AWS Well-Architected Framework

卓越运营支柱

卓越运营支柱: AWS Well-Architected Framework

Table of Contents

卓越运营支柱 – AWS Well-Architected Framework

发布日期： 2024 年 6 月 27 日 (文档修订)

本白皮书主要介绍 AWS Well-Architected Framework 的卓越运营支柱。它提供了指导，以帮助您在 AWS 工作负载的设计、交付和维护过程中应用最佳实践。

简介

此 AWS Well-Architected Framework 能够帮助您认识到您在 AWS 上构建工作负载时所做决策的收益和风险。通过使用此框架，您将了解在云中设计和运行可靠、安全、高效、经济实惠且可持续的工作负载的运营和架构最佳实践。它提供了一种统一的方法，使您能够根据最佳实践衡量运营和架构，并确定需要改进的方面。我们相信，拥有在设计时充分考虑了运营因素的架构完善的工作负载，可大大提高实现业务成功的可能性。

该框架基于六大支柱：

- 卓越运营
- 安全性
- 可靠性
- 性能效率
- 成本优化
- 可持续性

本白皮书重点介绍了卓越运营支柱，以及如何将其用作架构完善的解决方案的基础。卓越运营在环境中很难实现，因为在环境中，运营被视为一种独立的职能，与它支持的业务团队和开发团队是区分开的。通过采用本白皮书中的实践，您可以构建这样一种架构：提供状态洞察、实现有效且高效的运营和事件响应，并可以持续改进和支持您的业务目标。

本白皮书面向技术人员，例如首席技术官（CTO）、架构师、开发人员和运维团队成员。阅读本白皮书后，您将了解在设计云架构以实现卓越运营时可以采用的 AWS 最佳实践和策略。本白皮书不提供实施细节或架构模式，但会针对此类信息提供适当资源。

卓越运营

在亚马逊，我们将卓越运营定义为一种承诺，即正确地构建软件，同时持续提供卓越的客户体验。它包含组织团队、设计工作负载、大规模运营工作负载和随时间变化改进工作负载的最佳实践。卓越运营有助于您的团队将更多时间用在构建让客户受益的新功能上，并减少用于维护和处理突发事件的时间。为了正确构建，我们依赖最佳实践。这些实践将为您和您的团队带来运行良好的系统和平衡的工作负载，尤其是卓越的客户体验。

卓越运营的目标是快速可靠地将新功能和错误修复交付给客户。投资于卓越运营的组织在构建新功能、进行变革和应对失败时能够始终让客户满意。在这一过程中，卓越运营通过帮助开发人员始终如一地实现高质量的结果，推动了持续集成和持续交付（CI/CD）。

设计原则

以下是在云中实现卓越运营的设计原则：

- 围绕业务成果组织团队：领导层远见、有效的运营以及与业务相一致的运营模式能够协助团队实现业务成果。领导层应致力于 CloudOps 转型并全身心地投入其中，采用合适的云运营模式，激励团队以非常高效的方式运营并实现业务成果。正确的运营模式会利用人员、流程和技术能力来扩大规模，优化工作效率，并通过敏捷性、响应能力和适应能力打造差异化优势。组织的长期愿景会转化为一系列目标，并且这些目标将传达给整个组织内云服务的利益相关者和使用者。各个层面的目标和运营 KPI 将保持一致。这种做法能够维持通过实施以下设计原则所获得的长期价值。
- 实施可观测性以获得切实可行的见解：全面了解工作负载行为、性能、可靠性、成本和运行状况。建立关键绩效指标（KPI），利用可观测性遥测来作出明智的决策，并在业务结果面临风险时迅速采取行动。基于可操作的可观测性数据，主动提高性能和可靠性，降低成本。
- 尽可能安全地实现自动化：在云中，您可以将用于应用程序代码的工程规范应用于整个环境。您能够以代码形式定义整个工作负载及其运营（应用程序、基础设施、配置和程序），并对其进行更新。之后，您可以通过启动工作负载的运营来响应事件，从而实现运营的自动化。在云中，您可以通过配置防护机制（包括速率控制、错误阈值和审批）来实现自动化的安全。通过有效的自动化，您可以实现对事件的持续响应，限制人为错误并减少操作员的艰苦工作。
- 频繁进行小型、可回滚的变更：将工作负载设计为可扩展且松耦合，以允许定期更新组件。自动部署技术加上小型增量变更可缩小影响范围，并能够在发生故障时更快地进行回滚。这将增强您的信心，在保持质量和快速适应市场条件变化的同时，为您的工作负载提供有益的变化。
- 经常完善操作程序：随着工作负载的演变，应相应地改进操作程序。在使用运营程序时，要寻找机会改进它们。定期审查并验证所有流程是否有效，以及团队是否熟悉这些流程。在发现差距时，相应地

更新程序。向所有利益相关者和团队传达程序更新。将运营游戏化，以分享最佳实践并向团队传授知识。

- 预测失败：通过推动失败场景来了解工作负载的风险概况及其对业务成果的影响，实现极其出色的运营成功。测试您的程序的有效性以及团队对这些模拟失败作出的反应。制定明智的决策，管理测试中确定的开放风险。
- 从所有运营事件和指标中吸取经验教训：从所有运营事件和故障中吸取经验教训，推动改进。在多个团队乃至组织范围中分享经验教训。经验教训应重点介绍有关运营如何促进获得业务成果的数据和轶事。
- 使用托管服务：尽可能使用 AWS 托管服务，减少运营负担。围绕与这些服务的交互制定操作程序。

定义

在云中实现卓越运营有四个领域的最佳实践：

- 组织
- 准备
- 运营
- 演进

您的组织领导层负责定义业务目标。您的组织必须了解各种要求和重点，并利用它们来组织和开展工作，从而为获得业务成果提供支持。您的工作负载必须发出所需信息以提供支持。采用多种服务来支持工作负载的集成、部署和交付，这将通过自动化重复流程，增加对生产的有益更改。

工作负载的运营可能存在固有风险。您必须了解这些风险并作出明智的生产决策。您的团队必须能够支持您的工作负载。从预期业务成果中得出的业务和运营指标将有助于您了解工作负载的运行状况、运营活动以及对事件的响应。您的重点将随着您的业务需求和业务环境的变化而变化。将这些作为反馈循环，持续推动组织和工作负载运营的改进。

组织

您需要了解您组织的优先事项、组织结构以及您的组织如何支持您的团队成员，以便为您的业务成果提供支持。

要实现卓越运营，您必须了解以下内容：

主题

组织重点

您的团队需要对整个工作负载、他们在其中的角色以及共同的业务目标有一致的理解，以便设置运营重点以实现业务成功。明确运营重点可以让您的工作效益最大化。定期审查您的运营重点，以便在组织的需求发生变化时对其进行更新。

最佳实践

OPS01-BP01 评估客户需求

让包括业务、开发和运维团队在内的主要利益相关方参与进来，以便确定将工作重心放在哪里来满足外部客户的需求。这可以证实您充分了解实现您期望的业务成果所需的运营支持。

期望结果：

- 您能够从客户成果出发进行逆向思维。

- 您了解运营实践如何协助您获得业务成果和实现目标。
- 您让所有相关方参与进来。
- 您已建立用于捕获客户需求的机制。

常见反面模式：

- 您决定核心业务时间之外不再提供客户支持，但是您还没有查看历史支持请求数据。您不知道这是否会对客户产生影响。
- 您正在开发一项新功能，但尚未与客户沟通，不了解客户是否需要；如果需要，以什么形式提供；并且尚未通过试验来验证交付需求和方法。

建立此最佳实践的好处：需求得到满足的客户流失的可能性更小。评估和了解外部客户需求将为您提供相关信息，告知您如何通过安排工作的优先级来实现业务价值。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

了解业务需求：包括业务、开发和运营团队在内的利益相关方需要有共同的目标和共同的理解，才能实现业务成功。

审核外部客户的业务目标、需求和重点：让包括业务、开发和运营团队在内的主要利益相关方参与进来，讨论外部客户的目标、需求和重点。这可以确保您充分了解实现业务成果和客户成果所需的运营支持。

建立共识：建立共识，确定工作负载的业务功能、每个团队在运行工作负载方面的角色，以及这些因素如何支持内部和外部客户共同的业务目标。

资源

相关最佳实践：

- OPS11-BP03 实施反馈环路

OPS01-BP02 评估内部客户需求

让包括业务、开发和运营团队在内的主要利益相关方参与进来，以便确定怎样将工作重心放在内部客户的需求上。这可以确保您充分了解实现业务成果所需的运营支持。

期望结果：

- 您使用这些已明确的重点，将改进工作集中部署在能发挥最大影响（例如，培养团队技能、提高工作负载性能、降低成本、自动化运行手册或增强监控）的方面。
- 您随着需求的变化更新重点。

常见反面模式：

- 您决定更改产品团队的 IP 地址分配（没有与他们商议），以便更轻松地管理网络。您不知道这是否会对您的产品团队产生影响。
- 您正在采用一种新的开发工具，但尚未与内部客户沟通，不了解他们是否需要，或者是否与他们的现有实践兼容。
- 您正在实施一个新的监控系统，但尚未与内部客户沟通，不了解他们是否有监控或报告需求需要考虑。

建立此最佳实践的好处：评估和了解内部客户需求将为您提供相关信息，告知您如何通过安排工作的优先级来实现业务价值。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

- 了解业务需求：包括业务、开发和运营团队在内的利益相关方需要有共同的目标和共同的理解，才能实现业务成功。
- 分析内部客户的业务目标、需求和重点：让包括业务、开发和运营团队在内的主要利益相关方参与进来，讨论内部客户的目标、需求和重点。这可以确保您充分了解实现业务成果和客户成果所需的运营支持。
- 建立共识：建立共识，确定工作负载的业务功能、每个团队在运行工作负载方面的角色，以及这些因素如何支持内部和外部客户共同的业务目标。

资源

相关最佳实践：

- OPS11-BP03 实施反馈环路

OPS01-BP03 评估治理要求

治理是公司用来实现其业务目标的一系列策略、规则或框架。从组织内部生成治理要求。它们会影响您选择的技术类型或影响您运营工作负载的方式。将组织治理要求纳入工作负载中。合规是证明您已实施治理要求的能力。

期望结果：

- 将治理要求纳入架构设计和工作负载运营中。
- 您可以提供证据来证明您遵循治理要求。
- 定期审核和更新治理要求。

常见反模式：

- 您的组织要求根账户具有多重身份验证。您未能实施此要求，根账户已泄露。
- 在设计工作负载时，您选择了未得到 IT 部门批准的实例类型。您无法启动工作负载，必须重新设计。
- 您需要制定灾难恢复计划。您没有制定灾难恢复计划，且您的工作负载遭受了长时间的中断。
- 您的团队希望使用新实例，但您的治理要求没有更新，不允许使用新实例。

建立此最佳实践的好处：

- 遵循治理要求，使您的工作负载与更广泛的组织政策保持一致。
- 治理要求反映了行业标准和您组织的最佳实践。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

与利益攸关方和治理组织合作，确定治理要求。将治理要求包括在您的工作负载中。可以提供证据来证明您遵循治理要求。

客户示例

在 AnyCompany Retail，云运营团队与整个组织内的利益攸关方一起制定治理要求。例如，他们禁止对 Amazon EC2 实例进行 SSH 访问。如果团队需要系统访问，他们需要使用 AWS Systems Manager Session Manager。随着新服务推出，云运营团队定期更新治理要求。

实施步骤

1. 为您的工作负载确定利益攸关方，包括任何集中式团队。
2. 与利益攸关方一起确定治理要求。
3. 生成列表之后，按优先顺序列出改进项，并开始将它们实施到您的工作负载中。
   a. 使用 AWS Config 等服务来创建治理即代码，并确认遵循治理要求。
   b. 如果您使用 AWS Organizations，则可以利用服务控制策略来实施治理要求。
4. 提供用于验证实施的文档。

实施计划的工作量级别：中等。实施时未满足治理要求可能会导致工作负载返工。

资源

相关最佳实践：

- OPS01-BP04 评估合规性要求 - 合规性类似于治理，但它来自组织外部。

相关文档：

- AWS 管理和治理云环境指南
- 多账户环境中的 AWS Organizations 服务控制策略的最佳实践
- AWS Cloud 中的治理：敏捷性和安全性之间的适当平衡
- 什么是治理、风险与合规性（GRC）？

相关视频：

- AWS 管理和治理：配置、合规性和审计 - AWS 在线技术讲座
- AWS re:Inforce 2019：云时代的治理（DEM12-R1）
- AWS re:Invent 2020：使用 AWS Config 实现合规性即代码
- AWS re:Invent 2020：AWS GovCloud (US) 中的敏捷治理

相关示例：

- AWS Config 合规包示例

相关服务：

- AWS Config
- AWS Organizations - 服务控制策略

OPS01-BP04 评估合规性要求

监管、行业和内部合规性要求是定义组织优先级的重要驱动因素。您的合规性框架可能会阻止您使用特定技术或地理位置。如果未确定外部合规框架，则进行尽职调查。生成验证合规性的审计或报告。

如果您宣称自己的产品符合特定的合规性标准，则您必须有内部流程来确保持续合规。合规性标准的示例包括 PCI DSS、FedRAMP 和 HIPAA。适用的合规性标准由各种因素决定，例如解决方案存储或传输的数据类型，以及解决方案支持的地理区域。

期望结果：

- 将监管、行业和内部合规性要求纳入架构选择。
- 您可以验证合规性并生成审计报告。

常见反模式：

- 部分工作负载属于支付卡行业数据安全标准（PCI-DSS）框架，但您的工作负载以未加密方式存储信用卡数据。
- 软件开发人员和架构师不了解组织必须遵循的合规性框架。
- 年度系统与组织控制（SOC2）类型 II 审计即将开始，您无法验证控制措施是否已到位。

建立此最佳实践的好处：

- 评估和了解适用于工作负载的合规性要求将为您提供相关信息，告知您如何通过安排工作的优先级来实现业务价值。
- 选择与合规性框架保持一致的适当位置和技术。
- 设计工作负载以实现可审计性，这样就可以证明您遵守合规性框架。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

实施此最佳实践意味着将合规性要求纳入架构设计过程。您的团队成员了解所需的合规性框架。您依照框架验证合规性。

客户示例

AnyCompany Retail 存储客户的信用卡信息。卡存储团队的开发人员明白他们需要遵守 PCI-DSS 框架。他们已采取措施来确认按照 PCI-DSS 框架安全地存储和访问信用卡信息。他们每年都会与安全团队一起验证合规性。

实施步骤

1. 与我们的安全性和治理团队合作，确定您的工作负载必须遵守哪些行业、监管或内部合规性框架。将合规性框架纳入您的工作负载。
   a. 使用 AWS Compute Optimizer 和 AWS Security Hub 等服务验证 AWS 资源是否持续合规。
2. 向团队成员介绍合规性要求，以便他们可以根据要求运行和改进工作负载。架构和技术选择中应包括合规性要求。
3. 根据合规性框架，您可能需要生成审计或合规性报告。与组织合作，尽可能使此过程实现自动化。
   a. 使用 AWS Audit Manager 等服务验证合规性和生成审计报告。
   b. 您可以使用 AWS Artifact 下载 AWS 安全性和合规性文档。

实施计划的工作量级别：中等。实施合规性框架并非易事。生成审计报告或合规性文档带来了额外的复杂性。

资源

相关最佳实践：

- SEC01-BP03 识别并验证控制目标 - 安全控制目标是整体合规性的重要组成部分。
- SEC01-BP06 在管道中自动测试和验证安全控制措施 - 作为管道的一部分，验证安全控制。您还可以生成新变更的合规性文档。
- SEC07-BP02 定义数据保护控制措施 - 许多合规性框架都基于数据处理和存储策略。
- SEC10-BP03 准备取证能力 - 有时候审计合规性中会使用取证功能。

相关文档：

- AWS 合规中心

- AWS 合规性资源
- AWS 风险和合规性白皮书
- AWS 责任共担模式
- 合规性计划范围内的 AWS 服务

相关视频：

- AWS re:Invent 2020：使用 AWS Compute Optimizer 实现合规性即代码
- AWS re:Invent 2021 - 云合规性、保证和审计
- AWS Summit ATL 2022 - 在 AWS 上实施合规性、保证和审计（COP202）

相关示例：

- AWS 上的 PCI DSS 和 AWS 基础安全最佳实践

相关服务：

- AWS Artifact
- AWS Audit Manager
- AWS Compute Optimizer
- AWS Security Hub

OPS01-BP05 评估威胁形势

评估对业务的威胁（例如竞争、业务风险和负债、运营风险和信息安全威胁），并在风险注册表中维护当前信息。在确定工作重心时，将风险的影响考虑在内。

Well-Architected Framework 强调学习、衡量和改进。该框架为您提供了一种一致的方法，来评估架构，并实施将随着时间推移而扩展的设计。AWS 提供了 AWS Well-Architected Tool，可帮助您在开发之前审核方法，在生产之前审核工作负载状态，以及在生产过程中审核工作负载状态。您可以将其与最新的 AWS 架构最佳实践进行比较，监控工作负载的整体状态，并深入了解潜在风险。

AWS 客户可以使用针对关键任务型工作负载的指导式 Well-Architected Review，以根据 AWS 最佳实践来衡量其架构。企业支持客户可以使用运营审核，该审核旨在帮助他们找出云中的运营方法所存在的漏洞。

这些审核需要跨团队参与，可帮助各团队就工作负载形成共识，并理解彼此的团队角色如何助力取得成功。通过审核所确定的需求可以帮助确定您的运营重点。

AWS Trusted Advisor 是一种工具，会通过一组核心检查，为您提供优化建议，帮助您确定运营重点。商业支持和企业支持客户可以使用其他检查，这些检查重点关注安全性、可靠性、性能和成本优化，可进一步帮助他们确定运营重点。

期望结果：

- 您定期审核 Well-Architected 和 Trusted Advisor 输出并采取行动
- 您了解服务的最新补丁状态
- 您了解已知威胁的风险和影响，并能采取相应的行动
- 您能够根据需要实施缓解措施
- 您能够传达行动和背景

常见反面模式：

- 您在产品中使用的是旧版软件库。对于可能会对工作负载产生意外影响的问题，需要对库进行安全更新，而您忽略了这一点。
- 您的竞争对手刚刚发布了新的产品版本，可以解决许多客户对您产品的投诉。您没有优先解决这些已知问题。
- 监管机构一直在追查像您这样的不符合法律法规要求的公司。您没有优先处理任何未解决的合规性要求。

建立此最佳实践的好处：发现并了解对企业和工作负载的威胁，这有助于确定需解决的威胁、威胁的优先级以及采取对策所需的资源。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

- 评估威胁形势：评估对业务的威胁（例如竞争、业务风险和负债、运营风险和信息安全威胁），以便您在确定工作重心时可以将其影响考虑在内。
  - AWS 最新安全公告
  - AWS Trusted Advisor

- 维护威胁模型：建立并维护威胁模型，确定潜在威胁、计划内和已实施的缓解措施及其优先级。审核威胁酿成意外事件的可能性、从意外事件恢复的成本和预期造成的危害，以及防止这些意外事件发生的成本。根据威胁模型内容的更改修订优先级。

资源

相关最佳实践：

- SEC01-BP07 使用威胁模型识别威胁并确定缓解措施的优先级

相关文档：

- AWS Cloud合规
- AWS 最新安全公告
- AWS Trusted Advisor

相关视频：

- AWS re:Inforce 2023 - A tool to help improve your threat modeling

OPS01-BP06 在管理效益和风险的同时评估各种权衡因素

多方利益竞争可能会使确定工作优先顺序、构建能力和交付符合业务战略的成果变得具有挑战性。例如，您可能需要加快新功能的上市速度，而不是优化 IT 基础设施的成本。这可能导致两个利益相关方之间发生冲突。在这种情况下，应由更高级别的权威机构作出决定，以便解决冲突。需要使用数据来消除决策制定过程中的情感依附。

在战术层面，您可能也面临类似挑战。例如，选择使用关系数据库技术还是非关系数据库技术，可能会对应用程序的运营产生重大影响。了解各种选择的可预测结果非常重要。

AWS 帮助您向团队介绍 AWS 及其服务，让他们深入了解自己的选择会如何影响工作负载。使用由 AWS Support（AWS Knowledge Center、AWS 开发论坛和 AWS Support 中心）提供的资源和 AWS 文档来培训您的团队。如有其他问题，请联系 AWS Support。

AWS 还分享了 Amazon Builders' Library 中的运营最佳实践和模式。您可以通过 AWS 博客和 AWS 官方播客，获得各种其他有用信息。

期望结果：您拥有一个明确定义的决策治理框架，以加快云交付企业内各个级别的重要决策制定。此框架包括风险登记单、有权作出决策的已定义角色，以及针对各级决策制定的模型等特色内容。此框架预先定义了冲突解决方式、需要提供的数据以及选项优先顺序的确定方式，因此，一旦作出决策，您便能立即执行。决策制定框架包括一种标准化方法，用于审核和认真考虑每项决策的收益和风险以了解权衡。这可能包括外部因素，例如遵守法规合规性要求。

常见反面模式：

- 您的投资者要求您证明符合支付卡行业数据安全标准（PCI DSS）。您没有在满足他们的要求和继续进行当前的开发工作之间进行权衡，而是在没有证明合规性的情况下继续进行开发工作。投资者出于对平台安全性和投资的担忧停止了对公司的支持。
- 您决定设置一个库，这是您的一位开发人员在互联网上找到的库。您尚未评估从未知来源采用此库的风险，也不知道其中是否包含漏洞或恶意代码。
- 对于迁移，最初的业务计划是实现 60% 的应用程序工作负载的现代化。但由于遇到技术难题，您决定仅实现 20% 的应用程序工作负载的现代化，这导致了长期计划收益的减少；基础设施团队的操作负担加重，需要手动支持遗留系统；并且更加依赖于培养基础设施团队的新技能组合，而团队并未对此做好规划。

建立此最佳实践的好处：充分协调和支持董事会层面的业务优先事项，了解取得成功的风险，制定明智的决策，并在风险可能阻碍成功机遇时采取适当的行动。了解决策的影响和后果有助于您设定选择的优先顺序，更快地让各个领导达成共识，从而改善业务成果。确定选择可以带来的收益并了解企业所面临的风险，有助于您依据数据而不是轶事来制定决策。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

应由推动关键决策制定要求的管理机构来定义如何管理收益和风险。您需要先了解所涉风险，然后基于决策对企业的益处，来制定决策，并确定其优先级。准确的信息对于制定组织决策至关重要。这应基于可靠的测量值，并由常见的成本效益分析行业惯例来定义。要制定这些类型的决策，需要在集权和分权之间取得平衡。始终要进行权衡，并且必须了解每种选择如何影响已定义的战略和所需的业务成果。

实施步骤

1. 在整体云治理框架内正式确定收益衡量实践。
    a. 平衡决策制定的集权与某些决策的分权。
    b. 要明白一点，将繁冗的决策过程强加于每个决策之上，只会拖慢您的决策速度。
    c. 将外部因素纳入您的决策制定过程（例如合规性要求）。

2. 为各级决策建立一致认可的决策制定框架，包括谁负责疏解受利益冲突影响的决策。
    a. 共同制定不可更改的单向门决策。
    b. 允许较低级别的组织领导者制定双向门决策。
3. 了解和管理收益与风险。在决策的收益与涉及的风险之间取得平衡。
    a. 确定收益：根据业务目标、需求和优先事项来确定收益。例如业务案例影响、上市时间、安全性、可靠性、性能和成本等。
    b. 确定风险：根据业务目标、需求和优先事项来确定风险。例如上市时间、安全性、可靠性、性能和成本等。
    c. 对照风险评估收益并作出明智决策：根据包括业务、开发和运营团队在内的主要利益相关方的目标、需求和优先事项，确定收益和风险的影响。对照发生风险的可能性及其影响产生的成本，来评估收益的价值。例如，强调上市速度而不是可靠性可能会带来竞争优势。但是如果出现可靠性问题，就可能会导致正常运行时间缩短。
4. 采用程序化方式执行关键决策，以便自动遵守合规性要求。
5. 利用已知的行业框架和能力，如价值流分析和精益生产，来衡量当前状态的绩效和业务指标，并定义逐步改进这些指标的迭代过程。

实施计划的工作量级别：中等 – 高

资源

相关最佳实践：

- OPS01-BP05 评估威胁形势

相关文档：

- Amazon 第 1 天文化的要素 | 作出高质量、高速度的决策
- 云监管
- Management & Governance Cloud Environment
- Governance in the Cloud and in the Digital Age: Parts One & Two

相关视频：

- Podcast | Jeff Bezos | On how to make decisions

相关示例：

- Make informed decisions using data (The DevOps Sagas)
- Using development value stream mapping to identify constraints to DevOps outcomes

运营模式

您的团队必须了解他们在实现业务成果方面所发挥的作用。团队需要了解自己在其他团队获得成功过程中所扮演的角色、其他团队在他们获得成功的过程中所扮演的角色，并设定共同的目标。了解责任分配、所有权归属、决策制定方式以及决策者将有助于集中精力，最大限度地发挥团队的优势。

团队的需求将由其所在行业、组织、团队的组成以及工作负载的特征决定。期望单个运营模式能够支持所有团队及其工作负载是不合理的。

随着开发团队数量的增加，组织中存在的运营模式数量也可能会增加。您可能需要使用运营模式组合。

采用标准和消费服务可以简化运营，并限制运营模式中的支持负担。通过确定采用标准和采用新功能的团队的数量，可以放大在共享标准方面的开发工作的益处。

一定要建立相应的请求增加、更改标准和标准例外的机制，以支持团队的活动。如果没有这样的机制，标准将成为创新的约束。对收益和风险进行评估之后，批准可行的和确认适当的请求。

明确定义职责范围将减少发生冲突和导致产生冗余工作的频率。如果业务团队、开发团队和运营团队之间存在紧密的协作关系，则更容易实现业务成果。

运营模式 2:2 展示图

这些运营模式 2:2 展示图可帮助您了解您环境中的团队之间的关系。这些图表着重说明成员的职责以及团队之间的关系，但我们也将通过这些示例讨论监管和做出决策。

我们的团队可能需要在多个模式的多个部分承担责任，具体取决于他们支持的工作负载。您可能希望打破比所描述的高级规范更专业的规范。当您分离或汇总活动，或叠加团队并提供更具体的细节时，这些模式可能会出现无穷无尽的变化。

您可能发现团队中存在重叠或未被认可的能力，这些能力可以提供额外优势或提高效率。您可能还发现您可以计划解决的组织中未满足的需求。

在评估组织变革时，请检查模式之间的权衡，您的各个团队采用的模式（现在和变革之后），您的团队的关系和职责将如何变化，以及所获得的益处是否抵得过对您的组织产生的影响。

您可以成功使用以下四种运营模式。某些模式更适合于特定使用案例或您的开发中的特定点。其中一些模式可能比在您的环境中使用的模式更具优势。

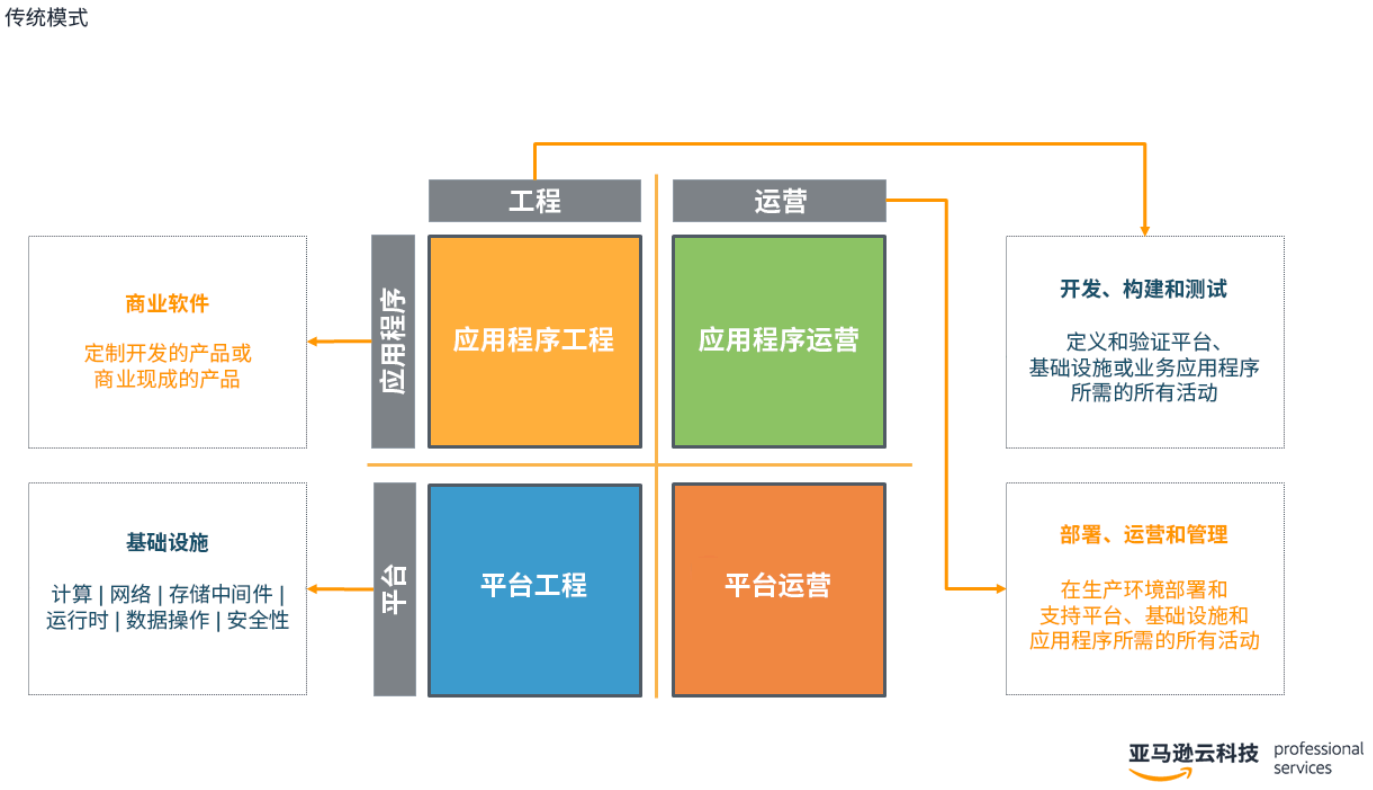
主题

- 完全分离运营模式
- 分离的应用程序工程和运营（AEO，Application Engineering and Operations）与基础设施工程和运营（IEO，Infrastructure Engineering and Operations），采用集中监管
- 分离的 AEO 和 IEO，采用集中监管并具有服务提供商
- 分离的 AEO 和 IEO，采用集中监管并具有内部服务提供商咨询合作伙伴
- 分离的 AEO 和 IEO，采用分散监管

完全分离运营模式

在下图中，纵轴上为应用程序和基础设施。应用程序是指促进取得业务成果的工作负载，可以是自定义开发或购买的软件。基础设施是指支持该工作负载的物理和虚拟基础设施以及其他软件。

横轴上为我们的工程和运营。工程是指应用程序和基础设施的开发、构建和测试。运营是应用程序和基础设施的部署、更新和持续支持。



运营模式 2:2 展示图

在许多组织中，存在这种“完全分离”模式。每个象限中的活动由单独的小组执行。 通过工作请求、工作队列、票证等机制或使用 IT 服务管理 (ITSM) 系统在团队之间分配工作。

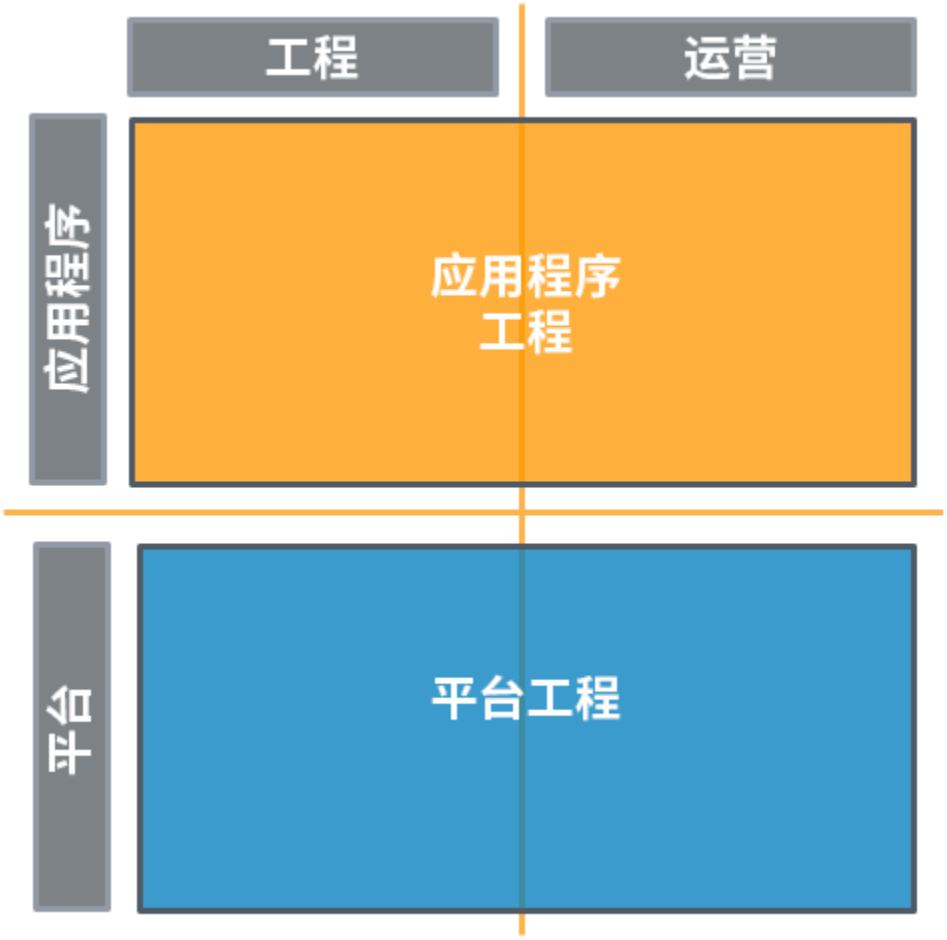
任务向团队或在团队之间的转移会增加复杂性，并造成瓶颈和延迟问题。请求可能会被延迟，直至它们成为重点事项。较迟发现缺陷可能需要大量返工，可能需要再次经历相同的团队及其职能部门。如果存在需要工程团队采取行动的事件，则将因转移活动而延迟响应。

当围绕正在执行的活动或职能组织业务团队、开发团队和运营团队时，出现工作重点偏失的风险较高。这可能导致团队专注于其特定职责，而不是专注于实现业务成果。团队可能专业化水平受限、被物理隔离或逻辑隔离，阻碍了沟通和协作。

分离的应用程序工程和运营（AEO，Application Engineering and Operations）与基础设施工程和运营（IEO，Infrastructure Engineering and Operations），采用集中监管

这种“分离的 AEO 和 IEO”模式采用“你构建，你运行”的方法。

应用程序工程师和开发人员同时执行工作负载工程设计和运营。同样，您的基础设施工程师可以对他们用以支持应用程序团队的平台同时进行工程设计和运营。



在本示例中，我们采用集中监管。标准会被分发、提供或共享给应用程序团队。

您应使用能够跨账户集中监管环境的工具或服务，例如 AWS Organizations.诸如 AWS Control Tower 之类的服务扩展了这一管理功能，使您能够定义账户设置的蓝图（支持您的运营模式），使用 AWS Organizations 进行持续监管以及自动预置新账户。

“你构建，你运行”并不意味着应用程序团队负责完全堆栈、工具链和平台。

平台工程设计团队为应用程序团队提供一套标准化的服务（例如，开发工具、监控工具、备份和恢复工具以及网络）。平台团队还可以为应用程序团队提供对经批准的云提供商服务、相同或两个团队的特定配置的访问权限。

提供部署已批准的服务和配置的自助服务功能的机制，如 Service Catalog，可以在实施监管的同时帮助限制与执行请求相关的延迟。

平台团队实现了完全堆栈可见性，因此应用程序团队可以区分应用程序组件的问题以及应用程序所使用的服务和基础设施组件。平台团队还可以提供配置这些服务的辅助措施，以及有关如何改进应用程序团队运营的指导。

如前所述，应用程序团队一定要建立相应的请求增加、更改标准和标准例外的机制，以支持团队的活动及其应用程序的创新。

分离的 AEO 和 IEO 模式为应用程序团队提供了强大的反馈循环。工作负载的日常运营通过直接交互或通过支持和功能请求间接增加与客户的联系。这种更高的可见性使应用程序团队能够更快地解决问题。更深入的互动和更密切的关系可提供对客户需求的洞察，并实现更快速的创新。

这也完全适用于为应用程序团队提供支持的平台团队。

采用的标准可以预先批准以供使用，从而减少投产所需的审核量。采用由平台团队提供的受支持的、业经测试的标准可以减少这些服务出现问题的频率。标准的采用可帮助应用程序团队专注于差异化工作负载。

