框架

AWS Well-Architected 框架

AWS Well-Architected 框架: 框架

Table of Contents

AWS Well-Architected 框架

发布日期：2024 年 11 月 6 日（文档修订）

AWS Well-Architected Framework 能够帮助您认识到您在 AWS 上构建系统时所作决策的优缺点。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系统的架构最佳实践。

简介

AWS Well-Architected Framework 能够帮助您认识到您在 AWS 上构建系统时所作决策的优缺点。使用该框架有助于您了解在 AWS Cloud 中设计和运行安全、可靠、高效且经济实惠的可持续工作负载的架构最佳实践。它为您提供了一种持续对照最佳实践测评您的架构并确定待改进领域的方法。对架构进行审核的过程是针对架构决策开展的建设性沟通和交流，而不是一种审计机制。我们相信，拥有架构完善的系统能够大大提高实现业务成功的可能性。

AWS 解决方案架构师拥有多年为各种垂直行业和使用案例设计解决方案的经验。我们也已经帮助成千上万客户对其 AWS 之上的架构进行设计与审查。从这些经验中，我们得以总结出在云中设计系统的最佳实践和核心策略。

AWS Well-Architected Framework 囊括了一系列基础性问题，有助于您了解某种架构是否符合云最佳实践。该框架为您提供了一种一致的方法，来对标您所期望的现代云端系统能力，建立一整套质量评估体系，以及评估实现这样的质量需要采取的具体措施。随着 AWS 不断发展，我们将继续与客户协作并增进了解，同时将实际经验融入到 Well-Architected 定义的持续完善当中。

此框架面向各类技术性角色，例如首席技术官（CTO）、架构师、开发人员和运维团队成员。它介绍了可在设计和运行云工作负载时使用的 AWS 最佳实践和策略，提供了进一步实施细节和架构模式的链接。有关更多信息，请参阅 AWS Well-Architected 主页。

AWS 还提供可用于审查您的工作负载的免费服务。AWS Well-Architected Tool（AWS WA Tool）是一种云服务，它提供统一的流程，可帮助您使用 AWS Well-Architected Framework 审核和衡量您的架构。AWS WA 工具为您提供建议，以使您的工作负载变得更可靠、安全、高效且经济有效。

为了帮助您应用最佳实践，我们创建了 AWS Well-Architected Lab，它可以为您提供代码和文档的存储库，让您亲自体验最佳实践的实施。我们还与精选 AWS 合作伙伴网络（APN）合作伙伴合作，它们是 AWS Well-Architected 合作伙伴计划的成员。这些 AWS 合作伙伴拥有丰富的 AWS 知识，可以帮助您审查并改进工作负载。

定义

AWS 的专家每天都在帮助客户设计系统，以利用云中的最佳实践。在设计过程中，我们与您一起对架构进行权衡调整。当您在真实环境中部署这些系统时，我们将关注这些系统的运作状况，同时衡量上述调整的效果。

依托于实践经验，我们构建了 AWS Well-Architected Framework，它为客户和合作伙伴评估架构提供了一系列最佳实践，并提供了相应的可用于评估架构是否符合 AWS 最佳实践的问题。

AWS Well-Architected Framework 建立在六个支柱的基础上，它们分别是卓越运营、安全性、可靠性、性能效率、成本优化和可持续性。

表 1. AWS Well-Architected Framework 的支柱

名称	描述
卓越运营	能够有效地支持发展和运行工作负载，获取对运营的洞察，以及不断改进支持流程和程序以实现业务价值。
安全性	安全支柱介绍了如何利用云技术来保护数据、系统和资产，从而改善您的安全状况。
可靠性	可靠性支柱涵盖相关工作负载按照计划正确而稳定执行其预期功能的能力。这包括在其全部生命周期内运行和测试工作负载的能力。本白皮书深入介绍了有关在 AWS 中实施可靠工作负载的最佳实践指导。
性能效率	有效地使用计算资源以满足系统要求的能力以及在需求变化和技术改进时保持此效率的能力。
成本优化	运行系统以最低价位交付商业价值的能力。
可持续性	这指的是通过最大程度提高预设资源的回报，同时最大程度减少总体资源需求，提高工作负载各组件的效率以减低能源消耗，从而持续改善可持续性影响的能力。

在 AWS Well-Architected Framework 中，我们使用了以下术语：

- 组件是指针对相关需求提供的代码、配置和 AWS 资源的组合。组件通常是技术处理单元，与其他组件分离。
- 工作负载一词是指共同提供业务价值的组件集合。工作负载通常是业务和技术领导者沟通的细节层次。
- 我们将架构定义为组件在工作负载中协同工作的方式。架构图的重点通常是组件如何通信和交互。
- 里程碑记录了架构在整个产品生命周期（设计、实施、测试、上线和生产）中不断演进的关键变化。
- 组织内的技术产品组合是业务运营所需的工作负载集合。
- 工作量可用于对执行任务所需的时间、精力和任务本身的复杂性进行分类。每个组织都需要考虑团队的规模和专业知识以及工作负载的复杂性，了解更多背景信息，以便对组织的工作量进行正确分类。
  - 高：这项工作可能需要数周或数月。它可以分解为若干案例、发布和任务。
  - 中：这项工作可能需要数天或数周。它可以分解为若干发布和任务。
  - 低：这项工作可能需要数小时或数天。它可以分解为若干任务。

在设计工作负载时，您会基于您的业务环境在各个支柱之间做出权衡。这些业务决策可以推动您的工程优先事务。在开发环境中，您可能会进行优化，牺牲一部分可靠性来改进可持续性影响并降低成本；而对于任务关键型解决方案，您可能会在成本和可持续性影响方面做出妥协，来提高可靠性。在电子商务解决方案中，性能可能会影响收入和客户的购买偏好。安全性和卓越操作通常不能与其他支柱交换。

关于架构

在本地环境中，客户通常有一个技术架构中心团队，来监督其他产品或功能团队，从而确保他们遵循最佳实践。技术架构团队通常包含一组角色，比如技术架构师（基础设施）、解决方案架构师（软件）、数据架构师、网络架构师和安全架构师。这些团队一般将 TOGAF 或 Zachman Framework 作为企业架构能力的一部分。

在 AWS，我们倾向于将能力分配到多个团队，而不是只让一个核心团队具有这种能力。当您选择分配决策权限时，会存在一定的风险，例如，确保团队达到内部标准。我们通过两种方法降低这些风险。第一，我们制定实践（即行为方式、流程、标准和公认的规范），专注于让每个团队都具有这种能力，并且我们通过设立专家团队，来确保团队不断提高以突破新的标准。第二，我们实施机制来自动执行检查，以确保满足各项标准。

> “徒有良好的心愿没有用，需要良好的机制来实现它们”– 杰夫·贝索斯（Jeff Bezos）。

这意味着用机制（通常是自动的）来替代人类工作，检查是否遵守了规则或流程。这种分布式方法由 Amazon 领导力原则提供支持，在所有角色中建立一种从客户逆推的文化。逆向思维是我们的创新过程的基本组成部分。我们从客户和客户需求出发，定义和指导我们的工作。只有以客户为中心的团队才能开发出真正满足客户需求的产品。

对于架构，这意味着我们希望每个团队都有能力创建架构并遵循最佳实践。为了帮助新团队获得这些能力或帮助现有团队提高其标准，我们创建了一个由首席工程师组成的虚拟社群，这些工程师可以检查现有团队的设计，帮助他们了解 AWS 最佳实践。首席工程师社群旨在让您能够接触和了解最佳实践。例如，通过午间谈话交流如何将最佳实践应用到实例中。这些谈话会被记录下来，用作新团队成员入门材料的一部分。

AWS 最佳实践源于我们在互联网规模上运行成千上万个系统的经验。我们倾向于使用数据定义最佳实践，同时我们还通过首席工程师等主题专家来设定最佳实践。当首席工程师发现新的最佳实践时，他们将以社群的形式确保所有团队遵循这些最佳实践。同时，这些最佳实践还会被正式纳入我们的内部审查流程以及强制性合规机制中。Well-Architected Framework 是面向客户实施我们的内部审查流程，其中将我们在不同领域角色（例如解决方案架构和内部工程团队）中的主要设计思维编制成文。Well-Architected Framework 是一种可扩展的机制，使您能够有效利用现有的经验。

通过首席工程师在社群内分散架构责任的方法，我们相信设计良好的企业架构是由客户的需求驱动的，并且可以付诸实现。通过让技术主管（例如首席信息官或开发经理）针对所有工作负载执行良好架构审查，您能够更好地了解技术栈存在的风险。以此方法，您可以确定不同团队间可以使用的主题，通过机制、培训或午间谈话等方式，让首席工程师可以与多个团队分享他们在特定领域的想法。

一般设计原则

Well-Architected Framework 定义了一系列一般性设计原则，以促进良好的云端设计：

- 停止猜测您的容量需求：如果您在部署工作负载时作出糟糕的容量决策，结果常常造成昂贵的资源闲置或因容量不足而影响性能。利用云计算，这些问题都不复存在。您可以按需使用容量，并自动对容量规模进行横向缩减和扩展。
- 以生产规模进行系统测试：在云中，您可以根据需要创建一套生产规模等级的测试环境，完成测试，然后停用资源。由于测试环境只需在运行时付费，您模拟真实环境的成本仅为本地测试成本的一小部分。
- 在考虑架构实验的情况下实现自动化：通过自动化操作，您可以低成本创建和复制工作负载，避免人力支出。您可以跟踪自动化变更，审核所产生的影响，并在必要时恢复到以前的参数。
- 考虑架构演进：在传统环境中，架构决策通常作为静态的一次性事件实现，在其生命周期内包含几个重要的系统版本。随着业务及其环境继续演进，这些初始决策可能无法适应不断变化的业务能力需

求。在云中，自动化和按需测试能力将显著降低设计变更所带来影响的风险。这使系统能够随时间推移不断演进，以便企业能够不断地发展创新。

- 利用数据驱动架构：在云中，您可以收集有关您的架构选择如何影响工作负载表现的数据。这使您能够基于事实做出如何改进工作负载的决策。您的云基础设施以代码形式存在，因此您可以随着时间的推移，基于这些数据做出明智的架构选择和改进。
- 通过实际演练不断改进：通过定期安排实际演练来模拟生产中的各种事件，测试架构和流程的性能。这样将帮助您了解可以从哪些方面作出改进，并有助于培养组织处理各种事件的经验。

框架的支柱

构建软件系统与建楼很像。如果基础不牢固，结构问题将会破坏整栋大楼的完整性和功能。在设计技术解决方案时，如果忽视卓越运营、安全性、可靠性、性能效率、成本优化和可持续性这六大支柱，就很难构建一个能够满足您的期望和需求的系统。通过把这些支柱整合到架构中，您将能构建稳定而高效的系统，这将使您能够专注于设计的其他方面，例如功能性需求。

支柱

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

卓越运营

卓越运营（OE）是一项承诺，即正确地构建软件，同时持续提供卓越的客户体验。卓越运营支柱包含组织团队、设计工作负载、大规模运营工作负载和随时间推移改进工作负载的最佳实践。

卓越运营支柱概述了各种设计原则、最佳实践和问题。有关具体实施的说明性指导，请参阅《卓越运营支柱白皮书》。

主题

- 设计原则
- 定义
- 最佳实践
- 资源

设计原则

以下是在云中实现卓越运营的设计原则：

- 围绕业务成果组织团队：团队实现业务成果的能力来自领导力愿景、有效的运营和与业务协调的运营模式。领导层应致力于 CloudOps 转型并全身心地投入其中，采用合适的云运营模式，激励团队以

非常高效的方式运营并实现业务成果。正确的运营模式会利用人员、流程和技术能力来扩大规模，优化工作效率，并通过敏捷性、响应能力和适应能力打造差异化优势。组织的长期愿景会转化为一系列目标，并且这些目标将传达给整个组织内云服务的利益相关方和使用者。各个层面的目标和运营 KPI 将保持一致。这种做法能够维持通过实施以下设计原则所获得的长期价值。

- 实施可观测性以获得切实可行的洞察：全面了解工作负载行为、性能、可靠性、成本和运行状况。建立关键绩效指标（KPI），利用可观测性遥测来作出明智的决策，并在业务结果面临风险时迅速采取行动。基于可操作的可观测性数据，主动提高性能和可靠性，降低成本。
- 尽可能安全地实现自动化：在云中，您可以将用于应用程序代码的工程规范应用于整个环境。您能够以代码形式定义整个工作负载及其运营（应用程序、基础设施、配置和程序），并对其进行更新。之后，您可以通过启动工作负载的运营来响应事件，从而实现运营的自动化。在云中，您可以通过配置防护机制（包括速率控制、错误阈值和审批）来实现自动化的安全。通过有效的自动化，您可以实现对事件的持续响应，限制人为错误并减少操作员的艰苦工作。
- 频繁进行可逆的小规模更改：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更改可缩小影响范围，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，为工作负载提供有益的更改。
- 经常优化运营程序：随着工作负载的发展变化，相应地改进运营。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程序是否有效，以及团队是否熟悉这些程序。在发现差距时，相应地更新程序。向所有利益相关方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：通过推动故障场景来了解工作负载的风险状况及其对业务成果的影响，从而最大限度地提高运营成功率。测试程序的有效性以及团队对这些模拟故障作出的反应。制定明智的决策，管理通过测试确定的开放风险。
- 从所有运营事件和指标中吸取经验教训：从所有运营事件和故障中吸取经验教训，推动改进。在多个团队乃至组织范围中分享经验教训。经验教训应重点介绍有关运营如何促进取得业务成果的数据和轶事。
- 使用托管服务：尽可能使用 AWS 托管服务，减少运营负担。围绕与这些服务的交互制定操作程序。

定义

在云中实现卓越运营有四个领域的最佳实践：

- Organization（组织）
- 准备
- 运营

- 改进

组织领导层负责定义业务目标。组织必须了解各项要求和优先事项，并利用它们来组织和开展工作，为取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。实施多项服务来实现工作负载的集成、部署和交付，这将通过自动执行重复流程，为生产环境带来更多有益更改。

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

最佳实践

> **Note**
>
> 所有卓越运营问题都将 OPS 前缀用作支柱的简写。

主题

组织

团队必须对整个工作负载、他们在其中的角色以及共同的业务目标有一致的理解，以便确立优先事项以实现业务成功。明确优先事项可以让您的工作效益最大化。评估内部和外部客户需求，让包括业务、开发和运营团队在内的关键利益相关方参与进来，以便确定工作重心。评估客户需求将确保您充分了解实现业务成果所需的支持。确保了解组织治理规定的指导原则或义务，以及监管合规性要求和行业标准等可能需要遵循或重视的外部因素。验证您是否具有确定内部治理和外部合规性要求更改的机制。如果未确定要求，请验证您是否已对此决定进行尽职调查。定期审查优先事项，以便在需求发生变化时对其进行更新。

评估业务面临的威胁（例如业务风险和负债以及信息安全威胁），并在风险注册表中维护这些信息。评估风险的影响，在有冲突的利益或替代方法之间作出权衡。例如，新功能的加速上市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来简化系统迁移工作，而无需重构。管理益处与风

险，以便在确定工作重心时作出明智的决策。有些风险或选择可能在一段时间内可以接受，或许可以降低相关风险，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。团队必须了解自己在其他团队获得成功的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，并制定共同的目标。了解责任分配、所有权归属、决策制定方式以及决策者，将有助于集中精力并最大限度地发挥团队的优势。团队的需求将由其所支持的客户、所在组织、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组织中的所有团队及其工作负载是不合理的。

确保每个应用程序、工作负载、平台和基础设施组件都有确定的负责人，并且每个流程和程序都有确定的负责人负责其定义，有负责人负责其性能。

了解每个组件、流程和程序的业务价值，了解为什么要配置这些资源或为什么要执行这些活动，以及为什么要拥有该所有权，这些都有助于确定团队成员的行动。清晰定义团队成员的责任以便他们可以适当地采取行动，并制定相关机制，确定责任和所有权。制定用于请求添加、更改和例外的机制，以免限制创新。在团队之间定义协议，描述团队之间如何开展合作以相互支持以及您的业务成果。

为团队成员提供支持，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。参与其中的高层领导应设定期望并衡量是否成功。高层领导应是采用最佳实践和组织发展的发起人、倡导者和推动者。允许团队成员在成果面临风险时采取行动以尽可能减少影响，并鼓励他们在认为存在风险时向决策者和利益相关方上报，以便解决问题并避免意外事件。及时、清晰、可行地传达已知风险和计划内事件，以便团队成员可以及时采取适当行动。

鼓励进行试验，以加快学习速度，并使团队成员保持兴趣和参与热情。团队必须增强自己的技能组合，以采用新技术，并随需求和责任的变化继续提供支持。专门安排学习时间，以提供支持并鼓励参与其中。确保团队成员拥有取得成功和进行扩展所需的资源（包括工具和团队成员），以便为您的业务成果提供支持。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。

如果存在适用于组织的外部法规或合规性要求，则应使用 AWS 云合规性提供的资源来帮助培训团队，以便他们能够确定优先事项会受到的影响。Well-Architected Framework 强调学习、衡量和改进。它为您提供了一致的方法来评估架构，并实施将随着时间推移而扩展的设计。AWS 提供了 AWS Well-Architected Tool，可协助您在开发之前审查方法，在生产之前审查工作负载状态，以及在生产过程中审查工作负载状态。您可以将工作负载与最新的 AWS 架构最佳实践进行比较，监控整体状态，并深入了解潜在风险。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，有助于确定您的优先事项。Business Support 和 Enterprise Support 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、成本优化和可持续性，可进一步帮助他们确定优先事项。

AWS 有助于您就 AWS 及其服务对团队进行培训，让他们深入了解自己的选择会如何影响工作负载。使用由 AWS 支持 提供的资源（AWS 知识中心、AWS 讨论论坛和 AWS 支持 中心）和 AWS 文档来

培训团队。请通过 AWS 支持 中心联系 AWS 支持，获取与 AWS 问题有关的协助。AWS 还分享了我们通过在 Amazon Builders' Library 中的 AWS 运营学到的最佳实践和模式。您可以通过 AWS Blog 和 The Official AWS Podcast，获得各种其他有用信息。AWSTraining and Certification 提供了一些培训，可以通过自定进度的数字课程，学习 AWS 的基础知识。您还可以报名参加讲师指导培训，进一步培养团队的 AWS 技能。

使用能够跨 AWS Organizations 等账户集中治理环境的工具或服务，协助管理运营模式。AWS Control Tower 等服务扩展了这一管理功能，让您能够定义账户设置的蓝图（支持您的运营模式），使用 AWS Organizations 进行持续治理以及自动预置新账户。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。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，并让内部团队保持精干，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。

以下问题主要针对卓越运营方面的注意事项。（有关卓越运营问题的列表和最佳实践，请参阅附录。）

OPS 1：如何确定自己的优先事项？

每个人员都必须了解自己在实现业务成功方面所发挥的作用。制定共同的目标，以便为资源设定优先事项。这可以让您的工作效益最大化。

OPS 2：如何构建组织结构来为业务成果提供支持？

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。团队必须了解自己在其他团队获得成功的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，并制定共同的目标。了解责任分配、所有权归属、决策制定方式以及决策者，将有助于集中精力并最大限度地发挥团队的优势。

OPS 3：组织文化如何为业务成果提供支持？

为团队成员提供支持，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。

您可能会发现，您需要在某个时间点侧重于一小部分优先事项。长期使用平衡的方法来确保培养所需能力和管理风险。定期审查优先事项，并根据需求变化进行更新。当责任和所有权不确定或未知时，您将

面临以下风险：没有及时执行必要的活动，以及在处理这些需求时可能出现工作冗余和潜在冲突。组织文化会直接影响团队成员的工作满意度和保留率。提升团队成员的参与度和能力，取得业务成功。创新必须进行试验，才能将创意转化为成果。应认识到，取得非预期结果也算试验成功，因为这种试验发现了无法实现成功的途径。

准备

要为卓越运营做好准备，您必须了解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来支持这些工作负载。

将工作负载设计成能够提供必要的信息，以便您了解其所有组件的内部状态（例如指标、日志、事件和跟踪数据），为可观测性和调查问题提供支持。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洞察。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作出数据驱动型决策，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，而不是被动作出反应。团队可以了解其系统中的因果关系，以此预测和预防问题，而不仅仅是对问题作出反应。随着工作负载的发展变化，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。

采用的方法需能够改进将更改应用于生产环境的流程，并且支持重构、快速质量反馈和错误修复。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或在环境中发现的问题。

采用的方法需能够提供快速质量反馈，并在更改没有达到预期结果时实现快速恢复。使用这些实践可以减轻因部署更改而产生的问题的影响。制定计划以防更改不成功，这样在必要时能够更快速地响应，并测试和验证所做的更改。了解环境中的计划活动，以便管理更改风险，避免影响计划活动。强调频繁、小规模、可逆更改，以限制更改范围。这样可以加快故障排除和修复速度，并支持回滚更改。此外，还意味着能够更频繁地从有价值的更改中获益。

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，了解与工作负载相关的运营风险。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。这也有助于您发现必须制定计划予以解决的任何问题。准备好记录日常活动的运行手册和指导问题解决流程的行动手册。了解益处与风险，以便作出明智的决策，从而将更改应用于生产环境。

AWS 让您能够将整个工作负载（应用程序、基础设施、策略、治理和运营）视为代码。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，并在团队或组织之间共享，提高开发工作的效益。使用云中的运营即代码功能和安全试验功能来开发工作负载、运营程序并进行故障演练。使用 AWS CloudFormation，您可以实现一致的模板化沙盒开发、测试和生产环境，提高运营管理水平。

以下问题主要针对卓越运营方面的注意事项。

OPS 4：如何在工作负载中实现可观测性？

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要求作出数据驱动型决策。

OPS 5：如何减少缺陷、简化修复和改进生产流程？

采用的方法需能够改进将更改应用于生产环境的流程，实现重构、快速质量反馈和错误修复。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。

OPS 6：如何缓解部署风险？

采用的方法需能够提供快速质量反馈，并在更改没有达到预期结果时实现快速恢复。使用这些实践可以减轻因部署更改而产生的问题的影响。

OPS 7：如何知道您已经准备好支持某种工作负载？

评估工作负载、流程及程序和工作人员的操作准备情况，以便了解与工作负载相关的操作风险。

投资实现运营活动即代码，以最大限度地提高运营人员的工作效率，最大限度地降低错误率，并实现自动响应。使用“故障演练”来预测故障，并在适当的时候创建程序。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运营活动。利用云的弹性特点结合相应部署实践，推动开发活动和系统的预部署，以加快实施速度。当您对用于评估工作负载的检查清单进行更改时，请计划要对不再符合条件的活动系统执行哪些操作。

运营

可观测性让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。通过专注于基本洞察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，确保工作负载平稳运行并满足业务需求。

工作负载运营是否成功通过业务成果和客户结果的实现情况加以衡量。定义预期结果、确定成功的衡量方式，并确定将在这些计算中使用的指标，以确定工作负载和运营是否成功。运营状况包括工作负载的运行状况，以及为支持工作负载而执行之运营活动的运行状况和成败（例如，部署和意外事件响应）。设立改进、调查和介入的指标基准，收集和分析您的指标，然后验证您对运营成功的理解及其随时间变化的规律。使用收集的指标来确定您是否可以满足客户需求和业务需求，并确定需要改进的领域。

要实现卓越运营，您需要进行有效且高效的运营事件管理。这适用于计划内和计划外的运营事件。使用已确定的运行手册处理易于理解的事件，并使用行动手册来帮助调查和解决问题。根据对业务和客户的影响，对事件的响应进行优先级排序。确保在出现事件警报时，会有指定负责人运行相关流程。事先定义解决事件所需的人员，并配备一个上报流程，以便根据紧急程度和影响在必要时引入额外人员。确定并引入有权决定行动方案的人员，这些行动方案将对之前未解决的事件响应产生业务影响。

通过为目标受众（例如，客户、业务人员、开发人员、运营人员）定制的控制面板和通知来发布工作负载的运行状态，以便他们可以采取相应措施、管理预期，并在恢复正常运营时收到通知。

在 AWS 中，您可以为收集的工作负载指标和 AWS 自带指标生成控制面板视图。您可以利用 CloudWatch 或第三方应用程序来汇总和呈现运营活动的业务、工作负载和运营级别视图。AWS 通过日志记录功能（包括 AWS X-Ray、CloudWatch、CloudTrail 和 VPC 流日志）提供工作负载洞察，从而协助发现工作负载问题，以支持根本原因分析和修复。

以下问题主要针对卓越运营方面的注意事项。

OPS 8：如何在组织中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，全面了解工作负载的性能并有效地解决问题。

OPS 9：如何了解自己的运营状况？

定义、记录和分析运营指标以便了解运营事件，从而采取适当的行动。

OPS 10：如何应对工作负载事件和运营事件？

制定和验证用于响应事件的程序，以便尽可能减少其对工作负载的干扰。

您收集的所有指标都应该与业务需求及其支持的结果相符。为充分理解的事件开发脚本式响应，并自动执行响应以识别事件。

改进

学习、分享和不断改进，以保持卓越运营。将工作周期专用于持续进行渐进式改进。对影响客户的所有意外事件执行意外事件后分析。确定成因和预防措施，以限制或防止再次事件发生。视情况与受影响的团体沟通成因。定期评估并优先处理改进机会（例如，功能请求、问题修复和合规性要求），包括工作负载和运营程序。

将反馈环路纳入您的程序，以快速确定需要改进的领域，并从正在执行的运营中获取经验教训。

在团队中分享得到的经验教训和其中的效益。分析经验教训中的趋势，并对运营指标进行跨团队回顾性分析，以确定改进的机会和方法。实施更改以便改进，并评估结果以确定是否成功。

在 AWS 上，您可以将日志数据导出到 Amazon S3 或将日志直接发送到 Amazon S3，以便长期存储。使用 AWS Glue，您可以在 Amazon S3 中发现并准备日志数据以供分析，并将相关元数据存储在 AWS Glue Data Catalog 中。然后，Amazon Athena 通过与 AWS Glue 的原生集成，可用于分析日志数据，并使用标准 SQL 进行查询。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。发现可能推动改进的相关趋势和活动。

以下问题主要针对卓越运营方面的注意事项。

OPS 11：如何改进运营？

分配专门的时间和资源用于近乎持续的渐进式改进，以便提高运营的有效性和效率。

运营的成功改进建立在以下基础上：频繁的小规模改进；提供安全的环境和时间来试验、开发和测试改进；以及鼓励人们从失败中获取经验教训的整体氛围。随着运营控制水平的提高，对于沙盒、开发、测试和生产环境的运营支持促进了开发，并提高了对生产环境中部署的变更结果成功与否的可预测性。

资源

请参阅以下资源，详细了解卓越运营的最佳实践。

文档

- DevOps 和 AWS

白皮书

- 卓越运营支柱

视频

- Amazon 的 DevOps

安全性

安全性支柱包括保护数据、系统和资产以利用云技术来改善安全性的能力。

安全性支柱概述了设计原则、最佳实践和问题。有关具体实施的说明性指导，请参阅《安全性支柱白皮书》。

主题

- 设计原则
- 定义
- 最佳实践
- 资源

设计原则

在云中，有很多原则可帮助您提高工作负载的安全性：

- 实施强大的身份验证基础：实施最低权限原则，并通过对每一次与 AWS 资源之间的交互进行适当授权来强制执行职责分离。集中进行身份管理，并努力消除对长期静态凭证的依赖。
- 保持可追溯性：实时监控和审查对环境执行的操作和更改并发送警报。为系统集成日志和指标收集功能，以自动调查并采取行动。
- 在所有层面应用安全措施：利用多种安全控制措施实现深度防御。应用到所有层面（例如网络边缘、VPC、负载均衡、每个实例和计算服务、操作系统、应用程序和代码）。
- 自动化安全性最佳实践：借助基于软件的自动化安全机制，您能够以更为快速且更具成本效益的方式实现安全扩展。创建安全架构，包括实施可在版本控制模板中以代码形式定义和管理的控制措施。
- 保护传输中数据和静态数据：按敏感程度对数据进行分类，并根据情况采用加密、令牌化和访问控制等适当机制。

- 限制对数据的访问：利用相关机制和工具来减少或消除对于直接访问或手动处理数据的需求。这样可以降低处理敏感数据时数据处理不当、被修改以及人为错误的风险。
- 做好应对安全性事件的准备工作：制定符合组织要求的事件管理和调查政策和流程，做好应对意外事件的准备工作。开展意外事件响应模拟演练，并使用具有自动化功能的工具来提高检测、调查和恢复的速度。

定义

在云中实现安全性有七个领域的最佳实践：

- 安全基础知识
- Identity and Access Management
- 检测
- 基础设施保护
- 数据保护
- 事件响应
- 应用程序安全性

在构建任何工作负载之前，您需要确定可能影响安全性的实践。您需要控制谁可以执行什么操作。另外，您需要能够识别安全事件、保护系统和服务，并通过数据保护机制来保持数据的机密性和完整性。您应该具备一个定义明确且经过实践的流程来响应安全事件。这些工具和技术非常重要，因为它们有助于实现诸如避免财务损失或履行监管义务等目标。

借助 AWS 责任共担模式，组织能够利用云服务实现其安全性与合规性目标。AWS 负责保护用于支持云服务的基础设施的安全，作为 AWS 的客户，您能够专注于使用这些服务来实现目标。还可通过 AWS 云更好地访问安全数据，并以自动化方式响应安全事件。

最佳实践

主题

- 安全性
- Identity and Access Management
- 检测
- 基础设施保护

- 数据保护
- 事件响应
- 应用程序安全性

安全性

以下问题主要针对安全性方面的注意事项。（有关安全性问题的列表和最佳实践，请参阅附录。）

SEC 1：如何安全地运行工作负载？

为了安全地操作您的工作负载，您必须将纲领性最佳实践应用于每个安全领域。将您在组织和工作负载级别的卓越运营中定义的要求和流程应用于所有领域。

及时了解 AWS 提供的建议、行业资源以及威胁情报，可帮助改进您的威胁模型和控制目标。通过自动化安全流程、测试和验证，您可以扩大安全运营规模。

在 AWS 中，建议根据账户的功能和合规性或数据敏感性要求分割不同的工作负载。

Identity and Access Management

身份和访问管理是信息安全计划的关键部分，可以确保只有经过授权和通过身份验证的用户和组件才能访问您的资源，并且只能以您要求的方式进行访问。例如，您应该定义一些主体（即可以在您的账户中执行操作的账户、用户、角色和服务）、创建与这些主体相匹配的策略，并实施严格的凭证管理。这些权限管理元素构成了身份验证和授权的核心。

在 AWS 中，权限管理主要由 AWS Identity and Access Management（IAM）服务提供支持，您可以使用该服务控制对 AWS 服务和资源的用户和编程访问。您应该应用细粒度的策略向用户、组、角色或资源分配权限。您还可以应用强密码原则（例如复杂程度）来避免重复使用并强制执行多重身份验证（MFA）。您可以将联合身份验证与现有的目录服务配合使用。对于需要系统接入 AWS 的工作负载，IAM 支持通过角色、实例配置文件、身份联合验证和临时凭证进行安全访问。

以下问题主要针对安全性方面的注意事项。

SEC 2：如何管理人员和计算机的身份？

在操作安全 AWS 工作负载时，您需要管理两类身份。了解您需要管理和授予访问权限的身份类型，有助于确保正确的身份能够在正确的条件下访问正确的资源。

SEC 2：如何管理人员和计算机的身份？

人类身份：您的管理员、开发人员、操作员和最终用户需要身份才能访问您的 AWS 环境和应用程序。他们是您组织的成员或与您合作的外部用户，他们通过 Web 浏览器、客户端应用程序或交互式命令行工具与您的 AWS 资源进行交互。

机器身份：您的服务应用程序、操作工具和工作负载需要一个身份来向 AWS 服务发出请求以执行某种操作，例如读取数据。这些身份包括运行在您的 AWS 环境（例如 Amazon EC2 实例或 AWS Lambda 函数）中的计算机。您还可以为需要访问权限的外部各方管理计算机身份。此外，您还可以拥有位于 AWS 以外且需要访问您的 AWS 环境的计算机。

SEC 3：如何管理人员和计算机的权限？

管理权限以控制对需要访问 AWS 和工作负载的人员和机器身份的访问。权限用于控制哪些人可以在什么条件下访问哪些内容。

凭证不得在任何用户或系统之间共享。应使用最低权限原则授予用户访问权限，并采用密码要求和强制执行 MFA 等最佳实践。应使用临时凭证和有限权限凭证（例如 AWS Security Token Service 发放的凭证）来执行编程访问（包括对 AWS 服务的 API 调用）。

如果用户需要在 AWS Management Console 之外与 AWS 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 AWS 的用户类型。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI 以使用 AWS IAM Identity Center。

哪个用户需要编程式访问权限？	目的	方式
		• 有关 AWS SDK、工具和 AWS API 的更多信息，请参阅《AWS SDK 和工具参考指南》中的 IAM Identity Center 身份验证。
IAM	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照《IAM 用户指南》中将临时凭证用于 AWS 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 IAM 用户凭证进行身份验证。 • 有关 AWS SDK 和工具的更多信息，请参阅《AWS SDK 和工具参考指南》中的使用长期凭证进行身份验证。 • 有关 AWS API 的更多信息，请参阅《IAM 用户指南》中的管理 IAM 用户的访问密钥。

AWS 提供了能够帮助您使用身份和访问管理的资源。为了帮助学习最佳实践，请浏览我们关于管理凭证和身份验证、控制人员访问和控制编程访问的动手实验室。

检测

您可以使用检测控制来发现潜在的安全威胁或意外事件。检测控制是治理框架的重要组成部分，并且可以用于支持质量流程、法律或合规性义务，还可以用于威胁识别和响应工作。检测控制分为多种不同类

型。例如，编制资产清单及其详细属性，有助于更有效地制定决策（以及进行生命周期管理），从而帮助建立运营基准。您还可以通过内部审核（是指对信息系统相关的控制措施进行的检查）来确保实践符合策略和要求，并确保您已根据定义的条件设置了正确的自动化警报通知。这些控制措施都是重要的响应手段，可以帮助组织确定和了解异常活动的范围。

在 AWS 中，您可以通过处理可用于审核、自动化分析和触发警报的日志、事件以及监控来实施检测控制。CloudTrail 日志、AWS API 调用和 CloudWatch 可以提供对指标进行监控以及报警的功能，AWS Config 可以提供配置历史记录。Amazon GuardDuty 是一种托管的威胁检测服务，可以持续监控恶意或未经授权的行为，从而帮助您保护 AWS 账户和工作负载。还可以使用服务水平日志，例如，您可以使用 Amazon Simple Storage Service（Amazon S3）来记录访问请求。

以下问题主要针对安全性方面的注意事项。

SEC 4：如何检测和调查安全事件？

从日志和指标中捕获和分析事件以获得可见性。对安全事件和潜在威胁采取行动，从而保护您的工作负载。

日志管理对于架构完善的工作负载至关重要，这其中原因众多，包括安全性或取证、法律或法规要求。分析日志并相应地做出响应至关重要，以便您能够识别潜在的安全事件。借助 AWS 提供的功能，您能够定义数据留存生命周期或定义数据保存、存档或最终删除的位置，从而更轻松地管理日志。这样，您就能够以更简单且更具成本效益的方式进行可预测且可靠的数据处理。

基础设施保护

基础设施保护包括满足最佳实践和组织或监管义务所必需的控制方法（例如深度防御）。使用这些方法对于在云中或本地持续成功运营至关重要。

在 AWS 中，您可以通过使用 AWS 原生技术或使用 AWS Marketplace 提供的合作伙伴产品和服务来实施有状态和无状态数据包检查。您应该使用 Amazon Virtual Private Cloud（Amazon VPC）创建一个安全且可扩展的私有环境，您可以在其中定义拓扑结构，包括网关、路由表以及公有子网和私有子网。

以下问题主要针对安全性方面的注意事项。

SEC 5：如何保护网络资源？

任何以某种形式连接至网络（互联网或专用网络）的工作负载都需要多层防御，以帮助防御基于外部和内部网络的威胁。

SEC 6：如何保护计算资源？

工作负载中的计算资源需要多层防御，以帮助抵御外部和内部威胁。计算资源包括 EC2 实例、容器、AWS Lambda 函数、数据库服务、IoT 设备等。

在任何类型的环境中，我们都建议使用多层防御。在基础设施保护方面，许多概念和方法在跨云和本地模型中都有效。实施边界保护、监控入站点和出站点以及建立全面的日志记录、监控和警报机制对于制定有效的信息安全计划至关重要。

AWS 客户能够定制或加强 Amazon Elastic Compute Cloud（Amazon EC2）、Amazon Elastic Container Service（Amazon ECS）容器或 AWS Elastic Beanstalk 实例的配置，并将配置捆绑到不可变的亚马逊机器映像（AMI）。之后，无论是由 Auto Scaling 启动还是手动启动，使用此 AMI 启动的所有新虚拟服务器（实例）都会收到加强的配置。

数据保护

在构建任何系统之前，您应确定可能影响安全性的基本实践。例如，数据分类提供了一种基于敏感程度对组织数据进行分类的方法，加密通过让未经授权的用户无法获知数据的真正内容来保护数据。这些工具和技术非常重要，因为它们有助于实现诸如避免财务损失或履行监管义务等目标。

在 AWS 中，以下实践有助于保护数据：

- 作为 AWS 客户，您拥有对自己的数据的完全控制权。
- AWS 可帮助您更轻松地加密数据和管理密钥（包括定期密钥轮换），这些操作可以由 AWS 轻松自动执行，也可由您执行。
- 我们还提供包含文件访问和更改等重要内容的详细日志记录。
- AWS 设计的存储系统具有优异的韧性。例如，Amazon S3 Standard、S3 Standard–IA、S3 One Zone-IA 和 Amazon Glacier 都设计为可以在一年内实现 99.999999999% 的对象持久性。这一持久性级别相当于平均每年有 0.000000001% 的对象丢失。
- 作为较大规模数据生命周期管理流程中的一部分，版本控制可以防止意外覆盖、删除数据和类似损害。

- AWS 永远不会主动在区域之间移动数据。除非您明确使用相关功能或利用提供该功能的服务移动数据，否则放置在某个区域中的内容将保留在该区域中。

以下问题主要针对安全性方面的注意事项。

SEC 7：如何对数据进行分类？

分类提供了一种基于重要程度和敏感度对数据进行分类的方法，以帮助您确定适当的保护和保留控制措施。

SEC 8：如何保护静态数据？

通过实施多种控制措施来保护静态数据，以降低未经授权的访问或处理不当的风险。

SEC 9：如何保护传输中数据？

通过实施多种控制措施来保护传输中数据，以降低未经授权的访问或丢失的风险。

AWS 提供了多种加密静态数据和传输中数据的方法。我们将这些功能内置在我们的服务中，这样您可以更轻松地加密数据。例如，我们为 Amazon S3 实施了服务器端加密（SSE），这样您可以更轻松地以加密方式存储数据。您还可以将整个 HTTPS 加密和解密过程（通常称为 SSL 终端）交给弹性负载均衡（ELB）来完成。

事件响应

即使采用极为成熟的预防和检测控制机制，组织仍应制定相关流程来响应安全事件并缓解安全事件可能带来的影响。工作负载的架构会极大地影响团队在意外事件发生期间采取有效行动、隔离或约束系统并将运行状态恢复到已知良好状态的能力。在安全事件发生之前确保相关工具和访问权限部署到位，而后通过 GameDay 活动定期进行意外事件响应演练，将有助于确保您的架构能够及时进行调查和恢复。

在 AWS 中，以下实践有助于做出有效的意外事件响应：

- 我们提供包含文件访问和更改等重要内容的详细日志记录。
- 事件可以被自动处理，并且会启动通过使用 AWS API 自动做出响应的工具。

- 您可以使用 AWS CloudFormation 预置工具和“洁净室”。这样您就可以在安全且隔离的环境中进行取证。

以下问题主要针对安全性方面的注意事项。

SEC 10：如何预测、响应事件以及从意外事件中恢复？

准备工作对于及时有效地调查、响应安全事件以及从安全事件中恢复至关重要，可以尽可能减少对组织的破坏。

确保您能够快速授予安全团队访问权限，而且系统可以自动隔离实例并自动捕获数据与状态信息用于取证。

应用程序安全性

应用程序安全（AppSec）介绍了如何设计、构建和测试所开发工作负载的安全属性的整个过程。组织中应该有经过适当培训的人员，了解构建和发布基础设施的安全属性，并使用自动化来发现安全问题。

在软件开发生命周期（SDLC）和发布后流程的常规部分采用应用程序安全测试，有助于确保您拥有一种结构化机制来发现、修复和防止应用程序安全问题进入生产环境。

在设计、构建、部署和操作工作负载时，应用程序开发方法应该包括安全控制机制。在此过程中，协调流程以持续减少缺陷并尽可能减少技术债务。例如，在设计阶段使用威胁建模有助于及早发现设计缺陷，这使得缺陷更易于修复，修复的成本更低，而不是等到以后再来缓解这些缺陷。

在 SDLC 中，越早的阶段，解决缺陷的成本和复杂性通常就会越低。解决问题最简单的方法就是从一开始就不要有问题，所以从威胁模型开始有助于您在设计阶段专注于实现正确的结果。随着 AppSec 计划日渐成熟，您可以增加使用自动化执行的测试数量，提高向构建者提出的反馈的准确性，并减少安全审查所需的时间。所有这些操作都可以提高所构建软件的质量，并加快将新功能推向生产环境的速度。

这些实施指南侧重于四个方面：组织和文化、管道的安全性、管道中的安全性以及依赖关系管理。每个方面都提供了一组可以实施的原则，并提供了有关如何设计、开发、构建、部署和操作工作负载的端到端视图。

在 AWS 中，可以使用很多方法来处理应用程序安全计划。其中有些方法依赖于技术，而有些方法侧重于应用程序安全计划的人员和组织方面。

以下问题主要针对应用程序安全方面的注意事项。

SEC 11：如何在整个设计、开发和部署生命周期中纳入并验证应用程序的安全属性？

开展员工培训、执行自动化测试、了解依赖项，并验证各种工具和应用程序的安全属性，有助于降低生产工作负载中出现安全问题的可能性。

资源

请参阅以下资源，详细了解安全方面的最佳实践。

文档

- AWS 云安全性
- AWS 合规性
- AWS 安全博客
- AWS 安全成熟度模型

白皮书

- 安全支柱
- AWS 安全概述
- AWS 风险和合规性

视频

- AWS 安全中心
- 责任共担模式概述

可靠性

可靠性支柱涵盖相关工作负载按照计划正确而稳定执行其预期功能的能力。这包括在其全部生命周期内运行和测试工作负载的能力。本白皮书深入介绍了有关在 AWS 中实施可靠工作负载的最佳实践指导。

可靠性支柱概述了设计原则、最佳实践和问题。有关具体实施的说明性指导，请参阅《可靠性支柱白皮书》。

主题

设计原则

在云中实现可靠性有五个设计原则：

- 自动从故障中恢复：通过监控工作负载的关键性能指标（KPI），您可以在指标超过阈值时启动自动化响应机制。这些 KPI 应该是对业务价值（而不是服务运营的技术方面）的一种度量。这可实现自动发送故障通知和跟踪故障，以及启动解决或修复故障的自动恢复流程。借助更高级的自动化功能，可以在故障发生之前预测和修复故障。
- 测试恢复程序：在本地环境中，经常会通过执行测试来证明工作负载能够在特定场景中正常运作。通常不会利用测试来验证恢复策略。在云中，您可以测试工作负载的故障情况，并验证恢复程序。您可以采用自动化方式来模拟不同的故障，也可以重新建立之前导致故障的场景。此方式可以在实际的故障发生以前揭示您可以测试与修复的故障路径，从而降低风险。
- 横向扩展以提高聚合工作负载的可用性：使用多个小型资源取代一个大型资源，以降低单个故障对整个工作负载的影响。跨多个较小的资源分配请求，确保它们不共用常见故障点。
- 无需预估容量：本地工作负载出现故障的常见原因是资源饱和，即对工作负载的需求超过该工作负载的容量（这通常是拒绝服务攻击的目标）。在云中，您可以监控需求和工作负载利用率，并自动添加或删除资源，以保持更高效的水平来满足需求，而不会出现过度预置或预置不足的问题。虽然还有很多限制，但有些配额是可控的，其他配额也可以管理（请参阅“管理服务配额和限制”）。
- 通过自动化管理变更：使用自动化方式对基础设施进行变更。必须管理的变更包括对自动化的变更，可对其进行跟踪与审查。

定义

在云中实现可靠性有四个领域的最佳实践：

- 基本原理
- 工作负载架构
- 变更管理

- 故障管理

要实现可靠性，您必须从基础入手，而基础是服务配额和网络拓扑适应工作负载的环境。在设计时，分布式系统的工作负载架构必须能够预防与减少故障。工作负载必须处理需求或要求的变化，而且它的设计必须能够检测故障，并自动加以修复。

最佳实践

主题

基本原理

基本要求是指其范围超出单个工作负载或项目的因素。在为任何系统设计架构之前，应确定影响可靠性的基本要求。例如，您必须为数据中心提供足够的网络带宽。

在您使用 AWS 时，这些基本要求中的大部分已经包含在内，并且可以根据需要进行处理。云环境在设计层面拥有几乎无限的资源，因此 AWS 要负责满足对联网和计算容量的需求，让您可以根据需求更改资源大小和分配。

以下问题主要针对可靠性的注意事项。（有关可靠性问题的列表和最佳实践，请参阅附录。）

REL 1：如何管理服务配额和约束？

对于基于云的工作负载架构，存在服务配额（也称为服务限制）。这些限额的存在是为了防止意外调配超出所需的资源，并限制 API 操作的请求率，以保护服务不被滥用。还存在资源限制，例如光缆的比特传输速率或物理磁盘的存储量。

REL 2：如何规划网络拓扑？

工作负载通常存在于多个环境中。其中包括多个云环境（可公开访问和私有），可能还包括您现有的数据中心基础架构。计划必须包括网络注意事项，例如系统内和系统间连接、公有 IP 地址管理、私有 IP 地址管理和域名解析。

工作负载架构

可靠的工作负载始于前期的软件和基础设施设计决策。您的架构选择将影响所有 Well-Architected 支柱的工作负载行为。针对可靠性，您必须遵循特定的模式。

使用 AWS 时，工作负载开发人员可以选择要使用的语言和技术。AWSSDK 通过为 AWS 服务提供特定于语言的 API，省去了复杂的编码过程。通过这些 SDK，以及语言选择，开发人员可以实现此处列出的可靠性最佳实践。开发人员还可以通过以下资料库阅读并了解亚马逊构建和运行软件的方法：Amazon Builders' Library。

以下问题主要针对可靠性的注意事项。

REL 3：如何设计工作负载服务架构？

使用服务导向型架构（SOA）或微服务架构构建高度可扩展的可靠工作负载。服务导向型架构（SOA）可通过服务接口使软件组件可重复使用。微服务架构则进一步让组件变得更小、更简单。

REL 4：如何在分布式系统中进行交互设计以预防发生故障？

分布式系统依靠通信网络来互连组件，例如服务器或服务。即使这些网络中出现数据丢失或延迟情况，您的工作负载也必须可靠运行。分布式系统组件的运行方式不得对其他组件或工作负载产生负面影响。这些最佳实践可以防止故障并缩短平均故障间隔时间 (MTBF)。

REL 5：如何在分布式系统中进行交互设计，从而缓解或承受故障影响？

分布式系统依赖于通信网络实现组件（例如服务器或服务）的互联。尽管这些网络中存在数据丢失或延迟，但是您的工作负载必须可靠运行。分布式系统组件的运行方式不得对其他组件或工作负载产生负面影响。这些最佳实践使工作负载能够承受压力或故障，从中更快地恢复，并且降低此类损坏的影响。其结果是缩短平均恢复时间（MTTR）。

变更管理

您必须提前为工作负载或其环境的更改做好准备，从而实现工作负载的可靠运行。此类更改包括，外部因素施加到工作负载上的更改（如需求高峰），以及内部更改（如功能部署和安全补丁）。

您可以使用 AWS 来监控工作负载的行为，并自动对 KPI 做出响应。例如，您的工作负载可以在某个工作负载的用户增加时，添加更多服务器。您可以控制谁有权进行工作负载变更并审核这些变更的历史记录。

