AWS规范性指南

使用架构决策记录简化软件开发项目的技术决策

AWS规范性指南: 使用架构决策记录简化软件开发项目的技术决策

Table of Contents

使用架构决策记录简化软件开发项目的技术决策

应用程序架构师大流士 Kunce 和多米尼克·戈比AWS专业服务

2022 年 3 月

本指南介绍了软件工程项目的体系结构决策记录 (ADR) 流程。ADR 支持团队协调，记录项目或产品的战略方向，并减少反复出现且耗时的决策工作。

在项目和产品开发期间，软件工程团队需要做出体系结构决策才能实现自己的目标。这些决策可以是技术性的，例如决定使用命令查询责任分离 (CQRS) 模式，也可以是与流程相关的，例如决定使用 GitFlow 管理源代码的工作流程。做出这些决策是一个耗时且困难的过程。团队必须证明、记录这些决策并将其传达给相关利益相关者。

在做出建筑决策时，经常出现三种主要的反模式：

- 由于害怕做出错误的选择，根本没有做出任何决定。
- 决定是毫无道理的，人们也不明白为什么做出决定。这导致同一主题被多次讨论。
- 决策没有在架构决策存储库中捕获，因此团队成员忘记或不知道决策是做出的。

在产品或项目的开发过程中，这些反模式特别重要。

以 ADR 的形式获取导致决策的背景和考虑因素，使当前和 future 的利益相关者能够收集有关作出的决策和每项决策背后的思考过程的信息。这可以缩短软件开发时间，并为 future 团队提供更好的文档。

目标业务成果

ADR 的目标是三个业务成果：

- 他们协调当前和 future 的团队成员。
- 他们为项目或产品设定了战略方向。
- 他们通过定义正确记录和传达架构决策的流程来避免决策反模式。

发展成果评估记录了通知 future 利益相关者的决定的背景。ADR 集合提供了移交经验和参考文档。团队或项目成员使用 ADR 集合进行后续项目和产品功能规划。能够参考 ADR 可以减少开发、审查和体系结构决策过程中所需的时间。ADR 还允许其他团队学习并深入了解其他项目和产品开发团队的考虑因素。

ADR 过程

架构决策记录 (ADR) 是一份文档，它描述了团队对他们计划构建的软件体系结构的一个重要方面所作的选择。每个 ADR 都描述了体系结构决策、背景及其后果。ADR 有状态，因此遵循生命周期。有关 ADR 的示例，请参阅附录 (p. 9).

ADR 流程输出了一系列体系结构决策记录。此集合创建决策日志。决策日志提供了项目上下文以及详细的实施和设计信息。项目成员跳过了每个 ADR 的头条新闻，以获得项目背景的概览。他们阅读了 ADR，深入了解项目实施和设计选择。

当团队接受 ADR 时，它变得不可变。如果新见解需要做出不同的决定，该团队会提出新的 ADR。当团队接受新的 ADR 时，它将取代之前的 ADR。

ADR 过程的范围

项目成员应为影响软件项目或产品的每个具有体系结构意义的决策创建 ADR，包括以下内容（理查兹和福特 2020 (p. 8)）：

- 结构（例如，微服务等模式）
- 非功能性要求（安全性、高可用性和容错能力）
- 依赖关系（组件的耦合）
- 接口（API 和已发布的合同）
- 构造技术（库、框架、工具和流程）

