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： 2023 年 10 月 3 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：

- 执行运营即代码： 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。您可以将整个工作负载（应用程序、基础设施等）定义为代码，并使用该代码进行更新。您可以为运营流程编写脚本，并通过启动这些脚本来自动执行流程，以响应事件。通过执行运营即代码，您可以减少人为错误并实现对事件的一致响应。
- 频繁进行可逆的小规模更改： 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。
- 经常优化运营流程： 在改进工作负载的同时，您也要适当改进一下运营。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。
- 预测故障： 执行“故障演练”，找出潜在的问题，以便消除和缓解问题。测试您的故障场景，并确认您了解相应影响。测试您的响应程序，以确保它们的有效性，以及团队熟练他们的流程。设置定期的实际演练，以测试工作负载和团队对模拟事件的响应。
- 从所有运营故障中吸取经验教训： 从所有运营事件和故障中吸取的经验教训，推动改进。在多个团队乃至组织范围中分享经验教训。
- 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。

- 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。

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做出明智的生产决策。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 评估外部客户需求

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。这可以确保您充分了解实现您期望的业务成果所需的运营支持。

常见反模式：

- 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商业价值。

未建立此最佳实践暴露的风险等级： 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。
  - 审查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

相关文档：

- AWS Well-Architected Framework 概念 – 反馈循环

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。

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，开发团队技能、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。要随着需求的变化更新重点。

常见反模式：

- 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，告知您通过安排工作的优先级来实现商业价值。

未建立此最佳实践暴露的风险等级： 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

相关文档：

- AWS Well-Architected Framework 概念 – 反馈循环

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。

如示例所示，Well-Architected Framework 强调学习、衡量和改进。它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 AWS Well-Architected Tool，可帮助您在开发之前查看方法、生产前的工作负载状态以及生产中的工作负载状态。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 客户可以使用针对任务关键型工作负载的指导式 Well-Architected 审核，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，可帮助各团队在工作负载以及他们在实现成功中的角色方面达成一致的理解。通过审查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，让您可以访问一组核心检查，这些检查会提出优化建议，帮助确定您的运营重点。商业和企业支持客户可以访问其他检查，这些检查重点关注安全性、可靠性、性能和成本优化，可进一步帮助他们帮助确定运营重点。

常见反模式：

- 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： 发现并了解对组织和工作负载的威胁后，您可以确定要解决的威胁、需解决的威胁的优先级以及执行操作所需的资源。

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

- 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

相关文档：

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

OPS01-BP06 评估权衡

在有冲突的利益或替代方法之间做出权衡并评估其影响，以便在确定工作重心或选择行动方案时做出明智的决策。例如，加快新功能上市的速度可能会比成本优化更重要，或者您可以为非关系数据选择关系数据库，以简化迁移系统的工作，而不是迁移到针对您的数据类型优化的数据库和更新您的应用程序。

AWS 可以帮您向团队介绍 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。您应该使用由 AWS Support （AWS 知识中心， AWS 开发论坛和 AWS Support 中心）和 AWS 文档提供

的资源 来培训您的团队。请通过 AWS Support 中心联系 AWS Support，获取与 AWS 问题相关的帮助。

AWS 还在 Amazon Builders' Library 中分享了我们通过 AWS 运营 学到的最佳实践和模式。您可以通过 AWS Blog 和 AWS 官方播客，获得各种其他有用信息。

常见反模式：

- 您正在使用关系数据库管理时间序列和非关系数据。有一些数据库选项经过优化后可以支持您正在使用的数据类型，但是您没有意识到这些好处，因为您尚未评估解决方案之间的权衡。
- 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。

建立此最佳实践的好处： 了解您的选择产生的影响和后果可以帮助您排定选择的优先顺序。

未建立此最佳实践暴露的风险等级： 中

实施指导

- 评估权衡：在利益互有冲突的目标之间做出权衡并评估其影响，以便在确定工作重心时做出明智的决策。例如，加快新功能上市的速度可能比成本优化更重要。
- AWS 可以帮您向团队介绍 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。您应该使用由 AWS Support（AWS 知识中心、AWS 开发论坛和 AWS Support 中心）和 AWS 文档提供的资源来培训您的团队。请通过 AWS Support 中心联系 AWS Support，获取与 AWS 问题相关的帮助。
- AWS 还在 Amazon Builders' Library 中分享了我们通过 AWS 运营学到的最佳实践和模式。您可以通过 AWS Blog 和 AWS 官方播客，获得各种其他有用信息。

资源

相关文档：

- AWS Blog
- AWS Cloud 合规性
- AWS 开发论坛
- AWS 文档

- AWS 知识中心
- AWS Support
- AWS Support 中心
- Amazon Builders' Library
- AWS 官方播客

OPS01-BP07 管理收益和风险

管理收益和风险，以便在确定工作重心时做出明智的决策。例如，为了向客户提供重要的新功能，部署仍存在未决问题的工作负载是可以接受的。这可能会降低相关风险，或者允许风险继续存在可能会令人无法接受，在这种情况下，您将采取措施来化解风险。

您可能会发现，您需要在某个时间点侧重于一小部分运营重点。长期使用平衡的方法来确保所需能力的发展和风险管理。要随着需求的变化更新重点

常见反模式：

- 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万能库。您尚未评估从未知来源采用此库的风险，也不知道它是否包含漏洞或恶意代码。
- 您决定开发和部署新功能，而不是修复现有问题。您尚未评估在部署功能之前将问题继续留存的风险，也无从知晓会对客户产生哪些影响。
- 由于合规团队提出了未指明的顾虑，您决定不部署客户频繁请求的功能。

建立此最佳实践的好处： 确定选择可以带来的收益并了解组织所面临的风险有助于您做出明智的决定。

未建立此最佳实践暴露的风险等级： 低

实施指导

- 管理收益和风险：在决策的收益与涉及的风险之间取得平衡。
  - 确定收益：根据业务目标、需求和优先事项来确定收益。例如上市时间、安全性、可靠性、性能和成本等。
  - 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。
  - 对照风险评估收益并做出明智决策：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。