分离的 AEO 和 IEO，采用集中监管并具有服务提供商

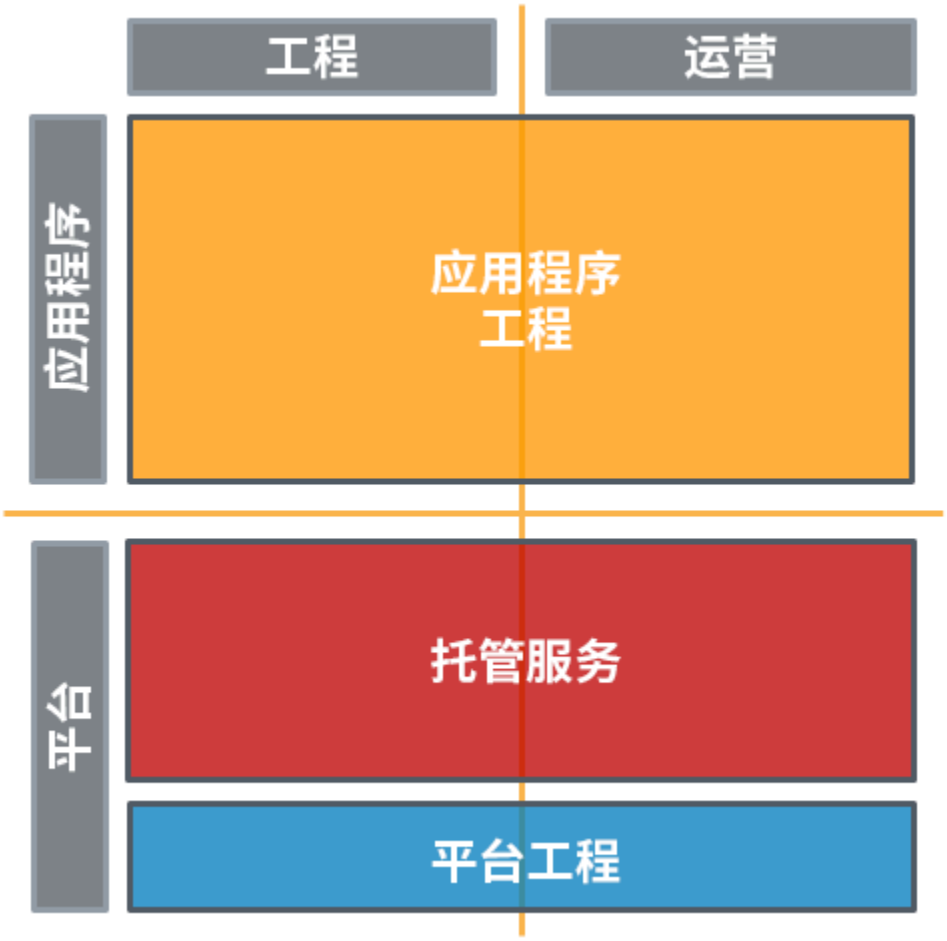
这种“分离的 AEO 和 IEO”模式采用“你构建，你运行”的方法。

应用程序工程师和开发人员同时执行工作负载工程设计和运营。

您的组织现在可能无法为专门的平台工程和运营团队提供相应的技能或团队人员支持，或者您可能不想为此花费时间和精力。

或者，您可能希望有一个平台团队能够专注于打造凸显业务优势的能力，不过您希望将千篇一律的日常运营工作交给外包商。

托管服务提供商，如 AWS Managed Services，AWS Managed Services 合作伙伴或 AWS 合作伙伴网络中的托管服务提供商会提供实施云环境的专业知识，并为您的安全性和合规性要求以及业务目标提供支持。



对于这一变体，我们视为监管由平台团队集中管理，并使用 AWS Organizations 和 AWS Control Tower 管理账户创建和策略。

此模式需要您修改自身机制，以便使用服务提供商的机制。它不能解决由于团队（包括您的服务提供商）之间的任务转换所造成的瓶颈和延迟，也无法解决由于发现缺陷较晚而存在的潜在返工。

提供商的标准、最佳实践、流程和专业知识将让您受益良多。此外，他们还会不断开发服务产品，您也会从中获益。

将托管服务添加到您的运营模式可以节省您的时间和资源，并使您的内部团队保持精干，专注于凸显业务优势的战略成果，而不是开发新的技能和功能。

分离的 AEO 和 IEO，采用集中监管并具有内部服务提供商咨询合作伙伴

这种“分离的 AEO 和 IEO”模式寻求建立“你构建，你运行”的方法。

您可能会要求应用程序团队执行工作负载的工程设计和运营活动，以及采用更接近于 DevOps 的文化。

您的应用程序团队可能正处在迁移、采用云服务或者打造现代化工作负载的过程中，目前尚不具备相应技能，无法为云和云运维提供足够的支持。应用程序团队这种在能力或熟悉度方面的欠缺可能会对工作造成阻碍。

为了解决这种问题，您可以成立一个云支持中心（CCoE，Cloud Center of Enablement）团队，提供一个可供提问、讨论需求和确定解决方案的论坛。根据组织的需求，CCoE 可以是由专家组成的专职团队，也可以是从整个组织中选择的参与者组成的虚拟团队。CCoE 可以支持团队的云转型，建立集中的云监管机制，并定义账户和组织管理标准。他们还要确定适合企业使用的成功参考架构和模式。

我们将 CCoE 称为云支持中心，而不是更常见的云卓越中心，以便强调重点是推动所支持的团队取得成功以及实现的业务成果。

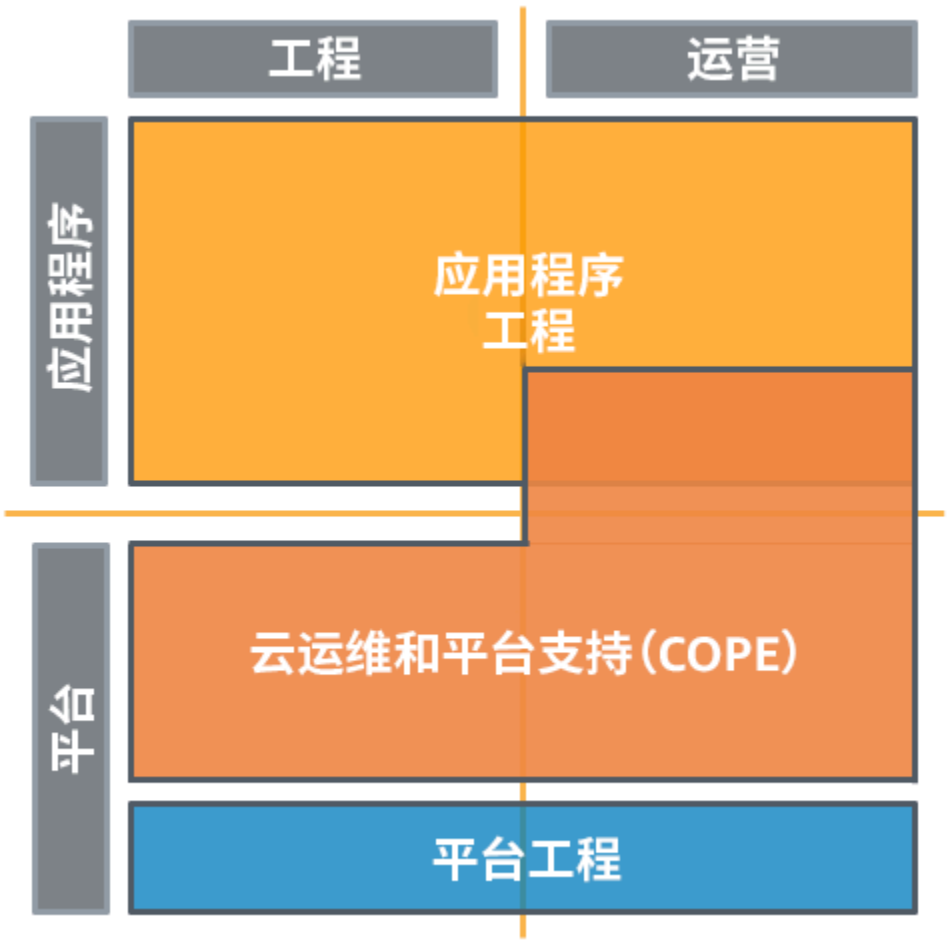
您的平台工程设计团队根据这些标准构建核心共享平台功能，供应用程序团队使用。他们通过自助服务机制，编纂提供给应用程序团队的企业参考架构和模式。使用 AWS Service Catalog 等服务，应用程序团队可以部署经过批准的参考架构、模式、服务和配置，这些自动符合集中监管和安全标准。

平台工程设计团队还可以为应用程序团队提供一套标准化的服务（例如，开发工具、监控工具、备份和恢复工具以及网络）。

您的组织有一个“内部 MSP 和咨询合作伙伴”，负责管理和支持标准化服务，并根据参考体系结构和模式，为应用程序团队在云端打造服务提供帮助。这个“云运维和平台支持（COPE，Cloud Operations and Platform Enablement）”团队与应用程序团队合作，帮助他们建立基准操作，然后随着时间推移，应用程序团队逐步承担起更多的系统和资源责任。COPE 团队与 CCoE 以及平台工程设计团队一起推动持续改进，并作为应用程序团队的支持方。

应用程序团队可以获取帮助来设置环境、CICD 管道、更改管理、可观测性和监控，以及与 COPE 团队一起建立意外事件与事件管理流程，这些流程可根据需要与企业的相应流程集成起来。COPE 团队与应用程序团队一起执行这些运营活动，随着时间的推移，应用程序团队逐步接管这些活动的责任，而 COPE 团队逐步退出参与。

应用程序团队可以从 COPE 团队的技能以及组织学到的经验教训中获益。通过集中监管建立的防护机制为他们提供了保护。应用程序团队建立在经过认可的成功经验的基础之上，并从他们采用的组织标准的持续发展中获益。他们通过建立可观测性和监控流程，可以更深入地了解工作负载的运营，并且能够更好地理解他们所做的更改对工作负载的影响。



COPE 团队保留支持运营活动所需的访问权限，提供跨应用程序团队的企业运营视图，并提供关键的意外事件管理支持。COPE 团队继续负责被认为是无差别的繁重工作，他们通过可大规模提供支持的标准解决方案来完成这些任务。他们还会继续为应用程序团队管理众所周知的程序化、自动化的运营活动，让应用程序团队能够专注于打造独特的应用程序。

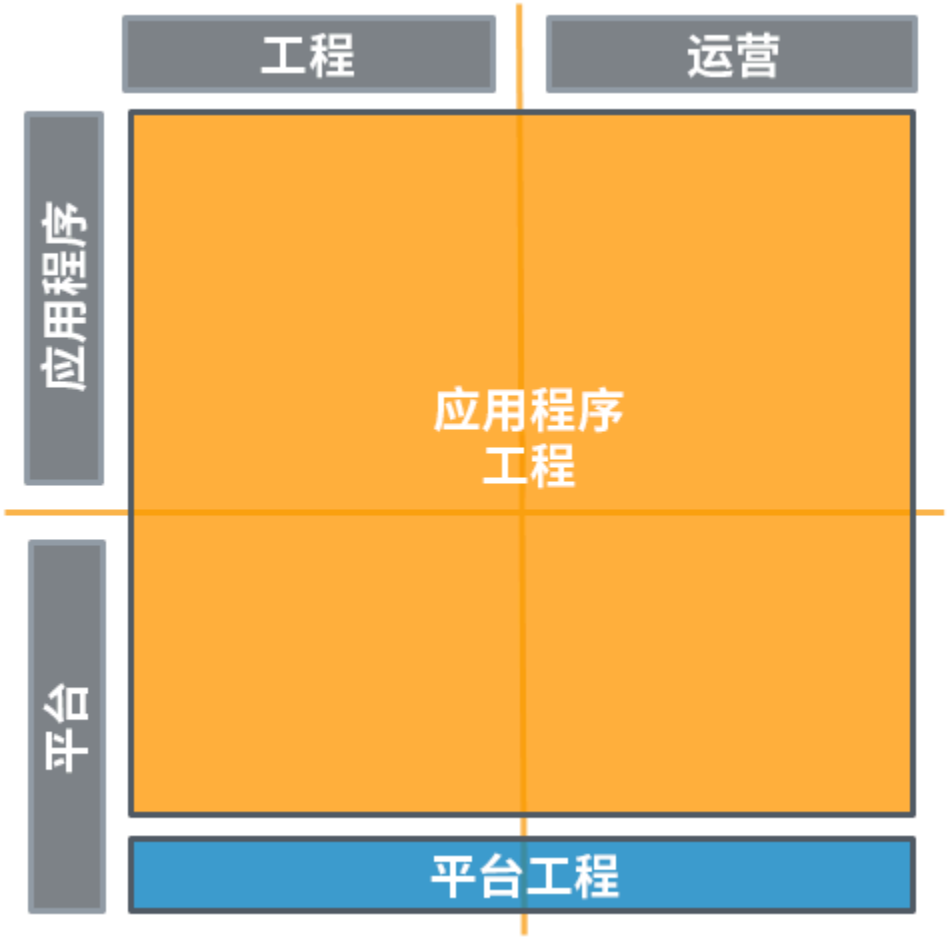
从团队的成功中，您可以获得组织的标准、最佳实践、流程和专业知识方面的进步。您可以建立一种机制，复制这些成功模式供新团队采用，或者在云端实现现代化。此模式强调 COPE 团队帮助建立应用程序团队以及转换知识和构件的能力。它减少了应用程序团队的运营负担，同时也降低了应用程序团队无法高度独立的风险。它在 CCoE、COPE 与应用程序团队之间建立了关系，创建反馈循环用于支持进一步的演进和创新。

在定义组织范围内的标准的同时，建立您的 CCoE 和 COPE 团队可以推动云的采用并支持现代化工作。以顾问和合作伙伴的身份，通过 COPE 团队向应用程序团队提供额外的支持，这可以帮助您消除阻碍应用程序团队采用有益的云功能的障碍。

分离的 AEO 和 IEO，采用分散监管

这种“分离的 AEO 和 IEO”模式采用“你构建，你运行”的方法。

应用程序工程师和开发人员同时执行工作负载工程设计和运营。同样，您的基础设施工程师可以对他们用以支持应用程序团队的平台同时进行工程设计和运营。



在本示例中，我们采用分散监管。

标准仍由平台团队分发、提供或共享给应用程序团队，但是应用程序团队可以自由设计和操作新的平台功能来支持其工作负载。

在此模式中，对应用程序团队的约束较少，但是随之而来的是责任的显著增加。必须具备更多技能以及潜在的团队成员，才能支持其他平台功能。如果缺乏相应技能且不能及早发现缺陷，则会增加大量返工的风险。

您应该执行那些没有专门委托给应用程序团队的策略。您应使用能够跨账户集中监管环境的工具或服务，例如 AWS Organizations.诸如 AWS Control Tower 之类的服务扩展了这一管理功能，使您能够定

义账户设置的蓝图（支持您的运营模式），使用 AWS Organizations 进行持续监管以及自动预置新账户。

为应用程序团队设定可请求添加和变更标准的机制，这作用很大。他们也许能够提出新标准，让其他应用程序团队也因此受益。平台团队可以决定，为这些附加功能提供直接支持是否是对业务成果的有效支持。

由于该模式具有重要技能和团队成员要求，因此限制了创新。它解决了团队之间由于任务转换所造成的诸多瓶颈和延迟，同时还促进了团队与客户之间有效关系的发展。

关系和所有权

您的运营模式定义了团队之间的关系，并为可识别的所有权和责任提供支持。

最佳实践

- OPS02-BP01 确定资源所有者
- OPS02-BP02 确定流程和程序负责人
- OPS02-BP03 确定对运营活动绩效负责的所有者
- OPS02-BP04 制定用于管理责任和所有权的机制
- OPS02-BP05 制定用于请求添加、更改和例外的机制
- OPS02-BP06 预先定义或协商团队间的职责

OPS02-BP01 确定资源所有者

工作负载的资源必须具有已确定的所有者，以便实现变更控制、故障排除和其他功能。为工作负载、账户、基础设施、平台和应用程序分配所有者。使用集中登记册或附加到资源的元数据等工具记录所有权。组件的商业价值指明应用于它们的流程和程序。

期望结果：

- 使用元数据或集中登记册确定资源所有者。
- 团队成员可以确定谁拥有资源。
- 在可能的情况下，账户只有一个所有者。

常见反面模式：

- 未填入 AWS 账户 的备用联系人。
- 资源缺少用于标识哪些团队拥有它们的标记。
- 您的 ITSM 队列没有电子邮件映射。
- 两个团队对一个关键基础设施的所有权出现重叠。

建立此最佳实践的好处：

- 通过分配所有权，资源的变更控制变得非常简单。
- 在排查问题时，您可以让适合的所有者参与进来。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

定义所有权对于环境中的资源使用案例的意义。所有权表示谁监督资源的变更、在排查故障时对资源提供支持或谁在财务上负责。指定并记录资源所有者，包括名称、联系信息、组织和团队。

客户示例

AnyCompany Retail 将所有权定义为控制资源变更和支持的团队或个人。他们利用 AWS Organizations 来管理其 AWS 账户。使用组收件箱配置账户备用联系人。每个 ITSM 队列映射到一个电子邮件别名。标签确定谁拥有 AWS 资源。对于其他平台和基础设施，他们有 Wiki 页面，其中确定了所有权和联系信息。

实施步骤

1. 首先定义企业的所有权。所有权意味着谁承担资源的风险、谁控制对资源的变更，或在排查故障时谁为资源提供支持。所有权还意味着资源的财务或管理所有权。
2. 使用 AWS Organizations 来管理账户。您可以集中管理账户的备用联系人。
   a. 联系信息使用公司拥有的电子邮件地址和电话号码，这样一来，即使其所属员工离开了公司，也不会影响您的正常访问。例如，为账单、运营和安全性创建单独的电子邮件分发列表，并在各个活跃的 AWS 账户中将它们配置为账单、安全性和运营联系人。有多个人会收到 AWS 通知，所以即使有人在度假、变更角色或离开公司，也有其他人能够作出回复。
   b. 如果一个账户未由 AWS Organizations 管理，则在需要时，备用账户联系人可帮助 AWS 与相关人员联系。将账户的备用联系人配置为指向群组而不是指向个人。
3. 使用标签来识别 AWS 资源的所有者。您可以用单独的标签指定所有者及其联系信息。

   a. 您可以使用 AWS Config 规则强制使资源具有所需的所有权标签。

   b. 有关如何为企业构建标记策略的深入指导，请参阅 AWS 标记最佳实践白皮书。

4. 使用 Amazon Q Business，这是一个对话式助手，会使用生成式人工智能来提高员工工作效率、回答问题，并根据企业系统中的信息完成任务。

   a. 将 Amazon Q Business 连接到您公司的数据来源。Amazon Q Business 为超过 40 个受支持的数据来源提供预构建的连接器，包括 Amazon Simple Storage Service（Amazon S3）、Microsoft SharePoint、Salesforce 和 Atlassian Confluence。有关更多信息，请参阅 Amazon Q Business 连接器。

5. 对于其他资源、平台和基础设施，创建用于标识所有权的文档。所有团队成员应该都可以访问此文档。

实施计划的工作量级别：低。利用账户联系信息和标签来分配 AWS 资源的所有权。对于其他资源，您可以使用像 Wiki 中的表格这样简单的东西来记录所有权和联系信息，或使用 ITSM 工具来映射所有权。

资源

相关最佳实践：

- OPS02-BP02 确定流程和程序负责人
- OPS02-BP04 制定用于管理责任和所有权的机制

相关文档：

- AWS Account Management - Updating contact information
- AWS Organizations - Updating alternative contacts in your organization
- AWS 标记最佳实践白皮书
- Build private and secure enterprise generative AI apps with Amazon Q Business and AWS IAM Identity Center
- Amazon Q Business, now generally available, helps boost workforce productivity with generative AI
- AWS Cloud Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

相关讲习会：

- AWS Workshop - Tagging

相关示例：

- AWS Config 规则 – 带有必需标签和有效值的 Amazon EC2

相关服务：

- AWS Config 规则 – required-tags
- AWS Organizations

OPS02-BP02 确定流程和程序负责人

了解谁负责定义各个流程和程序、为何使用这些特定的流程和程序，以及为什么应由此人负责。了解使用特定流程和程序的原因有助于发现改进机会。

期望结果：针对运维任务，企业有一套明确定义并良好维护的流程和程序。流程和程序集中存储在一个位置，可供团队成员使用。按照明确指派的责任归属，经常更新流程和程序。尽可能将脚本、模板和自动化文档作为代码实施。

常见反面模式：

- 流程未记录在案。脚本呈现碎片化，可能分布在许多孤立的操作员工作站上。
- 脚本的使用方法只有少数人了解，或作为团队知识非正式地交流。
- 旧的流程需要更新，但不明确应由谁负责更新，原作者已不在企业中。
- 无法发现流程和脚本，因此在需要时无法使用（例如，在响应意外事件时）。

建立此最佳实践的好处：

- 流程和程序可改进您操作工作负载的工作。
- 新的团队成员可以更快地投入工作中。
- 缩短了缓解意外事件的用时。
- 不同的团队成员（以及不同的团队）可以一致地使用相同的流程和程序。
- 团队可以使用可重复的流程来扩展其流程。

- 在团队之间移交工作负载责任时，标准化的流程和程序有助于减轻移交造成的影响。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

- 确定了负责定义流程和程序的负责人。
  - 确定为支持工作负载而开展的运营活动。将这些活动记录在易于发现的位置。
  - 唯一标识负责活动规范的个人或团队。他们负责确保由技能娴熟且具有正确的权限、访问权限和工具的团队成员来成功执行活动。如果执行活动时遇到问题，那么执行活动的团队成员有责任提供详细反馈，推进活动改进。
  - 通过 AWS Systems Manager 等服务、文档和 AWS Lambda，在活动构件的元数据中收集责任信息。使用标签或资源组收集资源责任信息，详细说明负责人和联系信息。使用 AWS Organizations 创建标记策略，收集负责人和联系信息。
- 随着时间推移，这些程序应该逐步进化为可以作为代码运行，从而减少人工干预的需求。
  - 例如，考虑 AWS Lambda 函数、CloudFormation 模板或 AWS Systems Manager Automation 文档。
  - 在相应的存储库中执行版本控制。
  - 包括适当的资源标记，以便可以轻松识别负责人和文档。

客户示例

AnyCompany Retail 对“负责人”的定义是：负责某个应用程序或应用程序组（共享通用架构实践和技术）的流程的团队或个人。最初，这些流程和程序以分步指南的形式记录在文档管理系统中，可在托管应用程序的 AWS 账户上以及账户中的特定资源组上，使用标签进行发现。他们利用 AWS Organizations 来管理其 AWS 账户。随着时间的推移，这些流程会转换为代码，并使用基础设施即代码（例如 CloudFormation 或 AWS Cloud Development Kit (AWS CDK)模板）定义资源。运维流程成为 AWS Systems Manager 中的自动化文档或 AWS Lambda 函数，这些流程可以作为计划任务启动，用于响应 AWS CloudWatch 警报等事件或 AWS EventBridge 事件，也可以通过 IT 服务管理（ITSM，IT Service Management）平台内的请求启动。所有流程都有标签，用于标识负责人。用于自动化和流程的文档，保存在由该流程的代码存储库生成的 Wiki 页面中。

实施步骤

1. 记录现有的流程和程序。
   a. 查看并保持最新状态。

   b. 确定每个流程或程序的负责人。
   c. 对流程和程序实施版本控制。
   d. 只要可能，对具有相同架构设计的工作负载和环境，分享流程和程序。
2. 建立反馈和改进机制。
   a. 定义审查流程频率的政策。
   b. 定义审核者和审批者流程。
   c. 实施问题队列或票证队列，以便提供和跟踪反馈。
   d. 在可能时，流程和程序应由变更审批委员会（CAB，Change Approval Board）预先审批并进行风险分类。
3. 确认需要运行这些流程和程序的人员能够访问和搜索到它们。
   a. 使用标签来指示可以在哪里访问工作负载的流程和程序。
   b. 使用有意义的错误和事件消息，指明用于解决问题的正确流程或程序。
   c. 使用 Wiki 和文档管理，确保可在整个企业内稳定地搜索流程和程序。
4. 在适当时实现自动化。
   a. 当服务和技术提供 API 时，应开发自动化功能。
   b. 针对流程充分开展培训。开发用户案例和要求，用于实现这些流程的自动化。
   c. 衡量流程和程序的成功使用情况，并提出问题以支持迭代改进。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS02-BP01 确定资源所有者
- OPS02-BP04 制定用于管理责任和所有权的机制
- OPS11-BP04 执行知识管理

相关文档：

- AWS Whitepaper - Introduction to DevOps on AWS
- AWS Whitepaper - Best Practices for Tagging AWS Resources
- AWS Whitepaper - Organizing Your AWS Environment Using Multiple Accounts

- AWS Cloud Operations & Migrations Blog - Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Cloud Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

相关讲习会：

- AWS Well-Architected Operational Excellence Workshop
- AWS Workshop - Tagging

相关视频：

- How to automate IT Operations on AWS
- AWS re:Invent 2020 - Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 - Automating patch management and compliance using AWS (NIS306)
- AWS Supports You - Diving Deep into AWS Systems Manager

相关服务：

- AWS Systems Manager - Automation
- AWS 服务管理连接器

OPS02-BP03 确定对运营活动绩效负责的所有者

了解谁负责针对定义的工作负载执行特定活动，以及为什么负责。了解谁负责执行活动可让我们知晓谁来开展活动、验证结果并向活动所有者提供反馈。

期望结果：

您的企业明确定义了在已定义的工作负载上执行特定活动，以及响应工作负载生成的事件时，需要承担的相关责任。企业记录了流程的所有权和实施方法，并使这些信息可供搜索。您在发生组织变更时审核和更新责任，团队跟踪和衡量缺陷和低效率识别活动的绩效。您实施反馈机制来跟踪缺陷和改进，并支持迭代改进。

常见反面模式：

- 您未记录责任。
- 脚本呈现碎片化，分布在许多孤立的操作员工作站上。只有少数人知道如何使用这些脚本，或者非正式地将其称作团队知识。
- 旧的流程需要更新，但没有人知道该流程的负责人是谁，原作者已不在企业中。
- 无法发现流程和脚本，并且在需要时无法使用（例如，在响应意外事件时）。

建立此最佳实践的好处：

- 您了解谁负责执行活动、需要采取行动时要通知谁，以及谁将执行操作、验证结果并向活动所有者提供反馈。
- 流程和程序可改进您操作工作负载的工作。
- 新的团队成员可以更快地投入工作中。
- 您可以减少用于缓解意外事件的时间。
- 不同的团队使用相同的流程和程序来一致地执行任务。
- 团队可以使用可重复的流程来扩展其流程。
- 在团队之间移交工作负载责任时，标准化的流程和程序有助于减轻移交造成的影响。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要开始定义责任，请从现有文档开始，例如责任矩阵、流程和程序、角色和责任，以及工具和自动化。审核记录的流程责任，并主持围绕流程责任开展的讨论。与团队一起审核，找出文档中的责任和实际流程之间的不一致之处。讨论向该团队的内部客户提供的服务，从而确定团队之间的期望差距。

分析并解决差异。确定改进机会，并寻找经常被请求开展的资源密集型活动，这些活动通常是可改进的有力候选方案。探索最佳实践、模式和规范性指南，以便简化和标准化改进。记录改进机会并一直跟踪改进，直至完成。

随着时间推移，这些程序应该逐步进化为可作为代码运行，从而减少人工干预的需求。例如，程序可以作为 AWS Lambda 函数、AWS CloudFormation 模板或 AWS Systems Manager Automation 文档启动。验证这些程序在相应的存储库中是否受版本控制，并包含适当的资源标记，以便团队能够轻松识别所有者和文档。记录开展活动的责任，然后监控自动化的成功启动和运行，以及期望结果的实现情况。

客户示例

AnyCompany Retail 对“负责人”的定义是：负责某个应用程序或应用程序组（共享通用架构实践和技术）的流程的团队或个人。最初，公司以分步指南的形式将流程和程序记录到文档管理系统中。然后，使用托管应用程序的 AWS 账户上的标签，以及账户内特定资源组上的标签，来使程序可供搜索，并使用 AWS Organizations 管理其 AWS 账户。随着时间的推移，AnyCompany Retail 将这些流程转换为代码，并使用基础设施即代码（通过 CloudFormation 或 AWS Cloud Development Kit (AWS CDK) 模板等服务）定义资源。运维流程成为 AWS Systems Manager 中的自动化文档或 AWS Lambda 函数，这些流程可以作为计划任务启动，用于响应 Amazon CloudWatch 警报等事件或 Amazon EventBridge 事件，也可以通过 IT 服务管理（ITSM，IT Service Management）平台内的请求启动。所有流程都有标签，用于标识其负责人。团队在由该流程的代码存储库生成的 Wiki 页面中，管理用于自动化和流程的文档。

实施步骤

1. 记录现有的流程和程序。
    a. 审核并确认它们是最新的。
    b. 确认每个流程或程序都有负责人。
    c. 对程序实施版本控制。
    d. 只要可能，对具有相同架构设计的工作负载和环境，分享流程和程序。
2. 建立反馈和改进机制。
    a. 定义审查流程频率的政策。
    b. 定义审核者和审批者流程。
    c. 实施问题队列或票证队列，以便提供和跟踪反馈。
    d. 在可能时，流程和程序将由变更审批委员会（CAB，Change Approval Board）预先审批并进行风险分类。
3. 让需要运行这些流程和程序的人员能够访问和搜索到它们。
    a. 使用标签来指示可以在哪里访问工作负载的流程和程序。
    b. 使用有意义的错误和事件消息，指明用于解决问题的正确流程或程序。
    c. 使用 Wiki 和文档管理，确保可在整个企业内一致地搜索流程和程序。
4. 在适当时，实现自动化。
    a. 在服务和技术提供 API 时，开发自动化功能。
    b. 验证流程是否已被充分理解，并开发用户案例和要求以实现这些流程的自动化。
    c. 衡量流程和程序的成功使用情况，并跟踪问题以支持迭代改进。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS02-BP01 确定资源所有者
- OPS02-BP02 确定流程和程序负责人
- OPS02-BP04 制定用于管理责任和所有权的机制
- OPS02-BP05 制定用于确定责任和所有权的机制
- OPS11-BP04 执行知识管理

相关文档：

- AWS Whitepaper | Introduction to DevOps on AWS
- AWS Whitepaper | Best Practices for Tagging AWS Resources
- AWS Whitepaper | Organizing Your AWS Environment Using Multiple Accounts
- AWS Cloud Operations & Migrations Blog | Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Workshop - Tagging
- AWS Service Management Connector

相关视频：

- AWS Knowledge Center Live | Tagging AWS Resources
- AWS re:Invent 2020 | Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 | Automating patch management and compliance using AWS (NIS306)
- AWS Supports You | Diving Deep into AWS Systems Manager

相关示例：

- AWS Well-Architected Operational Excellence Workshop

OPS02-BP04 制定用于管理责任和所有权的机制

了解您的角色具有哪些责任以及如何为业务成果做出贡献，因为这有助于确定任务的优先级以及自身角色的重要性。这有助于团队成员了解需求并做出适当响应。在团队成员知道自己的角色后，他们可以确立所有权，确定改进机会，并了解如何产生影响或做出适当的改变。

有时，一项责任可能没有明确的承担者。在此类情况下，需要设计一种机制来弥补这种不足。创建定义明确的上报路径，上报至有相应权限的人员，由其分配权限或制定计划，来解决这种需求。

期望结果：企业内部的团队具有明确定义的责任，包括他们与资源、要采取的行动、流程和程序的关系。这些责任与该团队的责任和目标以及其他团队的责任保持一致。您可以通过一致且可搜索的方式记录上报路线，并将这些决策输入到文档构件（例如责任矩阵、团队定义或 Wiki 页面）中。

常见反面模式：

- 团队的责任不明确或定义不清。
- 团队的角色与责任不一致。
- 团队的方向性目标和目的与责任不一致，这导致难以衡量成功。
- 团队成员的责任与团队和整个企业的责任不一致。
- 您的团队未及时更新责任，导致责任与团队执行的任务不一致。
- 用于确定责任的上报路径未定义或不明确。
- 上报路径没有单一主线负责人来确保及时响应。
- 无法发现角色、责任和上报路径，也无法在需要时（例如，在响应意外事件时）使用它们。

建立此最佳实践的好处：

- 在您了解谁负责或拥有所有权后，可以与合适的团队或团队成员联系，以提出请求或转换任务。
- 您已确定有权分配责任或所有权的人员，可以降低不作为和需求无法得到满足的风险。
- 在明确定义责任范围后，您的团队成员就能获得自主权和所有权。
- 您的责任可帮助明确所做的决定、采取的行动以及需要将哪些活动交给适当的所有者。
- 轻松确定已放弃的责任，因为您清楚地了解团队责任范围边界，这有助于您上报以进行澄清。
- 团队可以避免混乱和紧张的情况，更充分地管理其工作负载和资源。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

确定团队成员的角色和责任，并确认他们了解角色预期。公示这些信息，以便企业的成员有特定需求时，可以确定需要联系的人员（无论是团队还是个人）。在各个企业寻求利用 AWS 上的迁移与现代化机会时，角色和责任可能会发生变化。让您的团队及其成员了解他们的责任，并对他们进行适当的培训，以便在这一变化期间执行任务。

确定应接受上报的角色或团队，以确定责任和所有权。该团队可以与各种利益相关方互动来作出决策。但是，他们应负责管理决策制定过程。

为企业成员提供可访问机制，以发现和确定所有权和责任。利用这些机制，他们可根据具体需求获知联系对象。

客户示例

AnyCompany Retail 最近通过直接迁移方式，完成了将工作负载从本地环境迁移到 AWS 中的登录区的工作。他们进行了运营审核，以反思完成共同运营任务的方式，并验证了现有的责任矩阵是否体现了新环境中的运营。在他们从本地迁移到 AWS 后，减少了基础设施团队在硬件和物理基础设施方面所承担的责任。这一迁移也为其工作负载的运营模式改进带来了新的机会。

他们已确定、处理并记录大多数责任，还定义了上报路线，来涵盖任何遗漏的责任，或可能需要随运营实践的演变而更改的责任。要探究跨工作负载实现标准化和提高效率的新机会，可提供对运营工具（如 AWSSystems Manager）和安全工具（如 AWS Security Hub 和 Amazon GuardDuty）的访问权限。AnyCompany Retail 根据他们希望首先解决的改进，来审核责任和策略。在公司采用新的工作方式和技术模式时，他们也更新了责任矩阵，以便与之相匹配。

实施步骤

1. 从现有文档开始。一些典型的源文档可能包括：
    a. 责任或负责、问责、咨询和知情（RACI，Responsible, Accountable, Consulted, and Informed）矩阵
    b. 团队定义或 Wiki 页面
    c. 服务定义和产品/服务
    d. 角色或工作描述
2. 审核记录的责任，并主持围绕责任开展的讨论：
    a. 与团队一起进行审核，找出记录的责任与团队通常履行的责任之间的不一致之处。
    b. 讨论内部客户提供的潜在服务，确定团队之间的期望差距。
3. 分析并解决差异。

4. 确定改进机会。
    a. 确定经常提出的资源密集型请求，这些请求通常是可改进的有力候选方案。
    b. 寻找最佳实践、模式和规范性指南，并使用本指南简化和标准化改进。
    c. 记录改进机会，并一直跟踪它们，直至完成。
5. 如果一个团队尚未承担管理和跟踪责任分配的责任，请指定一个团队成员来承担这一责任。
6. 为团队定义一个流程来请求澄清责任。
    a. 审查该流程，确认流程明确并且简单易用。
    b. 确保有人负责和跟踪上报情况，直至得出结论。
    c. 建立运营指标来衡量有效性。
    d. 创建反馈机制，验证团队能否突出改进机会。
    e. 实施定期审核机制。
7. 在可搜索和访问的位置保存文档。
    a. Wiki 或文档门户是常见选择。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS01-BP06 评估权衡
- OPS03-BP02 赋能团队成员在结果有风险时采取行动
- OPS03-BP03 鼓励上报
- OPS03-BP07 为团队配置适当的资源
- OPS09-BP01 使用指标衡量运营目标和 KPI
- OPS09-BP03 审查运营指标并确定改进优先顺序
- OPS11-BP01 设置持续改进流程

相关文档：

- AWS Whitepaper - Introduction to DevOps on AWS
- AWS Whitepaper - AWS Cloud Adoption Framework: Operations Perspective

- AWS Well-Architected Framework Operational Excellence - Workload level Operating model topologies
- AWS Prescriptive Guidance - Building your Cloud Operating Model
- AWS Prescriptive Guidance - Create a RACI or RASCI matrix for a cloud operating model
- AWS Cloud Operations & Migrations Blog - Delivering Business Value with Cloud Platform Teams
- AWS Cloud Operations & Migrations Blog - Why a Cloud Operating Model?
- AWS DevOps Blog - How organizations are modernizing for cloud operations

相关视频：

- AWS Summit Online - Cloud Operating Models for Accelerated Transformation
- AWS re:Invent 2023 - Future-proofing cloud security: A new operating model

OPS02-BP05 制定用于请求添加、更改和例外的机制

您可以向流程、程序和资源的所有者提出请求。请求包括添加、更改和例外。这些请求都要经过变更管理流程。对收益和风险进行评估之后，作出明智的决定，批准可行和确认合适的请求。

期望结果：

- 您可以根据分配的所有权提出变更流程、程序和资源的请求。
- 以慎重的态度作出变更，权衡益处和风险。

常见反面模式：

- 您必须更新部署应用程序的方式，但运营团队无法请求更改部署过程。
- 必须更新灾难恢复计划，但没有可向其请求变更的已确定所有者。

建立此最佳实践的好处：

- 流程、程序和资源会随着需求变化而演进。
- 进行变更时，所有者可以作出明智的决定。
- 以慎重的态度作出变更。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

为实施这种最佳实践，您需要能够请求对流程、程序和资源作出变更。变更管理流程可以很轻巧。记录变更管理流程。

客户示例

AnyCompany Retail 使用责任分配（RACI）矩阵来确定谁控制流程、程序和资源的变更。他们有书面的变更管理流程，这些流程轻巧且易于遵循。使用 RACI 矩阵和流程，任何人都可以提交变更请求。