以下问题主要针对可靠性的注意事项。

REL 6：如何监控工作负载资源？

日志和指标是深入了解工作负载运行状况的强大工具。您可以将工作负载配置为监控日志和指标，并在超过阈值或发生重大事件时发送通知。通过监控，您的工作负载可以发现超出低性能阈值和发生故障的情形，从而自动恢复以做出响应。

REL 7：如何设计工作负载，以适应需求变化？

可扩展工作负载提供了自动添加或删除资源的弹性，因此资源在任何给定时间点都非常符合当前需求。

REL 8：如何实施更改？

要部署新功能，必须对更改加以控制，以确保工作负载和操作环境正在运行已知的软件，并以可预测的方式进行修补和替换。如果这些更改不受控制，那么就很难预测这些更改的影响，也很难解决由此产生的问题。

当您构建工作负载来根据需求变化自动添加和删除资源时，这不仅可以提高可靠性，还可以确保业务成功不至于带来额外负担。有了监控功能后，当 KPI 偏离预期标准时，系统会自动向团队发送警报。通过自动记录环境变更，您可以审核并快速发现可能影响可靠性的操作。对变更管理的控制确保您可以实施可提供所需可靠性的规则。

故障管理

在任何具备一定复杂度的系统中，发生故障在意料之中。可靠性要求您的工作负载知晓故障的发生，并采取相应行动以避免对可用性产生影响。工作负载必须既能承受故障，又能自动解决问题。

您可以使用 AWS，发挥自动化优势对监控数据做出响应。例如，当特定指标超过阈值时，您可以启动自动操作来解决问题。此外，与其尝试诊断并修复作为生产环境一部分的失败资源，您可以将其替换为

新的资源，并对被替换的失败资源进行分析。由于云让您能够以低成本构建整个系统的临时版本，您可以使用自动化测试来验证完整的恢复流程。

以下问题主要针对可靠性的注意事项。

REL 9：如何备份数据？

备份数据、应用程序和配置，以满足您对恢复时间目标 (RTO) 和恢复点目标 (RPO) 的要求。

REL 10：如何使用故障隔离来保护工作负载？

故障隔离可将组件或系统故障的影响限制在定义的界限内。通过适当的隔离，界限之外的组件不受故障影响。跨多个故障隔离界限运行工作负载，可以提高工作负载对故障的韧性。

REL 11：如何将工作负载设计为可承受组件故障的影响？

在构建具有高可用性和较短平均恢复时间（MTTR）要求的工作负载时必须考虑到韧性。

REL 12：如何测试可靠性？

在为工作负载采用韧性设计以应对生产压力以后，测试是确保其按设计预期运行，并且提供所预期韧性的唯一方式。

REL 13：如何规划灾难恢复（DR）？

拥有适当的备份和冗余工作负载组件是灾难恢复策略的开始。RTO 和 RPO 是您恢复工作负载的目标。根据业务需求设置这些目标。通过实施策略来实现这些目标，同时考虑工作负载资源和数据的位置和功能。中断概率和恢复成本也是关键因素，有助于了解为工作负载提供灾难恢复的业务价值。

请定期备份数据并测试备份文件，确保您可以从逻辑和物理错误中恢复。管理故障的关键在于自动且频繁地测试工作负载以致其出现故障，然后观察它们如何恢复。请定期执行此操作，并确保在工作负载发

生重大变更后也会启动此测试。主动跟踪 KPI 及恢复时间目标（RTO）和恢复点目标（RPO）以评测工作负载的韧性（特别是在故障测试场景中）。跟踪 KPI 将有助于您发现和减少单点故障。目标是充分测试工作负载恢复流程，确保可以恢复所有数据并继续为客户提供服务，即使面对持续存在的问题也是如此。恢复流程应该与标准生产流程一样完备而有效。

资源

请参阅以下资源，详细了解可靠性的最佳实践。

文档

- AWS 文档
- AWS 全球基础设施
- AWS Auto Scaling: How Scaling Plans Work
- 什么是 AWS Backup？

白皮书

- 可靠性支柱：AWS Well-Architected
- 在 AWS 上实施微服务

性能效率

性能效率支柱涉及高效地使用云资源以满足性能要求的能力，以及在需求变化和技术发展时保持该效率的能力。

性能效率支柱概述了设计原则、最佳实践和问题。有关具体实施的说明性指导，请参阅《性能效率支柱白皮书》。

主题

- 设计原则
- 定义
- 最佳实践
- 资源

设计原则

在云中实现性能效率有五个设计原则：

- 普及先进技术：通过将复杂的任务委派给云供应商，使您的团队更顺利地实施高级技术。与要求您的 IT 团队学习有关托管和运行新技术的知识相比，考虑将新技术作为服务使用是一种更好的选择。例如，NoSQL 数据库、媒体转码和机器学习都是需要专业知识才能使用的技术。在云中，这些技术会转变为团队可以使用的服务，让团队能够专注于产品开发，而不是资源预置和管理。
- 数分钟内实现全球化部署：您可以在全球多个 AWS 区域中部署工作负载，从而以最低的成本为客户提供更低的延迟和更好的体验。
- 使用无服务器架构：借助无服务器架构，您无需运行和维护物理服务器即可执行传统计算活动。例如，无服务器存储服务可以充当静态网站（从而无需再使用 Web 服务器），事件服务则可以实现代码托管。这不仅能够消除管理物理服务器产生的运营负担，还可以借由以云规模运行的托管服务来降低业务成本。
- 提升实验频率：利用虚拟资源和可自动化的资源，您可以使用不同类型的实例、存储或配置来快速进行比较测试。
- 因对系统运作方式了如指掌而采用最合适的工具或系统：了解如何使用云服务，并始终使用最适合工作负载目标的技术方法。例如，在选择数据库或存储方法时考虑数据访问模式。

定义

在云中实现性能效率包括五个方面的最佳实践：

- 架构选择
- 计算和硬件
- 数据管理
- 网络和内容分发
- 流程和文化

采用数据驱动型方法来构建高性能架构。收集架构各方面的数据，从总体设计到资源类型的选择与配置都包括在内。

定期审核您的选择，确保充分利用不断发展的 AWS 云的优势。监控可以确保您随时发现与预期性能的偏差。对架构作出权衡以便提高性能，例如使用压缩或缓存，或放宽一致性要求。

最佳实践

主题

架构选择

针对特定工作负载的最佳解决方案各不相同，而且解决方案通常会结合多种方法。Well-Architected 工作负载会使用多种解决方案，并且允许使用各种不同的功能来提高性能。

我们提供多种类型和配置的 AWS 资源，可让您更轻松地找到最能满足您需求的方法。此外，我们还提供了无法使用本地基础设施轻松实现的选项。例如，Amazon DynamoDB 之类的托管服务可以提供完全托管的 NoSQL 数据库，确保在任何规模下都只会有几毫秒的延迟。

以下问题主要针对性能效率方面的注意事项。（有关性能效率问题的列表和最佳实践，请参阅附录。）

PERF 1：如何为工作负载选择合适的云资源和架构模式？

一个工作负载通常需要采用多种方法才能实现更高效的性能。Well-Architected 系统会使用多种解决方案和功能来提高性能。

计算和硬件

适合特定工作负载的最佳计算方案会因应用程序设计、使用模式和配置设置而有所不同。架构可能会使用不同的计算方案来支持各种组件，并允许使用不同的功能来提高性能。为架构选择错误的计算方案可能会降低性能效率。

在 AWS 中，计算资源有三种形式：实例、容器和函数：

- 实例是虚拟化服务器，因此您只需通过一个按钮或一次 API 调用即可对其功能进行调整。因为云中的资源决策不是固定不变的，所以您可以尝试使用不同的服务器类型。在 AWS，这些虚拟服务器实例具有不同的系列和大小，并且可以提供各种功能，包括固态硬盘（SSD）和图形处理单元（GPU）。

- 容器是一种操作系统虚拟化方法，允许您在资源隔离的流程中运行应用程序及其依赖项。AWS Fargate 是适用于容器的无服务器计算引擎。如果您需要控制计算环境的安装、配置和管理，则可以使用 Amazon EC2。此外，您还可以从多个容器编排平台中进行选择：Amazon Elastic Container Service（ECS）或 Amazon Elastic Kubernetes Service（EKS）。
- 函数从您要应用的代码中抽象出运行环境。例如，AWS Lambda 允许您在不运行实例的情况下运行代码。

以下问题主要针对性能效率方面的注意事项。

PERF 2：如何在工作负载中选择和使用计算资源？

适合工作负载的更高效的计算解决方案会根据应用程序设计、使用模式和配置设置而有所不同。架构可以使用不同的计算解决方案来支持各种组件，并且可以开启各种不同的功能来提高性能。为架构选择错误的计算解决方案可能会降低性能效率。

数据管理

针对特定系统的最佳数据管理解决方案往往取决于数据类型（数据块、文件或对象）、访问模式（随机或连续）、所需吞吐量、访问频率（在线、离线、归档）、更新频率（WORM、动态）以及可用性与持久性限制等因素。Well-Architected 工作负载使用专门构建的数据存储，这些存储允许使用不同的功能来提高性能。

在 AWS 中，存储有三种形式：对象、数据块和文件：

- 对象存储提供了一个可扩展的耐用平台，允许从任何互联网位置访问数据，适用于用户生成的内容、活动存档、无服务器计算、大数据存储或备份以及恢复。Amazon Simple Storage Service（Amazon S3）是一种对象存储服务，提供行业领先的可扩展性、数据可用性、安全性和性能。Amazon S3 的耐用性可达到 99.999999999%（11 个 9），为全球各地的公司存储数百万个应用程序的数据。
- 数据块存储为每个虚拟主机提供高可用性、一致性、低延迟的数据块存储，类似于直连存储（DAS）或存储区域网络（SAN）。Amazon Elastic Block Store（Amazon EBS）旨在满足需要持久性存储的工作负载的需求，此类持久性存储可通过 EC2 实例访问，帮助您根据适合的存储容量、性能和成本对应用程序进行微调。
- 文件存储可以跨多个系统提供对共享文件系统的访问。Amazon Elastic File System（Amazon EFS）等文件存储解决方案非常适合大型内容存储库、开发环境、媒体存储或用户主目录等应用场

景。Amazon FSx 让您可以经济高效地启动和运行热门文件系统，因此您可以利用应用广泛的开源和商业许可文件系统的丰富功能集和快速性能。

以下问题主要针对性能效率方面的注意事项。

PERF 3：如何存储、管理和访问工作负载中的数据？

针对某个系统的更高效存储解决方案往往取决于访问操作类型（数据块、文件或对象）、访问模式（随机或连续）、所需吞吐量、访问频率（在线、离线、归档）、更新频率（WORM、动态）以及可用性与持久性限制等因素。架构良好的系统使用多种存储解决方案，并且可以开启各种不同的功能，以便提高性能和高效地使用资源。

网络和内容分发

适合某个工作负载的最佳网络解决方案会因延迟、吞吐量要求、抖动和带宽而有所不同。物理约束（例如用户资源或本地资源）决定位置选项。这些约束可以通过边缘站点或资源置放来抵消。

在 AWS，网络资源以虚拟化形式存在，而且以多种类型和配置提供。这让您可以更轻松地找到贴合您需求的网络方案。AWS 提供多种产品功能（例如增强联网、Amazon EC2 联网优化实例、Amazon S3 传输加速和动态 Amazon CloudFront）来优化网络流量。AWS 还可以提供多种联网功能（例如 Amazon Route 53 延迟路由、Amazon VPC 端点、AWS Direct Connect 和 AWS Global Accelerator）来减少网络距离或抖动。

以下问题主要针对性能效率方面的注意事项。

PERF 4：如何在工作负载中选择和配置网络资源？

该问题涵盖在云端设计、配置和运行高效的联网和内容分发解决方案的指导和最佳实践。

流程和文化

在最初构建工作负载时，您可以采用一些原则和实践，协助您更好地运行高效、高性能的云工作负载。要采用能提高云工作负载性能效率的文化，请考虑以下关键原则和实践。

要打造这种文化，请考虑以下关键原则：

- 基础设施即代码：使用 AWS CloudFormation 模板之类的方法定义您的基础设施即代码。使用模板，您可以将基础设施与应用程序代码和配置一道放入源代码控制中。这让您能够将用于开发软件的实践应用到基础设施，从而能够快速迭代。
- 部署管道：使用持续集成/连续部署（CI/CD）管道（例如，源代码存储库、构建系统、部署和测试自动化）来部署基础设施。这让您能够以可重复、一致且低成本的方式进行迭代部署。
- 明确定义的指标：设置和监控指标以捕获关键性能指标（KPI）。我们建议您使用技术和业务指标。网站或移动应用程序的关键指标是首个字节捕获时间或渲染时间。其他常规的适用指标包括线程计数、垃圾回收速率以及等待状态。业务指标，如单次请求累计总成本，可以提醒您留意降低成本的方法。仔细考虑解读指标的方式。例如，您可以选择最大值或第 99 个百分位数，而不是平均值。
- 自动性能测试：作为部署过程的一部分，在快速运行测试成功通过后自动启动性能测试。自动化应创建新环境、设置初始条件（如测试数据），然后运行一系列基准和负载测试。这些测试的结果应回绑到构建中，以便您可以随着时间推移跟踪性能变化。对于长时间运行的测试，您可以使管道的这一部分与构建的剩余部分异步进行。或者，您也可以使用 Amazon EC2 竞价型实例来通宵运行性能测试。
- 负载生成：您应该创建复制综合或预先记录的用户旅程的一系列测试脚本。这些脚本应该是幂等的，而不是耦合，您可能需要包含预热脚本以便产生有效结果。测试脚本应尽可能再现生产中的使用行为。您可以使用软件或软件即服务（SaaS）解决方案来生成负载。考虑使用 AWS Marketplace 解决方案和竞价型实例：它们是用于生成负载的经济高效的方法。
- 性能可见性：关键指标应该对您的团队可见，尤其是针对每个构建版本的指标。这让您能够随着时间推移看到所有重大的正面或负面趋势。您还应展示有关错误或异常数量的指标，确保测试的是正常工作的系统。
- 可视化：使用可视化技术，清楚了解出现性能问题、热点、等待状态或低利用率的位置。在架构图上叠加性能指标：调用图表或代码有助于快速发现问题。
- 定期审核流程：通常，不存在或不完整的性能审核流程会导致架构性能不佳。如果您的架构性能不佳，请实施性能审核流程，以便推动迭代改进。
- 持续优化：采用一种文化，不断优化云工作负载的性能效率。

以下问题主要针对性能效率方面的注意事项。

PERF 5：您使用什么流程来提高工作负载的性能效率？

在最初构建工作负载时，您可以采用一些原则和实践，协助您更好地运行高效、高性能的云工作负载。要采用能提高云工作负载性能效率的文化，请考虑以下关键原则和实践。

资源

请参阅以下资源，详细了解性能效率的最佳实践。

文档

- Amazon S3 性能优化
- Amazon EBS 卷性能

白皮书

- 性能效率支柱

视频

- AWS re:Invent 2019: Amazon EC2 foundations (CMP211-R2)
- AWS re:Invent 2019: Leadership session: Storage state of the union (STG201-L)
- AWS re:Invent 2019: Leadership session: AWS purpose-built databases (DAT209-L)
- AWS re:Invent 2019: Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2)
- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)

成本优化

成本优化支柱包括以最低价格运行系统来实现业务价值的能力。

成本优化支柱概述了设计原则、最佳实践和问题。有关具体实施的说明性指导，请参阅《成本优化支柱白皮书》。

主题

- 设计原则
- 定义
- 最佳实践
- 资源

设计原则

在云中实现成本优化有五个设计原则：

- 实施云财务管理：要获得财务上的成功并加速在云中实现业务价值，请投资云财务管理和成本优化。您的组织应当投入时间和资源增强自身在这个新的技术和使用情况管理领域中的能力。与安全性或卓越运营能力类似，您的组织需要通过知识构建、计划、资源和流程来培养能力，从而成为一家具有成本效益的组织。
- 采用消费模式：仅为所需计算资源付费，并可根据业务需求而非复杂的预测增加或减少使用情况。例如，开发和测试环境通常只需要在每个工作日运行八个小时。您可以在不需要时停用这些资源，从而实现 75% 的潜在成本节约（40 小时对比 168 小时）。
- 衡量整体效率：衡量工作负载的业务产出及其交付成本。使用这种衡量方式了解您通过提高产出和降低成本获得的收益。
- 不再把钱花在千篇一律的繁重工作上：AWS 会帮您料理繁重的数据中心运营工作，如安装、堆叠和驱动服务器。它还消除了使用托管服务管理操作系统和应用程序的运营负担。因此，您可以集中精力处理客户和业务项目而非 IT 基础设施。
- 对支出进行分析和归因：使用云，您可以更轻松地确定系统的准确使用情况和成本，从而将 IT 成本透明地分摊到各个工作负载拥有者。这有助于衡量投资回报率（ROI），并让工作负载拥有者能够据此优化资源和降低成本。

定义

在云中实现成本优化包括五个方面的最佳实践。

- 践行云财务管理
- 支出和使用情况意识
- 具有成本效益的资源
- 管理需求和供应资源
- 持续优化

与良好架构框架中的其他支柱一样，成本优化支柱也需要权衡各种因素，例如，是优化上市速度还是优化成本。在某些情况下，最好优化上市速度以便快速上市、交付新功能或按时完成任务，而不是优化预付成本。设计决策有时是在仓促中而非是由数据决定的，并且人们总是倾向于过度补偿“以防万一”，而不是花时间进行基准测试以获得成本最优的部署。这可能会导致过度预置和优化不足的部署。但是，当您必须将资源从本地环境“直接迁移”到云，然后再进行优化时，这是一个合理的选择。通过预先在成本

优化策略中投入适量的精力，您可以确保始终如一地遵守最佳实践，避免不必要的过度预置，从而更轻松地实现云的经济优势。以下部分介绍了一些技巧和最佳实践，可帮助您开始并持续实施工作负载的云财务管理和成本优化。

最佳实践

主题

- 践行云财务管理
- 支出和使用情况意识
- 具有成本效益的资源
- 管理需求和供应资源
- 持续优化

践行云财务管理

采用云后，由于缩短了审批、采购和基础设施部署周期，技术团队的创新速度会更快。要实现业务价值和财务成功，需要实施一种在云中管理财务的新方法。这种方法便是云财务管理，通过实施组织范围的知识构建、计划、资源和流程，在整个组织内培养能力。

许多组织由许多不同的单位构成，而这些单位又具有不同的要务。若能让组织遵循一组商定的财务目标并为组织提供实现这些目标的机制，将会打造一个更高效的组织。一个有能力的组织的创新和构建速度更快，更敏捷，并能够适应任何内部或外部因素。

在 AWS 中，您可以使用 Cost Explorer，也可以选择使用 Amazon Athena 和 Amazon QuickSight 查看成本和使用情况报告（CUR），从而了解整个组织的成本和使用情况。AWSBudgets 可主动发出成本和使用情况通知。AWS 博客提供有关新服务和新功能的信息，以确保您及时了解新发布的服务。

以下问题主要针对成本优化方面的注意事项。（有关成本优化问题的列表和最佳实践，请参阅附录。）

COST 1：如何实施云财务管理？

实施云财务管理有助于组织在 AWS 上优化成本和使用情况并进行扩展，从而实现商业价值和财务成功。

在组建成本优化部门时，需要包括成员并为团队配备 CFM 和成本优化方面的专家。现有的团队成员将了解组织的当前运作方式以及如何快速实施改进。此外，还可以考虑配备拥有辅助或专业技能组合的人员，例如具备分析和项目管理能力的人员。

在组织中树立成本意识时，需要改进现有计划和流程或基于现有计划和流程进行构建。与构建新流程和计划相比，向现有流程和计划增添内容要快得多。这样将更快地取得成果。

支出和使用情况意识

通过云，您可以获得更大的灵活性和敏捷性，从而支持创新以及快速的开发和部署。这样便节省了自建本地基础设施所需的人工环节和时间，包括确定硬件规格、协商报价、管理购买订单、安排发货和部署资源。然而，要实现这种易用性并利用近乎无限的按需容量，我们需要以新方式考虑支出。

很多企业有多个由不同团队运行的系统。将资源成本分摊到各个组织或产品拥有者可以推动更高效的资源使用模式，减少浪费。准确的成本分摊能够帮助您了解哪些产品是真正盈利的，让您能够做出更明智的预算分配决策。

在 AWS 中，您可以使用 AWS Organizations 或 AWS Control Tower 创建账户结构，这种方式不仅实现了分离，而且有助于分配成本和使用。此外，也可以通过资源标记在使用情况和成本中标注业务和组织信息。使用 AWS Cost Explorer 查看您的成本和使用情况，或者使用 Amazon Athena 和 Amazon QuickSight. 创建自定义控制面板和分析。成本和使用情况控制通过 AWS 预算的通知来实现，并使用 AWS Identity and Access Management（IAM）和服务配额进行控制。

以下问题主要针对成本优化方面的注意事项。

COST 2：您如何管理使用情况？

制定各种策略和机制，确保花费适当的成本来达到目标。采用制约与平衡方法，您可以在不超支的情况下进行创新。

COST 3：如何监控使用情况和成本？

建立策略和程序以便监控并适当分配您的成本。这让您能够衡量和改进此工作负载的成本效益。

COST 4：您如何停用资源？

在从项目开始到结束的过程中实施变更控制和资源管理。这有助于您关闭未使用的资源，以便减少浪费。

您可以使用成本分配标记对 AWS 使用情况和成本进行分类并跟踪。当您对 AWS 资源（例如 EC2 实例或 S3 存储桶）应用标记后，AWS 将通过使用情况和成本标记生成成本和使用情况报告。您可以使用代表组织类别的标记（例如成本中心、工作负载名称或拥有者）整理您的多个服务的成本。

确保在成本和使用情况报告和监控中使用正确的详细级别和粒度。要获得大概见解和趋势，请在 AWS Cost Explorer 中使用每日粒度。要更深入地进行分析和检查，请在 AWS Cost Explorer 中使用每小时粒度，或者在 Amazon Athena 和 Amazon QuickSight 中以每小时为粒度查看成本和使用情况报告（CUR）。

结合标记资源和实体生命周期跟踪（员工、项目），您可以确定无法再为组织创造价值而应停用的孤立资源或项目。您可以设置账单提醒，以在预计超支时通知您。

具有成本效益的资源

为工作负载使用合适的实例和资源是节约成本的关键。例如，在小型服务器上运行某个报告需要五个小时，而在另一个两倍成本的大型服务器上运行只需要一个小时。虽然两个服务器提供同样的结果，但小型服务器会逐渐产生更多成本。

良好架构的工作负载会使用最具有成本效益的资源，这样可以产生巨大而积极的经济效益。您还可以使用托管服务降低成本。例如，您可以使用按电子邮件收费的服务，而无需自己维护电子邮件服务器。

AWS 提供各种灵活且具有成本效益的定价选项，您可以从 Amazon EC2 和其他服务获取能最快满足您需求的实例。按需型实例允许按小时支付计算容量的费用，且无需承诺最低用量。节省计划和预留实例与按需定价相比最高可节约 75% 的成本。使用竞价型实例，您可以利用未使用的 Amazon EC2 容量，并且与按需定价相比最高可节约 90% 的成本。竞价型实例适用于以下情况：系统可以容忍使用服务器实例集，其中单个服务器可以动态装卸（例如无状态 Web 服务器）、批处理、高性能计算及大数据场景。

选择合适的服务还可以减少使用情况和降低成本：例如，使用 CloudFront 可以最大限度地减少数据传输成本；或降低成本，例如，使用 Amazon Aurora on RDS 可以消除昂贵的数据库许可成本。

以下问题主要针对成本优化方面的注意事项。

COST 5：您在选择服务时如何评估成本？

Amazon EC2、Amazon EBS 和 Amazon S3 属于基础 AWS 服务。托管服务（如 Amazon RDS 和 Amazon DynamoDB）属于更高级别或应用程序级别的 AWS 服务。通过选择适当的基础服务和托管服务，您可以优化此工作负载，从而降低成本。例如，使用托管服务，您可以节省或消除大部分管理和运营开销，让您有精力从事应用程序和业务相关活动。

COST 6：在选择资源类型、规模和数量时，如何实现成本目标？

确保选择适合当前任务的资源规模和资源数量。选择最经济实惠的资源类型、规模和数量可以尽可能减少浪费。

COST 7：您如何使用定价模式来降低成本？

使用最适合的资源定价模式可以尽可能减少支出。

COST 8：您如何规划数据传输费用？

确保规划并监控数据传输费用，以便制定架构决策，尽可能降低成本。持续以小步迭代的方式进行架构优化可以实现运营成本的大幅降低。

通过在选择服务时考虑成本因素，并使用 Cost Explorer 和 AWS Trusted Advisor 等工具定期检查 AWS 使用情况，您可以主动监控利用率并相应地调整部署。

管理需求和供应资源

在您迁移到云时，您仅为所需内容付费。您可以在需要时供应与工作负载需求匹配的资源，从而降低昂贵且浪费的过度预置需求。还可以通过节流、缓冲区或队列来修改需求，以满足需求并以更少的资源达成目标，从而降低成本，或者在以后使用批处理服务处理需求。

在 AWS 中，您可以自动预置资源来满足工作负载需求。通过使用基于需求或时间的方法进行 Auto Scaling，您可根据需要添加和删除资源。如果您可以预测需求变化，便可以节省更多资金并确保资源与工作负载需求匹配。您可以使用 Amazon API Gateway 实施节流，也可以使用 Amazon SQS 在工作负载中实施队列。这两种方法都允许您修改工作负载组件的需求。

以下问题主要针对成本优化方面的注意事项。

COST 9：如何管理需求和供应资源？

为了工作负载的性能与支出实现平衡，请确保您支付过费用的所有资源都得到利用，并避免出现实例利用率过低的情况。无论是从运营成本（由于过度使用导致性能下降）还是从浪费 AWS 支出（由于过度预置）的角度衡量，利用率指标过高或过低都会对组织产生负面影响。

当进行修改需求和供应资源的设计时，请主动考虑资源使用模式、预置新资源所需要耗费的时间，以及需求模式的可预测性。当管理需求时，确保您具有大小正确的队列或缓冲区，并在所需的时间内响应工作负载需求。

持续优化

AWS 不断发布新服务和功能，因此您最好不断审视现有架构决策，确保其始终最具成本效益。当您的需求发生变化时，请主动停用不再需要的资源、整体服务和系统。

实施新功能或资源类型可以逐步优化您的工作负载，同时最大程度地减少实施变更所需的工作量。这样可不断提高效率，并确保您始终使用最新的技术，从而降低运营成本。您还可以使用新服务替换或向工作负载中添加新组件。这可以显著提高效率，因此必须定期审查您的工作负载，并实施新服务和新功能。

以下问题主要针对成本优化方面的注意事项。

COST 10：如何评估新服务?

AWS 不断发布新服务和功能，因此您最好不断审视现有架构决策，确保其始终最具成本效益。

定期审查部署时，评估更新的服务如何帮助您节省成本。例如，Amazon Aurora on Amazon RDS 可以降低关系数据库的成本。使用无服务器（例如 Lambda）服务，无需操作和管理实例即可运行代码。

COST 11：如何评估工作量成本？

评估云端运营的工作量成本，审查耗时的云运营，并通过采用相关 AWS 服务、第三方产品或自定义工具实现自动化，以减少人力和成本。

资源

请参阅以下资源，详细了解成本优化的最佳实践。

文档

- AWS 文档

白皮书

- 成本优化支柱

可持续性

可持续性支柱侧重于环境影响，尤其是能源消耗和效率，因为它们是构架师在直接采取行动以减少资源使用时依据的重要杠杆。有关具体实施的说明性指导，请参阅《可持续性支柱白皮书》。

主题

- 设计原则
- 定义
- 最佳实践
- 资源

设计原则

实现云中可持续性有六项设计原则：

- 了解您的影响：衡量您的云工作负载的影响并为工作负载的未来影响建模。包括所有影响来源，例如客户使用您的产品所产生的影响，以及产品最终淘汰和停用所产生的影响。通过查看每个工作单元所需的资源和排放量，将生产性输出与云工作负载的总体影响进行比较。使用这些数据来建立关键绩效指标（KPI），评估在降低影响的同时提高生产力的方法，并估计提议的更改随时间的推移所产生的影响。
- 建立可持续性目标：对于每个云工作负载，建立长期可持续性目标，例如减少每个事务所需的计算和存储资源。针对现有工作负载的可持续性改进的投资回报进行建模，并为负责人提供必须投资于可持续性目标的资源。规划增长并构建您的工作负载，以便增长可降低影响强度（以适当的单位衡量，例如每用户或每事务）。目标可帮助您支持您的企业或组织更广泛的可持续性目标、识别回归并确定潜在改进领域的优先级。
- 最大限度提高利用率：适当调整工作负载规模并实施高效设计，确保实现高利用率，最大程度提高底层硬件的能源效率。由于每台主机的基准功耗，两台以 30% 利用率运行的主机的效率低于一台以 60% 利用率运行的主机。同时，减少或尽可能减少空闲资源、处理和存储，以减少支持工作负载所需的总能源。

- 预测并采用更高效的新硬件和软件产品：支持您的合作伙伴和供应商进行上游改进，以帮助您减少云工作负载的影响。持续监控和评估更高效的新硬件和软件产品。设计灵活性以允许快速采用高效的新技术。
- 使用托管服务：在庞大的客户群中共享服务有助于更充分地利用资源，从而减少支持云工作负载所需的基础设施数量。例如，客户可以通过将工作负载迁移到 AWS Cloud 并采用托管式服务（例如用于无服务器容器的 AWS Fargate，AWS 在其中大规模运行并负责其高效运行）来分散电力和网络等常见数据中心组件的影响。使用有助于将影响降至最低的托管式服务，例如使用 Amazon S3 生命周期配置将不经常访问的数据自动移动到冷存储，或使用 Amazon EC2 Auto Scaling 来调整容量以满足需求。
- 减少云工作负载对下游的影响：减少使用服务所需的能源或资源量。减少客户为了使用您的服务而升级其设备的需要。使用设备场进行测试以了解预期影响，并对客户进行测试以了解使用您服务的实际影响。

定义

在云中实现可持续性包括六个方面的最佳实践：

- 区域选择
- 符合需求
- 软件和架构
- 数据
- 硬件和服务
- 流程和文化

云中的可持续性是一项近乎持续的工作，侧重于通过从所预置的资源中获得最大收益并尽力减少所需的总资源，让工作负载的所有组件降低能耗并提高能效。这项工作可能包括最初选择高效的编程语言、采用现代算法、使用高效的数据存储技术、部署大小合适且高效的计算基础设施，以及最大限度地减少对高性能终端用户硬件的需求。

最佳实践

主题

- 数据管理
- 硬件和服务
- 流程和文化

区域选择

为工作负载选择区域会显著影响其 KPI，包括性能、成本和碳足迹。为了提高这些 KPI，您应该根据业务需求和可持续性目标为工作负载选择区域。

以下问题主要针对安全方面的注意事项。（有关可持续性问题的列表和最佳实践，请参阅附录。）

SUS 1：您如何为工作负载选择区域？

为工作负载选择区域会显著影响其 KPI，包括性能、成本和碳足迹。为了提高这些 KPI，您应该根据业务需求和可持续性目标为工作负载选择区域。

符合需求

用户和应用程序使用您的工作负载及其他资源的方式可以帮助您确定改进方面，以实现可持续性目标。扩展基础设施以持续匹配需求，并确认您仅使用了支持用户所需的最少资源。使服务水平与客户需求保持一致。定位资源以限制用户和应用程序使用这些资源所需的网络。删除未使用的资产。为团队成员提供满足其需求的设备，并尽可能降低他们的可持续性影响。

以下问题主要针对可持续性方面的注意事项：

SUS 2：如何将云资源与您的需求相匹配？

用户和应用程序使用您的工作负载及其他资源的方式可以帮助您确定改进方面，以实现可持续性目标。扩展基础设施以持续匹配需求，并确认您仅使用了支持用户所需的最少资源。使服务水平与客户需求保持一致。定位资源以限制用户和应用程序使用这些资源所需的网络。删除未使用的资产。为团队成员提供满足其需求的设备，并尽可能降低他们的可持续性影响。

使用用户负载扩展基础设施：确定利用率低或无利用率的时段，缩减资源以减少过剩容量并提高效率。

根据可持续性目标调整 SLA：定义和更新服务等级协议（SLA），例如可用性或数据留存期，以最大限度地减少支持工作负载所需的资源数量，同时继续满足业务需求。

减少未使用资产的创建和维护：分析应用程序资产（例如预编制的报告、数据集和静态图像）和资产访问模式，以识别冗余、利用率低下的情况和潜在的淘汰目标。整合具有冗余内容的生成资产（例如，具有重叠或共用数据集和输出的月度报告），以减少重复输出时消耗的资源。淘汰未使用的资产（例如，已停售产品的图片）以释放消耗的资源，并减少用于支持工作负载的资源数量。

针对用户位置优化工作负载的地理位置：分析网络访问模式以识别您的客户建立连接的地理位置。选择可减少网络流量必须传输的距离的区域和服务，以减少支持您的工作负载所需的总网络资源。

针对执行的活动优化团队成员资源：优化提供给团队成员的资源，在支持其需求的同时最大程度地降低对可持续性的影响。例如，在利用率高的共享云桌面上，而不是在利用率不高的强力单用户系统上，执行渲染和编译等复杂的操作。

软件和架构

实施用于执行负载平滑和保持已部署资源始终如一的高利用率的模式，以最大限度地减少资源消耗。由于用户行为会随着时间的推移而发生变化，组件可能会因缺乏使用而变得空闲。修改模式和架构以整合未充分利用的组件，从而提高整体利用率。停用不再需要的组件。了解工作负载组件的性能，并优化消耗最多资源的组件。注意客户用来访问您服务的设备，并实施相应的模式以最大限度地减少设备升级需求。

以下问题主要针对可持续性方面的注意事项。

SUS 3：您如何利用软件和架构模式来支持您的可持续性目标？

实施用于执行负载平滑和保持已部署资源始终如一的高利用率的模式，以最大限度地减少资源消耗。由于用户行为会随着时间的推移而发生变化，组件可能会因缺乏使用而变得空闲。修改模式和架构以整合未充分利用的组件，从而提高整体利用率。停用不再需要的组件。了解工作负载组件的性能，并优化消耗最多资源的组件。注意客户用来访问您服务的设备，并实施相应的模式以最大限度地减少设备升级需求。

针对异步和计划作业优化软件和架构：使用高效的软件设计和架构来尽可能减少每个工作单元所需的平均资源。实施可促成均匀的组件利用率的机制，以减少任务之间的空闲资源并最大限度地减少负载峰值的影响。

移除或重构工作负载组件的用处很低或根本没用：监控工作负载活动，以确定一定时间内各个组件的利用率的变化。移除未使用且不再需要的组件，并重构利用率低的组件，以限制资源浪费。

优化消耗最多时间或资源的代码区域：监控工作负载活动以确定消耗最多资源的应用程序组件。优化在这些组件中运行的代码，以最大限度地减少资源使用和提高性能。

优化对客户设备的影响：了解客户用来使用您服务的设备、它们的预期生命周期，以及更换这些组件对财务和可持续性的影响。实施软件模式和架构，以最大限度地减少客户更换和升级设备的需求。例如，使用与旧硬件和操作系统版本向后兼容的代码实现新功能，或管理有效负载的大小，使其不超过目标设备的存储容量。

使用最有效地支持数据访问和存储模式的软件模式和架构：了解数据在工作负载中被使用、被用户使用、被传输和存储的方式。选择相应的技术以最大限度地减少数据处理和存储要求。

数据管理

以下问题主要针对可持续性方面的注意事项。

SUS 4：您如何利用数据管理策略和模式来支持可持续性目标？

实施数据管理实践以减少支持工作负载所需的预置存储，以及使用存储所需的资源。了解您的数据，并使用能够更有效地支持数据的商业价值及其使用方式的存储技术和配置。当需求减少时，将数据移到更高效、性能更低的存储中，并删除不再需要的数据。

实施数据分类策略：对数据进行分类以了解其对业务结果的重要性。使用此信息来确定何时可以将数据移动到更节能的存储，或者何时可以安全删除数据。

使用支持数据访问和存储模式的技术：使用最能支持您的数据访问和存储方式的存储技术，以在支持您的工作负载的同时最大限度地减少预置资源。例如，固态硬盘（SSD）比磁性驱动器更耗能，应该仅用于活跃的数据使用场景。对不常访问的数据使用节能的存档级存储。

使用生命周期策略删除不必要的数据：管理所有数据的生命周期并自动执行删除时间表，以最大限度地减少工作负载的总存储需求。

最大限度地减少数据块存储中的过度预置：要尽可能减少总预置存储，请创建大小分配适合工作负载的数据块存储。随着数据的增长，使用弹性卷扩展存储，而无需调整附加到计算资源的存储大小。定期检查弹性卷并缩小过度预置的卷，以适应当前数据大小。

删除不需要或多余的数据：仅在必要时复制数据，以最大程度地减少消耗的总存储空间。使用备份技术在文件和数据块级别进行重复数据删除。限制使用独立驱动器冗余阵列（RAID）配置，除非需要满足SLA。

使用共享文件系统或对象存储来访问通用数据：采用共享存储和单一事实来源，以避免重复数据删除并降低工作负载的总存储需求。仅在需要时从共享存储中获取数据。分离未使用的卷以释放资源。最大限

度地减少跨网络的数据移动：使用共享存储和访问区域数据存储中的数据，以最大限度地减少支持工作负载数据移动所需的总网络资源。

仅在难以重新创建时备份数据：为了最大限度地减少存储消耗，仅备份具有商业价值或满足合规性要求所必需的数据。检查备份策略并在恢复方案中排除没有价值的临时存储。

硬件和服务

寻找机会，通过更改硬件管理实践来降低工作负载可持续性影响。最大限度地减少预置和部署所需的硬件数量，并为各项工作负载选择最高效的硬件和服务。

以下问题主要针对可持续性方面的注意事项。

SUS 5：您如何选择并使用架构中的云硬件和服务来支持自己的可持续性目标？

寻找机会，通过更改硬件管理实践来降低工作负载可持续性影响。最大限度地减少预置和部署所需的硬件数量，并为各项工作负载选择最高效的硬件和服务。

使用最少的硬件来满足您的需求：通过使用云的功能，您可以对工作负载实施进行频繁更改。在需求变化时更新已部署的组件。

使用影响最小的实例类型：持续监控新实例类型的发布并利用能效改进，包括那些旨在支持特定工作负载（例如机器学习训练和推理以及视频转码）的实例类型。

使用托管服务：托管服务将维持已部署硬件的高平均利用率和可持续性优化的责任转移给 AWS。使用托管服务将服务的可持续性影响分散到服务的所有租户，从而减少您的个人份额。

优化您对 GPU 的使用：图形处理单元（GPU）可能是高功耗的来源，许多 GPU 工作负载是高度可变的，例如渲染、转码以及机器学习训练和建模。仅在需要时运行 GPU 实例，并在不需要时自动停用它们，以最大限度地减少资源消耗。

流程和文化

寻找机会，通过更改开发、测试和部署实践来降低可持续性影响。

以下问题主要针对可持续性方面的注意事项。

SUS 6：您的组织流程如何支持您的可持续性目标？

寻找机会，通过更改开发、测试和部署实践来降低可持续性影响。

采用可以快速引入可持续性改进的运营：在将潜在潜在改进部署到生产环境之前，对其进行测试和验证。在计算改进的潜在未来收益时，考虑测试成本。开发低成本的测试运营，以实现细微的改进。

使您的工作负载处于最新状态：最新的操作系统、库和应用程序可以提高工作负载效率，并促进更高效技术的采用。最新的软件可能还包括更准确地衡量工作负载对可持续性的影响的功能，因为供应商提供的功能是为了满足其自身的可持续性目标。

提高构建环境的利用率：使用自动化功能和基础设施作为代码，在需要时启动预生产环境，并在不使用时将其关闭。一种常见模式是安排与开发团队成员的工作时间相吻合的可用时段。休眠是一个有用的工具，它可以保存状态，并且只在需要时才快速将实例上线。使用具有容量爆增的实例类型、竞价型实例、弹性数据库服务、容器和其他技术，使开发和测试能力与使用相一致。

使用托管式设备场进行测试：使用托管式设备场将硬件制造和资源使用的可持续性影响分散到多个租户。托管式设备场提供多种设备类型，使您能够支持不太受欢迎的较旧硬件，并避免不必要的设备升级对客户可持续性的影响。

资源

请参阅以下资源，了解关于我们的可持续性最佳实践的信息。

白皮书

- 可持续性支柱

视频

- The Climate Pledge

审查流程

必须持续不断对架构进行审查，同时要允许试错，建立良好的研究探索氛围。架构审查本身应该是一个简单流程（数小时，而不是几天），是一种对话，而不是审核。审查架构的目的是找出任何需要解决的关键问题或可以改进之处。审查后应采取一些措施，以改善客户使用工作负载的体验。

正如在“关于架构”部分讨论的那样，团队中的每位成员都应该对架构质量负责。我们建议负责架构的团队利用 Well-Architected Framework 持续审查架构，而不仅仅只是召开一场正式的审查会议。近乎持续的审查使团队成员能够随着架构的演进不断获得知识体系与对架构认识的更新，并在您推出新功能时改进架构。

AWS Well-Architected Framework 高度借鉴了 AWS 在内部审查系统和服务的方式，并与之保持一致。它基于一套可以影响架构方法的设计原则，并确保不会忽略常见于根因分析（RCA）中的那些因素。当内部系统、AWS 产品或客户遇到严重问题时，我们会查看 RCA，寻求改进当前审查流程的可能性。

应在设计阶段早期，针对产品生命周期中的关键里程碑进行审查，以避免难以更改的单向决策，然后在上线日期之前重复次此流程。（许多决策都是可逆的，是双向的。这些决策可以使用轻量级进程。单向决策很难，不可逆，所以在做出决策之前需要更加全面的检查。）进入生产阶段后，您的工作负载会随着您不断添加新功能和更改技术实施而继续演进。工作负载的架构也会随之演进。您必须遵循良好的架构实践，避免出现架构退化。当架构面临重大变更时，您应遵循一套系统健康规范与流程，包括执行 Well-Architected 审查。

如果您想把审查用作一次性快照或独立的衡量方法，则需要确保让所有相关人员参与对话。我们经常发现，通过审查，团队才第一次真正了解他们实施了什么。在审查其他团队的工作负载时，一种有效的方法是围绕架构展开一系列非正式的对话，在此过程中您可以收集到大多数问题的答案。然后通过一两次会议进行跟进，来帮助您理清思路，或深入了解不明确的方面或已感知的风险。

下面建议了一些召开会议需要准备的事项：

- 配有白板的会议室
- 任何图表或设计说明的打印件
- 需要带外研究来获取答案的问题（例如，“是否已启用加密？”）

完成审查后，您应有一个问题清单，并基于您的业务环境来确定这些问题的优先级。您还需要考虑这些问题对团队日常工作的影响。如能及早解决这些问题，您就可以腾出时间开展创造业务价值的工作，而不是解决重复出现的问题。在解决问题时，您可以反复进行审查，来确认架构的改进效果。

虽然在完成审查后，审查的价值显而易见，但您可能发现新团队在开始时可能会对审查抱有抵触情绪。可以通过与团队沟通审查的益处来解决下列异议：

- “我们实在太忙了！”（一般会在团队准备重大发布时这么说。）
  - 如果您正在为重大发布做准备，您会希望一切进展顺利。审查能够帮助您发现您可能错过的任何问题。
  - 我们建议您在产品生命周期早期执行审查，以及时发现风险，并制定与功能交付路线图一致的规避计划。
- “我们已经没有时间了，结果已成定局！”（一般会在他们面临无法改变的事件 [比如超级碗] 时这么说。）
  - 这些事件是无法改变的。您真的想在不了解架构风险的情况下将它投入使用吗？ 即使不能解决所有问题，您仍然可以编制一个潜在问题处理手册。
- “我们不希望其他人知道我们实施解决方案的秘诀！”
  - 如果您向团队指出 Well-Architected Framework 的问题，他们不会在这些问题中发现任何商业或技术专有信息。

在您与团队进行多次审核后，您可能会发现一些问题。例如，您可能发现一些团队在某个支柱或主题方面出现较多问题。建议您以全局眼光看待所有审查，找出能够帮助解决这些问题的任何机制、培训或首席工程师会谈方案。

结论

AWS Well-Architected Framework 涵盖六大支柱，提供了在云中设计和运行可靠、安全、高效、经济实惠且可持续的系统的架构最佳实践。该框架提供了一系列问题清单，帮助您审查现有或将要实现的架构。它还为每个支柱提供了一组 AWS 最佳实践。在架构中应用该框架将能帮助您打造稳定且高效的系统，从而让您能够专注于功能需求。

贡献者

以下个人和组织参与了本文档的编撰：

- Brian Carlson，亚马逊云科技 Well-Architected 操作主管
- Ben Potter，亚马逊云科技 Well-Architected 安全主管
- Seth Eliot，亚马逊云科技 Well-Architected 可靠性主管
- Eric Pullen，亚马逊云科技高级解决方案架构师
- Rodney Lester，亚马逊云科技首席解决方案架构师
- Jon Steele，亚马逊云科技高级技术客户经理
- Max Ramsay，亚马逊云科技首席安全解决方案架构师
- Callum Hughes，亚马逊云科技解决方案架构师
- Ben Mergen，亚马逊云科技高级成本首席解决方案架构师
- Chris Kozlowski，亚马逊云科技 Enterprise Support 高级专业技术客户经理
- Alex Livingstone，亚马逊云科技云运维首席专业解决方案架构师
- Paul Moran，亚马逊云科技 Enterprise Support 首席技术专家
- Peter Mullen，亚马逊云科技专业服务团队咨询顾问
- Chris Pates，亚马逊云科技 Enterprise Support 高级专业技术客户经理
- Arvind Raghunathan，亚马逊云科技 Enterprise Support 首席专家技术客户经理
- Sam Mokhtari，亚马逊云科技高级效率首席解决方案架构师

延伸阅读

AWS 架构中心

AWS 云合规性

AWS Well-Architected 合作伙伴计划

AWS Well-Architected Tool

AWS Well-Architected 主页

《卓越运营支柱白皮书》

《安全性支柱白皮书》

《可靠性支柱白皮书》

《性能效率支柱白皮书》

《成本优化支柱白皮书》

《可持续性支柱白皮书》

Amazon Builders' Library

文档修订

如需获取有关该白皮书更新的通知，请订阅 RSS 信息源。

变更	说明	日期
主要更新	最佳实践已根据以下各领域的新指导进行了更新：可靠性、安全性、卓越运营、可持续性和性能效率。可靠性支柱获得了对许多最佳实践的大规模改进和更新。有关安全性和卓越运营的指导已通过新的服务和生成式人工智能建议进行了更新和完善。可持续发展已根据 AWS 服务和新的最佳实践获得了多项更新。	2024 年 11 月 6 日
主要更新	对各支柱进行了大规模的最佳实践更新。安全和成本都采用了新的最佳实践。	2024 年 6 月 27 日
主要更新	主要支柱更新。	2023 年 10 月 3 日
主要更新	为最佳实践更新了规范性指南并增加了新的最佳实践。在“安全性”和“成本优化”支柱中增加了新问题。	2023 年 4 月 10 日
次要更新	在附录中增加了工作量定义和更新的最佳实践。	2022 年 10 月 20 日
主要更新	增加了可持续性支柱并更新了链接。	2021 年 12 月 2 日
主要更新	可持续性支柱已添加到框架中。	2021 年 11 月 20 日

次要更新	删除了非包容性用语。	2021 年 4 月 22 日
次要更新	修复了许多链接。	2021 年 3 月 10 日
次要更新	贯穿全文的次要编辑更改。	2020 年 7 月 15 日
主要更新	审核并重写大多数问题和答案。	2020 年 7 月 8 日
已更新白皮书	增加了 AWS Well-Architected Tool 以及指向 AWS Well-Architected Lab 和 AWS Well-Architected 合作伙伴的链接，进行了小的修复以支持框架的多语言版本。	2019 年 7 月 1 日
已更新白皮书	审核并重写大多数问题和答案，确保问题一次集中在一个主题上。这导致某些之前的问题被拆分成多个问题。向定义中添加了常见术语（工作负载、组件等）。更改了正文中问题的表达以包含描述性文本。	2018 年 11 月 1 日
已更新白皮书	进行了更新，以简化问题文本、实现答案的标准化和提高可读性。	2018 年 6 月 1 日
已更新白皮书	将卓越运营移至第一个支柱，并进行重新编写，以此引出其他支柱。更新了其他支柱以反映 AWS 的发展。	2017 年 11 月 1 日