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。

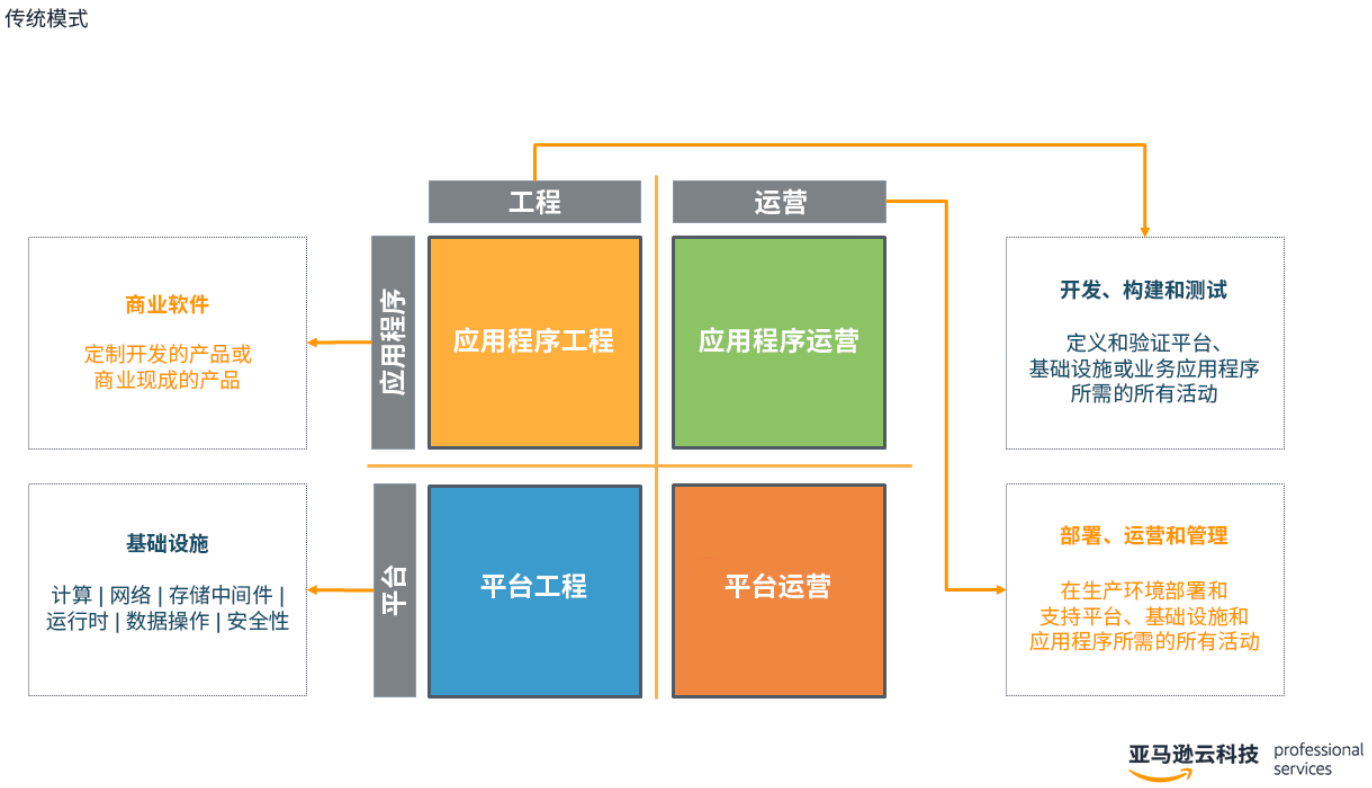
主题

- 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。



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。

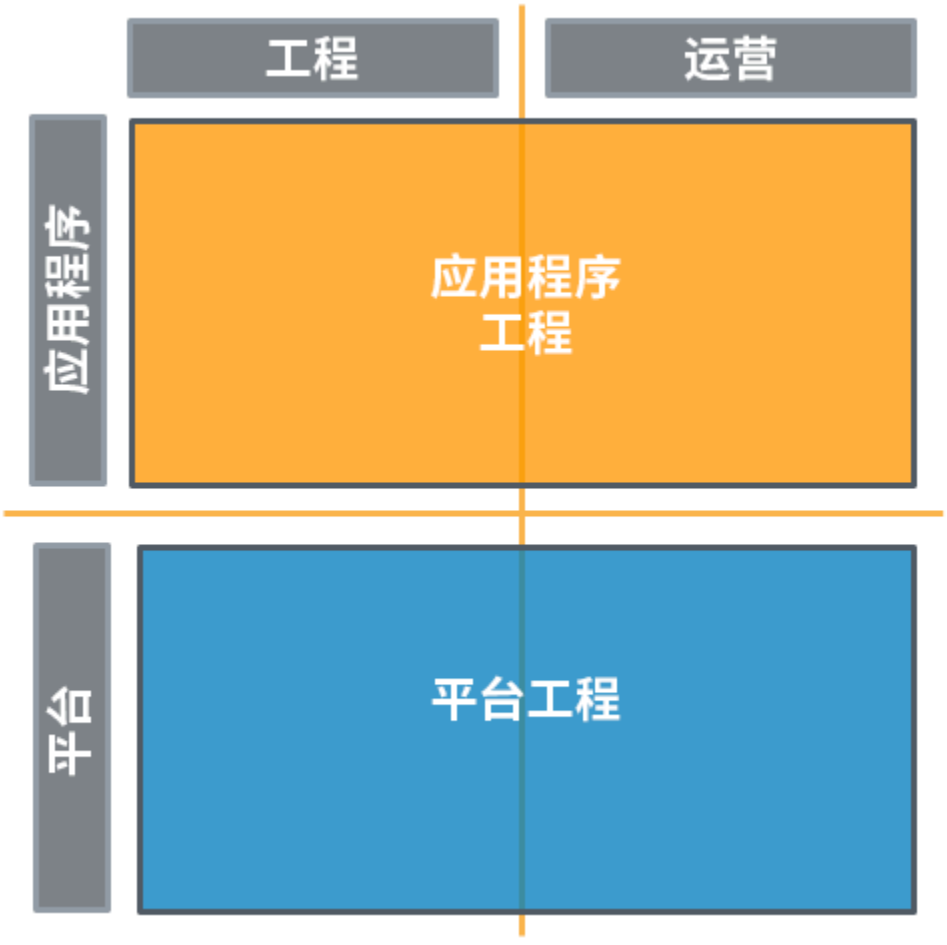
任务向团队或在团队之间的转移会增加复杂性，并造成瓶颈和延迟问题。请求可能会被延迟，直至它们成为重点事项。较迟发现缺陷可能需要大量返工，可能需要再次经历相同的团队及其职能部门。如果存在需要工程团队采取行动的事件，则将因转移活动而延迟响应。

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

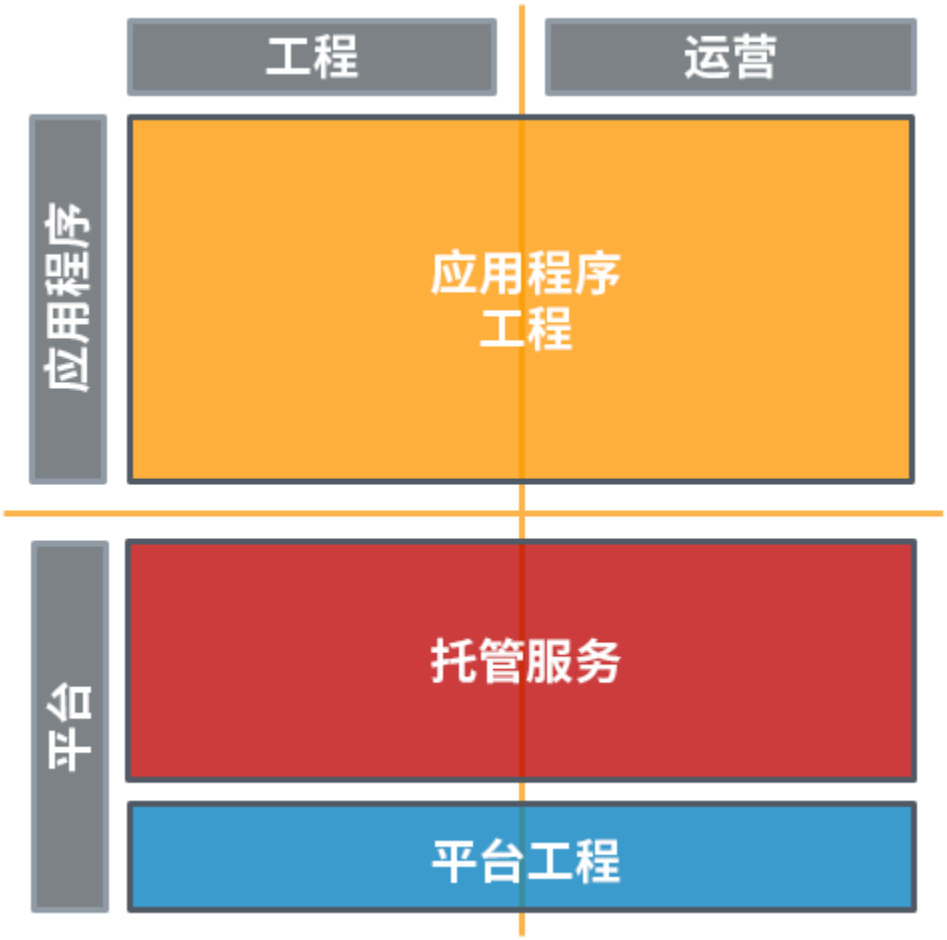
这种“分离的 AEO 和 IEO”模式采用“你构建，你运行”的方法。

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。

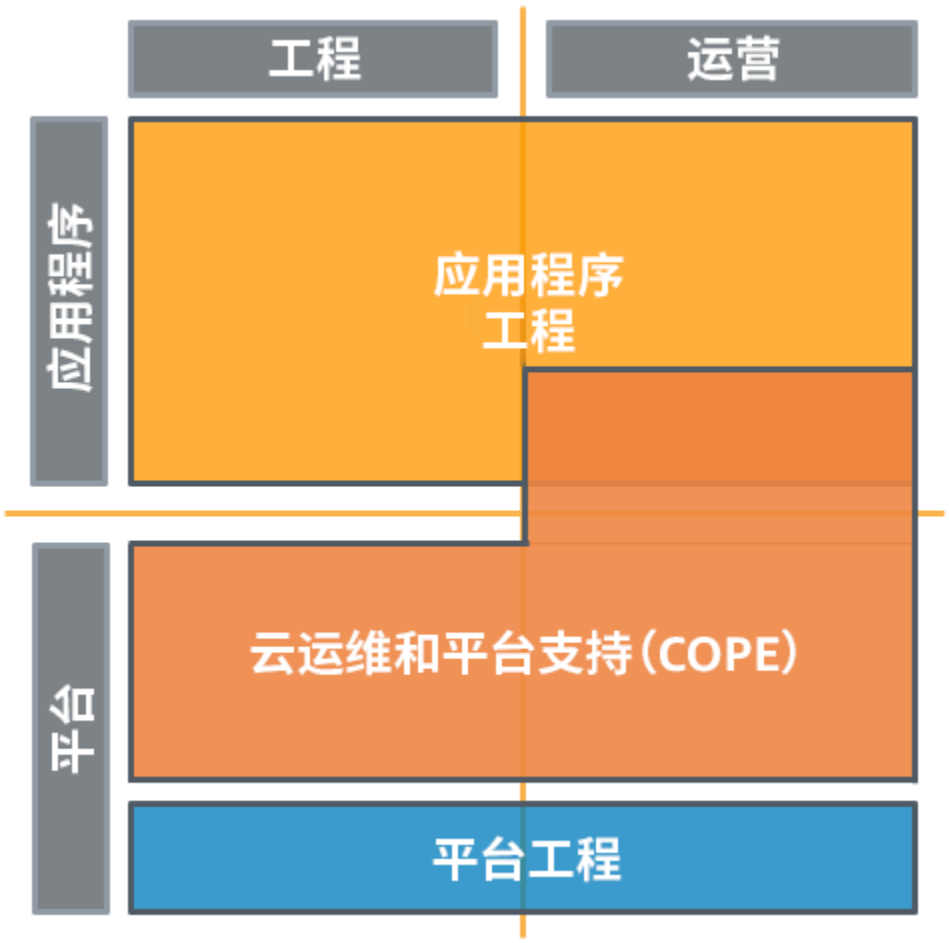
您的平台工程设计团队根据这些标准构建核心共享平台功能，供应用程序团队使用。他们通过自助服务机制，编纂提供给应用程序团队的企业参考架构和模式。使用 AWS Service Catalog 等服务，应用程序团队可以部署经过批准的参考架构、模式、服务和配置，这些自动符合集中监管和安全标准。

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。

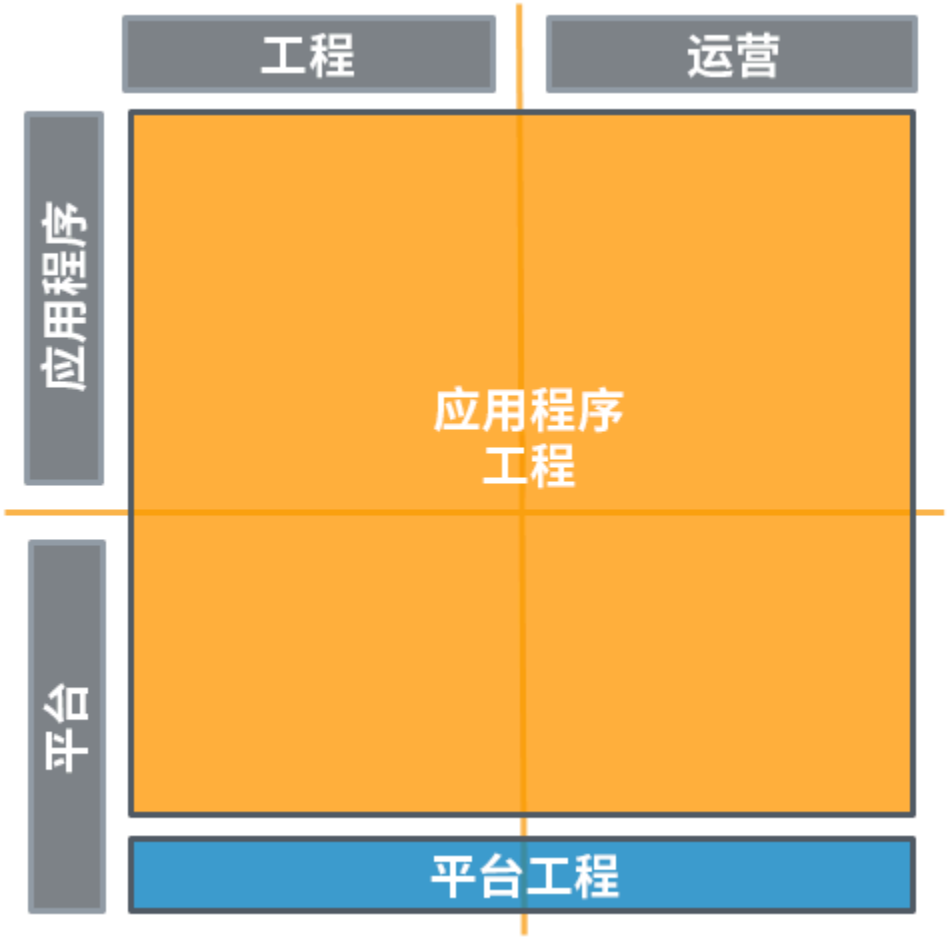
从团队的成功中，您可以获得组织的标准、最佳实践、流程和专业知识方面的进步。您可以建立一种机制，复制这些成功模式供新团队采用，或者在云端实现现代化。此模式强调 COPE 团队帮助建立应用程序团队以及转换知识和构件的能力。它减少了应用程序团队的运营负担，同时也降低了应用程序团队无法高度独立的风险。它在 CCoE、COPE 与应用程序团队之间建立了关系，创建反馈循环用于支持进一步的演进和创新。