功能性和非功能性要求是 ADR 过程中最常见的投入。

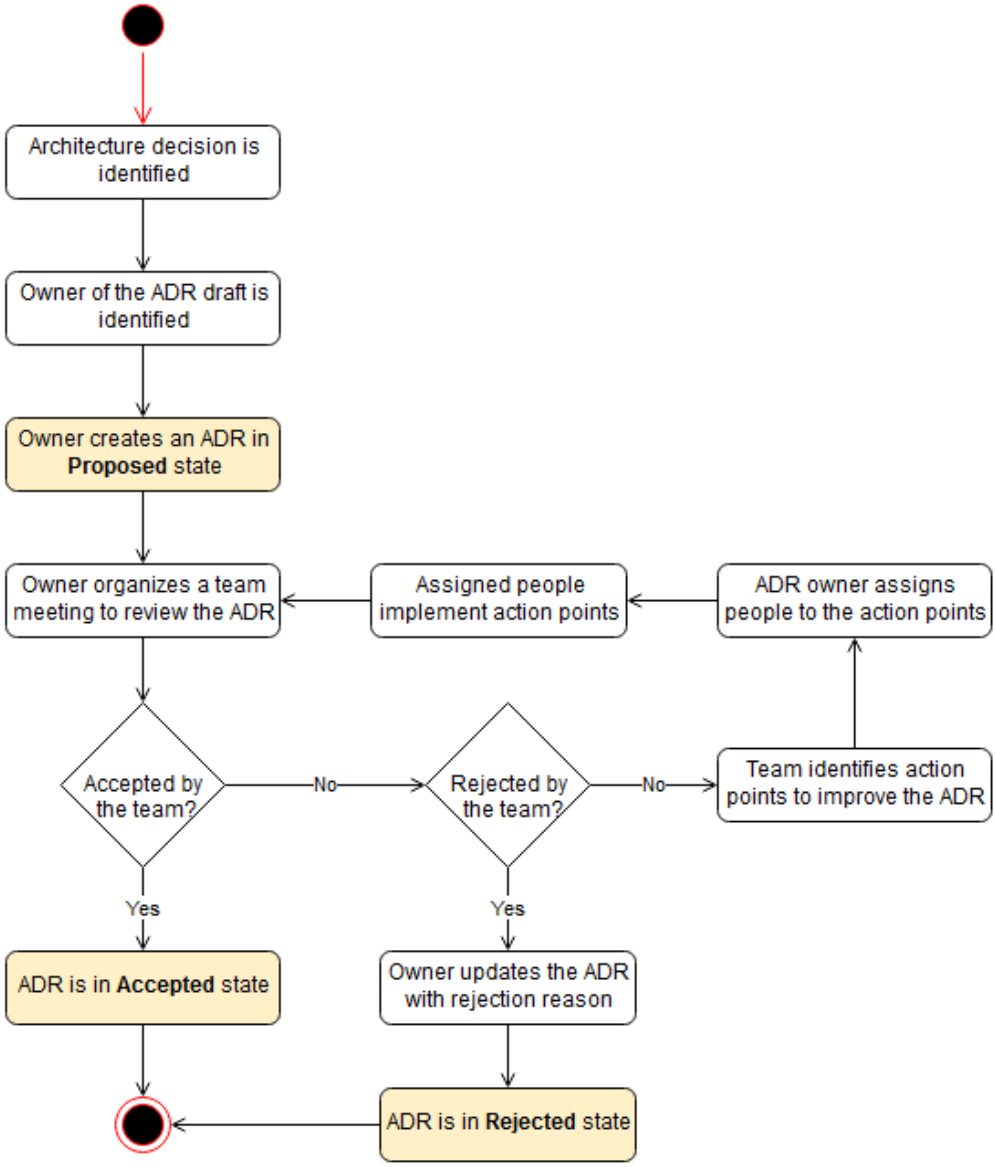
ADR 内容

当团队确定需要 ADR 时，团队成员开始基于项目范围的模板编写 ADR。（请参阅ADR GitHub 组织例如模板。）该模板简化了 ADR 的创建，并确保 ADR 捕获所有相关信息。至少，每个 ADR 都应定义决定的上下文、决定本身以及决定对项目及其交付成果的后果。（有关这些部分的示例，请参阅附录 (p. 9)。）ADR 结构最强大的方面之一是，它侧重于做出决定的原因，而不是团队如何实施决策。了解团队做出决定的原因使其他团队成员更容易采纳决策，并防止其他未参与决策过程的架构师在 future 推翻该决定。

ADR 采用流程

每个团队成员都可以创建 ADR，但团队应该为 ADR 确定所有权的定义。作为 ADR 所有者的每位作者都应积极维护和传达 ADR 内容。为了澄清此所有权，本指南将 ADR 作者称为ADR 拥有者请参阅以下各节。其他团队成员始终可以为 ADR 做出贡献。如果 ADR 的内容在团队接受 ADR 之前发生了变化，则所有者应批准这些更改。

下图显示了 ADR 的创建、拥有权和采用流程。



团队确定体系结构决策及其所有者之后，ADR 所有者在提议的在进程开始时的状态。中的 ADR提议的州已准备好进行审查。

然后，ADR 所有者启动 ADR 的审核流程。ADR 审核流程的目标是决定团队是否接受 ADR、确定它需要返工还是拒绝 ADR。包括所有者在内的项目团队审查 ADR。审查会议应该从专门的时间段开始阅读 ADR。平均而言，10 到 15 分钟应该足够了。在此期间，每个团队成员阅读文档并添加评论和问题以标记不明确的主题。审核阶段结束后，ADR 所有者会宣读并与团队讨论每条评论。

