使用术语表末尾的“提供反馈”链接。

数字

7 Rs 将应用程序迁移到云端的七种常见迁移策略。这些策略构建于

基于 Gartner 在 2011 年提出的 5R，并包括以下内容：

- **重构/重新架构 (Refactor/re-architect)** ——迁移应用并修改其架构，充分利用云原生特性，以提升敏捷性、性能和可扩展性。这通常涉及迁移操作系统和数据库。示例：将本地 Oracle 数据库迁移到 Amazon Aurora PostgreSQL-Compatible Edition。
- **重新平台化 (Replatform, 提升并调整 / lift and reshape)** ——将应用迁移到云端，并引入一定程度的优化，以利用云能力。示例：将本地 Oracle 数据库迁移到 AWS 云中的 Amazon Relational Database Service (Amazon RDS) for Oracle。
- **重新购买 (Repurchase, 弃用并选购 / drop and shop)** ——切换到不同的产品，通常是从传统许可模式转为 SaaS 模式。示例：将客户关系管理 (CRM) 系统迁移到 Salesforce.com。
- **重新托管 (lift and shift)** ——在不做任何更改的情况下将应用程序迁移到云端，以利用云功能。示例：将本地 Oracle 数据库迁移到 AWS 云中 EC2 实例上的 Oracle。
你将服务器从本地平台迁移到云服务中相同的平台。
示例：将 Microsoft Hyper-V 应用程序迁移到 AWS。
- **保留 (重新评估)** ——将应用程序保留在源环境中。这些可能包括需要进行重大重构的应用程序，而你希望将这项工作推迟到以后再做；以及你希望保留的遗留应用程序，因为迁移它们没有业务上的正当理由。

- 淘汰——对源环境中不再需要的应用程序进行退役或移除。

A

ABAC

请参阅基于属性的访问控制。

抽象化服务

请参阅托管服务。

ACID

请参阅原子性、一致性、隔离性、持久性。

双活迁移

一种数据库迁移方法，其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），并且在迁移期间两个数据库都处理来自连接应用程序的事务。该方法支持以小规模、可控的批次进行迁移，而不是要求一次性切换。它更灵活，但比主动-被动迁移需要更多工作。

主动-被动迁移

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

聚合函数

一种对一组行进行操作并为该组计算单一返回值的 SQL 函数。聚合函数的示例包括 SUM 和 MAX。

AI

参见人工智能。

AIOps

请参阅人工智能运维。

匿名化

在数据集中永久删除个人信息的过程。匿名化有助于保护个人隐私。匿名化后的数据不再被视为个人数据。

反模式

针对反复出现的问题而经常采用的一种解决方案，但该方案会适得其反、无效，或不如替代方案有效。

应用程序控制

一种安全方法，只允许使用已批准的应用程序，以帮助保护系统免受恶意软件侵害。

应用程序组合

关于组织所使用的每个应用程序的详细信息集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是组合发现与分析过程的关键，有助于识别并确定需要迁移、现代化改造和优化的应用程序的优先级。

人工智能 (AI)

计算机科学领域，致力于使用计算技术来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？》。

人工智能运维 (AIOps)

使用机器学习技术来解决运维问题、减少运维事件和人工干预，并提升服务质量的过程。有关 AIOps 在 AWS 迁移策略中如何使用的更多信息，请参阅《运维集成指南》。

非对称加密

一种使用一对密钥的加密算法：公钥用于加密，私钥用于解密。你可以共享公钥，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，用于保证数据库的数据有效性和操作可靠性，即使在发生错误、断电或其他问题的情况下也是如此。

基于属性的访问控制 (ABAC)

一种基于用户属性（例如部门、工作角色和团队名称）创建细粒度权限的做法。有关更多信息，请参阅 AWS Identity and Access Management (IAM) 文档中的《ABAC for AWS》。

权威数据源

用于存储数据主版本的位置，被视为最可靠的信息来源。你可以将数据从权威数据源复制到其他位置，以便对数据进行处理或修改，例如对其进行匿名化、脱敏或假名化。

可用区

AWS 区域内的一个独立位置，与其他可用区的故障相互隔离，并为同一区域内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)

由 AWS 提供的一套指南和最佳实践框架，帮助组织制定高效且有效的计划，从而成功迁移到云端。AWS CAF 将指导内容组织为六个称为“视角”的重点领域：业务、人员、治理、平台、安全和运营。业务、人员和治理视角侧重于业务技能和流程；平台、安全和运营视角侧重于技术技能和流程。例如，人员视角面向负责处理人力资源 (HR)、人员配置职能以及人员管理的相关方。针对该视角，AWS CAF 提供关于人员发展、培训和沟通的指导，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 工作负载资格评估框架 (AWS WQF)

一款用于评估数据库迁移工作负载、推荐迁移策略并提供工作量估算的工具。AWS WQF 随 AWS Schema Conversion Tool (AWS SCT) 一并提供。它会分析数据库架构和代码对象、应用程序代码、依赖关系以及性能特征，并提供评估报告。

B

恶意机器人

旨在扰乱或对个人或组织造成伤害的机器人。

BCP

参见业务连续性规划。

行为图

对资源行为及其随时间变化的交互进行统一的交互式视图。你可以在 Amazon Detective 中使用行为图来检查失败的登录尝试、可疑的 API 调用以及类似操作。有关更多信息，请参阅 Detective 文档中的“行为图中的数据”。

大端系统

一种先存储最高有效字节的系统。另请参阅字节序。

二元分类

一种预测二元结果（两种可能类别之一）的过程。例如，你的机器学习模型可能需要预测诸如“这封邮件是垃圾邮件还是非垃圾邮件？”或“这个产品是书还是汽车？”之类的问题。

布隆过滤器

一种概率型、内存高效的数据结构，用于测试某个元素是否属于某个集合。

蓝/绿部署

一种部署策略：创建两个彼此独立但完全相同的环境。在一个环境（蓝）中运行当前应用版本，在另一个环境（绿）中运行新应用版本。该策略有助于你在影响最小的情况下快速回滚。

机器人

一种软件应用程序，可在互联网上运行自动化任务，并模拟人类活动或交互。有些机器人是有用或有益的，例如用于在互联网上索引信息的网络爬虫。另一些机器人被称为恶意机器人（bad bots），其目的是干扰或对个人或组织造成伤害。

僵尸网络

由恶意软件感染并由单一方控制的机器人网络，该控制方称为僵尸网络操控者或僵尸网络运营者。僵尸网络是扩展机器人规模及其影响的最知名机制。

分支

代码仓库中的一个独立区域。在仓库中创建的第一个分支是主分支。你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缺陷。为构建某项功能而创建的分支通常称为功能分支。当该功能准备发布时，你会将功能分支合并回主分支。更多信息，请参阅 [About branches \(GitHub documentation\)](#)。