已更新白皮书	更新了框架，添加了卓越运营支柱，修订并更新了其他支柱，以减少重复并整合从成千上万的客户审查实践中吸取的经验。	2016 年 11 月 1 日
次要更新	为附录更新了当前的 Amazon CloudWatch Logs 信息。	2015 年 11 月 1 日
初次发布	发布了 AWS Well-Architected Framework。	2015 年 10 月 1 日

> **Note**
>
> 要订阅 RSS 更新，您必须为当前使用的浏览器启用 RSS 插件。

框架版本：

- 2024 年 6 月 27 日
- 2023 年 10 月 3 日
- 2023 年 4 月 10 日
- 2022 年 3 月 31 日

附录：问题和最佳实践

此附录汇总了 AWS Well-Architected Framework 中的所有问题和最佳实践。

支柱

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

卓越运营

卓越运营（OE）是一项承诺，即正确地构建软件，同时持续提供卓越的客户体验。卓越运营支柱包含组织团队、设计工作负载、大规模运营工作负载和随时间推移改进工作负载的最佳实践。有关具体实施的说明性指导，请参阅《卓越运营支柱白皮书》。

最佳实践领域

- 组织
- 准备
- 运营
- 改进

组织

问题

- OPS 1. 您如何确定自己的优先事项？
- OPS 2. 如何构建组织结构来为业务成果提供支持？
- OPS 3. 组织文化如何为业务成果提供支持？

OPS 1. 您如何确定自己的优先事项？

每个人都应该了解自己在实现业务成功方面所发挥的作用。制定共同的目标，以便为资源设定优先事项。这可以让您的工作效益最大化。

最佳实践

- OPS01-BP01 评估外部客户需求
- OPS01-BP02 评估内部客户需求
- OPS01-BP03 评估治理要求
- OPS01-BP04 评估合规性要求
- OPS01-BP05 评估威胁形势
- OPS01-BP06 在管理益处与风险的同时评估各种权衡因素

OPS01-BP01 评估外部客户需求

让包括业务、开发和运营团队在内的关键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期望的业务成果所需的运营支持。

期望结果：

- 能够从客户成果出发进行逆向思维。
- 了解运营实践如何协助您获得业务成果和实现目标。
- 让所有相关方参与进来。
- 您已拥有用于捕获外部客户需求的机制。

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业务价值。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

审查外部客户的业务目标、需求和优先事项：让包括业务、开发和运营团队在内的关键利益相关方参与进来，讨论外部客户的目标、需求和优先事项。这可以确保您充分了解实现业务成果和客户成果所需的运营支持。

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

资源

相关最佳实践：

- OPS11-BP03 实施反馈环路

OPS01-BP02 评估内部客户需求

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

期望结果：

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

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业务价值。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

资源

相关最佳实践：

- OPS11-BP03 实施反馈环路

OPS01-BP03 评估治理要求

治理是公司用来实现业务目标的一系列政策、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工作负载运行方式。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行中。
- 您可以提供证据来证明您遵循了治理要求。
- 定期审查和更新治理要求。

常见反模式：

- 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

与利益相关方和治理组织合作，确定治理要求。将治理要求包括在工作负载中。可以提供证据来证明您遵循了治理要求。

客户示例

在 AnyCompany Retail，云运维团队与整个组织内的利益相关方合作制定治理要求。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权限，他们需要使用 AWS Systems Manager Session Manager。随着新服务推出，云运维团队定期更新治理要求。

实施步骤

1. 为工作负载确定利益相关方，包括任何集中式团队。
2. 与利益相关方合作确定治理要求。
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工作负载中。
   a. 使用诸如 AWS Config 之类的服务创建治理即代码，并验证是否遵循了治理要求。
   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。实施缺失的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 – 合规性就像治理，但来自组织外部。

相关文档：

- AWS Management and Governance Cloud Environment Guide
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- Governance in the AWS Cloud: The Right Balance Between Agility and Safety
- 什么是治理、风险与合规性（GRC）？

相关视频：

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks
- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1)
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)

相关示例：

- AWS Config Conformance Pack Samples

相关服务：

- AWS Config
- AWS Organizations - Service Control Policies

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事项的重要驱动因素。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性框架，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 审核即将开始，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核性，以便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。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. 与安全和治理团队合作，确定工作负载必须遵守哪些行业、监管或内部合规性框架。将合规性框架纳入工作负载。
   a. 使用 AWS Compute Optimizer 和 AWS Security Hub 之类的服务，验证 AWS 资源的持续合规性。
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，您可能需要生成审核或合规性报告。与组织合作，尽可能使此过程实现自动化。
   a. 使用诸如 AWS Audit Manager 之类的服务验证合规性并生成审计报告。
   b. 您可以通过 AWS Artifact 下载 AWS 安全与合规性文档。

实施计划的工作量级别：中。实施合规性框架并非易事。生成审核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 识别和验证控制目标 – 安全控制目标是整体合规性的重要组成部分。
- SEC01-BP06 自动测试和验证管道中的安全控制 – 作为管道的一部分，验证安全控制。还可以生成新变更的合规性文档。
- SEC07-BP02 定义数据保护控制 – 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 – 取证能力有时可用于审核合规性。

相关文档：

- AWS 合规中心
- AWS 合规性资源
- AWS 风险与合规性白皮书
- AWS 责任共担模式
- 按合规性计划提供的范围内 AWS 服务

相关视频：

- AWS re:Invent 2020: Achieve compliance as code using AWS Compute Optimizer
- AWS re:Invent 2021 - Cloud compliance, assurance, and auditing
- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (COP202)

相关示例：

- AWS 上的 PCI DSS 和 AWS 基础安全最佳实践

相关服务：

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

OPS01-BP05 评估威胁形势

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

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

AWS 客户可以使用针对关键任务型工作负载的指导式架构完善的审核，以根据 AWS 最佳实践来衡量其架构。Enterprise Support 客户可以使用运营审核，该审核旨在帮助他们找出云中的运营方法所存在的漏洞。

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

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，帮助确定优先事项。Business Support 和 Enterprise Support 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，可进一步帮助他们确定优先事项。

期望结果：

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

常见反模式：

- 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，帮助确定要解决的威胁、需解决的威胁的优先级以及执行操作所需的资源。

在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

相关最佳实践：

- SEC01-BP07 使用威胁模型识别威胁并确定缓解措施的优先级

相关文档：

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

相关视频：

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

OPS01-BP06 在管理益处与风险的同时评估各种权衡因素

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

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

AWS 有助于您就 AWS 及其服务对团队进行培训，让他们深入了解自己的选择会如何影响工作负载。使用由 支持 提供的资源（AWS 知识中心、AWS 讨论论坛和 支持 中心）和 AWS 文档来培训团队。如有其他问题，请联系 支持。

AWS 还在 Amazon Builders' Library 中分享了运营最佳实践和模式。您可以通过 AWS Blog 和 The Official AWS Podcast，获得各种其他有用信息。

期望结果：您有明确定义的决策治理框架，可以促进云交付组织内各个级别的重要决策。此框架包括风险登记单、有权作出决策的已定义角色，以及针对各级可制定之决策的已定义模型等特色内容。此框架预先定义了冲突解决方式、需要提供的数据以及选项优先级的确定方式，因此，一旦作出决策，您便能立即执行。决策制定框架包括一种标准化方法，用于审核和认真考虑每项决策的益处与风险以了解权衡。这可能包括外部因素，例如遵守法规合规性要求。

常见反模式：

- 您的投资者要求您证明符合支付卡行业数据安全标准（PCI DSS）。您没有在满足他们的要求和继续进行当前的开发工作之间进行权衡，而是在没有证明合规性的情况下继续进行开发工作。投资者出于对平台安全性和投资的担忧停止了对公司的支持。
- 您决定纳入一个库，这是您的一位开发人员在互联网上找到的库。您尚未评估从未知来源采用此库的风险，也不知道其中是否包含漏洞或恶意代码。
- 对于迁移，最初的业务理由是实现 60% 的应用程序工作负载的现代化。但由于遇到技术难题，您决定仅实现 20% 的应用程序工作负载的现代化，这导致了长期计划收益的减少；基础设施团队的操作负担加重，需要手动支持遗留系统；并且更加依赖于培养基础设施团队的新技能组合，而团队并未对此变更做好规划。

建立此最佳实践的好处：充分调整和支持董事会级别的业务优先事项，了解取得成功的风险，作出明智的决策，并在风险阻碍成功时采取适当行动。了解决策的影响和后果有助于您确定选项的优先级，更快

地让各个领导达成共识，从而改进业务成果。确定选择可以带来的益处并了解组织面临的风险，有助于您依据数据而不是轶事来制定决策。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

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

实施步骤

1. 在整体云治理框架内正式确定效益衡量实践。
    a. 平衡决策制定的集权与某些决策的分权。
    b. 要明白一点，将繁冗的决策过程强加于每个决策之上，只会拖慢您的决策速度。
    c. 将外部因素纳入您的决策制定过程（例如合规性要求）。
2. 为各级决策建立一致认可的决策制定框架，包括谁负责疏解受利益冲突影响的决策。
    a. 共同制定不可更改的单向门决策。
    b. 允许较低级别的组织领导者制定双向门决策。
3. 了解和管理益处与风险。在决策的益处与涉及的风险之间取得平衡。
    a. 确定效益：根据业务目标、需求和优先事项来确定效益。例如业务案例影响、上市时间、安全性、可靠性、性能和成本等。
    b. 确定风险：根据业务目标、需求和优先事项来确定风险。例如上市时间、安全性、可靠性、性能和成本等。
    c. 对照风险评测益处并作出明智决策：根据包括业务、开发和运营团队在内的关键利益相关方的目标、需求和优先事项，确定益处与风险的影响。对照发生风险的可能性及其影响产生的成本，评估效益的价值。例如，强调上市速度而不是可靠性可能会带来竞争优势。但是如果出现可靠性问题，就可能会导致正常运行时间缩短。
4. 采用程序化方式执行关键决策，以便自动遵守合规性要求。
5. 利用已知的行业框架和能力，如价值流分析和精益生产，衡量当前状态的绩效和业务指标，并定义逐步改进这些指标的迭代过程。

实施计划的工作量级别：中高

资源

相关最佳实践：

- OPS01-BP05 评估威胁形势

相关文档：

- 亚马逊第 1 天文化的要素 | 做出高质量、高速的决策
- 云治理
- 管理和治理云环境
- Governance in the Cloud and in the Digital Age: Parts One & Two

相关视频：

- Podcast | Jeff Bezos | On how to make decisions

相关示例：

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

OPS 2. 如何构建组织结构来为业务成果提供支持？

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。团队应该了解自己在其他团队获得成功的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，并制定共同的目标。了解责任分配、所有权归属、决策制定方式以及决策者，将有助于集中精力并最大限度地发挥团队的优势。

最佳实践

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

OPS02-BP01 确定资源所有者

工作负载的资源必须具有已确定的所有者，以便实现变更控制、故障排除和其他功能。为工作负载、账户、基础设施、平台和应用程序分配所有者。使用集中登记册或附加到资源的元数据等工具记录所有权。组件的商业价值指明了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。
- 资源缺少用于标识其所属团队的标签。
- 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，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应用场景的意义。所有权表示谁监督资源的变更、谁在排除故障时对资源提供支持或谁负责财务。指定并记录资源所有者，包括姓名、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。使用组收件箱配置备用账户联系人。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，使用 Wiki 页面来确定所有权和联系信息。

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，或在排除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 管理账户。可以集中管理账户的备用联系人。
   a. 使用公司拥有的电子邮件地址和电话号码作为联系信息，这样一来，即使其所属员工离开了公司，也不会影响您的正常访问。例如，为账单、运营和安全性创建单独的电子邮件分发列表，并在各个活跃的 AWS 账户 中将它们配置为账单、安全性和运营联系人。有多人会收到 AWS 通知，所以即使有人在度假、职责变更或离开公司，也有其他人能够作出回复。
   b. 如果账户不是由 AWS Organizations 管理，备用账户联系人可以根据需要帮助 AWS 联系相应人员。将账户的备用联系人配置为指向群组而不是指向个人。
3. 使用标签来标识 AWS 资源的所有者。可以用单独的标签指定所有者及其联系信息。
   a. 可以使用 AWS Config 规则强制资源具有所需的所有权标签。
   b. 有关如何为组织制定标记策略的深入指导，请参阅《AWS 标记最佳实践白皮书》。
4. 使用 Amazon Q 企业版，这是一款对话助手，使用生成式人工智能来提高员工的工作效率、回答问题并根据企业系统中的信息完成任务。
   a. 将 Amazon Q 企业版连接到贵公司的数据来源。Amazon Q 企业版为 40 多个支持的数据来源提供预先构建的连接器，包括 Amazon Simple Storage Service（Amazon S3）、Microsoft SharePoint、Salesforce 和 Atlassian Confluence。有关更多信息，请参阅 Amazon Q 企业版连接器。
5. 对于其他资源、平台和基础设施，创建用于标识所有权的文档。所有团队成员应该都可以访问此文档。

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

资源

相关最佳实践：

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

相关文档：

- AWS Account Management - Updating contact information

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

相关讲习会：

- AWS 讲习会 – Tagging

相关示例：

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

相关服务：

- AWS Config 规则 - required-tags
- AWS Organizations

OPS02-BP02 确定流程和程序负责人

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

期望结果：针对运营任务，组织制定了一套明确定义并良好维护的流程和程序。流程和程序集中存储在一个位置，可供团队成员使用。按照明确指派的责任归属，经常更新流程和程序。尽可能将脚本、模板和自动化文档作为代码实施。

常见反模式：

- 流程未记录在案。脚本呈现碎片化，可能分布在许多孤立的操作员工作站上。
- 脚本的使用方法只有少数人了解，或作为团队知识非正式地交流。

- 旧的流程需要更新，但不明确应由谁负责更新，原作者已离开了组织。
- 无法发现流程和脚本，因此在需要时（例如，在响应意外事件时）无法使用。

建立此最佳实践的好处：

- 流程和程序可改进运行工作负载的工作。
- 新的团队成员可以更快地投入工作中。
- 缩短了缓解意外事件的用时。
- 不同的团队成员（以及不同的团队）可以一致地使用相同的流程和程序。
- 团队可以使用可重复的流程来扩展其流程。
- 在团队之间移交工作负载责任时，标准化的流程和程序有助于减轻移交造成的影响。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

客户示例

AnyCompany Retail 对“负责人”的定义是：负责某个应用程序或应用程序组（共享通用架构实践和技术）的流程的团队或个人。最初，这些流程和程序以分步指南的形式记录在文档管理系统中，

可在托管应用程序的 AWS 账户 上以及账户中的特定资源组上，使用标签来发现。他们利用 AWS Organizations 来管理其 AWS 账户。随着时间的推移，这些流程会转换为代码，并使用基础设施即代码（例如 CloudFormation 或 AWS Cloud Development Kit (AWS CDK) 模板）定义资源。运营流程成为 AWS Systems Manager 中的自动化文档或 AWS Lambda 函数，这些流程可以作为计划任务启动，用于响应 AWS CloudWatch 警报等事件或 AWS EventBridge 事件，也可以通过 IT 服务管理（ITSM）平台内的请求启动。所有流程都有标签，用于标识负责人。用于自动化和流程的文档，保存在由该流程的代码存储库生成的 Wiki 页面中。

实施步骤

1. 记录现有的流程和程序。
    a. 查看并保持最新状态。
    b. 确定每个流程或程序的负责人。
    c. 对流程和程序实施版本控制。
    d. 只要可能，对具有相同架构设计的工作负载和环境，共享流程和程序。
2. 建立反馈和改进机制。
    a. 定义有关流程审查频率的政策。
    b. 定义审核者和审批者流程。
    c. 实施问题队列或票证队列，以便提供和跟踪反馈。
    d. 在可能时，流程和程序应由变更审批委员会（CAB）预先审批并进行风险分类。
3. 确认需要运行这些流程和程序的人员能够访问和搜索到流程和程序。
    a. 使用标签来指示可以在哪里访问工作负载的流程和程序。
    b. 使用有意义的错误和事件消息，指明用于解决问题的正确流程或程序。
    c. 使用 Wiki 和文档管理，确保可在整个组织内一致地搜索流程和程序。
4. 使用 Amazon Q 企业版，这是一款对话助手，使用生成式人工智能来提高员工的工作效率、回答问题并根据企业系统中的信息完成任务。
    a. 将 Amazon Q 企业版连接到贵公司的数据来源。Amazon Q 企业版为 40 多个支持的数据来源提供预先构建的连接器，包括 Amazon S3、Microsoft SharePoint、Salesforce 和 Atlassian Confluence。有关更多信息，请参阅 Amazon Q 连接器。
5. 在适当时实现自动化。
    a. 当服务和技术提供 API 时，应开发自动化功能。
    b. 针对流程充分开展培训。开发用户案例和要求，用于实现这些流程的自动化。
    c. 衡量流程和程序的成功使用情况，并提出问题或票证来支持迭代改进。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

- AWS 白皮书 – AWS 上的 DevOps 简介
- AWS 白皮书 – Best Practices for Tagging AWS Resources
- AWS 白皮书 – Organizing Your AWS Environment Using Multiple Accounts
- AWS Cloud Operations and Migrations Blog - Using Amazon Q Business to streamline your operations
- AWS Cloud Operations & Migrations Blog - Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Cloud Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

相关讲习会：

- AWS Well-Architected Operational Excellence 讲习会
- AWS 讲习会 – Tagging

相关视频：

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

相关服务：

- AWS Systems Manager – 自动化
- AWS 服务管理连接器

OPS02-BP03 确定对运营活动绩效负责的责任人

了解谁负责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，可告知谁来开展活动、验证结果并向活动负责人提供反馈。

期望结果：

组织明确定义了在对定义的工作负载执行特定活动以及响应由工作负载生成的事件时，需要承担的相关责任。组织记录了流程的所属责任和实施方法，并让这些信息可供搜索。在发生组织变更时审查和更新责任，并且团队跟踪和衡量缺陷和低效率识别活动的绩效。实施反馈机制来跟踪缺陷和改进，并支持迭代改进。

常见反模式：

- 未记录责任。
- 脚本呈现碎片化，分布在许多孤立的操作员工作站上。脚本的使用方法只有少数人了解，或将其非正式地称为团队知识。
- 旧的流程需要更新，但没有人知道该流程的负责人是谁，原作者已不在组织中。
- 无法发现流程和脚本，并且在需要时（例如，在响应意外事件时）无法使用。

建立此最佳实践的好处：

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

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

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

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

客户示例

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

实施步骤

1. 记录现有的流程和程序。
    a. 审核并确认它们是否为最新。
    b. 确认每个流程或程序都有负责人。
    c. 对程序实施版本控制。
    d. 只要可能，对具有相同架构设计的工作负载和环境，共享流程和程序。
2. 建立反馈和改进机制。
    a. 定义有关流程审查频率的政策。

b. 定义审核者和审批者流程。

c. 实施问题队列或票证队列，以便提供和跟踪反馈。

d. 在可能时，流程和程序将由变更审批委员会（CAB）预先审批并进行风险分类。

3. 让需要运行这些流程和程序的人员能够访问和搜索到流程和程序。

a. 使用标签来指示可以在哪里访问工作负载的流程和程序。

b. 使用有意义的错误和事件消息，指明用于解决问题的正确流程或程序。

c. 使用 Wiki 或文档管理，确保可在整个组织内一致地搜索流程和程序。

4. 在适当时，实现自动化。

a. 在服务和技术提供 API 时，开发自动化功能。

b. 验证是否能充分理解流程，并开发用户案例和要求来实现这些流程的自动化。

c. 衡量流程和程序的成功使用情况，并跟踪问题来支持迭代改进。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

- AWS 白皮书 | AWS 上的 DevOps 简介
- AWS 白皮书 | Best Practices for Tagging AWS Resources
- AWS 白皮书 | Organizing Your AWS Environment Using Multiple Accounts
- AWS Cloud Operations & Migrations Blog - Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS 讲习会 – Tagging
- AWS Service Management Connector

相关视频：

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

OPS02-BP04 制定用于管理责任和所有权的机制

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

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

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

常见反模式：

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

建立此最佳实践的好处：

- 在了解谁负责或拥有所有权后，可以与合适的团队或团队成员联系，提出请求或转换任务。
- 已确定有权分配责任或所有权的人员，可以降低不作为和需求无法得到满足的风险。
- 在明确定义责任范围后，团队成员就能获得自主权和所有权。

- 责任可帮助明确所作的决定、采取的行动以及需要将哪些活动交给适当的所有者。
- 轻松确定已放弃的责任，因为您清楚地了解团队责任范围边界，这有助于上报来进行澄清。
- 团队可以避免混乱和紧张的情况，更充分地管理其工作负载和资源。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

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

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

客户示例

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

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

实施步骤

1. 从现有文档开始。一些典型的源文档可能包括：
    a. 责任或负责、问责、咨询和知情（RACI）矩阵
    b. 团队定义或 Wiki 页面
    c. 服务定义和产品/服务
    d. 职责或工作描述

2. 审核记录的责任，并主持围绕责任开展讨论：
    a. 与团队一起进行审核，找出记录的责任与团队通常履行的责任之间不一致之处。
    b. 讨论内部客户提供的潜在服务，确定团队之间的期望差距。
3. 分析并解决差异。
4. 确定改进机会。
    a. 确定经常提出的资源密集型请求，这些请求通常是可改进的有力候选方案。
    b. 寻找最佳实践、了解模式和遵守规范性指南，并简化和标准化改进方案。
    c. 记录改进机会并一直跟踪它们，直至完成。
5. 如果一个团队尚未承担管理和跟踪责任分配的责任，请指定一个团队成员来承担这一责任。
6. 为团队定义一个流程来请求澄清责任。
    a. 审查该流程，确认流程明确并且简单易用。
    b. 确保有人负责和跟踪上报情况，直至得出结论。
    c. 建立运营指标来衡量有效性。
    d. 创建反馈机制，验证团队能否突出改进机会。
    e. 实施定期审查机制。
7. 在可搜索和访问的位置保存文档。
    a. Wiki 或文档门户是常见选择。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

- AWS 白皮书 – AWS 上的 DevOps 简介
- AWS 白皮书 – AWS Cloud Adoption Framework: Operations Perspective
- AWS Well-Architected Framework 卓越运营 – 工作负载级别运营模式拓扑
- AWS Prescriptive Guidance - Building your Cloud Operating Model
- AWS Prescriptive Guidance - Create a RACI or RASCI matrix for a cloud operating model
- AWS Cloud Operations & Migrations Blog - Delivering Business Value with Cloud Platform Teams
- AWS Cloud Operations & Migrations Blog - Why a Cloud Operating Model?
- AWS DevOps Blog - How organizations are modernizing for cloud operations

相关视频：

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

OPS02-BP05 制定用于请求添加、更改和例外的机制

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。对益处与风险进行评估之后，作出明智的决定，批准可行和确认合适的请求。

期望结果：

- 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，权衡益处与风险。

常见反模式：

- 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流程。
- 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、程序和资源会随着要求变化而演进。
- 进行变更时，所有者可以作出明智的决策。
- 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

实施指导

为实施这种最佳实践，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简单。记录变更管理流程。

客户示例

AnyCompany Retail 使用责任分配（RACI）矩阵来确定谁负责流程、程序和资源的变更。他们制定了书面变更管理流程，这些流程简单且易于遵循。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、程序和资源及各自的所有者。将这些信息记录在知识管理系统中。
   a. 如果还没有实施 OPS02-BP01 确定资源所有者、OPS02-BP02 确定流程和程序负责人 或 OPS02-BP03 确定对运营活动绩效负责的责任人，请先从这些开始。
2. 与组织中的利益相关方合作，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. 可以将 AWS Systems Manager Change Manager 用作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。制定变更管理流程需要与整个组织的多个利益相关方达成一致。

资源

相关最佳实践：

- OPS02-BP01 确定资源所有者 – 在构建变更管理流程之前，需要确定资源的所有者。
- OPS02-BP02 确定流程和程序负责人 – 在构建变更管理流程之前，需要确定流程的所有者。
- OPS02-BP03 确定对运营活动绩效负责的责任人 – 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 Prescriptive Guidance - Foundation palybook for AWS large migrations: Creating RACI matrices
- 《Change Management in the Cloud 白皮书》

相关服务：

- AWS Systems Manager Change Manager

OPS02-BP06 预先定义或协商团队间的职责

团队之间具有明确或协商好的协议，规定了团队之间的合作和相互支持方式（例如，响应时间、服务水平目标或服务水平协议）。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顺序，并帮助他们作出适当的响应。

当责任和所有权不确定或未知时，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。请求卡滞在积压工作中。

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了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水平协议。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. 与整个组织的利益相关方合作，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共享了流程或程序，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。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相关方达成一致。

资源

相关最佳实践：

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

相关文档：

- AWS Executive Insights – 与双披萨团队一起推动创新
- AWS 上的 DevOps 简介 – 双披萨团队

OPS 3. 组织文化如何为业务成果提供支持？

为团队成员提供支持，以便他们可以更有效地采取行动并为您的业务成果提供支持。

最佳实践

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

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

OPS03-BP01 提供高管支持

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

期望结果：致力于采用、转型和优化云运营的组织，为实现期望结果建立了明确的领导和责任界限。组织了解实现新成果所需的每项能力，并授权职能团队针对相关能力进行培养。领导层要积极确定这一方向、分配所有权、承担责任并界定工作。因此，整个组织中的每个人都能动员起来，受到鼓舞，并积极努力实现预期目标。

常见反模式：

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

建立此最佳实践的好处：当高管清楚地传达并分享愿景、方向和目标时，团队成员就会知道对他们的期望。当领导者积极参与时，个人和团队就会开始集中精力朝着同一个方向努力，完成既定目标。因此，组织最大限度地提高了获得成功的能力。评估成功时，可以更好地发现成功之路上的障碍，以便通过执行发起人的干预来克服这些障碍。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

- 在云之旅的每个阶段（迁移、采用或优化），成功都需要最高领导层的积极参与，并指定一名执行发起人。执行发起人能够让团队的思维方式、技能组合和工作方法与既定策略保持一致。
  - 解释原因：阐明并解释愿景和策略背后的原因。
  - 设定期望：为组织定义和发布目标，包括如何衡量进展和成功。
  - 跟踪目标的实现情况：定期衡量目标的逐步实现情况（而不仅仅是任务的完成情况）。分享结果，以便在结果面临风险时可以采取适当的行动。

- 提供实现目标所需的资源：让人员和团队齐心协力，制定正确的解决方案，实现既定结果。这可以减少乃至消除组织内部的摩擦。
- 为团队提供支持：与团队保持互动，以便了解他们的表现以及是否有外部因素影响他们。确定阻碍团队进度的障碍。代表团队采取行动，帮助消除障碍，除去不必要的负担。团队受外部因素影响时，需重新评估目标并适当地调整执行性目标。
- 推动最佳实践的采用：认可可量化收益的最佳实践以及创建者和采用者。鼓励进一步采用，实现更大收益。
- 鼓励团队的发展：营造持续改进的文化，主动从进步和失败中吸取教训。鼓励个人和组织的成长与发展。利用数据和轶事来发展愿景和策略。

客户示例

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

实施步骤

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

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

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

相关视频：

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

相关示例：

- Prosci: Primary Sponsor's Role & Importance

OPS03-BP02 赋能团队成员在结果有风险时采取行动

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

例如，亚马逊使用领导力原则作为准则，推动员工实现在各种情况下前进、解决问题、处理冲突和采取行动等期望行为。

期望结果：在领导力的影响下产生了一种新文化，这种文化支持个人和团队作出关键决策，即使在组织的较低层级也是如此（只要决策是用可审计的权限和安全机制定义的）。失败并不可怕，团队会不断学

习，改进决策和响应措施，从而应对今后出现的类似情况。如果某个人的行动带来了改进，能让其他团队受益，这些团队就会主动分享从这些行动中获得的知识。领导层衡量运营改进情况，并激励个人和组织采用此类模式。

常见反模式：

- 组织内没有明确的指导或机制来说明在发现风险时该怎么做。例如，当员工发现网络钓鱼攻击时，他们没有向安全团队报告，导致组织中的大部分人遭受攻击。这会造成数据泄露。
- 客户抱怨服务不可用，主要原因是部署失败。SRE 团队负责部署工具，而他们的长期路线图中包括自动回滚部署。在最近一次的应用程序推广中，一位工程师设计了一种解决方案，可以自动将应用程序回滚到以前的版本。虽然他们的解决方案可以成为 SRE 团队采用的模式，但其他团队并不采用，因为没有流程能跟踪此类改进。组织继续受到部署失败的困扰，这影响了客户，造成了更多负面情绪。
- 为了保持合规性，信息安全团队会监督一个长期建立的流程，代表连接到 Amazon EC2 Linux 实例的操作员定期轮换共享的 SSH 密钥。信息安全团队需要花几天的时间才能完成密钥的轮换，并且您将无法连接到这些实例。信息安全团队内部和外部的任何人都不建议使用 AWS 上的其他选项来实现相同的结果。

建立此最佳实践的好处：通过下放决策权并授权团队决定关键决策，您可以更快地解决问题，并提高成功率。此外，团队开始具有主人翁意识，并意识到失败是可以接受的。实验成为一种文化主流。经理和主管不会觉得他们在工作的各个方面都受到微观管理。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

1. 培养一种会预见失败的文化。
2. 明确规定组织内各职能领域的所有权和责任。
3. 向每个人传达所有权和问责制，让大家都知道谁能帮助他们促进分散决策。
4. 定义单向门决策和双向门决策，让个人了解何时确实需要上报给更高级别的领导。
5. 树立组织意识，让所有员工都有能力在结果面临风险时，从各个层级采取行动。为团队成员提供治理文件、权限级别、工具以及机会，让团队成员练习有效应对所需的技能。
6. 为团队成员提供机会，练习应对各种决策所需的技能。一旦确定了决策级别，就应开展 GameDay 活动，确保所有参与人员都能理解并演示流程。
   a. 提供替代的安全环境，以便在其中对流程和程序进行测试和培训。
   b. 承认并让团队成员认识到，当结果达到预先定义的风险水平时，他们有权采取行动。

c. 通过为团队成员所支持的工作负载和组件分配权限和访问权限，定义团队成员的行动权限。

7. 让团队能够分享他们的经验教训（运营方面的成功和失败经验教训）。
8. 授权团队挑战现状，并建立一些机制，让团队跟踪和衡量改进情况及其对组织的影响。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

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

相关视频：

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

相关示例：

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

OPS03-BP03 鼓励上报

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

期望结果：整个组织中的个人都乐于将问题上报给直属和更高级别的领导层。领导层刻意并有意识地建立期望，让他们的团队可以毫无顾虑地上报任何问题。在组织内部的每个层级，制定上报问题的机制。当员工将问题上报给经理时，他们共同决定问题的影响程度以及是否应该上报。要启动上报程序，员工需要提交一份解决问题的建议工作计划。如果直属管理层没有及时采取行动，而员工强烈认为组织面临的风险需要上报，则组织鼓励员工将问题上报至最高领导层。

常见反模式：

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

建立此最佳实践的好处：

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

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

组织各个层级中的自由上报意愿和能力是一种组织和文化基础，应通过强调培训、领导层沟通、期望设定，以及在整个组织的各个层面部署机制，有意识地加以培养。

实施步骤

1. 制定组织的政策、标准和期望。
    a. 确保政策、期望和标准得到广泛采纳和理解。
2. 鼓励、培训工作人员，并赋予他们权力，以便在不符合标准时他们会尽早、频繁地上报。
3. 从组织的角度确认，及早和频繁上报是最佳实践。接受上报的内容最终可能证明并无依据，但最好要抓住机会预防意外事件的发生，而不要因为没有上报而错失机会。

   a. 建立上报机制（比如 Andon Cord 系统）。
   b. 制定成文的程序，规定何时以及如何上报。
   c. 确定一系列有各级权力来采取或批准行动的人员，以及每个利益相关方的联系信息。
4. 当上报发生时，应有始有终，直到团队成员认为领导层推动的行动可以充分降低风险，并对结果满意。
   a. 上报内容应包括：
      i. 情况描述和风险性质
      ii. 情况的严重性
      iii. 受影响的人或事
      iv. 影响有多大
      v. 发生影响时的紧迫性
      vi. 建议的补救措施和减轻影响的计划
   b. 保护上报的员工。制定政策来保护团队成员，如果他们上报关于决策者或利益相关方未做出响应的问题，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适当响应。
5. 鼓励在组织的所有事项中建立持续改进的反馈环路文化。反馈环路起到向责任人进行小规模上报的作用，即使不需要上报，也能发现改进机会。持续改进的文化促使每个人更加积极主动。
6. 领导层应定期重新强调政策、标准、机制，以及公开上报和持续反馈环路而不受到报复的期望。

实施计划的工作量级别：中

资源

相关最佳实践：

- OPS02-BP05 制定用于请求添加、更改和例外的机制

相关文档：

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

相关视频：

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

相关示例：

- Working with escalation plans in Incident Manager

OPS03-BP04 沟通及时、清晰、可行

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

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

常见反模式：

- 组织有一个五年计划，要将所有工作负载迁移到 AWS。云业务案例包括对 25% 的工作负载进行现代化改造，以便利用无服务器技术。首席信息官将这一策略传达给直接下属，并希望每位领导者将这一策略传达给经理、总监和个人贡献者，而无需进行任何面对面的沟通。首席信息官退居幕后，期望组织能够执行新策略。
- 领导层不提供或不使用反馈机制，期望差距变得越来越大，从而导致项目停滞不前。
- 有人要求您对安全组进行更改，但却没有告诉您详细信息，例如需要进行哪些更改，更改会对所有工作负载产生什么影响，以及何时进行更改等。经理转发了一封来自信息安全副总裁的电子邮件，并添加了“实现此目标”的信息。

- 迁移策略发生了变化，计划的现代化改造数量从 25% 减少到 10%。这会对运营组织的下游产生影响。下游组织未被告知这一策略变化，因此没有足够的技术能力协助将更多的工作负载直接迁移到 AWS。

建立此最佳实践的好处：

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

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

客户示例

AnyCompany Retail 注册了 AWS Enterprise Support，并依赖其他第三方提供商进行云运营。该公司将聊天和 ChatOps 工具作为运营活动的主要沟通媒介。警报和其他信息会填入特定渠道。当有人必须采取行动时，他们会清楚地说明期望结果，而且在很多情况下，他们会收到一份运行手册或行动手册以供使用。他们借助变更日历来安排生产系统的重大更改。

实施步骤

1. 在组织内建立一个核心团队，负责为组织内多个层级的更改制定和启动沟通计划。
2. 建立单线程所有权，以便实现监督。赋予各个团队独立创新的能力，并平衡使用一致的机制，从而实现适当程度的检查和方向性愿景。
3. 与整个组织的利益相关方合作，就沟通标准、实践和计划达成一致。
4. 确认核心沟通团队是否与组织和项目领导层合作，代表领导者向相关人员传达信息。

5. 建立策略沟通机制，通过公告、共享日历、全体员工会议、面对面或一对一的方式管理更改，让团队成员对自己应采取的行动有正确的预期。
6. 提供必要的背景、详细信息和时间（如有可能），以便确定是否有必要采取行动。需要采取行动时，提供所需的行动及其影响。
7. 实施促进战术沟通的工具，例如内部聊天、电子邮件和知识管理。
8. 实施各种机制，以便衡量和确认所有沟通活动是否都取得了期望结果。
9. 建立反馈环路来衡量所有沟通的效果，尤其是当沟通涉及到整个组织对更改的抵触时。
10. 对于所有 AWS 账户，请为账单、安全性和运营创建备用联系人。理想情况下，每个联系人都应是电子邮件分发的收件人，而不是特定的个人联系人。
11. 制定上报和逆向上报沟通计划，与内部团队和外部团队（包括 AWS Support 和其他第三方提供商）进行沟通。
12. 在每个转型计划的整个生命周期内，始终如一地启动和执行沟通策略。
13. 优先考虑可重复执行的行动，尽可能安全地实现大规模自动化。
14. 当需要在自动化操作的场景中进行沟通时，沟通目的应该是通知团队、进行审核或作为变更管理流程的一部分。
15. 分析来自警报系统的通信，判断误报或不断生成的警报。删除或更改这些警报，以便在需要人工干预时启动。如果启动了警报，则提供运行手册或行动手册。
    a. 您可以使用 AWS Systems Manager 文档为警报制定行动手册和运行手册。
16. 制定合理的机制，以清晰、可操作的方式提供风险或计划内事件的通知，而且要引起足够的注意，以便适当响应。使用电子邮件列表或聊天频道在计划内事件之前发送通知。
    a. AWS Chatbot 可用于发送警报并响应组织消息平台中的事件。
17. 提供可访问的信息源，其中包含计划内事件。通知来自同一系统的计划内事件。
    a. 发生更改时，可使用 AWS Systems Manager Change Calendar 来创建变更窗口。因而在团队成员可以安全地进行变更时，向他们发送通知。
18. 监控漏洞通知和补丁程序信息，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 漏洞的通知。
19. 寻求不同的意见和观点：鼓励所有人做出贡献。为代表性不足的群体提供沟通机会。在会议中轮换职责和责任。
    a. 扩大职责和责任：让团队成员有机会尝试他们可能不会担任的角色。他们可以从职责以及与其他团队成员的互动中获得经验和见解，而之前可能并没有机会与这些成员互动。他们还可以将自己的经验和见解赋予新角色，以及就此与新团队成员沟通交流。随着见解不断增多，需要确定新出

现的业务机会或新的改进机会。在团队成员之间轮流执行其他人通常执行的日常任务，了解执行这些任务的需求和影响。

b. 提供安全舒适的环境：制定政策和控制措施，保护组织内团队成员的身心安全。团队成员应该能够彼此敞开心扉，而不是处在会受到报复的担惊受怕之中。当团队成员处于安全舒适的环境中时，才能有更高的参与热情、更高的工作成效。组织越多元化，就越能更好地理解所支持的人，包括客户。当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取时，他们会更愿意分享有价值的洞察（例如营销机会、可访问性需求、尚待开发的细分市场以及环境中未发现的风险）。

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

资源

相关最佳实践：

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

相关文档：

- AWS Blog 文章 | Accountability and empowerment are key to high-performing agile organizations
- AWS Executive Insights | 学会扩大创新规模，而不是增加复杂性 | 单线程领导者
- AWS 安全公告
- OpenCVE
- 支持 App in Slack to Manage Support Cases
- Manage AWS resources in your Slack channels with Amazon Q Developer in chat applications

相关服务：

- 聊天应用程序中的 Amazon Q 开发者版
- AWS Systems Manager Change Calendar
- 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。它可以加快学习速度，让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，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：

- 组织鼓励试验来促进创新。
- 将试验当作学习的机会。

常见反模式：

- 想要运行 A/B 测试，但没有运行试验的机制。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。
- 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，可以更快地对用户的反馈作出反应。
- 组织培养了一种学习文化。

在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

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。让团队成员能够在沙盒环境中进行试验。

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，用真实的用户反馈进行验证。

实施步骤

1. 与整个组织的领导层合作来支持试验。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 可创建功能标记。
    b. 您可以使用 AWS Lambda 版本部署函数的新版本来进行测试版测试。

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。可能还需要修改应用程序代码来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 – 从意外事件中吸取教训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组成部分。

相关文档：

- An Inside Look at the Amazon Culture: Experimentation, Failure, and Customer Obsession
- Best practices for creating and managing sandbox accounts in AWS
- Create a Culture of Experimentation Enabled by the Cloud
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros
- Experiment More, Fail Less
- Organizing Your AWS Environment Using Multiple Accounts - Sandbox OU
- Using AWS AppConfig Feature Flags

相关视频：

- AWS On Air ft. Amazon CloudWatch Evidently | AWS Events
- AWS On Air San Fran Summit 2022 ft. AWS AppConfig Feature Flags integration with Jira
- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R)
- Programmatically Create an AWS 账户 with AWS Control Tower

- 为 AWS Organizations 设置使用最佳实践的多账户 AWS 环境

相关示例：

- AWS 创新沙盒
- End-to-end Personalization 101 for E-Commerce

相关服务：

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 鼓励团队成员保持和增强自己的技能组合

团队必须增强自己的技能组合，以便采用新技术；并随需求和责任的变化继续提供支持，从而支持工作负载。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。支持团队成员获取和维持行业认证，以便验证和认可他们不断增强的技能。进行交叉培训，促进知识转移并降低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 活动和网络研讨会和 AWS Well-Architected Lab，这些资源提供了培训团队所需的指导、示例和详细演练。

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

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 The Official AWS Podcast 分享了各种其他有用的教育材料。

AWS 培训 和认证包括通过自定进度的数字课程进行的免费培训，以及按角色或领域制定的学习计划。您还可以报名参加讲师指导培训，进一步支持培养团队的 AWS 技能。

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

常见反模式：

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

建立此最佳实践的好处：组织有意识地投资于提高团队技能时，这还有助于加速和扩大云的采用和优化。有针对性的学习计划可推动创新，培养团队的运营能力，为处理各种事件做好准备。团队有意识地投资于最佳实践的实施和发展。团队士气高昂，团队成员重视自己对企业的贡献。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

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

实施步骤

1. 使用结构化的云宣传计划：AWS Skills Guild 提供咨询培训，可提高在云技能方面的信心并激发持续学习的文化。
2. 提供教育资源：专门安排时间，提供培训材料和实验室资源，并支持参加会议和加入专业组织，以便有机会向讲师和同行学习。让初级团队成员有机会接触资深团队成员，并让后者担任导师，或者让初级团队成员跟随资深团队成员工作，接触后者的工作方法和技能。鼓励学习与工作没有直接关系的内容，拓展视野。
3. 鼓励使用专家技术资源：利用 AWS re:Post 之类的资源来访问精选知识和活跃社区。
4. 建立和维护最新的知识库：使用 Wiki 和运行手册等知识共享平台。使用 AWS re:Post Private 创建自己的可重复使用的专家知识源，简化协作、提高工作效率并加速员工入职。
5. 团队教育和跨团队参与：为团队成员的继续教育需求进行规划。为团队成员提供（临时或永久）加入其他团队的机会，以便分享技能和最佳实践，惠及整个组织。
6. 支持获取和维护行业认证：支持团队成员获取和维护行业认证，以便验证他们所学到的知识并认可他们的成就。

实施计划的工作量级别：高

资源

相关最佳实践：

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

相关文档：

- AWS 白皮书 | Cloud Adoption Framework: People Perspective
- Investing in continuous learning to grow your organization's future
- AWS Skills Guild
- AWS 培训 和认证
- 支持
- AWS re:Post
- AWS 入门资源中心
- AWS Blog
- AWS Cloud 合规性
- AWS 文档
- The Official AWS Podcast。
- AWS 在线技术讲座
- AWS 活动和网络研讨会
- AWS Well-Architected Lab
- Amazon Builders' Library

相关视频：

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

OPS03-BP07 为团队配置适当的资源

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

期望结果：

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

常见反模式：

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

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

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

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

实施步骤

1. 借助员工生产率等相关的运营指标（例如，支持工作负载的成本或操作员在意外事件期间花费的时间），制定团队成功的成功标准。

2. 制定资源能力规划和检查机制，以便核实在需要时，是否有适当平衡的合格能力，并且这些能力是否可随时间进行调整。
3. 建立机制（例如，每月向团队发送调查问卷），以期了解影响团队的、与工作相关的挑战（如责任增加、技术变化、人员流失或支持的客户增加）。
4. 利用这些机制与团队互动，发现可能导致员工生产率面临挑战的趋势。团队受外部因素影响时，需重新评估目标并适当地调整执行性目标。确定阻碍团队进度的障碍。
5. 定期审查当前预置的资源是否仍然足够，是否需要额外资源，并做出适当调整来支持团队。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

- AWS Cloud Adoption Framework: People Perspective
- Becoming a Future-Ready Enterprise
- Prioritize your Employees' Skills to Drive Business Growth
- 高绩效组织 – 亚马逊双披萨团队
- How Cloud-Mature Enterprises Succeed

准备

问题

- OPS 4. 如何在工作负载中实现可观测性？
- OPS 5. 如何减少缺陷、简化修复和改进生产流程？
- OPS 6. 如何缓解部署风险？

- OPS 7. 如何知道您已经准备好支持某种工作负载？

OPS 4. 如何在工作负载中实现可观测性？

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要求作出数据驱动型决策。

最佳实践

- OPS04-BP01 确定关键绩效指标
- 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

OPS04-BP01 确定关键绩效指标

要在工作负载中实现可观测性，首先要了解其状态，并根据业务要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：

- 不明确的 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发展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：

- 易于发现问题：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，深入研究业务 KPI 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发现问题并加以解决。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要有效地定义工作负载 KPI，请执行以下操作：

1. 从业务成果开始：在深入研究指标之前，请先了解期望的业务成果。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成果产生直接影响。确定那些确实会产生直接影响的指标，但使用业务 KPI 来发现问题通常更为简单。
3. 使用 Amazon CloudWatch：使用 CloudWatch 来定义和监控代表 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，请保持 KPI 的相关性。
5. 让利益相关方参与其中：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 Observability Best Practices
- 《Amazon CloudWatch 用户指南》
- AWS Observability Skill Builder Course

相关视频：

- Developing an observability strategy

相关示例：

- One Observability 讲习会

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洞察，便于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，应用程序遥测可为构建、操作和改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，还可协助创建基准和识别异常情况。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发现问题。

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

期望结果：获得有关工作负载性能的切实可行的洞察。这些洞察有助于主动作出性能优化决策，提高工作负载稳定性，简化 CI/CD 流程，并有效地利用资源。

常见反模式：

- 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 中获得的洞察，可以作出以数据驱动型决策。
- 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：从遥测数据获得的洞察有助于完善流程和可靠地交付代码。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要为工作负载实现应用程序遥测，请使用 Amazon CloudWatch 和 AWS X-Ray 等 AWS 服务。Amazon CloudWatch 提供了一套全面的监控工具，可观察 AWS 和本地环境中的资源和应用程序。该服务会收集、跟踪和分析指标，整合和监控日志数据，并对资源的变化做出响应，从而增进对工

作负载运行方式的了解。同时，利用 AWS X-Ray，还可以跟踪、分析和调试应用程序，深入了解工作负载的行为。借助服务地图、延迟分布和跟踪时间表等功能，AWS X-Ray 可让您深入了解工作负载的性能和影响工作负载性能的瓶颈。

实施步骤

1. 确定要收集哪些数据：确定有助于深入了解工作负载运行状况、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。CloudWatch 代理还可用于收集 OpenTelemetry 或 X-Ray 跟踪数据，并将其发送到 X-Ray。
3. 对日志和指标实施异常检测：使用 CloudWatch Logs 异常检测功能和 CloudWatch Metrics 异常检测功能来自动识别应用程序操作中的异常活动。这些工具使用机器学习算法来检测异常情况并发出警报，从而增强了监控能力，加快了对潜在中断或安全威胁的响应速度。设置这些功能可主动管理应用程序的运行状况和安全性。
4. 保护敏感日志数据：使用 Amazon CloudWatch Logs 数据保护来掩蔽日志中的敏感信息。此功能会在访问敏感数据之前自动检测和掩蔽敏感数据，有助于维护隐私和合规性。实施数据掩蔽，以期安全地处理和保护个人身份信息（PII）等敏感详细信息。
5. 定义和监控业务 KPI：建立与业务结果一致的自定义指标。
6. 使用 AWS X-Ray 检测应用程序：除了部署 CloudWatch 代理之外，还必须对应用程序进行检测，以便发出跟踪数据。此过程可让您进一步了解工作负载的行为和性能。
7. 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。
8. 实现跨账户可观测性：借助 Amazon CloudWatch 跨账户可观测性，提高对多个 AWS 账户 的监控效率。利用该功能，可以将不同账户中的指标、日志和警报整合到一个视图中，从而简化管理，并提高对整个组织的 AWS 环境中已发现问题的响应速度。
9. 分析数据并据此采取行动：数据收集和标准化完成后，使用 Amazon CloudWatch 进行指标和日志分析，并使用 AWS X-Ray 进行跟踪分析。此类分析可得出有关工作负载运行状况、性能和行为的重要洞察，从而指导决策过程。