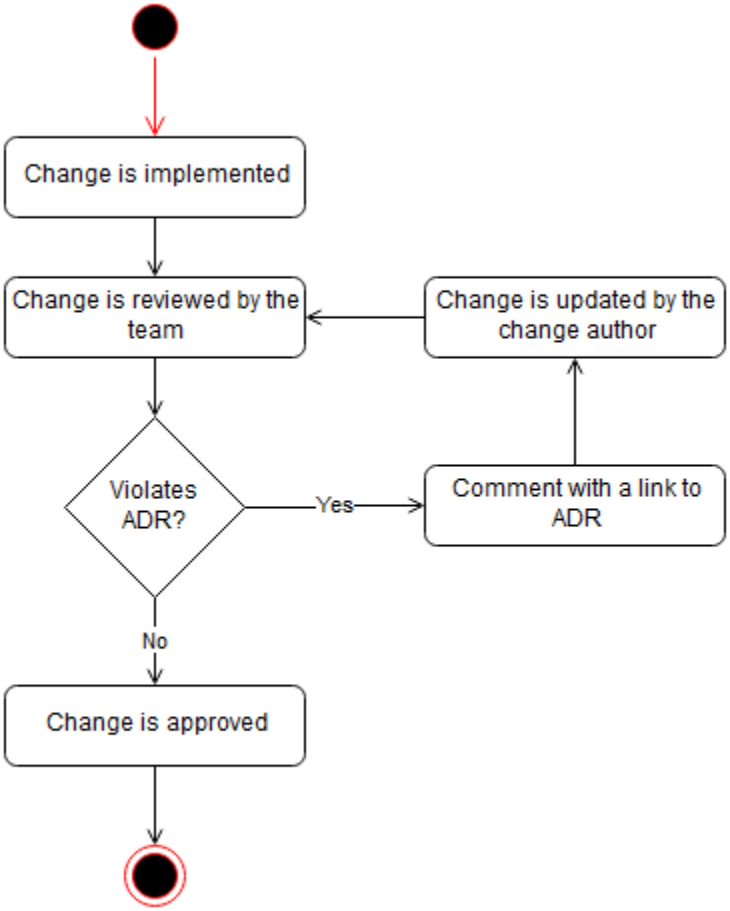
如果团队找到改善 ADR 的行动点，则 ADR 的状态将保持不变提议的. ADR 所有者制定操作，并与团队合作，为每个操作添加一个受让人。每个团队成员都可以贡献和解决行动要点。ADR 所有者有责任重新安排审核流程。

团队还可以决定拒绝 ADR。在这种情况下，ADR 所有者添加了拒绝的理由，以防止将 future 就同一主题进行讨论。所有者将 ADR 状态更改为已拒绝.

如果团队批准 ADR，则所有者将添加时间戳、版本和利益相关者列表。然后，所有者将州更新为已接受.

ADR 及其创建的决策日志代表团队做出的决策，并提供所有决策的历史记录。团队尽可能在代码和体系结构审查期间使用 ADR 作为参考。除了执行代码审查、设计任务和实施任务外，团队成员还应咨询 ADR 以获取产品的战略决策。