实施步骤

1. 确定工作负载的流程、程序和资源及它们各自的所有者。在知识管理系统中记录它们。
   a. 如果您未实施 OPS02-BP01 确定资源所有者、OPS02-BP02 确定流程和程序负责人 或 OPS02-BP03 确定对运营活动绩效负责的所有者，请先从实施这些项开始。
2. 与您组织中的利益攸关方合作，制定变更管理流程。该流程应涵盖资源、流程和程序的添加、更改和例外。
   a. 您可以使用 AWS Systems Manager Change Manager 作为工作负载资源的变更管理平台。
3. 在知识管理系统中记录变更管理流程。

实施计划的工作量级别：中等。制定变更管理流程需要与整个组织的多个利益攸关方达成一致。

资源

相关最佳实践：

- OPS02-BP01 确定资源所有者 - 在构建变更管理流程之前，需要确定资源的所有者。
- OPS02-BP02 确定流程和程序负责人 - 在构建变更管理流程之前，需要确定流程的所有者。
- OPS02-BP03 确定对运营活动绩效负责的所有者 - 在构建变更管理流程之前，需要确定运营活动的所有者。

相关文档：

- AWS 规范性指南 - AWS 大型迁移的基础行动手册：创建 RACI 矩阵
- 云中的变更管理白皮书

相关服务：

- AWS Systems Manager Change Manager

OPS02-BP06 预先定义或协商团队间的职责

团队之间具有明确或协商好的协议，规定了团队之间的合作和相互支持方式（例如响应时间、服务级别目标或服务等级协议）。记录团队间沟通渠道。了解团队工作对业务成果以及其他团队和组织的成果的影响，可以确定其任务的优先级，并帮助他们做出适当的响应。

当责任和所有权不确定或未知时，您将面临以下风险：没有及时处理必要的活动，以及在处理这些需求时可能出现工作冗余和潜在冲突。

期望结果：

- 商定并记录团队间工作或支持协议。
- 相互支持或合作的团队有明确的沟通渠道和响应期望。

常见反面模式：

- 生产中出现问题，两个单独的团队开始彼此独立地排查问题。他们各自为政，这延长了中断时间。
- 运营团队需要开发团队提供帮助，但没有商定好响应时间。请求卡滞在待办事项中。

建立此最佳实践的好处：

- 团队知道如何互动和相互支持。
- 知道响应期望。
- 明确定义沟通渠道。

在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

实施这种最佳实践意味着明确团队相互合作的方式。正式协议规定了团队如何协同工作或相互支持。记录团队间沟通渠道。

客户示例

AnyCompany Retail 的 SRE 团队与其开发团队达成服务等级协议。开发团队在其工单系统中提出请求后，预计可以在十五分钟内得到答复。如果站点发生中断，则 SRE 团队在开发团队的支持下主导调查。

实施步骤

1. 与整个组织的利益攸关方合作，根据流程和程序在团队之间达成一致。
   a. 如果在两个团队之间分享了流程或程序，则编制有关团队如何协同工作的运行手册。
   b. 如果团队之间存在依赖关系，请商定请求的响应 SLA。
2. 在知识管理系统中记录责任。

实施计划的工作量级别：中等。如果团队之间还没有达成一致，则需要努力与组织中的利益攸关方达成一致。

资源

相关最佳实践：

- OPS02-BP02 确定流程和程序负责人 - 在团队之间达成协议之前，必须先确定流程所有权。
- OPS02-BP03 确定对运营活动绩效负责的所有者 - 在团队之间达成协议之前，必须先确定运营活动所有权。

相关文档：

- AWS Executive Insights - 利用双披萨团队助力创新
- AWS 上的 DevOps 简介 - 双披萨团队

组织文化

为您的团队成员提供支持，以便他们可以更有效地采取行动 并为您的业务成果提供支持。

最佳实践

- OPS03-BP01 提供高管支持
- OPS03-BP02 赋能团队成员在结果有风险时采取行动
- OPS03-BP03 鼓励上报
- OPS03-BP04 沟通及时、清晰、可行

- OPS03-BP05 鼓励试验
- OPS03-BP06 鼓励团队成员保持和增强自己的技能组合
- OPS03-BP07 为团队配置适当的资源

OPS03-BP01 提供高管支持

在最高层面，高层领导作为执行发起人，为组织的结果明确设定期望和方向，包括评估成果成功与否。发起人倡导并推动最佳实践的采用和组织的发展壮大。

期望结果：努力采用、改造和优化云运营的组织应建立明确的领导关系，并对预期结果负责。组织了解完成新结果所需的每项能力，并授权职能团队进行对应能力的培养。领导层要积极确定这一方向、合理授权、承担责任并界定工作。因此，整个组织中的每个人都能动员起来，受到鼓舞，并积极努力实现预期目标。

常见反面模式：

- 工作负载所有者有义务将工作负载迁移到 AWS，但却没有明确的发起人和云运营规划。这就导致团队不能有意识地开展合作，来提高业务能力并使之成熟。缺乏运营最佳实践标准（例如操作员疲劳、随时待命和技术债务），使团队不堪重负，从而限制了创新。
- 在没有领导层发起人和战略的情况下，就在整个组织范围内设定了采用新兴技术的新目标。各团队对目标的理解各不相同，这就在工作重点、目标为何重要以及如何衡量影响等方面造成了混乱。因此，组织会失去采用技术的动力。

建立此最佳实践的好处：当高管明确传达和分享愿景、努力方向和目标时，团队成员就会知道对他们的期望是什么。当领导者积极参与时，个人和团队就会开始集中精力朝着同一个方向努力，以完成既定目标。因此，组织极大限度地提高了成功的能力。当您评估成功时，您可以更好地识别成功之路上的障碍，以便通过执行发起人的干预来克服这些障碍。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

- 在云之旅的每个阶段（迁移、采用或优化），成功都需要最高领导层的积极参与，并指定一名执行发起人。执行发起人能够使团队的思维方式、技能组合和工作方法与既定战略保持一致。
  - 解释原因：阐明并解释愿景和战略背后的原因。
  - 设定期望：确定并公布组织目标，包括如何衡量进展和成功。

- 跟踪目标实现情况：定期衡量目标的逐步实现情况（而不仅仅是任务的完成情况）。分享结果，以便在结果面临风险时可以采取适当的行动。
- 为实现目标提供必要的资源：将人员和团队团结在一起，共同协作，制定恰当的解决方案，实现既定成果。这可以减少乃至消除组织内部的摩擦。
- 支持您的团队：保持与团队的互动，以便了解他们的表现以及是否有外部因素影响他们。确定阻碍团队进度的障碍。代表团队做出行动，帮助消除障碍，除去不必要的负担。团队受外部因素影响时，需重新评估目标并适当地调整执行性目标。
- 推动采用最佳实践：认可能够量化收益的最佳实践，并表彰创造者和采用者。鼓励进一步采用，实现更大收益。
- 鼓励团队发展：营造一种持续改进的文化，积极主动地从取得的进步和失败中吸取经验教训。鼓励个人和组织的成长与发展。利用数据和轶事来发展愿景和战略。

客户示例

AnyCompany Retail 正在通过快速重塑客户体验、提高生产力，以及利用生成式人工智能加速增长，来实现业务转型。

实施步骤

1. 建立单线程领导层，指派一名主要执行发起人来领导和推动转型。
2. 明确转型的业务成果，分配所有权和责任。赋予主要负责人领导和作出关键决策的权力。
3. 确认您的转型战略非常明晰，并由执行发起人广泛传达给组织的每一个层级。
    a. 为 IT 和云计划明确制定业务目标。
    b. 记录关键业务指标，推动 IT 和云转型。
    c. 向负责各个战略部分的所有团队和个人持续传达愿景。
4. 制定沟通规划矩阵，明确需要向特定的领导、管理人员和个人贡献者传递哪些信息。指定应传递此信息的人员或团队。
    a. 持续可靠地完成沟通计划。
    b. 通过定期的面对面活动来设定和管理期望值。
    c. 接受有关沟通效果的反馈，并相应地调整沟通和计划。
    d. 安排沟通活动，主动了解各个团队提出的挑战，并建立持续的反馈环路，以便在必要时纠正方向。
5. 从领导层的角度积极参与每项计划，以便核实所有受影响的团队是否都了解他们负责实现的成果。

6. 在每次状态会议上，执行发起人都应寻找阻碍因素，检查既定指标、轶事或团队反馈，并衡量实现目标的进展情况。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS03-BP04 沟通及时、清晰、可行
- OP11-BP01 设置持续改进流程
- OPS11-BP07 审核运营指标

相关文档：

- Untangling Your Organisational Hairball: Highly Aligned
- The Living Transformation: Pragmatically approaching changes
- Becoming a Future-Ready Enterprise
- 7 Pitfalls to Avoid When Building a CCOE
- Navigating the Cloud: Key Performance Indicators for Success

相关视频：

- AWS re:Invent 2023: A leader's guide to generative AI: Using history to shape the future (SEG204)

相关示例：

- Prosci: Primary Sponsor's Role & Importance

OPS03-BP02 赋能团队成员在结果有风险时采取行动

由领导层灌输的主人翁文化行为，会让任何员工感到自己有能力代表整个公司行事，超越为其规定的职责和责任范围。员工可以在风险出现时主动识别风险并采取适当行动。这样的文化使员工能够在了解情况的前提下作出高价值的决策。

例如，Amazon 将领导力准则作为指导原则，以便推动员工的预期行为，使其在各种情况下取得进展、解决问题、处理冲突并采取行动。

期望结果：领导层倡导了一种新的文化，这种文化使个人和团队可以作出关键决策，即使在组织中职级不高，只要决策有可审计的权限和安全机制，就可以作出关键决策。失败并不可怕，团队会不断学习，改进决策和应对措施，以应对今后出现的类似情况。如果某个人的行动带来了改进，能让其他团队受益，这些团队就会主动分享这些行动带来的知识。领导层衡量业务改进情况，并激励个人和组织采用此类模式。

常见反面模式：

- 组织内没有明确的指导或机制来说明在发现风险时该怎么做。例如，当员工发现网络钓鱼攻击时，他们没有向安全团队报告，导致组织中的大部分人遭受攻击。这会造成数据泄露。
- 您的客户抱怨服务不可用，主要原因是部署失败。您的 SRE 团队负责部署工具，而他们的长期路线图中包括自动回滚部署。在最近的一次应用程序推广中，一位工程师设计了一种解决方案，可以自动将应用程序回滚到以前的版本。虽然他们的解决方案可以成为 SRE 团队采用的模式，但其他团队并不采用，因为没有流程跟踪此类改进。组织继续受到部署失败的困扰，这影响了客户，造成了更多负面情绪。
- 为了保持合规性，您的信息安全团队会监督一个长期建立的流程，代表连接到 Amazon EC2 Linux 实例的操作员定期轮换共享 SSH 密钥。信息安全团队需要花几天的时间才能完成密钥的轮换，而且您会被阻止连接到这些实例。信息安全团队内外没有人建议使用 AWS 上的其他选项来实现相同的结果。

建立此最佳实践的好处：通过下放决策权以及授权团队作出关键决策，您能够更快地解决问题，并提高成功率。此外，团队开始意识到主人翁意识，失败是可以接受的。实验成为一种文化主流。经理和主管不会觉得他们在工作的各个方面都受到微观管理。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

1. 培养一种预期会发生失败的文化。
2. 明确规定组织内各职能领域的所有权和责任。
3. 向每个人传达所有权和问责制，让大家都知道谁能帮助他们促进分散决策。
4. 定义您的单向门决策和双向门决策，让个人了解何时确实需要上报给更高级别的领导。
5. 树立组织意识，让所有员工都有能力在结果面临风险时，从各个层面采取行动。为团队成员提供治理文件、权限级别、工具，还提供机会，让团队成员练习有效应对所需的技能。

6. 让团队成员有机会练习应对各种决策所需的技能。一旦确定了决策级别，就应开展“演练日”活动，以便确保所有参与人员都能理解并演示流程。
   a. 提供替代的安全环境，以便在其中对流程和程序进行测试和培训。
   b. 承认并使团队成员认识到，当结果达到预先确定的风险水平时，他们有权采取行动。
   c. 通过为团队成员所支持的工作负载和组件分配权限和访问权限，定义团队成员的行动权限。
7. 让团队能够分享他们的经验（运营方面的成功和失败）。
8. 授权团队挑战现状，并建立一些机制，让团队跟踪和衡量改进情况及其对组织的影响。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS01-BP06 在管理效益和风险的同时评估各种权衡因素
- OPS02-BP05 制定用于确定责任和所有权的机制

相关文档：

- AWS 博客文章 | The agile enterprise
- AWS 博客文章 | Measuring success : A paradox and a plan
- AWS 博客文章 | Letting go : Enabling autonomy in teams
- Centralize or Decentralize?

相关视频：

- re:Invent 2023 | How to not sabotage your transformation (SEG201)
- re:Invent 2021 | Amazon Builders' Library: Operational Excellence at Amazon
- Centralization vs. Decentralization

相关示例：

- Using architectural decision records to streamline technical decision-making for a software development project

OPS03-BP03 鼓励上报

领导层鼓励团队成员在认为预期结果面临风险和预期标准未得到满足时，将问题和疑虑上报给更高层次的决策者和利益相关者。这是组织文化的一个特点，并在各个层面得到推动。应经常尽早上报，以便能够确定风险，并防止造成意外事件。领导层不会训斥将问题上报的个人。

期望结果：整个组织的每个人都乐于将问题上报给直属领导和更高层领导。领导层有意识地建立一种期望，让他们的团队可以毫无顾虑地上报任何问题。在组织内部的每个层级，都有上报问题的机制。当员工将问题上报给经理时，他们共同决定问题的影响程度以及是否应该上报。要启动上报程序，员工需要提交一份解决问题的建议工作计划。如果直属管理层没有及时采取行动，而员工强烈认为组织面临的风险需要上报，则组织鼓励他们将问题提交给最高领导层。

常见反面模式：

- 在云转型项目状态会议上，执行领导没有提出足够的探究性疑问来发现问题和阻碍因素。大家都报喜不报忧。首席信息官 (CIO) 明确表示，她只喜欢听到好消息，因为提出的任何挑战都会让首席执行官 (CEO) 认为项目要失败。
- 您是一名云运营工程师，您注意到应用团队并未广泛采用新的知识管理系统。公司花了一年时间并投资了数百万美元，来实施这一新的知识管理系统，但人们仍在本地编写运行手册，并在组织云共享上共享这些手册，因此很难找到与支持的工作负载相关的知识。您努力让领导层注意到这一点，因为坚持使用这一系统可以提高运营效率。当您向领导实施知识管理系统的主管提出这个问题时，她会斥责您，因为这使投资受到质疑。
- 负责强化计算资源的信息安全团队决定实施一项流程，要求在计算团队发布资源以供使用之前，进行必要的扫描，确保 EC2 实例完全安全。这导致资源的部署时间又延迟了一周，违反了他们的 SLA。计算团队不敢将此事上报给负责云事项的副总裁，因为这会让信息安全副总裁难堪。

建立此最佳实践的好处：

对于复杂问题或关键问题，在其对业务产生影响之前就加以解决。减少时间浪费。大幅降低风险。团队在解决问题时会更加积极主动，更加注重结果。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在组织的各个层面中自由上报的意愿和能力，是一种组织和文化基础，应通过强调培训、领导层沟通、设定期望，以及在整个组织的各个层面部署机制，来有意识地培养。

实施步骤

1. 制定组织的政策、标准和期望。
    1. 确保政策、期望和标准得到广泛采纳和理解。
2. 鼓励、培训工作人员，并赋予他们权力，以便在不符合标准时他们会尽早、频繁地上报。
3. 从组织的角度确认，及早和频繁上报是最佳实践。接受上报的内容最终可能证明并无依据，但最好要抓住机会预防意外事件的发生，而不要因为没有上报而错失机会。
    a. 建立上报机制（如 Andon cord 系统）。
    b. 制定成文的程序，规定何时以及如何上报。
    c. 确定一系列有各级权力来采取或批准行动的人员，以及每个利益相关者的联系信息。
4. 当上报发生时，应有始有终，直到团队成员认为领导层推动的行动可以充分降低风险，并对结果满意。
    a. 上报内容应包括：
        i. 情况描述和风险性质
        ii. 情况的严重性
        iii. 受影响的人或事
        iv. 影响有多大
        v. 发生影响时的紧迫性
        vi. 建议的补救措施和减轻影响的计划
    b. 保护上报的员工。制定政策保护团队成员，如果他们上报关于决策者或利益相关者未做出响应的问题，保护他们免遭报复。制定适当的机制，确定是否发生了这种情况并做出相应响应。
5. 鼓励在组织的所有事项中建立持续改进的反馈环路文化。反馈环路起到向责任人进行小规模上报的作用，即使不需要上报，也能发现改进机会。持续改进的文化促使每个人更加积极主动。
6. 领导层应定期重新强调政策、标准、机制，以及公开上报和持续反馈环路而不受到报复的愿望。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS02-BP05 制定用于请求添加、更改和例外的机制

相关文档：

- How do you foster a culture of continuous improvement and learning from Andon and escalation systems?
- The Andon Cord (IT Revolution)
- AWS DevOps Guidance | Establish clear escalation paths and encourage constructive disagreement

相关视频：

- Jeff Bezos on how to make decisions (& increase velocity)
- Toyota Product System: Stopping Production, a Button, and an Andon Electric Board
- Andon Cord in LEAN Manufacturing

相关示例：

- Working with escalation plans in Incident Manager

OPS03-BP04 沟通及时、清晰、可行

领导层有责任建立强有力的有效沟通，尤其是在组织采用新战略、新技术或新工作方式时。领导者应为所有员工设定期望，让他们为实现公司目标而努力。设计沟通机制，在负责实施由领导层资助和赞助的计划的团队中，树立和保持意识。利用跨组织的多样性，认真倾听多种独特观点。利用这种见解提高创新能力、对您的假设提出质疑，并降低确认偏差的风险。培养团队的包容性、多样性和无障碍性，以获得有益的观点。

期望结果：您的组织制定沟通策略，用来应对变革对组织的影响。团队保持信息畅通，有动力继续相互合作，而不是相互竞争。个人明白自己的角色对于实现既定目标有多么重要。电子邮件只是一种被动的通信机制，因此要合理使用。管理层花时间与个人贡献者沟通，让他们了解自己的责任、要完成的任务，以及他们的工作如何为整体使命做出贡献。必要时，领导者在规模较小的场合直接与员工接触，传达信息并核实这些信息是否得到有效传达。由于沟通策略良好，组织的表现达到或超过领导层的期望。领导层鼓励并征求团队内部和团队之间的不同意见。

常见反面模式：

- 贵组织有一个五年计划，要将所有工作负载迁移到 AWS。云业务案例包括对 25% 的工作负载进行现代化改造，以便利用无服务器技术。CIO 将这一战略传达给直接下属，并希望每位领导将这一战略传达给经理、总监和个人贡献者，而无需任何面对面的沟通。CIO 退居幕后，期望组织能够执行新战略。

- 领导层不提供或不使用反馈机制，期望差距变得越来越大，从而导致项目停滞不前。
- 有人要求您对安全组进行变革，但却没有告诉您详细信息，例如需要进行哪些变革，变革会对所有工作负载产生什么影响，以及何时进行变革等。经理转发了一封来自信息安全副总裁的电子邮件，并添加了信息 "Make this happen."
- 您的迁移战略发生了变化，计划的现代化改造数量从 25% 减少到 10%。这会对运营组织的下游产生影响。下游组织未被告知这一战略变化，因此没有足够的技术能力协助将更多的工作负载直接迁移到 AWS。

建立此最佳实践的好处：

- 您的组织对新战略或更改后的战略了如指掌，他们会积极采取相应行动，协助彼此实现领导层设定的总体目标和指标。
- 制定相应机制，用于将已知风险和计划内事件及时通知给团队成员。
- 新的工作方式（包括人员、组织、流程或技术的变化）以及所需的技能会更有效地为组织所采用，从而使组织更快地实现业务效益。
- 团队成员可以了解所接收信息的必要背景，从而更有效地开展工作。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

为实施这种最佳实践，您必须与整个组织的利益攸关方合作，商定沟通标准。向您的组织公布这些标准。对于任何重大的 IT 过渡，与忽视这一做法的组织相比，一个成熟的规划团队能够更成功地管理变革对员工的影响。规模较大的组织在管理变革时可能更具挑战性，因为要让所有个人贡献者对新战略产生强烈的认同感，这一点至关重要。如果缺乏这样的过渡规划团队，就需要领导层对有效沟通全权负责。在建立过渡规划团队时，指派团队成员与所有组织领导层合作，以便规定和管理各个层级的有效沟通。

客户示例

AnyCompany Retail 注册了 AWS Enterprise Support，并依赖其他第三方提供商进行云运营。该公司将聊天和聊天室作为业务活动的主要沟通媒介。警报和其他信息会填入特定渠道。当有人必须采取行动时，他们会清楚地说明预期的结果，而且在很多情况下，他们会收到一份运行手册或行动手册以供使用。他们通过一个变更日历来安排生产系统的重大变更。

实施步骤

1. 在组织内建立一个核心团队，负责为组织内多个层面的变革制定和启动沟通计划。

2. 建立单线程所有权，以便实现监督。赋予各个团队独立创新的能力，并平衡使用一致的机制，从而实现适当程度的检查和方向性愿景。
3. 与整个组织内部的利益相关者合作，就沟通标准、实践和计划达成一致。
4. 核实核心沟通团队是否与组织和项目领导层合作，代表领导者向相关人员传达信息。
5. 建立战略沟通机制，通过公告、共享日历、全体员工会议、面对面或一对一的方式管理变革，使团队成员对自己应采取的行动有正确的预期。
6. 提供必要的背景、细节和时间（如有可能），以便确定是否有必要采取行动。需要采取行动时，提供所需的行动及其影响。
7. 实施促进战术沟通的工具，例如内部聊天、电子邮件和知识管理。
8. 实施各种机制，以便衡量和核实所有沟通活动是否都取得了预期结果。
9. 建立反馈环路，衡量所有沟通的效果，尤其是当沟通涉及到整个组织对变革的抵触时。
10. 对于所有 AWS 账户，设立账单、安全和运营方面的备用联系人。理想情况下，每个联系人都应是电子邮件分发的收件人，而不是特定的个人联系人。
11. 制定上报和逆向上报沟通计划，与您的内部团队和外部团队（包括 AWS Support 和其他第三方提供商）进行沟通。
12. 在每个转型计划的整个生命周期内，始终如一地启动和执行沟通策略。
13. 优先考虑可重复执行的行动，尽可能安全地实现大规模自动化。
14. 当需要在自动化操作的场景中进行通信时，通信目的应该是通知团队、进行审计或作为变更管理流程的一部分。
15. 分析来自警报系统的通信，判断误报或不断生成的警报。删除或更改这些警报，使其在需要人工干预时启动。如果启动了警报，则提供运行手册或行动手册。
    a. 您可以使用 AWS Systems Manager 文档为警报编制行动手册和运行手册。
16. 制定合理的机制，以清晰、可操作的方式提供风险或计划内事件的通知，而且要引起足够的注意，以做出适当的响应。使用电子邮件列表或聊天频道在计划内事件之前发送通知。
    a. AWS Chatbot 可用于在组织的消息传送平台内发送警报和响应事件。
17. 提供可访问的信息源，其中包含计划内事件。通知来自同一系统的计划内事件。
    a. AWS Systems Manager Change Calendar 可用于创建变更时段，指明何时会发生变更。因而在团队成员可以安全地进行变更时，为他们提供通知。
18. 监控漏洞通知和补丁程序信息，以了解外部漏洞以及与工作负载组件相关的潜在风险。向团队成员发送通知，以便他们可以采取行动。
    a. 您可以订阅 AWS 安全公告，以便接收有关 AWS 上漏洞的通知。

19.寻求不同的观点和视角：鼓励所有人做出贡献。为代表性不足的群体提供沟通机会。在会议中轮换角色和职责。

a. 扩展角色和职责：让团队成员有机会尝试他们可能不会担任的角色。他们可以从角色以及与其他团队成员的互动中获得经验和见解，而之前可能并没有机会与这些成员互动。他们还可以将自己的经验和见解赋予新角色，以及就此与新团队成员沟通交流。随着见解不断增多，确定新出现的商业机会或新的改进机会。在团队成员之间轮流执行其他人通常执行的日常任务，了解执行这些任务的需求和影响。

b. 提供安全舒适的环境：制定政策和控制措施，保护组织内团队成员的身心安全。团队成员应该能够彼此敞开心扉，而不是处在会受到报复的担惊受怕之中。当团队成员处于安全舒适的环境中时，才能有更高的参与热情、更高的工作成效。您的组织越多元化，您就越能更好地理解您所支持的人，包括客户。当您的团队成员感到舒服自在、能够畅所欲言并确信自己的意见会被听到时，他们会更愿意分享有价值的见解（例如营销机会、可访问性需求、尚待开发的细分市场以及环境中未被发现的风险）。

c. 鼓励团队成员充分参与：为员工提供必要的资源，让他们充分参与到所有与工作相关的活动中。团队成员每天都要面对挑战，他们需要掌握应对挑战的技能。这些独特发展的技能可以为您的组织带来巨大的效益。为团队成员提供必要的后勤保障，让他们的贡献为您带来更多的收益。

资源

相关最佳实践：

- OPS03-BP01 提供高管支持
- OPS07-BP03 使用运行手册执行程序
- OPS07-BP04 根据行动手册调查问题

相关文档：

- AWS 博客文章 | Accountability and empowerment are key to high-performing agile organizations
- AWS Executive Insights | Learn to scale innovation, not complexity | Single-threaded Leaders
- AWS 安全公告
- Open CVE
- AWS Support App in Slack to Manage Support Cases
- Manage AWS resources in your Slack channels with AWS Chatbot

相关示例：

- Well-Architected Labs: Inventory and Patch Management (Level 100)

相关服务：

- AWS Chatbot
- AWS Systems Manager Change Calendar
- AWS Systems Manager Documents

OPS03-BP05 鼓励试验

试验是将新想法转化为产品和功能的催化剂。它可加快学习速度，并使团队成员保持兴趣和参与热情。鼓励团队成员经常试验，以便推动创新。即使出现了不希望看到的结果，我们知道什么不该做也是有价值的。团队成员不会因为试验成功但结果不理想而受到惩罚。

期望结果：

- 您的组织鼓励试验以促进创新。
- 将试验当作学习的机会。

常见反模式：

- 您想要运行 A/B 测试，但没有运行试验的机制。您部署了 UI 更改，但无法对其进行测试。这会造成负面的客户体验。
- 您的公司只有一个模拟和生产环境。没有沙盒环境来试验新功能或产品，因此您必须在生产环境中进行试验。

建立此最佳实践的好处：

- 试验推动创新。
- 通过试验，您可以更快地对用户的反馈作出反应。
- 您的组织发展了一种学习的文化。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

试验应以安全的方式进行。利用多个环境来试验，而不危及生产资源。使用 A/B 测试和功能标记来测试试验。使团队成员能够在沙盒环境中执行试验。

客户示例

AnyCompany Retail 鼓励试验。团队成员可以每周使用 20% 的工作时间来试验或学习新技术。他们有可以实现创新的沙盒环境。为新功能使用 A/B 测试，用真实的用户反馈来验证它们。

实施步骤

1. 与整个组织的领导层合作以支持试验。应鼓励团队成员以安全的方式进行试验。
2. 为团队成员提供可以安全进行试验的环境。他们必须能够访问类似于生产的环境。
    a. 您可以使用单独的 AWS 账户 来创建用于试验的沙盒环境。AWS Control Tower 可用于预置这些账户。
3. 使用功能标记和 A/B 测试安全地试验和收集用户反馈。
    a. AWS AppConfig Feature Flags 可用于创建功能标记。
    b. Amazon CloudWatch Evidently 可用于在有限的部署上运行 A/B 测试。
    c. 您可以使用 AWS Lambda 版本来部署一项功能的新版本以进行 Beta 测试。

实施计划的工作量级别：高。为团队成员提供试验环境和进行试验的安全方法需要大量投资。您可能还需要修改应用程序代码以使用功能标记或支持 A/B 测试。

资源

相关最佳实践：

- OPS11-BP02 在意外事件发生后执行分析 - 从事件中学习是创新和试验的重要驱动因素。
- OPS11-BP03 实施反馈环路 - 反馈环路是试验的重要部分。

相关文档：

- 深入了解亚马逊文化：试验、失败和客户至尚
- 在 AWS 中创建和管理沙盒账户的最佳实践
- 营造由云支持的试验文化

- 在 SulAmérica Seguros 实现云中试验和创新
- 试验更多，失败更少
- 使用多个账户组织 AWS 环境 - 沙盒 OU
- 使用 AWS AppConfig Feature Flags

相关视频：

- AWS On Air，主讲：Amazon CloudWatch Evidently | AWS Events
- AWS On Air San Fran Summit 2022，主讲：AWS AppConfig Feature Flags 与 Jira 集成
- AWS re:Invent 2022 - 部署不是发布：使用功能标记控制您的启动（BOA305-R）
- 使用 AWS Control Tower 以编程方式创建 AWS 账户
- 设置使用 AWS Organizations 最佳实践的多账户 AWS 环境

相关示例：

- AWS 创新沙盒
- 适合电子商务的端到端个性化 101

相关服务：

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 鼓励团队成员保持和增强自己的技能组合

团队必须增强自己的技能组合，以采用新技术；并随需求和职责的变化继续提供支持，以支持工作负载。新技术技能的增强通常能提升团队成员满意度并支持创新。支持您的团队成员获取和维护行业认证，以验证和认可他们不断增强的技能。进行交叉培训，以促进知识转移并降低在您失去熟练掌握机构知识、经验丰富的团队成员时产生重大影响的风险。专门安排时间进行学习。

AWS 提供了许多资源，包括 AWS 入门资源中心、AWS 博客、AWS 在线技术讲座、AWS 事件和网络研讨会，以及 AWS Well-Architected 实验室，这些资源提供了指导、示例和详细演练，用以培训您的团队。

AWS Support（AWS re:Post、AWS Support 中心）和 AWS 文档等资源有助于消除技术障碍并改善运营。请通过 AWS Support 中心联系 AWS Support，协助解决您的问题。

AWS 还在 The Amazon Builders' Library 中分享了我们通过 AWS 运营学到的最佳实践和模式；并通过 AWS 博客和 AWS 官方播客分享了各种实用的教材。

AWS 培训和认证包括通过自定进度的数字课程进行的免费培训，以及按角色或领域划分的学习计划。您还可以报名参加讲师指导培训，进一步培养您团队的 AWS 技能。

期望结果：您的组织不断评估技能差距，并通过结构化预算和投资来弥补这些差距。团队鼓励和激励其成员开展提高技能的活动，例如获得领先的行业认证。团队利用专门的知识交叉共享计划，例如午餐学习、沉浸日、黑客马拉松和实际演练活动。贵组织及时更新知识系统，并保持与交叉培训团队成员的相关性，包括新员工入职培训。

常见反面模式：

- 在缺乏结构化培训计划和预算的情况下，团队在努力跟上技术发展步伐的过程中会遇到不确定性，从而导致人员流失增加。
- 在向 AWS 迁移的过程中，贵组织表现出团队之间存在技能差距和不同的云熟悉度。如果不努力提高技能，团队就会受累于传统且效率低下的云环境管理，并导致操作员不堪重负。这种倦怠感会增加员工的不满情绪。

建立此最佳实践的好处：当贵组织有意识地进行投资以期提高团队技能时，也有助于加快和扩展云技术的采用和优化。有针对性的学习计划可推动创新，培养团队的业务能力，为处理各种事件做好准备。团队有意识地投资于最佳实践的实施和发展。团队士气高昂，团队成员重视自己对企业的贡献。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

为了采用新技术、推动创新、跟上需求和责任的变化，以便为您的工作负载提供支持，请持续投资于团队的专业发展。

实施步骤

1. 使用结构化的云宣传计划：AWS Skills Guild 提供咨询培训，以期增强对云技能的信心，并点燃持续学习的文化氛围。
2. 提供教育资源：专门安排时间，提供培训材料和实验室资源，并支持参加会议和加入专业组织，以便有机会向讲师和同行学习。让初级团队成员有机会接触资深团队成员作为导师，或者让初级团队

成员跟随资深团队成员工作，接触后者的工作方法和技能。鼓励学习与工作没有直接关系的内容，拓展视野。

3. 鼓励使用专家技术资源：利用 AWS re:Post 等资源，获取精选知识，加入充满活力的社区。
4. 建立和维护最新的知识库：使用知识共享平台，例如 Wiki 和运行手册。利用 AWS re:Post Private 创建您自己的可重复使用的专家知识来源，以便简化协作、提高工作效率并加快员工上岗速度。
5. 团队教育和跨团队合作：为团队成员的继续教育需求做好规划。为团队成员提供（临时或永久）加入其他团队的机会，以分享技能和最佳实践，惠及整个组织。
6. 支持获取和维护行业认证：支持团队成员获取和维护行业认证，以验证他们所学并认可他们的成就。

实施计划的工作量级别：高

资源

相关最佳实践：

- OPS03-BP01 提供高管支持
- OPS11-BP04 执行知识管理

相关文档：

- AWS 白皮书 | Cloud Adoption Framework: People Perspective
- Investing in continuous learning to grow your organization's future
- AWS Skills Guild
- AWS 培训与认证
- AWS Support
- AWS re:Post
- AWS 入门资源中心
- AWS Blog
- AWS Cloud合规
- AWS 文档
- The Official AWS Podcast.
- AWS 在线技术讲座
- AWS 活动和网络研讨会

- AWS Well-Architected Labs
- Amazon Builders' Library

相关视频：

- AWS re:Invent 2023 | Reskilling at the speed of cloud: Turning employees into entrepreneurs
- WS re:Invent 2023 | Building a culture of curiosity through gamification

OPS03-BP07 为团队配置适当的资源

配备适当数量的精通业务的团队成员，并提供工具和资源来支持您的工作负载需求。团队成员负担过重会增加人为出错的风险。对自动化技术等工具和资源的投资可以提高团队的效率，有助于他们支持更多的工作负载，而不需要具备额外的能力。

期望结果：

- 您已根据迁移计划为团队配备了适当的人员，以获得在 AWS 中操作工作负载所需的技能组合。在迁移项目过程中，随着团队规模不断扩大，他们已经熟练掌握了企业计划迁移应用程序，或对应用程序进行现代化改造时，所需使用的 AWS 核心技术。
- 您精心调整了人员配备计划，通过利用自动化和工作流程来高效使用资源。哪怕是规模较小的团队，现在也可以代表应用程序开发团队管理更多的基础设施。
- 随着业务优先事项的不断变化，任何资源人员配置方面的限制都会被主动识别出来，以便保护业务计划的成功。
- 对报告负担繁重（例如值班疲劳或过度传呼）的业务指标进行审查，以便核实工作人员是否存在不堪重负的情况。

常见反面模式：

- 在您的多年云迁移计划接近尾声时，您的员工尚未提高 AWS 技能，这可能会影响对工作负载的支持，并降低员工士气。
- 您的整个 IT 组织正在向敏捷工作方式转变。企业正在对产品组合进行优先排序，并设定需要首先开发哪些功能的指标。您的敏捷流程并不要求团队为其工作计划分配故事点。因此，不可能知道下一个工作量所需的能力水平，也不可能知道是否有合适的技能分配给工作。
- 您正在让 AWS 合作伙伴迁移工作负载，但合作伙伴迁移完项目后，您还没有为团队制定好支持过渡计划。您的团队难以高效而有效地支持工作负载。

建立此最佳实践的好处：贵组织中有具备适当技能的团队成员来支持工作负载。资源分配可适应优先事项的变化，而不会影响绩效。其结果是，团队能够熟练地支持工作负载，同时极大限度地利用时间专注于为客户创新，这反过来又提高了员工的满意度。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

云迁移的资源规划应在组织层面进行，与迁移计划以及为支持新云环境而实施的理想运营模式保持一致。这应该包括了解为业务和应用程序开发团队部署了哪些云技术。基础设施和运营领导层应该为领导云技术采用的工程师制定技能差距分析、培训和角色定义方面的计划。

实施步骤

1. 通过相关的运营指标，如员工生产率（例如，支持工作负载的成本或操作员在意外事件期间花费的时间），制定团队成功的成功标准。
2. 制定资源能力规划和检查机制，以便核实在需要时，是否有适当平衡的合格能力，并且这些能力是否可随时间进行调整。
3. 建立机制（例如，每月向团队发送调查问卷），以期了解影响团队的、与工作相关的挑战（如责任增加、技术变化、人员流失或支持的客户增加）。
4. 利用这些机制与团队互动，发现可能导致员工生产率挑战的趋势。团队受外部因素影响时，需重新评估目标并适当地调整执行性目标。确定阻碍团队进度的障碍。
5. 定期检查当前配备的资源是否仍然足够，是否需要额外资源，并做出适当调整以支持团队。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS03-BP06 鼓励团队成员保持和增强自己的技能组合
- OPS09-BP03 审查运营指标并确定改进优先顺序
- OPS10-BP01 使用流程来管理事件、意外事件和问题
- OPS10-BP07 自动响应事件

相关文档：

- AWS Cloud Adoption Framework: People Perspective

- Becoming a Future-Ready Enterprise
- Prioritize your Employees' Skills to Drive Business Growth
- 高绩效组织 - Amazon 两个披萨团队
- How Cloud-Mature Enterprises Succeed