破窗访问

在特殊情况下，通过经批准的流程，为用户提供一种快速方式，以获得对其通常没有权限访问的 AWS 账户的访问权限。有关更多信息，请参阅 AWS Well-Architected 指南中的“实施破窗流程”指标。

棕地策略

您环境中现有的基础设施。在为系统架构采用棕地策略时，您会围绕当前系统和基础设施的约束来设计架构。如果您正在扩展现有基础设施，可能会将棕地和绿地策略相结合。

缓冲区缓存

存储最常访问数据的内存区域。

业务能力

企业为创造价值而开展的活动（例如，销售、客户服务或市场营销）。

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“围绕业务能力组织”部分。

业务连续性规划 (BCP)

一项计划，用于应对破坏性事件（例如大规模迁移）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（AWS Cloud Adoption Framework）。

金丝雀部署

将某个版本缓慢且逐步地发布给最终用户。当你有信心时，便部署新版本并完全替换当前版本。

CCoE

请参阅云卓越中心。

CDC

请参阅变更数据捕获。

变更数据捕获（CDC）

跟踪对数据源（例如数据库表）的更改并记录有关该更改的元数据的过程。你可以将 CDC 用于多种目的，例如审计，或在目标系统中复制更改以保持同步。

混沌工程

有意引入故障或破坏性事件，以测试系统的韧性。你可以使用 AWS Fault Injection Service（AWS FIS）执行实验，对你的 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过程。用于分类问题的机器学习（ML）模型会预测一个离散值。离散值彼此之间始终是不同的。例如，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 服务接收数据之前，在本地对数据进行加密。

云卓越中心 (CCoE)

一个跨学科团队，负责推动整个组织的云采用工作，包括制定云最佳实践、调动资源、建立迁移时间表，并在大规模转型过程中引领组织。有关更多信息，请参阅 AWS Cloud Enterprise Strategy Blog 上关于 CCoE 的文章。

云计算

通常用于远程数据存储和物联网 (IoT) 设备管理的云技术。云计算通常与边缘计算技术相连接。

云运营模型

在 IT 组织中，用于构建、成熟并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

组织在迁移到 AWS 云时通常会经历的四个阶段：

- 项目——为概念验证和学习目的运行一些与云相关的项目
- 基础——进行基础性投入以扩展你的云采用（例如，创建着陆区、定义 CCoE、建立运营模型）
- 迁移——迁移单个应用程序
- 再造——优化产品和服务，并在云中进行创新

这些阶段由 Stephen Orban 在 AWS Cloud Enterprise Strategy 博客上的文章《The Journey Toward Cloud-First & the Stages of Adoption》中定义。有关它们与 AWS 迁移策略之间关系的信息，请参阅迁移就绪指南。

CMDB

请参阅配置管理数据库。

代码仓库

一个用于存储源代码及其他资产（如文档、示例和脚本）并通过版本控制流程进行更新的位置。常见的云端代码仓库包括 GitHub 或 Bitbucket Cloud。代码的每个版本称为一个分支。在微服务架构中，每个仓库专注于单一功能模块。单个 CI/CD 管道可以使用多个仓库。

冷缓存

一种为空、填充不足，或包含过期或无关数据的缓冲缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取数据，而这比从缓冲缓存读取更慢。

冷数据

很少被访问、通常为历史数据的数据。查询这类数据时，通常可以接受较慢的查询。将这些数据迁移到性能较低且成本更低的存储层级或存储类别可以降低成本。

计算机视觉 (CV)

人工智能的一个领域，使用机器学习来分析并从数字图像和视频等视觉格式中提取信息。例如，Amazon SageMaker AI 提供用于计算机视觉的图像处理算法。

配置漂移

对于某个工作负载，其配置相对于预期状态发生的变化。这可能导致该工作负载变得不合规，且通常是逐渐发生并且非故意的。

配置管理数据库 (CMDB)

一种用于存储和管理有关数据库及其 IT 环境信息的存储库，包括硬件和软件组件及其配置。你通常会在迁移的组合发现与分析阶段使用 CMDB 中的数据。

合规性包

一组 AWS Config 规则和补救操作，你可以将其组合起来，以自定义合规性与安全检查。你可以使用 YAML 模板，将合规性包作为单个实体部署到某个 AWS 账户和区域中，或跨组织进行部署。有关更多信息，请参阅 AWS Config 文档中的 Conformance packs。

持续集成和持续交付 (CI/CD)

在软件发布过程中，对源代码、构建、测试、预发布和生产阶段进行自动化的流程。CI/CD 通常被描述为一条流水线。CI/CD 可以帮助你实现流程自动化、提高生产力、提升代码质量并加快交付速度。更多信息，请参阅《持续交付的优势》。CD 也可以表示持续部署。

更多信息，请参阅《持续交付与持续部署：对比》。

CV

请参阅计算机视觉。

D

静态数据

在你的网络中处于静止状态的数据，例如存储中的数据。

数据分类

一种根据数据的重要性和敏感性来识别并对网络中的数据进行分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它有助于你确定适用于数据的恰当保护与保留控制措施。数据分类是 AWS Well-Architected Framework 中安全支柱的组成部分。有关更多信息，请参阅 [Data classification](#)。

数据漂移

生产数据与用于训练机器学习（ML）模型的数据之间出现的显著差异，或输入数据随时间发生的显著变化。数据漂移会降低机器学习模型预测的整体质量、准确性和公平性。

传输中的数据

正在通过你的网络主动传输的数据，例如在网络资源之间传输的数据。

数据网格

一种架构框架，通过集中式管理和治理，提供分布式、去中心化的数据所有权。

数据最小化

仅收集和处理严格必要数据的原则。在 AWS 云中践行数据最小化可以降低隐私风险、成本以及你的分析碳足迹。

数据边界

在你的 AWS 环境中设置的一组预防性防护栏，用于帮助确保只有受信任的身份从预期的网络访问受信任的资源。有关更多信息，请参阅 [Building a data perimeter on AWS](#)。

数据预处理

将原始数据转换为便于你的 ML 模型解析的格式。数据预处理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中跟踪其来源和历史的过程，例如数据是如何生成、传输和存储的。