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 确定流程和程序所有者
- OPS02-BP03 确定对运营活动绩效负责的所有者
- OPS02-BP04 团队成员知道自己的责任
- OPS02-BP05 制定用于确定责任和所有权的机制
- OPS02-BP06 制定用于请求添加、更改和例外的机制
- OPS02-BP07 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. 首先定义组织的所有权。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，即使它们所属的个人不再是公司的员工，您仍可以访问它们。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. 有关如何为组织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. 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 确定流程和程序所有者 - 用于支持资源的流程和程序取决于资源所有权。
- OPS02-BP04 团队成员知道自己的责任 - 团队成员应了解他们是哪些资源的所有者。
- OPS02-BP05 制定用于确定责任和所有权的机制 - 可以使用标签或账户联系人等机制来发现所有权。

相关文档：

- AWS 账户管理 - 更新联系信息
- AWS Config 规则 - 必需的标签
- AWS Organizations - 更新组织中的备用联系人
- AWS 标记最佳实践白皮书

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
- AWS Organizations

OPS02-BP02 确定流程和程序所有者

了解谁对各个流程和程序的定义拥有所有权、为何使用这些特定的流程和程序，以及为什么存在这种所有权。了解使用特定流程和程序的原因将有助于发现改进机会。

建立此最佳实践的好处： 了解所有权可以确定谁有权批准改进和/或实施改进。

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

- 确定负责定义流程和程序的所有者：捕获环境中使用的流程和程序，以及负责其定义的个人或团队。
  - 确定流程和程序：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。
  - 确定谁负责定义流程或程序：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。
  - 在活动构件的元数据中捕获所有权：在 AWS Systems Manager 之类的服务中通过文档和 AWS Lambda 函数自动执行的程序支持以标签形式捕获元数据信息。使用标签或资源组捕获资源所有权，详细说明所有权和联系信息。使用 AWS Organizations 创建标记策略，并确保捕获所有权和联系信息。

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。

建立此最佳实践的好处： 了解谁负责执行活动可让我们知晓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。

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

- 确定对运营活动绩效负责的所有者：了解环境中使用的流程和程序的责任分配
  - 确定流程和程序：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。
  - 确定各项活动的执行负责人：确定负责某项活动的团队。确保他们具有活动的详细信息，具备执行活动所需的技能以及正确的权限、访问权限和工具。他们必须了解活动执行条件（例如基于某个事件或计划）。显示这些信息，以便组织成员确定针对特定需求他们需要联系的人员（团队或个人）。

OPS02-BP04 团队成员知道自己的责任

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，这可帮助您确定任务的优先级以及自身角色的重要性。这使团队成员能够了解需求并做出适当响应。

建立此最佳实践的好处：了解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。

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

- 确保团队成员了解各自的角色和责任：确定团队成员的角色和责任，并确保他们了解对其角色的期望。显示这些信息，以便组织成员确定他们为满足特定需求而需要联系的人员（团队或个人）。

OPS02-BP05 制定用于确定责任和所有权的机制

在未确定个人或团队时，要为有权分配所有权或计划满足该需求的人定义升级路径。

建立此最佳实践的好处： 了解谁负责或拥有所有权可使您与合适的团队或团队成员联系，以提出请求或转换任务。确定谁有权分配责任或所有权或计划满足需求，可以降低不作为的风险并减少无法满足的需求。

未建立此最佳实践暴露的风险等级： 高

实施指导

- 制定用于确定责任和所有权的机制：为组织成员提供可访问机制，以发现和确定所有权和责任。这些机制将使他们能够根据特定的需求确定相关的联系人（团队或个人）。

OPS02-BP06 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，做出明智的决定，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 确定流程和程序所有者 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 确定流程和程序所有者 - 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-BP07 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 确定流程和程序所有者 - 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 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 支持和鼓励团队成员保持和增强他们的技能组合
- OPS03-BP07 为团队配置适当的资源
- OPS03-BP08 鼓励在团队内部和团队之间提出不同的观点

OPS03-BP01 高管支持

高层领导明确为组织设定期望并评估是否成功。高层领导是采用最佳实践和组织发展的发起人、倡导者和推动者

建立此最佳实践的好处： 积极参与的领导层、表达清晰的期望和共同的目标可确保团队成员了解组织对自己的期望。对成功进行评估可以确定阻碍成功的障碍，以便通过发起人、倡导者或他们的代表进行干预，从而消除这些障碍。

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

- 高管支持：高级领导层明确为组织设定期望并评估是否成功。高层领导是采用最佳实践和组织发展的发起人、倡导者和推动者
  - 设定期望：为您的组织制定和发布目标，包括目标衡量方式。
  - 跟踪目标实现情况：定期衡量目标的逐步实现情况并分享结果，以便在结果有风险时可以采取适当的措施。

- 为实现目标提供必要的资源：定期审核资源是否仍然合适，或者根据以下项目决定是否需要添加资源：新信息、目标变更、责任或您的业务环境。
- 支持您的团队：与团队保持沟通，以便您了解他们的进展情况以及是否受到外部因素的影响。团队受外部因素影响时，需重新评估目标并适当地调整目标。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。
- 推动最佳实践的采用：认可可量化收益的最佳实践并确定创建者和采用者。鼓励进一步采用，实现更大收益。
- 推动团队发展：打造持续改进的文化。鼓励个人和组织的成长与发展。制定长期目标并为此奋斗，逐步实现成功。根据需求、业务目标和业务环境的变化调整此愿景。

OPS03-BP02 赋能团队成员在结果有风险时采取行动

工作负载所有者定义了指南和范围，赋能团队成员在结果有风险时做出响应。当事件超出定义的范围时，使用上报机制获取指示。

建立此最佳实践的好处： 在早期进行测试和验证更改，可以将解决问题的成本降至最低，并降低对客户的影响。经过测试之后再部署，可以最大限度地减少错误。

未建立此最佳实践暴露的风险等级： 高

实施指导

- 赋能团队成员在结果有风险时采取行动：为您的团队成员提供权限、工具和机会，实践有效响应所需的技能。
  - 为您的团队成员提供机会，实践响应所需的技能：提供替代的安全环境，以便在其中安全地对流程和程序进行测试和培训。进行实际演练，让团队成员在模拟的安全环境中获得响应现实意外事件的经验。
  - 定义并确认团队成员采取行动的权限：通过分配对他们所支持的工作负载和组件的权限和访问权限，来明确定义团队成员采取行动的权限。确认授权他们在结果存在风险时采取行动。

OPS03-BP03 鼓励上报

团队成员具有相应机制，如果他们认为结果存在风险，鼓励他们向决策者和利益相关者上报问题。应经常尽早上报，以便能够确定风险，并防止造成意外事件。

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

- 鼓励经常尽早上报：从组织的角度认可，经常尽早上报是一种最佳实践。组织确认并接受，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。
  - 制定上报机制：我们设定明文程序来确定何时应上报以及上报方式。记录权限逐级提升（以采取行动或批准行动）的人员及其联系信息。上报应一直持续，直到团队成员认为他们已将风险移交给能够化解风险的人员，或者他们已联系到对运营该工作负载的风险和责任负责的人员。（即可以最终对工作负载做出决策的人员）。上报内容应包括风险的性质、工作负载的严重性、受影响的人、受到的影响以及紧迫性（即预计影响发生的时间）。
  - 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。

OPS03-BP04 沟通及时、清晰、可行

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。提供必要的相关信息、详细信息和时间（如果可能），为确定是否需要采取措施、需要采取什么措施以及及时采取措施提供支持。例如，提供软件漏洞通知可以加快修补过程；或者，提供计划内促销活动的通知可以实施变更冻结以避免发生服务中断的风险。可以将计划内事件记录在变更日历或维护时间表中，以便团队成员可以确定哪些活动待处理。

期望结果：

- 通过沟通提供背景、细节和时间期望。
- 团队成员清楚地知道何时以及如何采取行动回应沟通。
- 利用变更日历提供预期变更的通知。

常见反模式：

- 每周会出现几次误报警报。每次发生警报时，您都要关掉通知声音。
- 系统会要求您对安全组进行更改，但未告诉你应何时进行修改。
- 当系统纵向扩展时，您会在聊天中不断收到通知，但无需执行任何操作。您避开聊天频道，而错过了重要通知。
- 在不通知运营团队的情况下对生产作出更改。变更会触发警报，并激活随时待命的团队。

建立此最佳实践的好处：

- 您的组织可避免警报疲劳。
- 团队成员可以在必要的背景和期望下行动。
- 可以在变更时段进行变更，从而降低风险。

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

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。识别并删除误报或始终打开的警报。使用变更日历，以便团队成员知道何时可以采取行动以及哪些活动待处理。确认附带必要背景的沟通可带来明确的行动。

客户示例

AnyCompany Retail 使用聊天作为其主要沟通媒介。警报和其他信息会填入特定渠道。当有人必须采取行动时，会清楚地说明预期的结果，并提供运行手册或行动手册供他们使用。他们使用变更日历来安排生产系统的重大变更。

实施步骤

1. 分析警报是否为误报，或警报是否会持续触发。删除或更改这些警报，以便仅在需要人工干预时才触发这些警报。如果触发了警报，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。
2. 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。
3. 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 变更日历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。
4. 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。

资源

相关最佳实践：

- OPS07-BP03 使用运行手册执行程序 - 在结果已知时提供运行手册，使沟通具有可操作性。
- OPS07-BP04 根据行动手册调查问题 - 如果结果未知，则行动手册使沟通具有可操作性。

相关文档：

- AWS 安全公告
- 开放的 CVE

相关示例：

- Well-Architected 实验室：清单和补丁管理（级别 100）

相关服务：

- AWS Chatbot
- AWS Systems Manager 变更日历
- AWS Systems Manager 文档

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 支持和鼓励团队成员保持和增强他们的技能组合

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 Blog， AWS 在线技术讲座， AWS 活动和网络研讨会，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 还在 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 Blog 和 AWS 官方播客分享了各种实用的教材。

您应该利用 AWS 提供的教育资源，例如 Well-Architected 实验室、 AWS Support （ AWS 知识中心， AWS 开发论坛和 AWS Support 中心）和 AWS 文档 来培训您的团队。请通过 AWS Support 中心联系 AWS Support，获取与 AWS 问题相关的帮助。

AWS 培训与认证 提供了一些免费培训，可以通过自定进度的数字课程，学习 AWS 的基础知识。您还可以注册讲师指导培训，进一步帮助培养您团队的 AWS 技能。

未建立此最佳实践暴露的风险等级： 中

实施指导

- 支持和鼓励团队成员保持和增强他们的技能组合：我们必须不断学习，这样才能采用新技术、支持创新，并随需求和责任的变化继续提供支持，以支持工作负载。
  - 提供教育资源：专门安排时间，提供培训材料、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。为初级团队成员提供与高级团队成员接触的机会，可以请高级团队成员作为他们的导师，或允许他们跟随高级团队成员学习并了解方法和技能。鼓励学习与工作没有直接关系的内容，拓展视野。
  - 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
  - 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，以验证他们所学到的知识并认可他们的成就。

资源

相关文档：

- AWS 入门资源中心
- AWS Blog

- AWS Cloud 合规性
- AWS 开发论坛
- AWS 文档
- AWS 在线技术讲座
- AWS 活动和网络研讨会
- AWS 知识中心
- AWS Support
- AWS 培训与认证
- AWS Well-Architected 实验室，
- Amazon Builders' Library
- AWS 官方播客.