实施计划的工作量级别：高

资源

相关最佳实践：

- OPS04-BP01 定义工作负载 KPI

- OPS04-BP03 实施用户活动遥测
- OPS04-BP04 实施依赖项遥测
- OPS04-BP05 实施事务可追溯性

相关文档：

- AWS Observability Best Practices
- 《Amazon 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 Observability Skill Builder Course
- Amazon CloudWatch 的新功能
- AWS X-Ray 的新功能

相关视频：

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

相关示例：

- One Observability 讲习会
- 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，从未经筛选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这些问题。

期望结果：全面了解客户体验，主动检测问题，优化用户交互，从而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：

  - 问题检测延误：如果没有 RUM，可能要等到用户抱怨时，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，从而限制优化用户体验的能力。
- 应用程序没有综合事务功能：
  - 错过边缘案例：综合事务有助于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。没有综合事务，这些路径可能会出现故障并被忽视。
  - 在未使用应用程序时检查问题：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，发现并解决这些问题。
- 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洞察：了解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洞察，提高应用程序的响应能力和可靠性。

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

为利用 RUM 和综合事务进行用户活动遥测，AWS 提供了 Amazon CloudWatch RUM 和 Amazon CloudWatch Synthetics 等服务。指标、日志和跟踪数据，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，模拟这些路径的用户交互。

c. 安排和监控金丝雀按指定的间隔运行，确保进行一致的性能检查。

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洞察，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 确定关键绩效指标
- 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：

- Optimize applications through end user insights with Amazon CloudWatch RUM
- AWS on Air ft. Real-User Monitoring for Amazon CloudWatch

相关示例：

- One Observability 讲习会
- 适用于 Amazon CloudWatch RUM Web 客户端的 Git 存储库
- Using Amazon CloudWatch Synthetics to measure page load time

OPS04-BP04 实施依赖项遥测

要想监控工作负载所依赖的外部服务和组件的运行状况及性能，依赖项遥测必不可少。依赖项遥测提供有关与 DNS、数据库或第三方 API 等依赖项相关的可访问性、超时及其他关键事件的宝贵洞察。对应

用程序进行检测，发布有关这些依赖项的指标、日志和跟踪数据时，能更清楚地了解可能影响工作负载的潜在瓶颈、性能问题或故障。

期望结果：确保工作负载所依赖的依赖项按预期执行，让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，主动发现和解决这些问题。
- 全面视图：全面了解影响工作负载运行状况的内部和外部组件。
- 增强工作负载可扩展性：通过了解外部依赖项的可扩展性限制和性能特征来实现。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

从确定工作负载所依赖的服务、基础设施和流程开始，实施依赖项遥测。量化这些依赖项按预期运行时的良好状况，然后确定将需要哪些数据来衡量这些状况。利用这些信息，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洞察。当依赖项无法按需交付时，使用 AWS 工具来发现和量化影响。不断重新审视策略，考虑优先事项、目标和所获洞察的变化。

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：一旦清楚地了解了外部依赖项，就可以针对它们构建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水平协议或目标（SLA 或 SLT）。设置主动警报，在出现状态变化或性能偏差时发出通知。

3. 使用网络监控：使用网络检测仪和网络监视器，全面了解全球互联网和网络状况。这些工具有助于了解并应对影响外部依赖项的中断、破坏或性能下降。
4. 随时了解 AWS Health 的最新信息：AWS Health 是有关 AWS Cloud资源运行状况的权威信息来源。使用 AWS Health 可视化并接收有关任何当前服务事件和即将发生的更改（例如计划的生命周期事件）的通知，以便您可以采取措施来减轻影响。
   a. 通过 AWS 用户通知服务 创建要发送到电子邮件和聊天渠道且契合目标的 AWS Health 事件通知，并通过 Amazon EventBridge 或 AWS Health API 以编程方式与监控和警报工具集成。
   b. 通过与您可能已经通过 Amazon EventBridge 或 AWS Health API 使用的变更管理或 ITSM 工具（如 Jira 或 ServiceNow）集成，规划和跟踪需要采取行动的运行状况事件的进度。
   c. 如果您使用 AWS Organizations，请启用 organization view for AWS Health 以跨账户聚合 AWS Health 事件。
5. 使用 AWS X-Ray 检测应用程序：AWS X-Ray 让您能够深入了解应用程序及其底层依赖项的运行情况。通过从头到尾跟踪请求，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，可发现操作问题，预测何时可能出现严重问题，并建议可采取的具体行动。其可贵之处在于，可以让您深入了解依赖项，并确保这些依赖项不会成为操作问题的根源。
7. 定期监控：持续监控与外部依赖项相关的指标和日志。针对意外行为或性能下降设置警报。
8. 更改后进行验证：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 定义工作负载 KPI
- OPS04-BP02 实施应用程序遥测
- OPS04-BP03 实施用户活动遥测
- OPS04-BP05 实施事务可追溯性
- OP08-BP04 创建可操作的警报

相关文档：

- 《Amazon Personal AWS Health Dashboard User Guide》

- 《AWS Internet Monitor User Guide》
- AWS X-Ray 开发人员指南
- 《AWS DevOps Guru User Guide》

相关视频：

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

相关示例：

- AWS Health Aware
- Using Tag-Based Filtering to Manage AWS Health Monitoring and Alerting at Scale

OPS04-BP05 实施分布式跟踪

分布式跟踪提供了一种方法，可在请求遍历分布式系统的各个组件时对请求进行监控和可视化。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：快速发现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。

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

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，以便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 集成到应用程序中，以便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. 检测服务：验证从 AWS Lambda 函数到 EC2 实例的每项服务是否发送了跟踪数据。检测的服务越多，端到端视图就越清晰。
3. 整合 CloudWatch 真实用户监控和综合监控：将真实用户监控（RUM）和综合监控与 X-Ray 集成。这可捕捉实际的用户体验并模拟用户交互，从而发现潜在问题。
4. 使用 CloudWatch 代理：该代理可发送来自 X-Ray 或 OpenTelemetry 的跟踪数据，从而增强所获得洞察的深度。
5. 使用 Amazon DevOps Guru：DevOps Guru 使用来自 X-Ray、CloudWatch、AWS Config 和 AWS CloudTrail 的数据来提供切实可行的建议。
6. 分析跟踪数据：定期审查跟踪数据，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 中配置针对异常模式或延迟时间过长的警报，从而主动解决问题。
8. 持续改进：在添加或修改服务时，重新审视跟踪策略，以便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 确定关键绩效指标
- 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 User Guide》

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 for AWS X-Ray

OPS 5. 如何减少缺陷、简化修复和改进生产流程？

采用的方法需能够改进将更改应用于生产环境的流程，激活重构、快速质量反馈和错误修复。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。

最佳实践

- OPS05-BP01 使用版本控制
- OPS05-BP02 测试并验证更改
- 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 Git 等修订或源代码控制系统来管理代码和其它构件，例如基础设施的版本受控的 AWS CloudFormation 模板。

期望结果：团队协作编写代码。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，代码丢失了。
- 用更改内容覆盖现有代码后，重新启动应用程序，但其无法运行。您无法撤销所做更改。
- 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。他们与您联系要求停止写入锁定，以便他们可以完成自己的任务。
- 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。您无法撤销更改，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，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。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在版本受控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

相关视频：

- AWS re:Invent 2023 - How Lockheed Martin builds software faster, powered by DevSecOps

- AWS re:Invent 2023 - How GitHub operationalizes AI for team collaboration and productivity

OPS05-BP02 测试并验证更改

部署的每一项更改都必须经过测试，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更改。除应用程序代码更改外，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。

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。但有些情况下，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，以便创建反馈环路，提高软件质量。

期望结果：软件更改在交付前经过测试。开发人员可以访问测试结果和验证结果。组织制定了适用于所有软件更改的测试标准。

常见反模式：

- 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，而没有在预生产环境中进行测试。这些安全组导致客户无法访问应用程序。
- 修改了一个方法，但没有进行单元测试。该软件在部署到生产环境中后无法运行。

建立此最佳实践的好处：降低了软件部署更改的失败率。软件质量得到改进。开发人员提高了对其代码可行性的认识。可以放心地推出安全策略，支持组织实现合规性。可以提前测试自动扩缩策略更新等基础设施更改，从而满足流量需求。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

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

客户示例

作为持续集成管道的一部分，AnyCompany Retail 对所有软件构件进行几种类型的测试。他们实行测试驱动型开发，因此所有软件都要进行单元测试。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？ 测试范围内是否必须包含合规性或治理要求？ 是否需要进行代码质量测试？ 测试完成后，需要通知谁？
    1. AWS 部署管道参考架构包含一个权威的测试类型列表，可作为集成管道的一部分对软件构件执行这些测试。
2. 根据软件测试标准，利用必要的测试来检测应用程序。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. 使用 Amazon Q 开发者版，这是一款生成式人工智能工具，可以帮助创建单元测试案例（包括边界条件）、使用代码和注释生成函数以及实现已知算法。
    b. 使用 Amazon CodeGuru Reviewer 来测试应用程序代码是否存在缺陷。
    c. 使用 AWS CodeBuild 对软件构件执行测试。
    d. AWS CodePipeline 可以将软件测试编排到管道中。

资源

相关最佳实践：

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

相关文档：

- 采用测试驱动型开发方法
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE

- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q 开发人员中心
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Automated AWS CloudFormation Testing Pipeline with TaskCat and CodePipeline
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST, and DAST tools
- Getting started with testing serverless applications
- My CI/CD pipeline is my release captain
- 《在 AWS 上练习持续集成和持续交付白皮书》

相关视频：

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

相关资源：

- AWS Deployment Pipeline Reference Architecture - Application
- AWS Kubernetes DevSecOps 管道
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild
- Use Serverspec for test-driven development of infrastructure code

相关服务：

- Amazon Q 开发者版
- 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，这些值预计在资源的生命周期内会保持一致。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计会发生变化。例如，可以设置功能切换，通过配置更改来激活代码中的功能，或者在意外事件发生期间更改日志详细信息级别。

配置应在已知且一致的状态下部署。应该使用自动化检查功能来持续监控跨环境和区域的资源配置。这些控制措施应定义为自动化代码和管理，从而确保在各个环境中始终如一地应用规则。配置更改应通过商定的更改控制程序进行更新，并一致地应用，从而遵守版本控制。应用程序配置应独立于应用程序和基础设施代码进行管理。这可实现在多个环境中进行一致的部署。配置更改不会导致重建或重新部署应用程序。

期望结果：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终端用户和客户的任何影响。

常见反模式：

- 手动更新整个实例集中的 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实例集需要花费很长时间。在更改过程中，如果配置不一致会导致意外行为发生。
- 有人更新了安全组，Web 服务器变得无法访问。如果不知道进行了哪些更改，就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。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。

在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导致的错误，让配置更改可重复且可审核，并减少工作量。

在 AWS 上，可以使用 AWS Config 持续监控跨账户和区域的 AWS 资源配置。这有助于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 合规包根据预期或期望的配置对其进行审查。

对于在 Amazon EC2 实例、AWS Lambda、容器、移动应用程序或 IoT 设备上运行的应用程序中的动态配置，可以使用 AWS AppConfig 在各环境中对其进行配置、验证、部署和监控。

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 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. 根据业务要求和交付管道，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，以便尽可能减少错误配置的影响。
5. 将配置管理集成到 CI/CD 管道中。
6. 验证推送的所有更改。

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
- AWS CodeBuild
- AWS CodePipeline
- AWS CodeDeploy

相关视频：

- AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- 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 AWS 开发人员工具（例如 AWS CodeBuild、AWS CodePipeline 和 AWS CodeDeploy）等服务，来构建持续集成/持续部署（CI/CD）管道。

期望结果：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，该系统提供使用正确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，将可执行文件复制到生产系统中，但它无法启动。本地日志文件显示这是因为缺少依赖项所致。
- 成功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，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，无法重复支持成功构建应用程序的步骤。
- 执行为新版本创建的测试。花费了接下来一周的时间来设置测试环境，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。

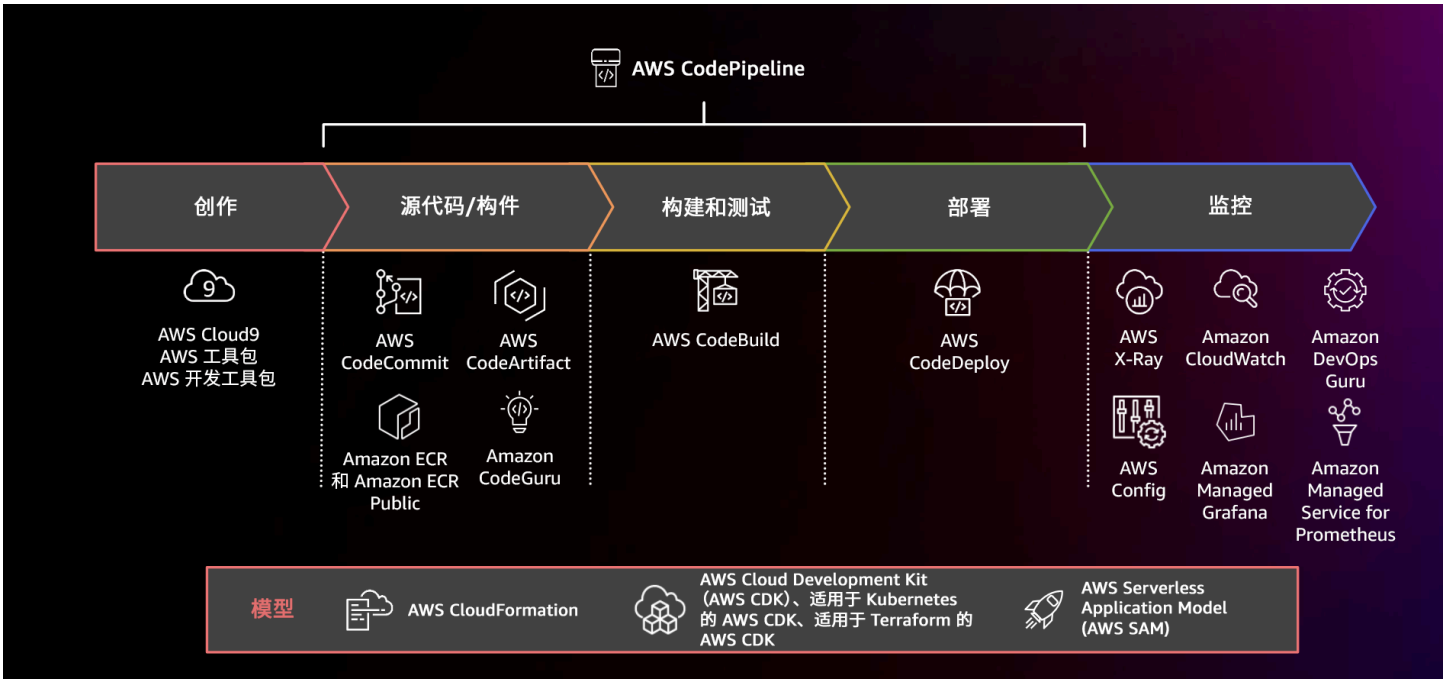
建立此最佳实践的好处：制定相应机制来管理活动的构建和部署。这样，可以减少执行重复任务的工作量，让团队成员腾出时间专注于高价值的创造性任务，还可以减少手动程序导致的错误。

在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，减少手动流程导致的错误，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，鼓励更频繁地进行更改，减少工作量并增进协作。

实施步骤



图中显示了使用 AWS CodePipeline 和相关服务的 CI/CD 管道

1. 使用版本控制系统来存储和管理资产（例如文档、源代码和二进制文件）。
2. 使用 CodeBuild 来编译源代码，运行单元测试，并生成可供部署的构件。
3. 使用 CodeDeploy 作为部署服务，可以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 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，减少手动流程导致的错误，进行扩展，并减少修补工作量。

补丁和漏洞管理是益处与风险管理活动的一部分。最好是具有不可变的基础设施并在经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。

AWS Health 是有关计划的生命周期事件以及其它影响 AWS Cloud 资源运行状况而需采取措施的事件的权威信息来源。您应该知道即将进行的更改和应执行的更新。计划的重大生命周期事件至少提前六个月发送。

Amazon EC2 Image Builder 提供了更新机器映像的管道。作为补丁管理的一部分，可以考虑使用 AMI 映像管道的亚马逊机器映像（AMI）或带有 Docker 映像管道的容器映像，而 AWS Lambda 提供适用于自定义运行时和其他库的模式用来消除漏洞。

应使用 Amazon EC2 Image Builder 管理适用于 Linux 或 Windows Server 映像的亚马逊机器映像（AMI）的更新。可以结合使用 Amazon Elastic Container Registry（Amazon ECR）和现有管道来管理 Amazon ECS 映像和 Amazon EKS 映像。Lambda 包含版本管理功能。

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。仅当补丁支持运营或业务成果时，才应该应用补丁。在 AWS 上，可以使用 AWS Systems Manager Patch Manager 自动完成托管系统的修补过程，并使用 Systems Manager 维护时段安排活动。

期望结果：AMI 和容器映像经过修补、处于最新状态，随时可以启动。可以跟踪所有已部署映像的状态，并了解补丁合规性。可以报告当前状态，并制定流程来满足合规性需求。

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工作负载。
- 在未通知开发人员的情况下自动修补开发人员环境。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。
- 尚未修补持久性实例上的现有商用软件。遇到软件问题并与供应商联系时，他们告知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。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。
- 您不知道维护资源所需的关键操作，例如强制版本更新，因为您未查看即将发生的计划生命周期事件和其它信息。您错失了计划和执行的关键时间，导致团队发生紧急变化，并可能造成影响或意外停机。

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，可以进行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，可减少部署补丁的工作量，并减少手动流程导致的错误。

在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、避免手动流程导致的错误，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. 为方案命名并确定其版本

d. 选择基础映像

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. 选择补丁操作方法。
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
- 使用 Patch Manager
- 使用补丁合规性报告
- AWS 开发人员工具

相关视频：

- CI/CD for Serverless Applications on AWS
- Design with Ops in Mind

相关示例：

- 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，记录最佳实践并使其保持最新。如果在组织中强制实施了共享标准，则必须制定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，记录标准并使其保持最新。

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更改基础设施，各个团队以不同的方式实施更改。

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，并充分实现开发工作的效益。记录和更新设计标准，让组织可以了解最新的最佳实践以及安全和合规性要求。

在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，制定一种机制来让设计标准保持最新。

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治理的架构。采用这些共享标准的团队可以从内置合规性和治理中受益。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. 组建一个跨职能团队，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、指南和治理要求。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。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，制定一种机制来让设计标准保持最新。
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相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 – 治理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营准备情况进行审查 – 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：

- Automate AWS Backups with AWS Service Catalog
- AWS Service Catalog Account Factory-Enhanced
- How Expedia Group built Database as a Service (DBaaS) offering using AWS Service Catalog
- Maintain visibility over the use of cloud architecture patterns
- Simplify sharing your AWS Service Catalog portfolios in an AWS Organizations setup

相关视频：

- AWS Service Catalog – Getting Started
- AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert

相关示例：

- AWS Service Catalog Reference Architecture

- AWS Service Catalog 讲习会

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。将这些实践合并到持续集成和交付流程中。

期望结果：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。更改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。
- 团队在生产中进行手动更改来解决缺陷问题。更改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得到捕获或记录。

建立此最佳实践的好处：通过采用提高代码质量的实践，能够帮助最大限度地减少引入生产中的问题。代码质量最佳实践包括结对编程、代码审查和实施人工智能生产力工具等。

在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。

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

客户示例

AnyCompany Retail 采用几种实践来提高代码质量。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. 使用 Amazon Q 开发者版，这是一款生成式人工智能工具，可以帮助创建单元测试案例（包括边界条件）、使用代码和注释生成函数、实现已知算法、检测代码中的安全策略违规行为和漏洞、检测机密、扫描基础设施即代码（IaC）、记录代码以及更快地学习第三方代码库。
   b. Amazon CodeGuru Reviewer 可以使用机器学习为 Java 和 Python 代码提供编程建议。

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-BP02 测试并验证更改
- OPS05-BP06 共享设计标准

相关文档：

- 采用测试驱动型开发方法
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q 开发人员中心
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- 《Agile Software Guide》
- My CI/CD pipeline is my release captain

- Automate code reviews with Amazon CodeGuru Reviewer
- 采用测试驱动型开发方法
- How DevFactory builds better applications with Amazon CodeGuru
- On Pair Programming
- RENGA Inc. automates code reviews with Amazon CodeGuru
- The Art of Agile Development: Test-Driven Development
- Why code reviews matter (and actually save time!)

相关视频：

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

相关服务：

- Amazon Q 开发者版
- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler

OPS05-BP08 使用多个环境

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

期望结果：您有多个环境，这些环境均反映合规性和治理需求。在通往生产的道路上，可以通过环境来测试和推广代码。

1. 组织可通过建立登录区来实现这一目标，登录区可提供治理、控制、账户自动化、联网、安全性和运营可观测性。通过使用多个环境来管理这些登录区功能。一个常见的例子是沙盒组织，用于开发和测试对基于 AWS Control Tower 的登录区的更改，其中包括 AWS IAM Identity Center 和 service control policies (SCPs) 等策略。所有这些因素都会对登录区内 AWS 账户的访问和操作产生重大影响。

2. 除了这些服务外，您的团队还可通过 AWS 和 AWS 合作伙伴发布的解决方案，或作为组织内部开发的自定义解决方案，来扩展登录区的功能。AWS 发布的解决方案示例包括 AWS Control Tower 的自定义选项（CfCT）和 AWS Control Tower Account Factory for Terraform (AFT)。
3. 您的组织对登录区的测试、推广代码和策略变更采用相同的原则，贯穿通往生产之路的各个环境。该策略为您的应用程序和工作负载团队提供了一个稳定且安全的登录区环境。

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。
- 对共享开发环境严苛的安全控制导致您无法试验新的服务和功能。
- 对生产系统执行负载测试，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，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。
- 您升级了一项关键的登录区功能，但此项更改会削弱您的团队为新项目或现有工作负载销售账户的能力。
- 您对 AWS 账户应用了新的控制，但此项更改会影响工作负载团队在其 AWS 账户内部署更改的能力。

建立此最佳实践的好处：当您部署多个环境时，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。对于诸如登录区之类的复杂功能，它可以显著降低变更风险，简化改进过程，并降低对环境进行关键更新的风险。使用登录区的组织自然会因其 AWS 环境中的多个账户而受益，包括账户结构、治理、网络和安全配置。随着时间推移，组织不断成长，登录区可以不断发展，以保护和组织您的工作负载和资源。

在未建立这种最佳实践的情况下暴露的风险等级：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，协助进行试验。提供单独的开发环境来协助并行工作，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，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。

诸如平台工程、网络和安全运营等团队通常在组织层面管理可满足不同要求的功能。单靠分离账户不足以为实验、开发和测试提供和维护单独的环境。在此类情况下，请创建单独的 AWS Organizations 实例。

资源

相关文档：

- AWS 实例调度器
- 什么是 AWS CloudFormation ？
- Organizing Your AWS Environment Using Multiple Accounts - Multiple organizations - Test changes to your overall AWS environment
- AWS Control Tower Guide

OPS05-BP09 频繁进行可逆的小规模更改

频繁进行可逆的小规模更改可以减少更改的范围和影响。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更改的范围和影响。这样可以提高故障排除工作的效率、加快修复速度，并支持回滚更改。

常见反模式：

- 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。
- 经常更改数据库架构，而不跟踪管理系统中的更改。
- 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：通过频繁部署小更改，可以加快开发速度。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，由于简化了恢复过程，因此实施更改的风险更小。更改过程的风险降低，更改失败的影响也减小。

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

频繁进行可逆的小规模更改可以减小更改的范围和影响。这可以简化故障排除、加快修复速度，并支持回滚更改。这还可以加快实现业务价值。

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
- Microservices - Observability

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。这可以减少手动流程导致的错误，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，以标识资源。标记资源，以便进行整理、成本核算、访问控制并有针对性地自动执行运营活动。

期望结果：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。对更改和配置进行全面的审计跟踪记录，可满足治理和合规性需求。流程是可重复的，并且可跨团队实现标准化。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，您完成了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，以便将其部署到生产环境中。
- 作为开发人员，您可以登录 AWS Management Console，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：通过实施自动化的构建和部署管理系统，可以减少手动流程导致的错误，并减少部署更改的工作量，有助于团队成员专注于实现业务价值。推广到生产环境后，交付速度也随之提高。

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

OPS 6. 如何缓解部署风险？

采用的方法需能够提供快速质量反馈，并在更改没有达到预期结果时实现快速恢复。使用这些实践可以减轻因部署更改而产生的问题的影响。

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。这样的示例策略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。单个发布版本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。

期望结果：已为更改失败准备了详细的恢复计划。此外，还缩小了发布内容的大小，以便最大限度地减少对其他工作负载组件的潜在影响。因此，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，但是无法访问其中一个子网。必须决定是回滚所有内容还是尝试修复无法访问的子网。做决定时，子网仍然无法访问。
- 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，很难撤销这些大批量的更改。
- 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。无法有效地跟踪和撤销手动更改。
- 由于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。
- 没有衡量因更改失败而导致的中断的总持续时间。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），并减少对业务的影响。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

发布团队采用的一致、有据可查的策略和实践，让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. 记录策略，该策略要求团队制定有效计划以便在指定时间内撤销更改。
   a. 策略应指定何时允许出现向前修复的情况。
   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程，与执行更改策略的工作流程保持一致，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. 确保回滚程序将代码恢复到已知的良好状态，尽可能避免意外事件。
3. 集成工具和工作流程，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，以便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，从而加快制定回滚决策的速度。
6. 衡量更改失败时的停机时间，以便不断改进恢复计划。

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 | 确保部署期间安全回滚
- 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程序。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，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。

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。

期望结果：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实现业务价值，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，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。
- 发布版本包含用于更新现有资源的基础设施即代码（IaC）。不确定 IaC 是会成功运行，还是会对资源造成影响。
- 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：在预生产环境中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。

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

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，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）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. 这会触发 AWS CodePipeline 中的手动批准步骤，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，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. 使用 Amazon CloudWatch、AWS CloudTrail 和 Amazon SNS 事件通知监控部署。
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. 部署问题疑难解答。
8. 成功验证上述步骤后应启动手动审批工作流程，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 测试并验证更改

相关文档：

- AWS Builders' Library | 自动实现无需干预的安全部署 | 测试部署
- AWS 白皮书 | 在 AWS 上练习持续集成和持续交付
- 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，用于验证是否达成期望结果，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。

期望结果：组织使用持续集成/持续交付（CI/CD）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：

- 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。这限制了为客户更新功能的频率。
- 通过修改现有系统来执行可变部署。发现更改不成功后，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。

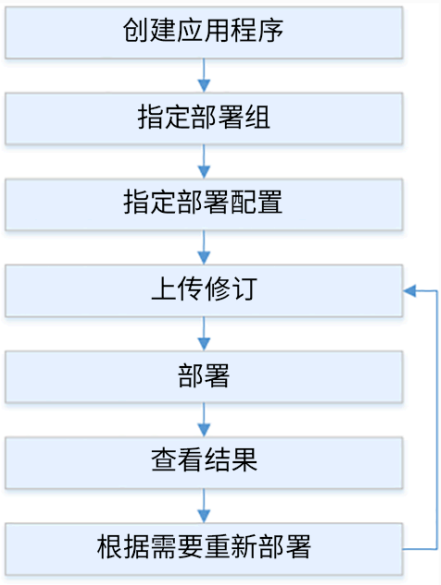
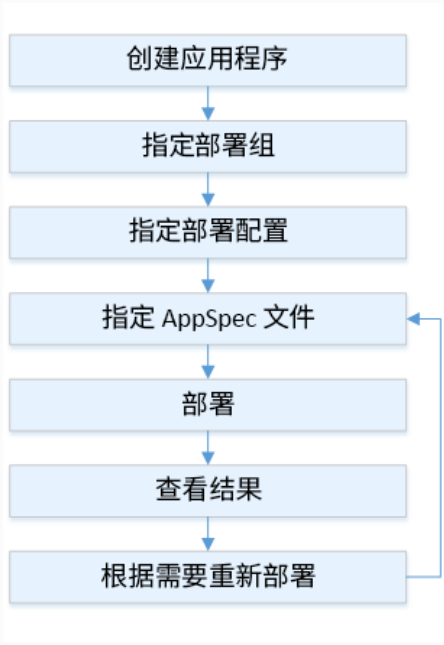
在未建立这种最佳实践的情况下暴露的风险等级：中

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，以便最大限度地减少部署错误，以达成零部署失败为目标。

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，不会对用户造成任何可察觉影响。为了实现这一目标，公司建立了部署模式（参阅以下工作流程图），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。

适用于 Amazon EC2 的 CodeDeploy 工作流程	适用于 Amazon ECS 的 CodeDeploy 工作流程	适用于 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 AWS CodeDeploy 等自动化部署系统。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（参阅前面的工作流程图）。
   a. 在适用的情况下，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. 对诸如 Amazon Aurora 和 Amazon RDS 之类的数据库使用蓝/绿部署。
4. 使用 Amazon CloudWatch、AWS CloudTrail 和 Amazon Simple Notification Service（Amazon SNS）监控部署。
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. 部署问题疑难解答。

实施计划的工作量级别：中

资源

相关最佳实践：

- OPS05-BP02 测试并验证更改
- 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 | 自动实现无需干预的安全部署 | 生产部署
- AWS Builders Library | My CI/CD pipeline is my release captain | Safe, automatic production releases
- AWS 白皮书 | 在 AWS 上练习持续集成和持续交付 | 部署方法
- AWS CodeDeploy 用户指南
- Working with deployment configurations in AWS CodeDeploy
- 设置 API Gateway 金丝雀版本部署
- 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
- Workshop | Building CI/CD pipelines for Lambda canary deployments using AWS CDK
- Workshop | Building your first DevOps Blue/Green pipeline with Amazon ECS
- Workshop | Building your first DevOps Blue/Green pipeline with Amazon EKS
- Workshop | EKS GitOps with ArgoCD

- Workshop | CI/CD on AWS Workshop
- Implementing cross-account CI/CD with AWS SAM for container-based Lambda functions

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，需要制定一项策略，用于在预生产环境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，从而验证已经部署的更改。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。

期望结果：自动化测试和回滚策略已集成到持续集成/持续交付（CI/CD）管道中。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终端用户和客户的任何影响。例如，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，则在管道工作流程中启动自动回滚。

常见反模式：

- 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，很难撤销这些大批量的更改。
- 部署流程包括一系列手动步骤。将更改部署到工作负载后，启动部署后测试。完成测试之后，发现工作负载不可操作，而且客户断开了连接。然后，开始回滚到之前的版本。所有这些手动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。
- 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，这极大地降低了在自动化测试功能上的投资回报率。
- 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，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撤销所有更改，导致回滚过程复杂且效率低下。
- 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，必须先解决这些缺陷，才能开始测试冲刺二的可交付成果，这延误了整个发布过程，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。通过自动回滚到之前的版本，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。

在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，以便更快地确认期望结果。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，并减少手动流程导致的错误。将测试工具与管道工作流程集成，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. 根据业务要求和管道投资，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程。
   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. 将自动化测试和回滚工具集成到 CI/CD 管道中。
   a. 为更改制定明确的成功标准。
   b. 监控并观察以便检测这些标准，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，允许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，允许在应用程序外部，每次将新版本的单个功能标记为打开和关闭，以便每次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，以便确定对自动化和工作流程的哪些部分进一步进行优化。

a. 编制测试结果报告，以便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 | 确保部署期间安全回滚
- Redeploy and rollback a deployment with AWS CodeDeploy
- 8 best practices when automating your deployments with AWS CloudFormation

相关示例：

- Serverless UI testing using Selenium, AWS Lambda, AWS Fargate, and AWS Developer Tools

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

OPS 7. 如何知道您已经准备好支持某种工作负载？

评估工作负载、流程及程序和工作人员的操作准备情况，以便了解与工作负载相关的操作风险。

最佳实践

- OPS07-BP01 确保员工能力
- OPS07-BP02 确保以一致的方式对运营准备情况进行审查
- 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 作出明智的决策来部署系统和变更
- OPS07-BP06 为生产工作负载创建支持计划

OPS07-BP01 确保员工能力

制定一种机制来验证是否有适当数量训练有素的员工来支持工作负载。必须针对构成工作负载的平台和服务对他们进行培训。为他们提供运营工作负载所需的知识。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。拥有足够数量的员工，以便在值班和休假期间轮换，避免出现倦怠。

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。
- 针对构成工作负载的软件和服务，对员工进行培训。

常见反模式：

- 部署工作负载，但没有对团队成员进行培训来运营所使用的平台和服务。
- 员工数量不足，无法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，同时降低出现倦怠的风险。

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

确认是否有足够的训练有素的员工来支持工作负载。确认是否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配备了适当的人员，并经过培训。他们有足够的工程师支持轮流值班。他们针对构建工作负载的软件和平台对员工进行培训，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. 分配足够数量的人员来运营和支持工作负载，包括随叫随到的职责、安全问题和生命周期事件，如终止支持和证书轮换任务。
2. 针对构成工作负载的软件和平台方面对员工进行培训。
   a. AWS 培训和认证有一个关于 AWS 的课程库。这里提供线上和线下的免费和付费课程。
   b. AWS 举办活动和网络研讨会，可以从中向 AWS 专家学习。
3. 定期执行以下操作：
   - 随着运营条件和工作负载发生变化，评估团队规模和技能。
   - 调整团队规模和技能来满足运营要求。
   - 验证通过 AWS Health 来解决计划内生命周期事件、计划外安全性和运营通知的能力。

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

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营准备情况进行审查

使用运营准备情况审查（ORR），确保可以运营工作负载。ORR 是 Amazon 开发的一种机制，用于验证团队是否可以安全地运营工作负载。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、运营流程、事件管理和发布质量。我们的错误更正（CoE）流程是这些项目的主要推动因素。意外事件后分析应该可以推动自己的 ORR 演进。ORR 不仅在于遵循最佳实践，还在于预防以前的事件再次发生。最后，ORR 中还可以包括安全、治理和合规性方面的要求。

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。在工作负载上定期重新运行 ORR 可以收集任何偏离最佳实践的情况。可以准备用于新服务发布的 ORR 核对清单以及用于定期审查的 ORR。这有助于遵循最新制定的最佳实践，并吸取从意外事件后分析中学到的经验教训。随着对云的使用日趋成熟，可以将 ORR 要求作为默认设置整合到自己的架构中。

期望结果：已准备好 ORR 核对清单，其中包括适用于组织的最佳实践。在工作负载发布之前执行 ORR。在整个工作负载生命周期中定期执行 ORR。

常见反模式：

- 您启动了工作负载，但不知道自己是否能够运营工作负载。
- 在验证工作负载以便发布时，没有包括治理和安全要求。
- 没有定期重新评估工作负载。
- 在未准备好所需程序的情况下发布工作负载。
- 在多个工作负载中发现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：

- 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程序。
- 在工作负载的整个软件生命周期中执行 ORR。

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

ORR 关系到两点：流程和核对清单。ORR 流程应该由组织采用并获得执行发起人支持。至少，在工作负载正式公开发布之前执行 ORR。在整个软件开发生命周期中执行 ORR，可确保软件始终遵循新的最佳实践或新要求。ORR 核对清单应包括组织的配置项目、安全要求和治理要求以及最佳实践。随着时间的推移，可以使用 AWS Config、AWS Security Hub 和 AWS Control Tower Guardrails 等服务将源自 ORR 的最佳实践构建到防护机制中，以便自动检测最佳实践。

客户示例

在经历了多起生产意外事件之后，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、治理要求和合规性要求，以及从中断中学到的经验教训。他们在发布新工作负载之前执行 ORR。每个工作负载会每年执行一次 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。随着时间的推移，AnyCompany Retail 使用 AWS Config 来检测某些最佳实践，加快了 ORR 流程。

实施步骤

有关 ORR 的更多信息，请阅读《Operational Readiness Reviews（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、运营和开发部门的代表。
2. 让每个利益相关方至少提一项要求。对于第一次迭代，请尝试将项目数限制为不超过三十个。
    - Appendix B: Example ORR questions 源自《Operational Readiness Reviews（ORR）白皮书》，包含在开始着手时可借鉴的示例问题。
3. 在电子表格中收集要求。
    - 您可以使用 AWS Well-Architected Tool 中的自定义剖析来开发 ORR，并在账户和 AWS 组织中共享这些剖析。
4. 确定一个工作负载来执行 ORR。最好选择发布前的工作负载或内部工作负载。
5. 运行 ORR 核对清单并记录任何发现结果。如果采取了缓解措施，发现结果可能是可接受的。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积压工作项中，并在发布之前处理它们。
6. 在一段时间后，继续在 ORR 核对清单中添加最佳实践和要求。

使用 Enterprise Support 的 支持 客户可以向其技术客户经理申请运营准备情况审查讲习会。该讲习会是一个交互式逆向工作会议，旨在制定自己的 ORR 核对清单。

实施计划的工作量级别：高。在组织中采用 ORR 实践需要获得高管支持以及利益相关方的支持。利用整个组织的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 – 治理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，必须具备所需的人员。
- SEC01-BP03 确定和验证控制目标 – 安全控制目标进一步完善了 ORR 要求。
- REL13-BP01 定义停机和数据丢失的恢复目标 – 灾难恢复计划是一项很好的 ORR 要求。
- COST02-BP01 根据组织要求制定政策 – 成本管理政策非常适合包含在 ORR 核对清单中。

相关文档：

- AWS Control Tower - Guardrails in AWS Control Tower
- AWS Well-Architected Tool - Custom Lenses
- Operational Readiness Review Template by Adrian Hornsby
- Operational Readiness Reviews（ORR）白皮书

相关视频：

- AWS 支持s You | Building an Effective Operational Readiness Review (ORR)

相关示例：

- 运营准备情况审查（ORR）剖析示例

相关服务：

- AWS Config

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

OPS07-BP03 使用运行手册执行程序

运行手册是实现特定结果的书面流程。运行手册由人们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期望结果。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主要版本，运行手册都是一个成文的流程，无论谁使用，都能获得一致的结果。运行手册应发布在中央位置，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。运行手册还应包括关于错误处理、工具、权限、异常和出现问题时进行上报的指导。

随着组织日益成熟，应开始实现运行手册自动化。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让步骤更容易执行。自动化前几本运行手册后，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：团队有一系列执行工作负载任务的分步指南。运行手册包含期望结果、必要的工具和权限，以及关于错误处理的说明。运行手册存储在一个中央位置（版本控制系统）并经常更新。例如，在应用程序发出警报、出现操作问题和计划内生命周期事件期间，运行手册可为团队提供监控、沟通和响应关键账户 AWS Health 事件的功能。

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。
- 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：

- 降低手动任务的错误率。
- 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。
- 可以自动化运行手册来减少工作量。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

根据组织的成熟程度，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中央位置。一旦运行手册编写完成，让团队中的其他人运行它来进行验证。随着程序的发展，根据变更管理流程更新运行手册。

随着组织日益成熟，文本运行手册应实现自动化。使用 AWS Systems Manager Automation 等服务，可以将纯文本转换为可以根据工作负载运行的自动化代码。这些自动化代码可以根据发生的事件运行，从而减轻维持工作负载的运营负担。AWSSystems Manager Automation 还提供了低代码视觉对象设计体验，可以更轻松地创建自动化运行手册。

客户示例

AnyCompany Retail 必须在软件部署期间执行数据库架构更新。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。该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

如果当前没有文档存储库，则版本控制存储库是开始构建运行手册库的理想之处。可以使用 Markdown 构建运行手册。我们提供了一个示例运行手册模板，您可以用该模板开始构建运行手册。

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

1. 如果当前没有文档存储库或 Wiki，请在版本控制系统中创建一个新的版本控制存储库。
2. 确定没有运行手册的流程。理想流程是半定期执行的流程，步骤少，且故障影响小。
3. 在文档存储库中，使用模板创建新的草稿 Markdown 文档。填写运行手册书名以及“运行手册信息”下的必填字段。

4. 从第一步开始，填写运行手册的“步骤”部分。
5. 将运行手册分发给团队成员。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. 将运行手册发布到内部文档存储区。发布后，告诉团队和其他利益相关方。
7. 随着时间的推移，将构建运行手册库。随着该库的增长，开始努力实现运行手册的自动化。

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

资源

相关最佳实践：

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

相关文档：

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

相关视频：

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

相关示例：

- Well-Architected Lab：使用行动手册和运行手册实现运营自动化
- AWS Blog 文章：Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- 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
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Gitlab – 运行手册
- Rubix – 用于在 Jupyter Notebook 中构建运行手册的 Python 库
- 使用文档生成器创建自定义运行手册

相关服务：

- AWS Systems Manager Automation

OPS07-BP04 根据行动手册调查问题

行动手册是用于调查意外事件的分步指南。发生意外事件时，行动手册用于开展调查，以及确定影响范围和根本原因。行动手册可用于从失败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根本原因带来的风险。行动手册是组织意外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的发现过程。引导用户由外而内地进行思考，应执行哪些步骤来诊断意外事件？ 如果行动手册中需要特殊工具或提升的权限，行动手册中会明确定义。制定沟通计划，以便向利益相关方提供有关调查状态的最新信息，这是事件响应计划的关键组成部分。在无法确定根本原因的情况下，行动手册应具有上报计划。如果确定了根本原因，行动手册应指向介绍如何解决根本原因的运行手册。行动手册应集中存储并定期维护。如果行动手册用于特定提醒，请向团队提供关于提醒中行动手册的提示。

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

期望结果：组织有针对常见意外事件的行动手册。行动手册存储在中心位置，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：

- 没有调查意外事件的标准方法。

- 团队成员依靠肌肉记忆或对机构的了解，对失败的部署进行故障排除。
- 新的团队成员将了解如何通过试错法来调查问题。
- 调查问题的最佳实践无法在团队间分享。

建立此最佳实践的好处：

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

在未建立这种最佳实践的情况下暴露的风险等级：中

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，制定了完全自动化的行动手册。

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

随着组织日趋成熟，文本形式的行动手册应实现自动化。使用 AWS Systems Manager Automations 等服务，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来响应事件，从而减少发现和解决意外事件所需的平均时间。

客户可以使用 AWS Systems Manager Incident Manager 来响应意外事件。此服务提供了一个单一界面，可对意外事件进行分类、在发现和缓解问题期间通知利益相关方，并在整个意外事件中进行协作。其使用 AWS Systems Manager Automations 加快检测和恢复的速度。

客户示例

一个生产意外事件影响了 AnyCompany Retail。随时待命的工程师根据行动手册调查了问题。随着他们逐步地解决问题，他们不断为行动手册中确定的关键利益相关方提供最新信息。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

如果当前没有文档存储库，建议为行动手册库创建版本控制存储库。可以使用 Markdown 制定行动手册，该服务与大多数行动手册自动化系统兼容。如果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. 确定需要进行调查的常见问题。这应该是根本原因范围限于几个问题且解决方案风险较低的场景。
3. 利用 Markdown 模板，填写“行动手册书名”部分，并填写“行动手册信息”下的字段。
4. 填写故障排除步骤。尽可能清楚地填写要采取哪些行动，或者应调查哪些方面。
5. 将行动手册分发给团队成员，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行动手册，并告知团队和任何利益相关方。
7. 随着添加更多的行动手册，这个行动手册库将会不断扩大。拥有多个行动手册后，可以开始使用 AWS Systems Manager Automations 等工具自动执行行动手册，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。行动手册应该是存储在中心位置的文本文档。对于更加成熟的组织，将转为自动化行动手册。

资源

相关最佳实践：

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

相关文档：

- AWS Well-Architected Framework: Concepts: Playbook development
- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager：使用运行手册
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

相关视频：

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

相关示例：

- AWS 客户行动手册框架
- AWS Systems Manager：自动化演练
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Rubix – 用于在 Jupyter Notebook 中构建运行手册的 Python 库
- 使用文档生成器创建自定义运行手册

相关服务：

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 作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益处和产生的风险。确认所有变更符合治理要求。

期望结果：

- 将变更部署到工作负载时作出明智的决策。
- 变更符合治理要求。

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治理要求的变更。
- 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治理政策。

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。确保所有变更均符合治理要求。评估将变更部署到工作负载所获得益处和产生的风险。

客户示例

AnyCompany Retail 定期执行故障演练来验证不成功变更的流程。他们在共享的 Wiki 中记录流程并经常更新。所有变更均符合治理要求。

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。制定将启动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产生的风险之间进行权衡。

2. 确认所有变更均符合治理政策。
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。运行桌面演练，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：中。实施故障演练的实践需要整个组织内的利益相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 – 治理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 – 制定计划来缓解失败的部署，并使用故障演练来进行验证。
- 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策很重要。

相关文档：

- 亚马逊云科技：风险与合规性
- AWS 责任共担模式
- Governance in the AWS Cloud: The Right Balance Between Agility and Safety

OPS07-BP06 为生产工作负载创建支持计划

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来满足生产服务水平需求。以防出现服务中断或软件问题，这些依赖项的支持计划必不可少。记录支持计划以及如何向所有服务和软件供应商请求支持。实施机制，以便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。
- 根据服务水平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：

- 没有制定面向关键软件供应商的支持计划。工作负载受到影响，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。没有关于如何提出支持案例的文档。

建立此最佳实践的好处：

- 通过适当的支持级别，可以在满足服务水平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。
- 发生意外事件时，软件和服务供应商可协助排除故障。

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来满足服务水平需求。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激活 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。实施机制，让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，他们在具有生产工作负载的所有账户上都激活了 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. 与组织内的利益相关方合作确定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水平需求。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 支持 更快响应，强烈建议订阅此支持。如果没有高级支持，则必须制定行动计划来处理问题，而这需要 AWS 支持 的帮助。AWS 支持 提供工具和技术的组合、人员和计划，旨在主动协助您优化性能、降低成

本和加快创新速度。此外，AWS Business Support 还提供其它优势，包括 API 对 AWS Trusted Advisor 和 AWS Health 的访问权限以便与系统进行编程集成，以及其它访问方法，例如 AWS Management Console和 Amazon EventBridge 渠道。

4. 在知识管理工具中记录支持计划。包括如何请求支持、在提出支持案例时向谁发出通知以及在发生意外事件时如何上报。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。在知识管理系统中记录和分享支持最佳实践，可以确认团队是否知道在出现生产问题时该怎么做。

资源

相关最佳实践：

- OPS02-BP02 确定流程和程序负责人

相关文档：

- AWS 支持 Plans

相关服务：

- AWS Business Support
- AWS Enterprise Support

运营

问题

- OPS 8. 如何在组织中利用工作负载可观测性？
- OPS 9. 如何了解自己的运营状况？
- OPS 10. 如何应对工作负载事件和运营事件？

OPS 8. 如何在组织中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，全面了解工作负载的性能并有效地解决问题。

最佳实践

- 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
- OPS08-BP05 创建控制面板

OPS08-BP01 分析工作负载指标

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

期望结果：准确洞察工作负载性能，推动作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。
- 在问题影响业务成果之前主动发现和缓解问题。

在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。Amazon CloudWatch 异常检测和 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，或行为模式更适合进行异常检测时。

实施步骤

1. 分析和审查：定期审查和解读工作负载指标。
   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. 了解数据中高峰、低谷或模式的重要性。
2. 利用 Amazon CloudWatch：使用 Amazon CloudWatch 获取集中视图和进行深入分析。
   a. 配置 CloudWatch 控制面板，以可视化形式呈现指标，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 CloudWatch 异常检测，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，以监控跨越一个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区域的指标数据，从而识别趋势和异常情况。
   f. 应用 CloudWatch 指标数学，对指标进行转换、汇总或执行计算，从而获得更深入的洞察。
3. 采用 Amazon DevOps Guru：加入 Amazon DevOps Guru，以便利用其机器学习增强的异常检测功能，识别无服务器应用程序操作问题的早期迹象，并在对客户造成影响之前将其修复。
4. 根据洞察进行优化：根据指标分析作出明智的决策，以便调整和改进工作负载。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

- The Wheel b博客 – 强调持续审查指标的重要性
- Percentile are important
- 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 讲习会
- 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方面至关重要。通过高效地筛选、以可视化方式呈现和解读日志数据，可以持续优化应用程序性能和安全性。

期望结果：通过全面的日志分析获得对应用程序行为和运行的丰富洞察，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。
- 没有使用可进行日志分析的全套工具，导致错过关键洞察。
- 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：