数据主体

其数据正在被收集和处理的个人。

数据仓库

一种支持商业智能（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）

用于创建或修改数据库中表和对象结构的语句或命令。

数据库操作语言（DML）

用于修改（插入、更新和删除）数据库中信息的语句或命令。

DDL

请参阅数据库定义语言。

深度集成

将多个深度学习模型组合用于预测。你可以使用深度集成来获得更准确的预测，或用于估计预测中的不确定性。

深度学习

机器学习的一个子领域，使用多层人工神经网络来识别输入数据与关注的目标变量之间的映射关系。

纵深防御

一种信息安全方法，通过在计算机网络中有意识地分层部署一系列安全机制和控制措施，以保护网络及其数据的机密性、完整性和可用性。当你在 AWS 上采用此策略时，会在 AWS Organizations 结构的不同层级添加多重控制措施，以帮助保护资源。例如，纵深防御方法可能会结合多因素身份验证、网络分段和加密。

委派管理员

在 AWS Organizations 中，兼容的服务可以注册一个 AWS 成员账户来管理组织的账户，并为该服务管理权限。该账户称为该服务的委派管理员。有关更多信息以及兼容服务列表，请参阅 AWS Organizations 文档中的《与 AWS Organizations 配合使用的服务》。

部署

将应用程序、新功能或代码修复发布到目标环境的过程。部署包括在代码库中实施变更，然后在应用程序的各个环境中构建并运行该代码库。

开发环境

参见 环境。

侦测性控制

一种安全控制，旨在在事件发生后进行检测、记录并发出告警。这些控制是第二道防线，用于提醒你那些绕过现有预防性控制的安全事件。更多信息，请参阅《Implementing security controls on AWS》中的 Detective controls。

开发价值流映射 (DVSM)

一种用于识别并确定会对软件开发生命周期中的速度和质量产生不利影响的约束的优先级的过程。DVSM 扩展了最初为精益制造实践设计的价值流映射过程。它聚焦于在软件开发过程中创建并传递价值所需的步骤和团队。

数字孪生

对现实世界系统的虚拟表示，例如建筑、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

在星型架构中，一种较小的表，用于包含事实表中定量数据的相关数据属性。维度表属性通常是文本字段或表现得像文本的离散数字。这些属性常用于查询约束、筛选以及结果集标注。

灾难

导致工作负载或系统无法在其主要部署位置实现业务目标的事件。这些事件可能是自然灾害、技术故障，或由人为行为导致，例如无意的错误配置或恶意软件攻击。

灾难恢复 (DR)

用于将灾难造成的停机时间和数据丢失降至最低的策略和流程。有关更多信息，请参阅 AWS Well-Architected Framework 中的《Disaster Recovery of Workloads on AWS: Recovery in the Cloud》。

DML

请参阅数据库操作语言。

领域驱动设计

一种通过将复杂软件系统的各个组件与其所服务的不断演进的领域（或核心业务目标）相连接来开发该系统的方法。该概念由 Eric Evans 在其著作《领域驱动设计：应对软件核心的复杂性》（波士顿：Addison-Wesley Professional, 2003）中提出。有关如何将领域驱动设计与绞杀者无花果模式结合使用的信息，请参阅《使用容器和 Amazon API Gateway 逐步现代化遗留 Microsoft ASP.NET (ASMX) Web 服务》。

DR

参见灾难恢复。

漂移检测

跟踪与基线配置的偏差。例如，你可以使用 AWS CloudFormation 来检测系统资源中的漂移，或者使用 AWS Control Tower 来检测你的登录区中可能影响治理要求合规性的更改。

DVSM

请参阅开发价值流映射。

E

EDA

参见探索性数据分析。

EDI

参见电子数据交换。

边缘计算

一种在物联网（IoT）网络边缘为智能设备提升计算能力的技术。与云计算相比，边缘计算可以降低通信延迟并改善响应时间。

电子数据交换（EDI）

组织之间业务文档的自动化交换。有关更多信息，请参阅 [What is Electronic Data Interchange](#)。

加密

一种计算过程，将人类可读的明文数据转换为密文。

加密密钥

由加密算法生成的随机比特组成的加密字符串。密钥长度可能不同，每个密钥都被设计为不可预测且唯一。

字节序

字节在计算机内存中存储的顺序。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

一种你可以在虚拟私有云（VPC）中托管并与其他用户共享的服务。你可以使用 AWS PrivateLink 创建端点服务，并向其他 AWS 账户或 AWS Identity and Access Management（IAM）主体授予权限。这些账户或主体可以通过创建接口 VPC 端点，以私有方式连接到你的端点服务。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）

一种为企业自动化并管理关键业务流程（如会计、MES 和项目管理）的系统。

信封加密

使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“Envelope encryption”。

环境

正在运行的应用程序实例。以下是云计算中常见的环境类型：

- 开发环境——仅对负责维护该应用程序的核心团队开放的正在运行的应用程序实例。开发环境用于在将更改提升到更高层级环境之前进行测试。这类环境有时也称为测试环境。
- 低层级环境——某个应用程序的所有开发环境，例如用于初始构建和测试的环境。
- 生产环境——终端用户可以访问的正在运行的应用程序实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上游环境——所有可由核心开发团队以外的用户访问的环境。这可能包括生产环境、预生产环境以及用于用户验收测试的环境。

史诗

在敏捷方法论中，用于帮助组织和确定工作优先级的功能类别。史诗提供对需求和实施任务的高层级描述。例如，AWS CAF 安全史诗包括身份与访问管理、侦测控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中史诗的更多信息，请参阅项目实施指南。

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。你收集或汇总数据，然后进行初步调查以发现模式、检测异常并检验假设。EDA 通过计算汇总统计量和创建数据可视化来完成。

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，事实表包含两类列：包含度量值的列，以及包含指向维度表的外键的列。

快速失败

一种通过频繁且增量式测试来缩短开发生命周期的理念。它是敏捷方法中的关键组成部分。

故障隔离边界

在 AWS 云中，用于限制故障影响范围并帮助提升工作负载弹性的边界，例如可用区、AWS 区域、控制平面或数据平面。更多信息，请参阅 [AWS Fault Isolation Boundaries](#)。

功能分支

参见分支。

特征

用于进行预测的输入数据。例如，在制造业场景中，特征可以是从小规模定期采集的图像。

特征重要性

某个特征对模型预测的重要程度。通常以数值分数表示，可通过多种技术计算，例如 Shapley Additive Explanations (SHAP) 和集成梯度。更多信息，请参阅《使用 AWS 的机器学习模型可解释性》。

特征转换

为了优化用于机器学习 (ML) 过程的数据，包括通过额外来源丰富数据、缩放数值，或从单个数据字段中提取多组信息。这使得 ML 模型能够从数据中受益。例如，如果你将“2021-05-27 00:15:37”这个日期拆分为“2021”“May”“Thu”和“15”，就可以帮助学习算法学习与不同数据组成部分相关的细微模式。