下图显示了应用 ADR 验证软件组件的更改是否符合商定决策的过程。

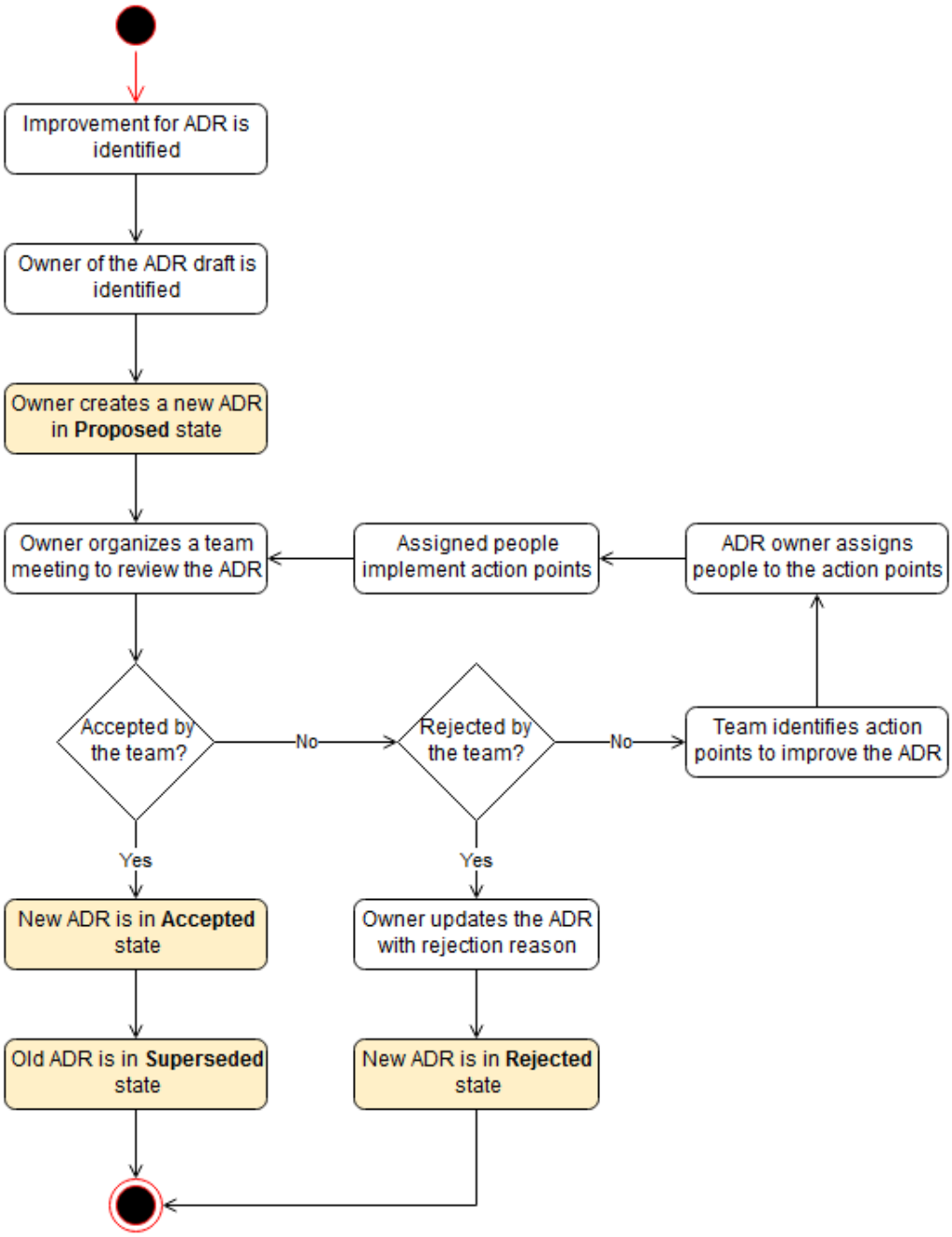


作为一种良好做法，每次软件更改都应经过同行审查，并且至少需要一次批准。在代码审阅期间，代码审阅者可能会发现违反一个或多个 ADR 的更改。在这种情况下，审阅者要求代码更改的作者更新代码，并共享指向 ADR 的链接。当作者更新代码时，它会得到同行审核者的批准并合并到主代码库中。

ADR 审核流程

在团队接受或拒绝 ADR 之后，团队应将其视为不可变的文档。对现有的 ADR 进行更改需要创建新的 ADR、为新的 ADR 建立审查流程以及批准 ADR。如果团队批准了新的 ADR，则所有者应将旧 ADR 的状态更改为已取代. 下图显示了更新过程。

ADR 审核流程


Improvement for ADR is identified
Owner of the ADR draft is identified
Owner creates a new ADR in **Proposed** state
Owner organizes a team meeting to review the ADR
Assigned people implement action points
ADR owner assigns people to the action points
Accepted by the team?
No
Rejected by the team?
No
Team identifies action points to improve the ADR
Yes
Yes
New ADR is in **Accepted** state
Owner updates the ADR with rejection reason
Old ADR is in **Superseded** state
New ADR is in **Rejected** state

最佳实践

促进所有权。应授权每个项目团队成员创建和拥有 ADR。这种做法将架构研究工作分配给团队成员，并卸载解决方案架构师或团队负责人的工作负担。它还促进了决策过程中的主人翁意识。这有助于团队更快地通过这些决策，而不是将它们视为组织更高级别强加的决策。

保留 ADR 历史记录。ADR 应该有变更历史记录，每次更改都应有所有者。当 ADR 所有者更新 ADR 时，他们应将旧 ADR 的状态更改为已取代，请注意他们在新 ADR 变更历史记录中的变化，并将旧的 ADR 保留在决策日志中。

安排定期审查会议。如果你在一个新的（绿地）项目中，ADR 过程在一开始就会相当紧张。我们建议您在每日站立之前或之后建立定期的 ADR 讨论和审查会议的节奏。通过这种方法，定义的 ADR 将在两到三个冲刺中稳定下来，并且您可以通过更少的会议建立坚实的基础。

将 ADR 存储在中心位置。每个项目成员都应有权访问 ADR 集合。我们建议您将 ADR 存储在中心位置，然后在项目文档的主页上参考它们。存储 ADR 有两种常见的选项：

- Git 存储库，这让您能够更轻松地对 ADR 进行版本
- Wiki 页面，使所有团队成员都可以访问 ADR

解决不合规的代码。ADR 流程不能解决不合规的旧代码的问题。如果您的旧代码不支持既定 ADR，则可以在引入新更改的同时逐步更新过时的代码库或工件，或者您的团队可以决定通过创建技术债务任务来明确重构代码。

常见问题

创建 ADR 流程有哪些优势？

项目团队应创建一个 ADR 流程，以简化架构决策，防止重复讨论相同的架构主题，并有效地沟通架构决策。

项目团队应该什么时候创建 ADR？

项目团队应为影响结构（例如微服务等模式）、非功能性要求（安全性、高可用性和容错能力）、依赖关系（组件的耦合）、接口（API 和已发布的合同）和构造的各个方面创建 ADR技术（库、框架、工具和流程）。

项目团队应该多久审查一次 ADR？

项目团队应在接受 ADR 之前至少审查一次。

谁应该创建 ADR？

每个团队成员都可以创建 ADR。我们建议您推广 ADR 的所有权概念。拥有 ADR 的作者应积极维护和传播 ADR 内容。其他团队成员始终可以为 ADR 做出贡献。ADR 所有者应批准对 ADR 的更改。

ADR 应该包含哪些信息？

至少，每个 ADR 都必须定义决定的上下文、决定本身以及决定对项目及其交付成果的后果。上下文应提及团队考虑的可能解决方案。它还应包含与项目、客户或技术堆栈相关的任何相关信息。该决定必须用当务之急的语言明确说明团队决定采取的解决方案。避免使用 “应该” 之类的词语，并将每个决定短语说 “我们使用...” 或 “团队必须使用...” 后果部分应提及作出决策的所有已知权衡。每个 ADR 必须具有状态和更改日志，其中包含变更日期和负责更改的人员。

在哪里可以找到 ADR 模板？

ADR 模板有多个版本和变体可供选择。有关常用的 ADR 模板的公开集合，请参阅ADR GitHub 知识库.