- 主动发现运行瓶颈、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，有助于进行调试和故障排除。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

Amazon CloudWatch Logs 是一款用于日志分析的强大工具。利用 CloudWatch Logs Insights 和 Contributor Insights 等集成功能，可以直观且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，以便将日志发送到 CloudWatch Logs。
2. 使用日志异常检测：利用 Amazon CloudWatch Logs 异常检测功能来自动识别异常日志模式并发出警报。该工具有助于主动管理日志中的异常情况，及早检测到潜在问题。
3. 设置 CloudWatch Logs Insights：使用 CloudWatch Logs Insights 以交互方式进行搜索，并分析日志数据。
   a. 创建查询来提取模式、以可视化形式呈现日志数据并获得切实可行的洞察。
   b. 使用 CloudWatch Logs Insights 模式分析来分析和可视化频繁使用的日志模式。该功能有助于了解日志数据中的常见运行趋势和潜在异常值。
   c. 使用 CloudWatch Logs 比较（diff）对不同时间段或不同日志组之间进行差异分析。利用这一功能可查明变更，并评测其对系统性能或行为的影响。
4. 使用 Live Tail 实时监控日志：使用 Amazon CloudWatch Logs Live Tail 实时查看日志数据。可以在应用程序运行活动发生时主动对其进行监控，即时了解系统性能和潜在问题。
5. 利用 Contributor Insights：使用 CloudWatch Contributor Insights 来识别 IP 地址或用户代理等高基数维度的用量最高者。
6. 实施 CloudWatch Logs 指标筛选条件：配置 CloudWatch Logs 指标筛选条件，将日志数据转换为可操作的指标。这允许设置警报或进一步分析模式。
7. 实施 CloudWatch 跨账户可观测性：监控跨越一个区域内多个账户的应用程序并对其进行故障排除。
8. 定期审查和完善：定期审查日志分析策略，以便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 确定关键绩效指标
- OPS04-BP02 实施应用程序遥测
- OPS08-BP01 分析工作负载指标

相关文档：

- Analyzing Log Data with CloudWatch Logs Insights

- Using CloudWatch Contributor Insights
- Creating and Managing CloudWatch Log Metric Filters

相关视频：

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

相关示例：

- CloudWatch Logs Sample Queries
- One Observability 讲习会

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操作过程至关重要。通过以可视化方式呈现和理解各个组件之间的交互情况，可以微调性能，识别瓶颈并增强用户体验。

期望结果：清晰地了解应用程序的分布式操作，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作出明智的决策。
- 通过优化的组件交互来改善用户体验。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

AWS X-Ray 提供了分析跟踪数据的完整套件，可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行洞察的深度。

实施步骤

以下步骤提供了一种结构化方法，以便使用 AWS 服务有效地实施跟踪数据分析：

1. 集成 AWS X-Ray：确保 X-Ray 已与应用程序集成，以便捕获跟踪数据。
2. 分析 X-Ray 指标：使用服务地图深入研究从 X-Ray 跟踪数据中得出的指标，例如延迟、请求率、故障率和响应时间分布等，以便监控应用程序的运行状况。
3. 使用 ServiceLens：利用 ServiceLens 地图增强服务和应用程序的可观测性。这允许以集成方式查看跟踪数据、指标、日志、警报和其他运行状况信息。
4. 启用 X-Ray Insights：
   a. 开启 X-Ray Insights，可自动检测跟踪数据中的异常情况。
   b. 研究洞察以查明模式并确定根本原因，例如故障率或延迟增加。
   c. 查阅洞察时间表，按时间顺序分析检测到的问题。
5. 使用 X-Ray Analytics：X-Ray Analytics 允许全面探索跟踪数据、查明模式并提取洞察。
6. 使用 X-Ray 中的组：在 X-Ray 中创建组，根据高延迟等标准筛选跟踪数据，从而进行更有针对性的分析。
7. 加入 Amazon DevOps Guru：利用 Amazon DevOps Guru 从机器学习模型中受益，查明跟踪数据中的操作异常。
8. 使用 CloudWatch Synthetics：使用 CloudWatch Synthetics 创建用于持续监控端点和工作流程的金丝雀。这些金丝雀可以与 X-Ray 集成来提供跟踪数据，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）：借助 AWS X-Ray 和 CloudWatch RUM，可以分析和调试从应用程序的终端用户开始，经过下游 AWS 托管服务的请求路径。可帮助您识别影响最终用户的延迟趋势和错误。
10. 与日志关联：将跟踪数据与 X-Ray 跟踪视图中的相关日志关联，从而详细了解应用程序行为。这允许查看与跟踪的事务直接关联的日志事件。
11. 实施 CloudWatch 跨账户可观测性：监控跨越一个区域内多个账户的应用程序并对其进行故障排除。

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：

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

相关视频：

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

相关示例：

- One Observability 讲习会
- 使用 AWS Lambda 实施 X-Ray
- CloudWatch Synthetics Canary Templates

OPS08-BP04 创建可操作的警报

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

期望结果：接收及时、相关且可操作的警报，以便快速发现和缓解潜在问题，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，因此很难了解问题对业务的影响。
- 忽视解决根本原因，导致针对同一问题出现重复警报。

建立此最佳实践的好处：

- 关注可操作的相关警报，减少警报疲劳。
- 主动检测和缓解问题，增加系统的正常运行时间并提高可靠性。
- 与常用的警报和通信工具集成，增强团队协作并更快解决问题。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

实施步骤

1. 确定关键绩效指标（KPI）：确定应用程序的 KPI。警报应与这些 KPI 相关联，以便准确反映业务影响。
2. 实施异常检测：
   - 使用 Amazon CloudWatch 异常检测：将 Amazon CloudWatch 异常检测设置为自动检测异常模式，这有助于仅针对真正的异常生成警报。
   - 使用 AWS X-Ray Insights：
     a. 设置 X-Ray Insights，检测跟踪数据中的异常。
     b. 配置 X-Ray Insights 的通知，以便在检测到问题时收到警报。
   - 与 Amazon DevOps Guru 集成：
     a. 利用 Amazon DevOps Guru 的机器学习功能，结合现有数据来检测操作异常。
     b. 导航到 DevOps Guru 中的通知设置以设置异常警报。
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。
   1. 使用 Amazon EventBridge 规则监控 AWS Health 事件，或者以编程方式与 AWS Health API 集成，以便在收到 AWS Health 事件时自动执行操作。这些可以是常规操作，例如将所有计划的生命周期事件消息发送到聊天界面，也可以是特定操作，例如在 IT 服务管理工具中启动工作流程。
4. 减少警报疲劳：尽量减少非关键警报。团队接收到大量无关紧要的警报时，他们可能无法监督关键问题，从而降低警报机制的整体有效性。

5. 设置复合警报：使用 Amazon CloudWatch 复合警报合并多个警报。
6. 与警报工具集成：纳入 Ops Genie 和 PagerDuty 等工具。
7. 加入聊天应用程序中的 Amazon Q 开发者版：集成聊天应用程序中的 Amazon Q 开发者版，以便将警报转发给 Amazon Chime、Microsoft Teams 和 Slack。
8. 基于日志的警报：使用 CloudWatch 中的日志指标筛选条件，根据特定的日志事件创建警报。
9. 审查和迭代：定期重新审视和完善警报配置。

实施计划的工作量级别：中

资源

相关最佳实践：

- OPS04-BP01 确定关键绩效指标
- 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 告警
- Create a composite alarm
- Create a CloudWatch alarm based on anomaly detection
- DevOps Guru Notifications
- X-ray insights notifications
- 使用交互式 ChatOps 对 AWS 资源进行监控、操作和故障排除
- Amazon CloudWatch Integration Guide | PagerDuty
- Integrate Opsgenie with Amazon CloudWatch

相关视频：

- Create Composite Alarms in Amazon CloudWatch
- Amazon Q Developer in chat applications Overview
- AWS On Air ft. Mutative Commands in Amazon Q Developer in chat applications

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 Amazon Q Developer in chat applications
- One Observability 讲习会

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。虽然控制面板提供了重要的可视化界面，但不应取代警报机制，而是作为警报机制的补充。经过精心设计的控制面板不仅能迅速洞察系统的运行状况和性能，还能为利益相关方提供有关业务成果和问题影响的实时信息。

期望结果：

使用可视化形式，清晰地了解系统和业务运行状况，并据此采取行动。

常见反模式：

- 指标过多，控制面板过于复杂。
- 依靠没有警报功能的控制面板进行异常检测。
- 不会随着工作负载的发展变化而更新控制面板。

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。

在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清楚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，控制面板应该有助于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，可用于在单一视图中监控资源，即便是分布在不同 AWS 区域 和账户的资源，也能对其进行监控。

实施步骤

1. 创建基本控制面板：在 CloudWatch 中创建一个新的控制面板，并给它起一个描述性名称。
2. 使用 Markdown 小组件：在深入研究指标之前，请使用 Markdown 小组件在控制面板顶部添加文本上下文。文本上下文应该说明控制面板涵盖的内容、所呈现指标的重要性，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：在适当的地方加入控制面板变量，从而实现动态和灵活的控制面板视图。
4. 创建指标小组件：添加指标小组件，以可视化形式呈现应用程序发出的各种指标，定制这些小组件，以便有效呈现系统运行状况和业务成果。
5. 日志洞察查询：利用 CloudWatch Log Insights 从日志中获取可操作的指标，并在控制面板上显示这些洞察。
6. 设置警报：将 CloudWatch Alarms 集成到控制面板中，以便快速查看任何超出阈值的指标。
7. 使用 Contributor Insights：加入 CloudWatch Contributor Insights 来分析高基数字段，更清楚地了解资源的主要贡献者。
8. 设计自定义小组件：对于标准小组件无法满足的特定需求，可以考虑创建自定义小组件。自定义小部件可以从各种数据来源中提取数据，也可以以独特方式呈现数据。
9. 使用 AWS Health：AWS Health 是有关 AWS Cloud资源运行状况的权威信息来源。开箱即用 AWS Health Dashboard，或者在您自己的控制面板和工具中使用 AWS Health 数据，这样您就可以获得正确的信息来做出明智的决策。

10迭代和完善：随着应用程序的发展，请定期重新审视控制面板，确保其仍然适用。

资源

相关最佳实践：

- OPS04-BP01 确定关键绩效指标
- OPS08-BP01 分析工作负载指标
- OPS08-BP02 分析工作负载日志
- OPS08-BP03 分析工作负载跟踪数据
- OPS08-BP04 创建可操作的警报

相关文档：

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

相关视频：

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

相关示例：

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

OPS 9. 如何了解自己的运营状况？

定义、记录和分析运营指标以便了解运营事件，从而采取适当的行动。

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

从组织获取定义运营成功的目标和 KPI，并确定指标可反映这些目标和 KPI。将基线设置为参考点，并定期重新评估。制定机制，从团队收集这些指标以供评估。DevOps Research and Assessment (DORA) 指标提供了一种常用的方法来衡量软件交付 DevOps 实践的进展。

期望结果：

- 组织发布并分享运营团队的目标和 KPI。
- 您建立反映这些 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：

- 由于开发人员被抽调去执行故障排除任务，而错过部署截止日期。开发团队主张增加人手，但由于无法衡量所占用的时间，因此无法量化他们需要多少人手。
- 设置了 1 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 1 级服务台分配人手。随着呼叫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。
- 有问题的工作负载已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排除和解决故障。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，可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水平目标的情况。

在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商谈，来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、平均问题解决时间或成本效益。

根据 KPI，确定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。统计数据显示了不同类型问题所花费的时间（或这些问题的数量），其可以提供一个窗口，便于了解需要在哪些方面开展有针对性的工作。

资源

相关文档：

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

相关示例：

- Monitor the performance of your software delivery using native AWS monitoring and observability tools
- Balance deployment speed and stability with DORA metrics
- Example MLOps operational metrics in the financial services industry
- How to track your cost optimization KPIs with the 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。在运营事件期间，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。

期望结果：

- 运营主管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报告，以便显示运营统计数据，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时长。

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。各个运营团队都加倍努力调查问题。
- 由于人们都想获得新功能，导致几名人员被重新分配到工程工作中。没有提供候补人员，问题解决时间激增。没有记录这些信息，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，例如是否检测到了问题、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。

此外，控制面板和报告可以为决策者和利益相关方提供洞察，以便了解运营团队响应业务需求以及分配资源的方式。这对于确定是否有足够的资源来支持业务至关重要。

在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，并让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，显示意外事件或事件何时发生、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，并广泛告知该位置。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运营状况的报告，并将其分发给领导者和决策者，以便说明运营工作以及挑战和需求。

在团队之间分享这些指标和报告，这些指标和报告最能反映目标和 KPI，以及在推动变革方面的影响力。投入时间开展这些活动，提升运营在团队内部和团队之间的重要性。

将 AWS Health 与您自己的控制面板一起使用，或者将 AWS Health 事件集成到控制面板中，这样您的团队就可以将应用程序问题与 AWS 服务状态相关联。

资源

相关最佳实践：

- OPS09-BP01 使用指标衡量运营目标和 KPI

相关文档：

- Measure Progress
- 构建控制面板以获取操作可见性

相关示例：

- Data Operations
- How to track your cost optimization KPIs with KPI Dashboard
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations

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者通报运营状况，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、运营和维护之间的权衡。

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充分培训，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意外事件量增加的指标。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。尽管人们一直想要实现自动化，但考虑到该系统的重要性较低，自动化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

在这些手动流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，这导致随着工作负载的增加，员工疲惫不堪。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，很快就跟不上业务的节奏。如果不定期审查运营团队收集的洞察，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显而易见，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即将实施的计划，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，让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。

在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和报告数据。结合组织的长期和短期目标来审查这些报告，以便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 代理收集 Amazon EC2 实例和本地服务器的指标和日志
- 使用 Amazon CloudWatch 指标

OPS 10. 如何应对工作负载事件和运营事件？

制定和验证用于响应事件的程序，以便尽可能减少其对工作负载的干扰。

最佳实践

- OPS10-BP01 使用流程来管理事件、意外事件和问题
- OPS10-BP02 针对每个警报设置一个流程
- OPS10-BP03 根据业务影响确定运营事件的优先顺序
- OPS10-BP04 定义上报路径
- OPS10-BP05 为影响服务的事件定义客户沟通计划
- OPS10-BP06 通过控制面板传达状态信息
- OPS10-BP07 自动响应事件

OPS10-BP01 使用流程来管理事件、意外事件和问题

要想维持工作负载的运行状况和性能，对事件、意外事件和问题的高效管理能力非常关键。因此务必要认识和理解这些要素之间的不同，这样才能制定有效的响应和解决策略。针对各个方面确立并遵循明确的流程，有助于团队快速有效地应对出现的任何运营挑战。

期望结果：组织通过记录详实且集中存储的流程，高效地管理运营事件、意外事件和问题。这些流程会不断更新来反映变更，并简化处理过程，保持出色的服务可靠性和工作负载性能。

常见反模式：

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

建立此最佳实践的好处：

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

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

实施这种最佳实践意味着您正在跟踪工作负载事件。建立用于处理意外事件和问题的流程。记录、分享并经常更新这些流程。发现问题，确定问题优先级并加以解决。

了解事件、意外事件和问题

- 事件：事件是观察到的动作、事件或状态变化。事件可以是预先计划的，也可以是计划外的，可以源自工作负载内部，也可以源自工作负载外部。
- 意外事件：意外事件是需要响应的事件，例如计划外的中断或服务质量下降。意外事件表示出现了中断，需要立即采取行动才能恢复工作负载正常运行。
- 问题：问题是一起或多起意外事件的根本原因。发现和解决问题需要对意外事件进行更深入的研究，以防将来再次发生。

实施步骤

事件

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

意外事件

1. 响应意外事件：
   - 使用来自可观测性工具的洞察快速识别和响应意外事件。
   - 实施 AWS Systems Manager Ops Center 来汇总和整理运营项目及意外事件，并确定其优先级。
   - 使用 Amazon CloudWatch 和 AWS X-Ray 等服务进行更深入的分析和故障排除。

- 考虑使用 AWS Managed Services（AMS）来增强事件管理，利用其主动、预防和侦查能力。AMS 借助监控、意外事件检测和响应以及安全管理等服务来扩展运营支持。
- Enterprise Support 客户可以使用 AWS 事件检测和响应，为生产工作负载提供持续的主动监控和事件管理。

2. 创建事件管理流程：
   - 建立结构化的事件管理流程，包括明确的角色、通信协议和解决步骤。
   - 将事件管理与聊天应用程序中的 Amazon Q 开发者版等工具集成，来实现高效的响应和协调。
   - 按严重性对意外事件进行分类，并针对每个类别预先制定意外事件响应计划。
3. 学习和改进：
   - 执行意外事件后分析，了解根本原因和解决方案的有效性。
   - 根据审查结果和不断发展的做法，持续更新和改进响应计划。
   - 记录学到的经验教训，并在各个团队之间分享，从而增强运营韧性。
   - Enterprise Support 客户可以向其技术客户经理申请事件管理讲习会。这场有指导意义的讲习会可测试现有的意外事件响应计划，并帮助找出需要改进之处。

Problems（问题）

1. 确定问题：
   - 使用先前意外事件的数据来确定反复出现的模式，这些模式可能表明出现了更深层次的系统性问题。
   - 利用 AWS CloudTrail 和 Amazon CloudWatch 等工具来分析趋势并发现潜在问题。
   - 让运营、开发和业务部门等跨职能团队参与进来，从多元化的视角来审视根本原因。
2. 创建问题管理流程：
   - 制定结构化的问题管理流程，重点在于制定长期解决方案，而不是快速的权宜之计。
   - 采用根本原因分析（RCA）技术来调查和了解意外事件的根本原因。
   - 根据调查发现来更新运营策略、程序和基础设施，以防问题再次发生。
3. 持续改进：
   - 培养持续学习和改进的文化，鼓励团队主动发现和解决潜在问题。
   - 定期审查和修订问题管理流程及工具，适应不断变化的业务和技术形势。
   - 在整个组织内分享洞察和最佳实践，以便建立更具韧性、更高效的运营环境。
4. 利用 AWS 支持：
   - 使用 AWS Trusted Advisor 等 AWS 支持资源，获取主动指导和优化建议。

- Enterprise Support 客户可以在关键事件期间访问 AWS Countdown 等专业计划，以便获取支持。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

- 《AWS Security Incident Response Guide》
- AWS Incident Detection and Response
- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- Incident Management in the Age of DevOps and SRE
- PagerDuty - What is Incident Management?

相关视频：

- Top incident response tips from AWS
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 yrs of Amazon operational excellence
- AWS re:Invent 2022 - AWS Incident Detection and Response (SUP201)
- Introducing Incident Manager from AWS Systems Manager

相关示例：

- AWS Proactive Services – Incident Management 讲习会
- How to Automate Incident Response with PagerDuty and AWS Systems Manager Incident Manager

- Engage Incident Responders with the On-Call Schedules in AWS Systems Manager Incident Manager
- Improve the Visibility and Collaboration during Incident Handling in AWS Systems Manager Incident Manager
- Incident reports and service requests in AMS

相关服务：

- Amazon EventBridge

OPS10-BP02 针对每个警报设置一个流程

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

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

常见反模式：

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

建立此最佳实践的好处：

- 仅发出可操作的警报，缓解警报疲劳情况。
- 缩短了运营问题的平均解决时间（MTTR）。
- 缩短了平均调查时间（MTTI），这有助于减少 MTTR。
- 增强了大范围运营响应的能力。
- 提高了处理运营事件的一致性和可靠性。

例如，您为关键客户的 AWS Health 事件定义了一个流程，包括应用程序警报、运营问题和计划的生命周期事件（例如，在自动更新集群之前更新 Amazon EKS 版本），并且您为团队提供了主动监控、沟

通和响应这些事件的功能。这些操作有助于防止由 AWS 方更改所造成的服务中断，或在出现意外问题时更快地缓解此类中断。

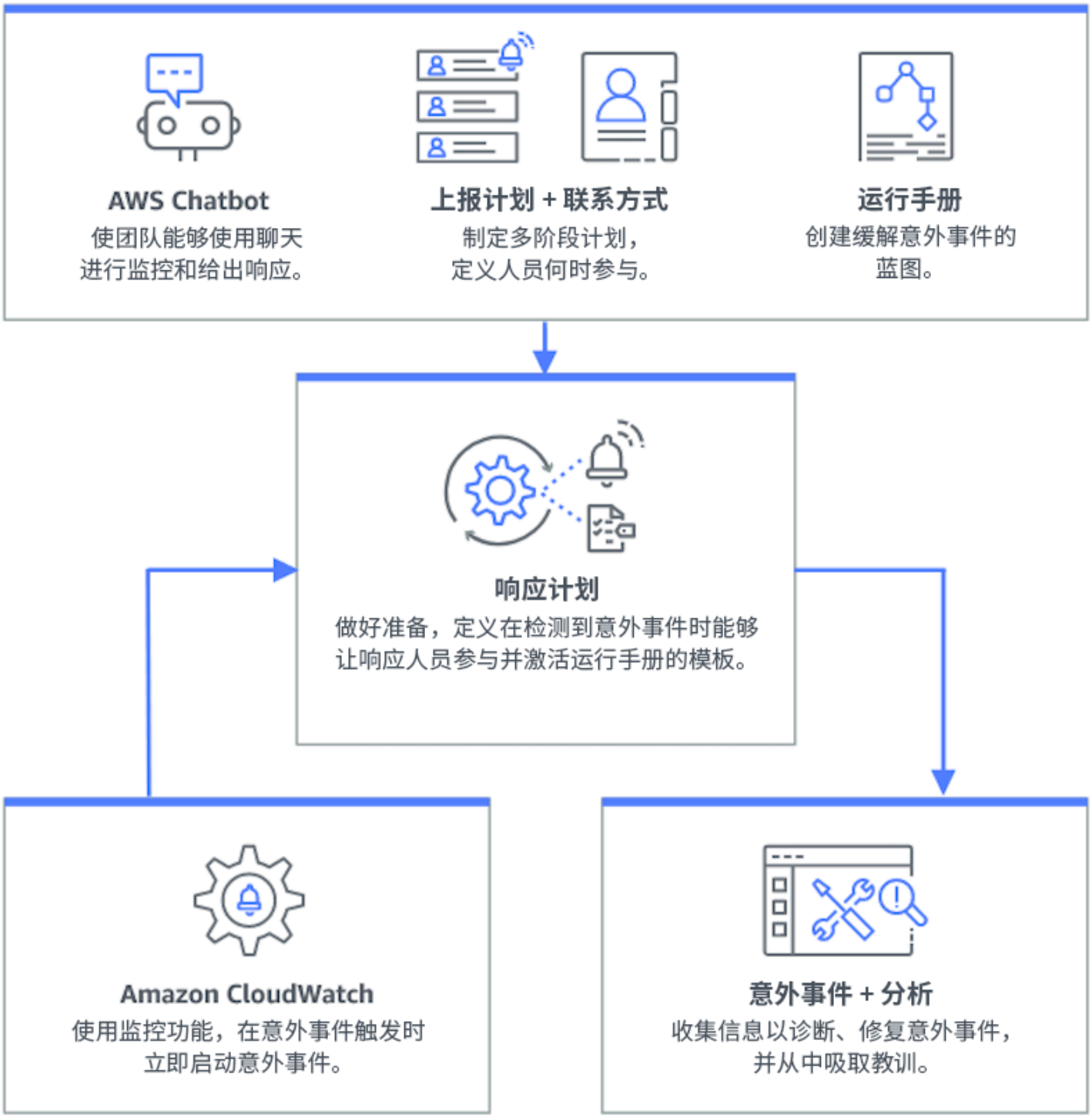
在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

实施步骤

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



1. 使用复合警报：在 CloudWatch 中创建复合警报，以便对相关警报进行分组，减少噪音并实现更有意义的响应。

2. 随时了解 AWS Health 的最新信息：AWS Health 是有关 AWS Cloud资源运行状况的权威信息来源。使用 AWS Health 可视化并获得有关任何当前服务事件和即将发生的更改（例如计划的生命周期事件）的通知，以便您可以采取措施来减轻影响。
   a. 通过 AWS 用户通知服务 创建要发送到电子邮件和聊天渠道且契合目标的 AWS Health 事件通知，并通过 Amazon EventBridge 或 AWS Health API 以编程方式与监控和警报工具集成。
   b. 通过与您可能已经通过 Amazon EventBridge 或 AWS Health API 使用的变更管理或 ITSM 工具（如 Jira 或 ServiceNow）集成，规划和跟踪需要采取行动的运行状况事件的进度。
   c. 如果您使用 AWS Organizations，请启用 organization view for AWS Health 以跨账户聚合 AWS Health 事件。
3. 将 Amazon CloudWatch 警报与 Incident Manager 集成：配置 CloudWatch 警报，以便在 AWS Systems Manager Incident Manager 中自动创建事件。
4. 将 Amazon EventBridge 与 Incident Manager 集成：创建 EventBridge 规则，以便对事件做出反应，并使用定义的响应计划创建意外事件。
5. 在 Incident Manager 中为意外事件做准备：
   - 在 Incident Manager 中为每种类型的警报制定详细的响应计划。
   - 通过 Amazon Q Developer in chat applications 建立聊天频道，连接到 Incident Manager 中的响应计划，在发生事件时，协调 Slack、Microsoft Teams 和 Amazon Chime 等各个平台之间的实时沟通。
   - 将 Systems Manager Automation 运行手册纳入 Incident Manager 中，推动对意外事件的自动响应。

资源

相关最佳实践：

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

相关文档：

- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- 使用 Amazon CloudWatch 告警
- Setting up AWS Systems Manager Incident Manager
- Preparing for incidents in Incident Manager

相关视频：

- Top incident response tips from AWS
- re:Invent 2,023 | Manage resource lifecycle events at scale with AWS Health

相关示例：

- AWS 讲习会 – AWS Systems Manager Incident Manager – Automate incident response to security events

OPS10-BP03 根据业务影响确定运营事件的优先顺序

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

期望结果；根据对业务运营和目标的潜在影响，确定运营事件响应的优先顺序。这使得应对措施既高效又有效。

常见反模式：

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

建立此最佳实践的好处：

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

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

面对多个运营事件时，基于影响力和紧急程度制定优先顺序的结构化方法至关重要。这种方法有助于作出明智的决策，将工作重心放在最需要的地方，并降低影响业务连续性的风险。

实施步骤

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

影响等级	描述
高	影响许多员工或客户，严重的财务影响，严重的声誉损害，或者造成人身伤害。
中	影响一群员工或客户，中度财务影响，或者中度声誉损害。
低	影响个别员工或客户，低财务影响，或者低声誉损害。

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

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

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

紧急程度和影响力	高	中	低
高	重大	紧急	高

紧急程度和影响力	高	中	低
中	紧急	高	正常
低	高	正常	低

4. 培训和沟通：培训响应团队，让其了解优先级矩阵以及在发生事件时遵循矩阵的重要性。与所有利益相关方沟通优先次序流程，并设定明确的期望。
5. 与意外事件响应集成：
   - 将优先级矩阵纳入意外事件响应计划和工具中。
   - 尽可能自动对事件进行分类和优先级排序，以便加快响应速度。
   - Enterprise Support 客户可以使用 AWS 事件检测和响应，为生产工作负载提供全天候的主动监控和事件管理。
6. 审查和调整：定期审查优先次序流程的有效性，并根据反馈和业务环境的变化进行调整。

资源

相关最佳实践：

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

相关文档：

- Atlassian - Understanding incident severity levels
- IT Process Map - Checklist Incident Priority

OPS10-BP04 定义上报路径

在意外事件响应协议中确立明确的上报路径，有助于及时地采取有效措施。这包括指定上报提示、详细说明上报流程，以及预先批准相关措施，以便加快决策速度并缩短平均解决时间（MTTR）。

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

常见反模式：

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

建立此最佳实践的好处：

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

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

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

实施步骤

1. 设置上报提示：设置 CloudWatch 警报，在 AWS Systems Manager Incident Manager 中创建意外事件。
2. 设置随时待命方案：在 Incident Manager 中创建与上报路径一致的随时待命方案。为随时待命人员提供必要的权限和工具，以便迅速采取行动。
3. 详细说明上报程序：
   - 确定上报意外事件的具体条件。
   - 在 Incident Manager 中创建上报计划。
   - 上报渠道应包括联系人或随时待命方案。
   - 定义团队在每个上报级别的角色和职责。
4. 预先批准缓解措施：与决策者合作，针对预期场景预先批准措施。使用与 Incident Manager 集成的 Systems Manager Automation 运行手册来加快意外事件的解决速度。
5. 指定负责人：明确指定上报路径中每个环节的内部负责人。
6. 详细说明第三方上报情况：

- 记录第三方服务水平协议（SLA），将其与内部目标保持一致。
- 针对发生意外事件时的供应商沟通情况，制定明确的协议。
- 将供应商联系人集成到事件管理工具中，以便直接访问。
- 定期开展演习，包括第三方响应场景。
- 确保详细记录了供应商上报信息，以便轻松访问。

7. 针对上报计划进行培训和演习：针对上报流程对团队进行培训，并定期进行意外事件响应演习或 GameDay 活动。Enterprise Support 客户可以申请事件管理讲习会。
8. 不断改进：定期审查上报路径的有效性。根据从意外事件事后分析中吸取的经验教训和持续反馈来更新流程。

实施计划的工作量级别：中

资源

相关最佳实践：

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

相关文档：

- AWS Systems Manager Incident Manager Escalation Plans
- Working with on-call schedules in Incident Manager
- 创建和管理运行手册
- Temporary elevated access management with AWS IAM Identity Center
- Atlassian - Escalation policies for effective incident management

OPS10-BP05 为影响服务的事件定义客户沟通计划

在发生影响服务的事件时，为了维护客户的信任和进行开诚布公地交流，有效的沟通至关重要。在发生意外事件时，明确定义的沟通计划有助于组织以快速清晰的方式，在内部和外部分享信息。

期望结果：

- 在发生影响服务的事件时，可靠的沟通计划可有效地通知客户和利益相关方。

- 开诚布公的交流可以建立信任关系，减少客户焦虑。
- 尽可能减少影响服务的事件对客户体验和业务运营的影响。

常见反模式：

- 未能充分或及时地进行沟通，导致客户困惑和不满。
- 过于技术性或含糊不清的消息传递，无法传达对用户的实际影响。
- 没有预定义的沟通策略，导致被动地传达消息，且不能确保消息的一致性。

建立此最佳实践的好处：

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

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

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

实施步骤

1. 定义角色和职责：
    - 指派一名重大意外事件经理，负责监管意外事件响应活动。
    - 指定一名沟通经理，负责协调所有内外部沟通。
    - 让支持经理参与进来，借助支持工单实现一致的沟通。
2. 确定沟通渠道：选择工作聊天工具、电子邮件、短信、社交媒体、应用程序内通知和状态页面等渠道。这些渠道应具有韧性，能够在发生影响服务的事件期间独立运行。
3. 快速、清晰地与客户开展定期沟通：
    - 针对各种服务受损场景开发模板，注重简化性和关键细节。提供有关服务受损、预期解决时间和影响的信息。
    - 使用 Amazon Pinpoint，通过推送通知、应用程序内通知、电子邮件、短信、语音消息以及自定义渠道消息，向客户发送提醒。

- 使用 Amazon Simple Notification Service（Amazon SNS），以编程方式或通过电子邮件、移动推送通知和短信提醒订阅用户。
- 通过公开分享 Amazon CloudWatch 控制面板，使用控制面板传达状态信息。
- 鼓励进行社交媒体互动：
  - 积极监控社交媒体，了解客户情绪。
  - 在社交媒体平台上发布内容，面向公众提供最新信息，并参与社区互动。
  - 编制模板，以便实现一致、清晰的社交媒体沟通。

4. 协调内部沟通：实施内部协议，使用聊天应用程序中的 Amazon Q 开发者版等工具进行团队协调和沟通。使用 CloudWatch 控制面板来传达状态信息。
5. 使用专用工具和服务来协调沟通：
   - 将 AWS Systems Manager Incident Manager 与聊天应用程序中的 Amazon Q 开发者版结合使用来设置专用的聊天频道，以便在发生事件时进行实时内部沟通和协调。
   - 发生意外事件时，使用 AWS Systems Manager Incident Manager 运行手册，通过 Amazon Pinpoint、Amazon SNS 或社交媒体平台等第三方工具，自动通知客户。
   - 将审批工作流程纳入运行手册，以便在所有外部通信渠道发送信息之前，进行审核和授权（如需要）。
6. 练习和改进：
   - 开展有关使用沟通工具和策略的培训。增强团队能力，以便在发生意外事件时及时作出决策。
   - 通过定期演习或 GameDay 活动来测试沟通计划。使用这些测试来完善消息传递流程，并评估渠道的有效性。
   - 实施反馈机制来评测发生意外事件时的沟通有效性。根据反馈和不断变化的需求，不断改进沟通计划。

实施计划的工作量级别：高

资源

相关最佳实践：

- OPS07-BP03 使用运行手册执行程序
- OPS10-BP06 通过控制面板传达状态信息
- OPS11-BP02 在意外事件发生后执行分析

相关文档：

- Atlassian - Incident communication best practices
- Atlassian - How to write a good status update
- PagerDuty - A Guide to Incident Communications

相关视频：

- Atlassian - Create your own incident communication plan: Incident templates

相关示例：

- AWS Health 控制面板

OPS10-BP06 通过控制面板传达状态信息

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

期望结果：

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

常见反模式：

- 工程师加入事件管理呼叫，需要了解状态更新才能跟得上节奏。
- 依赖人工报告进行管理，这会导致延迟和潜在的不准确性。
- 在意外事件发生时，运营团队经常被状态更新打断。

建立此最佳实践的好处：

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

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

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

实施步骤

1. 确定利益相关方的需求：确定技术团队、领导层和客户等不同受众群体的特定信息需求。
2. 选择正确的工具：选择合适的工具，例如用于系统监控的 Amazon CloudWatch 控制面板，以及用于交互式商业智能的 Amazon QuickSight。AWS Health 在 AWS Health Dashboard 中提供即用型体验，或者您可以在 Amazon EventBridge 中或通过 AWS Health API 使用运行状况事件来增强自己的控制面板。
3. 设计有效的控制面板：
   - 设计控制面板，清晰地显示相关指标和 KPI，确保这些指标易于理解且可操作。
   - 根据需要，纳入系统级和业务级视图。
   - 包括高层控制面板（用于整体概述）和底层控制面板（用于详细分析）。
   - 在控制面板中集成自动警报，以便突出显示关键问题。
   - 在控制面板中添加重要指标阈值和目标等注释，以便即时查看。
4. 集成数据来源：
   - 使用 Amazon CloudWatch 汇总和显示各种 AWS 服务的指标，并查询源自其他数据来源的指标，从而创建系统运行状况和业务指标的统一视图。
   - 使用 CloudWatch Logs Insights 等功能来查询和可视化源自不同应用程序和服务的日志数据。
   - 使用 AWS Health 事件，通过 AWS Health API 或 AWS Health events on Amazon EventBridge，随时了解 AWS 服务中运营状态和已确认的运营问题。
5. 提供自助访问：
   - 与相关利益相关方分享 CloudWatch 控制面板，以便使用控制面板分享功能进行自助信息访问。
   - 确保控制面板易于访问，并可实时提供最新信息。
6. 定期更新和完善：
   - 不断更新和完善控制面板，以便适应不断变化的业务需求，并与利益相关方的反馈保持一致。
   - 定期审查控制面板，确保其信息贴近用户需求，并有效地传达必要信息。

资源

相关最佳实践：

- OPS08-BP05 创建控制面板

相关文档：

- 构建控制面板以获取操作可见性
- Using Amazon CloudWatch Dashboards
- 使用控制面板变量创建灵活的控制面板
- 共享 CloudWatch 控制面板
- 查询源自其他数据来源的指标
- 将自定义小组件添加到 CloudWatch 控制面板

相关示例：

- One Observability 讲习会 – Dashboards

OPS10-BP07 自动响应事件

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

期望结果：

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

常见反模式：

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

建立此最佳实践的好处：

- 加快事件响应速度，减少系统停机时间。
- 通过自动化和一致的事件处理，实现可靠的运营。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

纳入自动化功能，创建高效的运营工作流程，并尽可能减少人工干预。

实施步骤

1. 发现自动化机会：确定可以自动处理的重复性任务，例如问题修复、工单信息补充、容量管理、扩展、部署和测试。
2. 发现自动化提示：
   - 使用 Amazon CloudWatch 警报操作评测并定义启动自动响应的特定条件或指标。
   - 使用 Amazon EventBridge 响应 AWS 服务、自定义工作负载和 SaaS 应用程序中的事件。
   - 考虑启动事件，例如特定日志条目、性能指标阈值或 AWS 资源中的状态变更。
3. 实现事件驱动型自动化：
   - 使用 AWS Systems Manager Automation 运行手册来简化维护、部署和修复任务。
   - 在 Incident Manager 中创建意外事件，自动收集并添加与意外事件相关的 AWS 资源的详细信息。
   - 使用适用于 AWS 的配额监控程序主动监控配额。
   - 使用 AWS Auto Scaling 自动调整容量，维持可用性和性能。
   - 使用 Amazon CodeCatalyst 实现开发管道自动化。
   - 使用综合监控进行烟雾测试或持续监控端点和 API。
4. 通过自动化功能执行风险缓解：
   - 实施自动安全响应，以便快速应对风险。
   - 使用 AWS Systems Manager State Manager 减少配置偏差。
   - 使用 AWS Config 规则 修复不合规的资源。

实施计划的工作量级别：高

资源

相关最佳实践：

- OPS08-BP04 创建可操作的警报
- OPS10-BP02 针对每个警报设置一个流程

相关文档：

- Using Systems Manager Automation runbooks with Incident Manager
- Creating incidents in Incident Manager
- AWS 服务限额
- Monitor resource usage and send notifications when approaching quotas
- AWS Auto Scaling
- What is Amazon CodeCatalyst?
- 使用 Amazon CloudWatch 告警
- 使用 Amazon CloudWatch 警报操作
- Remediating Noncompliant Resources with AWS Config 规则
- Creating metrics from log events using filters
- AWS Systems Manager State Manager

相关视频：

- Create Automation Runbooks with AWS Systems Manager
- How to automate IT Operations on AWS
- AWS Security Hub automation rules
- Start your software project fast with Amazon CodeCatalyst blueprints

相关示例：

- Amazon CodeCatalyst Tutorial: Creating a project with the Modern three-tier web application blueprint
- One Observability 讲习会

- Respond to incidents using Incident Manager

改进

问题

- OPS 11. 如何改进运营？

OPS 11. 如何改进运营？

分配专门的时间和资源用于近乎持续的渐进式改进，以便提高运营的有效性和效率。

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 审查运营指标
- 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工作负载。经常开展目标明确的工作负载审查工作。将改进机会优先纳入软件开发周期。

期望结果：

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

常见反模式：

- 自从几年前部署工作负载以来，没有对其进行过架构审查。

- 不重视改进机会。相比新功能的开发，这些机会仍在积压工作中。
- 不存在对组织最佳实践实施修改的标准。

建立此最佳实践的好处：

- 工作负载符合最新的架构最佳实践。
- 按照明确的目的来改进工作负载。
- 可以利用组织最佳实践来改进所有工作负载。
- 所获边际收益带来的影响会不断累积，从而推动效率的提升。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

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

实施步骤

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

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

资源

相关最佳实践：

- OPS11-BP02 在意外事件发生后执行分析
- OPS11-BP08 记录和分享经验教训

- OPS04 实施可观测性

相关文档：

- AWS Well-Architected Tool - Custom lenses
- AWS Well-Architected 白皮书 – 审查流程
- Customize Well-Architected Reviews using Custom Lenses and the AWS Well-Architected Tool
- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization

相关视频：

- AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization

相关示例：

- AWS Well-Architected Tool

OPS11-BP02 在意外事件发生后执行分析

审查影响客户的事件，确定这些事件的成因和预防措施。利用这些信息来制定缓解措施，限制或防止再次发生同类事件。制定程序，以便迅速有效地做出响应。根据目标受众，适当传达事件成因和纠正措施。

期望结果：

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

常见反模式：

- 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。已尝试找出应用程序服务器上出现的问题。曾怀疑可能是网络问题，但由于网络团队工作繁忙无法提供支持，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。

- 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。您认为数据丢失不重要，因为可以重新创建数据。数据丢失变得愈发频繁，并对客户造成影响。还原丢失的数据时，这也会增加运营负担。

建立此最佳实践的好处：

- 建立了预定义流程，可确定导致意外事件发生的要素、条件、操作和事件，有助于找到改进机会。
- 可以使用来自意外事件后分析的数据进行改进。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

通过流程来确定事件成因。审查所有影响客户的意外事件。设置流程来确定并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，并且还可以据此制定及时有效的响应程序。酌情传达造成意外事件的根本原因，并针对目标受众量身定制传达内容。在组织内公开分享经验教训。

实施步骤

1. 收集各种指标，例如部署更改、配置更改、意外事件开始时间、警报时间、参与时间、缓解措施开始时间和意外事件解决时间。
2. 在时间表上描述关键时间点，用于了解意外事件。
3. 提出以下问题：
    a. 能否缩短检测时间？
    b. 是否更新了可以更快地检测到事件的指标和警报？
    c. 能否缩短诊断时间？
    d. 您的响应计划或上报计划是否有更新，可以更快地与正确的响应者进行互动？
    e. 能否缩短缓解时间？
    f. 可以添加或改进哪些运行手册或行动手册步骤？
    g. 未来能否防止意外事件再次发生？
4. 创建检查清单和操作。跟踪并交付所有操作。

实施计划的工作量级别：中

资源

相关最佳实践：

- OPS11-BP01 设置持续改进流程
- OPS4 – 实施可观测性

相关文档：

- Performing a post-incident analysis in Incident Manager
- Operational Readiness Review

OPS11-BP03 实施反馈环路

反馈环路提供了可操作的洞察，可推动决策的制定。在程序和工作负载中建立反馈环路。这有助于确定问题和需要改进的领域。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，可以获取对一段时间内的运营成果和指标的审查反馈。这些回顾可在冲刺结束时、按节奏或在重大发布或事件之后进行。这种类型的反馈环路可验证在运营或工作负载方面的投入。它有助于衡量成功并验证策略。

期望结果：可以使用即时反馈和回顾性分析来推动改进。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：

- 推出了一项新功能，但无法接收客户对此新功能的反馈。
- 在投资运营改进后，无需进行回顾来验证改进。
- 收集客户反馈，但不定期审查。
- 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。

建立此最佳实践的好处：

- 可以从客户的角度逆向展开工作，以便推动新功能。
- 组织文化能够更快地对变化做出反应。
- 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。

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。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。定期使用反馈来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
   - 需要一种机制来接收客户和团队成员提供的反馈。也可以将运营活动配置为提供自动反馈。
   - 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。
   - 必须将反馈纳入软件开发过程中。
   - 在实施改进时，请跟进反馈提交者。
     - 可以使用 AWS Systems Manager OpsCenter 以 OpsItems 的形式创建和跟踪这些改进。
2. 回顾性分析
   - 在开发周期结束时、按设定的节奏或在重大发布后进行回顾。
   - 召开回顾性会议，让工作负载中涉及的利益相关方参加。
   - 在白板或电子表格上创建三列：“停止”、“开始”和“继续”。
     - 停止列针对的是希望团队停止执行的任何工作。
     - 开始列针对的是要开始付诸行动的想法。
     - 继续列针对的是要继续执行的项目。
   - 在会议室里四处走动，从利益相关方那里收集反馈。
   - 确定反馈的优先级。将操作和利益相关方分配给“开始”或“继续”项目。

- 将操作添加到软件开发过程中，并在实施改进时将状态更新传达给利益相关方。

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，还需要建立一个回顾性分析流程。

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：内部利益相关方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：意外事件后分析是发生意外事件后进行回顾性分析的重要形式。
- OPS11-BP07 审查运营指标：运营指标审查可确定趋势和需要改进的方面。

相关文档：

- 7 Pitfalls to Avoid When Building a CCOE
- Atlassian 团队行动手册 – 回顾
- Email Definitions: Feedback Loops
- Establishing Feedback Loops Based on the AWS Well-Architected Framework Review
- IBM Garage 方法 – 保持回顾
- Investopedia – The PDCS Cycle
- Tim Cochran 所著的《Maximizing Developer Effectiveness》
- Operations Readiness Reviews（ORR）白皮书 – Iteration
- ITIL CSI - Continual Service Improvement
- When Toyota met e-commerce: Lean at Amazon

相关视频：

- Building Effective Customer Feedback Loops

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其工作所需的信息。在学习型组织中，成员自由分享信息，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。制定可创建新信息、更新现有信息和归档过时信息的机制。知识管理平台最常见例子是 Wiki 之类的内容管理系统。

期望结果：

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

常见反模式：

- 没有集中式知识存储。团队成员在其本地计算机上管理自己的笔记。
- 有自托管 Wiki，但没有制定机制来管理信息，导致信息过时。
- 有人发现了缺失的信息，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：

- 由于可以自由分享信息，团队成员的能力得到了增强。
- 由于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、准确且富有实用价值。

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，需要一个中央存储库来存储知识（一个常见的例子是自托管 Wiki）。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，并让每一个人都能做出贡献。

客户示例

AnyCompany Retail 托管了一个内部 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。每个季度，跨职能团队会评估哪些页面的内容更新最少，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相关方的同意。
   a. 如果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 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。与利益相关方一起对添加的知识进行优先级排序。
4. 定期与利益相关方一起找出过时的信息并将其归档或更新。

实施计划的工作量级别：中。如果还没有内容管理系统，则可以设置自托管 Wiki 或版本控制文档存储库。

资源

相关最佳实践：

- OPS11-BP08 记录和分享经验教训 – 知识管理可促进有关经验教训的信息分享。

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，有助于根据数据和反馈环路来评估机会并进行优先级排序。探索系统和流程的改进机会，并根据具体情况相应采用自动化功能。

期望结果：

- 跟踪整个环境中的数据。
- 将事件和活动与业务成果相关联。
- 可以在环境和系统之间进行比较和对比。
- 保留部署和成果的详细活动历史记录。
- 收集数据来支持安全态势。

常见反模式：

- 您从整个环境中收集数据，但没有关联事件和活动。
- 收集所有资产的详细数据，而这导致 Amazon CloudWatch 和 AWS CloudTrail 的活动及成本增加。但是，并没有让这些数据发挥出作用。
- 在确定推动改进的因素时，没有考虑业务成果。
- 没有衡量新功能的效果。

建立此最佳实践的好处：

- 通过确定用于改进的标准，尽可能减小基于事件的动机或情感投入所带来的影响。
- 可以响应业务事件，而不仅仅是技术事件。
- 可以衡量环境来确定需要改进的方面。

在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：应该只在能够实现期望结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期望的特性和功能。
  - AWS 的新功能
- 无法接受的问题：在评估改进机会时，评估无法接受的问题、错误和漏洞。跟踪合理调整大小选项，寻找优化机会。
  - AWS 最新安全公告
  - AWS Trusted Advisor
  - Cloud Intelligence Dashboards
- 合规性要求：在分析改进机会时，评估为了保持监管和政策合规性，或获取第三方支持，所需的更新和更改。
  - AWS 合规性
  - AWS 合规性计划
  - AWS 合规性最新消息

资源

相关最佳实践：

- OPS01 组织优先事项
- OPS02 关系和所有权
- OPS04-BP01 确定关键绩效指标
- OPS08 利用工作负载可观测性
- OPS09 了解运营状况
- OPS11-BP03 实施反馈环路

相关文档：

- Amazon Athena
- QuickSight
- AWS 合规性
- AWS 合规性最新消息
- AWS 合规性计划
- AWS Glue

- AWS 最新安全公告
- AWS Trusted Advisor
- Export your log data to Amazon S3
- AWS 的新功能
- 以客户为中心的创新势在必行
- Digital Transformation: Hype or a Strategic Necessity?