少样本提示

在让 LLM 执行类似任务之前，先向其提供少量示例，以展示任务内容和期望输出。这种技术是上下文学习 (in-context learning) 的一种应用，模型从嵌入在提示中的示例 (shots) 中学习。少样本提示对于需要特定格式、推理或领域知识的任务可能很有效。另见零样本提示。

FGAC

请参阅细粒度访问控制。

细粒度访问控制 (FGAC)

使用多个条件来允许或拒绝访问请求。

闪切迁移

一种数据库迁移方法，通过变更数据捕获 (change data capture) 进行持续数据复制，以在尽可能短的时间内迁移数据，而不是采用分阶段方法。其目标是将停机时间降至最低。

FM

请参阅 foundation model。

基础模型 (FM)

一种大型深度学习神经网络，已在海量的通用且未标注数据集上完成训练。基础模型能够执行多种通用任务，例如理解语言、生成文本和图像，以及用自然语言进行对话。更多信息，请参阅 [What are Foundation Models](#)。

G

生成式人工智能

一类 AI 模型的子集，这些模型在大量数据上训练，并且可以通过一个简单的文本提示来创建新的内容和产物，例如图像、视频、文本和音频。

更多信息，请参阅 [What is Generative AI](#)。

地理屏蔽

请参阅[地理限制](#)。

地理限制（地理屏蔽）

在 Amazon CloudFront 中，用于防止特定国家/地区的用户访问内容分发的一项选项。您可以使用允许列表或阻止列表来指定获准和被禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档中的《[限制内容的地理分发](#)》。

Gitflow 工作流

一种方法：在源代码仓库中，较低环境和较高环境使用不同的分支。Gitflow 工作流被认为是遗留做法，而基于主干的工作流是现代且更受青睐的方法。

黄金镜像

系统或软件的快照，用作模板来部署该系统或软件的新实例。例如，在制造业中，黄金镜像可用于在多台设备上配置软件，并有助于提升设备制造运营的速度、可扩展性和生产力。

绿地策略

在新环境中不存在现有基础设施。在为系统架构采用绿地策略时，你可以不受限制地选择全新的技术。

与现有基础设施的兼容性，也称为棕地（brownfield）。如果你正在扩展现有基础设施，可能会将棕地和绿地（greenfield）策略结合使用。

防护栏

一种高层级规则，用于帮助在组织单元（OU）之间治理资源、策略与合规性。预防性防护栏通过强制执行策略来确保符合合规标准。

它们通过服务控制策略（service control policies）和 IAM 权限边界来实现。侦测性防护栏用于检测策略违规和合规问题，并生成告警以便修复。它们通过 AWS Config、AWS Security Hub CSPM、Amazon GuardDuty、AWS Trusted Advisor、Amazon Inspector 以及自定义 AWS Lambda 检查来实现。

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 AWS SCT 来帮助进行架构转换。

高可用性（HA）

工作负载在遇到挑战或灾难时无需人工干预即可持续运行的能力。HA 系统旨在自动故障转移，持续提供高质量性能，并以最小的性能影响处理不同负载和故障。

历史数据系统现代化

一种用于对运营技术（OT）系统进行现代化改造和升级的方法，以更好地满足制造业的需求。Historian 是一种数据库，用于收集并存储工厂中来自各种来源的数据。

留出数据

从用于训练机器学习模型的数据集中保留的一部分历史标注数据。你可以通过将模型预测与留出数据进行比较来评估模型性能。

同构数据库迁移

将源数据库迁移到使用相同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）。同构迁移通常是重新托管或重新平台化工作的一部分。你可以使用原生数据库实用工具来迁移架构。

热数据

频繁访问的数据，例如实时数据或近期的事务数据。这类数据通常需要高性能的存储层级或存储类别，以提供快速的查询响应。

热修复

针对生产环境中关键问题的紧急修复。由于其紧迫性，热修复通常在典型的 DevOps 发布工作流之外进行。

超关怀期

紧接切换之后的一段时间，在此期间迁移团队在云中管理并监控已迁移的应用程序，以便解决任何问题。通常，这一时期持续 1 – 4 天。超关怀期结束时，迁移团队通常会将应用程序的责任移交给云运维团队。

I

IaC

参见“基础设施即代码”。

基于身份的策略

附加到一个或多个 IAM 主体的策略，用于定义其在 AWS 云环境中的权限。

闲置应用程序

在 90 天内平均 CPU 和内存使用率介于 5% 到 20% 之间的应用程序。在迁移项目中，通常会将这些应用程序退役或继续保留在本地。

IIoT

参见工业物联网。

不可变基础设施

一种为生产工作负载部署全新基础设施的模型，而不是对现有基础设施进行更新、打补丁或修改。与可变基础设施相比，不可变基础设施在一致性、可靠性和可预测性方面具有天然优势。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 (ingress) VPC

在 AWS 多账户架构中，用于接收、检查并路由来自应用外部的网络连接的 VPC。AWS Security Reference Architecture 建议在你的 Network 账户中设置入站、出站和检查 VPC，以保护你的应用与更广泛互联网之间的双向接口。

增量迁移

一种切换策略，即将应用程序分小部分迁移，而不是进行一次性、完整的切换。例如，你可能最初只将少量微服务或用户迁移到新系统。在验证一切正常运行后，你可以逐步迁移更多微服务或用户，直到可以下线你的遗留系统。该策略可降低与大规模迁移相关的风险。

工业 4.0

Klaus Schwab 于 2016 年提出的一个术语，用于指通过连接性、实时数据、自动化、分析以及 AI/ML 等方面的进步，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)

通过一组配置文件来预置和管理应用程序基础设施的过程。IaC 旨在帮助你集中化管理基础设施、标准化资源并快速扩展，从而使新环境可重复、可靠且一致。

工业物联网 (IIoT)

在工业领域（如制造、能源、汽车、医疗保健、生命科学和农业）中使用联网传感器和设备。有关更多信息，请参阅 [Building an industrial Internet of Things \(IIoT\) digital transformation strategy](#)。

检查 VPC

在 AWS 多账户架构中，用于集中管理 VPC 之间（同一或不同 AWS 区域）、互联网与本地网络之间网络流量检查的 VPC。AWS Security Reference Architecture 建议在你的 Network 账户中设置入站、出站和检查 VPC，以保护你的应用程序与更广泛互联网之间的双向接口。

物联网 (IoT)

由嵌入式传感器或处理器的互联物理对象构成的网络，这些对象通过互联网或本地通信网络与其他设备和系统进行通信。更多信息，请参阅 [What is IoT?](#)

可解释性

机器学习模型的一种特性，用于描述人类理解模型预测如何依赖其输入的程度。更多信息，请参阅 [Machine learning model interpretability with AWS](#)。

物联网

参见 [Internet of Things](#)。

IT 信息库 (ITIL)