OPS03-BP07 为团队配置适当的资源

培养团队成员的能力，并提供工具和资源来支持工作负载需求。团队成员超负荷工作会增加人为错误导致事故发生的风险。投资于工具和资源（例如，对频繁执行的活动实现自动化）可以提高团队的效率，让他们为其他活动提供支持。

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

- 为团队配置适当的资源：确保您了解团队取得的成功，以及推动团队成功或导致团队不成功的因素。采取行动为团队提供适当的资源。
  - 了解团队绩效：衡量团队运营成果的实现和资产的开发。跟踪输出和错误率随时间发生的变化。与团队沟通，了解会对他们工作产生影响的挑战（例如责任增加、技术变化、人员流失或支持的客户增加）。
  - 了解对团队绩效的影响：与团队保持沟通，以便您了解他们的进展情况以及是否受到外部因素的影响。团队受外部因素影响时，需重新评估目标并适当地调整目标。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。
  - 为团队提供必要资源，助力团队取得成功：定期审核资源是否仍然合适，或者是否需要添加新资源，并做出适当的调整，为团队提供支持。

OPS03-BP08 鼓励在团队内部和团队之间提出不同的观点

利用跨组织的多样性来寻求多种独特的见解。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。在团队内部提升包容性、多样性和可达性有助于获取有益的见解。

组织文化会直接影响团队成员的工作满意度和保留率。增强团队成员的参与度和能力，助力业务成功。

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

- 寻求不同的观点和视角：鼓励所有人做出贡献。为弱势群体发声。在会议中轮换角色和职责。
  - 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。他们将从角色以及与其他团队成员的互动中获得经验和见解，而之前他们可能没有机会与他们互动。他们会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。随着见解的不断增多，可能会出现新的商机，或者可能会发现新的改进机会。让团队成员轮流体验他人日常执行的任务，了解执行这些任务的需求和影响。
  - 提供安全舒适的环境：制定相应的政策和控制措施，保护组织内团队成员的身心健康。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、环境中未被发现的风险）。
  - 让团队成员充分参与：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。每天都面临各种挑战的团队成员已不断开发应对这些挑战的技能。这些开发的技能独一无二，可以为您的组织带来巨大的效益。根据需要为团队成员提供住所将有助于他们增加对团队的贡献，从而提升您的效益。

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。

期望的结果： 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做出性能优化决策，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，您可以做出以数据为导向的决策。
- 提高运营效率：数据驱动的资源利用率可提高成本效益。
- 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。

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。它会收集、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，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. 定义和监控业务 KPI： 建立 自定义指标 并使其与您的 业务成果相一致访问 AWS 资源。
4. 使用 AWS X-Ray 来检测您的应用程序： 除了部署 CloudWatch 代理外，还必须 检测您的应用程序 以发出跟踪数据。此过程可让您进一步了解工作负载行为和性能。
5. 标准化整个应用程序中的数据收集： 标准化整个应用程序中的数据收集实践。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。
6. 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 来分析指标和日志，并使用 AWS X-Ray 来分析跟踪数据。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 识别关键绩效指标
- 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

相关文档：

- AWS 可观测性最佳实践
- CloudWatch 用户指南
- AWS X-Ray 开发人员指南
- 检测分布式系统的运营可见性
- AWS 可观测性 Skill Builder 课程
- Amazon CloudWatch 新增功能
- AWS X-Ray 新增功能

相关视频：

- AWS re:Invent 2022 – Amazon 的可观测性最佳实践
- AWS re:Invent 2022 - 开发可观测性战略

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。它提供有关与 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。通过检测您的应用程序以发出有关这些依赖关系的指标、日志和跟踪，您可以更清楚地了解可能影响您的工作负载的潜在瓶颈、性能问题或故障。

期望的结果： 您的工作负载所依赖的依赖项按预期执行，使您能够主动解决问题并确保最佳的工作负载性能。

常见反模式：

- 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。

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，就可以针对它们设计一个量身定制的监控策略。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. 利用 Amazon CloudWatch 网络监测仪： 它提供对全球互联网的见解，有助于了解可能影响外部依赖项的中断。
4. 随时了解最新信息 - 借助 AWS Health Dashboard： 当 AWS 遇到可能会影响您的服务的事件时，它会提供警报和修正指导。
5. 检测您的应用程序 - 使用 AWS X-Ray： 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。这对于深入了解依赖项并确定它们是不是造成操作问题的根源非常重要。
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，并检查它们是否符合应用程序的要求。

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-BP05 实施分布式跟踪

相关文档：

- 什么是 AWS Health?
- Using Amazon CloudWatch Internet Monitor
- AWS X-Ray 开发人员指南
- Amazon DevOps Guru 用户指南

相关视频：

- Visibility into how internet issues impact app performance
- Introduction to Amazon DevOps Guru

相关示例：

- Gaining operational insights with AIOps using Amazon DevOps Guru
- AWS Health Aware

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。

期望的结果： 软件更改须在交付前进行测试。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： 软件部署的更改失败率降低。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。

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测试。

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？
   a. 此 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 CodeGuru Reviewer 可以测试您的应用程序代码是否存在缺陷。
   b. 您可以使用 AWS CodeBuild 对软件构件执行测试。
   c. 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-BP10 完全自动化集成和部署

相关文档：

- 采用测试驱动型开发方法

- 使用 TaskCat 和 AWS CloudFormation 的 CodePipeline 自动化测试管道
- 使用开源 SCA、SAST 和 DAST 工具构建端到端 AWS DevSecOps CI/CD 管道
- 开始测试无服务器应用程序
- 我的 CI/CD 管道统领我的发布
- 《在 AWS 上实践持续集成和持续交付》白皮书

相关视频：

- AWS re:Invent 2020：可测试的基础设施：AWS 上的集成测试
- 2021 AWS 峰会（澳大利亚和新西兰）- 用 CDK 和测试驱动型开发推动测试优先战略
- 用 AWS CDK 测试您的基础设施即代码

相关资源：

- AWS 部署管道参考架构 - 应用程序
- AWS Kubernetes DevSecOps 管道
- 策略即代码研讨会 – 测试驱动型开发
- 通过使用 AWS CodeBuild 从 GitHub 为 Node.js 应用程序运行单元测试
- 使用 Serverspec 对基础设施代码进行测试驱动型开发

相关服务：

- 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。

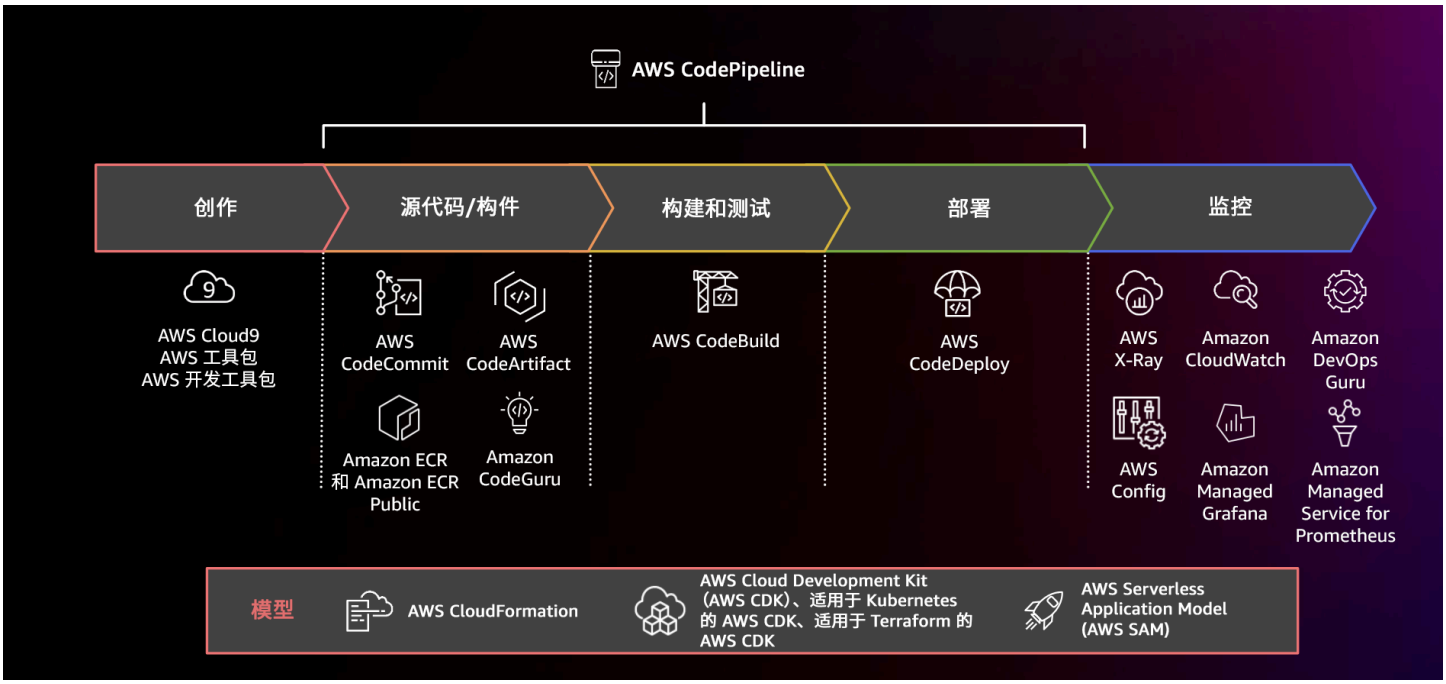
建立此最佳实践的好处： 制定相应机制来管理活动的构建和部署。这样，您可以减少执行重复任务的工作量，让团队成员腾出时间专注于高价值的创造性任务，还可以减少手动程序导致的错误。