准备

要为卓越运营做好准备，您必须了解您的工作负载及其预期行为。然后，您需要能够针对它们进行设计，以提供对其状态的洞察并构建程序以提供支持。

要为卓越运营做好准备，您需要执行以下操作：

主题

实现可观测性

在工作负载中实现可观测性，以便您可以了解其状态并根据业务需求做出数据驱动型决策。

可观测性不仅仅是简单的监控，它让您可以根据系统的外部输出全面了解系统的内部运作。可观测性源于指标、日志和跟踪数据，可提供对系统行为和动态的深刻见解。通过有效的可观测性，团队可以识别模式、异常和趋势，从而能够主动解决潜在问题并保持最佳系统运行状况。

要想确保监控活动与业务目标协调一致，确定关键绩效指标（KPI）至关重要。这种一致性可确保团队使用真正重要的指标做出数据驱动型决策，从而优化系统性能和业务成果。

此外，可观测性使企业能够积极采取行动，而不是被动做出反应。团队可以了解其系统中的因果关系，以此预测和预防问题，而不仅仅是对问题做出反应。随着工作负载的变化，必须重新审视和完善可观测性策略，确保其仍然适用且有效。

最佳实践

OPS04-BP01 识别关键绩效指标

要在工作负载中实现可观测性，首先要了解其状态并根据业务需求做出数据驱动型决策。确保监控活动与业务目标相一致的最有效方法之一是，定义和监控关键绩效指标（KPI）。

期望的结果： 与业务目标紧密协调的高效可观测性实践，确保监控工作始终为切实的业务成果服务。

常见反模式：

- 未定义 KPI：在没有明确 KPI 的情况下工作可能会导致监控过多或过少内容，从而缺少重要信号。
- 静态 KPI：不会随着工作负载或业务目标的变化而重新审视或完善 KPI。
- 不一致：重点关注与业务成果不直接相关或难以与现实问题关联的技术指标。

建立此最佳实践的好处：

- 易于识别问题：业务 KPI 通常比技术指标能够更清楚地揭示问题。与筛查众多技术指标相比，业务 KPI 的下降有助于更有效地查明问题。
- 业务协调：确保监控活动直接支持业务目标。
- 效率：将监控资源和注意力优先放在重要的指标上。
- 积极主动：在问题对业务产生更广泛影响之前识别并解决问题。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

要有效地定义工作负载 KPI，请执行以下操作：

1. 从业务成果着手： 在深入研究指标之前，请先了解所需的业务成果。是销售额增加、用户参与度提高还是响应时间更短？
2. 将技术指标与业务目标相关联： 并非所有技术指标都会对业务结果产生直接影响。确定那些确实会产生直接影响的指标，但使用业务 KPI 来识别问题通常更为简单。
3. 使用 Amazon CloudWatch： 利用 CloudWatch 定义和监控代表您的 KPI 的指标。
4. 定期审查和更新 KPI： 随着工作负载和业务的发展，保持 KPI 的相关性。
5. 让利益相关方参与进来： 让技术和业务团队参与定义和审查 KPI。

实施计划的工作量级别： 中

资源

相关最佳实践：

- the section called “OPS04-BP02 实施应用程序遥测”
- the section called “OPS04-BP03 实施用户体验遥测”
- the section called “OPS04-BP04 实施依赖项遥测”
- the section called “OPS04-BP05 实施分布式跟踪”

相关文档：

- AWS 可观测性最佳实践
- CloudWatch 用户指南
- AWS 可观测性 Skill Builder 课程

相关视频：

- 开发可观测性战略

相关示例：

- One Observability Workshop

OPS04-BP02 实施应用程序遥测

应用程序遥测是实现工作负载可观测性的基础。发射遥测数据至关重要，它可以提供切实可行的见解，让您了解应用程序的状态以及技术和业务成果的实现情况。从故障排除到衡量新功能的影响或确保与业务关键绩效指标（KPI）保持一致，应用程序遥测可为您构建、操作和演进工作负载的方式提供指导。

指标、日志和跟踪数据构成了可观测性的三个主要支柱。它们用作诊断工具来描述应用程序状态。随着时间的推移，它们会协助创建基线和识别异常情况。但是，为了确保监控活动与业务目标协调一致，定义和监控 KPI 至关重要。与只考虑纯粹的技术指标相比，业务 KPI 通常有助于更轻松地识别问题。

其他遥测类型，例如真实用户监控（RUM）和综合事务，是对这些主要数据源的补充。RUM 让您可以了解实时用户交互，而综合事务则模拟潜在的用户行为，有助于提前发现瓶颈，以防真实用户遇到瓶颈。

期望结果：获得有关工作负载性能的可操作见解。这些见解使您能够主动作出性能优化决策，提高工作负载稳定性，简化 CI/CD 流程，并有效地利用资源。

常见反面模式：

- 可观测性不完整：忽略将可观测性纳入工作负载的每一层，造成盲点，从而掩盖重要的系统性能和行为洞察。
- 支离破碎的数据视图：当数据分散在多个工具和系统中时，要全面了解工作负载的运行状况和性能，会非常困难。
- 用户报告的问题：这表明缺乏通过遥测和业务 KPI 监控来主动发现问题的功能。

建立此最佳实践的好处：

- 明智的决策：借助从遥测和业务 KPI 中获得的见解，您可以作出以数据为导向的决策。
- 提高运营效率：以数据为驱动来利用资源，可提高成本效益。
- 增强工作负载稳定性：更快地检测和解决问题，延长正常运行时间。
- 简化 CI/CD 流程：从遥测数据获得的见解有助于完善流程和可靠地交付代码。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要为您的工作负载实现应用程序遥测，请使用 AWS 服务，例如 Amazon CloudWatch 和 AWS X-Ray。Amazon CloudWatch 提供了一套全面的监控工具，让您能够观察 AWS 和本地环境中的资源和应用程序。该服务会收集、跟踪和分析指标，整合和监控日志数据，并对资源的变化做出响应，从而增进您对工作负载运行方式的了解。同时，利用 AWS X-Ray，您还可以跟踪、分析和调试应用程序，从而深入了解工作负载的行为。借助服务地图、延迟分布和跟踪时间表等功能，AWS X-Ray 可让您深入了解工作负载的性能和影响工作负载性能的瓶颈。

实施步骤

1. 确定要收集哪些数据：确定有助于您深入了解工作负载的运行状况、性能和行为的基本指标、日志和跟踪数据。
2. 部署 CloudWatch 代理：CloudWatch 代理在从您的工作负载及其底层基础设施中获取系统和应用程序指标和日志方面发挥着重要作用。该 CloudWatch 代理还可用于收集 OpenTelemetry 或 X-Ray 跟踪数据，并将其发送到 X-Ray。

3. 对日志和指标实施异常检测：使用 CloudWatch Logs 异常检测和 CloudWatch 指标异常检测自动识别应用程序操作中的异常活动。这些工具使用机器学习算法来检测异常情况并发出警报，从而增强了监控能力，加快了对潜在中断或安全威胁的响应速度。设置这些功能可主动管理应用程序的运行状况和安全性。
4. 保护敏感日志数据：使用 Amazon CloudWatch Logs 数据保护来掩蔽日志中的敏感信息。此功能会在访问敏感数据之前自动检测和掩蔽敏感数据，有助于维护隐私和合规性。实施数据掩蔽，以期安全地处理和保护敏感详细信息，如个人身份信息（PII）。
5. 定义和监控业务 KPI：建立与业务成果相一致的自定义指标。
6. 使用 AWS X-Ray 检测应用程序：除了部署 CloudWatch 代理外，还必须检测应用程序，以便发出跟踪数据。此过程可让您进一步了解工作负载的行为和性能。
7. 在整个应用程序中实现数据收集标准化：在整个应用程序中实现数据收集实践的标准化。统一性有助于关联和分析数据，从而全面了解应用程序的行为。
8. 实现跨账户可观测性：利用 Amazon CloudWatch 跨账户可观测性提高跨多个 AWS 账户的监控效率。利用该功能，您可以将不同账户中的指标、日志和警报整合到一个视图中，从而简化管理，并提高对整个组织的 AWS 环境中已发现问题的响应速度。
9. 分析数据并据此采取行动：数据收集和规范化完成后，使用 Amazon CloudWatch 进行指标和日志分析，使用 AWS X-Ray 进行跟踪分析。此类分析可得出有关您的工作负载的运行状况、性能和行为的重要见解，从而指导您的决策过程。

实施计划的工作量级别：高

资源

相关最佳实践：

- OPS04-BP01 定义工作负载 KPI
- OPS04-BP03 实施用户活动遥测
- OPS04-BP04 实施依赖项遥测
- OPS04-BP05 实施事务可追溯性

相关文档：

- AWS Observability Best Practices
- CloudWatch User Guide
- AWS X-Ray Developer Guide

- 检测分布式系统的运营可见性
- AWS Observability Skill Builder Course
- Amazon CloudWatch 最新资讯
- AWS X-Ray 最新资讯

相关视频：

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2022 - Developing an observability strategy

相关示例：

- One Observability Workshop
- AWS 解决方案库：使用 Amazon CloudWatch 进行应用程序监控

OPS04-BP03 实施用户体验遥测

深入了解客户体验以及与应用程序的交互至关重要。真实用户监控（RUM）和综合事务是实现此目的的强大工具。RUM 提供有关真实用户交互的数据，从未经过滤的视角反映用户满意度，而综合事务可模拟用户交互，有助于在潜在问题影响真实用户之前就发现它们。

期望的结果： 全面了解客户体验，主动检测问题，优化用户互动，以提供无缝的数字体验。

常见反模式：

- 应用程序没有真实用户监控（RUM）功能
  - 问题检测被延误：如果没有 RUM，可能要等到用户抱怨时，您才会意识到性能瓶颈或问题。这种被动应对的方法可能会导致客户不满。
  - 缺乏对用户体验的了解：不使用 RUM 意味着您无法掌握揭示真实用户如何与应用程序交互的关键数据，从而限制您优化用户体验的能力。
- 应用程序缺乏综合事务
  - 错过边缘案例：综合事务有助于您测试普通用户可能不经常使用、但对某些业务职能至关重要的路径和功能。没有它们，这些路径可能会出现故障并被忽视。
  - 在应用程序未使用时检查问题：定期的综合测试可以模拟真实用户未积极与应用程序交互时的情况，确保系统始终正常运行。

建立此最佳实践的好处：

- 主动检测问题：在潜在问题影响真实用户之前，识别并解决这些问题。
- 优化用户体验：来自 RUM 的持续反馈有助于完善和增强整体用户体验。
- 获得有关设备和浏览器性能的见解：了解您的应用程序在各种设备和浏览器上的表现，从而实现进一步优化。
- 经过验证的业务工作流程：定期的综合事务可确保核心功能和关键路径始终可以使用且高效。
- 增强应用程序性能：利用从真实用户数据中收集的见解，提高应用程序的响应能力和可靠性。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

为了利用 RUM 和综合事务进行用户活动遥测，AWS 提供多项服务，例如 Amazon CloudWatch RUM 和 Amazon CloudWatch Synthetics。指标、日志和跟踪，再加上用户活动数据，可让您全面了解应用程序的运行状态和用户体验。

实施步骤

1. 部署 Amazon CloudWatch RUM： 将您的应用程序与 CloudWatch RUM 集成，收集、分析和呈现真实的用户数据。
   a. 使用 CloudWatch RUM JavaScript 库 将 RUM 与您的应用程序集成。
   b. 设置控制面板以可视化形式呈现和监控真实的用户数据。
2. 配置 CloudWatch Synthetics： 创建金丝雀或脚本化例程，模拟用户与应用程序的交互。
   a. 定义关键应用程序工作流程和路径。
   b. 使用 CloudWatch Synthetics 脚本 设计金丝雀，模拟用户在这些路径上的交互。
   c. 安排和监控金丝雀按指定的间隔运行，确保一致的性能检查。
3. 分析数据并据此采取行动： 利用来自 RUM 和综合事务的数据来获取见解，并在检测到异常时采取纠正措施。使用 CloudWatch 控制面板和警报及时了解情况。

实施计划的工作量级别： 中

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标
- OPS04-BP02 实施应用程序遥测
- OPS04-BP04 实施依赖项遥测
- OPS04-BP05 实施分布式跟踪

相关文档：

- Amazon CloudWatch RUM 指南
- Amazon CloudWatch Synthetics 指南

相关视频：

- 使用 Amazon CloudWatch RUM 通过最终用户洞察优化应用程序
- AWS on Air ft.Real-User Monitoring for Amazon CloudWatch

相关示例：

- 可观测性研讨会
- 适用于 Amazon CloudWatch RUM Web 客户端的 Git 存储库
- 使用 Amazon CloudWatch Synthetics 来测量页面加载时间

OPS04-BP04 实施依赖项遥测

要想监控您的工作负载所依赖的外部服务和组件的运行状况及性能，依赖项遥测必不可少。依赖项遥测提供有关与 DNS、数据库或第三方 API 等依赖项相关的可访问性、超时及其他关键事件的宝贵见解。当您对应用程序进行检测，以发布有关这些依赖项的指标、日志和跟踪时，您就能更清楚地了解可能影响工作负载的潜在瓶颈、性能问题或故障。

期望结果：确保您的工作负载所依赖的依赖项按预期运行，让您能够主动解决问题，确保实现极佳的工作负载性能。

常见反面模式：

- 忽略外部依赖项：仅关注内部应用程序指标，而忽略与外部依赖项相关的指标。
- 缺乏主动监控：等待问题出现，而不是持续监控依赖项运行状况和性能。

- 孤立监控：使用多种不同的监控工具，这可能会导致依赖项运行状况视图支离破碎且不一致。

建立此最佳实践的好处：

- 提高工作负载可靠性：通过确保外部依赖项始终可用且性能出色来实现。
- 更快地检测和解决问题：在依赖项问题影响工作负载之前，主动识别和解决这些问题。
- 全面视图：全面了解影响工作负载运行状况的内部和外部组件。
- 增强工作负载可扩展性：通过了解外部依赖项的可扩展性限制和性能特征来实现。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

从确定您的工作负载所依赖的服务、基础设施和流程开始，实施依赖项遥测。量化这些依赖项按预期运行时的良好状况，然后确定将需要哪些数据来衡量这些状况。利用这些信息，您可以创建控制面板和警报，为运营团队提供有关这些依赖项状态的见解。当依赖项无法按需交付时，使用 AWS 工具来发现和量化影响。不断重新审视您的策略，以考虑优先事项、目标和所获见解的变化。

实施步骤

要有效地实现依赖项遥测，请执行以下操作：

1. 确定外部依赖项：与利益相关方合作，查明您的工作负载所依赖的外部依赖项。外部依赖项可包括外部数据库、第三方 API、通往其他环境的网络连接路由以及 DNS 服务等内容。实现有效的依赖项遥测的第一步是全面了解这些依赖项是什么。
2. 制定监控策略：一旦您清楚地了解了外部依赖项，就可以为其量身定制监控策略。这包括了解每个依赖项的重要程度、其预期行为以及任何相关的服务级别协议或目标（SLA 或 SLT）。设置主动警报，在出现状态变化或性能偏差时通知您。
3. 使用网络监控：使用网络检测仪和网络监视器，全面了解全球互联网和网络状况。这些工具有助于您了解并应对影响外部依赖项的中断、破坏或性能下降。
4. 使用 AWS Health Dashboard 随时了解情况：当 AWS 发生可能影响服务的事件时，该控制面板会发出警报并提供修正指导。
    a. 使用 Amazon EventBridge 规则监控 AWS Health 事件，或者以编程方式与 AWS Health API 集成，以便在收到 AWS Health 事件时自动执行操作。这些可以是常规操作，例如将所有计划的生命周期事件消息发送到聊天界面，也可以是特定操作，例如在 IT 服务管理工具中启动工作流。
    b. 如果您使用 AWS Organizations，则跨账户汇总 AWS Health 事件。

5. 使用 AWS X-Ray 检测应用程序：AWS X-Ray 让您能够深入了解应用程序及其底层依赖项的运行情况。通过从头到尾跟踪请求，您可以找出应用程序所依赖的外部服务或组件中的瓶颈或故障。
6. 使用 Amazon DevOps Guru：这项服务由机器学习驱动，可识别操作问题，预测何时可能出现严重问题，并建议可采取的具体行动。其可贵之处在于，可以让您深入了解依赖项，并确保这些依赖项不会成为操作问题的根源。
7. 定期监控：持续监控与外部依赖项相关的指标和日志。针对意外行为或性能下降设置警报。
8. 更改后验证：每当任何外部依赖项有更新或更改时，都应验证其性能，并检查这些依赖项是否符合应用程序的要求。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS04-BP01 定义工作负载 KPI
- OPS04-BP02 实施应用程序遥测
- OPS04-BP03 实施用户活动遥测
- OPS04-BP05 实施事务可追溯性
- OP08-BP04 创建可操作的警报

相关文档：

- Amazon Personal AWS Health Dashboard User Guide
- AWS Internet Monitor User Guide
- AWS X-Ray Developer Guide
- AWS DevOps Guru User Guide

相关视频：

- Visibility into how internet issues impact app performance
- Introduction to Amazon DevOps Guru
- Manage resource lifecycle events at scale with AWS Health

相关示例：

- Gaining operational insights with AIOps using Amazon DevOps Guru
- AWS Health Aware
- Using Tag-Based Filtering to Manage AWS Health Monitoring and Alerting at Scale

OPS04-BP05 实施分布式跟踪

分布式跟踪提供了一种方法，可在请求通过分布式系统的各个组件时对其进行监控和可视化。通过从多个来源捕获跟踪数据并在一个统一视图中对其进行分析，团队可以更好地了解请求是如何流动的、哪里存在瓶颈以及优化工作的重点。

期望的结果： 全面了解流经分布式系统的请求，从而精确调试、优化性能和改善用户体验。

常见反模式：

- 检测不一致：并非分布式系统中的所有服务都经过跟踪检测。
- 忽略延迟：只关注错误，而不考虑延迟或性能逐渐下降的情况。

建立此最佳实践的好处：

- 全面了解系统：以可视化方式呈现请求从进入到退出的整个路径。
- 增强调试功能：快速识别出现故障或性能问题的地方。
- 改善用户体验：监控并根据实际用户数据进行优化，确保系统满足现实需求。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

首先确定工作负载中所有需要检测的元素。将所有组件都考虑在内后，利用 AWS X-Ray 和 OpenTelemetry 之类的工具收集跟踪数据，以便使用 X-Ray 和 Amazon CloudWatch ServiceLens Map 等工具进行分析。定期与开发人员一起进行审查，并使用 Amazon DevOps Guru、X-Ray Analytics 和 X-Ray Insights 等工具来补充这些讨论，以挖掘更深层次的信息。根据跟踪数据确立警报，以便在结果面临风险时，按照工作负载监控计划中定义的流程发出通知。

实施步骤

要有效地实施分布式跟踪，请执行以下操作：

1. 采用 AWS X-Ray： 将 X-Ray 集成到您的应用程序中，以深入了解其行为和性能并查明瓶颈。利用 X-Ray Insights 自动分析跟踪数据。
2. 检测您的服务： 确认从 AWS Lambda 函数到 EC2 实例的每项服务都发送跟踪数据。您检测的服务越多，端到端视图就越清晰。
3. 纳入 CloudWatch 真实用户监控 和 合成监控： 将真实用户监控（RUM）和合成监控与 X-Ray 集成。这允许捕捉现实世界的用户体验并模拟用户交互，以识别潜在问题。
4. 使用 CloudWatch 代理： 代理可以从 X-Ray 或 OpenTelemetry 发送跟踪，从而增强所获得见解的深度。
5. 使用 Amazon DevOps Guru： DevOps Guru 使用来自 X-Ray、CloudWatch、AWS Config 和 AWS CloudTrail 的数据来提供可行的建议。
6. 分析跟踪数据： 定期查看跟踪数据，以识别可能影响应用程序性能的模式、异常或瓶颈。
7. 设置警报： 在 CloudWatch 中针对异常模式或过长的延迟时间配置警报，以便于主动解决问题。
8. 持续改进： 在添加或修改服务时，重新审视您的跟踪策略，以捕获所有相关数据点。

实施计划的工作量级别： 中

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标
- OPS04-BP02 实施应用程序遥测
- OPS04-BP03 实施用户体验遥测
- OPS04-BP04 实施依赖项遥测

相关文档：

- AWS X-Ray 开发人员指南
- Amazon CloudWatch 代理用户指南
- Amazon DevOps Guru 用户指南

相关视频：

- Use AWS X-Ray Insights

- AWS on Air ft.Observability: Amazon CloudWatch and AWS X-Ray

相关示例：

- Instrumenting your Application with AWS X-Ray

运营设计

采用可改进生产调整流程并协助重构、快速质量反馈和错误修复的方法。这些方法可以加快有益更改进入生产环境的速度、减少产生的问题，并能够快速识别和修复通过部署活动引入的问题。

在 AWS 中，您可以将整个工作负载（应用程序、基础设施、策略、监管和运营）视为代码。这些全部可以使用代码来定义和更新。这意味着您可以将用于应用程序代码的工程规范应用于堆栈的每个元素。

最佳实践

- OPS05-BP01 使用版本控制
- OPS05-BP02 测试并验证变更
- OPS05-BP03 使用配置管理系统
- OPS05-BP04 使用构建和部署管理系统
- OPS05-BP05 执行补丁管理
- OPS05-BP06 共享设计标准
- OPS05-BP07 实施提高代码质量的实践
- OPS05-BP08 使用多个环境
- OPS05-BP09 频繁进行可逆的小规模更改
- OPS05-BP10 完全自动化集成和部署

OPS05-BP01 使用版本控制

使用版本控制来跟踪更改和发布。

许多 AWS 服务都提供版本控制功能。使用修订或源代码控制系统（如 AWS CodeCommit ）管理代码和其他构件，如基础设施的版本控制的 AWS CloudFormation 模板。

期望的结果： 您的团队就代码开展协作。合并后，代码将保持一致，并且不会丢失任何更改。通过正确的版本控制，可以很容易纠正错误。

常见反模式：

- 您一直在工作站上开发和存储代码。工作站上发生了不可恢复的存储故障，您的代码丢失了。
- 用更改内容覆盖现有代码后，您重新启动应用程序，但其无法运行。您无法撤消所做更改。
- 您对报告文件执行了写入锁定，而其他人需要对此文件进行编辑。他们与您联系要求您停止写入锁定，以便他们可以完成自己的任务。
- 您的研究团队一直在进行详细的分析，以便对未来的工作进行规划。有人不小心把购物单保存在最终报告上了。您无法撤消更改，不得不重新创建报告。

建立此最佳实践的好处： 借助版本控制功能，您可以轻松地恢复到已知的良好状态和以前的版本，并降低资产丢失的风险。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

在受到版本控制的存储库中维护资产。这让您能够跟踪更改、部署新版本、检测对现有版本的更改，以及恢复到以前的版本（例如在发生故障时回滚到已知的良好状态）。将配置管理系统的版本控制功能集成到程序中。

资源

相关最佳实践：

- OPS05-BP04 使用构建和部署管理系统

相关文档：

- 什么是 AWS CodeCommit？

相关视频：

- AWS CodeCommit 简介

OPS05-BP02 测试并验证变更

部署的每一项变更都必须经过测试，以避免在生产中出现错误。此最佳实践的重点是测试从版本控制到构件构建的变更。除应用程序代码变更外，测试还应该包括基础设施、配置、安全控制和操作程序。测

试有多种形式，从单元测试到软件组件分析（SCA）等等。在软件集成和交付过程中，尽早进行测试可进一步确保构件质量。

您的组织必须为所有的软件构件制定测试标准。自动化测试可以减少工作量，并避免人工测试的错误。有些情况下，可能必须进行手动测试。开发人员必须能够访问自动化测试结果，以创建反馈循环，提高软件质量。

期望结果：软件更改在交付前进行测试。开发人员可以访问测试结果和验证结果。您的组织有一个适用于所有软件更改的测试标准。

常见反面模式：

- 您在没有进行任何测试的情况下部署一项新软件更改。它无法在生产环境中运行，从而导致中断。
- 使用 AWS CloudFormation 部署新安全组，而没有在生产前环境中进行测试。这些安全组使客户无法访问您的应用程序。
- 修改了一个方法，但没有进行单元测试。该软件在部署到生产环境中后无法运行。

建立此最佳实践的好处：降低了软件部署的变更失败率。软件质量得到改进。开发人员提高了对其代码可行性的认识。可以放心地推出安全策略，以支持组织实现合规性。可以提前测试基础设施更改（如自动扩缩策略的更新），以满足流量需求。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

作为持续集成实践的一部分，对从应用程序代码到基础设施的所有更改都进行测试。将公布测试结果，以便开发人员快速提供反馈。您的组织有一个测试标准，所有更改都必须通过该标准。

利用 Amazon Q Developer 生成式人工智能的强大功能，提高开发人员的工作效率和代码质量。Amazon Q Developer 包括生成代码建议（基于大型语言模型）、制作单元测试（包括边界条件），以及通过检测和修复安全漏洞来增强代码安全性。

客户示例

作为持续集成管道的一部分，AnyCompany Retail 对所有软件构件进行几种类型的测试。他们实行测试驱动型开发，因此所有软件都有单元测试。构件构建完毕后，他们会立即运行端到端测试。第一轮测试完成后，他们会运行静态应用程序安全扫描，寻找已知漏洞。在每个测试关口通过时，开发人员都会收到消息。所有测试均完成后，软件构件就会存储在构件库中。

实施步骤

1. 与您组织中的利益相关方合作，为软件构件制定测试标准。所有构件均应通过哪些标准测试？ 是否有合规性或治理要求必须包括在测试范围内？ 您是否需要进行代码质量测试？ 测试完成后，需要通知谁？
   1. AWS 部署管道参考架构包含一个权威的测试类型列表，可作为集成管道的一部分对软件构件执行这些测试。
2. 根据您的软件测试标准，利用必要的测试来检测您的应用程序。每组测试应在 10 分钟内完成。测试应该作为集成管道的一部分运行。
   a. Amazon Q Developer 是一款生成式人工智能工具，有助于创建单元测试用例（包括边界条件），使用代码和注释生成函数，并实现众所周知的算法。
   b. 使用 Amazon CodeGuru Reviewer 测试应用程序代码是否存在缺陷。
   c. 可以使用 AWS CodeBuild 对软件构件执行测试。
   d. AWS CodePipeline 可以将您的软件测试编排到管道中。

资源

相关最佳实践：

- OPS05-BP01 使用版本控制
- OPS05-BP06 共享设计标准
- OPS05-BP07 实施提高代码质量的实践
- OPS05-BP10 完全自动化集成和部署

相关文档：

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer

- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Automated AWS CloudFormation Testing Pipeline with TaskCat and CodePipeline
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST, and DAST tools
- Getting started with testing serverless applications
- My CI/CD pipeline is my release captain
- Practicing Continuous Integration and Continuous Delivery on AWS Whitepaper

相关视频：

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Testable infrastructure: Integration testing on AWS
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development
- Testing Your Infrastructure as Code with AWS CDK

相关资源：

- Building applications using generative AI with Amazon CodeWhisperer
- Amazon CodeWhisperer Workshop
- AWS Deployment Pipeline Reference Architecture - Application
- AWS Kubernetes DevSecOps Pipeline
- Policy as Code Workshop – Test Driven Development
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild
- Use Serverspec for test-driven development of infrastructure code

相关服务：

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- AWS CodeBuild

- AWS CodePipeline

OPS05-BP03 使用配置管理系统

使用配置管理系统来实现和跟踪配置更改。这些系统可以减少手动过程引起的错误，并减少部署更改的工作量。

静态配置管理在初始化资源时设置值，这些值在资源的生命周期内预期会保持一致。这样的例子包括为实例上的 Web 或应用程序服务器设置配置，或者定义 AWS 服务的配置（在 AWS Management Console 内或者通过 AWS CLI）。

动态配置管理在初始化时设置值，这些值在资源的生命周期内可能或预期会发生变化。例如，您可以设置一个功能切换，通过配置更改在代码中激活功能，或者在意外事件期间更改日志详细级别以捕获更多数据，然后在意外事件完成后更改回来，避免不再必要的日志记录及其相关费用。

在 AWS 上，您可以使用 AWS Config 持续监控 AWS 资源配置 - 跨账户和区域。这有助于您跟踪其配置历史记录，了解配置更改会如何影响其他资源，并使用 AWS Config 规则 和 AWS Config 合规包根据预期或所需的配置审计它们。

如果您在 Amazon EC2 实例、AWS Lambda、容器、移动应用程序或物联网设备上运行的应用程序具有动态配置，则可以使用 AWS AppConfig 在您的环境中配置、验证、部署和监控它们。

在 AWS 中，您可以使用像 AWS 开发人员工具（例如，AWS CodeCommit、 AWS CodeBuild、AWS CodePipeline、 AWS CodeDeploy和 AWS CodeStar）这样的服务来构建持续集成/持续部署（CI/CD）管道。

期望的结果： 可以作为持续集成、持续交付（CI/CD）管道的一部分进行配置、验证和部署。通过监控来验证配置是否正确。这样可以最大限度地减少对最终用户和客户的任何影响。

常见反模式：

- 您手动更新整个队列中的 Web 服务器配置，由于更新错误，许多服务器变得没有响应。
- 手动更新应用程序服务器队列需要花费很长时间。在变更过程中，如果配置不一致会导致意外行为发生。
- 有人更新了您的安全组，您的 Web 服务器无法访问了。如果不知道发生了哪些变更，您需要花费大量时间来调查问题，导致恢复时间延长。
- 未经验证即通过 CI/CD 将预生产配置推送到生产环境中。您让用户和客户接触到不正确的数据和服务。

建立此最佳实践的好处： 采用配置管理系统可以减少更改及对其进行跟踪的工作量，还可以降低手动程序导致错误的频率。配置管理系统为治理、合规性和监管要求提供了保障。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

配置管理系统用于跟踪和实施对应用程序和环境配置的更改。配置管理系统还用于减少手动流程引起的错误，使配置更改可重复且可审核，并减少工作量。

实施步骤

1. 确定配置负责人。
   a. 让配置负责人了解任何合规性、治理或监管需求。
2. 确定配置项和可交付成果。
   a. 配置项是受您的 CI/CD 管道内的部署影响的所有应用程序和环境配置。
   b. 可交付成果包括成功标准、验证和要监控的内容。
3. 根据您的业务需求和交付管道，选择配置管理工具。
4. 考虑使用加权部署，例如用于重大配置更改的金丝雀部署，以尽可能减少错误配置的影响。
5. 将您的配置管理集成到 CI/CD 管道中。
6. 验证所有推送的更改。

资源

相关最佳实践：

- OPS06-BP01 针对不成功的更改制定计划
- OPS06-BP02 测试部署
- OPS06-BP03 采用安全部署策略
- OPS06-BP04 自动测试和回滚

相关文档：

- AWS Control Tower
- AWS 登录区加速器
- AWS Config

- 什么是 AWS Config？
- AWS AppConfig
- 什么是 AWS CloudFormation？
- AWS 开发人员工具

相关视频：

- AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads
- AWS re:Invent 2020：使用 AWS Config 实现合规性即代码
- Manage and Deploy Application Configurations with AWS AppConfig

OPS05-BP04 使用构建和部署管理系统

使用构建和部署管理系统。这些系统可以减少手动过程引起的错误，并减少部署更改的工作量。

在 AWS 中，您可以使用像 AWS 开发人员工具（例如，AWS CodeCommit、 AWS CodeBuild、AWS CodePipeline、 AWS CodeDeploy和 AWS CodeStar）这样的服务来构建持续集成/持续部署（CI/CD）管道。

期望的结果： 您的构建和部署管理系统支持组织的持续集成/持续交付（CI/CD）系统，后者提供使用正确的配置自动进行安全部署的功能。

常见反模式：

- 在开发系统上编译代码后，您将可执行文件复制到生产系统上，但它无法启动。本地日志文件显示这是因为缺少依赖项。
- 您成功地在开发环境中构建了具有新功能的应用程序，并将代码送交质量检查（QA）。由于缺少静态资产，它没有通过质量检查。
- 星期五，经过大量的努力，您成功地在开发环境中手动构建了应用程序，包括新编码的功能。星期一，您无法重复这一成功构建应用程序的步骤。
- 您执行为新版本创建的测试。下周，您将设置测试环境，并执行所有现有的集成测试，然后执行性能测试。新代码产生了难以接受的性能影响，因此必须重新开发并测试。

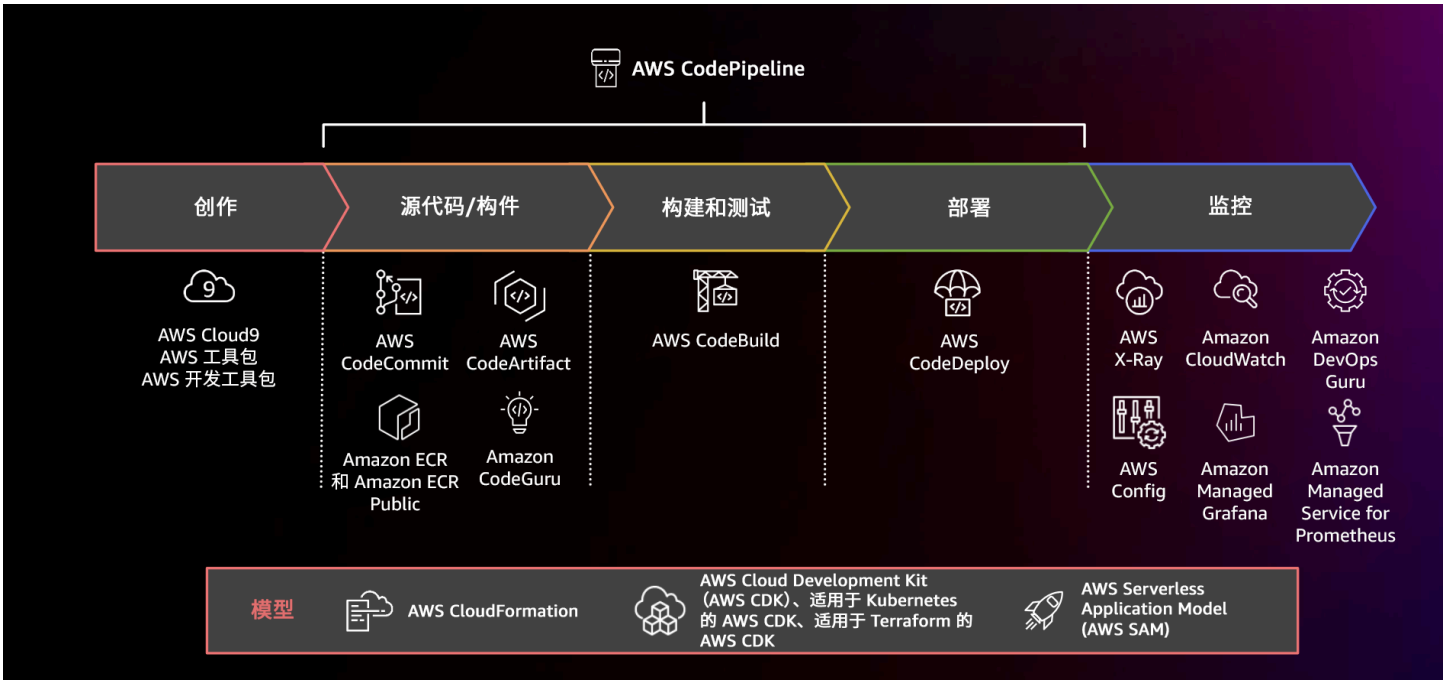
建立此最佳实践的好处： 制定相应机制来管理活动的构建和部署。这样，您可以减少执行重复任务的工作量，让团队成员腾出时间专注于高价值的创造性任务，还可以减少手动程序导致的错误。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

构建和部署管理系统用于跟踪和实施变更，减少手动过程引起的错误，并减少安全部署所需的工作量。将集成和部署管道完全自动化，从代码签入到构建、测试、部署和验证都包含在内。这可以缩短准备时间，降低成本，鼓励更频繁地进行变更，减少工作量并增进协作。

实施步骤



显示使用 AWS CodePipeline 和相关服务的 CI/CD 管道的示意图

1. 使用 AWS CodeCommit 对资产（例如文档、源代码和二进制文件）进行版本控制、存储和管理。
2. 使用 CodeBuild 编译源代码、运行单元测试和生成可随时部署的构件。
3. 使用 CodeDeploy 作为部署服务，自动将应用程序部署到 Amazon EC2 实例、本地实例、无服务器 AWS Lambda 函数或 Amazon ECS。
4. 监控您的部署。

资源

相关最佳实践：

- OPS06-BP04 自动测试和回滚

相关文档：

- AWS 开发人员工具
- 什么是 AWS CodeCommit？
- 什么是 AWS CodeBuild？
- AWS CodeBuild
- 什么是 AWS CodeDeploy？

相关视频：

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

OPS05-BP05 执行补丁管理

执行补丁管理以便实现功能、解决问题并保持监管合规性。实现自动补丁管理，以便减少手动过程引起的错误，进行扩展，并减少修补工作量。

补丁和漏洞管理是优势和风险管理活动的一部分。最好是具有不可变的基础设施和在已验证的已知良好状态下部署工作负载。如果该方法不可行，那就只能进行修补。

Amazon EC2 Image Builder 提供更新计算机映像的管道。作为补丁管理的一部分，请考虑 亚马逊机器映像（AMI）（使用 AMI 映像管道 ）或带 Docker 映像管道的容器镜像，同时 AWS Lambda 会提供自定义运行时模式和其他库 以消除漏洞。