后续步骤

我们建议你从小开始，看看 ADR 为你的团队带来的好处。如果您正在进行中的项目，请确定下一个体系结构更改并应用建议的 ADR 流程来创建第一个 ADR。

另一个起点是使用 ADR 记录整个软件开发过程。通常，开发过程基于团队在任何文档中都没有获得的默认知识。记录此过程可为团队的新成员提供更顺畅的体验。

如果你在一个绿地项目中，应用 ADR 流程，从一开始就从几句话开始捕获所有决策。然后，你可以迭代这些 ADR 并用新信息补充它们。建立 ADR 之后，您可以开始在代码审查过程中使用它们作为参考。

资源

- 体系结构决策记录。https://adr.github.io/.
- 理查兹、马克和尼尔·福特。2020 年。软件架构的基础知识. 塞瓦斯托波尔：O'Reilly Media。

附录：示例 ADR

职务

该决策定义了 ABC 应用程序开发的软件开发生命周期方法。

状态

已接受

日期

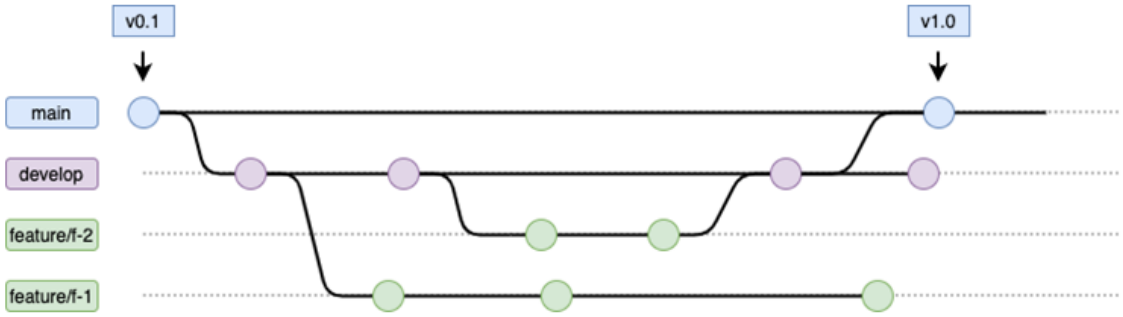
2022 年 3 月 11 日

上下文

ABC 应用程序是一种打包的解决方案，它将使用部署包部署到客户的环境中。我们需要有一个开发过程，使我们能够拥有可控的功能、修补程序和发布管道。

决策

我们使用的是改编版GitFlow 工作流程开发 ABC 应用程序。



为简单起见，我们不会使用`hotfix/*`和`release/*`分支，因为 ABC 应用程序将被打包而不是部署到特定环境中。出于这个原因，不需要额外的复杂性，这可能会阻止我们快速做出反应来修复生产版本中的错误，或者在单独的环境中测试版本。

以下是商定的分支策略：

- 每个存储库都必须有受保护`main`将用于标记发布版本的分支。
- 每个存储库都必须有受保护`develop`用于所有正在进行的开发工作。

后果

正值：

- 适配 GitFlow 进程将使我们能够控制 ABC 应用程序的发布版本控制。

负值：

- GitFlow 比基于中继的开发更复杂，或者 GitHub 流动并有更多的开销。

Compliance

- 这些区域有：`main`和`develop`每个仓库中的分支必须标记为`Protected`.
- 对的更改`main`和`develop`必须使用合并请求传播分支。
- 每个合并请求至少需要一次批准。

备注

- 作者：Jane Doe
- 版本：0.1
- 日志：
  - 0.1: 初始提议的版本

AWS指南词汇表

AI 和 ML 术语 (p. 11)|迁移期限 (p. 12)|现代化期限 (p. 15)

AI 和 ML 术语

以下是人工智能 (AI) 和机器学习 (ML) 相关策略、指南和模式中常用的术语AWS规范性指南。要建议参赛作品，请使用提供反馈词汇表末尾的链接。

二进制分类	一个预测二进制结果的过程（两个可能的类别之一）。例如，您的机器学习模型可能需要预测诸如“这个电子邮件是垃圾邮件还是垃圾邮件？”或者“此产品是一本书还是汽车？”
分类	有助于生成预测的分类过程。分类问题的 ML 模型可以预测离散值。离散值总是彼此不同。例如，模型可能需要评估图像中是否有汽车。
数据预处理	将原始数据转换为 ML 模型容易解析的格式。预处理数据可能意味着删除某些列或行并解决缺失、不一致或重复的值。
深度合奏	组合多个深度学习模型进行预测。您可以使用深度合奏来获得更准确的预测或估计预测中的不确定性。
深度学习	一个 ML 子字段，它使用多层人工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。
探索性数据分析 (EDA)	分析数据集以了解其主要特征的过程。您可以收集或聚合数据，然后执行初步调查以查找模式、检测异常情况并检查假设。EDA 是通过计算汇总统计数据和创建数据可视化来执行的。
features	用于进行预测的输入数据。例如，在制造环境中，要素可以是从生产线定期捕获的图像。
重要性功能	功能对模型的预测有多重要。这通常表示为可以通过各种技术计算的数值分数，例如 Shapley 加法解释 (SHAP) 和集成渐变。有关更多信息，请参阅 。AWS 的机器学习模型可解释性.
功能转换	为机器学习流程优化数据，包括使用额外的源丰富数据、扩展值或从单个数据字段中提取多组信息。这使机器学习模型能够从数据中受益。例如，如果将“2021-05-27 00:15:37”日期分解为“2021”、“5 月”、“周四”和“15”，则可以帮助学习算法学习与不同数据组件相关的细微差别模式。
可解释性	机器学习模型的一个特征，它描述了人类能够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 。AWS 的机器学习模型可解释性.
多类别分类	有助于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 模型可能会问“此产品是书、汽车还是手机？”或“此客户最感兴趣什么类别？”