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。

期望的结果： 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，能够最大限度地减少引入生产中的问题。使用最佳实践（例如，结对编程和代码审查）提高代码质量。

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

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。

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 CodeGuru Reviewer 可以使用机器学习为 Java 和 Python 代码提供编程建议。

b. 您可以使用 AWS Cloud9 创建共享开发环境，在那里您可以协作开发代码。

实施计划的工作量级别： 中。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-BP06 共享设计标准 - 作为代码质量实践的一部分，您可以共享设计标准。

相关文档：

- 敏捷软件指南
- 我的 CI/CD 管道统领我的发布
- 使用 Amazon CodeGuru Reviewer 自动审查代码
- 采用测试驱动型开发方法
- DevFactory 如何使用 Amazon CodeGuru 构建更好的应用程序
- 关于结对编程
- RENGA Inc. 使用 Amazon CodeGuru 自动进行代码审查
- 敏捷开发的艺术：测试驱动型开发
- 为什么代码审查很重要（而且实际上节省了时间！）

相关视频：

- AWS re:Invent 2020：使用 Amazon CodeGuru 持续改进代码质量
- 2021 AWS 峰会（澳大利亚和新西兰）- 用 CDK 和测试驱动型开发推动测试优先战略

相关服务：

- 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。

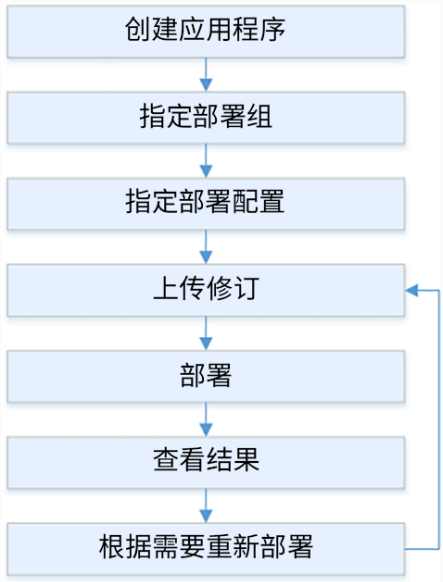
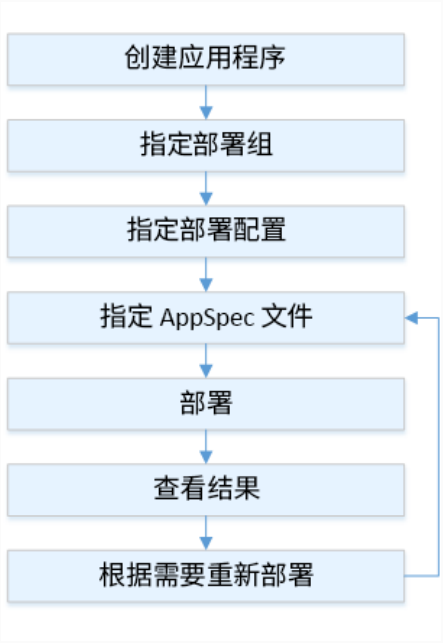
未建立这种最佳实践的情况下暴露的风险等级： 中

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 与其他 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 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

A 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。它们存储在一个中央位置并经常更新。

常见反模式：

- 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。

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

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。使用诸如 AWS Systems Manager 自动化之类的服务，您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。