相关视频

- AWS re:Invent 2023 - Improve operational efficiency and resilience with 支持 (SUP310)

OPS11-BP06 验证分析结果

与跨职能团队和业务负责人共同审查分析结果和响应措施。通过这些审查工作来建立共识、发现其他影响并确定行动方案。适当调整响应措施。

期望结果：

- 与业务负责人一起定期审查分析结果。业务负责人为新获得的洞察提供更多背景信息。
- 您审查分析结果并让技术同事提供反馈，然后在团队之间分享学到的经验教训。
- 您发布数据和分析结果，让其他技术团队和业务团队进行审查。将学到的经验教训融入到其他部门的新实践中。
- 与高层领导一起总结和审查新分析结果。高层领导使用新的分析结果来定义策略。

常见反模式：

- 您发布了新功能。此功能改变了一些客户行为。可观测性没有考虑到这些变化。您无法量化这些变化带来的益处。
- 推送新的更新，却忽略了刷新 CDN。CDN 缓存不再与最新版本兼容。衡量出错请求的百分比。所有用户在与后端服务器通信时，都报告了 HTTP 400 错误。调查客户端出现的错误时，发现是因为衡量了错误的维度，导致时间就这样白白浪费了。
- 服务水平协议规定正常运行时间为 99.9%，恢复点目标是 4 小时。服务负责人坚持认为系统应该是零停机时间。您实施了昂贵而复杂的复制解决方案，浪费了时间和金钱。

建立此最佳实践的好处：

- 与业务负责人和主题专家一起验证分析结果时，就可以建立共识并更有效地指导改进。
- 发现隐藏的问题，并在未来的决策中考虑到这些问题。
- 重心从技术成果转移到业务成果。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

- 验证分析结果：与业务负责人和主题专家沟通，确保对收集的数据价值达成共识和一致。找出其他问题、潜在影响并制定行动方案。

资源

相关最佳实践：

- OPS01-BP06 在管理益处与风险的同时评估各种权衡因素
- OPS02-BP06 预先定义或协商团队间的职责
- OPS11-BP03 实施反馈环路

相关文档：

- Designing a Cloud Center of Excellence (CCOE)

相关视频：

- Building observability to increase resiliency

OPS11-BP07 审查运营指标

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。寻找所有环境（例如，开发、测试和生产环境）中的改进机会。

期望结果：

- 经常审查影响业务的指标
- 通过可观测性功能来检测和审查异常
- 使用数据来支持实现业务成果和目标

常见反模式：

- 维护时段导致一次重要的零售促销活动中断。如果存在其他影响业务的事件，可能延迟标准维护时段，而业务部门对此并不知晓。
- 由于组织中广泛使用了过时的库，导致长时间停机。自此之后，迁移到受支持的库。组织中的其他团队尚未意识到风险的存在。
- 您没有定期审查客户 SLA 的达成情况。您目前正趋向于无法满足客户 SLA。如果无法满足客户 SLA，将会受到经济处罚。

建立此最佳实践的好处：

- 如果能够定期开会审查运营指标、事件和意外事件，就可以在团队之间达成共识。
- 团队定期开会来审查指标和意外事件，这样可以很好地针对风险采取行动并实现客户 SLA。
- 可以分享学到的经验教训，这样能提供数据，根据业务成果确定优先事项和有针对性的改进。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

- 定期与来自不同业务领域的跨团队参与者对运营指标进行回顾性分析。
- 与包括业务、开发和运营团队在内的利益相关方交流，共同验证通过即时反馈和回顾性分析得到的调查发现，并分享经验教训。
- 根据他们的洞察来确定改进机会和可能的行动方案。

资源

相关最佳实践：

- OPS08-BP05 创建控制面板
- OPS09-BP03 审查运营指标并确定改进优先顺序
- OPS10-BP01 使用流程来管理事件、意外事件和问题

相关文档：

- Amazon CloudWatch
- Amazon CloudWatch 指标和维度参考

- 发布自定义指标
- 使用 Amazon CloudWatch 指标
- Dashboards and visualizations with CloudWatch

OPS11-BP08 记录和分享经验教训

记录和分享在运营活动中获得的经验教训，以便在内部和不同团队中运用。应分享团队学到的经验教训，从而增加整个组织获得的效益。分享信息和资源来防止可避免的错误，简化开发工作，并将重心放在交付所需的功能上。

使用 AWS Identity and Access Management（IAM）定义权限，允许对要在账户内和账户之间分享的资源进行受控访问。

期望结果：

- 使用版本受控的存储库来分享应用程序库、脚本程序、程序文档和其他系统文档。
- 将基础设施标准作为版本受控的 AWS CloudFormation 模板分享。
- 查看各团队学到的经验教训。

常见反模式：

- 由于组织中广泛使用有错误的库，导致长时间的停机。自此之后，迁移到受支持的库。组织中的其他团队尚未意识到风险的存在。没有人记录和分享使用这个库的体验，也没人意识到风险。
- 您已经确定内部分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，更新了对服务的调用。组织中的其他团队尚未意识到风险的存在。
- 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。不知道其他团队是否可以利用这种技术。

建立此最佳实践的好处：分享经验教训，从而支持改进并最大限度地发挥经验的益处。

在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

- 记录和分享经验教训：设置程序来记录在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：设置程序来在不同团队中分享经验教训和相关项目。例如，通过方便访问的 Wiki，分享更新后的程序、指南、管理机制和最佳实践。通过公共存储库分享脚本、代码和库。
  - 利用 AWS re:Post Private 作为知识服务，来简化组织内的协作和知识共享。

资源

相关最佳实践：

- OPS02-BP06 预先定义或协商团队间的职责
- OPS05-BP01 使用版本控制
- OPS05-BP06 共享设计标准
- OPS11-BP03 实施反馈环路
- OPS11-BP07 审查运营指标

相关文档：

- Increase collaboration and securely share cloud knowledge with AWS re:Post Private
- Reduce project delays with a docs-as-code solution

相关视频：

- AWS re:Invent 2023 - Collaborate within your company and with AWS using AWS re:Post Private
- 支持s You | Exploring the Incident Management Tabletop Exercise

OPS11-BP09 分配时间进行改进

流程中专用的时间和资源可以实现持续渐进式改进。

期望结果：

- 创建了临时的环境副本，这可以降低试验和测试的风险、工作量及成本。
- 这些重复的环境可用于测试根据分析得出的结论，对计划的改进进行试验、开发和测试。
- 开展 GameDay 活动，并使用故障注入服务（FIS），提供团队在类似于生产环境的条件下开展实验所需的控制措施和防护机制。

常见反模式：

- 应用程序服务器中存在一个已知性能问题。将其添加到积压工作中，列在所有计划功能实施之后。如果一直保持这种添加计划功能的速度，性能问题将永远无法得到解决。
- 为了支持持续改进，批准管理员和开发人员利用他们所有的额外时间来选择和实施改进。没有完成任何改进。
- 运营验收完成后，再也没有测试过运营实践。

建立此最佳实践的好处：通过在流程中投入专门的时间和资源，可以实现持续渐进式改进。

在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

- 分配时间进行改进：在流程中投入专门的时间和资源，用于实现持续渐进式改进。
- 实施更改以便改进，并评估结果来确定是否成功。
- 如果结果不符合目标，并且仍然需要改进，则寻求其他行动方案。
- 通过 GameDay 活动来模拟生产工作负载，并使用从这些模拟中学到的经验教训进行改进。

资源

相关最佳实践：

- OPS05-BP08 使用多个环境

相关视频：

- AWS re:Invent 2023 - Improve application resilience with AWS Fault Injection Service

安全性

安全性支柱包括保护数据、系统和资产以利用云技术来改善安全性的能力。有关具体实施的说明性指导，请参阅《安全性支柱白皮书》。

最佳实践领域

- 安全基础知识
- 身份和访问管理
- 检测

- 基础设施保护
- 数据保护
- 事件响应
- 应用程序安全性

安全基础知识

问题

- SEC 1. 如何安全地操作工作负载？

SEC 1. 如何安全地操作工作负载？

为了安全地操作您的工作负载，您必须将纲领性最佳实践应用于每个安全领域。将您在组织和工作负载级别的卓越运营中定义的要求和流程应用于所有领域。及时了解 AWS 和行业建议及威胁情报可以帮助您发展威胁模型和控制目标。实现安全流程、测试和验证的自动化可扩展您的安全运营。

最佳实践

- SEC01-BP01 使用账户分隔工作负载
- SEC01-BP02 保护账户根用户和属性
- SEC01-BP03 识别并验证控制目标
- SEC01-BP04 随时了解安全威胁和建议
- SEC01-BP05 缩小安全管理范围
- SEC01-BP06 自动部署标准安全控制措施
- SEC01-BP07 使用威胁模型识别威胁并确定缓解措施的优先级
- SEC01-BP08 定期评估并实施新的安全服务和功能

SEC01-BP01 使用账户分隔工作负载

通过采取多账户策略，在环境（如生产、开发和测试）和工作负载之间建立共同的防护机制和隔离措施。强烈建议在账户层面进行分离管理，这样可为安全性、账单和访问提供强大的隔离边界。

期望结果：形成一种账户结构，可将云运维、无关工作负载和环境隔离到单独的账户中，从而提高整个云基础设施的安全性。

常见反模式：

- 将多个相互毫无关联，具有不同数据敏感度级别的工作负载放入同一账户中。
- 组织单位（OU）结构界定不清。

建立此最佳实践的好处：

- 即使不该访问的工作负载无意中被访问了，影响范围也会缩小。
- 能够对访问 AWS 服务、资源和区域进行集中治理。
- 可集中管理策略和安全服务，维护云基础设施的安全性。
- 实现账户创建和维护流程自动化。
- 集中审核基础设施状况，从而满足法规遵从性和监管要求。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

AWS 账户提供安全隔离边界，使不同敏感度的工作负载或资源相互分离。AWS 提供相应工具，以多账户策略来大规模管理云工作负载，从而利用此隔离边界。如要获得有关 AWS 多账户策略的概念、模式和实施的指导，请参阅《Organizing Your AWS Environment Using Multiple Accounts》白皮书。

如果需要集中管理多个 AWS 账户，账户应基于组织单位（OU）层建立层次结构。然后可以建立安全控制机制，并将其应用于 OU 和成员账户，从而为组织内的成员账户建立一致的预防性控制机制。安全控制机制是层层继承的，使您能够筛选位于 OU 层次结构较低层次的成员账户的可用权限。优秀的架构设计将能够利用这种层层继承的特性，减少设置安全策略，降低复杂性，并使每个成员账户的安全控制效果达到预期。

采用 AWS Organizations 和 AWS Control Tower 这两种服务，可在您的 AWS 环境中实施和管理多账户结构。AWS Organizations 使得您能够将账户建立成由一个或多个 OU 层定义的层次结构形式，每个 OU 均可包含若干成员账户。服务控制策略（SCP）使组织管理员能够对成员账户建立精细的预防性控制机制，而 AWS Config 可用于建立对成员账户的主动式和检测性控制。许多 AWS 服务与 AWS Organizations 集成，可提供委派型管理控制，并在组织内的所有成员账户中执行服务特定的任务。

AWS Control Tower 位于 AWS Organizations 之上，为具有登录区的多账户 AWS 环境提供了一键式最佳实践设置。登录区是由 Control Tower 建立的多账户环境的入口处。与 AWS Organizations 相比，采用 Control Tower 具有若干好处。可以改进账户治理状况的三种好处为：

- 将强制安全防护机制集成于系统中，可自动应用于准入组织的账户。
- 有多种防护机制可供选择，还能开启或关闭给定 OU 组的防护机制。

- AWS Control Tower Account Factory 可在组织内部自动部署账户，设置好预先批准的基准和配置选项。

实施步骤

1. 设计组织单位结构：设计良好的组织单位结构减少了创建和维护服务控制策略及其他安全控制机制所需的管理负担。组织单位结构应与业务需求、数据敏感度和工作负载结构看齐。
2. 为多账户环境创建登录区：登录区提供了一致的安全性和基础设施基础，让组织可以从中快速开发、启动和部署工作负载。您可以使用定制的登录区或 AWS Control Tower 来编排环境。
3. 建立防护机制：通过登录区为环境实施一致的安全防护机制。AWS Control Tower 提供了可部署的必选和可选控制机制的列表。实施 Control Tower 时会自动部署必选控制机制。查看高度推荐和可选控制机制的列表，并实施适合您需求的控制机制。
4. 限制访问新添加的区域：对于新的 AWS 区域，诸如用户和角色之类的 IAM 资源将仅传播到您指定的区域。可以在使用 Control Tower 时通过控制台执行此操作，也可以通过调整 AWS Organizations 中的 IAM 权限策略执行此操作。
5. 考虑使用 AWS CloudFormation StackSets：StackSets 可帮助您通过已批准的模板将资源（包括 IAM 策略、角色和组）部署到不同的 AWS 账户和区域中。

资源

相关最佳实践：

- SEC02-BP04 依赖集中式身份提供程序

相关文档：

- AWS Control Tower
- AWS 安全审计指南
- IAM 最佳实践
- Use CloudFormation StackSets to provision resources across multiple AWS 账户 and regions
- Organizations 常见问题
- AWS Organizations 术语和概念
- Best Practices for Service Control Policies in an AWS Organizations Multi-Account Environment
- AWS Account Management Reference Guide

- 使用多个账户整理您的 AWS 环境

相关视频：

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- Building and Governing Multiple Accounts using AWS Control Tower
- Enable Control Tower for Existing Organizations

SEC01-BP02 保护账户根用户和属性

根用户是 AWS 账户中权限最高的用户，对账户内的所有资源具有完全管理访问权限，在某些情况下不受安全策略的约束。停用对根用户的编程访问，为根用户建立适当的控制机制，并避免日常使用根用户，这样有助于降低无意中暴露根凭证以及随后破坏云环境的风险。

期望结果：保护根用户有助于减少因滥用根用户凭证而导致意外或故意损坏的可能性。建立检测性控制机制也可以在有人使用根用户执行操作时向适当人员发出警报。

常见反模式：

- 使用根用户执行各种任务，而非仅在必要时使用根用户凭证。
- 忽略定期测试应急计划，不验证关键基础设施、流程和人员在紧急情况下的运作情况。
- 只考虑典型的账户登录流程，而没有考虑或测试替代的账户恢复方法。
- 因为 DNS、电子邮件服务器和电话提供商要用于账户恢复流程，就不将其作为关键安全边界的一部分进行处理。

建立此最佳实践的好处：保护对根用户的访问可以建立信心，让账户中的操作受到控制和审核。

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

AWS 提供许多有助于保护账户安全的工具。但由于其中一些措施默认情况下未启用，因此您必须采取直接行动来实施这些措施。请将这些建议视为确保 AWS 账户安全的基本步骤。实施这些步骤时，务必建立一个可持续评测和监控安全控制机制的过程，这非常重要。

当您首次创建 AWS 账户时，最初使用的是一个对账户中所有 AWS 服务和资源有完全访问权限的身份。此身份称作 AWS 账户根用户。您可以使用在创建账户所用的电子邮件地址和密码以根用户身份登

录。由于授予 AWS 根用户的访问权限较高，您必须仅将 AWS 根用户用于执行特别需要它的任务。必须严格保护根用户登录凭证，并且应始终为 AWS 账户根用户使用多重身份验证（MFA）。

除了使用用户名、密码和多重身份验证（MFA）设备登录根用户的常规身份验证流程外，还可以使用账户恢复流程登录您的 AWS 账户 根用户，该用户可以访问与您的账户关联的电子邮件地址和电话号码。因此，保护发送恢复电子邮件的根用户电子邮件账户和保护与该账户关联的电话号码同样重要。还应考虑潜在的循环依赖关系，其中与根用户关联的电子邮件地址托管在同一 AWS 账户的电子邮件服务器或域名服务（DNS）资源上。

使用 AWS Organizations 时，有多个 AWS 账户（每个均有一个根用户）。将一个账户指定为管理账户，然后可以在管理账户下面添加几层成员账户。优先保护管理账户的根用户，然后解决成员账户根用户问题。保护管理账户根用户的策略可能与保护成员账户根用户的策略不同，您可以对成员账户根用户建立预防性安全控制机制。

实施步骤

建议使用以下实施步骤为根用户建立控制机制。在适用情况下，建议与 CIS AWS Foundations Benchmark 版本 1.4.0 交叉引用。除了这些步骤外，请参阅 AWS 最佳实践指导来确保 AWS 账户和资源安全。

预防性控制机制

1. 为账户设置准确的联系信息。
   a. 该信息用于丢失的密码恢复流程、丢失的 MFA 设备账户恢复流程，以及与您的团队进行关键的安全相关通信。
   b. 使用企业域托管的电子邮件地址（最好是通讯组列表）作为根用户的电子邮件地址。使用通讯组列表而不是个人的电子邮件账户可提供额外的冗余和连续性，以便在很长一段时间内访问根账户。
   c. 联系信息上所列的电话号码应该是为此目的而设置的专用安全电话的号码。电话号码不应列出或与任何人共享。
2. 不要为根用户创建访问密钥。如果存在访问密钥，请将其删除（CIS 1.4）。
   a. 消除根用户的任何长期编程凭证（访问密钥和私有密钥）。
   b. 如果已存在根用户访问密钥，您应将使用这些密钥的进程转换为使用 AWS Identity and Access Management（IAM）角色的临时访问密钥，然后删除根用户访问密钥。
3. 确定是否需要为根用户存储凭证。
   a. 如果您使用 AWS Organizations 创建新的成员账户，则新成员账户上根用户的初始密码将设置为一个不向您公开的随机值。如果需要，请考虑使用 AWS 组织管理账户的密码重置流程来访问成员账户。

b. 对于独立 AWS 账户或管理 AWS 组织账户，请考虑为根用户创建并安全地存储凭证。为根用户启用 MFA。

4. 在 AWS 多账户环境中，为成员账户根用户使用预防性控制机制。
   a. 考虑为成员账户启用不允许为根用户创建根访问密钥预防性防护机制。
   b. 考虑为成员账户启用不允许以根用户身份执行操作预防性防护机制。
5. 如果需要根用户凭证，请执行以下操作：
   a. 使用复杂密码。
   b. 为根用户启用多重身份验证（MFA），特别是 AWS Organizations 管理（付款人）账户（CIS 1.5）。
   c. 考虑使用硬件 MFA 设备来提高韧性和安全性，因为一次性设备可以减少包含 MFA 代码的设备被重复用于其他用途的可能性。验证是否定期更换由电池供电的硬件 MFA 设备。（CIS 1.6）
      - 要为根用户配置 MFA，请遵循创建虚拟 MFA 或硬件 MFA 设备的说明。
   d. 考虑注册多个 MFA 设备用于备份。每个账户最多允许 8 个 MFA 设备。
      - 请注意，为根用户注册多个 MFA 设备将自动禁用在 MFA 设备丢失的情况下恢复账户的流程。
   e. 安全地存储密码，如果以电子方式存储密码，则考虑循环依赖关系。不要以需要访问同一 AWS 账户才能获得密码的方式存储密码。
6. 可选：考虑为根用户制定定期密码轮换计划。
   - 凭证管理最佳实践取决于您的监管和政策要求。受 MFA 保护的根用户并不依赖密码作为单重身份验证。
   - 定期更改根用户密码可降低无意中暴露的密码被滥用的风险。

侦测性控制

- 创建警报来检测根凭证的使用情况（CIS 1.7）。启用 Amazon GuardDuty 将通过 RootCredentialUsage 调查发现对根用户 API 凭证的使用进行监控和发出警报。
- 评估并实施 AWS Well-Architected 安全性支柱合规包 AWS Config 中包含的检测性控制机制，或者如果使用 AWS Control Tower，则评估并实施 Control Tower 内强烈建议的控制机制。

运营指导

- 确定组织中应该有权访问根用户凭证的人员。
  - 采用双人规则，以便不会出现一个人就能够访问所有必要凭证和 MFA 来获得根用户访问权限的情况。

  - 验证组织（而不是个人）对与账户关联的电话号码和电子邮件别名（用于密码重置和 MFA 重置流程）保持控制。
- 仅在例外情况下使用根用户（CIS 1.7）。
  - 不得使用 AWS 根用户执行日常任务，即使是管理任务也不可以。仅以根用户身份登录，以执行需要根用户的 AWS 任务。所有其他操作都应由代入适当角色的其他用户执行。
- 定期检查对根用户的访问是否正常，以便在出现需要使用根用户凭证的紧急情况之前对过程进行测试。
- 定期检查与账户关联的电子邮件地址以及备用联系人下列出的电子邮件地址是否有效。监控这些电子邮件收件箱，查看您可能从 <abuse@amazon.com> 中收到的安全通知。还要确保与该账户相关的任何电话号码都有效。
- 准备事件响应程序，应对根账户滥用情况。请参阅《AWS Security Incident Response Guide》以及《安全性支柱》白皮书“事件响应”部分中的最佳实践，了解有关为 AWS 账户构建事件响应策略的更多信息。

资源

相关最佳实践：

- SEC01-BP01 使用账户分隔工作负载
- SEC02-BP01 使用强大的登录机制
- SEC03-BP02 授予最低访问权限
- SEC03-BP03 建立紧急访问流程
- SEC10-BP05 预置访问权限

相关文档：

- AWS Control Tower
- AWS 安全审计指南
- IAM 最佳实践
- Amazon GuardDuty – root credential usage alert
- 通过 CloudTrail 监控根凭证使用情况的分步指导
- 获准与 AWS 一起使用的 MFA 令牌
- Implementing break glass access on AWS

- Top 10 security items to improve in your AWS 账户
- 发现我的 AWS 账户中存在未经授权的活动时该怎么办？

相关视频：

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- Limiting use of AWS root credentials from AWS re:inforce 2022 – Security best practices with AWS IAM

SEC01-BP03 识别并验证控制目标

根据合规性要求以及从威胁模型中发现的风险，获得并验证需要应用于工作负载的控制目标和控制措施。持续验证控制目标和控制措施可帮助您衡量风险缓解措施的有效性。

期望结果：针对业务明确定义了安全控制目标，并且这些目标符合合规性要求。通过自动化方法以及策略来实施和强制执行控制措施，并持续评估这些措施在达成目标方面的有效性。收集某个时间点以及一段时间内的有效性证据，并能够随时报告给审核人员。

常见反模式：

- 没有充分了解用于确保业务安全性的监管要求、市场期望和行业标准
- 网络安全框架和控制目标与业务要求不一致
- 虽然实施了控制措施，但没有与控制目标保持高度一致，也难于衡量
- 不使用自动化方法来报告控制措施的有效性

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

根据安全控制目标，您可以利用许多常见的网络安全框架来奠定基础。根据业务考虑监管要求、市场期望和行业标准，确定哪些框架能够很好地满足需求。这些框架的例子包括 AICPA SOC 2、HITRUST、PCI-DSS、ISO 27001 和 NIST SP 800-53。

对于所确定的控制目标，了解所使用的 AWS 服务如何帮助实现这些目标。使用 AWS Artifact 来查找与目标框架相符的文档和报告，这些文档和报告介绍了 AWS 承担的责任范围，并且提供了针对您负责的其余责任范围的相关指导。如需进一步了解与各种框架控制声明对应的服务特定指导，请参阅《AWS Customer Compliance Guides》。

在定义用于实现目标的控制措施时，请使用预防性控制措施来规范执行方法，并使用检测性控制措施来自动执行缓解方法。在您的 AWS Organizations 中，使用服务控制策略（SCP）来协助防范不合规的资源配置和操作。在 AWS Config 中实施规则，用于监控并报告不合规的资源，然后在确信规则的行为正确无误时，将规则切换为强制执行模式。要部署与网络安全框架相一致的预定义托管规则集，请优先评估使用 AWS Security Hub 标准。AWS 基础服务最佳实践（FSBP，Foundational Service Best Practice）标准和 CIS AWS 基础基准可作为很好的起点，用于制定与多种标准框架所共有的多个目标相一致的控制措施。如果 Security Hub 实质上没有所需的控制检测措施，则可以使用 AWS Config 合规包予以补充。

使用 AWS Global Security and Compliance Acceleration（GSCA）团队推荐的 APN 合作伙伴服务包，可以根据需要，从安全顾问、咨询机构、证据收集和报告系统、审核人员以及其他补充性服务那里获取协助。

实施步骤

1. 评估常见的网络安全框架，让控制目标与所选框架保持一致。
2. 使用 AWS Artifact 获取所选框架的相关指导和责任文档。了解在责任共担模式下，合规性的责任有哪些归属于 AWS，哪些由您承担。
3. 使用 SCP、资源策略、角色信任策略和其他防护机制，防止出现不合规的资源配置和操作。
4. 评估与您的控制目标相一致的 Security Hub 标准和 AWS Config 合规包的部署。

资源

相关最佳实践：

- SEC03-BP01 定义访问要求
- SEC04-BP01 配置服务和应用程序日志记录
- SEC07-BP01 了解数据分类方案
- OPS01-BP03 评估治理要求
- OPS01-BP04 评估合规性要求
- PERF01-BP05 使用策略和参考架构
- COST02-BP01 根据组织的要求制定各种策略

相关文档：

- AWS Customer Compliance Guides

相关工具：

- AWS Artifact

SEC01-BP04 随时了解安全威胁和建议

关注行业威胁情报出版物和数据源来获取行业动态，及时了解最新的威胁和缓解措施。评估根据最新威胁数据自动进行更新的托管服务产品。

期望结果：在行业出版物发布最新的威胁和建议更新时及时了解情况。 您可以使用自动化功能来检测潜在的漏洞和暴露情况，以及识别新的威胁。您对这些威胁采取了缓解措施。 您采用 AWS 服务来自动更新最新的威胁情报。

常见反模式：

- 没有可靠且可重复的机制来随时了解最新的威胁情报。
- 手动维护技术产品组合、工作负载和依赖项清单，这些清单需要人工审查来发现潜在的漏洞和暴露情况。
- 没有采取机制来更新工作负载和依赖项，未获得可提供已知威胁缓解措施的最新版本。

建立此最佳实践的好处：使用威胁情报来源来了解最新信息，可以降低错过可能影响业务的重要威胁形势变化的风险。 与手动替代方案相比，采取自动化功能来扫描、检测和修复工作负载及其依赖项中存在潜在的漏洞或暴露情况，有助于您快速且可预测地降低风险。 这样就可以控制与漏洞缓解相关的时间和成本。

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

阅读可信的威胁情报出版物，随时掌握威胁形势。 有关已知的对抗策略、技巧和程序（TTP，Tactics, Techniques, and Procedures）的文档，请参阅 MITRE ATT&CK 知识库。查看 MITRE 的通用漏洞披露（CVE，Common Vulnerabilities and Exposures）列表，随时了解您依赖的产品中的已知漏洞。通过开放式全球应用程序安全项目（OWASP，Open Worldwide Application Security Project）的热门 OWASP Top 10 项目，了解 Web 应用程序面临的严重风险。

通过 CVE 的 AWS 安全公告，及时了解 AWS 安全事件和建议的修复措施。

为了减少保持最新状态所需的总体工作量和开销，您可以考虑使用 AWS 服务，这样就能随着时间的推移自动整合新威胁情报。 例如，Amazon GuardDuty 会及时了解行业威胁情报，从而检测账户中的异常行为和威胁特征。 Amazon Inspector 自动让其用于持续扫描功能的 CVE 数据库保持最新状

态。 AWS WAF 和 AWS Shield Advanced 均提供了托管规则组，这些规则组会在新威胁出现时自动更新。

查看用于自动化实例集管理和修补的 Well-Architected 卓越运营支柱。

实施步骤

- 订阅与业务和行业相关的威胁情报出版物，了解最新动态。订阅 AWS 安全公告。
- 考虑采用自动整合新威胁情报的服务，例如 Amazon GuardDuty 和 Amazon Inspector。
- 部署符合 Well-Architected 卓越运营支柱最佳实践的实例集管理和修补策略。

资源

相关最佳实践：

- SEC01-BP07 使用威胁模型识别威胁并确定缓解措施的优先级
- OPS01-BP05 评估威胁形势
- OPS11-BP01 设置持续改进流程

SEC01-BP05 缩小安全管理范围

确定是否可以使用 AWS 服务，将某些控制措施的管理工作转移给 AWS（托管服务），从而缩小安全管理范围。这些服务有助于减少安全维护任务，例如基础设施预置、软件设置、修补或备份。

期望结果：在为工作负载选择 AWS 服务时考虑到安全管理工作的范围。在应该考虑的其他 Well-Architected 注意事项之外，将管理开销和维护任务的成本（总拥有成本，简称 TCO）与您所选择服务的成本进行权衡。在控制措施评估和验证流程中，可以结合考虑 AWS 的控制和合规性文档。

常见反模式：

- 在部署工作负载时，未充分了解所选服务的责任共担模式。
- 在虚拟机上托管数据库和其他技术，但没有评估具备相同功能的托管服务。
- 在与托管服务方案对比时，虚拟机上托管技术的总拥有成本中没有包括安全管理任务。

建立此最佳实践的好处：使用托管服务可以减轻管理运营安全控制措施的整体负担，从而降低您的安全风险和总拥有成本。原本会用在某些安全任务上的时间，可以重新投入到能够为业务创造更多价值的任务上。托管服务还可以将一些控制要求转移给 AWS，从而减少您为满足合规性要求的工作范围。

在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

您可以通过多种方式将工作负载的组件集成到 AWS 上。如果您在 Amazon EC2 实例上安装和运行各种技术服务，那么在总体安全责任中，通常需要承担更大的份额。为了减轻运营某些控制措施的负担，请找出可以减少您在责任共担模式中所承担责任范围的 AWS 托管服务，并了解如何在现有架构中使用这些服务。例如，使用 Amazon Relational Database Service（Amazon RDS）部署数据库，使用 Amazon Elastic Kubernetes Service（Amazon EKS）或 Amazon Elastic Container Service（Amazon ECS）编排容器，或者使用无服务器方案。在构建新应用程序时，请仔细考虑哪些服务有助于减少实施和管理安全控制措施的时间及成本。

在选择服务时，合规性要求也可能是需要考虑的因素之一。托管服务可以将一些合规性要求转移给 AWS。请与合规团队讨论，了解他们对审核您所运营和管理的服务各个方面的满意程度，以及接受相关 AWS 审核报告中控制声明的满意程度。您可以将AWS Artifact 中的审核构件提供给审核人员或监管机构，作为 AWS 安全控制措施的证据。您还可以使用一些 AWS 审核构件提供的责任指导，并结合 AWS Customer Compliance Guides 来设计架构。这些指导可以让您了解到，为了支持系统的具体应用场景，您还应落实的其他安全控制措施。

使用托管服务时，您需要熟悉将这些服务的资源更新到新版本的过程（例如，更新 Amazon RDS 管理的数据库版本，或者更新 AWS Lambda 函数的编程语言运行时）。尽管托管服务可能会为您执行此操作，但配置更新时间以及了解这些更新会对您的运营产生何种影响，仍然是您的责任。您可以利用 AWS Health 等工具在整个环境中跟踪和管理这些更新。

实施步骤

1. 评估工作负载中可以用托管服务取代的组件。
   a. 如果您将工作负载迁移到 AWS，则在评测是要重新托管、重构、更换平台、重新构建还是更换工作负载时，请考虑减少的管理工作（时间和开支）和降低的风险。从长远来看，在迁移开始时进行额外的投入，有时可以节省大量资金。
2. 请考虑实施 Amazon RDS 等托管服务，而不是安装和管理自己的技术部署。
3. 使用 AWS Artifact 中的责任指导来帮助确定应针对工作负载采取的安全控制措施。
4. 记录所使用资源的清单，及时了解新的服务和方法，以便发现减少责任范围的新机会。

资源

相关最佳实践：

- PERF02-BP01 为工作负载选择最佳计算方案
- PERF03-BP01 使用最能满足数据访问和存储要求的专用数据存储

- SUS05-BP03 使用托管服务

相关文档：

- Planned lifecycle events for AWS Health

相关工具：

- AWS Health
- AWS Artifact
- AWS Customer Compliance Guides

相关视频：

- How do I migrate to an Amazon RDS or Aurora MySQL DB instance using AWS DMS?
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health

SEC01-BP06 自动部署标准安全控制措施

在开发和部署 AWS 环境中的标准安全控制措施时，应用现代化 DevOps 实践。 使用基础设施即代码（IaC）模板定义和配置标准安全控制措施，收集版本控制系统中的更改，测试作为 CI/CD 管道一部分的更改，并自动将更改部署到您的 AWS 环境。

期望结果：使用 IaC 模板收集标准化的安全控制措施，并将其提交给版本控制系统。 在检测到变化的地方部署了 CI/CD 管道，并自动测试和部署 AWS 环境。 在继续部署之前，采取了防护机制来检查模板中的错误配置并发出警报。 工作负载部署到采用标准控制措施的环境中。 团队具有访问权限，可以通过自助服务机制部署经批准的服务配置。 制定了安全的备份和恢复策略，用于控制配置、脚本和相关数据。

常见反模式：

- 通过 Web 控制台或命令行界面手动更改标准安全控制措施。
- 依靠各个工作负载团队来手动实施中心团队定义的控制措施。
- 依靠中心安全团队，根据工作负载团队的要求来部署工作负载级别的控制措施。
- 允许相同的个人或团队开发、测试和部署安全控制措施自动化脚本，而没有采取适当的职责分离或制衡措施。

建立此最佳实践的好处：使用模板来定义标准安全控制措施，这样您就可以通过版本控制系统来跟踪和比较随时间发生的变化。 使用自动化功能来测试和部署更改，这样可以实现标准化程序及可预测性，增加成功部署的可能性，减少重复的手动任务。 为工作负载团队提供了自助服务机制来部署经批准的服务和配置，可减少配置错误和滥用的风险。这样还可以让团队在开发过程的早期融入控制措施。

在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

按照 SEC01-BP01 使用账户分隔工作负载中描述的做法，您最终将多个 AWS 账户用于您通过 AWS Organizations 管理的不同环境。 虽然这些环境和工作负载可能会需要不同的安全控制措施，不过您可以对整个企业内的一些安全控制措施进行标准化。 这样的例子包括集成集中式身份提供程序、定义网络和防火墙，以及配置用于存储和分析日志的标准位置。 就像使用基础设施即代码（IaC）将同样严格的应用程序代码开发要求应用于基础设施预置一样，您也可以使用 IaC 来定义和部署标准安全控制措施。

尽可能以声明式方式（例如在 AWS CloudFormation 中）定义安全控制措施，并将这些安全控制措施存储在源代码控制系统中。 使用 DevOps 实践来自动部署控制措施，从而获得可预测性更强的发布，使用 AWS CloudFormation Guard 等工具自动进行测试，并检测已部署的控制措施与所需配置之间的偏差。 您可以使用 AWS CodePipeline、AWS CodeBuild 和 AWS CodeDeploy 等服务来构造 CI/CD 管道。请参考使用多个账户组织 AWS 环境中的指导，在每个服务自己的、独立于其他部署管道的账户中，配置这些服务。

您还可以定义模板来实现标准化的 AWS 账户、服务和配置的定义及部署。 利用这种技术，中心安全团队可以管理这些定义，并通过自助服务方法将这些定义提供给工作负载团队。 为此，您可以采取的一种方法是使用 Service Catalog，将模板作为产品发布，供工作负载团队整合到自己的管道部署中。 如果使用的是 AWS Control Tower，则可以使用一些模板和控制措施作为起点。 Control Tower 还提供 Account Factory 功能，让工作负载团队可以使用您定义的标准创建新的 AWS 账户。 在工作负载团队确定了需要新账户时，此功能可以避免需要依赖中心团队来审批和创建新账户。 您可能需要通过这些账户，根据工作负载提供的功能、所处理数据的敏感性或其行为等原因，来隔离不同的工作负载组件。

实施步骤

1. 确定如何在版本控制系统中存储和维护模板。
2. 创建 CI/CD 管道来测试和部署模板。 定义测试方法，用于检查配置是否有误，以及模板是否符合公司标准。
3. 构建标准化模板目录，供工作负载团队根据您的要求部署 AWS 账户和服务。
4. 为控制配置、脚本和相关数据实施安全的备份和恢复策略。

资源

相关最佳实践：

- OPS05-BP01 使用版本控制
- OPS05-BP04 使用构建和部署管理系统
- REL08-BP05 使用自动化功能部署更改
- SUS06-BP01 采用可以快速引入可持续性改进的方法

相关文档：

- 使用多个账户整理您的 AWS 环境

相关示例：

- Automate account creation, and resource provisioning using Service Catalog, AWS Organizations, and AWS Lambda
- Strengthen the DevOps pipeline and protect data with AWS Secrets Manager, AWS KMS, and AWS Certificate Manager

相关工具：

- AWS CloudFormation Guard
- Landing Zone Accelerator on AWS

SEC01-BP07 使用威胁模型识别威胁并确定缓解措施的优先级

执行威胁建模，以识别并维护一个针对工作负载的潜在威胁和相关缓解措施的最新登记表。确定威胁优先级并调整安全控制缓解措施，用于防范、检测和响应。根据工作负载以及不断变化的安全环境，重新审视和维护此登记表。

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

什么是威胁建模？

“威胁建模可识别、沟通和理解威胁及缓解措施，用于保护重要资产。” – 开源 Web 应用程序安全项目（OWASP）应用程序威胁建模

为什么应该进行威胁建模？

系统是复杂的，也会随着时间的推移会变得越来越复杂、越来越强大，从而提供更多业务价值，提高客户满意度和参与度。这意味着 IT 设计决策需要考虑数量不断增加的应用场景。由于复杂性和使用案例排列组合的数量过多，非结构化方法将通常无法有效地发现和减轻威胁。因此，您需要采用一种系统方法来列举对系统的潜在威胁，制定缓解措施并确定这些缓解措施的优先级，确保贵组织利用有限资源，在改善系统的整体安全态势方面发挥巨大作用。

威胁建模旨在提供这种系统方法，目的是在设计过程的早期发现和解决问题。此时与生命周期的后期相比，缓解措施的成本和工作量相对较低。这种方法与左移安全实践的行业原则相一致。最终，威胁建模将与组织的风险管理流程相互集成，并通过使用威胁驱动的方法，帮助您决定实施哪些控制措施。

应在什么时候进行威胁建模？

应在工作负载的生命周期中尽早开始进行威胁建模，这使您能够更灵活地处理已识别的威胁。就像软件漏洞一样，越早发现威胁，解决威胁的成本效益就越高。威胁模型是一个动态文档，应随着工作负载的变化而不断发展。随着时间的推移（包括当发生重大变更、威胁形势发生变化或您采用新功能或服务时），重新审视您的威胁模型。

实施步骤

我们如何进行威胁建模？

可以采用许多不同的方式来进行威胁建模。就像编程语言一样，每种方式都有优点和缺点，您应该选择最适合自己的方式。一种方法是从 Shostack 的威胁建模 4 问题框架开始，该框架提出开放式问题，可为威胁建模工作提供结构：

1. 我们正在做什么？

   该问题旨在帮助您了解所构建的系统并达成一致意见，以及了解与安全性相关的系统细节。创建模型或图表是回答该问题的常用方法，因为这有助于对所构建的内容进行可视化，例如使用数据流图。写下关于系统的假设和重要细节也有助于定义范围内的内容。这使每个参与威胁建模的人员都能专注于同一件事，避免因在超出范围的主题（包括过时的系统版本）上走弯路而浪费时间。例如，如果您要构建一个 Web 应用程序，那么可能不值得花时间为浏览器客户端操作系统可信引导顺序进行威胁建模，因为您无法在设计中影响这一点。

2. 会出现什么问题？

该问题可帮助您识别系统存在的威胁。威胁是指会产生不必要的影响，也可能影响系统安全的意外或故意行为或事件。如果不清楚哪里可能出现问题，您就无从应对。

对于可能出现的问题，并没有一个规范的列表。创建此列表需要团队中的所有个人和参与威胁建模工作的相关角色集思广益并展开协作。您可以通过使用识别威胁的模型（如 STRIDE）来帮助集思广益，该模型建议了不同的评估类别：欺骗、篡改、抵赖、信息披露、拒绝服务和权限提升。此外，您可能希望通过回顾现有的列表和研究来帮助集思广益，寻找灵感，其中包括 OWASP Top 10、HiTrust 威胁目录和贵组织自己的威胁目录。

3. 我们要怎么做？

   与前面的问题一样，我们不可能得到包含所有缓解措施的规范清单。这一步骤需要考虑的是上一步中确定的威胁、威胁行动者和要改进的领域。

   安全性和合规性是您与 AWS 共同承担的责任。重要的是要明白，当您问“我们要怎么做？”时，您也在问“谁负责做这件事？”。了解您和 AWS 之间的责任平衡有助于将威胁建模工作的范围限定在您控制的缓解措施范围内，这些缓解措施通常是 AWS 服务配置选项和您自己的系统特定缓解措施的组合。

   对于共担责任中 AWS 应承担的部分，您会发现 AWS 服务在许多合规计划的范围内。这些计划可帮助您理解 AWS 用以维持云安全性与合规性的可靠控制机制。AWS 客户可以从 AWS Artifact 下载这些计划的审核报告。

   无论您使用哪项 AWS 服务，客户始终要承担一部分责任，并且与这些责任相一致的缓解措施应包含在威胁模型中。对于 AWS 服务自身的安全控制缓解措施，您需要考虑跨域实施安全控制措施，包括身份和访问管理（身份验证和授权）、数据保护（静态和传输中）、基础设施安全性、日志记录和监控等域。每项 AWS 服务的文档都包含一个专门的安全章节，其中提供的安全控制机制指导可用作缓解措施。重要的是，需要考虑您正在编写的代码及其代码依赖项，并思考可以设置哪些控制机制来应对这些威胁。这些控制机制可以是输入验证、会话处理和边界处理等内容。通常，大多数漏洞都是在自定义代码中引入，因此请重点关注这一领域。

4. 我们做得好吗？

   该问题旨在随着时间的推移，让您的团队和组织提高威胁模型的质量并加快执行威胁建模的速度。通过将实践、学习、教学和回顾相结合可以取得这些改进。要想深入了解并亲身体验，建议您和团队完成“适合构建者的威胁建模方式”培训课程或讲习会。此外，如果您正在寻找如何将威胁建模集成到组织的应用程序开发生命周期中的指导，请参阅 AWS 安全博客上的《How to approach threat modeling》一文。

Threat Composer

为了协助并指导执行威胁建模，您可以考虑使用 Threat Composer 工具，缩短进行威胁建模时实现价值的时间。该工具有助于您执行以下操作：

- 撰写符合威胁语法、能够在自然非线性工作流程中发挥作用的有用威胁语句
- 生成人类可读的威胁模型
- 生成机器可读的威胁模型，允许您将威胁模型视为代码
- 使用 Insights 控制面板协助您快速确定质量和覆盖范围有待改进的方面

如需更多参考，请访问 Threat Composer 并切换到系统定义的示例工作区。

资源

相关最佳实践：

- SEC01-BP03 识别并验证控制目标
- SEC01-BP04 随时了解安全威胁和建议
- SEC01-BP05 缩小安全管理范围
- SEC01-BP08 定期评估并实施新的安全服务和功能

相关文档：

- How to approach threat modeling（AWS 安全博客）
- NIST: Guide to Data-Centric System Threat Modelling

相关视频：

- AWS Summit ANZ 2021 - How to approach threat modelling
- AWS Summit ANZ 2022 - Scaling security – Optimise for fast and secure delivery

相关培训：

- Threat modeling the right way for builders – AWS Skill Builder virtual self-paced training
- Threat modeling the right way for builders – AWS 讲习会

相关工具：

- Threat Composer

SEC01-BP08 定期评估并实施新的安全服务和功能

评估并实施 AWS 和 AWS 合作伙伴提供的安全服务和功能，以此改善工作负载的安全态势。

期望结果：已经采取标准做法，可获取 AWS 和 AWS 合作伙伴发布的新功能及服务的信息。针对环境和工作负载评估了这些新功能对当前和新控制措施的设计有何影响。

常见反模式：

- 未订阅 AWS 博客和 RSS 源，无法及时了解相关的新功能和服务
- 依赖二手来源来了解有关安全服务和功能的新闻与动态
- 没有鼓励企业中的 AWS 用户随时了解最新动态

建立此最佳实践的好处：如果能够随时了解新的安全服务和功能，就可以针对云环境和工作负载中控制措施的实施，制定出明智的决策。大家可以通过这些来源提高对不断变化的安全形势的认识，以及了解如何使用 AWS 服务来防范新兴的威胁。

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

AWS 通过多种渠道向客户告知新的安全服务和功能：

- AWS 的新功能
- AWS 新闻博客
- AWS 安全博客
- AWS 安全公告
- AWS 文档概览

您可以使用 Amazon Simple Notification Service（Amazon SNS）订阅 AWS 每日功能更新主题，获取全面的每日更新摘要。一些安全服务（例如 Amazon GuardDuty 和 AWS Security Hub）会提供自己的 SNS 主题，方便您随时了解这些特定服务的新标准、调查发现和其他更新。

每年在全球举行的会议、活动和网络研讨会上，同样会公布和详细介绍新的服务及功能。每年的 AWS re:Inforce 安全会议以及内容更全面的 AWS re:Invent 会议尤其需要注意。前面提到的 AWS 新闻频道

会分享这些有关安全和其他服务的会议公告，您可以在 YouTube 上的 AWS Events 频道上在线观看深度探讨分组会议，其中提供了丰富的信息。

您也可以向 AWS 账户账户团队询问最新的安全服务更新和建议。如果没有这些团队的直接联系信息，您可以填写销售支持表与该团队取得联系。同样，如果您订阅了 AWS Enterprise Support，技术客户经理（TAM，Technical Account Manager）每周都会向您发布更新内容，而您也可以安排与他们定期开展审查会议。

实施步骤

1. 使用您最喜欢的 RSS 阅读器订阅各种博客和公告，或订阅每日功能更新 SNS 主题。
2. 评估要参加哪些 AWS 活动，获得有关新功能和服务的第一手信息。
3. 如果有任何有关更新安全服务和功能的问题，请安排与您的 AWS 账户团队进行讨论。
4. 请考虑订阅 Enterprise Support，这样就可以定期咨询技术客户经理（TAM）。

资源

相关最佳实践：

- PERF01-BP01 了解并掌握可用的云服务和功能
- COST01-BP07 及时了解新发布的服务

身份和访问管理

问题

- SEC 2. 如何管理人员和计算机的身份验证？
- SEC 3. 如何管理人员和机器的权限？

SEC 2. 如何管理人员和计算机的身份验证？

在操作安全 AWS 工作负载时，您需要管理两类身份。

- 人员身份：需要访问 AWS 环境和应用程序的人员身份可以分类为三个组：员工、第三方和用户。

  员工 组包括作为组织成员的管理员、开发人员和操作员。他们需要访问权限才能管理、构建和运营您的 AWS 资源。

第三方 是外部协作者，如承包商、供应商或合作伙伴。他们与您的 AWS 资源交互，这是他们与您的组织互动的一部分。

用户 是应用程序的使用者。他们通过 Web 浏览器、客户端应用程序、移动应用程序或交互式命令行工具访问您的 AWS 资源。

- 机器身份：工作负载应用程序、操作工具和组件需要拥有身份，才能向 AWS 服务发出请求，例如读取数据。这些身份还包括在您的 AWS 环境（例如 Amazon EC2 实例或 AWS Lambda 函数）中运行的机器。您还可以管理需要访问 AWS 环境的外部方或 AWS 外部的机器的机器身份。

最佳实践

- SEC02-BP01 使用强大的登录机制
- SEC02-BP02 使用临时凭证
- SEC02-BP03 安全地存储和使用密钥
- SEC02-BP04 依赖集中式身份提供程序
- SEC02-BP05 定期审计和轮换凭证
- SEC02-BP06 使用用户组和属性

SEC02-BP01 使用强大的登录机制

当不使用多重身份验证（MFA）等机制时，登录（使用登录凭证的身份验证）可能会带来风险，特别是在登录凭证被无意泄露或很容易猜到的情况下。使用强大的登录机制，通过要求使用 MFA 和强密码策略来降低这些风险。

期望结果：通过为 AWS Identity and Access Management（IAM）用户、AWS 账户根用户、AWS IAM Identity Center 和第三方身份提供者使用强大的登录机制，降低意外访问 AWS 中凭证的风险。这意味着需要 MFA，强制执行强密码策略，并检测异常登录行为。

常见反模式：

- 没有为身份执行强密码策略，包括复杂密码和 MFA。
- 在不同的用户之间共享相同的凭证。
- 不对可疑的登录使用检测性控制。

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

有几种方法可以让人员身份登录到 AWS。在向 AWS 进行身份验证时，AWS 最佳做法是依赖于使用联合身份验证的集中式身份提供者（AWS IAM 和集中式 IdP 之间的直接 SAML 2.0 联合身份验证，或使用 AWS IAM Identity Center）。在这种情况下，请与身份提供者或 Microsoft Active Directory 建立安全登录过程。