一套用于交付 IT 服务并使这些服务与业务需求保持一致的最佳实践。ITIL 为 ITSM 提供基础。

IT 服务管理 (ITSM)

与为组织设计、实施、管理和支持 IT 服务相关的活动。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)

一种强制访问控制 (MAC) 的实现方式，其中用户和数据本身都会被明确分配一个安全标签值。用户安全标签与数据安全标签的交集决定用户可以查看哪些行和列。

着陆区

着陆区是一个架构良好、可扩展且安全的多账户 AWS 环境。这是一个起点，组织可以从这里快速启动并部署工作负载和应用程序，并对其安全性和基础设施环境充满信心。有关着陆区的更多信息，请参阅《设置安全且可扩展的多账户 AWS 环境》。

大型语言模型 (LLM)

一种深度学习 AI 模型，已在海量数据上进行预训练。LLM 可以执行多种任务，例如回答问题、总结文档、将文本翻译成其他语言以及补全句子。有关更多信息，请参阅《什么是 LLMs》。

大型迁移

迁移 300 台或更多服务器。

LBAC

请参阅基于标签的访问控制。

最小权限

一种安全最佳实践，即仅授予执行任务所需的最低权限。有关更多信息，请参阅 IAM 文档中的“应用最小权限”。

原样迁移

请参阅 7 Rs。

小端系统

一种先存储最低有效字节的系统。另请参阅字节序。

LLM

参见大型语言模型。

较低环境

参见 环境。

M

机器学习 (ML)

一种人工智能，使用算法和技术进行模式识别和学习。ML 会分析并从记录的数据（例如物联网 (IoT) 数据）中学习，以基于模式生成统计模型。有关更多信息，请参阅 Machine Learning。

主分支

查看分支。

恶意软件

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

托管服务

AWS 负责运营基础设施层、操作系统和平台的 AWS 服务，而你通过访问端点来存储和检索数据。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是托管服务的示例。这些也

被称为抽象化服务。

制造执行系统 (MES)

一种软件系统，用于在车间现场跟踪、监控、记录并控制将原材料转化为成品的生产过程。

MAP

参见 Migration Acceleration Program。

机制

一个完整的流程：你创建一个工具，推动该工具的采用，然后检查结果以便进行调整。机制是一个在运行过程中自我强化并自我改进的循环。有关更多信息，请参阅《AWS Well-Architected Framework》中的“构建机制”。

成员账户

在 AWS Organizations 中，组织内除管理账户之外的所有 AWS 账户。一个账户在同一时间只能属于一个组织。

MES

参见制造执行系统。

消息队列遥测传输 (MQTT)

一种轻量级的机器对机器 (M2M) 通信协议，基于发布/订阅模式，适用于资源受限的物联网设备。

微服务

一种小型、独立的服务，通过定义明确的 API 进行通信，通常由小型、自包含的团队负责。例如，一个保险系统可能包含与业务能力相对应的微服务，例如销售或市场营销，或与子域相对应的微服务，例如采购、理赔或分析。微服务的优势包括敏捷性、灵活扩展、易于部署、可复用代码以及弹性。有关更多信息，请参阅 Integrating microservices by using AWS serverless services。

微服务架构

一种构建应用程序的方法，由独立组件组成，并将每个应用程序进程作为一个微服务运行。这些微服务通过使用轻量级 API 的定义明确的接口进行通信。在这种架构中，每个微服务都可以更新、部署、

并可按需扩展以满足应用程序特定功能的需求。有关更多信息，请参阅《在 AWS 上实施微服务》。

迁移加速计划 (MAP)

一项 AWS 计划，提供咨询支持、培训和服务，帮助组织为迁移到云端建立强大的运营基础，并帮助抵消迁移的初始成本。MAP 包括一种迁移方法论，用于以系统化方式执行遗留系统迁移，以及一套用于自动化并加速常见迁移场景的工具。

大规模迁移

以波次方式将应用程序组合中的大部分迁移到云端的过程，并且在每一波中以更快的速度迁移更多应用程序。此阶段利用前几个阶段的最佳实践和经验教训，建立由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付来简化工作负载迁移。这是 AWS 迁移策略的第三阶段。

迁移工厂

通过自动化、敏捷的方法简化工作负载迁移的跨职能团队。迁移工厂团队通常包括运维人员、业务分析师和负责人、迁移工程师、开发人员以及以冲刺方式协作的 DevOps 专业人员。企业应用程序组合中有 20% 到 50% 由可重复的模式构成，可通过工厂化方法进行优化。有关更多信息，请参阅本内容集中关于迁移工厂的讨论以及《Cloud Migration Factory》指南。

迁移元数据

完成迁移所需的有关应用程序和服务器的信息。每种迁移模式都需要一组不同的迁移元数据。迁移元数据示例包括目标子网、安全组和 AWS 账户。

迁移模式

一种可重复执行的迁移任务，详细说明迁移策略、迁移目标，以及所使用的迁移应用程序或服务。示例：使用 AWS Application Migration Service 将工作负载重新托管 (Rehost) 迁移到 Amazon EC2。

迁移组合评估 (MPA)

一款在线工具，提供用于验证迁移到 AWS 云的业务案例的信息。MPA 提供详细的资产组合评估（服务器规格优化、定价、TCO

对比、迁移成本分析）以及迁移规划（应用程序数据分析与数据收集、应用程序分组、迁移优先级排序和波次规划）。MPA 工具（需要登录）向所有 AWS 顾问和 APN 合作伙伴顾问免费提供。

迁移就绪性评估（MRA）

使用 AWS CAF 获取对组织云就绪性状态的洞察，识别优势与劣势，并制定行动计划以弥补已识别的差距。有关更多信息，请参阅迁移就绪性指南。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 云所采用的方法。有关更多信息，请参阅本词汇表中的 7 Rs 条目，并参阅《动员您的组织以加速大规模迁移》。

ML

请参阅机器学习。

现代化

将过时的（遗留或单体）应用程序及其基础设施转型为云中敏捷、弹性且高可用的系统，以降低成本、提升效率并利用创新。有关更多信息，请参阅《在 AWS 云中实现应用程序现代化的策略》。

现代化就绪性评估

一种评估，用于帮助确定组织应用程序的现代化就绪性；识别收益、风险和依赖关系；并确定组织对这些应用程序未来状态的支持能力。评估的产出包括目标架构蓝图、详细说明现代化过程开发阶段与里程碑的路线图，以及用于弥补已识别差距的行动计划。有关更多信息，请参阅《在 AWS 云中评估应用程序的现代化就绪性》。

单体应用程序（单体）

作为单一服务运行且各个进程紧密耦合的应用程序。单体应用有几个缺点。如果某个应用功能的需求出现激增，就必须对整个架构进行扩展。随着代码库的增长，为单体应用添加或改进功能也会变得更加复杂。为了解决这些问题，你可以