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他运行手册。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，您可以用它开始构建运行手册。

```
# 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 `Runbook Title` 以及 `Runbook Info` `######`。
4. 从第一步开始，填写运行手册的 `Steps` 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，告诉您的团队和其他利益相关者。

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 确定流程和程序所有者：运行手册应该有一个负责人负责维护。
- OPS07-BP04 根据行动手册调查问题：运行手册和行动手册彼此相似，但有一个关键区别：运行手册包含期望结果。在许多情况下，一旦行动手册确定了根本原因，就会触发运行手册。
- OPS10-BP01 使用流程来管理事件、意外事件和问题：运行手册是良好的事件、意外事件和问题管理实践的一部分。
- OPS10-BP02 针对每个提醒设置一个流程：应使用运行手册和行动手册来响应警报。随着时间的推移，应自动进行这些响应。
- OPS11-BP04 执行知识管理：维护运行手册是知识管理的一个关键部分。

相关文档：

- 利用自动化行动手册和运行手册实现卓越运营
- AWS Systems Manager：使用运行手册
- 适用于 AWS 大型迁移的迁移行动手册 - 任务 4：改进迁移运行手册
- 使用 AWS Systems Manager Automation 运行手册解决运营任务

相关视频：

- AWS re:Invent 2019：运行手册、事件报告和事件响应 DIY 指南（SEC318-R1）
- 如何在 AWS | Amazon Web Services 上实现 IT 运营自动化
- 将脚本集成到 AWS Systems Manager

相关示例：

- AWS Systems Manager：自动化演练
- AWS Systems Manager：从最新的快照运行手册中还原根卷

- 使用 Jupyter Notebook 和 CloudTrail Lake 构建 AWS 意外事件响应运行手册
- Gitlab - 运行手册
- Rubix - 用于在 Jupyter Notebook 中构建运行手册的 Python 库
- 使用 Document Builder 创建自定义运行手册
- Well-Architected 实验室：使用行动手册和运行手册自动完成操作

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事件的分步指南。发生事件时，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。

期望的结果： 您的组织有针对常见事件的行动手册。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。

常见反模式：

- 要调查事件，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：

- 行动手册可帮助您减轻事件带来的影响。
- 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。

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

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。通过使用诸如 AWS Systems Manager Automations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事件所需的平均时间。

客户可以使用 AWS Systems Manager Incident Manager 来响应事件。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，并在整个事件中进行协作。它使用 AWS Systems Manager Automations 加快检测和恢复的速度。

客户示例

生产事件影响了 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。它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. 使用 Markdown 模板填写 `Playbook Name######## 部分以及 Playbook Info########`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 Automations 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 确定流程和程序所有者：行动手册应该有一个负责人来负责维护。
- OPS07-BP03 使用运行手册执行程序：运行手册和行动手册类似，但有一个关键区别：运行手册包含期望的结果。在许多情况下，一旦行动手册确定了根本原因，就会使用运行手册。
- OPS10-BP01 使用流程来管理事件、意外事件和问题：行动手册是良好的事件、意外事件和问题管理实践的一部分。
- OPS10-BP02 针对每个提醒设置一个流程：应使用运行手册和行动手册来响应警报。随着时间的推移，应自动进行这些响应。
- OPS11-BP04 执行知识管理：维护行动手册是知识管理的一个关键部分。

相关文档：

- 利用自动化行动手册和运行手册实现卓越运营
- AWS Systems Manager：使用运行手册
- 使用 AWS Systems Manager Automation 运行手册解决运营任务

相关视频：

- AWS re:Invent 2019：运行手册、事件报告和事件响应 DIY 指南（SEC318-R1）
- AWS Systems Manager Incident Manager – AWS 虚拟研讨会
- 将脚本集成到 AWS Systems Manager

相关示例：

- AWS 客户行动手册框架
- AWS Systems Manager：自动化演练
- 使用 Jupyter notebook 和 CloudTrail Lake 构建 AWS 事件响应运行手册
- Rubix – 用于在 Jupyter notebook 中构建运行手册的 Python 库
- 使用 Document Builder 创建自定义运行手册
- Well-Architected 实验室：根据行动手册和运行手册自动完成操作
- Well-Architected 实验室：使用 Jupyter 的事件响应行动手册

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 确定流程和程序所有者

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-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。

期望的结果： 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。

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

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. 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。
3. 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。

4. 实现 CloudWatch Logs 指标筛选器： 配置 CloudWatch 日志指标筛选器 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。
5. 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。

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：

- 使用 CloudWatch Logs Insights 分析日志数据
- 使用 CloudWatch Contributor Insights
- 创建和管理 CloudWatch Logs 日志指标筛选器

相关视频：

- 使用 CloudWatch Logs Insights 分析日志数据
- Use CloudWatch Contributor Insights to Analyze High-Cardinality Data

相关示例：

- CloudWatch Logs 示例查询
- 可观测性研讨会

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。

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

常见反模式：

- 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做出明智的决策。
- 通过优化组件交互来改善用户体验。

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

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 来监控应用程序运行状况。
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、指标、日志、警报和其他运行状况信息。
4. 启用 X-Ray Insights：
   a. 开启 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、查明模式和挖掘见解。

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，从而进行更有针对性的分析。
7. 纳入 Amazon DevOps Guru： 利用 Amazon DevOps Guru 受益于机器学习模型，查明跟踪数据中的操作异常。
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用户的延迟趋势和错误。
10. 与日志关联： 将跟踪数据与相关日志关联（在 X-Ray 跟踪视图中），可以详细了解应用程序行为。这允许您查看与跟踪的事务直接关联的日志事件。

实施计划的工作量级别： 中。

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：

- 使用 ServiceLens 监控应用程序运行状况
- 使用 X-Ray Analytics 探究跟踪数据
- 使用 X-Ray Insights 检测跟踪数据中的异常
- 使用 CloudWatch Synthetics 进行持续监控

相关视频：

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

相关示例：

- One Observability Workshop
- 使用 X-Ray 实现 AWS Lambda
- CloudWatch Synthetics 金丝雀模板

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。

期望的结果： 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。

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. 定义关键绩效指标（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. 实现异常检测：

- 使用 AWS Cost Anomaly Detection： 设置 AWS Cost Anomaly Detection 以自动检测异常模式，确保仅针对真正的异常情况生成警报。
- 使用 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 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操作异常。
  b. 导航到 通知设置 （在 DevOps Guru 中）以设置异常警报。

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。
4. 减少警报疲劳： 尽量减少非关键警报。如果存在大量无关紧要的警报，团队将不堪重负，可能导致忽视关键问题，并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. 利用 AWS Chatbot 集成 AWS Chatbot以便将警报转发到 Chime、Microsoft Teams 和 Slack。
8. 基于日志的警报： 使用 日志指标筛选器 （在 CloudWatch 中）来根据特定的日志事件创建警报。
9. 查看并迭代： 定期重新审视和完善警报配置。

实施计划的工作量级别： 中。

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：

- 使用 Amazon CloudWatch 警报
- 创建复合警报
- 基于异常检测创建 CloudWatch 警报
- DevOps Guru 通知
- X-Ray Insights 通知
- 使用交互式 ChatOps 对 AWS 资源进行监控、操作和故障排除
- Amazon CloudWatch 集成指南 | PagerDuty
- 将 OpsGenie 与 Amazon CloudWatch 集成

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。

期望的结果： 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。

建立此最佳实践的好处：

- 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暴露的风险等级： 中

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 中创建一个新控制面板，给它起一个描述性名称。
2. 使用 Markdown 小部件： 在深入研究指标之前，请使用 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 Logs 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 警报 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 以分析高基数字段，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。它们可以从各种数据源中提取数据，也可以以独特方式表示数据。
9. 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
- 使用 Amazon CloudWatch 控制面板

相关视频：

- 创建跨账户和跨区域 CloudWatch 控制面板
- AWS re:Invent 2021 - Gain enterprise visibility with AWS Cloud operation dashboards

相关示例：

- 可观测性研讨会
- 使用 Amazon CloudWatch 进行应用程序监控

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 针对每个提醒设置一个流程
- OPS10-BP03 根据业务影响确定运营事件的优先顺序
- OPS10-BP04 定义上报路径
- OPS10-BP05 定义中断时的客户沟通计划
- 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

贵组织拥有处理事件、意外事件和问题的流程。事件 是在工作负载中发生但可能不需要干预的事情。意外事件 是需要干预的事件。 问题 是需要干预或无法解决的反复发生的事件。您需要一些流程来减轻这些事件对业务的影响，并确保做出适当的响应。

当您的工作负载发生意外事件和问题时，您需要一些流程来处理它们。您将如何与利益相关者沟通事件的状态？ 谁负责监督领导应对工作？ 您用什么工具来减轻事件的影响？ 这些是您建立可靠的响应流程所需回答的一些问题的例子。

这些流程必须记录在一个中央位置，并可供参与您工作负载的任何人使用。如果您没有中央 Wiki 或文档存储区，可以使用版本控制存储库。随着流程的发展，您将不断更新这些计划。

接下来将需要对问题进行自动化。这些事情占用了您的时间，限制了您的创新能力。首先构建一个可重复的流程来缓解问题。随着时间的推移，将重点放在自动化缓解或修复根本问题上。这样就可以腾出时间来改进您的工作负载。