第一次开设 AWS 账户时，您会从 AWS 账户根用户开始。您应仅使用账户根用户为用户（以及为需要根用户的任务）设置访问权限。在开设 AWS 账户后立即为账户根用户开启多重身份验证（MFA），并使用 AWS 最佳实践指南来保护根用户的安全，这一点至关重要。

AWS IAM Identity Center 专为员工用户设计，您可以在服务中创建和管理用户身份，并使用 MFA 保护登录流程。另一方面，AWSCognito 专为客户身份和访问管理（CIAM）而设计，它为应用程序中的外部用户身份提供用户池和身份提供者。

如果您在 AWS IAM Identity Center 中创建用户，请确保该服务中的登录过程安全，并开启 MFA。对于应用程序中的外部用户身份，可以使用 Amazon Cognito 用户池并确保该服务中的登录过程安全，也可以通过 Amazon Cognito 用户池中支持的身份提供者之一进行登录。

此外，对于 AWS IAM Identity Center 中的用户，可以使用 AWS Verified Access，通过在向他们授予访问 AWS 资源的权限之前验证用户的身份和设备状态，来提供一层额外的安全性。

如果您使用的是 AWS Identity and Access Management（IAM）用户，请使用 IAM 来保护登录过程。

可以同时使用 AWS IAM Identity Center 和直接 IAM 联合身份验证来管理对 AWS 的访问权限。可以使用 IAM 联合身份验证来管理对 AWS Management Console和服务的访问权限，并使用 IAM Identity Center 来管理对诸如 QuickSight 或 Amazon Q Business 等业务应用程序的访问权限。

无论采用何种登录方法，执行强登录策略都非常关键。

实施步骤

以下是一般的强登录建议。应根据公司策略或使用 NIST 800-63 等标准，对配置的实际设置进行设定。

- 需要 MFA。对于人员身份和工作负载，要求使用 MFA 是 IAM 最佳实践。启用 MFA 提供了一层额外的安全保障，这会要求用户提供登录凭证和一次性密码（OTP），或从硬件设备加密验证和生成的字符串。
- 强制执行最小密码长度，这是密码强度的主要因素。
- 强制执行密码复杂性，使密码更难以猜到。
- 允许用户更改其密码。

- 创建个人身份而不是共享凭证。通过创建个人身份，您可以为每个用户提供一组唯一的安全凭证。个人用户可以审计每个用户的活动。

IAM Identity Center 建议：

- IAM Identity Center 在使用默认目录时提供了预定义的密码策略，该策略确定了密码长度、复杂性和重用要求。
- 当身份源为默认目录、AWS Managed Microsoft AD 或 AD Connector 时，启用 MFA 并为 MFA 配置“上下文感知”或“始终开启”设置。
- 允许用户注册自己的 MFA 设备。

Amazon Cognito 用户池目录建议：

- 配置密码长度设置。
- 对于用户，要求使用 MFA。
- 使用 Amazon Cognito 用户池高级安全设置可实现自适应身份验证（可阻止可疑登录）等功能。

IAM 用户建议：

- 最好是使用 IAM Identity Center 或直接联合。不过，您可能需要 IAM 用户。在这种情况下，为 IAM 用户设置密码策略。您可以使用密码策略来定义诸如最小长度、密码是否需要非字母字符之类的要求。
- 创建 IAM 策略来强制执行 MFA 登录，允许用户管理自己的密码和 MFA 设备。

资源

相关最佳实践：

- SEC02-BP03 安全地存储和使用密钥
- SEC02-BP04 依赖集中式身份提供程序
- SEC03-BP08 在组织内安全地共享资源

相关文档：

- AWS IAM Identity Center 密钥策略

- IAM 用户密码策略
- 设置 AWS 账户根用户密码
- Amazon Cognito password policy
- AWS 凭证
- IAM 安全最佳实践

相关视频：

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP02 使用临时凭证

进行任何类型的身份验证时，最好使用临时凭证而不是长期凭证，来降低或消除诸如凭证被无意泄露、共享或被盗之类的风险。

期望结果：为了降低长期凭证的风险，尽量对人类身份和机器身份使用临时凭证。长期凭证会带来诸多风险，例如通过上传到公共存储库而泄露。使用临时凭证可以大幅降低凭证被泄露的几率。

常见反模式：

- 开发人员使用 IAM 用户的长期访问密钥，而不是使用联合身份验证从 CLI 获得临时凭证。
- 开发人员在他们的代码中嵌入长期访问密钥，并将该代码上传到公有 Git 存储库。
- 开发人员在移动应用程序中嵌入长期访问密钥，然后在应用商店中提供这些密钥。
- 用户与其他用户共享长期访问密钥，或员工离开公司时仍持有长期访问密钥。
- 当可以使用临时凭证时，对机器身份使用长期访问密钥。

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

对所有 AWS API 和 CLI 请求使用临时安全凭证，而不是长期凭证。在几乎所有情况下，对 AWS 服务的 API 和 CLI 请求都必须使用 AWS 访问密钥进行签名。这些请求可以使用临时凭证或长期凭证进行签名。只有在使用 IAM 用户或 AWS 账户根用户时，才应该使用长期凭证（也称为长期访问密钥）。在联合到 AWS 或通过其他方法代入 IAM 角色时，系统会生成临时凭证。即使使用登录凭证访问 AWS Management Console，系统也会生成临时凭证供您调用 AWS 服务。需要用到长期凭证的情况很少，所以可以使用临时凭证完成几乎所有任务。

尽量不要使用长期凭证，要多用临时凭证，并且还应少用 IAM 用户进行能访问，而应多用联合身份验证和 IAM 角色进行访问。虽然过去常使用 IAM 用户来访问人类身份和机器身份，但现在不建议使用 IAM 用户，避免使用长期访问密钥所带来的风险。

实施步骤

人员身份

对于员工、管理员、开发人员和操作员等员工身份：

- 您应该依赖集中式身份提供程序，并要求人类用户配合使用联合身份验证与身份提供程序两种方法，以便使用临时凭证访问 AWS。可以通过对每个 AWS 账户进行直接联合身份验证或使用 AWS IAM Identity Center 和您选择的身份提供者，来完成用户的联合身份验证。与使用 IAM 用户相比，联合身份验证除了消除使用长期凭证的情况之外，还具有许多优势。用户也可以从直接联合的命令行或通过使用 IAM Identity Center 来请求获得临时凭证。这意味着能够大幅减少需要使用 IAM 用户或用户长期凭证的情况。

对于第三方身份：

- 在授予软件即服务（SaaS）提供商等第三方访问 AWS 账户中资源的权限时，您可以使用跨账户角色和基于资源的策略。此外，可以为 B2B SaaS 客户或合作伙伴使用 Amazon Cognito OAuth 2.0 grant 客户端凭证流。

通过 Web 浏览器、客户端应用程序、移动应用程序或交互式命令行工具访问 AWS 资源的用户身份：

- 如果需要批准消费者或客户申请访问 AWS 资源，您可以使用 Amazon Cognito 身份池或 Amazon Cognito 用户池提供临时凭证。凭证的权限是通过 IAM 角色配置的。您也可以为未经身份验证的来宾用户定义一个具有有限权限的单独的 IAM 角色。

机器身份

对于机器身份，您就可能需要使用长期凭证了。在这些情况下，您应该要求工作负载使用具有 IAM 角色的临时凭证来访问 AWS。

- 对于Amazon Elastic Compute Cloud（Amazon EC2），您可以使用适用于 Amazon EC2 的角色。
- AWS Lambda 让您能够配置 Lambda 执行角色授予权限，以便利用临时凭证执行 AWS 操作。AWS 服务还有许多其他类似的模型，可以使用 IAM 角色授予临时凭证。
- 对于 IoT 设备，您可以使用 AWS IoT Core 凭证提供程序来请求临时凭证。

- 对于需要访问 AWS 资源的本地系统或在 AWS 之外运行的系统，您可以使用 IAM Roles Anywhere。

某些情况下不支持临时凭证，此时需要使用长期凭证。在这些情况下，可以定期审计和轮换这些凭证和定期轮换访问密钥。对于高度受限的 IAM 用户访问密钥，请考虑以下其它安全措施：

- 授予高度受限的权限：
  - 遵守最低权限原则（特定于操作、资源和条件）。
  - 考虑仅向 IAM 用户授予针对一个特定角色的 AssumeRole 操作。根据本地架构，此方法有助于隔离和保护长期 IAM 凭证。
- 在 IAM 角色信任策略中限制支持的网络来源和 IP 地址。
- 监控使用情况并针对未使用的权限或滥用行为设置警报（使用 AWS CloudWatch Logs 指标筛选条件和警报）。
- 强制执行权限边界 [服务控制策略（SCP）和权限边界相辅相成 - SCP 是粗粒度的，而权限边界是细粒度的]。
- 实施用于预置和（在本地保管库中）安全存储凭证的过程。

对于需要长期凭证的场景，其它一些选项包括：

- 构建自己的令牌出售 API（使用 Amazon API Gateway）。
- 对于必须使用长期凭证或 AWS 访问密钥以外凭证的场景（如数据库登录），可以使用旨在处理密钥管理的服务，如 AWS Secrets Manager。Secrets Manager 可以简化加密密钥的管理、轮换和安全存储。许多 AWS 服务都支持与 Secrets Manager direct integration。
- 对于多云集成，可以使用基于源凭证服务提供商（CSP）凭证的身份联合验证（请参阅 AWS STS AssumeRoleWithWebIdentity）。

有关轮换长期凭证的更多信息，请参阅轮换访问密钥。

资源

相关最佳实践：

- SEC02-BP03 安全地存储和使用密钥
- SEC02-BP04 依赖集中式身份提供程序
- SEC03-BP08 在组织内安全地共享资源

相关文档：

- 临时安全凭证
- AWS 凭据
- IAM 安全最佳实践
- IAM 角色
- IAM Identity Center
- 身份提供程序和联合身份验证
- 轮换访问密钥
- 安全合作伙伴解决方案：访问和访问控制
- AWS 账户根用户
- Access AWS using a Google Cloud Platform native workload identity
- How to access AWS resources from Microsoft Entra ID tenants using AWS Security Token Service

相关视频：

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP03 安全地存储和使用密钥

工作负载需要能够自动向数据库、资源和第三方服务证明其身份。这是使用秘密访问凭证（如 API 访问密钥、密码和 OAuth 令牌）完成的。使用专门构建的服务来存储、管理和轮换这些凭证，有助于降低这些凭证泄露的可能性。

期望结果：实施安全管理应用程序凭证的机制来实现以下目标：

- 确定工作负载需要哪些密钥。
- 尽量使用短期凭证代替长期凭证，从而减少所需长期凭证的数量。
- 建立安全存储并自动轮换剩余的长期凭证。
- 审核对工作负载中存在的密钥的访问。
- 持续监控，验证开发期间没有在源代码中嵌入任何密钥。
- 降低凭证被无意中泄露的可能性。

常见反模式：

- 不轮换凭证。
- 将长期凭证存储在源代码或配置文件中。
- 在未加密状态下静态存储凭证。

建立此最佳实践的好处：

- 对存储的凭证进行静态和传输中加密。
- 通过 API 来把关对凭证的访问（可将 API 看作凭证自动售货机）。
- 审核和记录对凭证的访问（包括读和写）。
- 关注点分离：凭证轮换由一个单独的组件执行，该组件可与架构的其余部分隔离开来。
- 密钥自动按需分发给软件组件，并在中心位置进行轮换。
- 可以精细地控制对凭证的访问。

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

过去，用于对数据库、第三方 API、令牌和其他密钥进行身份验证的凭证，可能已嵌入到源代码或环境文件中。AWS 提供了几种机制来安全地存储这些凭证，自动轮换凭证，并审核凭证的使用情况。

妥善管理密钥的方法是遵循相关指导，正确地删除、替换和轮换密钥。最安全的凭证是不必存储、管理或处理的凭证。某些凭证可能不再是正常运行工作负载所必需的，可以安全地删除。

对于仍然是正常运行工作负载所必需的凭证，可能有机会用临时或短期凭证替换长期凭证。例如，相较于对 AWS 秘密访问密钥进行硬编码，不妨考虑使用 IAM 角色将长期凭证替换为临时凭证。

可能无法删除或替换某些长期密钥。这些密钥可以存储在 AWS Secrets Manager 等服务中，在其中得到集中存储、管理和定期轮换。

对工作负载的源代码和配置文件进行审计，可以发现许多类型的凭证。下表总结了处理常见凭证类型的策略：

凭证类型	描述	建议采取的策略
IAM 访问密钥	用于在工作负载内代入 IAM 角色的 AWS IAM 访问密钥和私有密钥	替换：改用分配给计算实例（例如 Amazon EC2 或 AWS Lambda）的 IAM 角色。为了与需要访问 AWS 账户中资源的第三方实现互操作性，请询问他们是否支持 AWS 跨账户访问。对于移动应用程序，请考虑通过 Amazon Cognito 身份池（联合身份）使用临时凭证。对于在 AWS 之外运行的工作负载，请考虑使用 IAM Roles Anywhere 或 AWS Systems Manager 混合激活。对于容器，请参阅 Amazon ECS 任务 IAM 角色或 Amazon EKS 节点 IAM 角色。
SSH 密钥	用于登录 Linux EC2 实例的 Secure Shell 私有密钥，可手动登录或作为自动流程的一部分登录	替换：使用 AWS Systems Manager 或 EC2 Instance Connect，通过 IAM 角色提供对 EC2 实例的编程访问权限和人类访问。
应用程序和数据库凭证	密码 – 纯文本字符串	轮换：将凭证存储在 AWS Secrets Manager 中，并尽量建立自动轮换机制。
Amazon RDS 和 Aurora 管理数据库凭证	密码 – 纯文本字符串	替换：使用 Secrets Manager 与 Amazon RDS 集成或 Amazon Aurora。此外，在某些应用场景中，一些 RDS 数据库类型可以使用 IAM 角色代替密码（有关更多详细信息，请参阅 IAM 数据库身份验证）。

凭证类型	描述	建议采取的策略
OAuth 令牌	密钥令牌 – 纯文本字符串	轮换：将令牌存储在 AWS Secrets Manager 中并配置自动轮换。
API 令牌和密钥	密钥令牌 – 纯文本字符串	轮换：存储在 AWS Secrets Manager 中，并尽量建立自动轮换机制。

一种常见的反模式是在源代码、配置文件或移动应用程序中嵌入 IAM 访问密钥。当需要 IAM 访问密钥与 AWS 服务通信时，请使用临时（短期）安全凭证。可以通过 EC2 实例的 IAM 角色、Lambda 函数的执行角色、移动用户访问的 Cognito IAM 角色和 IoT 设备的 IoT Core 策略，提供这些短期凭证。与第三方进行交互时，最好将访问权限委托给 IAM 角色，授予对账户资源的必要访问权限，而不是配置 IAM 用户并向第三方发送该用户的秘密访问密钥。

在许多情况下，工作负载需要存储与其他服务和资源进行互操作所必需的密钥。AWS Secrets Manager 旨在安全地管理这些凭证，以及 API 令牌、密码和其他凭证的存储、使用和轮换。

AWS Secrets Manager 提供五个关键功能，确保敏感凭证的安全存储和处理：静态加密、传输中加密、全面审核、精细访问控制和可扩展凭证轮换。AWS 合作伙伴提供的其他密钥管理服务或提供类似功能和保证的本地开发的解决方案，也可以接受。

在检索密钥时，可以使用 Secrets Manager 客户端缓存组件来缓存密钥，以备将来使用。检索已缓存密钥比从 Secrets Manager 中检索密钥的速度要快。此外，由于调用 Secrets Manager API 会产生费用，因此使用缓存可以降低成本。有关检索密钥的所有方法，请参阅 Get secrets。

> **Note**
>
> 某些语言可能要求您实施自己的内存加密来进行客户端缓存。

实施步骤

1. 使用自动化工具（如 Amazon CodeGuru）识别包含硬编码凭证的代码路径。
   a. 使用 Amazon CodeGuru 扫描代码存储库。审核完成后，在 CodeGuru 中按 Type=Secrets 进行筛选来查找有问题的代码行。
2. 识别可以删除或替换的凭证。

a. 识别不再需要的凭证并标明要删除。

b. 对于嵌入到源代码的 AWS 私有密钥，将其替换为与必要资源相关的 IAM 角色。如果部分工作负载在 AWS 之外，但需要 IAM 凭证才能访问 AWS 资源，请考虑采用 IAM Roles Anywhere 或 AWS Systems Manager 混合激活。

3. 对于其他需要使用轮换策略的第三方、长期密钥，请将 Secrets Manager 集成到代码中，以便在运行时检索第三方密钥。

a. CodeGuru 控制台可以使用发现的凭证在 Secrets Manager 中自动创建密钥。

b. 将 Secrets Manager 的密钥检索集成到应用程序代码中。

i. 无服务器 Lambda 函数可以使用与语言无关的 Lambda 扩展。

ii. 对于 EC2 实例或容器，AWS 用几种流行的编程语言提供了示例客户端代码，用于从 Secrets Manager 检索密钥。

4. 定期检查代码库并重新扫描，验证代码中没有添加新的密钥。

a. 考虑使用诸如 git-secrets 之类的工具，防止向源代码存储库提交新的密钥。

5. 监控 Secrets Manager 活动，发现意外使用、不适当的密钥访问或试图删除密钥的迹象。

6. 减少人类接触凭证的机会。将读取、写入和修改凭证的权限仅授予专用于此目的的 IAM 角色，并仅向一小部分操作用户提供代入该角色的权限。

资源

相关最佳实践：

- SEC02-BP02 使用临时凭证
- SEC02-BP05 定期审计和轮换凭证

相关文档：

- AWS Secrets Manager 入门
- 身份提供程序和联合身份验证
- Amazon CodeGuru 推出 Secrets Detector
- AWS Secrets Manager 如何使用 AWS Key Management Service
- Secret encryption and decryption in Secrets Manager
- Secrets Manager 博客系列文章
- Amazon RDS 宣布与 AWS Secrets Manager 集成

相关视频：

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Find Hard-Coded Secrets Using Amazon CodeGuru Secrets Detector
- Securing Secrets for Hybrid Workloads Using AWS Secrets Manager

相关讲习会：

- Store, retrieve, and manage sensitive credentials in AWS Secrets Manager
- AWS Systems Manager Hybrid Activations

SEC02-BP04 依赖集中式身份提供程序

对于工作人员身份（员工和合同工），请依赖允许您在集中位置管理身份的身份提供程序。这样就可以更轻松地跨多个应用程序和系统管理访问权限，因为您可以从单一位置创建、分配、管理、撤销和审核访问权限。

期望结果：您有一个集中式身份提供程序，可以在其中集中管理员工用户、身份验证策略 [例如要求多重身份验证（MFA）]，以及对系统和应用程序的授权（例如根据用户的群组成员资格或属性分配访问权限）。您的员工用户登录到中央身份提供程序并联合身份验证（单点登录）到内部和外部应用程序，这样用户就无需记住多个凭证。您的身份提供程序已与您的人力资源（HR）系统集成，因此人事变动会自动与身份提供程序同步。例如，如果有人离开组织，您可以自动撤消对联合应用程序和系统（包括 AWS）的访问权限。您已在身份提供程序中启用了详细的审核日志记录，并且正在监控这些日志，以便发现异常用户行为。

常见反模式：

- 不使用联合身份验证和单点登录。员工用户在多个应用程序和系统中创建单独的用户账户和凭证。
- 尚未实现员工用户身份生命周期的自动化，例如将身份提供程序与 HR 系统集成。当用户离职或变换角色时，您使用手动流程来删除或更新他们在多个应用程序和系统中的记录。

建立此最佳实践的好处：通过使用集中式身份提供程序，您可以在一个位置管理员工用户身份和策略，可以向用户和群组分配应用程序的访问权限，还可以监控用户登录活动。通过与您的人力资源（HR）系统集成，当用户的角色发生更改时，这些更改会同步到身份提供程序，并自动更新为他们分配的应用程序和权限。当用户离职时，其身份将在身份提供程序中自动被禁用，从而撤消他们对联合应用程序和系统的访问权限。

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

员工用户访问 AWS 的指南：员工用户（如组织中的员工和合同工）可能需要使用 AWS Management Console或 AWS Command Line Interface（AWS CLI）来访问 AWS，以便履行工作职能。您可以通过从集中式身份提供程序联合到 AWS，在两个层面上向员工用户授予对 AWS 的访问权限：直接联合到每个 AWS 账户，或联合到 AWS 组织中的多个账户。

要将员工用户与每个 AWS 账户直接联合，可以使用集中式身份提供程序来联合到该账户中的 AWS Identity and Access Management。IAM 的灵活性允许您启用单独的 SAML 2.0 或 Open ID Connect（OIDC）身份提供程序（针对每个 AWS 账户），并使用联合用户属性进行访问控制。员工用户会使用网络浏览器，通过提供凭证（如密码和 MFA 令牌码）来登录身份提供程序。身份提供程序向浏览器发出 SAML 断言，该断言将提交到 AWS Management Console登录 URL，以便允许用户通过代入 IAM 角色单点登录 AWS Management Console。用户还可以获取临时 AWS API 凭证用于 AWS CLI 或 AWS SDK（从 AWS STS），方法是使用身份提供程序的 SAML 断言代入 IAM 角色。

要对员工用户和 AWS 组织中的多个账户进行联合身份验证，可以使用 AWS IAM Identity Center 来集中管理员工用户对 AWS 账户和应用程序的访问权限。组织启用 Identity Center 并配置身份源。IAM Identity Center 提供一个默认身份源目录，可用来管理用户和组。您也可以选择外部身份源，方法是使用 SAML 2.0 连接外部身份提供程序并使用 SCIM 自动预置用户和组，或使用 AWS Directory Service 连接到 Microsoft AD 目录。配置身份源后，即可通过以下方法为用户和组分配对 AWS 账户的访问权限：在权限集中定义最低权限策略。员工用户可以通过中央身份提供程序进行身份验证，登录 AWS 访问门户并单点登录到 AWS 账户以及分配给他们的云应用程序。用户可以配置 AWS CLI v2 来使用 Identity Center 进行身份验证，并获取用于运行 AWS CLI 命令的凭证。Identity Center 还支持通过单点登录访问 AWS 应用程序，例如 Amazon SageMaker AI Studio 和 AWS IoT Sitewise Monitor portals。

遵循上述指导后，员工用户在 AWS 上管理工作负载时，将不再需要使用 IAM 用户和组来进行通用的操作。相反，用户和组是在 AWS 外部进行管理，并且能够以联合身份访问 AWS 资源。联合身份使用集中式身份提供程序定义的组。您应该识别并删除 AWS 账户中不再需要的 IAM 组、IAM 用户和长期用户凭证（密码和访问密钥）。您可以查找未使用的凭证（使用 IAM 凭证报告）、删除相应的 IAM 用户和删除 IAM 组。您可以将服务控制策略（SCP）应用于组织，帮助防止创建新的 IAM 用户和组，并强制通过联合身份访问 AWS。

> **Note**
>
> 您负责处理 SCIM 访问令牌的轮换，如自动预置文档中所述。此外，您还负责轮换支持身份联合验证的证书。

应用程序用户的指南：通过将 Amazon Cognito 用作集中式身份提供者，您可以管理应用程序（例如移动应用程序）用户的身份。Amazon Cognito 支持对 Web 和移动应用程序进行身份验证、授权和用户管理。Amazon Cognito 提供可扩展到数百万用户的身份存储，支持社交网络和企业身份联合验证，并提供高级安全功能来协助保护用户和业务。您可以将自定义 Web 或移动应用程序与 Amazon Cognito 集成，以便在几分钟内为应用程序添加用户身份验证和访问控制。Amazon Cognito 以 SAML 和 Open ID Connect（OIDC）等开放式身份标准为基础构建，支持各种合规性法规，并与前端和后端开发资源集成。

实施步骤

员工用户访问 AWS 的步骤

- 通过以下某种方法，使用集中式身份提供程序向 AWS 联合验证员工身份：
  - 通过与身份提供程序联合，使用 IAM Identity Center 来允许单点登录到 AWS 组织中的多个 AWS 账户。
  - 使用 IAM 将身份提供程序直接连接到每个 AWS 账户，从而实现精细的联合访问。
- 识别并移除被联合身份取代的 IAM 用户和群组。

适用于应用程序用户的步骤

- 将 Amazon Cognito 用作应用程序的集中式身份提供程序。
- 使用 OpenID Connect 和 OAuth 将自定义应用程序与 Amazon Cognito 集成。您可以使用 Amplify 库开发自定义应用程序，这些库提供了与各种 AWS 服务（例如用于身份验证的 Amazon Cognito）集成的简单接口。

资源

相关最佳实践：

- SEC02-BP06 使用用户组和属性
- SEC03-BP02 授予最低访问权限
- SEC03-BP06 基于生命周期管理访问权限

相关文档：

- AWS 中的身份联合验证
- IAM 安全最佳实操

- AWS Identity and Access Management 最佳实践
- IAM Identity Center 委派管理入门
- How to use customer managed policies in IAM Identity Center for advanced use cases
- AWS CLI v2: IAM Identity Center credential provider

相关视频：

- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2018: Mastering Identity at Every Layer of the Cake

相关示例：

- Workshop: Using AWS IAM Identity Center to achieve strong identity management

相关工具：

- AWS 安全能力合作伙伴：身份和访问管理
- saml2aws

SEC02-BP05 定期审计和轮换凭证

定期审计和轮换凭证，以限制凭证可用于访问资源的时间。长期凭证会产生许多风险，可通过定期轮换长期凭证来降低这些风险。

期望结果：实施凭证轮换，以帮助降低长期凭证相关风险。定期审计并纠正不符合凭证轮换策略的情况。

常见反模式：

- 不审计凭证的使用情况。
- 不必要地使用长期凭证。
- 使用长期凭证，不定期轮换。

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

当您无法依赖于临时凭证而需要长期凭证时，请审计凭证，以验证定义的控制措施 [例如多重身份验证（MFA）] 得以实施、定期轮换并具有适当的访问级别。

（最好通过自动化工具）定期验证，以确保实施正确的控制措施。对于人员身份，您应要求用户定期更改他们的密码并停用访问密钥，以支持临时凭证。从 AWS Identity and Access Management（IAM）用户转向集中式身份时，您可以生成凭证报告来审计您的用户。

我们还建议您在身份提供者中实施并监控 MFA。您可以设置 AWS Config 规则或使用 AWS Security Hub 安全标准来监控用户是否配置了 MFA。考虑使用 IAM Roles Anywhere 为机器身份提供临时凭证。在无法使用 IAM 角色和临时凭证的情况下，需要经常审计和轮换访问密钥。

实施步骤

- 定期审计凭证：对您的身份提供者和 IAM 中配置的身份进行审计，这有助于验证只有经过授权的身份才能访问您的工作负载。此类身份可能包括但不限于 IAM 用户、AWS IAM Identity Center 用户、Active Directory 用户或不同上游身份提供者中的用户。例如，删除离开组织的人员，并删除不再需要的跨账户角色。制定流程，以定期审计 IAM 实体所访问服务的权限。这有助于您确定需要修改的策略，以删除任何未使用的权限。使用凭证报告和 AWS Identity and Access Management Access Analyzer 来审计 IAM 凭证和权限。您可以使用 Amazon CloudWatch 为 AWS 环境中调用的特定 API 调用设置警报。Amazon GuardDuty 还可以提醒您注意意外活动，出现这种提醒，可表明对 IAM 凭证的访问过于宽松，或出现了意外访问情况。
- 定期轮换凭证：当您无法使用临时凭证时，请定期轮换长期 IAM 访问密钥（最多每 90 天一次）。如果在您不知情的情况下无意中泄露了访问密钥，这将限制凭证用于访问资源的时间。有关轮换 IAM 用户的访问密钥的信息，请参阅《轮换访问密钥》。
- 查看 IAM 权限：要增强您的 AWS 账户的安全性，请定期查看和监控每个 IAM 策略。验证这些策略是否遵循最低权限原则。
- 考虑自动创建和更新 IAM 资源：IAM Identity Center 自动执行许多 IAM 任务，比如角色和策略管理。或者，AWS CloudFormation 可用于自动部署 IAM 资源（包括角色和策略），以减少人为错误的机会，因为可以验证模板和控制版本。
- 对于机器身份，使用 IAM Roles Anywhere 替换 IAM 用户：IAM Roles Anywhere 将使您能够在传统上无法使用角色的领域（例如本地服务器）使用角色。IAM Roles Anywhere 使用可信的 X.509 certificate 向 AWS 进行身份验证并接收临时凭证。使用 IAM Roles Anywhere 便无需轮换这些凭证，因为长期凭证不再存储在本地环境中。请注意，您需要监控 X.509 证书，并在该证书即将到期时轮换它。

资源

相关最佳实践：

- SEC02-BP02 使用临时凭证
- SEC02-BP03 安全地存储和使用密钥

相关文档：

- AWS Secrets Manager 入门
- IAM 最佳实践
- 身份提供程序和联合身份验证
- 安全合作伙伴解决方案：访问和访问控制
- 临时安全凭证
- 获取您的 AWS 账户的凭证报告

相关视频：

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP06 使用用户组和属性

根据用户组和属性来定义权限，可以减少策略的数量和复杂性，这样就更容易实现最低权限原则。您可以通过用户组，根据用户在企业中履行的职能，从一个位置管理多个用户的权限。当用户履行类似的职能但面向的是不同资源子集时，可以利用部门、项目或位置等属性来提供额外的一层权限范围。

期望结果：您可以将基于职能的权限更改应用到履行该职能的所有用户。利用组成员资格和属性管控用户的权限，减少在单个用户级别管理权限的需求。您在身份提供者（IdP）中定义的组和属性会自动传播到您的 AWS 环境。

常见反模式：

- 分别管理每个用户的权限，然后在多个用户之间复制。
- 定义过于宽泛的组，授予过于宽泛的权限。
- 定义过于精细的组，造成成员重复和混淆。

- 对不同资源子集使用具有重复权限的组，但其实可以改为使用属性来进行控制。
- 在管理组、属性和成员资格时，没有使用与您的 AWS 环境集成的标准化身份提供者。
- 使用 AWS IAM Identity Center 会话时使用角色链

在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

在称为策略 的文档中定义 AWS 权限，这些策略与某个主体关联，例如用户、组、角色或资源。可以根据工作职能、工作负载和 SDLC 环境来组织权限分配（组、权限、账户），从而扩展权限管理。对于员工团队，通过这种方法，您可以根据用户在企业中履行的职能来定义组，而不是根据所访问的资源来定义。例如，WebAppDeveloper 组可能附有策略，用于在开发账户中配置 Amazon CloudFront 等服务。`AutomationDeveloper` 组可能与 `WebAppDeveloper` 组具有一些重叠的权限。可以将这些共有的权限放置在单独的策略中并与这两个组关联，而不是将来自这两个职能部门的用户都放在同一个 `CloudFrontAccess` 组中。

除了使用组之外，您还可以使用属性来进一步限定访问范围。例如，您可以为 `WebAppDeveloper` 组中的用户设置“项目”属性，用来限定对其项目特定资源的访问权限。使用这种技巧，您就无需为处理不同项目的应用程序开发人员设置不同的组，因为他们除了项目不同外，所需的权限其实并无不同。在权限策略中引用属性的方式取决于权限的来源，这些权限可能包含在联合身份验证协议（例如 SAML、OIDC 或 SCIM）中，可能是自定义的 SAML 断言，也可能是在 IAM Identity Center 中设置。

实施步骤

1. 确定要在何处定义组和属性：
   a. 按照 SEC02-BP04 依赖集中式身份提供者中的指南，您可以确定是要在身份提供者中或 IAM Identity Center 中定义组和属性，还是在特定账户中使用 IAM 用户组。
2. 定义组：
   a. 根据职能和所需访问权限范围来确定组。考虑使用分层结构或命名约定来有效地整理组。
   b. 如果在 IAM Identity Center 中定义，则创建组并使用权限集关联所需的访问权限级别。
   c. 如果在外部身份提供者中定义，请确定该提供者是否支持 SCIM 协议，并考虑在 IAM Identity Center 中启用自动预置。此功能可在您的提供者和 IAM Identity Center 之间同步组的创建、成员指派和删除。
3. 定义属性：
   a. 如果使用外部身份提供者，则默认情况下，SCIM 和 SAML 2.0 协议都提供一些属性。使用 `https://aws.amazon.com/SAML/Attributes/PrincipalTag` 属性名称，可以通过

SAML 断言来定义并传递其它属性。有关定义和配置自定义属性的指南，请咨询身份提供者的文档。

b. 如果在 IAM Identity Center 中定义角色，请启用基于属性的访问权限控制（ABAC）功能，然后根据需要定义属性。考虑符合组织的结构或资源标签策略的属性。

如果您需要从通过 IAM Identity Center 代入的 IAM 角色中获得的 IAM 角色链，则诸如 `source-identity` 和 `principal-tags` 的值将不会传播。有关更多详细信息，请参阅启用并配置访问控制属性。

1. 根据组和属性限定权限范围：

   a. 考虑在权限策略中加入条件，将主体的属性与所访问资源的属性进行比较。例如，您可以定义一个条件，仅当 `PrincipalTag` 条件键的值与相同名称的 `ResourceTag` 键的值匹配时，才支持访问资源。

   b. 定义 ABAC 策略时，请遵循 ABAC 授权最佳实践和示例中的指导。

   c. 随着组织需求的发展，请定期审核和更新组和属性结构，以确保最佳的权限管理。

资源

相关最佳实践：

- SEC02-BP04 依赖集中式身份提供程序
- SEC03-BP02 授予最低访问权限
- COST02-BP04 实施组和角色

相关文档：

- IAM 最佳实践
- 在 IAM Identity Center 中管理身份
- 什么是适用于 AWS 的 ABAC？
- 在 IAM Identity Center 中使用 ABAC
- ABAC 策略示例

相关视频：

- Managing user permissions at scale with AWS IAM Identity Center

- Mastering identity at every layer of the cake

SEC 3. 如何管理人员和机器的权限？

管理权限，控制需要访问 AWS 和您工作负载的人员和机器身份的访问权限。权限可让您控制哪些人可以在什么条件下访问哪些内容。通过为特定的人员身份和机器身份设置权限，可以向这些身份授予对特定资源执行特定服务操作的访问权限。此外，您可以为要授予的访问权限指定必须满足的条件。

最佳实践

- SEC03-BP01 定义访问要求
- SEC03-BP02 授予最低访问权限
- SEC03-BP03 建立紧急访问流程
- SEC03-BP04 持续减少权限
- SEC03-BP05 为您的组织定义权限防护机制
- SEC03-BP06 基于生命周期管理访问权限
- SEC03-BP07 分析公共和跨账户访问
- SEC03-BP08 在组织内安全地共享资源
- SEC03-BP09 与第三方安全地共享资源

SEC03-BP01 定义访问要求

管理员、最终用户或其他组件都需要访问工作负载的每个组件或资源。明确定义什么人或什么内容应该有权访问每个组件，选择适当的身份类型以及身份验证和授权方法。

常见反模式：

- 在应用程序中进行硬编码或存储密码。
- 向每个用户授予自定义权限。
- 使用长期有效的凭证。

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

管理员、最终用户或其他组件都需要访问工作负载的每个组件或资源。明确定义什么人或什么内容应该有权访问每个组件，选择适当的身份类型以及身份验证和授权方法。

应提供对组织内 AWS 账户的常规访问，方法是使用联合身份访问或集中式身份提供者。您还应将身份管理集中处理，确保对于 AWS 将访问集成到员工访问生命周期中已建立了既定做法。例如，当员工转岗到具有不同访问级别的职位时，该员工的小组成员资格也应进行更改以反映新的访问要求。

在定义非人类身份的访问要求时，请确定哪些应用程序和组件需要访问权限以及如何向其授予权限。建议使用通过最低权限访问模型构建的 IAM 角色。AWS托管式策略提供了预定义的 IAM 策略，这些策略涵盖了大多数常见使用案例。

AWS 服务（例如 AWS Secrets Manager 和 AWS Systems Manager Parameter Store）可以帮助在无法使用 IAM 角色的情况下，安全地将密码与应用程序或工作负载分离。在 Secrets Manager 中，您可以为凭证建立自动轮换。您可以通过使用您在创建参数时指定的唯一名称，使用 Systems Manager 来引用脚本、命令、SSM 文档、配置和自动化工作流中的参数。

您可以使用 AWS IAM Roles Anywhere 来获取 IAM 中的临时安全凭证，这种凭证适用于在 AWS 外部运行的工作负载。您的工作负载可以使用与 AWS 应用程序相同的 IAM 策略和 IAM 角色来访问 AWS 资源。

如果可能，请优先选择短期临时凭证而不是长期静态凭证。在一些场景中，需要具有编程访问权限和长期凭证的用户，此时请使用访问密钥上次使用的信息来轮换和删除访问密钥。

如果用户需要在 AWS Management Console 之外与 AWS 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 AWS 的用户类型。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI 以使用 AWS IAM Identity Center。 • 有关 AWS SDK、工具和 AWS API 的更多信息，请参阅《AWS SDK 和工具参

哪个用户需要编程式访问权限？	目的	方式
		考指南》中的 IAM Identity Center 身份验证。
IAM	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照《IAM 用户指南》中将临时凭证用于 AWS 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 IAM 用户凭证进行身份验证。 • 有关 AWS SDK 和工具的更多信息，请参阅《AWS SDK 和工具参考指南》中的使用长期凭证进行身份验证。 • 有关 AWS API 的更多信息，请参阅《IAM 用户指南》中的管理 IAM 用户的访问密钥。

资源

相关文档：

- 基于属性的访问控制（ABAC）
- AWS IAM Identity Center
- IAM Roles Anywhere
- IAM Identity Center 的 AWS 托管式策略
- AWS IAM 策略条件

- IAM 使用案例
- 移除不必要的凭证
- 使用 策略
- 如何根据 AWS 账户、OU 或组织来控制对 AWS 资源的访问
- 使用 AWS Secrets Manager 中的增强搜索来轻松标识、安排和管理密钥

相关视频：

- 在最多 60 分钟的时间内成为 IAM 策略高手
- 职责分离、最低权限、委托和 CI/CD
- 简化身份和访问管理以实施创新

SEC03-BP02 授予最低访问权限

仅授予用户在特定条件下对特定资源执行特定操作所需的访问权限。使用组和身份属性来大规模动态设置权限，而不是为单个用户定义权限。例如，您可以允许一组开发人员访问，以便仅管理其项目的资源。使用这种方法，如果某个开发人员离开项目，则可以自动撤销该开发人员的访问权限，而无需更改底层访问策略。

期望结果：用户仅拥有其特定工作职能所需的最低权限。可以使用单独的 AWS 账户来将开发人员与生产环境隔离开来。当开发人员需要访问生产环境以执行特定任务时，他们仅在这些任务期间被授予有限和受控的访问权限。在他们完成必要的工作后，他们的生产访问权限会被立即撤销。您可以定期审核权限，并在不再需要时立即撤销权限，例如当用户变更角色或离开组织时。您可以将管理员权限限制在一个小型、受信任的组中，以降低暴露的风险。您仅向计算机或系统账户授予执行其预期任务所需的最低权限。

常见反模式：

- 默认情况下，您向用户授予管理员权限。
- 您使用根用户账户进行日常活动。
- 您创建过于宽松的策略，而没有限定适当的范围。
- 您的权限审核不频繁，这会导致权限蔓延。
- 您完全依赖基于属性的访问权限控制来实现环境隔离或权限管理。

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

最低权限原则规定，只应允许身份执行完成特定任务所需的最小一组操作。这样可使可用性、效率和安全性达到平衡。根据此原则运营，有助于限制意外访问，还有助于跟踪谁有权访问哪些资源。默认情况下，IAM 用户和角色没有任何权限。默认情况下，根用户具有完全访问权限，应受到严格控制和监控，并且仅用于需要根访问权限的任务。

IAM 策略用于显式地为 IAM 角色或特定资源授予权限。例如，基于身份的策略可以附加到 IAM 组，而 S3 存储桶可由基于资源的策略控制。

创建 IAM 策略时，可以指定服务操作、资源以及为使 AWS 允许或拒绝访问而必须满足的条件。AWS 支持多种条件以协助您缩小访问权限范围。例如，通过使用 PrincipalOrgID 条件键，如果请求者不属于您的 AWS 组织，则您可以拒绝操作。

您还可以控制 AWS 服务代表您发出的请求，例如要求 AWS CloudFormation 创建一个 AWS Lambda 函数，方法是使用 CalledVia 条件键。您可以对不同的策略类型进行分层，以建立纵深防御并限制用户的总体权限。您还可以限制可以授予哪些权限以及在什么条件下授予权限。例如，可以让工作负载团队为他们所构建的系统创建自己的 IAM 策略，但前提是他们应用权限边界来限制他们可以授予的最大权限。

实施步骤

- 实施最低权限策略：向 IAM 组和角色分配具有最低权限的访问策略，以反映您所定义的用户的角色或职能。
- 通过单独的 AWS 账户隔离开发和生产环境：为开发和生产环境使用单独的 AWS 账户，并使用服务控制策略、资源策略和身份策略来控制它们之间的访问权限。
- 基于 API 使用情况制定策略：确定所需权限的一种方法是查看 AWS CloudTrail 日志。您可以使用此审核，根据用户在 AWS 内实际执行的操作来创建权限。IAM Access Analyzer 可以基于访问活动自动生成 IAM 策略。您可以在组织或账户级别，使用 IAM Access Advisor 来跟踪上次访问的关于某个特定策略的信息。
- 考虑使用工作职能的 AWS 托管式策略：当您开始创建细粒度的权限策略时，将 AWS 托管式策略用于常见的工作角色（如计费、数据库管理员和数据科学家）可能会有所帮助。这些策略有助于缩减用户具备的访问权限，同时您可以确定如何实施最低权限策略。
- 移除不必要的权限：检测并移除未使用的 IAM 实体、凭证和权限，以实现最低权限原则。可以使用 IAM Access Analyzer 来识别外部和未使用的访问权限，而 IAM Access Analyzer 策略生成有助于微调权限策略。

- 确保用户对生产环境的访问权限有限：用户应只能访问具有有效使用案例的生产环境。在用户执行需要生产访问权限的特定任务后，应撤销访问权限。限制对生产环境的访问可帮助防止发生影响生产的意外事件，并缩小意外访问的影响范围。
- 考虑权限边界：权限边界功能使用托管式策略，来设置基于身份的策略可向 IAM 实体授予的最大权限。实体的权限边界仅允许实体执行其基于身份的策略和权限边界同时允许的操作。
- 使用基于属性的访问权限控制和资源标签优化访问权限：使用资源标签的基于属性的访问权限控制（ABAC）可用于优化权限（如果支持）。可以使用将主体标签与资源标签进行比较的 ABAC 模型，根据您定义的自定义维度来优化访问权限。这种方法可以简化组织中的权限策略并减少其数量。
  - 建议仅当主体和资源均归您的 AWS 组织拥有时，才使用 ABAC 进行访问权限控制。外部各方可以为自己的主体和资源使用与您的组织相同的标签名称和值。如果您仅依靠这些名称/值对来授予对外部主体或资源的访问权限，则可能会提供意想不到的权限。
- 对 AWS Organizations 使用服务控制策略：Service control policies 集中控制您组织中成员账户的最大可用权限。重要的是，您可以使用服务控制策略来限制成员账户中的根用户权限。还要考虑使用 AWS Control Tower，它提供可充实 AWS Organizations 的规范性托管控制。您还可以在 Control Tower 内定义自己的控制。
- 为您的组织制定用户生命周期策略：用户生命周期策略定义了当用户加入 AWS、更改作业角色或范围或不再需要访问 AWS 时要执行的任务。在用户生命周期的每个步骤中执行权限审核，来验证权限受到适当限制并避免权限蔓延。
- 制定定期计划来审核权限并移除任何不需要的权限：您应该定期审核用户访问权限，以验证用户没有过于宽松的访问权限。AWS Config 和 IAM Access Analyzer 可以在审计用户权限时提供帮助。
- 建立工作角色矩阵：工作角色矩阵可直观显示您的 AWS 业务覆盖区域中所需的各种角色和访问级别。使用工作角色矩阵，您可以根据用户在组织内的职责来定义和分离权限。使用组，而不是将权限直接应用于各个用户或角色。

资源

相关文档：

- 授予最低权限
- IAM 实体的权限边界
- 用于编写最低权限 IAM 策略的方法
- IAM Access Analyzer makes it easier to implement least privilege permissions by generating IAM policies based on access activity
- 使用 IAM 权限边界将权限管理委派给开发人员
- 使用上次访问的信息优化权限

- IAM 策略类型及其使用时间
- 使用 IAM policy simulator 测试 IAM policy
- Guardrails in AWS Control Tower
- 零信任架构：AWS 视角
- 如何使用 CloudFormation StackSets 实施最低权限原则
- 基于属性的访问控制（ABAC）
- 查看用户活动以缩小策略范围
- 查看角色访问
- 使用标记来组织环境并推动问责制
- AWS 标记策略
- 为AWS资源添加标签

相关视频：

- 新一代权限管理
- 零信任：AWS 视角

SEC03-BP03 建立紧急访问流程

创建一个流程，便于在集中式身份提供程序偶尔出现问题时紧急访问您的工作负载。

必须针对可能导致紧急事件的不同故障模式设计流程。例如，在正常情况下，您的员工用户使用集中式身份提供程序联合到云端（SEC02-BP04）来管理其工作负载。但是，如果您的集中式身份提供程序出现故障，或者云中联合身份验证的配置被修改，则您的员工用户可能无法联合到云中。紧急访问流程允许授权管理员通过其他方式（例如其他联合形式或直接用户访问）访问云资源，以解决联合配置或工作负载的问题。在恢复正常的联合机制之前，将使用紧急访问流程。

期望结果：

- 您已经定义并记录了算是紧急情况的故障模式：考虑您的正常情况以及用户管理其工作负载所依赖的系统。考虑这些依赖项中的每一个在哪些情形下会发生故障并导致紧急情况。您可能会发现可靠性支柱中的问题和最佳实践有助于识别故障模式和构建更具韧性的系统，从而最大限度地降低发生故障的可能性。
- 您已记录了将故障确认为紧急情况所必须遵循的步骤。例如，您可以要求身份管理员检查主身份提供程序和备用身份提供程序的状态，如果两者均不可用，则宣布身份提供程序故障为紧急事件。

- 您已针对每种紧急情况或故障模式定义了紧急访问流程。应尽可能明确具体，这样可减少用户针对所有类型的紧急情况过度使用通用流程的倾向。紧急访问流程描述了每个流程的使用情形，以及哪些情况下不应使用该流程，并指出了可能适用的替代流程。
- 您的流程有详细的说明和行动手册，便于快速有效地遵循。请记住，对用户来说，紧急事件可能让人很煎熬，他们可能面临极大的时间压力，因此流程设计应尽可能简单。

常见反模式：

- 您没有详细记录并经过充分测试的紧急访问流程。您的用户没有为紧急情况做好准备，在出现紧急事件时遵循临时流程。
- 您的紧急访问流程依赖于与普通访问机制相同的系统（例如集中式身份提供程序）。这意味着，此类系统的故障可能会同时影响您的正常访问和紧急访问机制，并削弱您从故障中恢复的能力。
- 您的紧急访问流程被用于非紧急情况。例如，您的用户经常滥用紧急访问流程，因为他们发现直接进行更改比通过管道提交更改更容易。
- 您的紧急访问流程未生成足够的日志用于审核这些流程，或者没有监控日志以提醒可能存在的流程滥用。

建立此最佳实践的好处：

- 通过拥有记录详实且经过充分测试的紧急访问流程，您可以减少用户响应和解决紧急事件所花费的时间。这样可以缩短停机时间，提高您向客户提供的服务的可用性。
- 您可以跟踪每个紧急访问请求，检测未经授权企图对非紧急事件滥用该过程的行为，并发出警报。

在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

本节针对与部署在 AWS 上的工作负载相关的几种故障模式，提供创建紧急访问流程的指导，首先是适用于所有故障模式的通用指导，然后是不同故障模式类型的特定指导。

适用于所有故障模式的通用指南

在针对故障模式设计紧急访问流程时，请考虑以下几点：

- 记录流程的先决条件和假设：何时应使用该流程、何时不应使用该流程。它有助于详细说明故障模式并记录假设，例如其他相关系统的状态。例如，故障模式 2 的流程假设身份提供程序可用，但 AWS 上的配置已修改或已过期。