回归	一种预测数值的 ML 技术。例如，要解决 “这套房屋将以什么价格出售？” ML 模型可以使用线性回归模型根据有关房屋的已知事实（例如，平方英尺）预测房屋的销售价格。
培训	为机器学习模型提供数据。训练数据必须包含正确的答案。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。它输出了捕获这些模式的机器学习模型。然后，您可以使用 ML 模型对您不知道目标的新数据进行预测。
目标变量	你试图在监督机器学习中预测的值。这也称作结果变量. 例如，在制造设置中，目标变量可能是产品缺陷。
优化	更改训练过程的各个方面以提高机器学习模型的准确性。例如，您可以通过生成标记集，添加标签，然后在不同设置下几次重复这些步骤几次以优化模型，可以训练 ML 模型。
不确定性	一个概念，指的是不准确、不完整或未知的信息，这些信息可能会破坏预测机器学习模型的可靠性。有两种类型的不确定性：认识不确定性是由有限的、不完整的数据造成的，而恶心的不确定性是由数据固有的噪音和随机性引起的。有关更多信息，请参阅 。量化深度学习系统中的不确定性指南。

迁移期限

以下是由提供的迁移相关策略、指南和模式中常用的术语AWS规范性指南。要建议参赛作品，请使用提供反馈词汇表末尾的链接。

7 卢比	将应用程序迁移到云的七种常见迁移策略。这些策略建立在 Gartner 2011 年确定的 5 Rs 基础上，包括以下内容： • 重构/重新架构 — 通过充分利用云原生功能来提高敏捷性、性能和可扩展性，移动应用程序并修改其架构。这通常涉及移植操作系统和数据库。例如：将本地 Oracle 数据库迁移到 Amazon Aurora PostgreSQL 兼容版。 • 重新平台（提升和重塑）— 将应用程序移动到云端，并引入一定程度的优化以利用云功能。例如：将本地 Oracle 数据库迁移到 Amazon Relational Database Service (Amazon RDS)，在AWS云。 • 回购（直销和购物）— 切换到其他产品，通常通过从传统许可证转移到 SaaS 模式。例如：将客户关系管理 (CRM) 系统迁移到 Salesforce.com。 • 重新托管（提升和移动）— 将应用程序移动到云端，而无需进行任何更改以利用云功能。例如：将本地 Oracle 数据库迁移到AWS云。 • 重新定位（虚拟机管理程序级别的提升和移动）— 将基础设施迁移到云端，而无需购买新硬件、重写应用程序或修改现有操作。此迁移场景特定于 VMware CloudAWS，它支持虚拟机 (VM) 兼容性和本地环境之间的工作负载可移性AWS. 将基础架构迁移到 VMware Cloud 时，您可以使用本地数据中心中的 VMware Cloud Foundation 技术AWS. 例如：将托管 Oracle 数据库的虚拟机管理程序重新定位到 VMware 云AWS. • 保留（重新访问）— 将应用程序保留在源环境中。其中可能包括需要进行大规模重构的应用程序，并且您希望将该工作推迟到以后再进行，以及要保留的旧应用程序，因为迁移这些应用程序没有业务理由。 • 停用 — 停用或删除源环境中不再需要的应用程序。
应用程序集	有关组织使用的每个应用程序的详细信息的集合，包括构建和维护应用程序的成本及其业务价值。这些信息是至关重要的投资组合发现和分析过程并有助于确定要迁移、现代化和优化的应用程序并确定其优先级。
人工智能运营（AIOP）	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。有关如何使用 AIOP 的更多信息，请参阅AWS迁移策略，请参阅运营集成指南.