您应使用以下工具来管理适用于 Linux 或 Windows Server 映像的 亚马逊机器映像 的更新： Amazon EC2 Image Builder。您可以将 Amazon Elastic Container Registry (Amazon ECR) 与现有管道配合使用，以管理 Amazon ECS 映像和 Amazon EKS 映像。Lambda 包括 版本管理功能。

在未事先在安全环境中测试的情况下，不应对生产系统执行修补操作。仅当补丁支持操作或业务结果时，才应该应用补丁。在 AWS 上，您可以使用 AWS Systems Manager Patch Manager 来自动执行修补托管系统的过程和安排修补活动 - 同时使用 Systems Manager 维护时段。

期望的结果： 您的 AMI 和容器映像已修补、处于最新状态，随时可以启动。您可以跟踪所有已部署映像的状态，并了解补丁合规性。您可以报告当前状态，并有一个流程来满足您的合规需求。

常见反模式：

- 您接到任务，需要在两个小时内应用所有新的安全补丁，但由于应用程序与补丁不兼容，导致了多次停机。

- 没有安装补丁的库会引发意外后果，这是因为未知方会利用其中的漏洞来访问您的工作负载。
- 您在未通知开发人员的情况下自动修补开发人员环境。您收到来自开发人员的多起投诉，称他们的环境不能按预期运行。
- 您尚未修补持久性实例上的现有商用软件。当您遇到软件问题并与供应商联系时，他们告知您已不再为该版本提供支持，您必须安装特定级别的补丁才能获得帮助。
- 您使用的加密软件最近发布了新补丁，对性能进行了重大改进。您未安装补丁的系统仍然存在性能问题，恰恰是因为没有安装补丁造成的。
- 您收到通知，告知您存在零日漏洞，需要紧急修复，而您不得不手动为所有环境打补丁。

建立此最佳实践的好处： 通过建立补丁管理流程，包括修补标准以及在环境中分发补丁的方法，您可以扩展和报告补丁级别。这为安全补丁提供了保障，并确保清楚地了解已知修复程序的状态。这会促进采用所需特性和功能、快速解决问题并保持监管合规性。实施补丁管理系统和自动化，以减少部署补丁的工作量，并减少手动过程引起的错误。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

修补系统以便纠正问题、获得所需的特性或功能、符合监管政策并满足供应商支持需求。在不可变系统中，使用适当的补丁集进行部署，以便实现所需结果。自动执行补丁管理机制以便缩短修补时间、避免手动过程引起的错误，并减少修补工作量。

实施步骤

对于 Amazon EC2 Image Builder：

1. 使用 Amazon EC2 Image Builder，指定管道详细信息：
    a. 创建映像管道并为其命名
    b. 定义管道计划和时区
    c. 配置任何依赖项
2. 选择方案：
    a. 选择现有方案或创建新方案
    b. 选择映像类型
    c. 为您的方案命名并确定其版本
    d. 选择您的基础映像
    e. 添加构建组件并添加到目标注册表

3. 可选 - 定义您的基础设施配置。
4. 可选 - 定义配置设置。
5. 查看设置。
6. 定期检查方案，保持方案正常发挥作用。

对于 Systems Manager Patch Manager：

1. 创建补丁基准。
2. 选择路径操作方法。
3. 启用合规性报告和扫描。

资源

相关最佳实践：

- OPS06-BP04 自动测试和回滚

相关文档：

- What is Amazon EC2 Image Builder?
- Create an image pipeline using the Amazon EC2 Image Builder
- Create a container image pipeline
- AWS Systems Manager Patch Manager
- Working with Patch Manager
- 使用补丁合规性报告
- AWS 开发人员工具

相关视频：

- AWS 上面向无服务器应用程序的 CI/CD
- Ops 设计理念

相关示例：

- Well-Architected 实验室 - 清单和补丁管理

- AWS Systems Manager Patch Manager 教程

OPS05-BP06 共享设计标准

在不同团队间共享最佳实践，以便提高认识并最大程度地实现开发工作的效益。随着架构的发展，记录它们并使它们保持最新。如果在组织中强制实施了共享标准，则必须存在相应的机制来请求对标准进行添加、更改和例外处理。如果没有这样的机制，标准将成为创新的约束。

期望的结果： 在组织中的不同团队间共享设计标准。随着最佳实践的发展，记录标准并使它们保持最新。

常见反模式：

- 两个开发团队各自创建了一个用户身份验证服务。对于用户来说，他们想要访问系统的每一部分，都必须使用一套单独的凭据。
- 每个团队管理他们自己的基础设施。新的合规性要求迫使您变更基础设施，各个团队以不同的方式实施变更。

建立此最佳实践的好处： 使用共享标准支持最佳实践的采用，并充分发挥开发工作的作用。记录和更新设计标准，让您的组织可以了解最新的最佳实践以及安全和合规性要求。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

在不同团队间共享现有的最佳实践、设计标准、检查清单、操作程序、指南和监管要求。建立针对设计标准的更改、添加和例外请求程序，以便支持改进和创新。让团队了解已发布的内容。随着新最佳实践的出现，使用一种机制以使设计标准保持最新。

客户示例

AnyCompany Retail 拥有负责创建软件架构模式的跨职能架构团队。此团队构建具有内置合规性和监管的架构。采用这些共享标准的团队可以从内置合规性和监管中受益。他们可以在设计标准的基础上快速构建。架构团队每季度召开一次会议，评估架构模式，如有必要，更新架构模式。

实施步骤

1. 确定一个跨职能团队，负责开发和更新设计标准。此团队应与整个组织的利益相关方合作，制定设计标准、操作程序、检查清单、指南和监管要求。记录设计标准并在组织内共享。

   a. AWS Service Catalog 可用于使用基础设施即代码创建代表设计标准的产品组合。您可以在不同账户间共享产品组合。

2. 随着新最佳实践的确定，使用一种机制以使设计标准保持最新。
3. 如果集中执行设计标准，则制定一个流程来请求更改、更新和豁免。

实施计划的工作量级别： 中。制定一个流程来创建和共享设计标准，就可以与整个组织的利益攸关方进行协调与合作。

资源

相关最佳实践：

- OPS01-BP03 评估治理要求 - 监管要求会影响设计标准。
- OPS01-BP04 评估合规性要求 - 在创建设计标准时，合规性是一项至关重要的输入。
- OPS07-BP02 确保以一致的方式对运维准备情况进行审查 - 运营准备检查清单是一种在设计工作负载时实施设计标准的机制。
- OPS11-BP01 设置持续改进流程 - 更新设计标准是持续改进的一部分。
- OPS11-BP04 执行知识管理 - 在知识管理实践过程中，记录和共享设计标准。

相关文档：

- 使用 AWS Service Catalog 实现 AWS Backup 自动化
- AWS Service Catalog Account Factory 增强版
- Expedia Group 如何使用 AWS Service Catalog 构建数据库即服务（DBaaS）产品
- 对云架构模式的使用保持可见性
- 简化在 AWS Organizations 设置中共享 AWS Service Catalog 产品组合的操作

相关视频：

- AWS Service Catalog – 入门
- AWS re:Invent 2020：像专家一样管理您的 AWS Service Catalog 产品组合

相关示例：

- AWS Service Catalog 参考架构
- AWS Service Catalog 研讨会

相关服务：

- AWS Service Catalog

OPS05-BP07 实施提高代码质量的实践

实施能够提高代码质量并尽可能减少缺陷的最佳实践。一些示例包括测试驱动型开发、代码审查、标准采用和结对编程。将这些实践合并到您的持续集成和交付流程中。

期望结果：您的组织使用代码审查或结对编程等最佳实践来提高代码质量。在软件开发生命周期内，开发人员和操作人员采用代码质量最佳实践。

常见反面模式：

- 在没有进行代码审查的情况下将代码提交到应用程序的主分支。变更会自动部署到生产环境并导致中断。
- 开发新应用程序，而不进行任何单元测试、端到端测试或集成测试。在部署之前无法测试应用程序。
- 您的团队在生产中进行手动变更以解决缺陷问题。变更没有经过测试或代码审查，也不会通过持续集成和交付流程捕获或记录。

建立此最佳实践的好处：通过采用提高代码质量的实践，能够极大限度地减少引入生产中的问题。代码质量有助于使用结对编程、代码审查和实施人工智能生产力工具等最佳实践。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

实施提高代码质量的实践，以便在部署代码之前尽可能减少缺陷。使用测试驱动型开发、代码审查和结对编程等实践来提高开发的质量。

利用 Amazon Q Developer 生成式人工智能的强大功能，提高开发人员的工作效率和代码质量。Amazon Q Developer 包括生成代码建议（基于大型语言模型）、制作单元测试（包括边界条件），以及通过检测和修复安全漏洞来增强代码安全性。

客户示例

AnyCompany Retail 采用几种做法来提高代码质量。他们采用了测试驱动型开发作为编写应用程序的标准。对于一些新功能，他们让开发人员在冲刺阶段结对编程。在集成和部署之前，由高级开发人员对每个拉取请求进行代码审查。

实施步骤

1. 在持续集成和交付流程中采用测试驱动型开发、代码审查和结对编程等代码质量实践。使用这些技术来提高软件质量。
   a. Amazon Q Developer 是一款生成式人工智能工具，有助于创建单元测试用例（包括边界条件）、使用代码和注释生成函数、实现众所周知的算法、检测代码中的安全策略违规和漏洞、检测密钥、扫描基础设施即代码（IaC）、记录代码并更快地学习第三方代码库。
   b. Amazon CodeGuru Reviewer 可以使用机器学习为 Java 和 Python 代码提供编程建议。
   c. 您可以使用 AWS Cloud9 创建共享开发环境，然后可以在这个环境中合作开发代码。

实施计划的工作量级别：中等。实施此最佳实践有很多方法，但获得组织采用可能并非易事。

资源

相关最佳实践：

- OPS05-BP02 测试并验证变更
- OPS05-BP06 共享设计标准

相关文档：

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer

- Agile Software Guide
- My CI/CD pipeline is my release captain
- Automate code reviews with Amazon CodeGuru Reviewer
- Adopt a test-driven development approach
- How DevFactory builds better applications with Amazon CodeGuru
- On Pair Programming
- RENGA Inc. automates code reviews with Amazon CodeGuru
- The Art of Agile Development: Test-Driven Development
- Why code reviews matter (and actually save time!)

相关视频：

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Continuous improvement of code quality with Amazon CodeGuru
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development

相关服务：

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler
- AWS Cloud9

OPS05-BP08 使用多个环境

使用多个环境来试验、开发和测试您的工作负载。当环境接近于生产环境时，逐步加强控制，以确保工作负载在部署后能够按预期运行。

期望的结果： 您有多个环境，这些环境均反映您的合规性和监管需求。在通往生产的道路上，您可以通过环境来测试和推广代码。

常见反模式：

- 您正在共享开发环境中执行开发，另一位开发人员将覆盖您的代码更改。
- 共享开发环境上严苛的安全控制令您无法试验新的服务和功能。
- 您在生产系统上执行负载测试，导致用户停机。
- 生产中发生了严重错误，导致数据丢失。在生产环境中，您尝试重新创建导致数据丢失的条件，以便能够确定它是如何发生的，并防止它再次发生。为了防止在测试期间再次丢失数据，您被迫采取措施，导致用户无法使用应用程序。
- 您正在运行多租户服务，无法支持客户对专用环境的请求。
- 您可能并不总是进行测试，但在需要测试时，您在生产环境中进行。
- 您认为单一环境的简单性比更改在环境中的影响范围更加重要。

建立此最佳实践的好处： 您可以为多个同时进行的开发、测试和生产环境提供支持，而不会在开发人员或用户社区间造成冲突。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

使用多个环境，为开发人员提供控制机制最少的沙盒环境，以协助进行试验。提供单独的开发环境以协助并行工作，并提高开发的敏捷性。在接近生产的环境中实施更严格的控制，让开发人员能够创新。使用基础设施即代码和配置管理系统来部署与生产环境中的控制机制配置一致的环境，以便确保系统在部署后按照预期运行。关闭不使用的环境，以免空闲资源（例如晚上和周末的开发系统）产生费用。在负载测试时部署与生产等效的环境，以改善有效结果。

资源

相关文档：

- AWS 上的实例计划程序
- 什么是 AWS CloudFormation？

OPS05-BP09 频繁进行可逆的小规模更改

频繁进行可逆的小规模变更可以减少变更的范围和影响。当与变更管理系统、配置管理系统以及构建和交付系统结合使用时，频繁进行可逆的小规模更改可以减少变更的范围和影响。这样可以提高故障排除工作的效果、加快修复速度，并支持回滚更改。

常见反模式：

- 您每季度部署一个新版本的应用程序，这会存在一个变更窗口，意味着核心服务将关闭。
- 您经常更改数据库架构，而不在管理系统中跟踪变更。
- 您执行手动就地更新，覆盖现有安装和配置，并且没有明确的回滚计划。

建立此最佳实践的好处： 频繁部署小的更改有助于提高开发速度。更改很小时，更易于确定是否会带来意外后果，并且更容易撤回。更改可逆时，由于简化了恢复，因此实施更改的风险更小。变更过程的风险降低，变更失败的影响也减小。

未建立这种最佳实践的情况下暴露的风险等级： 低

实施指导

频繁进行可逆的小规模变更可以减小变更的范围和影响。这可以简化故障排除、加快修复速度，并支持回滚更改。这还可以加快企业实现价值的速度。

资源

相关最佳实践：

- OPS05-BP03 使用配置管理系统
- OPS05-BP04 使用构建和部署管理系统
- OPS06-BP04 自动测试和回滚

相关文档：

- 在 AWS 上实施微服务
- 微服务 - 可观测性

OPS05-BP10 完全自动化集成和部署

实现自动构建、部署和测试工作负载。这可以减少手动过程引起的错误，并减少部署更改的工作量。

使用 资源标签 和 AWS Resource Groups ，按照一致的 标记策略 应用元数据，以协助标识您的资源。标记您的资源，以便进行整理、成本核算、访问控制并有针对性地自动执行操作活动。

期望的结果： 开发人员使用工具来交付代码并推广到生产环境。开发人员无需登录 AWS Management Console即可提供更新。对变更和配置进行全面的审计跟踪，可满足监管和合规需求。流程是可重复的，并且跨团队实现标准化。开发人员可以腾出时间专注于开发和代码推送，从而提高工作效率。

常见反模式：

- 星期五，您完成为功能分支编写新代码的工作。星期一，在运行代码质量测试脚本和各单元测试脚本后，您将代码签入计划发行的下一版本中。
- 您接到任务，需要为重要问题编写修复代码，该问题在生产中影响了大量客户。对修复代码进行测试后，您提交代码并通过电子邮件发送变更管理，请求批准，以将其部署到生产环境中。
- 作为开发人员，您可以登录 AWS Management Console，以使用非标准方法和系统创建新的开发环境。

建立此最佳实践的好处： 通过自动构建和部署管理系统，可以减少手动流程引起的错误，并减少部署更改的工作量，让您的团队成员能够专注于实现业务价值。可以在推广到生产环境时提高交付速度。

未建立这种最佳实践的情况下暴露的风险等级： 低

实施指导

您使用构建和部署管理系统来跟踪并实施更改，以便减少手动流程引起的错误，并减少工作量。将集成和部署管道完全自动化，从代码签入到构建、测试、部署和验证都包含在内。这可以缩短准备时间，鼓励更频繁地进行更改，减少工作量，提高面市速度，提升生产力，并增进代码在推广到生产环境时的安全性。

资源

相关最佳实践：

- OPS05-BP03 使用配置管理系统
- OPS05-BP04 使用构建和部署管理系统

相关文档：

- 什么是 AWS CodeBuild？
- 什么是 AWS CodeDeploy？

相关视频：

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

降低部署风险

采用可提供快速质量反馈，并且若更改没有达到目标成效，则支持快速恢复的方法。使用这些实践可以减轻因部署更改而产生的问题的影响。

工作负载的设计应包括其部署、更新和运营方式。您需要实施以减少缺陷并快速安全地修复为目标的工程实践。

最佳实践

- OPS06-BP01 针对不成功的更改制定计划
- OPS06-BP02 测试部署
- OPS06-BP03 采用安全部署策略
- OPS06-BP04 自动测试和回滚

OPS06-BP01 针对不成功的更改制定计划

制定计划，以便在部署没有达到期望结果时，在生产环境中恢复到已知良好状态，或者进行修复。制定一项策略来建立这样的计划，有助于所有团队制定从失败的更改中恢复的策略。这样的策略示例包括部署和回滚步骤、更改策略、功能标记、流量隔离和流量转移。单个发布可能包括多个相关的组件更改。该策略应提供承受任何组件更改的失败或从中恢复过来的能力。

期望的结果： 您已经为更改失败准备了详细的恢复计划。此外，您还缩小了发布内容的大小，以最大限度地减少对其他工作负载组件的潜在影响。因此，您通过缩短更改失败可能造成的停机时间，提高恢复时间的灵活性和效率，减少了对业务的影响。

常见反模式：

- 执行部署后，应用程序变得不稳定，但是系统上似乎还有活动用户。您必须决定是回滚更改并影响活动用户，还是等到知道用户无论如何都可能受到影响后再回滚更改。
- 执行例行更改后，可以访问新环境，但是其中一个子网无法访问。您必须决定是回滚所有内容还是尝试修复无法访问的子网。在您做决定时，子网仍然无法访问。
- 您的系统的架构不允许使用较小的发布版本进行更新。因此，在部署失败时，您很难撤销这些大批量的更改。
- 您没有使用基础设施即代码（IaC）模式，而且对基础设施进行的手动更新导致了不希望出现的配置。您无法有效地跟踪和撤销手动更改。
- 由于您没有将部署频率的增加作为衡量标准，因此团队没有动力来缩小更改规模，也不愿意改进每次更改的回滚计划，从而导致风险增加和失败率上升。

- 您没有衡量因更改失败而导致的中断的总持续时间。您的团队无法确定部署流程的优先顺序和恢复计划的有效性，也无法进行改进。

建立此最佳实践的好处： 制定从失败更改中恢复的计划可以最大限度地缩短平均恢复时间（MTTR，Mean Time To Recover），并减少对业务的影响。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

发布团队采用的一致的、有据可查的策略和实践，使组织能够计划在更改失败时应如何处理。该策略应允许在特定情况下向前修复。无论是哪种情况，在部署到实际生产环境之前，都应妥善记录并测试向前修复或回滚计划，以便最大限度地减少从更改中恢复所需的时间。

实施步骤

1. 记录要求团队制定有效计划以在指定时间内撤销更改的策略。
   a. 策略应指定何时允许出现向前修复情况。
   b. 要求所有相关人员都能查阅记录在案的回滚计划。
   c. 指定回滚要求（例如，当发现部署了未经授权的更改时）。
2. 分析与工作负载的每个组件相关的所有更改的影响级别。
   a. 如果可重复的更改遵循的工作流，与执行更改策略的工作流保持一致，则允许对这些更改进行标准化、模板化和预授权。
   b. 通过缩小更改的规模来减少任何更改的潜在影响，从而减少恢复所需的时间和对业务的影响。
   c. 确保回滚过程将代码恢复到已知的良好状态，以尽可能避免意外事件。
3. 集成工具和工作流，以编程方式执行策略。
4. 让其他工作负载所有者能够查看有关更改的数据，以提高对无法回滚的任何失败更改的诊断速度。
   a. 利用可见的更改数据来衡量这一做法是否成功，并确定迭代改进措施。
5. 使用监控工具来验证部署的成败，以加快制定回滚决策的速度。
6. 衡量更改失败时的停机时间，以不断改进恢复计划。

实施计划的工作量级别： 中

资源

相关最佳实践：

- OPS06-BP04 自动测试和回滚

相关文档：

- AWS Builders Library | Ensuring Rollback Safety During Deployments
- AWS 白皮书 | Change Management in the Cloud

相关视频：

- re:Invent 2019 | Amazon's approach to high-availability deployment

OPS06-BP02 测试部署

使用与生产环境相同的部署配置、安全控制、步骤和程序，在预生产环境中测试发布过程。验证所有部署步骤是否按预期完成，如检查文件、配置和服务。通过功能测试、集成测试和负载测试以及运行状况检查等各种监控方法，进一步测试所有更改。通过这些测试，您可以及早发现部署问题，并有机会在进入生产之前规划和缓解问题。

您可以创建临时的并行环境来测试每项更改。使用基础设施即代码（IaC）自动部署测试环境，有助于减少所涉及的工作量，确保稳定性、一致性和更快的功能交付。

期望的结果： 您的组织采用了包含测试部署在内的测试驱动型开发文化。这样可以确保团队专注于提供商业价值，而不是管理发布版本。各团队在发现部署风险后尽早参与进来，以确定适当的缓解方案。

常见反模式：

- 在发布生产版本期间，未经测试的部署会导致问题频发，需要进行故障排除和上报。
- 您的发布版本包含用于更新现有资源的基础设施即代码（IaC）。您不确定 IaC 是会成功运行，还是会对资源造成影响。
- 您在应用程序中部署一项新功能。此功能未按预期运行，并且在受影响的用户报告之前都无从了解问题。
- 您更新了证书。您不小心将证书安装到了错误的组件上，而这却没有被发现，于是因为无法建立与网站的安全连接而影响网站访客。

建立此最佳实践的好处： 在生产前对部署程序及其引入的更改进行全面测试，可最大限度地减少部署步骤对生产的潜在影响。这增强了生产版本发布过程中的信心，并最大限度地减少了运营支持，而且不会减慢更改交付速度。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

测试部署过程与测试部署所产生的更改同样重要。要完成这一步骤，可以在尽可能接近生产环境的预生产环境中测试部署步骤。可以在投入生产之前发现一些常见问题，如部署步骤不完整、不正确或配置错误。此外，您还可以测试恢复步骤。

客户示例

作为持续集成和持续交付（CI/CD，Continuous Integration/Continuous Delivery）管道的一部分，AnyCompany Retail 在类似生产的环境中执行为客户发布基础设施和软件更新所需的既定步骤。该管道包含预检查过程，用于在部署之前检测资源中的偏差（检测在 IaC 之外对资源执行的更改），以及验证 IaC 在启动时采取的操作。该管道会验证部署步骤，例如在向负载均衡器重新注册之前，验证特定文件和配置是否已准备就绪，服务是否处于正在运行状态，以及是否正确响应本地主机上的运行状况检查。此外，所有更改都要进行一系列自动测试，如功能测试、安全测试、回归测试、集成测试和负载测试。

实施步骤

1. 执行预安装检查，模拟生产环境打造预生产环境。
   a. 使用 偏差检测 功能，检测是否在 AWS CloudFormation 之外更改了资源。
   b. 使用 更改集 功能，验证堆栈更新的意图是否与 AWS CloudFormation 在启动更改集时所采取的操作相匹配。
2. 这会在 AWS CodePipeline 中触发手动审批步骤，以授权部署到预生产环境。
3. 使用 AWS CodeDeploy AppSpec 文件等部署配置来定义部署和验证步骤。
4. 在适用的情况下，可将 AWS CodeDeploy 与其他 AWS 服务集成 或 将 AWS CodeDeploy 与合作伙伴的产品和服务集成。
5. 监控部署 - 使用 Amazon CloudWatch、AWS CloudTrail 和 Amazon SNS 事件通知。
6. 执行部署后的自动化测试，包括功能测试、安全测试、回归测试、集成测试和负载测试。
7. 排查 部署问题。
8. 成功验证上述步骤后应启动手动审批工作流，以授权部署到生产环境。

实施计划的工作量级别： 高

资源

相关最佳实践：

- OPS05-BP02 测试并验证变更

相关文档：

- AWS Builders' Library | Automating safe, hands-off deployments | Test Deployments
- AWS 白皮书 | Practicing Continuous Integration and Continuous Delivery on AWS
- The Story of Apollo - Amazon's Deployment Engine
- How to test and debug AWS CodeDeploy locally before you ship your code
- Integrating Network Connectivity Testing with Infrastructure Deployment

相关视频：

- re:Invent 2020 | Testing software and systems at Amazon

相关示例：

- Tutorial | Deploy and Amazon ECS service with a validation test

OPS06-BP03 采用安全部署策略

在安全的生产环境滚动部署中，会对有益更改的流程进行控制，目标是尽可能减少这些更改让客户感知到的任何影响。安全控制措施提供检查机制，用于验证是否达成所期望的结果，并针对由于更改或部署失败所引入的任何缺陷，限制这些缺陷的影响范围。安全滚动部署可包括功能标记、单盒、滚动（金丝雀版本）、不可变、流量分割和蓝绿部署等策略。

期望的结果： 您的企业使用持续集成/持续交付（CI/CD，Continuous Integration/Continuous Delivery）系统，提供自动进行安全滚动部署的功能。团队必须使用适当的安全滚动部署策略。

常见反模式：

- 您将不成功的更改一次性部署到所有生产环境中。因此，所有客户同时受到影响。
- 在同时部署到所有系统时，引入的一个缺陷需要紧急进行修复。为所有客户修复该缺陷需要几天时间。
- 管理生产版本发布需要多个团队的规划和参与。这限制了您为客户更新功能的频率。
- 您通过修改现有系统来执行可变部署。发现更改不成功后，您被迫再次修改系统，还原旧版本，导致恢复时间延长。

建立此最佳实践的好处： 自动化的部署，在快速滚动部署与持续向客户提供有益更改之间取得平衡。限制影响范围可以防止代价高昂的部署失败，并最大限度地提高团队有效应对失败的能力。

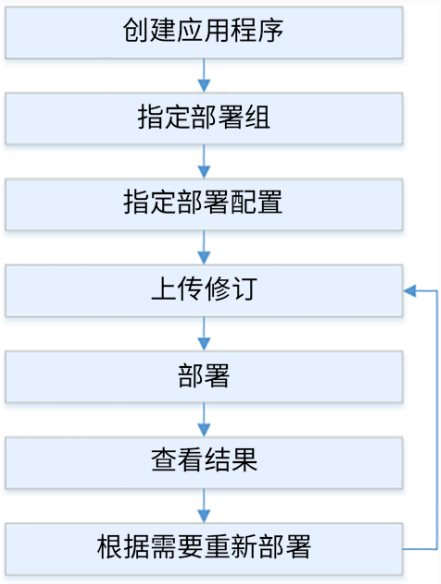
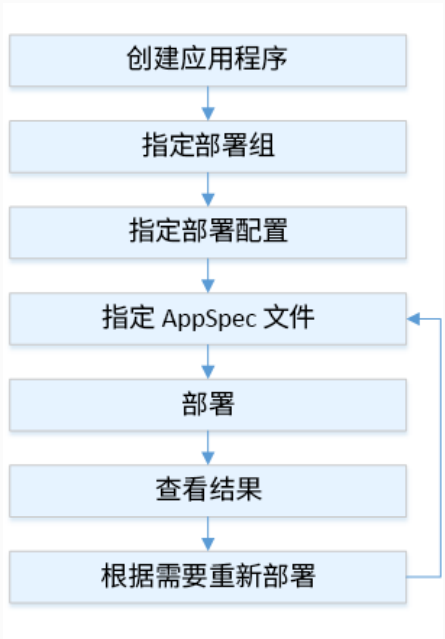
未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

持续交付失败会导致服务可用性降低，带来糟糕的客户体验。为了最大限度地提高部署成功率，请在端到端发布流程中实施安全控制措施，以最大限度地减少部署错误，将达成零部署失败作为目标。

客户示例

AnyCompany Retail 的目标是尽可能减少部署的停机时间，甚至实现零停机，这意味着在部署期间，用户不会感觉到任何影响。为了实现这一目标，公司建立了部署模式（参见以下工作流程图），例如滚动部署和蓝绿部署。所有团队在各自的 CI/CD 管道中都采用了其中一种或多种模式。

Amazon EC2 的 CodeDeploy 工作流程	Amazon ECS 的 CodeDeploy 工作流程	Lambda 的 CodeDeploy 工作流程
		

实施步骤

1. 使用审批工作流程在提升到生产版本后，启动生产版本滚动部署步骤序列。

2. 使用自动化部署系统，例如 AWS CodeDeploy。AWS CodeDeploy 部署选项 包括 EC2/本地就地部署及 EC2/本地蓝绿部署、AWS Lambda 和 Amazon ECS（参见前面的工作流程图）。
   a. 在适用的情况下，可将 AWS CodeDeploy 与其它 AWS 服务集成 或 将 AWS CodeDeploy 与合作伙伴的产品和服务集成。
3. 对于 Amazon Aurora 和 Amazon RDS 等数据库，使用蓝绿部署。
4. 监控部署 - 使用 Amazon CloudWatch、AWS CloudTrail 和 Amazon Simple Notification Service（Amazon SNS）事件通知。
5. 执行部署后的自动化测试，包括功能测试、安全测试、回归测试、集成测试以及任何负载测试。
6. 排查 部署问题。

实施计划的工作量级别： 中

资源

相关最佳实践：

- OPS05-BP02 测试并验证变更
- OPS05-BP09 频繁进行可逆的小规模更改
- OPS05-BP10 完全自动化集成和部署

相关文档：

- AWS Builders Library | Automating safe, hands-off deployments | Production deployments
- AWS Builders Library | My CI/CD pipeline is my release captain | Safe, automatic production releases
- AWS 白皮书 | Practicing Continuous Integration and Continuous Delivery on AWS | Deployment methods
- AWS CodeDeploy User Guide
- Working with deployment configurations in AWS CodeDeploy
- Set up an API Gateway canary release deployment
- Amazon ECS Deployment Types
- Fully Managed Blue/Green Deployments in Amazon Aurora and Amazon RDS
- Blue/Green deployments with AWS Elastic Beanstalk

相关视频：

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

相关示例：

- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- 研讨会 | Buiding CI/CD pipelines for Lambda canary deployments using AWS CDK
- 研讨会 | Blue/Green and Canary Deployment for EKS and ECS
- 研讨会 | Building a Cross-account CI/CD Pipeline

OPS06-BP04 自动测试和回滚

为了提高部署过程的速度和可靠性以及对该过程的信心，您需要制定一项策略，用于在预生产和生产环境中实现自动化的测试和回滚功能。在部署到生产环境时自动进行测试，模拟人与系统的交互，从而验证已经部署了更改。利用自动回滚功能，可以快速恢复到先前已知的良好状态。回滚应在预先定义的条件下自动启动，例如更改未达到期望结果或自动化测试失败时。自动执行这两项活动可以提高部署的成功率，尽可能缩短恢复时间，并减少可能对业务造成的影响。

期望的结果： 您的自动化测试和回滚策略已集成到持续集成/持续交付（CI/CD，Continuous Integration/Continuous Delivery）管道中。您的监控功能可以根据成功标准进行验证，并能在失败时启动自动回滚。这样可以最大限度地减少对最终用户和客户的任何影响。例如，当您对所有测试结果都感到满意时，可以将代码提升到生产环境中，该环境启动了使用相同测试案例的自动回归测试。如果回归测试结果与预期不符，则在管道工作流中启动自动回滚。

常见反模式：

- 您的系统的架构不允许使用较小的发布版本进行更新。因此，在部署失败时，您很难撤销这些大批量的更改。
- 您的部署过程包括一系列人工步骤。将更改部署到工作负载后，您启动了部署后测试。完成测试之后，您发现工作负载不可操作，而且客户断开了连接。然后，您开始回滚到之前的版本。所有这些人工步骤都会延误整个系统的恢复，并对客户造成长时间的影响。
- 您花时间为应用程序中不常用的功能开发了自动化测试案例，这极大地降低了自动化测试功能上的投资回报率。

- 您的发布版本由应用程序、基础设施、补丁和配置更新组成，这些组件相互独立。但是，您只有一个 CI/CD 管道，只能同时交付所有更改。一个组件中的失败会迫使您撤销所有更改，导致回滚过程复杂且效率低下。
- 您的团队完成了冲刺一的编码工作并开始冲刺二的工作，但是按照您的计划，直到冲刺三才会进行测试。结果，自动化测试发现冲刺一中存在缺陷，需要先解决这些缺陷，然后才能开始测试冲刺二的可交付成果，整个发布都被延误，这大大降低了自动化测试的价值。
- 生产版本的自动化回归测试案例已完成，但您没有监控工作负载运行状况。由于无法监控服务是否已重新启动，因此您不确定是否需要回滚或者是否已经进行了回滚。

建立此最佳实践的好处： 自动化测试可提高测试过程的透明度，以及在更短的时间内测试更多功能的能力。通过对生产环境中的更改进行测试和验证，可以让您立即发现问题。利用自动化测试工具改进一致性，可以更好地检测缺陷。通过自动回滚到以前的版本，可以将对客户的影响降至最低。自动回滚可以减少业务影响，最终提升对您部署功能的信心。总体而言，这些功能可在确保质量的同时缩短交付时间。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

自动测试部署的环境，以便更快地确认目标结果。在没有达到预定义的结果时，自动回滚到之前的已知良好状态，尽可能地缩短恢复时间，并减少手动过程引起的错误。将测试工具与管道工作流集成，以便一致地开展测试并尽可能减少手动输入。确定优先执行的自动化测试案例，例如能够降低最大风险的测试案例，以及每次更改都需要频繁测试的测试案例。此外，还可以根据在测试计划中预定义的特定条件自动回滚。

实施步骤

1. 为开发生命周期建立测试生命周期，针对测试过程，从需求规划到测试案例开发、工具配置、自动化测试和测试案例关闭，对每个阶段进行定义。
   a. 根据您的整体测试策略创建特定于工作负载的测试方法。
   b. 考虑在整个开发生命周期中适宜的阶段实施持续测试策略。
2. 根据您的业务需求和管道投资，选择用于测试和回滚的自动化工具。
3. 确定要自动执行哪些测试案例，以及应手动执行哪些测试案例。这个过程可以根据所测试功能的业务价值优先级来定义。确保所有团队成员都遵守该计划，并核实执行手动测试的责任人。
   a. 在自动化测试可以实现价值的特定测试案例上使用自动化测试功能，例如可重复或经常运行的案例、需要重复任务的案例或者多种配置中所需的案例。

   b. 在自动化工具中定义测试自动化脚本和成功标准，以便在特定案例失败时可以启动持续的自动化工作流。
   c. 为自动回滚定义具体的失败标准。
4. 优先考虑测试自动化，在过于复杂和人工交互会导致更高失败风险的案例中，通过开发全面的测试案例来获得一致的结果。
5. 将您的自动化测试和回滚工具集成到 CI/CD 管道中。
   a. 为您的更改制定明确的成功标准。
   b. 监控并观察以检测这些标准，以及在满足特定回滚标准时自动撤销更改。
6. 执行不同类型的自动化生产测试，例如：
   a. A/B 测试，显示在两个用户测试组之间当前版本的结果对比。
   b. 金丝雀测试，让您可以先对一部分用户部署更改，然后再向所有用户发布。
   c. 功能标记测试，让您可以在应用程序外部，每次将新版本的单个功能标记为打开和关闭，以便逐个验证各个新功能。
   d. 回归测试，用于验证现有相互关联组件的新功能。
7. 监控应用程序的操作情况、事务，以及与其他应用程序和组件的交互。编制报告，用于按工作负载显示更改是否成功，这样您就可以确定对自动化和工作流的哪些部分进一步进行优化。
   a. 编制测试结果报告，以便您快速决定是否应调用回滚程序。
   b. 实施策略，以便根据一种或多种测试方法的预定义失败条件进行自动回滚。
8. 开发自动化测试案例，以便在将来的可重复更改中重用。

实施计划的工作量级别： 中

资源

相关最佳实践：

- OPS06-BP01 针对不成功的更改制定计划
- OPS06-BP02 测试部署

相关文档：

- AWS Builders Library | Ensuring rollback safety during deployments
- Redeploy and rollback a deployment with AWS CodeDeploy
- 8 best practices when automating your deployments with AWS CloudFormation

相关示例：

- 使用 Selenium、AWS Lambda、AWS Fargate (Fargate) 和 AWS 开发人员工具进行无服务器 UI 测试

相关视频：

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

运营准备和更改管理

评估工作负载、流程和程序以及工作人员的运营准备就绪情况，以了解与工作负载相关的运营风险。管理环境中的更改流。

您应该使用一致的流程（包括手动或自动化检查清单）来了解何时可运营工作负载或进行更改。这也有助于您发现需要制定计划予以解决的任何问题。您需要有记录日常活动的运行手册和指导问题解决过程的行动手册。使用一种机制来管理支持交付商业价值的更改，并帮助减轻与更改相关的风险。

最佳实践

- OPS07-BP01 确保员工能力
- OPS07-BP02 确保以一致的方式对运维准备情况进行审查
- OPS07-BP03 使用运行手册执行程序
- OPS07-BP04 根据行动手册调查问题
- OPS07-BP05 做出明智的决策来部署系统和变更
- OPS07-BP06 为生产工作负载启用支持计划

OPS07-BP01 确保员工能力

通过一种机制来验证您是否有适当数量训练有素的员工来支持工作负载。他们必须接受构成工作负载的平台和服务方面的培训。为他们提供运营工作负载所需的知识。您必须有足够训练有素的员工来支持工作负载的正常运营和排查发生的意外事件。拥有足够的员工轮流值班和休假，避免疲劳。

期望结果：

- 在工作负载可用时，有足够训练有素的员工为工作负载提供支持。

- 为员工提供构成工作负载的软件和服务方面的培训。

常见反模式：

- 部署工作负载，但没有经过培训的团队成员来运营所使用的平台和服务。
- 没有足够的员工来支持轮流值班或休假。

建立此最佳实践的好处：

- 拥有技能娴熟的团队成员能够为您的工作负载提供有效支持。
- 有足够的团队成员，可以支持工作负载和实现轮流值班，同时降低疲劳风险。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