使用微服务架构。有关更多信息，请参阅《将单体应用分解为微服务》。

MPA

请参阅 Migration Portfolio Assessment。

MQTT

请参阅 Message Queuing Telemetry Transport。

多类别分类

一种有助于为多个类别生成预测的过程（预测两个以上结果中的一个）。例如，机器学习模型可能会问：“这个产品是书、汽车还是手机？”或“哪个产品类别对这位客户最感兴趣？”

可变基础设施

一种用于更新和修改现有基础设施以支持生产工作负载的模型。为了提高一致性、可靠性和可预测性，AWS Well-Architected Framework 建议将不可变基础设施作为最佳实践来使用。

O

OAC

请参阅 origin access control。

OAI

请参阅源访问身份（origin access identity）。

OCM

请参阅组织变更管理（organizational change management）。

离线迁移

一种迁移方法，在迁移过程中会将源工作负载停机。
该方法涉及较长的停机时间，通常用于小型、非关键的工作负载。

OI

请参阅 operations integration。

OLA

参见 operational-level agreement。

在线迁移

一种迁移方法，在不将源工作负载下线的情况下，将其复制到目标系统。在迁移期间，连接到该工作负载的应用程序可以继续运行。该方法可实现零到最小停机时间，通常用于关键生产工作负载。

OPC-UA

参见 Open Process Communications - Unified Architecture。

Open Process Communications - Unified Architecture (OPC-UA)

一种用于工业自动化的机器对机器 (M2M) 通信协议。OPC-UA 提供了一个互操作性标准，并具备数据加密、身份验证和授权机制。

运营级别协议 (OLA)

一种协议，用于明确各职能 IT 团队为支持服务级别协议 (SLA) 而相互承诺交付的内容。

运营就绪性评审 (ORR)

一份问题清单及其相关最佳实践，可帮助你了解、评估、预防或减少事件以及可能发生的故障范围。更多信息，请参阅 AWS Well-Architected Framework 中的《Operational Readiness Reviews (ORR)》。

运营技术 (OT)

与物理环境协同工作、用于控制工业运营、设备和基础设施的硬件与软件系统。在制造业中，OT 与信息技术 (IT) 系统的集成是工业 4.0 转型的关键重点。

运营集成 (OI)

在云中实现运营现代化的过程，涉及就绪性规划、自动化和集成。更多信息，请参阅运营集成指南。

组织跟踪

由 AWS CloudTrail 创建的一种跟踪，用于记录 AWS Organizations 中某个组织内所有 AWS 账户的所有事件。该跟踪会在属于该组织的每个 AWS 账户中创建，该组织的每个 AWS 账户中都会创建该跟踪。

组织并跟踪每个账户中的活动。有关更多信息，请参阅 CloudTrail 文档中的“为组织创建跟踪 (trail)”。

组织变更管理 (OCM)

一种从人员、文化和领导力视角管理重大、颠覆性业务转型的框架。OCM 通过加速变更采纳、解决过渡期问题，并推动文化与组织层面的变革，帮助组织为新系统和新战略做好准备并完成过渡。在 AWS 迁移策略中，该框架被称为“人员加速 (people acceleration)”，因为云采用项目所需的变更速度很快。

有关更多信息，请参阅 OCM 指南。

源站访问控制 (OAC)

在 CloudFront 中，用于限制访问以保护你的 Amazon Simple Storage Service (Amazon S3) 内容的增强选项。OAC 支持所有 AWS 区域中的所有 S3 存储桶、使用 AWS KMS 的服务器端加密 (SSE-KMS)，以及对 S3 存储桶的动态 PUT 和 DELETE 请求。

源站访问身份 (OAI)

在 CloudFront 中，用于限制访问以保护你的 Amazon S3 内容的一个选项。使用 OAI 时，CloudFront 会创建一个 Amazon S3 可用于进行身份验证的主体。通过身份验证的主体只能通过特定的 CloudFront 分配访问 S3 存储桶中的内容。

另请参阅 OAC，它提供更精细且增强的访问控制。

ORR

请参阅 operational readiness review。

OT

请参阅运营技术。

出站 (egress) VPC

在 AWS 多账户架构中，指用于处理从应用程序内部发起的网络连接的 VPC。AWS 安全参考架构建议在你的网络账户中设置入站、出站和检查 VPC，以保护应用程序与更广泛互联网之间的双向接口。

P

权限边界

一种附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可拥有的最大权限。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)

当直接查看或与其他相关数据配对时，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。

行动手册

一组预定义步骤，用于记录与迁移相关的工作，例如在云中交付核心运营职能。行动手册可以采用脚本、自动化运行手册的形式，或以流程摘要或步骤摘要的形式呈现，用于说明运行现代化环境所需的流程或步骤。

PLC

请参阅可编程逻辑控制器。

PLM

请参阅产品生命周期管理。

策略

一种对象，可用于定义权限（参见基于身份的策略）、指定访问条件（参见基于资源的策略），或在 AWS Organizations 中为组织内所有账户定义最大权限（参见服务控制策略）。

多语言持久化

根据数据访问模式和其他需求，独立为某个微服务选择其数据存储技术。如果你的微服务使用相同的数据存储技术，可能会遇到实现方面的挑战或出现性能不佳的情况。如果微服务使用数据存储，则更容易实现，并能获得更好的性能和可扩展性。

最能适配其需求。有关更多信息，请参阅《在微服务中启用数据持久性》。

应用组合评估

一种用于发现、分析并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《评估迁移就绪情况》。

谓词

返回真或假值的查询条件，通常位于 WHERE 子句中。

谓词下推

一种数据库查询优化技术，在传输之前先在查询中对数据进行过滤。这减少了必须从关系型数据库检索和处理的数据量，并提升查询性能。

预防性控制

一种旨在防止事件发生的安全控制。这些控制是第一道防线，有助于防止对网络的未授权访问或不希望发生的更改。有关更多信息，请参阅《Implementing security controls on AWS》中的 Preventative controls。

主体

AWS 中能够执行操作并访问资源的实体。该实体通常是 AWS 账户的根用户、IAM 角色或用户。有关更多信息，请参阅 IAM 文档中《Roles terms and concepts》里的 Principal。

设计即隐私 (privacy by design)

一种系统工程方法，在整个开发过程中都将隐私纳入考量。

私有托管区域

一种容器，用于保存你希望 Amazon Route 53 如何响应一个或多个 VPC 内某个域及其子域的 DNS 查询的信息。有关更多信息，请参阅 Route 53 文档中的《使用私有托管区域》。