AWS云采用框架 (AWSCAF)	来自的指南和最佳实践框架AWS以帮助组织制定高效而有效的计划，以便成功迁移到云端。AWSCAF 将指导组织成六个重点领域，称为视角：业务、人员、治理、平台、安全性和运营。业务、人员和治理观点侧重于业务技能和流程；平台、安全性和运营视角侧重于技术技能和流程。例如，人员视角针对负责处理人力资源 (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，AWSCAF 为人员发展、培训和沟通提供指导，以帮助组织为成功采用云做好准备。有关更多信息，请参阅 。AWSCAF 网站和AWSCAF 白皮书.
AWSlanding zone	landing zone 是架构完善的多账户AWS可扩展且安全的环境。这是一个起点，您的组织可以快速启动和部署工作负载和应用程序，同时对其安全和基础架构环境充满信心。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.
AWS工作负载鉴定框架 (AWSWQF)	评估数据库迁移工作负载、建议迁移策略并提供工作估算的工具。AWS包含 WQFAWS Schema Conversion Tool(AWS SCT)。它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，并提供评估报告。
业务连续性规划 (BCP)	该计划旨在解决破坏性事件（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。
Cloud Center of Excellence (CCoE)	一支多学科团队，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、建立迁移时间表以及领导组织完成大规模转型。有关更多信息，请参阅 。cCoE 帖子在AWS云企业策略博客。
采用的云阶段	组织迁移到AWS云： • 项目 — 为了概念验证和学习目的运行一些与云相关的项目 • 基金会 — 进行基本投资以扩大云采用率（例如，创建 landing zone、定义 cCoE、建立运营模式） • 迁移 — 迁移单个应用程序 • 重新发明 — 优化产品和服务，在云中进行创新 这些阶段是斯蒂芬·奥尔班在博客文章中定义的迈向云优先的旅程和采用阶段在AWS云企业策略博客。有关它们与AWS迁移策略，请参阅迁移准备指南.
配置管理数据库 (CMDB)	包含有关公司硬件和软件产品、配置以及相互依赖关系的信息的数据库。您通常在迁移的产品组合发现和分析阶段使用 CMDB 的数据。
史诗	在敏捷方法中，有助于组织工作并优先排序的功能类别。Epics 提供了有关要求和实施任务的概述。例如，AWSCAF 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。有关史诗的更多信息，请参阅AWS迁移策略，请参阅计划实施指南.
异构数据库迁移	将源数据库迁移到使用不同数据库引擎（例如，Oracle 到 Amazon Aurora）的目标数据库。异构迁移通常是重新架构工作的一部分，转换模式可能是一项复杂的任务。AWS提供AWS SCT这有助于模式转换。
同类数据库迁移	将源数据库迁移到共享相同数据库引擎的目标数据库（例如，Microsoft SQL Server 到针 Amazon RDS for SQL Server）。同类迁移通常是重新托管或重建平台工作的一部分。您可以使用本机数据库实用程序迁移模式。
空闲应用程	在 90 天内的平均 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。
IT 信息库 (ITIL)	一套提供 IT 服务并使这些服务与业务需求保持一致的最佳做法。ITIL 为 ITSM 奠定了基础。
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.

大型迁移	迁移 300 台或更多服务器。
Migration Acceleration Program (MAP)	网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的AWS该计划提供咨询支持、培训和服务，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 包括用于以有条不紊的方式执行旧版迁移的迁移方法以及一套用于自动化和加速常见迁移方案的工具。
迁移投资组合评估 (MPA)	一种在线工具，提供用于验证业务案例以便迁移到AWS云。MPA 提供详细的产品组合评估（服务器规模合适、定价、总拥有成本比较、迁移成本分析）以及迁移规划（应用程序数据分析和数据收集、应用程序分组、迁移优先级和浪潮规划）。这些区域有：MPA 工具（需要登录）对所有人免费使用AWS顾问和 APN 合作伙伴顾问。
迁移准备情况评估 (MRA)	通过使用AWSCAF。有关更多信息，请参阅 。迁移准备指南. MRA 是第一个阶段AWS 迁移策略.
大规模迁移	将大多数应用程序组合转移到云端的过程，在每波中，更多的应用程序以更快的速度移动。本阶段利用从早期阶段吸取的最佳做法和经验教训来实施迁移工厂通过自动化和敏捷交付来简化工作负载的迁移。这是第三阶段AWS迁移策略.
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营部门、业务分析师和业主、迁移工程师、开发人员和 DevOps 在 Sprint 中工作的专业人士。在一个企业应用程序组合中，20% 到 50% 包括可以通过工厂方法优化的重复模式。有关更多信息，请参阅 。关于移民工厂的讨论和CloudEndure Migration 工厂指南在此内容集中。
迁移元数据	完成迁移所需的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。迁移元数据的示例包括目标子网、安全组和AWSaccount.
迁移模式	可重复的迁移任务，详细介绍了迁移策略、迁移目标以及使用的迁移应用程序或服务。例如：使用重新托管到 Amazon EC2 的迁移AWSApplication Migration Service。
迁移策略	将工作负载迁移到AWS云。有关更多信息，请参阅 。7 卢比 (p. 12)在此词汇表中输入并请参阅动员组织以加快大规模迁移.
业务层面的协议（法律厅）	一份协议，明确了 IT 职能部门承诺相互交付的内容，以支持服务级别协议 (SLA)。
运营集成 (OI)	云中运营的现代化流程，包括就绪性规划、自动化和集成。有关更多信息，请参阅 。运营集成指南.
组织变更管理 (OCM)	一个框架，用于从人员、文化和领导力的角度管理重大、颠覆性的业务转型。OCM 通过加快变革的采纳、解决过渡性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并过渡到新的系统和战略。在AWS迁移策略，这个框架被称为人加速，因为云采用项目需要的变化速度。有关更多信息，请参阅 。OCM 指南.
行动手册	一组预定义的步骤，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。行动手册可以采取脚本、自动运行手册或操作现代化环境所需的流程或步骤摘要的形式。
项组合评估	发现、分析应用程序组合并确定其优先级以计划迁移的过程。有关更多信息，请参阅 。评估迁移就绪性.
负责任、负责、咨询、知情 (RACI) 矩阵	定义和分配项目中的角色和职责的矩阵。例如，您可以创建 RACI 来定义安全控制所有权，或者为迁移项目中的特定任务确定角色和责任。
运行手册	执行特定任务所需的一组手动或自动过程。它们通常是为了简化具有高错误率的重复性操作或程序而构建的。
服务级别协议 (SLA)	该协议阐明了 IT 团队承诺为客户提供的服务，例如服务正常运行时间和性能。