确认有足够的训练有素的员工来支持工作负载。确认有足够的团队成员来执行正常运营活动，包括轮流值班。

客户示例

AnyCompany Retail 确保支持工作负载的团队得到适当的人员配备和培训。他们有足够的工程师支持轮流值班。他们为员工提供有关构建工作负载的软件和平台方面的培训，并鼓励他们获得认证。他们有足够的员工，所以员工可以休假，同时仍然可以支持工作负载和轮流值班。

实施步骤

1. 分配足够数量的员工来运营和支持您的工作负载，并且支持轮流值班。
2. 在构成工作负载的软件和平台方面对员工进行培训。
   a. AWS 培训与认证包括有关 AWS 的课程库。这里提供线上和线下的免费和付费课程。
   b. AWS 主持活动和网络研讨会，以便向 AWS 专家学习。
3. 随着运营条件和工作负载发生变化，定期评估团队规模和技能。调整团队规模和技能以满足运营要求。

实施计划的工作量级别：高。雇用和培训团队以支持工作负载需要付出巨大的努力，但可带来可观的长期利益。

资源

相关最佳实践：

- OPS11-BP04 执行知识管理 - 团队成员必须具备运营和支持工作负载所需的信息。知识管理是提供这些信息的关键。

相关文档：

- AWS 活动和网络研讨会
- AWS 培训和认证

OPS07-BP02 确保以一致的方式对运维准备情况进行审查

使用运维准备情况审查（ORR，Operational Readiness Review），确保可以运营您的工作负载。ORR 是 Amazon 开发的一种机制，用于验证团队可以安全地运营其工作负载。ORR 是一个使用要求核对清单进行审查和检查的过程。ORR 是一种自助服务体验，供团队用于验证其工作负载。ORR 中包含的最佳实践源自我们多年构建软件的经验教训。

ORR 核对清单包括架构推荐、运维过程、事件管理和发布质量。我们的更正错误（CoE，Correction of Error）流程是这些项目的主要推动因素。您的事后分析应该可以推动自己的 ORR 演进。ORR 并不仅仅关系到遵循最佳实践，还关系到预防以前的事件再次发生。最后，ORR 中还可以包括安全性、监管和合规性要求。

在工作负载正式公开发布之前运行 ORR，然后在整个软件开发生命周期中运行 ORR。在发布之前运行 ORR 可以提升安全运营工作负载的能力。对工作负载定期重新运行 ORR 可以收集任何偏离最佳实践的情况。您可以准备用于新服务发布的 ORR 以及用于定期审查的 ORR。这可以帮助您遵循最新制定的最佳实践，并吸取从事后分析中学到的经验教训。随着您对云的使用日趋成熟，您可以将 ORR 要求作为默认设置整合到自己的架构中。

期望的结果： 您已准备好 ORR 核对清单，其中包括适合您组织的最佳实践。在工作负载发布之前运行 ORR。在整个工作负载生命周期中定期运行 ORR。

常见反模式：

- 您启动了工作负载，但不知道谁负责其运维工作。
- 在验证工作负载以便发布时，没有包括监管和安全性要求。
- 没有定期重新评估工作负载。

- 发布工作负载而没有准备好所需的规程。
- 您在多个工作负载中看到相同的根本原因反复导致出现故障。

建立此最佳实践的好处：

- 您的工作负载包括架构、流程和管理最佳实践。
- 学到的经验教训可合并到 ORR 流程中。
- 在工作负载发布时已准备好所需的规程。
- 在工作负载的整个软件生命周期中运行 ORR。

未建立这种最佳实践的情况下的风险等级： 高

实施指导

ORR 关系到两点：流程和核对清单。ORR 流程应该由您的组织采用并获得高管支持。至少，ORR 必须在工作负载正式公开发布之前已运行。在整个软件开发生命周期中运行 ORR 可确保软件始终遵循新的最佳实践或新要求。ORR 核对清单应包括配置项目、安全性和监管要求，以及组织的最佳实践。在一段时间后，您可以使用 AWS Config、 AWS Security Hub和 AWS Control Tower 防护机制等服务，将源自 ORR 的最佳实践整合到防护机制中，以实现自动化的最佳实践检测。

客户示例

在经历了多起生产事件之后，AnyCompany Retail 决定实施 ORR 流程。他们构建了核对清单，其中包括最佳实践、监管和合规性要求，以及从中断中学到的经验教训。在发布新工作负载之前，运行 ORR。每个工作负载会每年运行一次 ORR，其中包括一小组最佳实践，用于整合添加到 ORR 核对清单中的新最佳实践和要求。在一段时间后，AnyCompany Retail 使用 AWS Config 来检测一些最佳实践，以加快 ORR 流程。

实施步骤

如需详细了解 ORR，请阅读 运维准备情况审查（ORR）白皮书。其中详细介绍了 ORR 流程的历史，如何构建自己的 ORR 实践，以及如何制定自己的 ORR 核对清单。以下步骤是该文档的缩减版本。如需深入了解什么是 ORR 以及如何自行构建，建议您阅读该白皮书。

1. 让关键利益相关方聚在一起讨论，包括来自安全、运维和开发部门的代表。
2. 让每个利益相关方至少提一个要求。对于第一次迭代，请尝试将项目数限制为不超过三十个。
    - 附录 B：ORR 问题示例 源自运维准备情况审查（ORR）白皮书，包含您在开始着手时可借鉴的示例问题。

3. 在电子表格中收集您的要求。
    - 您可以使用 自定义剖析 （位于 AWS Well-Architected Tool 中）开发自己的 ORR，并跨账户以及在 AWS Organization 中分享它们。
4. 确定一个工作负载来运行 ORR。最好选择发布前的工作负载或者内部工作负载。
5. 运行 ORR 核对清单并记录任何发现结果。如果已经有防范措施，那么发现结果可能就不太重要。对于任何没有防范措施的发现结果，请将它们记录到项目的待办事项中，并在发布之前实施它们。
6. 在一段时间后，继续在 ORR 中添加最佳实践和要求。

具有 Enterprise Support 的 AWS Support 客户可以向其技术客户经理请求举行 运维准备情况审查研讨会 。该研讨会是一个交互式研讨会，采用 反推式工作方法 ，可帮助您制定自己的 ORR 核对清单。

实施计划的工作量级别： 高。在组织中采用 ORR 实践需要获得高管以及利益相关方的支持。使用整个组织中获得的反馈意见来构建和更新核对清单。

资源

相关最佳实践：

- OPS01-BP03 评估治理要求 – 监管要求非常适合包括在 ORR 核对清单中。
- OPS01-BP04 评估合规性要求 – 合规性要求有时候包括在 ORR 核对清单中。另一些时候它们可作为单独的流程。
- OPS03-BP07 为团队配置适当的资源 – 团队能力是很适合加入 ORR 要求的候选项。
- OPS06-BP01 针对不成功的更改制定计划 – 在发布工作负载之前，必须建立回滚或前滚计划。
- OPS07-BP01 确保员工能力 – 为了支持工作负载，您必须具备所需的人员。
- SEC01-BP03 识别并验证控制目标 – 安全控制目标会是非常合适的 ORR 要求。
- REL13-BP01 定义停机和数据丢失的恢复目标 – 灾难恢复计划是很好的 ORR 要求。
- COST02-BP01 根据组织的要求制定各种策略 – 成本管理策略非常适合包括在 ORR 核对清单中。

相关文档：

- AWS Control Tower – AWS Control Tower 中的防护机制
- AWS Well-Architected Tool – 自定义剖析
- Adrian Hornsby 提供的运维准备情况审查模板
- 运维准备情况审查（ORR）白皮书

相关视频：

- AWS Support 为您提供支持 | 构建高效的运维准备情况审查（ORR，Operational Readiness Review）

相关示例：

- 运维准备情况审查（ORR）剖析

相关服务：

- AWS Config
- AWS Control Tower
- AWS Security Hub
- AWS Well-Architected Tool

OPS07-BP03 使用运行手册执行程序

运行手册是实现特定结果的书面流程。运行手册由某人为完成某件事而遵循的一系列步骤组成。早在航空发展的早期，运行手册便已用于运营。在云运营中，我们使用运行手册来降低风险并实现预期结果。简单而言，运行手册就是完成一项任务的核对清单。

运行手册是运营工作负载的重要组成部分。从新团队成员入职到部署一个主要版本，运行手册都是一个成文的流程，无论谁使用它们，都能获得一致的结果。运行手册应发布在一个中央位置，并随着流程的发展而更新，因为更新运行手册是变更管理流程的一个关键组成部分。它们还应包括关于错误处理、工具、权限、异常和问题发生时上报的指导。

随着贵组织日益成熟，开始自动编写运行手册。从简短且经常使用的运行手册开始。使用脚本语言来实现步骤自动化或使步骤更容易执行。当您自动化前几本运行手册后，您将花时间自动化更复杂的运行手册。随着时间的推移，大多数运行手册应以某种方式实现自动化。

期望结果：您的团队有一系列执行工作负载任务的分步指南。运行手册包含期望结果、必要的工具和权限，以及关于错误处理的说明。运行手册存储在一个中央位置（版本控制系统）并经常更新。例如，在应用程序发出警报、出现操作问题和计划内生命周期事件期间，您的运行手册可为您的团队提供监控、沟通和响应关键账户的 AWS Health 事件的功能。

常见反面模式：

- 依靠记忆完成流程的每个步骤。
- 手动部署更改而不使用核对清单。
- 不同的团队成员执行相同的流程，但执行不同的步骤或取得不同的结果。
- 让运行手册与系统更改和自动化不同步。

建立此最佳实践的好处：

- 降低人工任务的错误率。
- 以一致的方式执行操作。
- 新的团队成员可以更早地开始执行任务。
- 可以自动编写运行手册以减少工作量。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

根据贵组织的成熟度级别，运行手册可以采用多种形式。它们至少应该包含一个分步文本文档。应明确指出期望结果。清楚地记录必要的特殊权限或工具。提供关于错误处理和出现问题时进行上报的详细指导。列出运行手册负责人，并将运行手册发布在一个中央位置。一旦运行手册编写完成，让您团队中的其他人运行它来进行验证。随着过程的发展，根据变更管理流程更新运行手册。

随着贵组织日益成熟，您的文本运行手册应实现自动化。使用 AWS Systems Manager Automation 等服务，可以将纯文本转换为可针对您的工作负载运行的自动化功能。这些自动化功能可以根据事件的发生而运行，从而减轻维持工作负载的运营负担。AWS Systems Manager Automation 还提供了低代码可视化设计体验，可以更轻松地创建自动化运行手册。

客户示例

AnyCompany Retail 必须在软件部署期间执行数据库模式更新。云运营团队与数据库管理团队合作，构建了一个用于手动部署这些更改的运行手册。运行手册以核对清单的形式列出了流程中的每个步骤。其中有一节是关于出错时的错误处理。他们在内部 Wiki 上发布了该运行手册和其它运行手册。云运营团队计划在未来的冲刺阶段实现运行手册的自动化。

实施步骤

如果您没有现有的文档存储库，那么版本控制存储库是开始构建运行手册库的绝佳场所。您可以使用 Markdown 构建运行手册。我们提供了一个示例运行手册模板，您可以用该模板开始构建运行手册。

```
# Runbook Title
## Runbook Info
| Runbook ID | Description | Tools Used | Special Permissions | Runbook Author | Last
 Updated | Escalation POC |
|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this runbook for? What is the desired outcome? | Tools | Permissions
 | Your Name | 2022-09-21 | Escalation Name |
## Steps
1. Step one
2. Step two
```

1. 如果您当前尚没有文档存储库或 Wiki，请在版本控制系统中创建一个新的版本控制存储库。
2. 识别一个没有运行手册的流程。一个理想的流程是半定期执行的流程，步骤少，且故障影响小。
3. 在文档存储库中，使用模板创建新的草稿 Markdown 文档。填写运行手册书名，并填写“运行手册信息”下的必填字段。
4. 从第一步开始，填写运行手册的“步骤”部分。
5. 将运行手册交给团队成员。让他们使用运行手册来验证这些步骤。如果有遗漏或需要澄清的地方，请更新运行手册。
6. 将运行手册发布到您的内部文档存储区。发布后，告诉您的团队和其它利益相关者。
7. 随着时间的推移，您将构建一个运行手册库。随着该库的增长，开始努力实现运行手册的自动化。

实施计划的工作量级别：低。运行手册的最低标准是一个分步文本指南。实现运行手册自动化可能会增加实施工作量。

资源

相关最佳实践：

- OPS02-BP02 确定流程和程序负责人
- OPS07-BP04 根据行动手册调查问题
- OPS10-BP01 使用流程来管理事件、意外事件和问题
- OPS10-BP02 针对每个警报设置一个流程
- OPS11-BP04 执行知识管理

相关文档：

- AWS Well-Architected Framework: Concepts: Runbook development
- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: Working with runbooks
- Migration playbook for AWS large migrations - Task 4: Improving your migration runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

相关视频：

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response
- How to automate IT Operations on AWS | Amazon Web Services
- Integrate Scripts into AWS Systems Manager

相关示例：

- Well-Architected Labs: Automating operations with Playbooks and Runbooks
- AWS 博客文章：Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Systems Manager: Automation walkthroughs
- AWS Systems Manager: Restore a root volume from the latest snapshot runbook
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Gitlab - Runbooks
- Rubix - A Python library for building runbooks in Jupyter Notebooks
- Using Document Builder to create a custom runbook

相关服务：

- AWS Systems Manager Automation

OPS07-BP04 根据行动手册调查问题

行动手册是用于调查意外事件的分步指南。发生意外事件时，行动手册用于开展调查，以及确定影响的范围和根本原因。行动手册可用于从失败的部署到安全事件的各种场景。在许多情况下，行动手册可确定根本原因，而运行手册可用来缓解其带来的风险。行动手册是贵组织事件响应计划的必要组成部分。

出色的行动手册有几个主要特点。它逐步指导用户完成事件发现过程。由外而内地思考，用户应执行哪些步骤来诊断意外事件？ 如果行动手册中需要特殊工具或提升的权限，请在行动手册中明确地定义。请制定沟通计划，以向利益相关者提供有关调查状态的最新信息，这是事件响应计划的一个重要组成部分。在无法确定根本原因的情况下，行动手册应制定上报计划。如果确定了根本原因，行动手册应指出介绍如何解决根本原因的运行手册。应集中存储并定期维护行动手册。如果行动手册用于特定提醒，请向团队提供关于提醒中的行动手册的提示。

随着组织日趋成熟，可自动实施行动手册。从包含低风险意外事件的行动手册开始实施。使用脚本自动执行发现步骤。确保有配套的运行手册来缓解常见根本原因带来的风险。

期望结果：贵组织有针对常见意外事件的行动手册。行动手册集中存储在一个位置，可供团队成员使用。行动手册经常进行更新。对于任何已知的根本原因，将制定配套的运行手册。

常见反面模式：

- 要调查意外事件，并没有标准方法。
- 团队成员依靠肌肉记忆或对机构的了解，对失败的部署进行排查。
- 新的团队成员将学习如何通过试错法来调查问题。
- 调查问题的最佳实践无法在不同团队之间共享。

建立此最佳实践的好处：

- 行动手册有助于您减轻意外事件带来的影响。
- 不同的团队成员可使用同一行动手册，以一致的方式确定根本原因。
- 可以针对已知的根本原因制定运行手册，从而加快恢复速度。
- 团队成员根据行动手册能够更快地开始行动。
- 团队可以使用可重复的行动手册来扩展其流程。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

制定和使用行动手册的方式取决于组织的成熟度。如果您是初次使用云，请在中央文档存储库中以文本形式制定行动手册。随着组织日趋成熟，可以使用 Python 等脚本语言实现行动手册的半自动化。可以在 Jupyter notebook 中运行这些脚本来加快发现速度。先进的组织已针对可通过运行手册自动修正的常见问题，制定完全自动化的行动手册。

通过列出工作负载所发生的常见意外事件，开始制定行动手册。为风险较低且根本原因范围已缩小到几个问题的意外事件选择行动手册。在为较简单的场景制定行动手册后，可以着手处理风险较高的场景或根本原因尚不确定的场景。

随着贵组织日趋成熟，您的文本样式的行动手册应实现自动化。使用 AWS Systems Manager Automation 之类的服务，可以将纯文本转换为自动化代码。可以针对工作负载运行这些自动化代码，从而加快调查速度。可以激活这些自动化代码以响应事件，从而减少发现和解决意外事件所需的平均时间。

客户可以使用 AWS Systems Manager Incident Manager 来响应意外事件。此服务提供了单一界面对事件进行分类，在发现和缓解问题期间通知利益相关者，并在整个意外事件中进行协作。此服务使用 AWS Systems Manager Automation 加快检测和恢复的速度。

客户示例

生产意外事件影响了 AnyCompany Retail。随时待命的工程师根据行动手册调查了问题。随着他们逐步地解决问题，他们不断为行动手册中确定的关键利益相关者提供最新信息。工程师最终确定，根本原因是后端服务中出现竞态条件。根据运行手册，工程师重新启动了该服务，并使 AnyCompany Retail 重新联机。

实施步骤

如果您当前没有文档存储库，建议您为行动手册库创建版本控制存储库。您可以使用 Markdown 制定您的行动手册，该服务兼容大多数行动手册自动化系统。如果您从头开始制定行动手册，请使用以下行动手册示例模板。

```
# Playbook Title
## Playbook Info
| Playbook ID | Description | Tools Used | Special Permissions | Playbook Author | Last
 Updated | Escalation POC | Stakeholders | Communication Plan |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this playbook for? What incident is it used for? | Tools |
 Permissions | Your Name | 2022-09-21 | Escalation Name | Stakeholder Name | How will
 updates be communicated during the investigation? |
## Steps
1. Step one
2. Step two
```

1. 如果您当前没有文档存储库或 Wiki，请在版本控制系统中为行动手册创建一个新的版本控制存储库。
2. 确定需要调查的一个常见问题。这应该是根本原因范围限于几个问题且解决方案风险较低的场景。

3. 利用 Markdown 模板，填写“行动手册书名”部分，并填写“行动手册信息”下的字段。
4. 填写问题排查步骤。尽可能清楚地知道要采取哪些行动，或者应调查哪些方面。
5. 将行动手册提供给团队成员，让他们仔细阅读并加以验证。如果发现有遗漏之处或某些内容不清楚，请更新行动手册。
6. 在文档存储库中发布您的行动手册，并告知您的团队和任何利益相关者。
7. 随着您添加更多的行动手册，这个行动手册库将会不断扩大。在您拥有多个行动手册后，可以开始使用 AWS Systems Manager Automation 等工具自动执行它们，从而使自动化操作和行动手册保持同步。

实施计划的工作量级别：低。行动手册应该是集中存储在一个位置的文本文档。对于更加成熟的组织，将转为自动实施行动手册。

资源

相关最佳实践：

- OPS02-BP02 确定流程和程序负责人
- OPS07-BP03 使用运行手册执行程序
- OPS10-BP01 使用流程来管理事件、意外事件和问题
- OPS10-BP02 针对每个警报设置一个流程
- OPS11-BP04 执行知识管理

相关文档：

- AWS Well-Architected Framework: Concepts: Playbook development
- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: Working with runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

相关视频：

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response (SEC318-R1)
- AWS Systems Manager Incident Manager - AWS Virtual Workshops
- Integrate Scripts into AWS Systems Manager

相关示例：

- AWS Customer Playbook Framework
- AWS Systems Manager: Automation walkthroughs
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Rubix – A Python library for building runbooks in Jupyter Notebooks
- Using Document Builder to create a custom runbook
- Well-Architected Labs: Automating operations with Playbooks and Runbooks
- Well-Architected Labs: Incident response playbook with Jupyter

相关服务：

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 做出明智的决策来部署系统和变更

为工作负载的成功和不成功变更制定恰当的流程。故障演练是一种演习，团队模拟发生故障的情况来制定缓解策略。使用故障演练来预测故障，并在适当的时候创建程序。评估将变更部署到工作负载所获得好处和产生的风险。确认所有变更符合监管要求。

期望结果：

- 将变更部署到工作负载时作出明智的决策。
- 变更符合监管要求。

常见反模式：

- 将变更部署到工作负载，而没有处理失败部署的流程。
- 对生产环境作出不符合监管要求的变更。
- 部署新版本的工作负载，而不为资源利用建立基准。

建立此最佳实践的好处：

- 为工作负载的不成功变更做好了准备。

- 工作负载的变更符合监管策略。

在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

使用故障演练制定不成功变更的流程。记录不成功变更的流程。确保所有变更符合监管要求。评估将变更部署到工作负载所获得好处和产生的风险。

客户示例

AnyCompany Retail 定期执行故障演练以验证他们的不成功变更流程。他们在共享的 Wiki 中记录他们的流程并经常更新。所有变更符合监管要求。

实施步骤

1. 将变更部署到工作负载时作出明智的决策。确立并审查成功部署的条件。制定将触发变更回滚的方案或条件。在部署变更带来的好处与不成功变更的风险之间进行权衡。
2. 确认所有变更符合监管政策。
3. 使用故障演练为不成功的变更制定计划，并记录缓解策略。运行桌面练习，为不成功的变更建模，并验证回滚程序。

实施计划的工作量级别：适中。实施故障演练的实践需要整个组织内的利益攸关方进行协调和付出努力。

资源

相关最佳实践：

- OPS01-BP03 评估治理要求 - 监管要求是确定是否部署变更的关键因素。
- OPS06-BP01 针对不成功的更改制定计划 - 制定计划来缓和失败的部署并使用故障演练来验证它们。
- OPS06-BP02 测试部署 - 在部署之前应适当地测试每项软件变更，以便减少生产中的缺陷。
- OPS07-BP01 确保员工能力 - 有足够训练有素的人员来支持工作负载，这对于作出部署系统变更的明智决定很重要。

相关文档：

- Amazon Web Services：风险与合规
- AWS 责任共担模式
- AWS Cloud 中的监管：敏捷性和安全性之间的适当平衡

OPS07-BP06 为生产工作负载启用支持计划

为生产工作负载所依赖的所有软件和服务启用支持。选择适当的支持级别以满足您的生产服务级别需求。以防出现服务中断或软件问题，这些依赖项的支持计划是必需的。记录支持计划以及如何要求所有服务和软件供应商提供支持。实施机制以确认主要支持联系人的信息保持最新。

期望结果：

- 为生产工作负载所依赖的软件和服务实施支持计划。
- 根据服务级别需求选择适当的支持计划。
- 记录支持计划、支持级别以及如何请求支持。

常见反模式：

- 您没有制定关键软件供应商的支持计划。您的工作负载受到影响，您无法采取任何措施来加快修复或从供应商获得及时更新。
- 作为软件供应商主要联系人的开发人员离开了公司。您无法直接联系供应商支持人员。您必须花时间研究和浏览通用联系系统，延长在需要时作出反应所需的时间。
- 软件供应商发生生产中断。没有关于如何提交支持案例的文档。

建立此最佳实践的好处：

- 通过适当的支持级别，您可以在满足服务级别需求所需的时间范围内获得响应。
- 作为受支持的客户，如果存在生产问题，您可以上报。
- 发生事件时，软件和服务供应商会协助排除故障。

在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

为生产工作负载所依赖的所有软件和服务供应商启用支持计划。设置适当的支持计划以满足服务级别需求。对于 AWS 客户，这意味着要在具有生产工作负载的任何账户上启用 AWS Business Support 或更

高级别的支持。定期与支持供应商会面，获取有关支持产品、流程和联系人的更新。记录如何向软件和服务供应商请求支持，包括在出现中断时如何上报。实施特定机制，使支持联系人的信息保持最新。

客户示例

在 AnyCompany Retail，所有商用软件和服务依赖项均有支持计划。例如，他们在有生产工作负载的所有账户上启用 AWS Enterprise Support。如果出现问题，任何开发人员都可以提出支持案例。有一个 Wiki 页面，其中包含有关如何请求支持、向谁发出通知以及加快处理案例最佳实践的信息。

实施步骤

1. 与组织内的利益攸关方一起确定您的工作负载所依赖的软件和服务供应商。记录这些依赖项。
2. 确定工作负载的服务级别需求。选择与需求匹配的支持计划。
3. 对于商用软件和服务，与供应商一起制定支持计划。
    a. 为所有生产账户订阅 AWS Business Support 或更高级别的支持，可以让 AWS Support 更快响应，并且我们强烈建议订阅此支持。如果您没有 Premium Support，则必须制定行动计划来处理问题，而这需要来自 AWS Support 的帮助。AWS Support 提供工具和技术的组合、人员和计划，旨在主动帮助您优化性能、降低成本和加快创新速度。AWS Business Support 可带来额外的好处，包括访问 AWS Trusted Advisor 和 AWS Personal Health Dashboard，并更快响应。
4. 在您的知识管理工具中记录支持计划。包括如何请求支持、在提出支持案例时应通知谁以及在发生事件时如何上报。Wiki 是一种很好的机制，让任何人都可以在发现支持流程或联系人的更改时对文档进行必要的更新。

实施计划的工作量级别：低。大部分软件和服务供应商都提供选择加入的支持计划。在知识管理系统上记录和分享支持最佳实践，可以确认您的团队知道在出现生产问题时该怎么做。

资源

相关最佳实践：

- OPS02-BP02 确定流程和程序负责人

相关文档：

- AWS Support 计划

相关服务：

- AWS Business Support
- AWS Enterprise Support

运营

成功是指按照您定义的指标衡量，实现了业务成果。通过了解工作负载和运营的运行状况，您可以确定何时组织和业务成果可能陷入风险或已遇到风险，并采取适当的响应措施。

要想取得成功，您必须能够：

主题

利用工作负载可观测性

利用可观测性确保最佳工作负载运行状况。利用相关的指标、日志和跟踪数据，来全面了解工作负载的性能并有效地解决问题。

可观测性使您可以专注于有意义的数据，并了解工作负载的交互和输出。通过专注于基本见解并消除不必要的数据，您可以直截了当地来了解工作负载性能。

这不仅对收集数据至关重要，对正确解读数据也至关重要。定义明确的基准，设置适当的警报阈值，并主动监控任何偏差。关键指标的改变，尤其是与其他数据关联时，可以精确定位特定的问题领域。

借助可观测性，您可以更好地预见和应对潜在挑战，从而确保您的工作负载平稳运行并满足业务需求。

AWS 提供特定的工具，比如用于监控和日志记录的 Amazon CloudWatch ，以及用于分布式跟踪的 AWS X-Ray 。这些服务可以轻松与各种 AWS 资源集成，从而实现高效的数据收集，根据预定义的阈值设置警报，并在控制面板上显示数据以方便解释。利用这些见解，您可以根据自己的运营目标做出以数据为导向的明智决策。

最佳实践

- OPS08-BP05 创建控制面板

OPS08-BP01 分析工作负载指标

实施应用程序遥测后，定期分析收集的指标。虽然延迟、请求、错误和容量（或配额）有助于深入了解系统性能，但优先审查业务成果指标至关重要。这样可以确保您做出与业务目标相一致的数据驱动型决策。

期望的结果： 准确洞察工作负载性能，推动做出以数据为依据的决策，确保与业务目标相一致。

常见反模式：

- 孤立地分析指标，而不考虑其对业务成果的影响。
- 过度依赖技术指标，而不重视业务指标。
- 很少审查指标，错过了实时决策机会。

建立此最佳实践的好处：

- 进一步了解技术性能与业务成果之间的相互关系。
- 以实时数据为依据改善决策流程。
- 在问题影响业务结果之前主动找出和缓解问题。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

利用 Amazon CloudWatch 之类的工具执行指标分析。AWS Cost Anomaly Detection 和 Amazon DevOps Guru 之类的 AWS 服务可用于检测异常，尤其是在静态阈值未知，或行为模式更适合异常检测时。

实施步骤

1. 分析和审查： 定期审查和解释您的工作负载指标。
   a. 优先考虑业务成果指标，而不是只考虑纯粹的技术指标。
   b. 了解数据中的高峰、低谷或模式的重要性。
2. 利用 Amazon CloudWatch： 使用 Amazon CloudWatch 获得集中式视图并进行深入分析。
   a. 配置 CloudWatch 控制面板，以可视化形式呈现您的指标，并对一段时间内的指标进行比较。

b. 使用 CloudWatch 中的百分位数 来清楚地了解指标分布，这有助于定义 SLA 和理解异常值。
c. 设置 AWS Cost Anomaly Detection 在不依赖静态阈值的情况下识别异常模式。
d. 实施 CloudWatch 跨账户可观测性 以监控跨区域内多个账户的应用程序并对其进行故障排除。
e. 使用 CloudWatch Metric Insights 来查询和分析跨账户和地区的指标数据，从而识别趋势和异常情况。
f. 应用 CloudWatch Metric Math 对您的指标进行转换、汇总或执行计算，从而获得更深入的见解。

3. 应用 Amazon DevOps Guru： 纳入 Amazon DevOps Guru 以利用其机器学习增强的异常检测，来识别无服务器应用程序操作问题的早期迹象，并在它们影响客户之前将其修复。
4. 根据见解进行优化： 根据您的指标分析做出明智的决策，以调整和改进您的工作负载。

实施计划的工作量级别： 中

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标
- OPS04-BP02 实施应用程序遥测

相关文档：

- The Wheel 博客 - 强调持续审查指标的重要性
- 百分位很重要
- 使用 AWS Cost Anomaly Detection
- CloudWatch 跨账户可观测性
- 使用 CloudWatch Metrics Insights 查询您的指标

相关视频：

- Enable Cross-Account Observability in Amazon CloudWatch
- Introduction to Amazon DevOps Guru
- Continuously Analyze Metrics using AWS Cost Anomaly Detection

相关示例：

- One Observability Workshop
- Gaining operation insights with AIOps using Amazon DevOps Guru

OPS08-BP02 分析工作负载日志

定期分析工作负载日志对于更深入地了解应用程序的运行至关重要。通过高效地筛选、以可视化方式呈现和解释日志数据，您可以持续优化应用程序性能和安全性。

期望结果：通过全面的日志分析获得对应用程序行为和操作的丰富见解，确保主动检测和缓解问题。

常见反面模式：

- 在出现严重问题之前，忽视对日志的分析。
- 没有使用可分析日志的全套工具，导致错过关键见解。
- 仅依靠人工查看日志，而不利用自动化和查询功能。

建立此最佳实践的好处：

- 主动识别运营瓶颈、安全威胁和其他潜在问题。
- 高效利用日志数据进行持续的应用程序优化。
- 增进对应用程序行为的理解，有助于调试和故障排除。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

Amazon CloudWatch Logs 是用于日志分析的强大工具。利用 CloudWatch Logs Insights 和 Contributor Insights 等集成功能，可以直观而高效地从日志中获取有意义的信息。

实施步骤

1. 设置 CloudWatch Logs：配置应用程序和服务以将日志发送到 CloudWatch Logs。
2. 使用日志异常检测：利用 Amazon CloudWatch Logs 异常检测功能，自动识别异常日志模式并发出警报。该工具有助于您主动管理日志中的异常情况，及早发现潜在问题。
3. 设置 CloudWatch Logs Insights：使用 CloudWatch Logs Insights 以交互方式搜索和分析您的日志数据。

a. 创建查询以提取模式、直观呈现日志数据并得出切实可行的见解。

b. 使用 CloudWatch Logs Insights 模式分析来分析和直观呈现常见的日志规律。该功能有助于您了解日志数据中的常见运行趋势和潜在异常值。

c. 使用 CloudWatch Logs 比较（差异）在不同时间段或不同日志组之间执行差异分析。利用这一功能可精确定位变化，并评测其对系统性能或行为的影响。

4. 使用 Live Tail 实时监控日志：使用 Amazon CloudWatch Logs Live Tail 实时查看日志数据。您可以在应用程序运行活动发生时对其进行主动监控，从而即时了解系统性能和潜在问题。
5. 利用 Contributor Insights：使用 CloudWatch Contributor Insights 在 IP 地址或用户代理等高基数维度中找到主要贡献者。
6. 实施 CloudWatch Logs 指标筛选器：配置 CloudWatch Logs 指标筛选器，以便将日志数据转换为可操作的指标。这允许您设置警报或进一步分析模式。
7. 实施 CloudWatch 跨账户可观测性：监控跨区域内多个账户的应用程序，并排除应用程序出现的故障。
8. 定期审查和优化：定期审查您的日志分析策略，以捕获所有相关信息并持续优化应用程序性能。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标
- OPS04-BP02 实施应用程序遥测
- OPS08-BP01 分析工作负载指标

相关文档：

- Analyzing Log Data with CloudWatch Logs Insights
- Using CloudWatch Contributor Insights
- Creating and Managing CloudWatch Log Metric Filters

相关视频：

- Analyze Log Data with CloudWatch Logs Insights
- Use CloudWatch Contributor Insights to Analyze High-Cardinality Data

相关示例：

- CloudWatch Logs Sample Queries
- One Observability Workshop

OPS08-BP03 分析工作负载跟踪数据

分析跟踪数据对于全面了解应用程序的运营过程至关重要。通过以可视化方式呈现和理解各个组件之间的交互，可以微调性能，识别瓶颈，并增强用户体验。

期望结果：清晰地了解应用程序的分布式运营，从而更快地解决问题并增强用户体验。

常见反面模式：

- 忽略跟踪数据，仅依赖日志和指标。
- 不将跟踪数据与关联日志联系起来。
- 忽略从跟踪数据中得出的指标，例如延迟和故障率。

建立此最佳实践的好处：

- 改善故障排除并缩短平均解决时间（MTTR）。
- 深入了解依赖项及其影响。
- 迅速发现并纠正性能问题。
- 利用从跟踪数据中得出的指标作出明智的决策。
- 通过优化组件交互来改善用户体验。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

AWS X-Ray 提供了一个完整套件来分析跟踪数据，从而提供服务交互的整体视图、监控用户活动并检测性能问题。ServiceLens、X-Ray Insights、X-Ray Analytics 和 Amazon DevOps Guru 等功能，可增强从跟踪数据中获得的可操作见解的深度。

实施步骤

以下步骤提供了一种结构化方法，可使用 AWS 服务有效地实施跟踪数据分析：

1. 集成 AWS X-Ray：确保 X-Ray 已与您的应用程序集成，来捕获跟踪数据。
2. 分析 X-Ray 指标：深入研究从 X-Ray 跟踪数据中得出的指标，例如延迟、请求速率、故障率和响应时间分布，方法是使用服务地图监控应用程序的运行状况。
3. 使用 ServiceLens：利用 ServiceLens 地图来增强您的服务和应用程序的可观测性。这允许以集成方式查看跟踪数据、指标、日志、警报和其他运行状况信息。
4. 启用 X-Ray Insights：
   a. 启用 X-Ray Insights 以自动检测跟踪数据中的异常。
   b. 研究见解以查明模式并确定根本原因，例如故障率或延迟增加。
   c. 查阅见解时间表，按时间顺序分析检测到的问题。
5. 使用 X-Ray Analytics：X-Ray Analytics 可用于全面探究跟踪数据、查明规律和挖掘见解。
6. 在 X-Ray 中使用群组：在 X-Ray 中创建群组，以根据高延迟等标准筛选跟踪数据，从而进行更有针对性的分析。
7. 加入 Amazon DevOps Guru：使用 Amazon DevOps Guru，受益于机器学习模型，查明跟踪数据中的运营异常。
8. 使用 CloudWatch Synthetics：使用 CloudWatch Synthetics 来创建用于持续监控您的端点和工作流程的金丝雀。这些金丝雀可以与 X-Ray 集成以提供跟踪数据，用于对正在测试的应用程序进行深入分析。
9. 使用真实用户监控（RUM）：使用 AWS X-Ray 和 CloudWatch RUM，您可以通过下游 AWS 托管服务，从应用程序的最终用户开始分析和调试请求路径。这有助于您识别影响最终用户的延迟趋势和错误。
10. 与日志关联：在 X-Ray 跟踪视图中关联跟踪数据与相关日志，以便从细粒度的角度了解应用程序行为。这允许您查看与跟踪的事务直接关联的日志事件。
11. 实施 CloudWatch 跨账户可观测性：监控跨区域内多个账户的应用程序，并排除应用程序出现的故障。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS08-BP01 分析工作负载指标
- OPS08-BP02 分析工作负载日志

相关文档：

- Using ServiceLens to Monitor Application Health
- Exploring Trace Data with X-Ray Analytics
- Detecting Anomalies in Traces with X-Ray Insights
- Continuous Monitoring with CloudWatch Synthetics

相关视频：

- Analyze and Debug Applications Using Amazon CloudWatch Synthetics & AWS X-Ray
- Use AWS X-Ray Insights

相关示例：

- One Observability Workshop
- Implementing X-Ray with AWS Lambda
- CloudWatch Synthetics Canary Templates

OPS08-BP04 创建可操作的警报

及时检测和响应应用程序行为的偏差至关重要。尤其重要的是，识别基于关键绩效指标（KPI）的结果何时处于风险当中，或何时出现意外的异常情况。基于 KPI 的警报可确保您收到的信号与业务或运营影响直接相关。这种可操作警报的方法可促进主动响应，并有助于维护系统性能和可靠性。

期望结果：接收及时、相关且可操作的警报，以便快速找出和缓解潜在问题，尤其是在 KPI 结果面临风险时。

常见反面模式：

- 设置过多非关键警报，导致警报疲劳。
- 不根据 KPI 对警报进行优先级排序，因此很难理解问题对业务的影响。
- 忽视根本原因，导致针对同一问题出现重复警报。

建立此最佳实践的好处：

- 通过关注可操作且相关的警报，减少警报疲劳。

- 通过主动检测和缓解问题，改善系统的正常运行时间和可靠性。
- 通过与常用的警报和通信工具集成，增进团队协作并更快解决问题。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要创建有效的警报机制，必须使用指标、日志和跟踪数据来标记基于 KPI 的结果何时存在风险，或何时检测到异常情况。