主动控制

一种旨在防止部署不合规资源的安全控制。这些控制会在资源预置之前扫描资源。如果资源不符合该控制要求，则不会进行预置。有关更多信息，请参阅《Controls 参考指南》中的相关内容。

查阅 AWS Control Tower 文档，并参见《在 AWS 上实施安全控制》中的“主动控制”。

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进行管理，从设计、开发和发布，到增长和成熟，再到衰退和下线。

生产环境

参见 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流程自动化。

提示链

将一个 LLM 提示的输出作为下一个提示的输入，以生成更好的响应。这种技术用于将复杂任务拆分为子任务，或对初步响应进行迭代式优化或扩展。它有助于提高模型响应的准确性和相关性，并实现更细粒度、更个性化的结果。

假名化

在数据集中用占位值替换个人标识符的过程。假名化有助于保护个人隐私。假名化数据仍被视为个人数据。

发布/订阅 (pub/sub)

一种支持微服务之间异步通信的模式，用于提升可扩展性和响应能力。例如，在基于微服务的 MES 中，一个微服务可以将事件消息发布到某个通道，其他微服务可以订阅该通道。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

一系列类似指令的步骤，用于在 SQL 关系数据库系统中访问数据。

查询计划回归

当数据库服务的优化器在数据库环境发生某项变更后，选择了一个不如之前最优的执行计划。其原因可能包括统计信息、约束、环境设置、查询参数绑定的变化，以及数据库引擎的更新。

R

RACI 矩阵

参见 responsible, accountable, consulted, informed (RACI)。

RAG

请参阅检索增强生成 (Retrieval Augmented Generation) 。

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，直到支付赎金为止。

RASCI 矩阵

参见负责、问责、咨询、知会 (RACI) 。

RCAC

参见行和列访问控制。

只读副本

用于只读目的的数据库副本。你可以将查询路由到只读副本，以降低主数据库的负载。

重新架构

参见 7 Rs。

恢复点目标 (RPO)

自上次数据恢复点以来可接受的最长时间。这决定了在上次恢复点与服务中断之间，何种数据丢失量被视为可接受。

恢复时间目标 (RTO)

服务中断与服务恢复之间可接受的最大延迟。

重构

参见 7 Rs。

区域

地理区域内的一组 AWS 资源。每个 AWS 区域都与其他区域隔离且相互独立，以提供容错能力、稳定性和弹性。有关更多信息，请参阅 [Specify which AWS Regions your account can use](#)。

回归

一种预测数值的机器学习技术。例如，为了解决“这套房子会以什么价格售出？”这一问题，机器学习模型可以使用线性回归模型，根据关于房屋的已知事实（例如，建筑面积）来预测房屋的售价。

重新托管

参见 7 Rs。

发布

在部署过程中，将变更提升到生产环境的行为。

重新迁移

参见 7 Rs。

重新平台化

参见 7 Rs。

回购

参见 7 Rs。

弹性

应用程序抵御或从中断中恢复的能力。在 AWS 云中规划弹性时，高可用性和灾难恢复是常见的考量因素。有关更多信息，请参阅 [AWS Cloud Resilience](#)。

基于资源的策略

附加到资源（例如 Amazon S3 存储桶、端点或加密密钥）上的策略。这类策略指定允许访问的主体、支持的操作，以及必须满足的任何其他条件。

负责、问责、咨询、知会（RACI）矩阵

用于定义参与迁移活动和云运营的所有相关方的角色与职责的矩阵。该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）和知会（I）。支持（S）类型为可选项。如果包含支持，则该矩阵称为 RASCI 矩阵；如果不包含，则称为 RACI 矩阵。

响应式控制

一种安全控制，旨在推动对不利事件或偏离安全基线情况的修复。有关更多信息，请参阅《在 AWS 上实施安全控制》中的“响应式控制”。

保留

参见 7 Rs。

退役

参见 7 Rs。

检索增强生成（RAG）

一种生成式 AI 技术，其中 LLM 在生成响应之前，会引用其训练数据源之外的权威数据源。例如，RAG 模型可能会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 What is RAG。

轮换

定期更新密钥的过程，以使攻击者更难访问凭证。

行和列访问控制（RCAC）

使用已定义访问规则的基础且灵活的 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组手动或自动化流程。这些通常用于简化重复性操作或错误率较高的流程。

S

SAML 2.0

许多身份提供商（IdP）使用的一种开放标准。此功能支持联合单点登录（SSO），因此用户可以登录 AWS Management Console 或调用 AWS API 操作，而无需你为组织中的每个人都在 IAM 中创建用户。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 [About SAML 2.0-based federation](#)。

SCADA

参见监控与数据采集。

SCP

参见服务控制策略。

密钥

在 AWS Secrets Manager 中，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 [《Secrets Manager 密钥中包含什么？》](#)。

安全内建

一种系统工程方法，在整个开发过程中都将安全性纳入考量。

安全控制

一种技术或管理性的防护措施，用于防止、检测或降低威胁行为者利用安全漏洞的能力。安全控制主要有四种类型：预防性、侦测性、响应性和主动性。

安全加固

通过减少攻击面来提高对攻击的抵抗力的过程。这可能包括移除不再需要的资源、实施授予最小权限这一安全最佳实践，或在配置文件中停用不必要的功能等操作。

安全信息与事件管理（SIEM）系统

将安全信息管理（SIM）与安全事件管理（SEM）系统相结合的工具和服务。SIEM 系统从服务器、网络、设备及其他来源收集、监控并分析数据，以检测威胁和安全漏洞，并生成告警。

安全响应自动化

一种预定义并编程的操作，旨在自动响应或修复安全事件。这些自动化可作为侦测型或响应型安全控制，帮助你落实 AWS 安全最佳实践。自动化响应操作的示例包括修改 VPC 安全组、为 Amazon EC2 实例打补丁或轮换凭证。

服务器端加密

由接收数据的 AWS 服务在数据目的地对数据进行加密。

服务控制策略（SCP）

一种策略，用于在 AWS Organizations 中对组织内所有账户的权限进行集中控制。SCP 定义防护栏或设置管理员可委派给用户或角色的操作上限。你可以将 SCP 用作允许列表或拒绝列表，以指定哪些服务或操作被允许或被禁止。有关更多信息，请参阅 AWS Organizations 文档中的 Service control policies。

服务端点

AWS 服务入口点的 URL。你可以使用该端点以编程方式连接到目标服务。有关更多信息，请参阅《AWS 通用参考》中的 AWS 服务端点。

《AWS 通用参考》。

服务级别协议 (SLA)

一种协议，用于明确 IT 团队承诺向客户交付的内容，例如服务正常运行时间和性能。

服务级别指标 (SLI)