期望结果： 贵组织拥有处理事件、意外事件和问题的流程。这些流程被记录下来并存储在一个中央位置。它们随着流程的更改而更新。

常见反模式：

- 周末发生了一起意外事件，值班工程师不知道该怎么办。
- 一位客户向您发送一封电子邮件，说应用程序关闭了。您重新启动服务器以修复该问题。这种情况经常发生。
- 有一起意外事件，多个团队独立工作，试图解决该问题。
- 部署发生在您的工作负载中，而不会被记录下来。

建立此最佳实践的好处：

- 您有一条关于工作负载中事件的审计跟踪。
- 从意外事件中恢复的时间缩短了。
- 团队成员能够一致地解决意外事件和问题。
- 调查意外事件时，大家更加团结一致。

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。

客户示例

AnyCompany Retail 的内部 Wiki 中有一部分专门用于事件、意外事件和问题管理的流程。所有事件均发送至 Amazon EventBridge。问题在 AWS Systems Manager OpsCenter 中被识别为 OpsItems，并按优先级进行修复，减少了无差别的劳动。当流程发生变化时，它们会在内部 Wiki 中进行更新。他们使用 AWS Systems Manager Incident Manager 来管理意外事件并协调缓解工作。

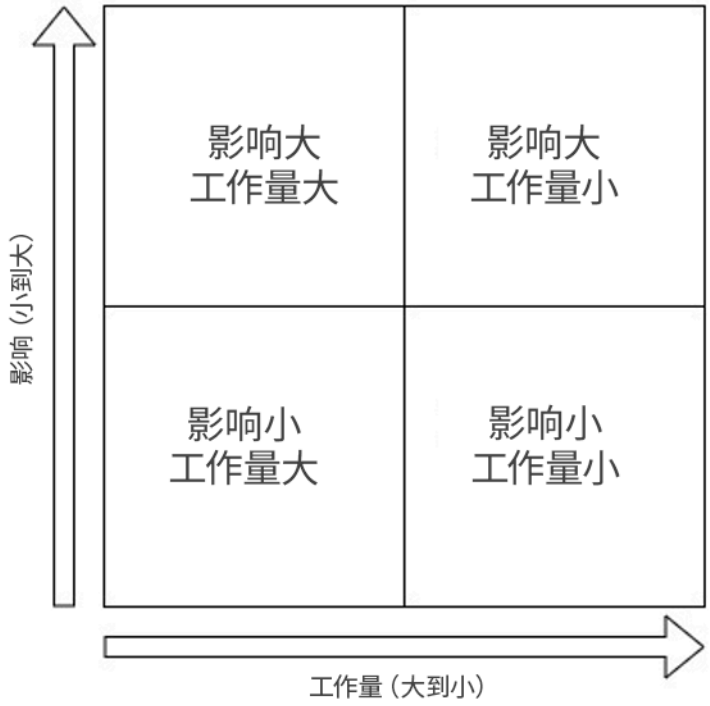
实施步骤

1. 事件
    - 跟踪工作负载中发生的事件，即使不需要人工干预。
    - 与工作负载利益相关者合作，制定一份应跟踪的事件清单。一些示例包括已完成的部署或成功的修补。
    - 您可以使用 Amazon EventBridge 或 Amazon Simple Notification Service 之类的服务生成自定义事件以进行跟踪。
2. 意外事件
    - 首先要确定意外事件的沟通计划。必须告知哪些利益相关者？ 您将如何让他们了解情况？ 谁负责监督协调工作？ 我们建议建立一个内部聊天渠道进行沟通和协调。
    - 为支持您工作负载的团队定义上报路径，特别是在团队没有随时待命的轮换情况下。根据您的支持级别，您还可以向 AWS Support 提交工单。
    - 创建一个调查该意外事件的行动手册。这应该包括沟通计划和详细的调查步骤。在您的调查中包括检查 AWS Health Dashboard 。

- 记录意外事件响应计划。沟通意外事件管理计划，以便内部和外部客户了解参与规则以及对他们的期望。就使用方法对您的团队成员进行培训。
- 客户可以使用 Incident Manager 来建立和管理他们的意外事件响应计划。
- 企业支持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。

3. 问题

- 必须在您的 ITSM 系统中识别和跟踪问题。
- 确定所有已知问题，并根据修复工作量和对工作负载的影响来确定它们的优先级。



- 先解决影响大、工作量小的问题。一旦这些问题得到解决，就继续处理那些属于“影响小且工作量小”象限的问题。
- 随着您的工作负载增长和扩展，您可以使用 Systems Manager OpsCenter 来识别这些问题，为它们附上运行手册，并跟踪它们。

实施计划的工作量级别： 中。您需要一个流程和工具来实施这种最佳实践。记录您的流程，让与工作负载相关的任何人都可以使用它们。经常更新它们。您建立了一个管理问题、缓解问题或解决问题的流程。

资源

相关最佳实践：

- OPS07-BP03 使用运行手册执行程序：已知问题需要一个相关的运行手册，以使缓解工作保持一致。
- OPS07-BP04 根据行动手册调查问题：必须使用行动手册对意外事件进行调查。
- OPS11-BP02 在意外事件发生后执行分析：从意外事件中恢复之后，务必要进行事后分析。

相关文档：

- Atlassian - DevOps 时代的意外事件管理
- AWS 安全意外事件响应指南
- 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 re:Invent 2020：分布式组织中的意外事件管理
- AWS re:Invent 2021 - 使用事件驱动型架构构建下一代应用程序
- AWS 支持您 | 探讨事件管理桌面练习
- AWS Systems Manager Incident Manager - AWS 虚拟研讨会
- AWS 后续举措主讲 Incident Manager | AWS 事件

相关示例：

- AWS 管理和监管工具研讨会 - OpsCenter
- AWS 主动式服务 – 意外事件管理研讨会
- 使用 Amazon EventBridge 构建事件驱动型应用程序
- 在 AWS 上构建事件驱动型架构

相关服务：

- Amazon EventBridge
- Amazon SNS
- AWS Health Dashboard
- AWS Systems Manager Incident Manager

- AWS Systems Manager OpsCenter

OPS10-BP02 针对每个提醒设置一个流程

针对引发提醒的任何事件制定明确的响应措施（运维手册或管理手册），并明确指定负责人。这样可以确保您及时有效地响应运营事件，并防止可以针对其采取措施的事件被不重要的通知所掩盖。

常见反模式：

- 监控系统会向您显示已批准的连接流和其他消息。由于消息量过大，导致您错过了需要干预的周期性错误消息。
- 您收到提醒，指示网站停机。发生这种情况时，没有明确的流程。您被迫采用临时方法来诊断和解决问题。边处理边开发流程会延长恢复时间。

建立此最佳实践的好处： 仅在需要采取措施时发出提醒可以防止低价值提醒遮掩高价值提醒。制定一个可随时采取措施的提醒流程，可以对环境中的事件做出一致而迅速的响应。

未建立此最佳实践暴露的风险等级： 高

实施指导

- 提醒响应流程：对于引发提醒的任何事件，都要制定明确的响应措施（运行手册或管理手册），并明确指定负责其成功完成的负责人（例如个人、团队或角色）。响应的执行可能是自动的，也可能由其他团队完成，但是负责人应负责确保响应流程获得预期的成果。设置这些流程可以确保您及时有效地响应运营事件，并防止可以针对其采取措施的事件被不重要的通知所掩盖。例如，可以实施自动扩展来扩展 Web 前端，但是运营团队应负责确保自动扩展规则和限制符合工作负载需求。

资源

相关文档：

- Amazon CloudWatch 功能
- 什么是 Amazon CloudWatch Events？

相关视频：

- 制定监控计划

OPS10-BP03 根据业务影响确定运营事件的优先顺序

确保在多个事件需要干预时，优先处理对业务最为重要的事件。人身伤亡、经济损失、名誉或信任损害都是一种影响。

常见反模式：

- 您收到一个支持请求，需为用户添加打印机配置。在处理该问题时，您收到一个支持请求，称您的零售网站停机。为您的用户完成打印机配置后，您着手处理网站问题。
- 您收到通知，指示您的零售网站和薪资系统都发生停机。您不知道应该优先处理哪个问题。

建立此最佳实践的好处： 优先响应对业务影响最大的意外事件，可以帮助您管理这种影响。

未建立此最佳实践暴露的风险等级： 中

实施指导

- 根据业务影响确定运营事件的优先顺序：确保在多个事件需要干预时，优先处理对业务最为重要的事件。影响可能包括人身伤亡、经济损失、违规、名誉或信任损害。

OPS10-BP04 定义上报路径

在运维手册和管理手册中定义上报路径，包括触发上报的事件和上报程序。明确指定每项措施的负责人，以便确保有效而及时地响应运营事件。

在采取措施之前，确定何时需要人为决定。与决策者合作，提前做出决策，这样 MTTR 便不会因为等待响应而延长。

常见反模式：

- 您的零售网站停机。您不了解用于网站恢复的运行手册。您开始打电话求助同事。
- 您收到一个关于应用程序无法访问的支持案例。您没有系统管理权限。您不知道谁具有权限。您尝试与创建案例的系统负责人联系，但没有得到响应。您无法联系到系统负责人，而您的同事对此也不太熟悉。

建立此最佳实践的好处： 通过定义上报、上报触发器和上报程序，您可以适当的影响速率系统地向意外事件添加资源。

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

- 定义上报路径：在运维手册和管理手册中定义上报路径，包括触发升级的事件和升级程序。例如，当运维手册无法解决问题或者预定义的时间已经过去时，将问题从支持工程师升级给高级支持工程师。当管理手册无法确定修复路径或者预定义的时间已经过去时，将问题从高级工程师升级给开发团队也是一种正确的升级路径。明确指定每项措施的负责人，以便确保有效而及时地响应运营事件。升级可以涉及第三方。例如某个网络连接提供商或软件供应商。升级可以涉及负责受影响的系统并且获得授权的决策者。

OPS10-BP05 定义中断时的客户沟通计划

定义和测试系统中断时的沟通计划，您可以依赖此计划在中断期间让客户和利益攸关方了解情况。当用户使用的服务受到影响和服务恢复正常时直接传达给用户。

期望结果：

- 为从计划维护到重大意外故障的各种状况（包括调用灾难恢复计划）制定沟通计划。
- 在沟通中，提供有关系统问题的清晰透明信息，帮助客户避免事后猜测其系统的性能。
- 使用自定义错误消息和状态页面减少服务台请求的激增并让用户了解相关情况。
- 定期测试沟通计划，以便确认在真正发生中断时计划会按预期执行。

常见反模式：

- 发生工作负载中断，但您没有沟通计划。因为用户不知晓关于中断的信息，他们不断发出请求，使您的故障单系统不堪重负。
- 您在中断期间向用户发送电子邮件通知。邮件中未包含恢复服务的时间表，因此用户无法针对中断制定计划。
- 您有针对中断的沟通计划，但从未经过测试。发生中断，但因为漏掉了一个本来可以在测试中发现的关键步骤，沟通计划失败。
- 在中断期间，您向用户发送通知，其中包含太多受 AWS 保密协议保护的技术细节和信息。

建立此最佳实践的好处：

- 在中断期间保持沟通可以确保客户可以了解有关问题的进度和预估的解决时间。

- 制定明确的沟通计划，确认您的客户和最终用户可以充分了解情况，以便他们可以采取必要的额外步骤来缓解中断的影响。
- 通过适当的沟通并提高对计划内和计划外中断的认识，您可以提高客户满意度、限制非预期的反应和提高客户保留率。
- 及时和透明的系统中断沟通可以树立信心和建立信任，从而维护您和客户之间的关系。
- 在中断或危机期间，行之有效的沟通策略可以减少猜测和流言蜚语，以免阻碍您恢复运营。

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

在中断期间让您的客户了解情况的沟通计划是全面的，涵盖多个接口，包括面向客户的错误页面、自定义 API 错误消息、系统状态横幅和运行状况状态页面。如果您的系统包括注册用户，您可以通过各种消息传递渠道（例如，电子邮件、短信或推送消息）进行沟通，向客户发送个性化的消息内容。

客户沟通工具

作为第一道防线，Web 和移动应用程序应在中断期间提供友好且信息丰富的错误消息，并能够将流量重定向到状态页面。Amazon CloudFront 是完全托管式内容分发网络（CDN），包括定义和提供自定义错误内容的功能。CloudFront 中的自定义错误页面是很好的第一层客户信息传递，适合组件级中断。CloudFront 还可以简化状态页的管理与激活，以便在计划内或计划外中断期间拦截所有请求。

当中断隔离到不相关联的服务时，自定义 API 错误消息可以帮助检测和减少影响。您可以使用 Amazon API Gateway 为 REST API 配置自定义响应。当 API Gateway 无法访问后端服务时，您可以利用它向 API 使用者提供清晰而有意义的信息。客户消息还可以用于支持中断横幅内容，以及在特定系统功能因服务层中断而降级时发出通知。

直接消息传递是最个性化的客户消息传递类型。Amazon Pinpoint 是适合可扩展多渠道通信的托管服务。您可以利用 Amazon Pinpoint 来构建活动，通过短信、电子邮件、语音、推送通知或您定义的自定义渠道在受影响的客户群中广泛广播消息。当您使用 Amazon Pinpoint 管理消息传递时，消息活动定义明确、可测试，并且可以智能地应用于目标客户群体。建立活动之后，就可以对其作出安排，或通过事件来触发，然后就可以很容易地进行测试。

客户示例

当工作负载受影响时，AnyCompany Retail 向其用户发送电子邮件通知。电子邮件描述了哪些业务功能受到影响，并对何时恢复服务作出合理的预估。此外，他们有一个状态页，其中显示了有关其工作负载运行状况的实时信息。每年在开发环境中对沟通计划测试两次，以便确认计划有效。

实施步骤

1. 确定消息传递策略的沟通渠道。考虑应用程序的架构方面，并确定向客户提供反馈的最佳策略。这包括概述的一个或多个指导策略，包括错误和状态页、自定义 API 错误响应或直接消息传递。
2. 设计应用程序的状态页面。如果您已确定状态或自定义错误页面适合您的客户，则需要为这些页面设计内容和信息。错误页面向用户说明为什么应用程序不可用、什么时候可以再次使用以及在此期间他们可以做什么。如果应用程序使用 Amazon CloudFront，您可以提供自定义错误响应或使用 Lambda at Edge 来转变错误并重写页面内容。CloudFront 还可以将目的地从应用程序内容转换为包含维护或中断状态页面的静态 Amazon S3 内容来源。
3. 为服务设计一组正确的 API 错误状态。当 API Gateway 无法访问后端服务时生成的错误消息以及服务层异常可能不包含适合向最终用户显示的友好消息。您无需对后端服务进行代码更改，而可以将 API Gateway 自定义错误响应配置为将 HTTP 响应代码映射到精选的 API 错误消息。
4. 从业务角度设计信息，使其与系统的最终用户相关，并且不包含技术细节。考虑您的受众并调整信息。例如，您可以引导内部用户转向利用替代系统的解决方法或手动流程。外部用户可能需要等到系统恢复，或订阅更新，以便在系统恢复后收到通知。定义多个场景的已批准消息传递，包括意外中断、计划维护以及会导致特定功能可能降级或不可用的部分系统故障。
5. 使您的客户消息传递模板化和自动化。确立消息内容之后，您可以使用 Amazon Pinpoint 或其他工具使消息传递活动实现自动化。借助 Amazon Pinpoint，您可以为特定的受影响用户创建客户目标细分，并将消息转换为模板。查看 Amazon Pinpoint 教程，了解如何设置消息传递活动。
6. 避免将消息传递功能与面向客户的系统紧密耦合。消息传递策略不应该严格依赖于系统数据存储或服务，以便确认在遇到中断时您可以成功发送消息。考虑培养从多个可用区或区域发送消息以实现消息传递可用性的能力。如果您使用 AWS 服务来发送消息，请利用数据面板操作而不是控制面板操作来调用消息传递。

实施计划的工作量级别：高。制定沟通计划和用于发送计划的机制需要付出巨大的努力。

资源

相关最佳实践：

- OPS07-BP03 使用运行手册执行程序 - 您的沟通计划应具有与之关联的运行手册，这样您的员工就知道如何应对。
- OPS11-BP02 在意外事件发生后执行分析 - 发生中断之后，执行事后分析以确定可防止再次中断的机制。

相关文档：

- Amazon API Gateway 和 AWS Lambda 中的错误处理模式
- Amazon API Gateway 响应

相关示例：

- AWS Health 控制面板
- 弗吉尼亚州北部（US-EAST-1）区域中的 AWS 服务事件的摘要

相关服务：

- AWS Support
- AWS 客户协议
- Amazon CloudFront
- Amazon API Gateway
- Amazon Pinpoint
- Amazon S3

OPS10-BP06 通过控制面板展现状况信息

提供为目标受众（例如内部技术团队、领导和客户）专门设计的控制面板，以传达业务当前的运营状况并提供值得关注的指标。

您可以使用 Amazon CloudWatch 控制面板， 在 CloudWatch 控制台中可自定义的主页上创建控制面板。借助像 Amazon QuickSight 这样的商业智能服务，您可以创建和发布工作负载和运营状况（例如，订单达成率、连接的用户和交易时间）的交互式控制面板。您可以创建控制面板，用来显示指标的系统级和业务级视图。

常见反模式：

- 您根据请求运行有关管理应用程序当前使用情况的报告。
- 在意外事件发生期间，相关系统负责人每二十分钟与您联系一次，想要知道问题是否已解决。

建立此最佳实践的好处： 创建控制面板后，您可以让客户自助访问信息，这样可让他们获知相关信息，并确定是否需要采取措施。

未建立此最佳实践暴露的风险等级： 中

实施指导

- 通过控制面板展现状况信息：提供为目标受众（例如内部技术团队、领导和客户）专门设计的控制面板，以传达业务当前的运营状况并提供值得关注的指标。提供用于获取状态信息的自助选项，可以减少负责处理状态请求的运营团队的中断。示例包括 Amazon CloudWatch 控制面板和 AWS Health Dashboard。
  - CloudWatch 控制面板创建并使用自定义指标视图

资源

相关文档：

- Amazon QuickSight
- CloudWatch 控制面板创建并使用自定义指标视图

OPS10-BP07 自动响应事件

自动响应事件以便减少由手动流程引起的错误，并确保响应及时并且一致。