实施步骤

1. 确定关键性能指标（KPI）：确定应用程序的 KPI。警报应与这些 KPI 相关联，以准确反映业务影响。
2. 实施异常检测：
   - 使用 Amazon CloudWatch 异常检测：设置 Amazon CloudWatch 异常检测功能以自动检测异常模式，这有助于您只对真正的异常情况生成警报。
   - 使用 AWS X-Ray Insights：
     a. 设置 X-Ray Insights 以检测跟踪数据中的异常。
     b. 配置 X-Ray Insights 通知，以便在检测到问题时收到提醒。
   - 与 Amazon DevOps Guru 集成：
     a. 利用 Amazon DevOps Guru 的机器学习功能，结合现有数据来检测运营异常。
     b. 导航至 DevOps Guru 中的通知设置，设置异常警报。
3. 实施可操作的警报：设计能够提供足够信息的警报，以便立即采取行动。
   1. 使用 Amazon EventBridge 规则监控 AWS Health 事件，或者以编程方式与 AWS Health API 集成，以便在收到 AWS Health 事件时自动执行操作。这些可以是常规操作，例如将所有计划的生命周期事件消息发送到聊天界面，也可以是特定操作，例如在 IT 服务管理工具中启动工作流。
4. 减少警报疲劳：极大限度地减少非关键警报。当团队被大量无关紧要的警报淹没时，他们可能会失去对关键问题的监督，从而降低警报机制的整体有效性。
5. 设置复合警报：使用 Amazon CloudWatch 复合警报来整合多个警报。
6. 与警报工具集成：纳入多个工具，例如 Ops Genie 和 PagerDuty。
7. 加入 AWS Chatbot：集成 AWS Chatbot 以便将警报转发到 Amazon Chime、Microsoft Teams 和 Slack。
8. 基于日志的警报：使用 CloudWatch 中的日志指标筛选器，根据特定的日志事件创建警报。

9. 审查和迭代：定期重新审视和完善警报配置。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标
- OPS04-BP02 实施应用程序遥测
- OPS04-BP03 实施用户体验遥测
- OPS04-BP04 实施依赖项遥测
- OPS04-BP05 实施分布式跟踪
- OPS08-BP01 分析工作负载指标
- OPS08-BP02 分析工作负载日志
- OPS08-BP03 分析工作负载跟踪数据

相关文档：

- Using Amazon CloudWatch alarms
- Create a composite alarm
- Create a CloudWatch alarm based on anomaly detection
- DevOps Guru Notifications
- X-ray insights notifications
- 通过交互式 ChatOps 对您的 AWS 资源进行监控、操作和故障排除
- Amazon CloudWatch Integration Guide | PagerDuty
- Integrate Opsgenie with Amazon CloudWatch

相关视频：

- Create Composite Alarms in Amazon CloudWatch
- AWS Chatbot Overview
- AWS On Air ft.Mutative Commands in AWS Chatbot

相关示例：

- Alarms, incident management, and remediation in the cloud with Amazon CloudWatch
- Tutorial: Creating an Amazon EventBridge rule that sends notifications to AWS Chatbot
- One Observability Workshop

OPS08-BP05 创建控制面板

控制面板是以人为本的视图，可用于查看工作负载的遥测数据。虽然它们提供了重要的可视化界面，但它们不应取代警报机制，而是补充警报机制。如果精心设计，它们不仅能迅速洞察系统的运行状况和性能，还能为利益相关方提供有关业务成果和问题影响的实时信息。

期望结果：

使用可视化形式，清晰地了解系统和业务运行状况，并可据此采取行动。

常见反面模式：

- 指标过多，使得控制面板过于复杂。
- 依靠的控制面板不会对检测到的异常情况发出警报。
- 不会随着工作负载的演进而更新控制面板。

这种最佳实践的好处：

- 即时了解关键系统指标和 KPI。
- 增进利益相关方的沟通和理解。
- 快速洞察运营问题的影响。

未建立这种最佳实践的情况下的风险等级：中等

实施指导

以业务为中心的控制面板

为业务 KPI 量身定制的控制面板可吸引更广泛的利益相关方。尽管这些人可能对系统指标不感兴趣，但他们热衷于了解这些数字对业务的影响。以业务为中心的控制面板可确保所监控和分析的所有技术和运营指标与总体业务目标同步。这种一致性可让每个人了解什么是至关重要的，什么不太重要，并就此

达成共识。此外，突出业务 KPI 的控制面板往往更具操作性。利益相关方可以快速了解运营状况、需要关注的领域以及对业务成果的潜在影响。

考虑到这一点，在创建控制面板时，请确保技术指标和业务 KPI 之间保持平衡。两者都至关重要，但它们面向不同的受众。理想情况下，您拥有的控制面板应该有助于您全面了解系统的运行状况和性能，同时还要强调关键业务成果及其影响。

Amazon CloudWatch 控制面板是 CloudWatch 控制台中的可自定义主页，方便您通过单一视图监控您的资源，即使这些资源分布在不同 AWS 区域和账户中。

实施步骤

1. 创建基本控制面板：在 CloudWatch 中创建新控制面板，为其指定一个描述性名称。
2. 使用 Markdown 小部件：在深入研究指标之前，使用 Markdown 小部件在控制面板顶部添加文字背景信息。这应该解释控制面板涵盖的内容、所表示的指标的重要性，还可以包含指向其他控制面板和故障排除工具的链接。
3. 创建控制面板变量：适当时纳入控制面板变量，以创建动态和灵活的控制面板视图。
4. 创建指标小部件：添加指标小部件，以可视化形式呈现应用程序发出的各种指标，定制这些小部件以有效呈现系统运行状况和业务成果。
5. Log Insights 查询：利用 CloudWatch Log Insights 从日志中获取可操作的指标，并在控制面板上显示这些见解。
6. 设置警报：将 CloudWatch Alarms 集成到您的控制面板，可以快速查看任何超出阈值的指标。
7. 使用 Contributor Insights：纳入 CloudWatch Contributor Insights 来分析高基数字段，并更清楚地了解您的资源的主要贡献者。
8. 设计自定义小部件：如果标准小部件无法满足特定需求，可以考虑创建自定义小部件。自定义小部件可以从各种数据来源中提取数据，也可以以独特方式表示数据。
9. 使用 AWS Health Dashboard：使用 AWS Health Dashboard 深入了解您的账户健康状况、事件，以及即将发生、可能影响您的服务和资源的变更。您还可以集中查看 AWS Organizations 中的运行状况事件，或者构建自己的自定义控制面板（有关更多详细信息，请参阅相关示例）。
10. 迭代和完善：随着应用程序的演进，请定期重新审视控制面板以确保其仍然适用。

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标

- OPS08-BP01 分析工作负载指标
- OPS08-BP02 分析工作负载日志
- OPS08-BP03 分析工作负载跟踪数据
- OPS08-BP04 创建可操作的警报

相关文档：

- 构建控制面板以获取操作可见性
- Using Amazon CloudWatch Dashboards

相关视频：

- Create Cross Account & Cross Region CloudWatch Dashboards
- AWS re:Invent 2021 - Gain enterprise visibility with AWS Cloud operation dashboards)

相关示例：

- One Observability Workshop
- 使用 Amazon CloudWatch 进行应用程序监控
- AWS Health Events Intelligence Dashboards and Insights
- Visualize AWS Health events using Amazon Managed Grafana

了解运营状况

定义、记录和分析运营指标，以便了解运营团队的活动，从而采取适当的措施。

您的组织应该能够轻松了解自己的运营状况。您需要定义运营团队的业务目标，确定反映这些目标的关键绩效指标，然后根据运营结果制定指标，以获得有用见解。您应该使用这些指标来实施提供业务和技术观点的控制面板和报告，以帮助领导者和利益相关方做出明智决策。

AWS 使您能够轻松地汇总和分析运营日志，以便生成指标，了解您的运营状况，并深入了解运营状况在一段时间内的变化情况。

最佳实践

- OPS09-BP01 使用指标衡量运营目标和 KPI

- OPS09-BP02 通报状态和趋势，确保了解运营情况
- OPS09-BP03 审查运营指标并确定改进优先顺序

OPS09-BP01 使用指标衡量运营目标和 KPI

从您的组织获取定义运营成功的目标和 KPI，并确定指标反映了这些目标和 KPI。将基线设置为参考点，并定期重新评估。制定机制，从团队那里收集这些指标以供评估。

期望的结果：

- 组织运营团队的目标和 KPI 已发布并共享。
- 已建立反映这些 KPI 的指标。示例可能包括：
  - 工单队列深度或平均工单时长
  - 按问题类型分组的工单数量
  - 使用或不使用标准化操作程序（SOP）时处理问题所花费的时间
  - 从失败的代码推送中恢复所花费的时间
  - 呼叫量

常见反模式：

- 由于开发人员被拉去执行故障排除任务，因此而错过部署截止日期。开发团队主张增加人手，但由于无法衡量被占用的时间，因此无法量化他们需要多少人手。
- 设置了一级服务台来处理用户呼叫。随着时间的推移，工作负载越来越多，但没有为一级服务台分配人手。随着通话次数的增加以及问题解决时间的延长，客户满意度下降，但管理层看不到此类问题的任何指标，因此未采取任何行动。
- 有问题的工作负载被移交给单独的运营团队进行处理。与其他工作负载不同，这种新的工作负载没有提供适当的文档和运行手册。因此，团队需要花费更长的时间解决和排除故障。但是，没有任何指标记录这一点，这使得问责制变得难以实施。

建立此最佳实践的好处： 工作负载监控可以显示应用程序和服务的状态，而监控运营团队则可以让所有者深入了解这些工作负载的使用者之间的变化，例如不断变化的业务需求。通过创建能够反映运营状态的指标，衡量这些团队的效率，并根据业务目标对其进行评估。指标可以突出显示支持问题，或确定何时出现偏离服务级别目标的情况。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

安排时间与业务主管和利益相关方会面，以确定服务的总体目标。确定各个运营团队的任务，以及他们可能应对哪些挑战。利用这些信息，针对可能反映这些运营目标的关键绩效指标（KPI）进行集思广益。这些指标可能是客户满意度、从功能构思到部署所花的时间、平均问题解决时间等。

根据 KPI，确定可能最能反映这些目标的指标和数据来源。客户满意度可能是各种指标的组合，例如呼叫等待或回复时间、满意度得分和提出的问题类型。部署时间可能是测试和部署所需的时间，加上需要添加的所有部署后修复的总和。显示不同类型问题所花费的时间（或这些问题的数量）的统计数据，可以提供一个窗口，让您了解需要在哪些方面开展有针对性的工作。

资源

相关文档：

- Amazon QuickSight - 使用 KPI
- Amazon CloudWatch - 使用指标
- 构建控制面板
- How to track your cost optimization KPIs with KPI Dashboard

OPS09-BP02 通报状态和趋势，确保了解运营情况

了解运营状况及其趋势非常有必要，这样才能确定结果何时可能面临风险、是否可以支持新增的工作，或者变更对团队的影响。在运营活动期间，用户和运营团队可通过状态页面获取信息，从而减轻通信渠道的压力并主动传播信息。

期望的结果：

- 运营领导者可以一目了然地了解其团队正在处理的呼叫量，以及可能正在开展的工作（如部署）。
- 当正常运营受到影响时，会向利益相关方和用户群体发出警报。
- 组织领导层和利益相关方可以查看状态页面，以响应警报或影响，并获取与运营事件相关的信息，如联系人、工单信息和预计恢复时间。
- 向领导层和其他利益相关方提供的报告可显示运营统计数据，例如一段时间内的呼叫量、用户满意度分数、未处理工单的数量及其存在时间。

常见反模式：

- 工作负载出现故障，导致服务不可用。用户想知道发生了什么情况，呼叫量激增。管理人员想知道谁在处理问题，从而进一步增加了呼叫量。各运营团队都在努力调查问题，导致工作重复。
- 由于人们渴望获得新功能，数名人员被重新分配到工程工作中。但没有提供后补人员，问题解决时间激增。这些信息没有被记录下来，几周后，在收到用户表达不满的反馈时，领导层才意识到这个问题。

建立此最佳实践的好处： 在业务受到影响的运营事件中，为了解情况而向不同团队查询信息可能会浪费大量时间和精力。通过建立广泛传播的状态页面和控制面板，利益相关方可以快速获得相关信息，例如是否发现了问题、谁在负责处理问题，或者预计何时可以恢复正常运营。这样，团队成员就不必花太多时间与他人沟通状态，而是可以将更多时间花在解决问题上。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

构建控制面板，显示运营团队当前的关键指标，并使运营主管和管理层都能随时访问这些指标。

构建可以快速更新的状态页面，以显示事件何时发生、由谁负责以及谁在协调响应。在此页面上分享用户应考虑的任何步骤或解决方法，并广为分发该位置。鼓励用户在遇到未知问题时先查看此位置。

收集并提供显示一段时间内的运营状况的报告，并将其分发给领导者和决策者，以阐述运营工作以及挑战和需求。

在团队之间共享这些指标和报告，这些指标和报告最能反映目标和 KPI，以及它们在推动变革方面的影响力。在这些活动中投入时间，提升运营在团队内部和团队之间的重要性。

资源

相关文档：

- 衡量进度
- 构建控制面板以获取操作可见性

相关解决方案：

- 数据操作

OPS09-BP03 审查运营指标并确定改进优先顺序

留出专门的时间和资源来审查运营状况，可确保为日常业务提供服务始终是优先事项。召集运营主管和利益相关方，定期审查指标，重申或修改长期和短期目标，并确定改进的优先顺序。

期望的结果：

- 运营主管和员工定期开会，审查给定报告期内的指标。交流挑战，庆祝胜利，分享经验教训。
- 定期向利益相关方和业务领导通报运营状况，并征求他们对目标、KPI 和未来举措的意见。结合相关背景，讨论服务交付、运行和维护之间的权衡。

常见反模式：

- 推出了一款新产品，但一级和二级运营团队没有接受过充分的培训，无法为其提供支持，或者没有相应地增加人手。领导者看不到表明工单解决时间缩短和事件量增加的指标。几周后，心怀不满的用户离开平台，订阅数量开始下降，此时才采取行动。
- 对工作负载进行维护的手动流程已经存在很长时间。尽管人们一直渴望实现自动化，但考虑到该系统的重要性较低，并未得到足够的重视。然而，随着时间的推移，该系统的重要性与日俱增，现在这些手动过程耗费了运营团队的大部分时间。没有安排资源为运营团队提供更多工具，这导致随着工作量的增加，员工疲惫不堪。当有人报告员工离职去了其他竞争对手那里时，领导层才意识到这一点。

建立此最佳实践的好处： 在一些组织中，如何将同样的时间和精力用于提供新产品或新服务，可能是一项挑战。一旦出现这种情况，预期的服务水平会慢慢降低，业务线就会受到影响。这是因为运营团队没有随着业务的增长而做出改变和发展，很快就被甩在后面。如果不定期审查运营团队收集的见解，等到发现业务面临的风险时，可能为时已晚。通过花时间与运营人员和领导层一起审查指标和程序，运营团队所发挥的关键作用将始终可见，并且能在风险达到临界水平之前及早发现。运营团队可以更好地洞察即将发生的业务变化和举措，从而积极主动地开展工作。领导层对运营指标的了解展示了这些团队在客户满意度（包括内部和外部客户满意度）方面所发挥的作用，使他们能够更好地权衡选择的优先事项，或确保运营团队有足够的时间和资源随着新业务和工作负载计划的变化而做出改变和发展。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

花时间与利益相关方和运营团队一起审查运营指标，并审查报告数据。结合组织的长期和短期目标来审视这些报告，以确定是否实现了这些目标。在目标不明确的地方，或在要求的东西和给予的东西之间可能存在冲突的地方，找出含糊不清的根源。

确定时间、人员和工具可以在哪些方面推动实现运营成果。确定这将影响哪些 KPI 以及成功的目标应该是什么。定期重新审视，确保运营团队有足够的资源来支持业务线。

资源

相关文档：

- Amazon Athena
- Amazon CloudWatch 指标和维度参考
- Amazon QuickSight
- AWS Glue
- AWS Glue Data Catalog
- 使用 Amazon CloudWatch 代理从 Amazon EC2 实例和本地服务器收集指标和日志
- 使用 Amazon CloudWatch 指标

响应事件

您应该预测运营事件，包括计划内（例如，促销、部署和故障测试）和计划外（例如，利用率激增和组件故障）事件。在响应警报时，您应该使用现有的运行手册和行动手册来交付一致的结果。定义的警报应由负责响应和升级的角色或团队所有。您还需要了解系统组件的业务影响，并在需要时使用它来设定工作目标。您应该在事件发生后执行根本原因分析（RCA），然后防止故障再次发生或记录解决方法。

AWS 可以提供工具，为工作负载和运营即代码的方方面面提供支持，从而简化您的事件响应过程。借助这些工具，您可以编写对运营事件的响应脚本，并启动这些脚本来响应监控数据。

在 AWS 中，您可以将故障组件替换为已知良好的版本，而不是尝试修复它们，以此来缩短恢复时间。然后，您可以在带外对失败的资源进行分析。

最佳实践

- OPS10-BP01 使用流程来管理事件、意外事件和问题
- OPS10-BP02 针对每个警报设置一个流程
- OPS10-BP03 根据业务影响确定运维事件的优先顺序
- OPS10-BP04 定义上报路径
- OPS10-BP05 为影响服务的事件定义客户沟通计划
- OPS10-BP06 通过控制面板展现状况信息

- OPS10-BP07 自动响应事件

OPS10-BP01 使用流程来管理事件、意外事件和问题

要想保持工作负载的正常运行和高性能，对事件、意外事件和问题的高效管理能力非常关键。因此务必要认识和理解这些要素之间的不同，这样才能制定有效的响应和解决策略。针对各个方面确立明确定义的流程并按照流程操作，能够促使团队快速有效地应对出现的任何运维挑战。

期望的结果： 企业通过记录详实且集中存储的流程，高效地管理运维事件、意外事件和问题。这些流程会不断更新反映变化，并简化处理过程，从而保持出色的服务可靠性和工作负载性能。

常见反模式：

- 您被动而不是主动地响应事件。
- 面对不同类型的事件或意外事件，采取不一致的方法。
- 企业没有分析意外事件并从中吸取教训，以防止将来再次发生。

建立此最佳实践的好处：

- 简化响应流程并使之标准化。
- 降低意外事件对服务和客户的影响。
- 加快问题解决速度。
- 持续改进运维流程。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

实施这种最佳实践意味着您正在跟踪工作负载事件。您建立了处理意外事件和问题的流程。这些流程被记录下来、共享并经常更新。发现问题，确定优先级，并加以解决。

了解事件、意外事件和问题

- 事件: 一个 事件 可以是观测到的动作、发生的事情或状态的变化。事件可以是预先计划的，也可以是计划外的，可以源自工作负载内部，也可以源自外部。
- 意外事件： 意外事件 是需要响应的事件，例如计划外中断或服务质量下降。意外事件表示出现了需要立即采取行动才能恢复工作负载正常运行的中断。

- 问题： 问题 是一起或多起意外事件的根本原因。发现和解决问题需要对意外事件进行更深入的研究，以防止将来再次发生。

实施步骤

事件

1. 监控事件：
   - 实施可观测性 和 利用工作负载可观测性。
   - 监控用户、角色或 AWS 服务执行的操作，并作为事件记录在 AWS CloudTrail中。
   - 使用 Amazon EventBridge实时响应应用程序中的操作变化。
   - 使用 AWS Config持续评测、监控和记录资源配置变化。
2. 创建流程：
   - 制定一个流程来评测有哪些事件足够重要，需要进行监控。这包括为正常活动和异常活动设置阈值和参数。
   - 确定将事件升级为意外事件的标准。这些标准可以基于严重程度、对用户的影响或与预期行为的偏差。
   - 定期审查事件监控和响应流程。这包括分析过去的意外事件、调整阈值和完善警报机制。

意外事件

1. 响应意外事件：
   - 使用来自可观测性工具的洞察来快速识别和响应意外事件。
   - 实施 AWS Systems Manager OpsCenter 来汇总和整理运维项目及意外事件，并确定其优先级。
   - 使用 Amazon CloudWatch 和 AWS X-Ray 等服务来进行更深入的分析和故障排除。
   - 请考虑使用 AWS Managed Services（AMS） ，利用其主动式的预防和检测功能来增强事件管理。AMS 提供监控、意外事件检测和响应以及安全管理等服务，从而能够扩展对运维的支持。
   - Enterprise Support 客户可以使用 AWS 事件检测及响应服务，该服务针对生产工作负载，提供持续的主动监控和事件管理。
2. 创建事件管理流程：
   - 建立结构化的事件管理流程，包括明确的角色、沟通协定和解决步骤。
   - 将事件管理与 AWS Chatbot 等工具集成，用于实现高效的响应和协调。
   - 按严重性分类，并对每种类别预先定义 意外事件响应计划 。

3. 学习和改进：
   - 开展 意外事件后分析 ，用于了解根本原因和解决方法的有效性。
   - 根据审核结果和不断演变的做法，持续更新和改进响应计划。
   - 记录学到的经验教训，并在各个团队之间分享，以增强运维韧性。
   - Enterprise Support 客户可以向他们的技术客户经理请求参加 意外事件管理研讨会 。这场有指导意义的研讨会可测试您现有的意外事件响应计划，并帮助您找出需要改进之处。

问题

1. 确定问题：
   - 使用先前意外事件的数据来确定反复出现的模式，这些模式可能表明出现了更深层次的系统性问题。
   - 利用 AWS CloudTrail 和 Amazon CloudWatch 等工具，分析趋势并发现潜在问题。
   - 让运维、开发和业务部门等跨职能团队参与进来，从多元化的视角来审视根本原因。
2. 创建问题管理流程：
   - 制定结构化的问题管理流程，重点在于制定长期解决方案，而不是快速的权宜之计。
   - 采用根本原因分析（RCA，Root Cause Analysis）技术来调查和了解事件的根本原因。
   - 根据调查发现来更新运维策略、程序和基础设施，以防止问题再次发生。
3. 持续改进：
   - 培养持续学习和改进的文化，鼓励团队主动发现和解决潜在问题。
   - 定期审查和修订问题管理流程及工具，以适应业务和技术形势的不断变化。
   - 在整个企业内分享洞察和最佳实践，以建立更具韧性、更高效的运维环境。
4. 利用 AWS Support：
   - 使用 AWS Support 资源，例如 AWS Trusted Advisor，以主动获取指导和优化建议。
   - 在发生重大事件时，Enterprise Support 客户可以访问专业计划来获取支持，例如 AWS Countdown 。
   -

实施计划的工作量级别： 中

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标
- OPS04-BP02 实施应用程序遥测
- OPS07-BP03 使用运行手册执行程序
- OPS07-BP04 根据行动手册调查问题
- OPS08-BP01 分析工作负载指标
- OPS11-BP02 在意外事件发生后执行分析

相关文档：

- AWS 安全事件响应指南
- AWS 事件检测及响应服务
- AWS Cloud Adoption Framework：运维视角 – 意外事件和问题管理
- DevOps 和 SRE 时代的意外事件管理
- PagerDuty - 什么是意外事件管理？

相关视频：

- AWS 的常见意外事件响应提示
- AWS re:Invent 2022 – Amazon Builders' Library：25 年的 Amazon 卓越运营
- AWS re:Invent 2022 – AWS 事件检测及响应服务（SUP201）
- 正式推出 AWS Systems Manager 中的 Incident Manager

相关示例：

- AWS 主动式服务 – 意外事件管理研讨会
- 如何使用 PagerDuty 和 AWS Systems Manager Incident Manager 实现自动化意外事件响应
- 让意外事件响应人员参与到 AWS Systems Manager Incident Manager 中的随时待命方案
- 在 AWS Systems Manager Incident Manager 中提高意外事件处理过程中的可见性和协作能力
- AMS 中的意外事件报告和服务请求

相关服务：

- Amazon EventBridge

OPS10-BP02 针对每个警报设置一个流程

要想实现有效和高效的事件管理，为系统中的每个警报建立清晰明确的流程至关重要。这种做法可确保对每个警报都采取具体的、可操作的响应，从而提高运维的可靠性和响应能力。

期望的结果： 每个警报都会启动一个具体的、明确的响应计划。在可能的情况下，将响应过程自动化，并具有明确的负责人和上报路径。警报关联到最新的知识库，以便所有操作员都可以一致、高效地做出响应。响应速度快且全面统一，提高运维效率和可靠性。

常见反模式：

- 没有针对警报的预定义响应流程，导致不及时的权宜解决方案。
- 警报过载会导致遗漏重要的警报。
- 由于缺乏明确的责任人和责任关系，警报的处理不一致。

建立此最佳实践的好处：

- 仅发出需要采取操作的警报，缓解警报疲劳情况。
- 缩短了运维问题的平均解决时间（MTTR，Mean Time To Resolution）。
- 缩短了平均调查时间（MTTI，Mean Time To Investigate)，这有助于降低 MTTR。
- 增强了大范围运维响应的能力。
- 提高了处理运维事件的一致性和可靠性。

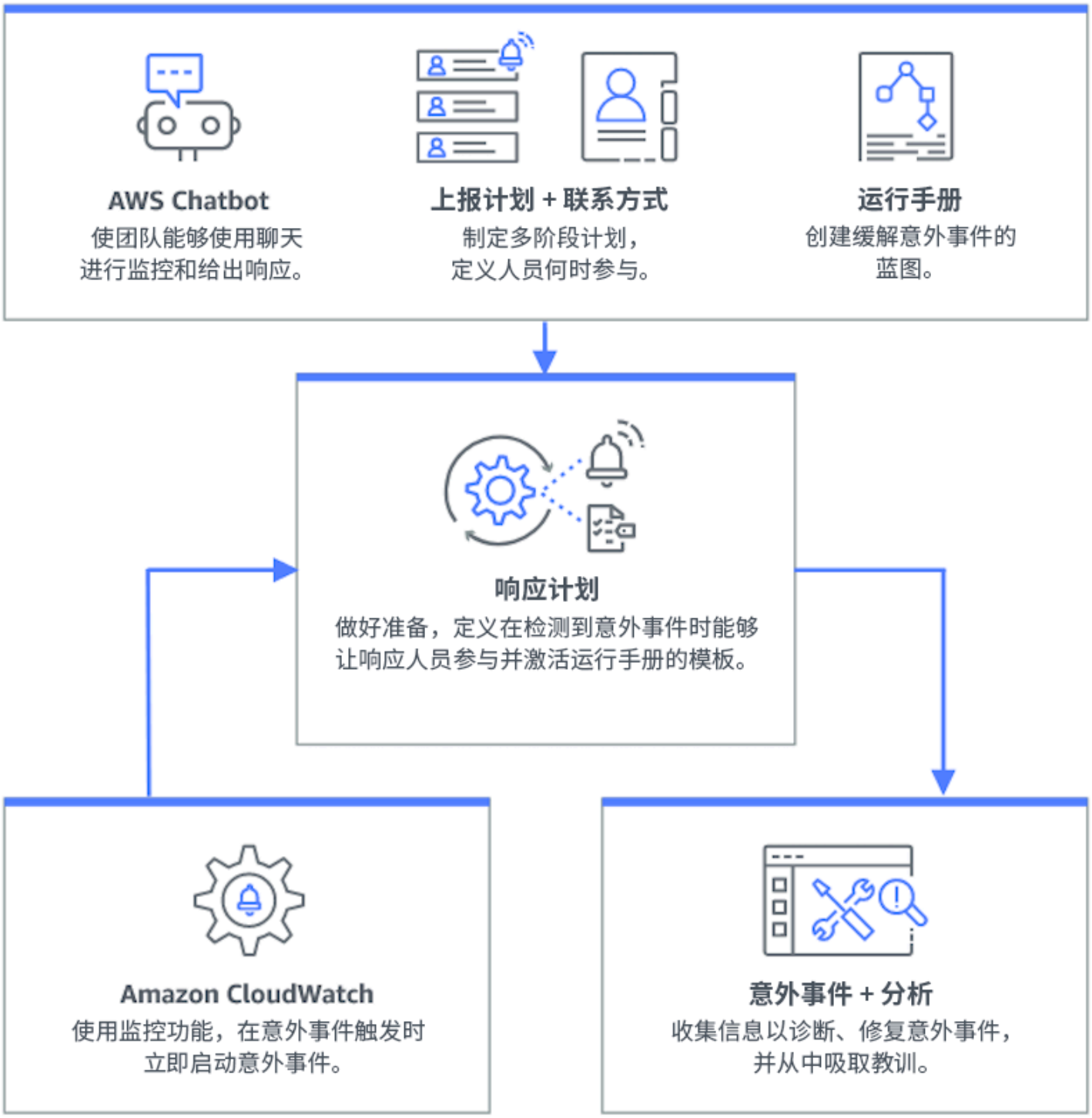
未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

针对每个警报设置一个流程，这包括为每个警报制定明确的响应计划，尽可能自动处理响应，并根据运维反馈和不断变化的需求不断完善这些流程。

实施步骤

下图说明了 AWS Systems Manager Incident Manager中的事件管理工作流。此服务设计为自动创建意外事件，用于快速响应来自 Amazon CloudWatch 或 Amazon EventBridge的特定事件，从而快速响应运维问题。创建意外事件时，无论是自动还是手动创建，Incident Manager 都会集中管理意外事件，整理相关的 AWS 资源信息，并启动预定义的响应计划。这包括运行 Systems Manager Automation 运行手册以立即采取行动，以及在 OpsCenter 中创建父运维工作项，用于跟踪相关任务和分析。这种简化的流程可以加快和协调整个 AWS 环境中的意外事件响应。



1. 使用复合警报： 创建 复合警报 （在 CloudWatch 中），用于对相关警报进行分组，从而减少噪音并实现更有意义的响应。
2. 将 Amazon CloudWatch 警报集成到 Incident Manager 配置 CloudWatch 警报，以在 AWS Systems Manager Incident Manager中自动创建意外事件。

3. 将 Amazon EventBridge 与 Incident Manager 集成： 创建 EventBridge 规则 ，以便使用已定义的响应计划来响应事件并创建意外事件。
4. 在 Incident Manager 中准备处理意外事件：
   - 针对每种警报类型，建立详细的 响应计划 （在 Incident Manager 中）。
   - 通过 AWS Chatbot 建立聊天频道，连接到 Incident Manager 中的响应计划，在发生意外事件时，协调在 Slack、Microsoft Teams 和 Amazon Chime 等各个平台之间的沟通。
   - 将 Systems Manager Automation 运行手册 纳入 Incident Manager 中，用于推动对意外事件的自动响应。

资源

相关最佳实践：

- OPS04-BP01 识别关键绩效指标
- OPS08-BP04 创建可操作的警报

相关文档：

- AWS Cloud Adoption Framework：运维视角 – 意外事件和问题管理
- 使用 Amazon CloudWatch 警报
- 设置 AWS Systems Manager Incident Manager
- 在 Incident Manager 中准备处理意外事件

相关视频：

- AWS 的常见意外事件响应提示

相关示例：

- AWS 研讨会 – AWS Systems Manager Incident Manager – 自动完成对安全事件的意外事件响应

OPS10-BP03 根据业务影响确定运维事件的优先顺序

及时响应运维事件至关重要，但并非所有事件都应该一概而论。当您根据业务影响确定优先顺序时，您同时确定了需要优先处理的、可能造成重大后果的意外事件，这些后果包括安全问题、财务损失、违反规章或声誉损害等。

期望的结果： 根据对业务运营和目标的潜在影响，对运维意外事件的响应顺序排列优先级。这使得应对措施既高效又有效。

常见反模式：

- 以同样的紧急程度处理所有事件，这会导致混乱，并且耽误解决关键问题。
- 您无法区分高影响力事件和低影响力事件，从而导致资源分配不当。
- 企业缺乏明确的优先级框架，导致对运维事件的响应不一致。
- 根据报告的顺序来确定事件的优先处理顺序，而不是根据对业务结果的影响。

建立此最佳实践的好处：

- 确保首先关注关键业务职能，从而尽可能减少潜在损失。
- 在同时发生多个事件时，可改善资源分配。
- 增强企业维护信任关系和满足监管要求的能力。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

面对多个运维事件时，基于影响力和紧迫性来确定优先次序的结构化方法至关重要。这种方法有助于作出明智的决策，将工作重心引导到最需要的地方，并降低影响业务连续性的风险。

实施步骤

1. 评测影响： 开发分类系统，根据事件对业务运营和目标的潜在影响来评估事件的严重性。以下示例展示了影响类别：

影响等级	描述
高	影响许多员工或客户，严重的财务影响，严重的声誉损害，或者造成人身伤害。

影响等级	描述
中	影响一群员工或客户，中度财务影响，或者中度声誉损害。
低	影响个别员工或客户，低财务影响，或者低声誉损害。

2. 评测紧迫性： 考虑安全、财务影响和服务水平协议（SLA）等因素，定义需要对某个事件进行响应的紧急程度。以下示例展示了紧急程度类别：

紧急程度	描述
高	损害呈指数级增长，影响到时间敏感型工作，需要立即上报，VIP 用户或群体受到影响。
中	损害会随着时间的推移而增加，或者单个 VIP 用户或群体受到影响。
低	边际损害会随着时间的推移而增加，或者影响到非时间敏感型工作。

3. 创建优先级矩阵：
   - 使用矩阵来交叉参考影响力和紧迫性，向不同的组合分配优先级。
   - 确保负责运维事件响应的所有团队成员都能访问并且理解矩阵。
   - 以下示例矩阵根据紧急程度和影响力显示意外事件的严重性：

紧迫性和影响力	高	中	低
高	严重	紧急	高
中	紧急	高	普通
低	高	普通	低

4. 培训和沟通： 培训响应团队，使其了解优先级矩阵以及在发生事件时遵循该矩阵的重要性。与所有利益相关方沟通优先次序流程，以设定明确的期望。
5. 与意外事件响应集成：

- 将优先级矩阵纳入意外事件响应计划和工具中。
- 尽可能自动对事件进行分类和排列优先级，以加快响应速度。
- Enterprise Support 客户可以利用 AWS 事件检测及响应服务，该服务针对生产工作负载，提供全天候的主动监控和事件管理。

6. 审查和调整： 定期审查优先次序流程的有效性，并根据反馈和业务环境的变化进行调整。

资源

相关最佳实践：

- OPS03-BP03 鼓励上报
- OPS08-BP04 创建可操作的警报
- OPS09-BP01 使用指标衡量运营目标和 KPI

相关文档：

- Atlassian – 了解意外事件严重性级别
- IT 流程图 – 意外事件优先级检查清单

OPS10-BP04 定义上报路径

在事件响应协定中确立明确的上报路径，以推动及时地采取有效措施。这包括指定上报提示、详细说明上报流程，以及预先批准相关措施，以便加快决策速度并缩短平均解决时间（MTTR，Mean Time To Resolution）。

期望的结果： 结构化的高效流程，可将意外事件上报给相应的人员，从而尽可能减少响应时间和影响。

常见反模式：

- 恢复程序不明确，导致在发生重大意外事件时采取权宜之计。
- 没有明确的权限和负责人，导致在需要采取紧急措施时出现延误。
- 发送给利益相关方和客户的通知不符合他们的预期。
- 重要的决策被推迟。

建立此最佳实践的好处：

- 通过预定义的上报程序简化意外事件响应。
- 通过预先批准的措施并明确负责人来减少停机时间。
- 根据意外事件严重程度，改进资源分配和支持级别调整。
- 改善与利益相关方和客户的沟通。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

妥善定义的上报路径对于快速响应意外事件至关重要。AWS Systems Manager Incident Manager 支持设置结构化上报计划和随时待命方案，这可以在发生意外事件时提醒相关人员，让他们准备好采取行动。

实施步骤

1. 设置上报提示： 设置 CloudWatch 警报 以在 AWS Systems Manager Incident Manager中创建意外事件。
2. 设置随时待命方案： 创建 随时待命方案 （在 Incident Manager 中），与您的上报路径相对齐。为随时待命人员提供必要的权限和工具，以便迅速采取行动。
3. 详细的上报程序：
   - 确定上报意外事件的具体条件。
   - 创建 上报计划 （在 Incident Manager 中）。
   - 上报渠道应包括联系人或随时待命方案。
   - 定义团队在每个上报级别的角色和职责。
4. 预先批准的缓解措施： 与决策者合作，预先批准针对预期情景的措施。使用 Systems Manager Automation 运行手册 （与 Incident Manager 集成）以加快意外事件解决。
5. 指定负责人： 清楚地确定上报路径中每个环节的内部负责人。
6. 详细说明第三方上报情况：
   - 记录第三方服务水平协议（SLA），并与内部目标保持一致。
   - 针对发生意外事件时的供应商沟通，设定明确的协定。
   - 将供应商联系方式集成到事件管理工具中以便直接访问。
   - 定期开展演习，包括第三方响应场景。
   - 确保详细记录了供应商上报信息并且易于访问。

7. 针对上报计划进行培训和演习： 培训团队了解上报流程，并定期进行事件响应演习或实际演练。Enterprise Support 客户可以申请 意外事件管理研讨会。
8. 持续改进： 定期检查上报路径的有效性。根据从意外事件后分析中吸取的经验教训和持续的反馈来更新您的流程。

实施计划的工作量级别： 适中

资源

相关最佳实践：

- OPS08-BP04 创建可操作的警报
- OPS10-BP02 针对每个警报设置一个流程
- OPS11-BP02 在意外事件发生后执行分析

相关文档：

- AWS Systems Manager Incident Manager 上报计划
- 在 Incident Manager 中处理随时待命方案
- 创建和管理运行手册
- 使用 AWS IAM Identity Center 临时提升访问权限的管理
- Atlassian – 有效事件管理的上报政策