对服务某个性能方面的度量，例如其错误率、可用性或吞吐量。

服务级别目标 (SLO)

一个目标指标，用于表示服务的健康状况，并通过服务级别指标进行衡量。

共享责任模型

一种模型，用于描述你与 AWS 在云安全与合规方面共同承担的责任。AWS 负责云的安全，而你负责云中的安全。有关更多信息，请参阅“共享责任模型”。

SIEM

请参阅安全信息与事件管理系统。

单点故障 (SPOF)

应用程序中某个单一关键组件发生的故障，可能会导致系统中断。

SLA

请参阅服务级别协议。

SLI

请参阅服务级别指标。

SLO

参见服务级别目标。

拆分并播种模型

一种用于扩展和加速现代化项目的模式。随着新功能和产品版本的定义，核心团队会拆分以组建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员生产力，并支持快速

创新。有关更多信息，请参阅《在 AWS 云中对应用程序进行现代化改造的分阶段方法》。

SPOF

请参阅单点故障。

星型架构

一种数据库组织结构，使用一个大型事实表来存储事务性或度量数据，并使用一个或多个较小的维度表来存储数据属性。该结构旨在用于数据仓库或商业智能用途。

绞杀者无花果模式

一种通过逐步重写并替换系统功能来实现单体系统现代化的方法，直到可以停用遗留系统。该模式借用了无花果藤蔓的类比：藤蔓在既有树木上生长，最终压制并取代其宿主。该模式由 Martin Fowler 提出，用于在重写单体系统时管理风险。有关如何应用此模式的示例，请参阅 *Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway*。

子网

VPC 中的一段 IP 地址范围。子网必须位于单个可用区内。

监控与数据采集 (SCADA)

在制造业中，一种使用硬件和软件来监控物理资产和生产运营的系统。

对称加密

一种加密算法，使用同一把密钥对数据进行加密和解密。

合成测试

以模拟用户交互的方式对系统进行测试，以检测潜在问题或监控性能。你可以使用 Amazon CloudWatch Synthetics 来创建这些测试。

系统提示词

一种向 LLM 提供上下文、指令或指南以引导其行为的技术。

系统提示有助于设置上下文并为与用户的交互建立规则。

T

标签

作为元数据的键值对，用于组织你的 AWS 资源。标签可以帮助你管理、识别、组织、搜索和筛选资源。有关更多信息，请参阅为你的 AWS 资源添加标签。

目标变量

在监督式机器学习中你试图预测的值。这也称为结果变量。例如，在制造场景中，目标变量可以是产品缺陷。

任务列表

用于跟踪运行手册执行进度的工具。任务列表包含运行手册的概览以及需要完成的一般任务列表。对于每项一般任务，它包括所需的预计时间、负责人和进度。

测试环境

参见 环境。

培训

为你的机器学习模型提供可供学习的数据。训练数据必须包含正确答案。学习算法会在训练数据中找到将输入数据属性映射到目标（你想要预测的答案）的模式。它会输出一个捕获这些模式的机器学习模型。然后，你可以使用该机器学习模型对你不知道目标的新数据进行预测。

transit gateway

一个网络传输枢纽，你可以用它来互连你的 VPC 和本地网络。有关更多信息，请参阅 AWS Transit Gateway 文档中的“什么是 transit gateway”。

基于主干的工作流

一种方法是开发人员在功能分支中本地构建并测试功能，然后将这些更改合并到主分支。随后，主分支会按顺序构建并部署到开发、预生产和生产环境中。

受信任的访问

向你指定的服务授予权限，使其能够在 AWS Organizations 中以及在其账户内代表你在组织中执行任务。受信任的服务会在每个账户中在需要该角色时创建一个服务关联角色，以便为你执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的《将 AWS Organizations 与其他 AWS 服务一起使用》。

调优

更改训练过程的某些方面以提高机器学习模型的准确性。例如，你可以通过生成标注集来训练机器学习模型、添加标签，然后在不同设置下多次重复这些步骤以优化模型。

两块披萨团队

一个小型 DevOps 团队，两块披萨就能喂饱。两块披萨团队的规模可确保在软件开发中获得尽可能好的协作机会。

U

不确定性

一个概念，指不精确、不完整或未知的信息，这些信息可能会削弱预测性机器学习模型的可靠性。不确定性有两种类型：认知不确定性（Epistemic uncertainty）由有限、不完整的数据导致，而偶然不确定性（aleatoric uncertainty）由数据中固有的噪声和随机性导致。有关更多信息，请参阅《Quantifying uncertainty in deep learning systems》指南。

无差异化任务

也称为“繁重工作”，指创建和运行应用程序所必需但不会直接为最终用户提供价值或带来竞争优势的工作。无差异化任务的示例包括采购、维护和容量规划。

上层环境

参见 环境。

V

清理

一种数据库维护操作，涉及在增量更新后进行清理，以回收存储空间并提升性能。

版本控制

用于跟踪更改的流程和工具，例如跟踪存储库中源代码的更改。

VPC 对等连接

两个 VPC 之间的连接，使你能够使用私有 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

危及系统安全的软件或硬件缺陷。

W

热缓存

包含当前、相关且经常被访问的数据的缓冲缓存。数据库实例可以从缓冲缓存中读取数据，这比从主内存或磁盘读取更快。

热数据

不经常访问的数据。查询此类数据时，通常可以接受查询速度适度较慢。

窗口函数

一种 SQL 函数，用于对与当前记录在某种方式上相关的一组行执行计算。窗口函数适用于处理任务，例如计算移动平均值，或根据相对于当前行的位置访问各行的值。

工作负载

一组能够交付业务价值的资源和代码，例如面向客户的应用程序或后端流程。

工作流

迁移项目中的职能小组，负责一组特定任务。每个工作流相互独立，但会在项目中支持其他工作流。例如，组合工作流负责对应用程序进行优先级排序、波次规划以及收集迁移元数据。组合工作流将这些资产交付给迁移工作流，后者随后迁移服务器和应用程序。

WORM

参见 write once, read many。

WQF

请参阅 AWS Workload Qualification Framework。

一次写入，多次读取 (WORM)

一种存储模型，数据只写入一次，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，但不能更改数据。该数据存储基础设施被视为不可变。

Z

零日漏洞利用

一种攻击（通常是恶意软件），利用零日漏洞发起攻击。

零日漏洞

生产系统中尚未缓解的缺陷或漏洞。威胁行为者可以利用此类漏洞攻击系统。开发人员往往是在攻击发生后才意识到该漏洞的存在。

零样本提示

向 LLM 提供执行某项任务的指令，但不提供任何可用于引导它的示例（样本）。LLM 必须使用其预训练知识来处理该任务。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，通常会淘汰这些应用程序。