有多种方法可以在 AWS 上自动执行运行手册和行动手册操作。要响应 AWS 资源中的状态更改事件或您自己的自定义事件，您应创建 CloudWatch Events 规则 以通过 CloudWatch 目标（例如，Lambda 函数、Amazon Simple Notification Service（Amazon SNS）主题、Amazon ECS 任务和 AWS Systems Manager Automation）触发响应。

要响应超过资源阈值的指标（例如，等待时间），您应创建 CloudWatch 警报 以使用 Amazon EC2 操作或 Auto Scaling 操作执行一个或多个操作，或者向 Amazon SNS 主题发送通知。如果您需要执行自定义操作以响应警报，请通过 Amazon SNS 通知调用 Lambda。使用 Amazon SNS 发布事件通知和升级消息，以便让人们了解情况。

AWS 还通过 AWS 服务 API 和 SDK 支持第三方系统。AWS 合作伙伴和第三方提供了许多用于监控、通知和响应的监控工具。其中一些工具包括 New Relic、Splunk、Loggly、SumoLogic 和 Datadog。

您应该保留关键的手动程序，以备在自动程序出故障时使用。

常见反模式：

- 开发人员检查其代码。发生此事件后，本可开始构建然后执行测试，但您没执行任何操作。

- 在停止运行前，您的应用程序记录了一个特定的错误。重新启动应用程序的流程易于理解，可编写成脚本。您可以使用日志事件来调用脚本并重新启动应用程序。否则的话，如果错误发生在星期天凌晨 3 点，您作为负责修复系统的随叫随到的资源，将不得不起床去处理。

建立此最佳实践的好处： 通过自动响应事件，您可以缩短响应时间并减少人工活动中发生的错误。

未建立此最佳实践暴露的风险等级： 低

实施指导

- 自动响应事件：自动响应事件以便减少由手动流程引起的错误，并确保响应及时并且一致。
  - 什么是 Amazon CloudWatch Events？
  - 创建在发生事件时触发的 CloudWatch Events 规则
  - 创建在 AWS API 调用上使用 AWS CloudTrail 触发的 CloudWatch Events 规则
  - 来自支持的服务的 CloudWatch Events 事件示例

资源

相关文档：

- Amazon CloudWatch 功能
- 来自支持的服务的 CloudWatch Events 事件示例
- 创建在 AWS API 调用上使用 AWS CloudTrail 触发的 CloudWatch Events 规则
- 创建在发生事件时触发的 CloudWatch Events 规则
- 什么是 Amazon CloudWatch Events？

相关视频：

- 制定监控计划

相关示例：

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

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
- OPS11-BP02 在意外事件发生后执行分析
- OPS11-BP03 实施反馈环路
- OPS11-BP04 执行知识管理
- OPS11-BP05 确定推动改进的因素
- OPS11-BP06 验证分析结果
- OPS11-BP07 审核运营指标
- OPS11-BP08 记录和分享经验教训
- OPS11-BP09 分配时间进行改进

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。至少每年执行一次工作负载审核。将改进机会优先纳入您的软件开发周期。

期望结果：

- 至少每年都根据架构最佳实践来分析工作负载。

- 在软件开发过程中，为改进机会赋予同等的优先级。

常见反模式：

- 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。
- 改进机会的优先级较低，并停留在待办事项列表中。
- 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。
- 深思熟虑地推进工作负载的演变。
- 您可以利用组织的最佳实践来改善所有工作负载。

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

至少每年一次，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。

客户示例

AnyCompany Retail 的所有工作负载都要经过每年一次的架构审查过程。他们开发了自己的最佳实践清单，适用于所有工作负载。使用 AWS Well-Architected Tool 的自定义剖析功能，他们通过该工具和最佳实践自定义剖析进行审查。通过审查发现的改进机会在他们的软件冲刺中被优先考虑。

实施步骤

1. 至少每年一次，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. 建议您使用您自己内部定义的标准来进行这些审查。如果没有内部标准，建议您使用 AWS Well-Architected Framework。
   b. 您可以使用 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. 客户可以联系他们的 AWS 解决方案架构师，对他们的工作负载进行一次指导式 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 - 事件后分析是改进项目的另一个来源。将学到的经验教训纳入您自己的内部架构最佳实践清单。
- OPS11-BP08 记录和分享经验教训 - 在开发自己的架构最佳实践时，在组织内分享这些最佳实践。

相关文档：

- AWS Well-Architected Tool – 自定义剖析
- AWS Well-Architected 白皮书 - 审核流程
- 使用自定义剖析和 AWS Well-Architected Tool 定制 Well-Architected 审查
- 在组织中实施 AWS Well-Architected 自定义剖析生命周期

相关视频：

- Well-Architected 实验室 - 级别 100：AWS Well-Architected Tool 自定义剖析

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。

常见反模式：

- 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，这也会增加您的操作负担。

建立此最佳实践的好处： 设置预定义的流程，以确定导致意外事件发生的要素、条件、操作和事件，从而帮助您找到改进机会。

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。在适当的情况下向目标受众说明根本原因。

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 评估外部客户需求：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，以便评估各种机会并确定其优先顺序。

在 AWS 上，您可以聚合所有运营活动、工作负载和基础设施的日志，以创建详细的活动历史记录。然后，您可以根据推动因素，使用 AWS 工具分析您在一段时间内的运营状况和工作负载运行状况（例如，确定趋势、将事件和活动与结果相关联，并在环境之间/跨系统进行比较和对比），以发现改进机会。

您应该使用 CloudTrail 跟踪 API 活动（通过 AWS Management Console、CLI、开发工具包和 API），以了解您账户中发生的情况。您可以使用 CloudTrail 和 CloudWatch 跟踪您的 AWS 开发人员工具部署活动。这将向您的 CloudWatch Logs 日志数据添加部署的详细活动历史记录及其结果。

将您的日志数据导出到 Amazon S3 以便长期存储。使用 AWS Glue，您可以在 Amazon S3 中发现并准备您的日志数据以供分析。使用 Amazon Athena，借助其与 AWS Glue 的原生集成来分析您的日志数据。使用像 Amazon QuickSight 这样的商业智能工具，您可以直观显示、浏览和分析您的数据

常见反模式：

- 您有一个脚本，可以正常运行但不完美。您投入时间重新编写。现在，它完美无缺。
- 您的初创公司正试图从风险投资人那里获得另一笔资金。他们希望您证明符合 PCI DSS。您希望让他们满意，因此您记录合规性，但却错过了客户的交付日期，导致客户流失。您没有做错，但是现在您怀疑这样做是否合适。

建立此最佳实践的好处： 确定希望用于改进的标准后，您可以最大程度地减小基于事件的动机或情感投入所带来的影响。

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

- 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 的新增功能
  - 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。
    - AWS 最新安全公告
    - AWS Trusted Advisor
  - 合规性要求：在分析改进机会时，评估保持监管和政策合规性或获取第三方支持所需的更新和更改。
    - AWS 合规性
    - AWS 合规性计划
    - AWS 合规性最新新闻

资源

相关文档：

- Amazon Athena
- Amazon QuickSight
- AWS 合规性
- AWS 合规性最新新闻
- AWS 合规性计划
- AWS Glue
- AWS 最新安全公告
- AWS Trusted Advisor
- 将您的日志数据导出到 Amazon S3
- AWS 的新增功能

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。

常见反模式：

- 您看到某个系统上的 CPU 利用率达到 95%，因此优先考虑寻找一种方法来减少系统上的负载。您认为最佳行动方案是进行扩展。系统是一个转码器，您可对它进行扩展，让它始终以 95% 的 CPU 利用率运行。如果您先与系统负责人联系，他们会向您做出解释。您浪费了时间。
- 系统负责人坚持认为他们的系统是关键任务型系统。系统未置于高安全性环境中。为了提高安全性，您需要对关键任务型系统实施额外的检测和预防控制措施。您通知系统负责人工作已完成，并向他收取额外资源的费用。在收到此通知后的沟通过程中，系统负责人了解到对于关键任务型系统有一个正式定义，而他的系统并不满足。

建立此最佳实践的好处： 通过与业务负责人和主题专家一起验证分析结果，您可以建立共识并更有效地指导改进。

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

- 验证分析结果：与业务负责人和主题专家沟通，以确保对您收集的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。

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。

常见反模式：

- 维护时段导致一次重要的零售促销中断。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。
- 由于使用了组织中常用的错误库，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。如果您定期开会并审核此意外事件，他们应该注意到这种风险。
- 转码器的性能一直在不断下降，这对媒体团队产生了影响。但这还不算多严重。真正糟糕的是，除非情况严重到足以引发意外事件，否则您将难以发现。如果您与媒体团队一起审核运营指标，就有机会发现指标的变化，同时认识到他们的经验并利用这些经验将问题解决。

- 您没有审核对客户 SLA 的满足程度。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。如果您定期开会审核这些 SLA 的指标，您将有机会发现并解决这一问题。

建立此最佳实践的好处： 您可以通过会议定期审核运营指标、事件和意外事件，在团队之间保持共识、分享经验教训，以及确定改进的优先级和目标。

未建立此最佳实践暴露的风险等级： 中

实施指导

- 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。与包括业务、开发和运营团队在内的利益相关方共同分析通过即时反馈和回顾性分析得到的发现，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。
  - Amazon CloudWatch
  - 使用 Amazon CloudWatch 指标
  - 发布自定义指标
  - Amazon CloudWatch 指标和维度参考

资源

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

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。您需要分享信息和资源，以防止出现可避免的错误并简化开发工作。这让您可以专注于交付所需的功能。

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。然后，您应该使用版本受控的 AWS CodeCommit 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。您可以共享对 AMI 的访问权限并授权跨账户使用 Lambda 函数，以此来分享您的计算标准。您还应将您的基础设施标准共享为 AWS CloudFormation 模板。

通过 AWS API 和 SDK，您可以集成外部和第三方工具和存储库（例如，GitHub、BitBucket 和 SourceForge）。在分享您学到的和开发的内容时，请注意设定权限以确保共享存储库的完整性。

常见反模式：

- 由于使用了组织中常用的错误库，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。如果您记录并分享对于此库的经验，他们就可能会注意到风险。
- 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。如果您记录并分享对于此库的经验，他们就可能会注意到风险。
- 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。如果您记录并分享对于此库的经验，他们将有机会加以利用。

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，并最大程度地从经验中获益。

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：设置程序来记录在运营活动执行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：设置程序在不同团队中分享经验教训和相关构件。例如，通过可以访问的 Wiki 共享更新后的程序、指南、管理机制和最佳实践。通过公共存储库共享脚本、代码和库。
    - 授权访问 AWS 环境
    - 共享 AWS CodeCommit 存储库
    - AWS Lambda 函数的简单授权
    - 将 AMI 与特定 AWS 账户共享
    - 利用 AWS CloudFormation Designer URL 快速共享模板
    - 将 AWS Lambda 与 Amazon SNS 配合使用

资源

相关文档：

- AWS Lambda 函数的简单授权
- 共享 AWS CodeCommit 存储库
- 将 AMI 与特定 AWS 账户共享
- 利用 AWS CloudFormation Designer URL 快速共享模板
- 将 AWS Lambda 与 Amazon SNS 配合使用

相关视频：

- 授权访问 AWS 环境

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。

在 AWS 上，您可以创建临时的环境副本，从而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。

常见反模式：

- 您的应用程序服务器中存在一个已知性能问题。它被添加到每个计划内功能实施之后的待办事项中。如果计划功能的添加速率保持不变，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。

建立此最佳实践的好处： 通过在流程中投入专用时间和资源，您可以实现持续增量改进。

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：在流程中抽出专门的时间和资源，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。

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
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