任务列表	一种用于通过运行手册跟踪进度的工具。任务列表包含运行手册的概述和待完成的常规任务列表。对于每项常规任务，它包括所需的估计时间、所有者和进度。
工作流	迁移项目中负责一组特定任务的职能组。每个工作流都是独立的，但支持项目中的其他工作流。例如，产品组合工作流负责确定应用程序的优先级、浪潮规划和收集迁移元数据。产品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。
Zombie 应用	平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常停用这些应用程序。

现代化期限

以下是由提供的与现代化相关的战略、指南和模式中常用的术语AWS规范性指南。要建议参赛作品，请使用提供反馈词汇表末尾的链接。

业务能力	企业如何创造价值（例如销售、客户服务或营销）。微服务体系结构和开发决策可以由业务能力驱动。有关更多信息，请参阅 。围绕业务能力组织的 部分在上运行容器化微服务AWS白皮书。
域驱动型设计	一种开发复杂软件系统的方法，将其组件与每个组件所服务的不断变化的领域或核心业务目标联系起来。这个概念是埃里克·埃文斯在他的书中介绍的，域驱动的设计：解决软件核心的复杂性（波士顿：艾迪生-韦斯利专业人员，2003 年）。有关如何将域驱动设计结合使用与绞线器无花果模式的信息，请参阅通过使用容器和 Amazon API Gateway 以增量方式对传统微软 ASP.NET (ASMX) Web 服务进行现代化.
微服务	一种小型、独立的服务，通过明确定义的 API 进行通信，通常由小型、独立的团队拥有。例如，保险系统可能包括与销售或营销等业务能力相对应的微服务或子域，例如购买、索赔或分析。微服务的优势包括敏捷性、灵活的扩展、易于部署、可重复使用的代码和弹性。有关更多信息，请参阅 。通过使用AWS无服务器服务.
微服务架构	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。这些微服务通过使用轻量级 API 通过定义明确的接口进行通信。此体系结构中的每个微服务都可以进行更新、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 。在上实施微服务AWS.
现代化	将过时的（旧版或单片）应用程序及其基础架构转变为云中灵活、弹性和高可用性的系统，以降低成本、提高效率并利用创新。有关更多信息，请参阅 。实现应用程序现代化的策略AWS云.
评估更新就绪性	一种评估，可帮助确定组织应用程序的现代化就绪性；确定好处、风险和依赖关系；并确定组织可以在多大程度上支持这些应用程序的 future 状态。评估的结果是目标架构的蓝图、详细说明现代化进程的开发阶段和里程碑的路线图以及解决已查明的差距的行动计划。有关更多信息，请参阅 。评估中的应用程序的现代化就绪AWS云.
整体式应用程序（整体）	作为单项服务运行的应用程序，进程紧密耦合。整体式应用程序有几个缺点。如果一个应用程序功能遇到需求激增，则必须扩展整个体系结构。随着代码库的增长，添加或改进单片应用程序的功能也变得更加复杂。要解决这些问题，你可以使用微服务体系结构。有关更多信息，请参阅 。将巨石分解为微服务.
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。如果您的微服务具有相同的数据存储技术，它们可能会遇到实施方面的挑战或性能不佳。如果微服务使用最适合其需求的数据存储，则更容易实施并实现更好的性能和可扩展性。有关更多信息，请参阅 。在微服务中启用数据持久性.
分割和种子模型	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，核心团队将分解成新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的生产力，并支持快速创新。有关更多信息，请参阅 。分阶段实现应用程序现代化的方法AWS云.

扼杀者无花果模式	一种通过逐步重写和替换系统功能，直到旧系统停用之前，对整体式系统进行现代化的方法。这种模式使用了一种无花果藤的类比，它生长成一棵既定的树，最终克服并取代了它的主机。模式是由马丁·福勒介绍作为重写单片系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅通过使用容器和 Amazon API Gateway 以增量方式对传统微软 ASP.NET (ASMX) Web 服务进行现代化.
两个披萨团队	一个小型 DevOps 你可以用两个比萨饼喂食的团队。两个比萨团队的规模确保了在软件开发方面进行协作的最佳机会。有关更多信息，请参阅 。两个披萨团队的 部分简介 DevOps 上AWS白皮书。

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 future 更新通知，可以订阅RSS 源.

update-history-change	update-history-description	update-history-date
初次发布 (p. 17)	—	2022 年 3 月 16 日

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。