OPS10-BP05 为影响服务的事件定义客户沟通计划

在发生影响服务的事件时，对于维护客户的信任和开诚布公地交流，有效的沟通至关重要。利用明确定义的沟通计划，企业在遇到意外事件时，可以快速、清晰地在内部和外部分享信息。

期望的结果：

- 可靠的沟通计划，可在发生了影响服务的事件时，高效地通知客户和利益相关方。
- 开诚布公的交流可以建立信任关系，减少客户焦虑。
- 尽可能减少影响服务的事件对客户体验和业务运营的影响。

常见反模式：

- 不能充分或及时地进行沟通，导致客户困惑和不满。
- 过于技术性或含糊不清的信息，无法传达对用户的实际影响。
- 没有预定义的沟通策略，导致被动地传达消息，且不能确保一致性。

建立此最佳实践的好处：

- 通过主动、清晰的沟通，增强客户的信任和满意度。
- 通过先行解决客户的问题，减轻支持团队的负担。
- 提高了高效管理意外事件和从中恢复的能力。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

针对影响服务的事件，制定全面的沟通计划，这涉及从选择合适的渠道到精心撰写消息和使用合适的语气等多个方面。该计划应具有适应性、可扩展性，并能根据不同的停机场景进行调整。

实施步骤

1. 定义角色和职责：
   - 指派一名重大意外事件经理，负责监管意外事件响应活动。
   - 指定一名沟通经理，负责协调所有内外部沟通。
   - 让支持经理参与进来，通过支持工单提供一致的沟通。
2. 确定沟通渠道： 选择工作聊天工具、电子邮件、短信、社交媒体、应用程序内通知和状态页面等渠道。这些渠道应具有韧性，能够在发生影响服务的事件期间独立运转。
3. 快速、清晰地与客户定期沟通：
   - 针对各种服务受损场景开发模板，注重简化性和关键细节。提供有关服务受损、预期解决时间和影响等信息。
   - 使用 Amazon Pinpoint，通过推送通知、应用程序内通知、电子邮件、短信、语音消息以及自定义渠道消息，来向客户发送提醒。
   - 使用 Amazon Simple Notification Service（Amazon SNS），以编程方式或通过电子邮件、移动推送通知和短信提醒订阅用户。
   - 通过公开分享 Amazon CloudWatch 控制面板，使用控制面板展现状况信息。
   - 鼓励社交媒体互动：
     - 积极监控社交媒体，了解客户情绪。

- 在社交媒体平台上发布内容以面向公众提供最新信息，并参与社区互动。
- 编制模板以实现一致、清晰的社交媒体沟通。

4. 协调内部沟通： 实施内部协定，使用 AWS Chatbot 等工具进行团队协调和沟通。使用 CloudWatch 控制面板展现状况信息。
5. 使用专用工具和服务来编排沟通：
   - 将 AWS Systems Manager Incident Manager 与 AWS Chatbot 结合使用来设置专用的聊天频道，以便在发生意外事件时，用于实时的内部沟通和协调。
   - 发生意外事件时，使用 AWS Systems Manager Incident Manager 运行手册，通过 Amazon Pinpoint、Amazon SNS 或社交媒体平台等第三方工具，自动通知客户。
   - 将审批工作流整合到运行手册中，以便在所有外部通信渠道发送信息之前，进行审核和授权（如需要）。
6. 练习和改进：
   - 开展有关使用沟通工具和策略的培训。增强团队能力，以便在发生意外事件时及时作出决策。
   - 通过定期演习或实际演练来测试沟通计划。使用这些测试来完善消息传递并评估渠道的有效性。
   - 实施反馈机制来评测发生意外事件时的沟通有效性。根据反馈和不断变化的需求，不断升级沟通计划。

实施计划的工作量级别： 高

资源

相关最佳实践：

- OPS07-BP03 使用运行手册执行程序
- OPS10-BP06 通过控制面板展现状况信息
- OPS11-BP02 在意外事件发生后执行分析

相关文档：

- Atlassian – 意外事件沟通最佳实践
- Atlassian – 如何撰写出色的状态更新
- PagerDuty – 意外事件沟通指南

相关视频：

- Atlassian – 制定自己的意外事件沟通计划：意外事件模板

相关示例：

- AWS Health 控制面板
- AWS 状态更新示例

OPS10-BP06 通过控制面板展现状况信息

使用控制面板作为战略工具，面向内部技术团队、领导层和客户等不同受众，实时展现运维状态和关键指标。这些控制面板集中直观地展现系统运行状况和业务性能，提高了透明度和决策效率。

期望的结果：

- 您的控制面板可面对不同利益相关方，提供与之相关的系统和业务指标的全面视图。
- 利益相关方可以主动访问运维信息，这样就无需频繁地请求查看状态。
- 增强了正常操作和发生意外事件期间的实时决策能力。

常见反模式：

- 工程师加入事件管理通话，需要了解状态更新才能跟得上节奏。
- 依赖人工报告进行管理，这会导致延迟，并可能导致不准确。
- 在意外事件发生时，运维团队经常被状态更新打断。

建立此最佳实践的好处：

- 使利益相关方能够立即获得关键信息，推动作出明智的决策。
- 尽可能减少人工报告和频繁的状态查询，减少运维效率低下的问题。
- 能够实时了解系统性能和业务指标，提高了透明度和信任度。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

控制面板可以高效地展现系统状态和业务指标，并且可以根据不同受众群体的需求进行定制。利用 Amazon CloudWatch 控制面板和 Amazon QuickSight 等工具，您可以创建交互式的实时控制面板，用于系统监控和商业智能。

实施步骤

1. 确定利益相关方的需求： 确定技术团队、领导层和客户等不同受众群体的特定信息需求。
2. 选择合适的工具： 选择合适的工具，例如 Amazon CloudWatch 控制面板 用于系统监控， Amazon QuickSight 用于交互式商业智能。
3. 设计高效的控制面板：
   - 设计控制面板，以清晰地显示相关指标和 KPI，确保这些指标易于理解且可操作。
   - 根据需要，纳入系统级和业务级视图。
   - 包括高层控制面板（用于整体概述）和底层控制面板（用于详细分析）。
   - 在控制面板中集成自动警报以突出显示关键问题。
   - 在控制面板中添加重要指标阈值和目标等注释，以方便即时查看。
4. 集成数据来源：
   - 使用 Amazon CloudWatch 汇总和显示来自各种 AWS 服务的指标，以及 查询源自其它数据来源的指标，用于创建系统运行状况和业务指标的整合视图。
   - 使用 CloudWatch Logs Insights 等功能，查询和可视化来自不同应用程序和服务的日志数据。
5. 提供自助访问：
   - 与利益相关方分享 CloudWatch 控制面板，提供自助的信息访问，这会用到 控制面板共享功能。
   - 确保控制面板易于访问，并可实时提供最新信息。
6. 定期更新和完善：
   - 不断更新和完善控制面板，以适应不断演变的业务需求，与利益相关方的反馈相对齐。
   - 定期审查控制面板，确保其信息贴近用户需求，并可以有效地传达必要信息。

资源

相关最佳实践：

- OPS08-BP05 创建控制面板

相关文档：

- 构建控制面板以获取运维可见性
- 使用 Amazon CloudWatch 控制面板
- 使用控制面板变量创建灵活的控制面板
- 共享 CloudWatch 控制面板
- 查询源自其它数据来源的指标
- 向 CloudWatch 控制面板添加自定义小部件

相关示例：

- One Observability Workshop – 控制面板

OPS10-BP07 自动响应事件

要想实现快速、一致和无错误的运维处理，对事件进行自动响应是关键所在。创建简化的流程，使用多种工具来自动管理和响应事件，尽可能减少人工干预并提高运维效率。

期望的结果：

- 利用自动化功能，减少人为错误并缩短解决问题的用时。
- 一致且可靠的运维事件处理。
- 提高运维效率和系统可靠性。

常见反模式：

- 手动处理事件，容易导致延误和出错。
- 忽视了自动化功能在重复性关键任务中的作用。
- 反复地手动执行任务，导致丧失了对警报的警惕性，可能会遗漏关键问题。

建立此最佳实践的好处：

- 加快响应事件的速度，减少系统停机时间。
- 通过自动化和一致的事件处理，实现可靠的运维。

未建立这种最佳实践的情况下暴露的风险等级： 中

实施指导

纳入自动化功能，用以创建高效的运维工作流，并尽可能减少人工干预。

实施步骤

1. 发现自动化机会： 确定可以自动处理的重复性任务，例如问题修复、工单信息补充、容量管理、扩展、部署和测试。
2. 发现自动化提示：
   - 评测和定义特定的条件或指标，以便能够通过使用 Amazon CloudWatch 警报操作启动自动化响应。
   - 使用 Amazon EventBridge 来响应 AWS 服务、自定义工作负载和 SaaS 应用程序中的事件。
   - 考虑启动事件，例如 特定日志条目、性能指标阈值或 有关 AWS 资源中的状态更改。
3. 实现事件驱动型自动化：
   - 使用 AWS Systems Manager Automation 运行手册来简化维护、部署和修复任务。
   - 在 Incident Manager 中创建事件 会自动收集所涉及 AWS 资源的详细信息，并将这些信息添加到事件。
   - 使用 适用于 AWS 的配额监控程序来主动监控配额。
   - 使用 AWS Auto Scaling 自动调整容量来保持可用性和性能。
   - 使用 Amazon CodeCatalyst自动处理开发管道。
   - 进行烟雾测试，或者 使用合成监控持续监控端点和 API。
4. 通过自动化功能执行风险缓解：
   - 实施 自动化安全响应 来快速应对风险。
   - 使用 AWS Systems Manager 状态管理器 来减少配置偏差。
   - 使用 AWS Config 规则 修复不合规的资源。

实施计划的工作量级别： 高

资源

相关最佳实践：

- OPS08-BP04 创建可操作的警报

- OPS10-BP02 针对每个警报设置一个流程

相关文档：

- 将 Systems Manager Automation 运行手册与 Incident Manager 配合使用
- 在 Incident Manager 中创建事件
- AWS 服务配额
- 监控资源使用情况，并在快要达到配额时发送通知
- AWS Auto Scaling
- 什么是 Amazon CodeCatalyst？
- 使用 Amazon CloudWatch 警报
- 使用 Amazon CloudWatch 警报操作
- 使用 AWS Config 规则 修正不合规资源
- 使用筛选条件根据日志事件创建指标
- AWS Systems Manager 状态管理器

相关视频：

- 使用 AWS Systems Manager 创建 Automation 运行手册
- 如何在 AWS 上实现 IT 运维自动化
- AWS Security Hub 自动化规则
- 使用 Amazon CodeCatalyst 蓝图快速启动软件项目

相关示例：

- Amazon CodeCatalyst 教程：使用现代化三层 Web 应用程序蓝图创建项目
- One Observability Workshop
- 使用 Incident Manager 来响应事件

演进

发展是指在一段时间内持续的改进周期。根据从您的运营活动中吸取的经验教训，实施频繁的小幅度增量变更，并评估其在带来改进方面的成效。

要持续改进您的运营，您必须能够：

主题

学习、分享和改进

要定期提供时间进行运营活动分析、故障分析、试验和改进，这一点很重要。如果事情失败，您需要确保团队和大型工程社区从能这些失败中学习。您应该进行失败分析，以获取经验教训并计划改进。您需要定期与其他团队一起查看学习到的经验教训，以验证您的见解。

最佳实践

OPS11-BP01 设置持续改进流程

根据内部和外部架构最佳实践评估您的工作负载。经常开展目标明确的工作负载审查工作。将改进机会优先纳入您的软件开发周期。

期望结果：

- 您经常根据架构最佳实践来分析工作负载。
- 在软件开发过程中，您同等重视性能改进机会。

常见反面模式：

- 自从几年前部署工作负载以来，您没有对其进行过架构审查。
- 您将改进机会放在较低的优先级。相比新功能的开发，这些改进机会仍积压在待办事项中。
- 不存在对组织的最佳实践实施修改的标准。

建立此最佳实践的好处：

- 您的工作负载符合最新的架构最佳实践。
- 您按照明确的目的来改进工作负载。
- 您可以利用组织的最佳实践来改善所有工作负载。
- 您获得的边际收益所带来的影响会不断累积，从而推动效率的提升。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

经常对工作负载进行架构审查。利用内部和外部最佳实践，评估您的工作负载并确定改进机会。将改进机会优先纳入您的软件开发周期。

实施步骤

1. 按照议定的频率，对您的生产工作负载进行定期架构审查。使用记录在册的架构标准，包括 AWS 特定的最佳实践。
   a. 使用您内部定义的标准完成这些审查工作。如果没有内部标准，请使用 AWS Well-Architected Framework。
   b. 使用 AWS Well-Architected Tool 来创建自己的内部最佳实践的自定义剖析，并进行架构审查。
   c. 联系您的 AWS 解决方案架构师或技术客户经理，在他们的指导下，对您的工作负载进行 Well-Architected Framework 审查。
2. 将审查中发现的改进机会优先纳入您的软件开发过程。

实施计划的工作量级别：低。可以使用 AWS Well-Architected Framework 执行年度架构审核。

资源

相关最佳实践：

- OPS11-BP02 在意外事件发生后执行分析
- OPS11-BP08 记录和分享经验教训
- OPS04 实施可观测性

相关文档：

- AWS Well-Architected Tool - Custom lenses
- AWS Well-Architected Whitepaper - The review process
- Customize Well-Architected Reviews using Custom Lenses and the AWS Well-Architected Tool
- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization

相关视频：

- Well-Architected Labs - Level 100: Custom Lenses on AWS Well-Architected Tool
- AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization

相关示例：

- AWS Well-Architected Tool

OPS11-BP02 在意外事件发生后执行分析

审核影响客户的事件，确定导致这些事件的因素和预防措施。利用这些信息来制定缓解措施，以限制或防止再次发生同类事件。制定程序以迅速有效地做出响应。根据目标受众，适当传达事件成因和纠正措施。

期望结果：

- 您已经建立了包括意外事件后分析在内的事件管理流程。
- 您已经制定了可观测性计划来收集事件数据。
- 利用这些数据，您可以了解并收集指标，用于支持意外事件后分析流程。
- 您可以从意外事件中吸取教训，以便改善以后的结果。

常见反面模式：

- 您管理应用程序服务器。大约每 23 小时 55 分钟，所有活动会话都会终止。您已尝试找出应用程序服务器上出现的问题。您怀疑可能是网络问题，但由于网络团队工作繁忙无法为您提供支持，因此无法与他们合作。由于缺乏可遵循的预定义流程，因此难以获取支持并收集必要的信息来确定发生了什么情况。
- 您的工作负载中出现了数据丢失的情况。这是第一次发生，原因不明。您认为它不重要，因为可以重新创建数据。数据丢失对客户的影响开始变得愈发频繁。还原丢失的数据时，这也会增加您的运维负担。

建立此最佳实践的好处：

- 您建立了预定义的流程，以确定导致意外事件发生的要素、条件、操作和事件，这可以推动您找到改进机会。
- 您可以使用来自意外事件后分析的数据进行改进。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

通过流程来确定事件成因。审查所有影响客户的意外事件。设置流程来确定和记录导致意外事件的因素，以便制定缓解措施来限制或防止事件再次发生，并且您还可以据此制定及时有效的应对措施。酌情传达造成意外事件的根本原因，并针对目标受众量身定制传达内容。在企业内开放地分享经验教训。

实施步骤

1. 收集各种指标，例如部署更改、配置更改、意外事件开始时间、警报时间、参与时间、缓解措施开始时间和意外事件解决时间。
2. 在时间表上描述关键时间点，用于了解意外事件。
3. 提出以下问题：
    a. 能否缩短检测时间？
    b. 对指标和警报进行哪些更新可以更快地发现意外事件？
    c. 能否缩短诊断时间？
    d. 对您的响应计划或上报计划进行哪些更新可以更快地让需要的响应人员参与进来？
    e. 能否缩短缓解时间？
    f. 可以添加或改进哪些运行手册或行动手册步骤？

g. 未来能否防止意外事件再次发生？

4. 创建检查清单和操作。跟踪并交付所有操作。

实施计划的工作量级别：中等

资源

相关最佳实践：

- OPS11-BP01 设置持续改进流程
- OPS 4 - 实施可观测性

相关文档：

- 在 Incident Manager 中执行意外事件后分析
- 运维准备情况审查

OPS11-BP03 实施反馈环路

反馈环路提供了可操作的见解，进而推动决策的制定。将反馈环路融入过程和工作负载中。这可帮助您确定问题和需要改进的领域。它们还可以验证在改进方面所做的投入。这些反馈环路为持续改进工作负载奠定了基础。

反馈环路分为两大类： 即时反馈 和 回顾性分析。通过审查运营活动的绩效和成果来收集即时反馈。此反馈来自团队成员、客户或活动的自动化输出。通过 A/B 测试和发布新功能等方式接收即时反馈，这对于快速失效机制至关重要。

定期执行回顾性分析，可以获得在运营成果审核和指标审核过程中产生的反馈。这些回顾在冲刺结束时进行、有节奏地进行或者在重大发布或事件之后进行。这种类型的反馈环路验证了在运营或工作负载方面的投入。它有助于衡量成功并验证您的策略。

期望的结果： 您可以使用即时反馈和回顾性分析来加快改进。有一种机制可用于捕获用户和团队成员的反馈。回顾性分析用于确定可推动改进的趋势。

常见反模式：

- 您推出了一项新功能，但无法接收客户对此新功能的反馈。
- 在投资进行运营改进后，您无需回顾来验证它们。

- 您可以收集客户反馈，但不用定期进行审查。
- 反馈环路会产生建议的操作项，但它们不包括在软件开发过程中。
- 对于所提出的改进事项，客户不会收到关于它们的反馈意见。

建立此最佳实践的好处：

- 您可以反过来从客户出发，以便推动新的功能。
- 您的组织文化能够更快地对变更做出回应。
- 趋势用于确定改进机会。
- 回顾将验证对工作负载和运营所做的投入。

未建立这种最佳实践的情况下暴露的风险等级： 高

实施指导

实施此最佳实践意味着同时使用即时反馈和回顾性分析。这些反馈环路将推动改进。有许多适用于即时反馈的机制，包括调查、客户投票或反馈表。您的组织还使用回顾来确定改进机会并验证计划。

客户示例

AnyCompany Retail 创建了一个 Web 表单，客户可使用此表单提供反馈或报告问题。在每周 Scrum 期间，软件开发团队将评估用户反馈。反馈定期用于引导相应平台的发展。他们在每个冲刺结束时进行回顾，确定需要改进的项目。

实施步骤

1. 即时反馈
    - 您需要一种机制来接收由客户和团队成员提供的反馈，也可以将您的运营活动配置为交付自动反馈。
    - 您的组织需要一个流程，来审查此反馈、确定要改进的方面并安排改进。
    - 必须将反馈纳入您的软件开发过程中。
    - 在实施改进时，请对反馈提交者进行跟进。
        - 您可以使用 AWS Systems Manager OpsCenter 将这些改进创建为 OpsItem 并进行跟踪。
2. 回顾性分析
    - 在开发周期结束时、按设定的节奏或在主要发布后进行回顾。

- 召开回顾性会议，让工作负载中涉及的利益相关者参加。
- 在白板或电子表格上创建三个列：“停止”、“开始”和“继续”。
  - 停止 针对的是您希望团队停止执行的任何工作。
  - 开始 针对的是要开始付诸行动的想法。
  - 继续 针对的是要继续执行的项目。
- 在会议室里四处走动，从利益相关者那里收集反馈。
- 确定反馈的优先级。将操作和利益相关者分配给任何“开始”或“继续”项目。
- 将操作纳入软件开发过程中，并在实施改进时将状态更新传达给利益相关者。

实施计划的工作量级别： 中。要实施此最佳实践，您需要一种方法来获取并分析即时反馈。此外，您需要建立一个回顾性分析过程。

资源

相关最佳实践：

- OPS01-BP01 评估客户需求：反馈环路是一种用于收集外部客户需求的机制。
- OPS01-BP02 评估内部客户需求：内部利益相关者可以使用反馈环路来传达需求和要求。
- OPS11-BP02 在意外事件发生后执行分析：事件后分析是发生事件后进行回顾性分析的重要形式。
- OPS11-BP07 审核运营指标：运营指标审查可确定趋势和需要改进的方面。

相关文档：

- 构建 CCOE 时应避免的 7 个陷阱
- Atlassian 团队行动手册 – 回顾
- 电子邮件定义：反馈环路
- 建立基于 AWS Well-Architected Framework 审查的反馈环路
- IBM Garage 方法 – 保持回顾
- Investopedia – PDCS 循环
- Tim Cochran 所著的《最大限度地提高开发人员效率》
- 运营准备情况审查（ORR）白皮书 – 迭代
- TIL CSI – 持续服务改进
- 当丰田转向电子商务：Amazon 的精益方法

相关视频：

- 构建有效的客户反馈环路

相关示例：

- Astuto – 客户反馈开源工具
- AWS 解决方案 – AWS 上的 QnABot
- Fider – 客户反馈整理平台

相关服务：

- AWS Systems Manager OpsCenter

OPS11-BP04 执行知识管理

知识管理帮助团队成员找到完成他们的工作所需的信息。在学习型组织中，自由分享信息，从而增强个人的能力。可以发现和搜索信息。信息准确且保持最新。制定有创建新信息、更新现有信息和归档过时信息的机制。知识管理平台的最常见例子是像 Wiki 这样的内容管理系统。

期望结果：

- 团队成员可以及时获取准确的信息。
- 信息可搜索。
- 制定有添加、更新和归档信息的机制。

常见反模式：

- 没有集中式知识存储。团队成员在他们的本地计算机上管理自己的笔记。
- 您有自托管的 Wiki，但没有制定机制来管理信息，导致信息过时。
- 有人识别出缺失的信息，但没有制定流程来请求将其添加到团队 Wiki 中。他们自己添加信息，但他们错过了一个关键步骤，导致发生中断。

建立此最佳实践的好处：

- 因为可以自由分享信息，所以增强了团队成员的能力。

- 因为文档保持最新且可搜索，新团队成员可以更快上手。
- 信息及时、准确和富有实用价值。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

知识管理是学习型组织的一个重要方面。首先，您需要一个中央存储库来存储您的知识（一个常见的例子是自托管 Wiki）。您必须制定用于添加、更新和归档知识的流程。为应该记录的内容制定标准，并让每一个人都能作出贡献。

客户示例

AnyCompany Retail 托管一个内部 Wiki，其中存储了他们的所有知识。公司鼓励团队成员在履行日常职责时向知识库中添加内容。跨职能团队每季度评估哪些页面更新最少，并确定这些页面是否需要归档或更新。

实施步骤

1. 首先确定用于存储知识的内容管理系统。获得整个组织的利益攸关方的同意。
   a. 如果您还没有内容管理系统，则在刚开始的时候请考虑运行自托管的 Wiki，或使用版本控制存储库。
2. 编制用于添加、更新和归档信息的运行手册。就这些流程对团队进行培训。
3. 确定应该在内容管理系统中存储哪些知识。从团队成员执行的日常活动（运行手册和行动手册）开始。与利益攸关方一起对增加的知识进行优先级排序。
4. 定期与利益攸关方一起识别过时的信息并将其归档或更新。

实施计划的工作量级别：中等。如果您还没有内容管理系统，则可以设置自托管的 Wiki 或版本控制文档库。

资源

相关最佳实践：

- OPS11-BP08 记录和分享经验教训 - 知识管理促进有关经验教训的信息共享。

相关文档：

- Atlassian - 知识管理

相关示例：

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 确定推动改进的因素

确定推动改进的因素，帮助您根据数据和反馈环路来评估机会并进行优先级排序。探索系统和流程中的改进机会，并根据具体情况相应采取自动化功能。

期望结果：

- 您跟踪整个环境中的数据。
- 您将事件和活动与业务成果相关联。
- 您可以在环境和系统之间进行比较和对比。
- 您保留部署和成果的详细活动历史记录。
- 您收集数据来支持您的安全态势。

常见反面模式：

- 您从整个环境中收集数据，但没有关联事件和活动。
- 您收集所有资产的详细数据，而这导致 Amazon CloudWatch 和 AWS CloudTrail 的活动及成本增加。但是，您并没有让这些数据发挥出作用。
- 在确定推动改进的因素时，您没有考虑业务成果。
- 您没有衡量新功能的效果。

建立此最佳实践的好处：

- 您可以确定用于改进的标准，从而尽可能减小基于事件的动机或情感投入所带来的影响。
- 您可以响应业务事件，而不仅仅是技术事件。
- 您可以衡量自己的环境来确定需要改进的领域。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

- 了解推动改进的因素：您只应该在能够实现所需成果的情况下更改某个系统。
  - 需要的功能：在评估改进机会时评估需要的特性和功能。
    - AWS 的新功能
  - 无法接受的问题：在评估改进机会时，评估无法接受的问题、错误和漏洞。跟踪合理调整大小选项，寻找优化机会。
    - AWS 最新安全公告
    - AWS Trusted Advisor
    - Cloud Intelligence 控制面板
  - 合规性要求：在分析改进机会时，评估为了保持监管和政策合规性，或获取第三方支持，而所需的更新和更改。
    - AWS 合规
    - AWS 合规性计划
    - AWS 合规性最新消息

资源

相关最佳实践：

- OPS01 组织重点
- OPS02 关系和所有权
- OPS04-BP01 识别关键绩效指标
- OPS08 利用工作负载可观测性
- OPS09 了解运营状况
- OPS11-BP03 实施反馈环路

相关文档：

- Amazon Athena
- Amazon QuickSight
- AWS 合规

- AWS 合规性最新消息
- AWS 合规性计划
- AWS Glue
- AWS 最新安全公告
- AWS Trusted Advisor
- Export your log data to Amazon S3
- AWS 的新功能
- 以客户为中心的创新势在必行
- Digital Transformation: Hype or a Strategic Necessity?

相关视频

- AWS re:Invent 2023 - Improve operational efficiency and resilience with AWS Support (SUP310)

OPS11-BP06 验证分析结果

与跨职能团队和业务负责人共同查看分析结果和响应措施。通过这些工作来建立共识、发现其他影响并确定行动方案。适当调整响应措施。

期望结果：

- 您与业务负责人一起定期审查分析结果。业务负责人为新获得的分析结果提供更多背景信息。
- 您审查分析结果并让技术同事提供反馈，然后在团队之间分享您学到的经验教训。
- 您发布数据和分析结果，让其他技术团队和业务团队审查。您将学到的经验教训融入到其他部门的新实践中。
- 与高层领导一起总结和审查新分析结果。高层领导使用新的分析结果来定义战略。

常见反面模式：

- 您发布新功能。此功能改变了一些客户的行为。您的可观测性没有考虑到这些变化。您无法量化这些变化带来的益处。
- 您推送新的更新，却忽略了刷新您的 CDN。CDN 缓存不再与最新版本兼容。您衡量了出错请求的百分比。您的所有用户在与后端服务器通信时，都报告了 HTTP 400 错误。您调查客户端出现的错误，发现是因为您衡量了错误的维度，时间就这样白白浪费了。

- 您的服务级别协议规定正常运行时间为 99.9%，您的恢复点目标是 4 小时。服务负责人坚持认为系统应该是零停机时间。您实施昂贵而复杂的复制解决方案，浪费了时间和金钱。

建立此最佳实践的好处：

- 当您与业务负责人和主题专家一起验证分析结果时，就可以建立共识并更有效地指导改进。
- 您会发现隐藏的问题，并在未来的决策中处理这些问题。
- 您的重心从技术成果转移到业务成果。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

- 验证分析结果：与业务负责人和主题专家沟通，以确保对您所收集数据的价值达成共识和一致。确定其他问题、潜在影响并制定行动方案。

资源

相关最佳实践：

- OPS01-BP06 在管理效益和风险的同时评估各种权衡因素
- OPS02-BP06 预先定义或协商团队间的职责
- OPS11-BP03 实施反馈环路

相关文档：

- Designing a Cloud Center of Excellence (CCOE)

相关视频：

- Building observability to increase resiliency

OPS11-BP07 审核运营指标

定期与来自不同业务领域的跨团队参与者对运营指标进行回顾性分析。通过这些分析来确定改进机会和可能的行动方案，并分享经验教训。寻找在所有环境（例如，开发、测试和生产环境）中改进的机会。

期望结果：

- 您经常审核影响业务的指标
- 您通过可观测性功能来检测和审查异常
- 您使用数据来支持实现业务成果和目标

常见反面模式：

- 您的维护窗口导致一次重要的零售促销活动中断。如果存在其他影响业务的事件，可以延迟标准维护时段，而业务部门对此并不知晓。
- 由于您在组织中广泛使用了过时的库，导致长时间停机。此后，您迁移到受支持的库。您组织中的其他团队尚未意识到风险的存在。
- 您没有定期审查客户 SLA 的达成情况。您目前正趋向于无法满足客户 SLA。如果无法满足客户 SLA，将会受到经济处罚。

建立此最佳实践的好处：

- 如果您能够定期开会审查运营指标、事件和意外事件，就可以在团队之间达成共识。
- 团队定期会面来审查指标和意外事件，这让您可以很好地针对风险采取行动并实现客户 SLA。
- 您可以分享学到的经验教训，这样就能提供数据，根据业务成果确定优先顺序和有针对性的改进。

在未建立这种最佳实践的情况下暴露的风险等级：中等

实施指导

- 定期与来自不同业务领域的跨团队参与者对运营指标进行回顾性分析。
- 与包括业务、开发和运营团队在内的利益相关方交流，共同分析通过即时反馈和回顾性分析得到的调查发现，并分享经验教训。
- 根据他们的见解来确定改进机会和可能的行动方案。

资源

相关最佳实践：

- OPS08-BP05 创建控制面板
- OPS09-BP03 审查运营指标并确定改进优先顺序

- OPS10-BP01 使用流程来管理事件、意外事件和问题

相关文档：

- Amazon CloudWatch
- Amazon CloudWatch 指标和维度参考
- 发布自定义指标
- 使用 Amazon CloudWatch 指标
- Dashboards and visualizations with CloudWatch

OPS11-BP08 记录和分享经验教训

记录和分享在运营活动中获得的经验教训，以便在内部和不同团队中利用。您应该分享团队学到的经验教训，以增加整个组织的效益。分享信息和资源来防止可避免的错误，简化开发工作，并将重心放在交付所需的功能上。

使用 AWS Identity and Access Management（IAM）定义权限，以允许对您要在账户内和账户之间共享的资源进行受控访问。

期望结果：

- 您使用版本受控的存储库来分享应用程序库、脚本程序、程序文档和其他系统文档。
- 您将基础设施标准作为版本受控的 AWS CloudFormation 模板分享。
- 您查看各团队学到的经验教训。

常见反面模式：

- 由于组织中广泛使用有错误的库，导致了长时间的停机。自此之后，您已经迁移到可靠的库。您组织中的其他团队尚未意识到风险的存在。没有人记录和分享使用这个库的体验，也没人意识到风险。
- 您已经确定了内部共享微服务中，导致会话中断的边缘案例。为了避免这一边缘案例的出现，您更新了对服务的调用。您组织中的其他团队尚未意识到风险的存在。
- 您已找到一种方法，可以显著降低其中一个微服务的 CPU 利用率要求。您不知道其他团队是否可以利用这种技术。

建立此最佳实践的好处：分享经验教训可以为改进提供支持，并尽可能地从经验中获益。

在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

- 记录和分享经验教训：设置程序来记录在开展运营活动和回顾性分析过程中获得的经验教训，供其他团队利用。
- 分享经验教训：设置程序，以便在不同团队中分享经验教训和相关构件。例如，通过方便访问的 Wiki，分享更新后的程序、指南、管理机制和最佳实践。通过公共存储库分享脚本、代码和库。
  - Delegating access to your AWS environment
  - Share an AWS CodeCommit repository

资源

相关最佳实践：

- OPS02-BP06 预先定义或协商团队间的职责
- OPS05-BP01 使用版本控制
- OPS05-BP06 共享设计标准
- OPS11-BP03 实施反馈环路
- OPS11-BP07 审核运营指标

相关文档：

- Reduce project delays with a docs-as-code solution

相关视频：

- Delegating access to your AWS environment
- AWS Supports You | Exploring the Incident Management Tabletop Exercise

OPS11-BP09 分配时间进行改进

流程中专用的时间和资源可以实现持续增量改进。

期望结果：

- 您创建了临时的环境副本，这可以降低试验和测试的风险、工作量及成本。
- 这些重复的环境可用于测试分析、试验、开发和测试计划改进时所得出的结论。
- 您开展实际演练活动，并使用故障注入服务（FIS，Fault Injection Service），供团队在类似于生产环境的条件下开展所需的控制措施和防护机制实验。

常见反面模式：

- 您的应用程序服务器中存在一个已知性能问题。它被添加到待办事项中，列在所有计划功能实施之后。如果一直保持这个速度添加计划功能，那么性能问题将永远无法解决。
- 为了支持持续改进，您批准管理员和开发人员利用他们所有的额外时间来选择和实施改进。没有完成任何改进。
- 运营验收完成后，您再也没有测试过运营实践。

建立此最佳实践的好处：通过在流程中投入专门的时间和资源，您可以实现持续增量改进。

在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

- 分配时间进行改进：在流程中投入专门的时间和资源，用于实现持续增量改进。
- 实施更改以便改进，并评估结果以确定是否成功。
- 如果结果不符合目标，并且仍然需要改进，则寻求其他行动方案。
- 通过演练日活动来模拟生产工作负载，并使用从这些模拟中学到的经验教训进行改进。

资源

相关最佳实践：

- OPS05-BP08 使用多个环境

相关视频：

- AWS re:Invent 2023 - Improve application resilience with AWS Fault Injection Service

总结

卓越运营是一项持续性和迭代性的工作。

拥有共同的目标可帮助您的组织迈向成功。确保每个人都了解自己在实现业务成果方面发挥的作用，以及他们如何影响他人取得成功的能力。为您的团队成员提供支持，以便他们可以支持您的业务成果。

我们应将每个运营事件和每次失败视为改进架构运营的机会。通过了解工作负载的需求，预定义记录日常活动的运行手册以及指导解决问题的行动手册，运用 AWS 中的运营即代码功能，并保持情景感知，让您的运营做好更充分的准备，并在事件发生时能更有效地做出响应。

通过专注于随着优先级的变化进行的增量改进，以及从事件响应和回顾性分析中汲取的经验教训，您将提高活动的效率和有效性，从而实现业务的成功。

AWS 致力于帮助您构建和运行架构，以便在构建响应迅速的自适应部署的同时最大限度地提高效率。为了提升工作负载的卓越运营，您应该使用本白皮书中讨论的最佳实践。

贡献者

- Rich Boyd，Amazon Web Services Well-Architected 部门的 Operational Excellence Pillar Lead
- Jon Steele，Amazon Web Services Well-Architected 部门的 Solutions Architect
- Ryan King，Amazon Web Services Sr. Technical Program Manager
- Chris Kunselman，Amazon Web Services Advisory Consultant
- Peter Mullen，Amazon Web Services Advisory Consultant
- Brian Quinn，Amazon Web Services Sr. Advisory Consultant
- David Stanley，Amazon Web Services Cloud Operating Model Lead
- Chris Kozlowski，Amazon Web Services Enterprise Support 部门的 Senior Specialist Technical Account Manager
- Alex Livingstone，Amazon Web Services Cloud Operations 部门的 Principal Specialist Solutions Architect
- Paul Moran，Amazon Web Services Enterprise Support 部门的 Principal Technologist
- Peter Mullen，Amazon Web Services Professional Services 部门的 Advisory Consultant,
- Chris Pates，Amazon Web Services Enterprise Support 部门的 Senior Specialist Technical Account Manager
- Arvind Raghunathan，Amazon Web Services Enterprise Support 部门的 Principal Specialist Technical Account Manager
- Ben Mergen，Amazon Web Services Senior Cost Lead Solutions Architect

延伸阅读

如需更多指导，请参考以下资源：

- AWS Well-Architected Framework
- AWS Architecture Center

文档修订

要获得有关此白皮书的更新通知，请订阅 RSS 源。

变更	说明	日期
已更新白皮书	为最佳实践更新了新的实施指导。	June 27, 2024
主要内容更新和整合	多个最佳实践领域的内容已进行更新和整合。两个最佳实践领域（OPS 04 和 OPS 08）已重新编写，增加了新的内容和重点。 以下领域的最佳实践已进行更新和整合：运营设计、降低部署风险和 了解运营状况访问 AWS 资源。最佳实践领域 OPS 04 已更新为 实现可观测性访问 AWS 资源。最佳实践领域 OPS 08 已更新为 利用工作负载可观测性访问 AWS 资源。	October 3, 2023
针对新框架进行了更新	为最佳实践更新了规范性指南并增加了新的最佳实践。	April 10, 2023
已更新白皮书	为最佳实践更新了新的实施指导。	December 15, 2022
已更新白皮书	扩展了最佳实践并增加了改进计划。	October 20, 2022
次要更新	较小的编辑性修改。	August 8, 2022
已更新白皮书	反映新的 AWS 服务和功能以及最新最佳实践的更新。	February 2, 2022

次要更新	在简介中添加了可持续性支柱。	December 2, 2021
针对新框架进行了更新	反映新的 AWS 服务和功能以及最新最佳实践的更新。	July 8, 2020
已更新白皮书	反映新的 AWS 服务和功能以及最新参考的更新。	July 1, 2018
原始版本	发布了卓越运营支柱 – AWS Well-Architected Framework。	November 1, 2017