- 预先创建紧急访问流程所需的资源（SEC10-BP05）。例如，预先创建带有 IAM 用户和角色的紧急访问 AWS 账户，并在所有工作负载账户中创建跨账户 IAM 角色。这样可以验证在发生紧急事件时这些资源是否已准备就绪并且可用。通过预先创建资源，您就不必依赖 AWS 控制面板 API（用于创建和修改 AWS 资源），在紧急情况下，这些 API 可能不可用。此外，通过预先创建 IAM 资源，您也无需考虑由于最终的一致性问题而可能出现的延迟。
- 将紧急访问流程纳入事件管理计划（SEC10-BP02）。记录如何跟踪紧急事件并将其传达给组织中的其他人，例如同级团队、您的领导层，如果适用，还包括外部的客户和业务合作伙伴。
- 在现有的服务请求工作流程系统（如果有）中定义紧急访问请求流程。通常，此类工作流程系统允许您创建受理表来收集有关请求的信息，在工作流程的每个阶段跟踪请求，并添加自动和手动审批步骤。将每个请求与事件管理系统中所跟踪的相应紧急事件关联起来。通过拥有统一的紧急访问系统，您可以在单个系统中跟踪这些请求，分析使用趋势并改进流程。
- 确保您的紧急访问流程只能由授权用户启动，并且需要用户的同事或管理层（视情况而定）的批准。审批流程在工作时间内和工作时间之外应该能够有效运作。明确在主审批人抽不开身的情况下，如何允许辅助审批人审批请求，并沿管理链条上报，直至获得批准。
- 为紧急访问流程和机制实施稳健的日志记录、监控和警报机制。为所有成功和失败的紧急访问尝试生成详细的审计日志。将活动与事件管理系统中正在发生的紧急事件关联起来，并在预期时间段之外发生操作时发出警报，或者在正常运行期间使用紧急访问账户时发出警报。紧急访问账户仅限在紧急情况下访问，因为“打碎玻璃”程序可视为后门。与安全信息和事件管理（SIEM）工具或 AWS Security Hub 集成，来报告和审计紧急访问期间的所有活动。恢复正常操作后，自动轮换紧急访问凭证，并通知相关团队。
- 定期测试紧急访问流程，以确保步骤清楚明了，并快速高效地授予正确的访问级别。您的紧急访问流程应作为事件响应模拟（SEC10-BP07）和灾难恢复测试（REL13-BP03）的一部分进行测试。

故障模式 1：用于联合到 AWS 的身份提供程序不可用

如 SEC02-BP04 依赖集中式身份提供程序中所述，我们建议依靠集中式身份提供程序，来联合您的员工用户以授予对 AWS 账户的访问权限。您可以使用 IAM Identity Center 联合到 AWS 组织中的多个 AWS 账户，也可以使用 IAM 联合到单个 AWS 账户。在这两种情况下，员工用户都要先通过集中式身份提供程序进行身份验证，然后才会被重定向到 AWS 登录端点进行单点登录。

万一集中式身份提供程序不可用，员工用户就无法联合到 AWS 账户或管理其工作负载。在这种紧急情况下，您可以为一小部分管理员提供紧急访问流程，让他们访问 AWS 账户，来执行等不及集中式身份提供程序恢复正常的关键任务。例如，您的身份提供程序在 4 小时内不可用，在此期间，您需要修改生产账户中 Amazon EC2 Auto Scaling 组的上限，以应对客户流量意外激增的情况。您的紧急状况管理员应遵循紧急访问流程，以获得对特定生产 AWS 账户的访问权限并进行必要的更改。

紧急访问流程依赖于预先创建的紧急访问 AWS 账户，该账户仅用于紧急访问，并拥有 AWS 资源（如 IAM 角色和 IAM 用户）以支持紧急访问流程。在正常运营期间，任何人都不得访问紧急访问账户，而且您必须对滥用该账户的行为进行监控并发出警报（有关更多详情，请参阅前面的“通用指南”部分）。

紧急访问账户具有紧急访问 IAM 角色，有权在需要紧急访问的 AWS 账户中代入跨账户角色。这些 IAM 角色是预先创建的，并配置有信任策略，可信任应急账户的 IAM 角色。

紧急访问过程可以使用以下方法之一：

- 您可以在紧急访问账户中为紧急状况管理员预先创建一组 IAM 用户，并使用相关的强密码和 MFA 令牌。这些 IAM 用户有权代入 IAM 角色，然后在需要紧急访问时，允许跨账户访问 AWS 账户。我们建议尽可能少地创建此类用户，并将每个用户分配给一个紧急状况管理员。在紧急情况下，紧急状况管理员用户使用其密码和 MFA 令牌码登录紧急访问账户，切换到紧急账户中的紧急访问 IAM 角色，最后切换到工作负载账户中的紧急访问 IAM 角色，以执行紧急更改操作。这种方法的优点是，每个 IAM 用户都分配给一个紧急状况管理员，您可以通过查看 CloudTrail 事件来了解哪个用户已登录。缺点是，您必须维护多个 IAM 用户及其关联的长寿命密码和 MFA 令牌。
- 您可以使用紧急访问 AWS 账户根用户来登录紧急访问账户，代入用于紧急访问的 IAM 角色，并代入工作负载账户中的跨账户角色。建议为根用户设置一个强密码和多个 MFA 令牌。我们还建议将密码和 MFA 令牌存储在安全的企业凭证保管库中，该保管库可执行强身份验证和授权。您应确保密码和 MFA 令牌重置因素的安全：将账户的电子邮件地址设置为由云安全管理员监控的电子邮件分发列表，将账户的电话号码设置为同样由安全管理员监控的共享电话号码。这种方法的优点是只需管理一组根用户凭证。缺点是，由于这是共享用户，多个管理员都能以根用户身份登录。您必须审计企业保管库日志事件，以确定是哪位管理员查看了根用户密码。

故障模式 2：AWS 上的身份提供程序配置已修改或已过期

要允许您的员工用户联合到 AWS 账户，您可以使用外部身份提供程序配置 IAM Identity Center，或创建 IAM 身份提供程序（SEC02-BP04）。通常，您需要通过导入身份提供程序提供的 SAML 元数据 XML 文档，来配置这些服务。元数据 XML 文档包含一个 X.509 证书，该证书对应于身份提供程序用来签署其 SAML 断言的私钥。

管理员可能会错误地修改或删除 AWS 端的这些配置。在另一种情形下，导入到 AWS 的 X.509 证书可能会过期，而带有新证书的新元数据 XML 尚未导入到 AWS。这两种情形都可能使您的员工用户无法联合到 AWS，从而出现紧急情况。

在这种紧急情况下，您可以向您的身份管理员提供对 AWS 的访问权限以修复联合问题。例如，身份管理员使用紧急访问流程登录紧急访问 AWS 账户，切换到 Identity Center 管理员账户中的角色，并通过从身份提供程序导入最新的 SAML 元数据 XML 文档来更新外部身份提供程序配置，从而重新启用联合。修复联合后，您的员工用户将继续使用正常操作流程联合到其工作负载账户。

您可以按照前面的故障模式 1 中详述的方法来创建紧急访问流程。您可以向您的身份管理员授予最低访问权限，使其只能访问 Identity Center 管理员账户，并使用该账户对 Identity Center 执行操作。

故障模式 3：Identity Center 中断

如果发生 IAM Identity Center 或 AWS 区域中断这样的小概率事件，我们建议您设置一个可用于临时访问 AWS Management Console的配置。

紧急访问流程使用从身份提供程序到您的紧急账户中的 IAM 的直接联合。有关流程和设计注意事项的详细信息，请参阅《设置对 AWS Management Console的紧急访问》。

实施步骤

针对所有故障模式的通用步骤

- 创建专门用于紧急访问流程的 AWS 账户。预先创建账户中所需的 IAM 资源，例如 IAM 角色或 IAM 用户，以及可选的 IAM 身份提供程序。此外，在工作负载 AWS 账户中预先创建跨账户 IAM 角色，并与紧急访问账户中的相应 IAM 角色建立信任关系。您可以将 AWS CloudFormation StackSets 与 AWS Organizations 结合使用，在您组织的成员账户中创建此类资源。

- 创建 AWS Organizations 服务控制策略（SCP），以拒绝删除和修改成员 AWS 账户中的跨账户 IAM 角色。
- 对紧急访问 AWS 账户启用 CloudTrail，并将跟踪事件发送到日志收集 AWS 账户中的中央 S3 存储桶。如果您使用 AWS Control Tower 来设置和管理您的 AWS 多账户环境，则您使用 AWS Control Tower 创建或在 AWS Control Tower 中注册的每个账户默认情况下都已启用 CloudTrail，并发送到专用日志存档 AWS 账户中的 S3 存储桶。
- 通过创建 EventBridge 规则来匹配紧急 IAM 角色所执行的控制台登录和 API 活动，来监控紧急访问账户的活动。当事件管理系统中所跟踪的正在发生的紧急事件之外出现活动时，向安全运营中心发送通知。

针对“故障模式 1：用于联合到 AWS 的身份提供程序不可用”和“故障模式 2：AWS 上的身份提供程序配置已修改或已过期”的其他步骤

- 根据您选择的紧急访问机制，预先创建资源：
  - 使用 IAM 用户：使用强密码和关联的 MFA 设备预先创建 IAM 用户。
  - 使用紧急账户的根用户：为根用户配置一个强密码，并将该密码存储在您的企业凭证库中。将多个物理 MFA 设备与根用户关联，并将设备存放在紧急状况管理员团队成员可以快速访问的位置。

针对“故障模式 3：Identity Center 中断”的其他步骤

- 如设置对 AWS Management Console的紧急访问中所详述的那样，在紧急访问 AWS 账户中，创建 IAM 身份提供程序，以启用从身份提供程序的直接 SAML 联合。
- 在 IdP 中创建没有成员的紧急行动组。
- 在紧急访问账户中创建与紧急行动组相对应的 IAM 角色。

资源

相关的 Well-Architected 最佳实践：

- SEC02-BP04 依赖集中式身份提供程序
- SEC03-BP02 授予最低访问权限
- SEC10-BP02 制定事件管理计划
- SEC10-BP07 执行实际演练

相关文档：

- 设置对 AWS Management Console的紧急访问
- 使 SAML 2.0 联合用户能够访问 AWS Management Console
- Break glass access

相关视频：

- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

相关示例：

- AWS Break Glass 角色
- AWS 客户行动手册框架
- AWS 事件响应行动手册样本

SEC03-BP04 持续减少权限

当您的团队确定所需的访问权限时，删除不需要的权限，并建立审核流程以实现最低权限。持续监控并删除供人类和机器访问的未使用的身份和权限。

期望结果：权限策略应遵循最低权限原则。随着工作职责和角色变得更加明确，需要审查您的权限策略以删除不必要的权限。如果无意中泄露或未经授权访问凭证，这种方法会缩小影响范围。

常见反模式：

- 默认为向用户授予管理员权限。
- 创建过于宽松但没有完全管理员权限的策略。
- 保留不再需要的权限策略。

在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

当团队和项目刚刚起步时，可以使用宽松的权限策略来激发创新并提高敏捷性。例如，在开发或测试环境中，开发人员可以获得广泛的访问权限以使用各种 AWS 服务。我们建议您持续评估访问权限，并仅限于访问完成当前作业所必需的服务和服务操作。对于人类和机器身份，均建议进行此项评估。机器身份有时称为系统或服务账户，是让 AWS 访问应用程序或服务器的身份。这种访问权限在生产环境中尤其重要，因为在该环境中，过于宽松的权限会产生广泛的影响，并可能暴露客户数据。

AWS 提供多种方法来帮助识别未使用的用户、角色、权限和凭证。AWS 还可帮助分析 IAM 用户和角色（包括关联的访问密钥）的访问活动，以及对 AWS 资源（如 Amazon S3 存储桶中的对象）的访问。AWS Identity and Access Management Access Analyzer 策略生成可帮助您根据主体与之交互的实际服务和操作来创建限制性权限策略。基于属性的访问权限控制（ABAC）可以帮助简化权限管理，因为您可以使用其属性向用户提供权限，而不必将权限策略直接附加到每个用户。

实施步骤

- 使用 AWS Identity and Access Management Access Analyzer：IAM Access Analyzer 帮助您标识企业和账户中与外部实体共享的资源，例如 Amazon Simple Storage Service（Amazon S3）存储桶或 IAM 角色。
- 使用 IAM Access Analyzer 策略生成：IAM Access Analyzer 策略生成可帮助您基于 IAM 用户或角色的访问活动创建精细的权限策略。
- 在生产之前跨较低环境测试权限：首先，使用 less critical sandbox and development environments，通过 IAM Access Analyzer 测试各种工作职能所需的权限。然后，在将这些权限应用

于生产之前，在测试、质量保证和暂存环境中逐步收紧和验证这些权限。较低的环境最初可以拥有更宽松的权限，因为服务控制策略（SCP）通过限制所授予的最大权限来强制实施护栏。

- 确定 IAM 用户和角色的可接受时间范围和使用策略：使用上次访问时间戳可识别未使用的用户和角色并将它们移除。查看关于服务和操作的上次访问情况的信息，以确定和设置特定用户和角色的权限范围。例如，您可以使用关于上次访问情况的信息，以确定您的应用程序角色需要执行的特定 Amazon S3 操作，并只允许该角色访问这些操作。AWS Management Console中提供了上次获取的信息功能，您也可以对这些功能进行编程，以便将它们整合到您的基础架构工作流程和自动化工具中。
- 考虑在 AWS CloudTrail 中记录数据事件：默认情况下，CloudTrail 不会记录 Amazon S3 对象级活动（例如 `GetObject` 和 `DeleteObject`）或 Amazon DynamoDB 表活动（例如 `PutItem` 和 `DeleteItem`）等数据事件。考虑对这些事件使用日志记录，以确定哪些用户和角色需要访问特定的 Amazon S3 对象或 DynamoDB 表项目。

资源

相关文档：

- 授予最低权限
- 移除不必要的凭证
- 什么是 AWS CloudTrail？
- 使用 策略
- 记录和监控 DynamoDB
- 对 Amazon S3 存储桶和对象使用 CloudTrail 事件日志记录
- 获取您的 AWS 账户 的凭证报告

相关视频：

- 在最多 60 分钟的时间内成为 IAM 策略高手
- 职责分离、最低权限、委托和 CI/CD
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

SEC03-BP05 为您的组织定义权限防护机制

使用权限防护机制来减少可向主体授予的可用权限的范围。权限策略评估链包括您的防护机制，用于在做出授权决策时确定主体的有效权限。 您可以使用基于层的方法定义防护机制。在整个企业内广泛地应用一些防护机制，而对临时访问会话则以细粒度方式应用另一些防护机制。

期望结果：您可以使用单独的 AWS 账户对环境进行明确隔离。 使用服务控制策略（SCP）定义整个企业内的权限防护机制。在更靠近组织根级别的层次结构上设置较为广泛的防护机制，而在更靠近单独账户的级别上设置更严格的防护机制。

在支持资源策略的情况下，使用资源策略定义主体获得资源访问权限必须满足的条件。在适用的情况下，还应使用资源策略缩小允许操作的范围。在管理工作负载权限的主体上设定了权限边界，将权限管理委托给单独的工作负载负责人。

常见反模式：

- 在 AWS 组织内创建成员 AWS 账户，但不使用 SCP 来限制其根凭证的使用和可用权限。
- 根据最低权限原则分配了权限，但没有对可以授予的最大权限集施加防护机制。
- 依靠 AWS IAM 的隐式拒绝基础来限制权限，相信策略不会授予非预期的显式允许权限。
- 在同一个 AWS 账户中运行多个工作负载环境，然后依靠 VPC、标签或资源策略等机制来强制实施权限边界。

建立此最佳实践的好处：权限防护机制有助于建立人们对无法授予不需要的权限的信心，即使权限策略尝试这样做也是如此。 这样便可以减少需要考虑的最大权限范围，从而简化权限的定义和管理。

在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

我们建议您使用基于层的方法，为企业定义权限防护机制。在应用了额外的层时，这种方法可以系统性地减少可能的最大权限集。利用这种方法，您可以根据最低权限原则授予访问权限，从而减少由于策略配置错误而导致意外访问的风险。

建立权限防护机制的第一步是将您的工作负载和环境隔离到单独的 AWS 账户中。如果没有显式权限，一个账户的主体无法访问另一个账户中的资源，即使两个账户位于同一 AWS 组织或在同一组织单位（OU）下也是如此。您可以使用 OU 将要管理的账户分组为一个单位。

接下来，您需要减少可向企业成员账户中的主体授予的最大权限集。为此，您可以使用服务控制策略（SCP），您可以将其应用于 OU 或账户。SCP 可以强制执行常见的访问控制措施，例如限制对特定

AWS 区域的访问，防止资源被删除，或者禁用存在潜在风险的服务操作。您应用到组织根的 SCP 仅影响其成员账户，不影响管理账户。 SCP 仅管理企业中的主体。您的 SCP 不管理企业外部访问您资源的主体。

如果您使用的是 AWS Control Tower，则可以利用其 controls 和 landing zones 作为权限护栏和多账户环境的基础。登录区提供了一个预先配置的安全基线环境，为不同的工作负载和应用程序提供了单独的账户。这些护栏通过结合使用服务控制策略（SCP）、AWS Config 规则和其它配置，围绕安全性、运营和合规性实施强制性控制措施。但是，在使用 Control Tower 护栏和登录区以及自定义组织 SCP 时，请遵循 AWS 文档中概述的最佳实践来避免冲突并确保适当的治理，这一点至关重要。有关在 Control Tower 环境中管理 SCP、账户和组织单位（OU）的详细建议，请参阅 AWS Control Tower guidance for AWS Organizations。

通过遵守这些准则，可以有效地利用 Control Tower 的护栏、登录区和自定义 SCP，同时缓解潜在冲突并确保对多账户 AWS 环境进行适当的治理和控制。

下一步是使用 IAM 资源策略来限定您可以对其所管理的资源采取的可用操作的范围，以及代理主体必须满足的任何条件。这一点可以很宽泛，只要主体是您的组织的一部分，就可以允许执行所有操作（使用 PrincipalOrgId 条件键），也可以精细到仅允许特定 IAM 角色执行特定操作。对于 IAM 角色信任策略中的条件，您可以采用类似的方法。 如果资源或角色信任策略在所管理的角色或资源的同一个账户中，明确指定了主体，则该主体不需要附加授予相同权限的 IAM 策略。 如果主体与资源位于不同的账户中，则主体就需要附加 IAM 策略来授予这些权限。

通常，工作负载团队需要管理其工作负载所需的权限。 这可能要求团队创建新的 IAM 角色和权限策略。 您可以在 IAM 权限边界内获取允许团队授予的最大权限范围，并将此文档关联到一个 IAM 角色，然后团队可以使用该角色来管理其 IAM 角色和权限。 通过这种方法，团队能够灵活地完成其工作，同时降低拥有 IAM 管理访问权限的风险。

更精细的步骤是实施特权访问管理（PAM）和临时提升的访问权限管理（TEAM）技术。 PAM 的一个例子是要求主体在采取特权操作之前进行多重身份验证。 有关更多信息，请参阅《配置受 MFA 保护的 API 访问》。TEAM 需要一种解决方案，来管理主体在获得提升访问权限时需经过的审批，以及允许获得提升访问权限的时间范围。 一种方法是将主体临时添加到具有更高访问权限的 IAM 角色的角色信任策略中。 另一种方法是，在正常运行情况下，使用会话策略缩小 IAM 角色向主体授予的权限范围，然后在批准的时段内暂时取消此限制。要了解有关 AWS 和精选合作伙伴验证的解决方案的更多信息，请参阅《临时提升访问权限》。

实施步骤

1. 将您的工作负载和环境存放在单独的 AWS 账户中。
2. 使用 SCP 来减少可向企业成员账户中的主体授予的最大权限集。

a. 在定义 SCP 来减少可向组织成员账户中的主体授予的最大权限集时，您可以选择允许列表 或拒绝列表 方法。允许列表策略显式指定允许的访问权限，并隐式阻止所有其它访问权限。拒绝列表策略显式指定不允许的访问权限，并且默认情况下允许所有其它访问权限。这两种策略都有其优势和权衡，适当的选择取决于您组织的具体要求和风险模型。有关更多详细信息，请参阅 Strategy for using SCPs。

b. 此外，请查看 service control policy examples，来了解如何有效地构造 SCP。

3. 使用 IAM 资源策略缩小范围，并指定允许对资源执行操作的条件。 使用 IAM 角色信任策略中的条件来创建对代入角色的限制。
4. 将 IAM 权限边界分配给 IAM 角色，然后工作负载团队可以使用该角色来管理自己的工作负载的 IAM 角色和权限。
5. 根据您的需求评估 PAM 和 TEAM 解决方案。

资源

相关文档：

- AWS 上的数据边界
- 使用数据边界建立权限防护机制
- 策略评估逻辑

相关示例：

- 服务控制策略示例

相关工具：

- AWS 解决方案：临时提升的访问权限管理
- 经过验证的 TEAM 安全合作伙伴解决方案

SEC03-BP06 基于生命周期管理访问权限

在企业内主体（用户、角色和群组）的整个生命周期中，监控和调整授予主体的权限。在用户更改角色时调整组成员资格，并在用户离开企业时移除访问权限。

期望结果：您可以在企业内主体的整个生命周期中监控和调整权限，从而降低不必要权限的风险。在创建用户时授予合适的访问权限。随着用户职责的变化，您会修改访问权限，当用户不再活跃或已离开企

业时，您会删除访问权限。您集中管理用户、角色和组的更改。您使用自动化方法将更改传播到 AWS 环境中。

常见反模式：

- 预先向身份授予过多或宽泛的访问权限，这些权限超出了最初的需求。
- 随着身份的角色和职责随着时间推移而发生变化，您未审核和调整访问权限。
- 让无效或离职的身份保留有效的访问权限。这样会增加未经授权访问的风险。
- 没有利用自动化功能来管理身份的生命周期。

在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

在身份（例如用户、角色、组）的整个生命周期中，谨慎管理和调整授予身份的访问权限。这一生命周期包括初始入职阶段、角色和职责的持续变化以及最终的离职或终止。根据生命周期的阶段主动管理访问权限，维护正确的访问权限级别。遵守最低权限原则，减少授予过多或不必要访问权限的风险。

您可以直接在 AWS 账户中管理 IAM 用户的生命周期，也可以通过从员工身份提供者到 AWS IAM Identity Center 的联合身份验证来进行管理。对于 IAM 用户，您可以在 AWS 账户中创建、修改和删除用户及其关联权限。对于联合用户，您可以使用 IAM Identity Center，通过 System for Cross-domain Identity Management（SCIM）协议，从组织的身份提供者同步用户和组信息，从而管理这些用户的生命周期。

SCIM 是一种开放标准协议，用于跨不同系统自动预置和取消预置用户身份。通过使用 SCIM 将身份提供者与 IAM Identity Center 集成，您可以自动同步用户和组信息，这有助于验证访问权限的授予、修改或撤销，而这些操作是基于企业中权威身份来源中的更改而进行的。

随着企业内员工角色和职责的变化，相应地调整员工的访问权限。您可以使用 IAM Identity Center 的权限集来定义不同的工作角色或职责，并将其关联到相应的 IAM 策略和权限。当员工的角色发生变化时，您可以更新向他们分配的权限集以反映他们的新职责。验证他们是否具有必要的访问权限，同时还遵守了最低权限原则。

实施步骤

1. 定义并记录访问权限管理生命周期流程，包括授予初始访问权限、定期审查和离职的程序。
2. 实施 IAM 角色、组和权限边界，以便从整体上管理访问权限并实施允许的最高访问权限级别。
3. 使用 IAM Identity Center，与联合身份提供者（例如 Microsoft Active Directory、Okta、Ping Identity）集成，作为用户和组信息的权威来源。

4. 使用 SCIM 协议，来将用户和组信息从身份提供者同步到 IAM Identity Center 的身份存储中。
5. 在 IAM Identity Center 中创建权限集，用于表示组织中的不同工作角色或职责。为每个权限集定义相应的 IAM 策略和权限。
6. 实施定期的访问权限审查，及时撤销访问权限，并持续改进访问权限管理生命周期流程。
7. 向员工提供访问权限管理最佳实践的培训，增强员工的意识。

资源

相关最佳实践：

- SEC02-BP04 依赖集中式身份提供程序

相关文档：

- 管理您的身份源
- 在 IAM Identity Center 中管理身份
- 使用 AWS Identity and Access Management Access Analyzer
- IAM Access Analyzer 策略生成

相关视频：

- AWS re:Inforce 2023 - Manage temporary elevated access with AWS IAM Identity Center
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2022 - Harness power of IAM policies & rein in permissions w/Access Analyzer

SEC03-BP07 分析公共和跨账户访问

持续监控重点关注公共访问和跨账户访问的调查发现。将公共访问和跨账户访问限制为仅限需要此访问的特定资源。

期望结果：了解您的哪些 AWS 资源是共享的，以及与谁共享。持续监控和审计您的共享资源，以验证它们仅与授权的主体共享。

常见反模式：

- 不保留共享资源的清单。

- 跨账户访问或公开访问资源时，没有遵循流程。

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

如果您的账户在 AWS Organizations 中，您可以向整个组织、特定组织单位或个人账户授予资源访问权限。如果您的账户不是某个组织的成员，您可以与个人账户共享资源。您可以使用基于资源的策略[例如 Amazon Simple Storage Service（Amazon S3）存储桶策略] 或通过允许其他账户中的主体代入您账户中的 IAM 角色来授予直接跨账户访问权限。使用资源策略时，请验证访问权限是否仅授予给经过授权的主体。建立一个流程来审批所有需要可公开访问的资源。

AWS Identity and Access Management Access Analyzer 使用可证明安全性来标识从账户的外部访问某个资源时的所有访问路径。它持续审核资源策略，并报告公开访问和跨账户访问的调查发现，以使您能够轻松分析可能非常宽泛的访问权限。请考虑将 IAM Access Analyzer 与 AWS Organizations 一起配置，以验证您是否能够查看所有账户。IAM Access Analyzer 还允许您在部署资源权限之前预览调查发现。这样，您便可以验证策略更改仅按照意图，授权对您资源的公共和跨账户访问。在设计多账户访问权限时，您可以使用信任策略来控制在何种情况下可代入某个角色。例如，您可以使用 `PrincipalOrgId` 条件键拒绝尝试从您的 AWS Organizations 之外代入角色。

AWS Config 可以报告配置错误的资源，并可以通过 AWS Config 策略检查来检测配置了公有访问权限的资源。诸如 AWS Control Tower 和 AWS Security Hub 等服务简化了跨 AWS Organizations 的侦测性控制和防护机制的部署，可以识别并修复公开暴露的资源。例如，AWS Control Tower 具有托管防护机制，可以检测是否有任何可由 AWS 账户恢复的 Amazon EBS 快照。

实施步骤

- 考虑将 AWS Config 用于 AWS Organizations：AWS Config 允许您将 AWS Organizations 中多个账户的结果聚合到委派管理员账户中。这样可提供全面的视图，并允许您跨账户部署 AWS Config 规则以检测可公开访问的资源。
- 配置 AWS Identity and Access Management Access Analyzer：IAM Access Analyzer 有助于您识别组织和账户中与外部实体共享的资源，例如 Amazon S3 存储桶或 IAM 角色。
- 在 AWS Config 中使用自动修复来响应 Amazon S3 存储桶的公共访问配置中的更改：您可以自动启用 Amazon S3 存储桶的阻止公共访问设置。
- 实施监控和警报，以确定 Amazon S3 存储桶是否已变为公共：您必须具备监控或警报机制，以便确定 Amazon S3 屏蔽公共访问权限何时关闭，以及 Amazon S3 存储桶是否变为公共。此外，如果您正在使用 AWS Organizations，您可以创建服务控制策略，以阻止对 Amazon S3 公共访问策略进行

更改。AWS Trusted Advisor 可对具有开放访问权限的 Amazon S3 存储桶进行检查。如果向每个人授予“上传/删除”权限，那么任何人都可以向存储桶添加项目或者修改或删除存储桶中的项目，这样会产生潜在的安全问题。Trusted Advisor 检查可检查显式存储桶权限，以及可能覆盖存储桶权限的关联存储桶策略。您也可以使用 AWS Config 来监控 Amazon S3 存储桶是否具有公共访问权限。有关更多信息，请参阅《如何使用 AWS Config 来监控和响应允许公共访问的 Amazon S3 存储桶》。

在审核 Amazon S3 存储桶的访问控制时，务必考虑存储在存储桶中的数据的性质。Amazon Macie 是一项旨在协助您发现和保护敏感数据的服务，如个人身份信息（PII）、受保护健康信息（PHI）以及诸如私有密钥或 AWS 访问密钥等凭证。

资源

相关文档：

- 使用 AWS Identity and Access Management Access Analyzer
- AWS Control Tower 控件库
- AWS 基础安全最佳实践标准
- AWS Config 托管规则
- AWS Trusted Advisor 检查引用
- 通过 Amazon EventBridge 监控 AWS Trusted Advisor 的检查结果
- 管理组织内所有账户的 AWS Config 规则
- AWS Config 和 AWS Organizations
- 将您的 AMI 设为可在 Amazon EC2 中公开使用

相关视频：

- 保护多账户环境的最佳实践
- 深入了解 IAM Access Analyzer

SEC03-BP08 在组织内安全地共享资源

随着工作负载数量的增长，您可能需要共享对这些工作负载中资源的访问权限，或者跨多个账户多次预置资源。您可能需要进行构造来划分环境，例如划分成开发、测试和生产环境。但是，采取相互分离的构造并不会限制您安全共享权限。通过共享重叠的组件，您可以降低运维开销，并提供一致的体验，而不必猜测在多次创建同一资源时可能遗漏了什么。

期望结果：通过使用安全的方法在组织内共享资源，最大限度地减少意外访问，并帮助实施数据丢失防护计划。与管理单个组件相比，降低了运维开销，减少了多次手动创建同一组件时引起的错误，并提高了工作负载的可扩展性。您可以在多点故障场景中缩短问题解决时间，并在确定何时不再需要某个组件时更有信心。有关分析外部共享资源的规范性指南，请参阅《SEC03-BP07 分析公共和跨账户访问》。

常见反模式：

- 缺少对意外的外部共享进行持续监控和自动发出警报的流程。
- 缺乏关于应分享什么和不应分享什么的基准。
- 默认采用广泛的开放政策，而不是在需要时明确地分享。
- 手动创建在需要时重叠的基础资源。

在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

设计您的访问控制和模式，以安全地管理共享资源的使用，并且仅与可信实体共享。监控共享资源，持续检查共享资源访问权限，并在不适当或意外共享时发出警报。查看分析公共和跨账户访问，以帮助您建立治理，减少仅对需要访问的资源的外部访问权限，并建立持续监控和自动警报的流程。

AWS Organizations 内的跨账户共享受多个 AWS 服务支持，比如 AWS Security Hub、Amazon GuardDuty 和 AWS Backup。这些服务允许将数据共享到中心账户，可从中心账户访问，或从中心账户管理资源和数据。例如，AWS Security Hub 可将调查发现从个人账户转移到中心账户，在那里您可以查看所有调查发现。AWS Backup 可以对资源进行备份并在多个账户之间共享。可以使用 AWS Resource Access Manager（AWS RAM）共享其它公用资源，如 VPC subnets and Transit Gateway attachments、AWS Network Firewall 或 Amazon SageMaker AI pipelines。

要限制您的账户只能在组织内共享资源，请使用服务控制策略（SCP）来防止向外部主体授予访问权限。共享资源时，将基于身份的控制与网络控制结合起来，为组织创建数据边界，以帮助防止意外的访问。数据边界是一组预防性防护机制，用于帮助验证是否只有您的可信身份才能访问预期网络中的可信资源。这些控制措施会施加适当的限制，确定哪些资源可以共享，并防止共享或暴露不应被外泄的资源。例如，作为数据边界的一部分，您可以使用 VPC 端点策略和 `AWS:PrincipalOrgId` 条件，确保访问您的 Amazon S3 存储桶的身份属于您的组织。请务必注意，SCP 不适用于服务相关角色或 AWS 服务主体。

使用 Amazon S3 时，请关闭您的 Amazon S3 存储桶的 ACL，并使用 IAM 策略来定义访问控制。为了从 Amazon CloudFront 限制对 Amazon S3 来源的访问，请从来源访问身份（OAI）迁移到来源访问控制（OAC），后者支持使用其他功能，包括使用 AWS Key Management Service 进行服务器端加密。

在某些情况下，您可能希望允许在组织外部共享资源，或授予第三方访问您资源的权限。有关管理外部共享资源的权限的规范性指南，请参阅《权限管理》。

实施步骤

1. 使用 AWS Organizations：AWS Organizations 是一项账户管理服务，使您可以将多个 AWS 账户整合到您所创建的组织中并集中进行管理。您可以将账户分组为组织单位（OU），并将不同的策略附加到每个 OU，以帮助您满足预算、安全性和合规性需求。您还可以控制 AWS 人工智能（AI）和机器学习（ML）服务收集和存储数据的方式，并使用与 Organizations 集成的 AWS 服务的多账户管理。
2. 将 AWS Organizations 与 AWS 服务集成：当您使用 AWS 服务在组织的成员账户中代表您执行任务时，AWS Organizations 会在每个成员账户中为该服务创建一个 IAM 服务相关角色（SLR）。您应使用 AWS Management Console、AWS API 或 AWS CLI 管理可信访问。有关开启可信访问权限的规范性指南，请参阅《将 AWS Organizations 与其它 AWS 服务结合使用》和《可与 Organizations 一起使用的 AWS 服务》。
3. 建立数据边界：数据边界提供了明确的信任和所有权边界。在 AWS 上，它通常表示为由 AWS Organizations 管理的 AWS 组织，以及访问您的 AWS 资源的任何本地网络或系统。建立数据边界的目标，是验证如果身份可信、资源可信并且网络符合预期，则支持进行访问。然而，建立数据边界并不是一个放之四海皆准的方法。根据您的特定安全风险模型和要求，评估并采纳 Building a Perimeter on AWS whitepaper 中概述的控制目标。您应仔细考虑自己独特的风险状况，并实施符合您安全需求的边界控制措施。
4. 在 AWS 服务中使用资源共享并相应进行限制：许多 AWS 服务支持您与另一账户共享资源，或以另一账户中的资源为目标，比如亚马逊机器映像（AMI）和 AWS Resource Access Manager（AWS RAM）。限制 `ModifyImageAttribute` API 以指定可信账户，从而与之共享 AMI。当需要使用 AWS RAM 来将共享限制于您的组织内部时，请指定 `ram:RequestedAllowsExternalPrincipals` 条件，以帮助防止来自不可信身份的访问。有关规范性指南和注意事项，请参阅《资源共享和外部目标》。
5. 使用 AWS RAM 在一个账户中或与其它 AWS 账户安全共享：AWS RAM 有助于您与账户中的角色和用户以及与其它 AWS 账户安全地共享已创建的资源。在多账户环境中，AWS RAM 使您能够一次性创建资源并与其他账户共享。这种方法有助于降低运维开销，同时通过与 Amazon CloudWatch 和 AWS CloudTrail 的集成提供一致性、可见性和可审计性，使用跨账户访问时无法获得这些好处。

   如果您拥有以前使用基于资源的策略共享的资源，则可以使用 `PromoteResourceShareCreatedFromPolicy API` 或等效 API 将资源共享升级为完全 AWS RAM 资源共享。

在某些情况下，您可能需要采取其他步骤来共享资源。例如，要共享加密快照，您需要共享 AWS KMS 密钥。

资源

相关最佳实践：

- SEC03-BP07 分析公共和跨账户访问
- SEC03-BP09 与第三方安全地共享资源
- SEC05-BP01 创建网络层

相关文档：

- 存储桶拥有者向并非其拥有的对象授予跨账户权限
- 如何将信任策略与 IAM 结合使用
- 在 AWS 上构建数据边界
- 如何在向第三方授予对 AWS 资源的访问权限时使用外部 ID
- 可与 AWS Organizations 一起使用的 AWS 服务
- 在 AWS 上建立数据边界：仅允许可信身份获取公司数据

相关视频：

- 使用 AWS Resource Access Manager 实现精细访问
- 使用 VPC 端点保护您的数据边界
- 在 AWS 上建立数据边界

相关工具：

- 数据边界策略示例

SEC03-BP09 与第三方安全地共享资源

云环境的安全性不仅仅限于您的组织。您的组织有一部分数据可能要依赖第三方来管理。管理第三方托管系统的权限，应遵循及时访问的做法，使用最低权限原则和临时凭证。通过与第三方密切合作，您既可以缩小影响范围，又可以降低意外访问的风险。

期望结果：您避免使用长期 AWS Identity and Access Management（IAM）凭证，例如访问密钥和私密密钥，因为如果滥用，它们会构成安全风险。相反，可以使用 IAM 角色和临时凭证来改善您的安全状况，并最大限度地减少管理长期凭证的运营开销。在向第三方授予访问权限时，请在 IAM 信任策略中将通用唯一标识符（UUID）用作外部 ID，并将附加到角色的 IAM 策略置于您的控制之下，来确保最低权限访问。有关分析外部共享资源的规范性指南，请参阅 SEC03-BP07 分析公共和跨账户访问。

常见反模式：

- 采用默认的 IAM 信任策略，不附加任何条件。
- 使用长期 IAM 凭证和访问密钥。
- 重用外部 ID。

在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

您可能会希望允许在 AWS Organizations 外部共享资源，或授予第三方访问您账户的权限。例如，第三方提供的监控解决方案可能会需要访问您账户内部的资源。在这些情况下，请创建 IAM 跨账户角色，并仅向该角色提供第三方所需的权限。此外，使用外部 ID 条件定义信任策略。使用外部 ID 时，您或第三方可以为每个客户、第三方或租赁生成唯一 ID。创建唯一 ID 后，不应由除您之外的任何人控制它。第三方必须实施具体流程，以一种安全、可审计且可复制的方式将外部 ID 与客户关联起来。

您也可以使用 IAM Roles Anywhere 来管理 AWS 之外使用 AWS API 的应用程序的 IAM 角色。

如果第三方不再需要访问您的环境，则删除该角色。应避免向第三方提供长期凭证。了解其它支持共享的 AWS 服务，例如 AWS Well-Architected Tool 支持与其它 AWS 账户共享工作负载，而 AWS Resource Access Manager 有助于您安全地与其它账户共享您拥有的 AWS 资源。

实施步骤

1. 使用跨账户角色提供对外部账户的访问。跨账户角色可减少外部账户和第三方为服务客户而存储的敏感信息量。跨账户角色可让您将账户中 AWS 资源的访问权限安全地授予第三方（如 AWS 合作伙伴或组织内的其它账户），同时保持管理和审计该访问权限的能力。第三方可能从混合基础设施向您提供服务，或者将数据提取到一个异地位置。IAM Roles Anywhere 有助于您使第三方工作负载能够安全地与 AWS 工作负载交互，并进一步减少对长期凭证的需求。

   不应使用长期凭证或与用户关联的访问密钥来提供外部账户访问权限。而应使用跨账户角色来提供跨账户访问。

2. 进行尽职调查并确保第三方 SaaS 提供商的安全访问。当与第三方 SaaS 提供商共享资源时，请进行彻底的尽职调查，来确保他们采用安全和负责任的方法访问您的 AWS 资源。评估他们的责任共担模式，来了解他们提供了哪些安全措施以及哪些方面属于您的责任。确保 SaaS 提供商采用安全且可审计的流程来访问您的资源，包括使用外部 ID 和最低权限访问原则。使用外部 ID 有助于解决 confused deputy problem。

   实施安全控制以确保安全访问，并在向第三方 SaaS 提供商授予访问权限时遵守最低权限原则。这可能包括使用外部 ID、通用唯一标识符（UUID）和 IAM 信任策略，从而将访问权限限制在严格必要的范围内。与 SaaS 提供商密切合作，以便建立安全访问机制，定期审查他们对您的 AWS 资源的访问权限，并进行审计以确保符合安全要求。

3. 弃用客户提供的长期凭证。弃用长期凭证，使用跨账户角色或 IAM Roles Anywhere。如果必须使用长期凭证，请制定相应计划，逐渐转变成基于角色进行访问。有关管理密钥的详细信息，请参阅身份管理。此外，与 AWS 账户团队和第三方合作来建立风险缓解运行手册。有关应对和缓解安全事件潜在影响的规范性指南，请参阅事件响应。

4. 验证设置是否具有规范性指南，或是否实现了自动化。外部 ID 不视为密钥，但外部 ID 不能是容易猜测的值，例如电话号码、姓名或账户 ID。将外部 ID 设置为只读字段，这样就无法为了冒充设置而更改外部 ID。

   您或第三方可以生成外部 ID。定义一个流程，确定谁负责生成 ID。无论创建外部 ID 的实体是什么，第三方都必须确保客户之间的唯一性和格式一致。

   为您账户中的跨账户访问创建的策略必须遵循最低权限原则。第三方必须为您提供使用 AWS CloudFormation 模板或等效模板的角色策略文档或自动化设置机制。这减少了手动创建策略时出错的机会，并提供了可审计的跟踪。有关使用 AWS CloudFormation 模板来创建跨账户角色的更多信息，请参阅 Cross-Account Roles。

   第三方应提供一个自动化的、可审计的设置机制。但是，通过使用角色策略文档（此文档大致列出了所需的访问权限），角色设置的自动化应该由您来完成。使用 AWS CloudFormation 模板或等效模板，您应将偏差检测纳入审计实践来监控变更。

5. 对变更做出解释。您的账户结构、您对第三方的需求或他们提供的服务可能会发生变更。您应预料到可能会发生变动和失败，并进行相应的规划：请安排合适的人员，建立适当的流程并采用正确的技术进行应对。应定期审计您提供的访问级别，并实施检测方法，以便在发生意外变更时向您发出警报。监控并审计角色的使用情况，以及外部 ID 的数据存储状态。若发生意外变更或存在不当访问模式，您应准备暂时或永久撤销第三方访问权限。此外，还要衡量撤销操作造成的影响，包括执行该操作所需的时间、涉及的人员、成本以及对其他资源的影响。

   有关检测方法的规范性指南，请参阅《检测最佳实践》。

资源

相关最佳实践：

- SEC02-BP02 使用临时凭证
- SEC03-BP05 为您的组织定义权限护栏
- SEC03-BP06 基于生命周期管理访问权限
- SEC03-BP07 分析公共和跨账户访问
- SEC04 检测

相关文档：

- 存储桶拥有者向并非其拥有的对象授予跨账户权限
- How to use trust policies with IAM roles
- 使用 IAM 角色委托跨 AWS 账户的访问权限
- 如何使用 IAM 访问其它 AWS 账户中的资源？
- IAM 安全最佳实操
- 跨账户策略评估逻辑
- 如何在向第三方授予对 AWS 资源的访问权限时使用外部 ID
- Collecting Information from AWS CloudFormation Resources Created in External Accounts with Custom Resources
- Securely Using External ID for Accessing AWS Accounts Owned by Others
- Extend IAM roles to workloads outside of IAM with IAM Roles Anywhere

相关视频：

- How do I allow users or roles in a separate AWS 账户 access to my AWS 账户?
- AWS re:Invent 2018: Become an IAM Policy Master in 60 Minutes or Less
- AWS Knowledge Center Live: IAM Best Practices and Design Decisions

相关示例：

- 配置对 Amazon DynamoDB 的跨账户访问
- AWS STS Network Query Tool

检测

问题

- SEC 4. 您如何检测和调查安全事件？

SEC 4. 您如何检测和调查安全事件？

从日志和指标中捕获和分析事件以获得可见性。对安全事件和潜在威胁采取行动，从而保护您的工作负载。

最佳实践

- SEC04-BP01 配置服务和应用程序日志记录
- SEC04-BP02 在标准化位置收集日志、调查发现和指标
- SEC04-BP03 关联和扩充安全警报
- SEC04-BP04 启动对不合规资源的修复

SEC04-BP01 配置服务和应用程序日志记录

保留服务和应用程序的安全事件日志。这是审计、调查和运营使用案例的基本安全原则，也是由监管、风险与合规性（GRC，Governance, Risk, and Compliance）标准、政策和程序驱动的共同安全要求。

期望结果：当需要履行内部流程或义务（如安全事件响应）时，组织应能够及时、可靠且一致地从 AWS 服务和应用程序中检索安全事件日志。考虑将日志集中起来，以取得更好的运营成果。

常见反模式：

- 日志被永久存储或过早删除。
- 每个人都可以访问日志。
- 完全依赖手动流程进行日志治理和使用。
- 存储每一种类型的日志，以备不时之需。
- 仅在必要时检查日志完整性。

建立此最佳实践的好处：为安全事件实施根本原因分析（RCA）机制，并为您的监管、风险与合规性义务提供证据来源。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在安全调查或基于要求的其他使用案例期间，您需要能够查看相关日志，以记录并了解事件的来龙去脉和时间线。警报生成也需要日志，以指示发生了某些感兴趣的操作。选择、启用、存储和设置查询、检索机制以及警报至关重要。

实施步骤

- 选择并启用日志源。进行安全调查之前，您需要捕获相关日志，以便以回溯方式重建 AWS 账户中的活动。选择与工作负载相关的日志源。

  日志源的选择标准应基于业务所需的使用案例。使用 AWS CloudTrail 或 AWS Organizations 跟踪为每个 AWS 账户 建立跟踪，并为其配置 Amazon S3 存储桶。

  AWS CloudTrail 是一项日志记录服务，可跟踪针对 AWS 账户捕获 AWS 服务活动所进行的 API 调用。它默认情况下启用，管理事件保留 90 天，可以使用 AWS Management Console、AWS CLI 或 AWS SDK 通过 CloudTrail 事件历史记录检索这些事件。为了更长久地保留和了解数据事件，请创建 CloudTrail 跟踪并将其与 Amazon S3 存储桶关联，也可以选择与 Amazon CloudWatch 日志组关联。或者，您可以创建 CloudTrail Lake，这可保留 CloudTrail 日志长达七年之久，并提供基于 SQL 的查询工具

  AWS 建议使用 VPC 的客户分别使用 VPC 流日志和 Amazon Route 53 Resolver 查询日志启用网络流量和 DNS 日志，并将其流式传输到 Amazon S3 存储桶或 CloudWatch 日志组。您可以为 VPC、子网或网络接口创建 VPC 流日志。对于 VPC 流日志，您可以选择使用流日志的方式和位置，以降低成本。

  AWS CloudTrail 日志、VPC 流日志和 Route 53 解析器查询日志是支持 AWS 中安全调查的基本日志记录源。您还可以使用 Amazon Security Lake 以 Apache Parquet 格式和开放网络安全架构框架（OCSF）收集、标准化和存储这些日志数据，以便于查询。安全数据湖还支持其他 AWS 日志和来自第三方的日志。

  AWS 服务可以生成基本日志源未捕获到的日志，如弹性负载均衡日志、AWS WAF 日志、AWS Config 记录器日志、Amazon GuardDuty 调查发现、Amazon Elastic Kubernetes Service（Amazon EKS）审计日志，以及 Amazon EC2 实例操作系统和应用程序日志。有关日志记录和监控选项的完整列表，请参阅《AWS Security Incident Response Guide》的 Appendix A: Cloud capability definitions – Logging and Events。
- 研究每项 AWS 服务和应用程序的日志记录功能：每项 AWS 服务和应用程序都为您提供了日志存储选项，每个选项都有自己的保留和生命周期功能。两种很常见的日志存储服务是 Amazon Simple Storage Service（Amazon S3）和 Amazon CloudWatch。如果保留期较长，建议使用 Amazon

S3，因为它具有成本效益和灵活的生命周期功能。如果主要日志记录选项是 Amazon CloudWatch Logs，作为一种选择，您应该考虑将不太经常访问的日志存档到 Amazon S3。

- 选择日志存储：日志存储的选择通常与您使用的查询工具、保留能力、熟悉程度和成本有关。日志存储的主要选项是 Amazon S3 存储桶或 CloudWatch 日志组。

  Amazon S3 存储桶提供持久且经济高效的存储，并具有可选的生命周期策略。可以使用 Amazon Athena 等服务查询存储在 Amazon S3 存储桶中的日志。

  CloudWatch 日志组通过 CloudWatch Logs Insights 提供持久存储和内置查询工具。

- 确定适当的日志保留时长：使用 Amazon S3 存储桶或 CloudWatch 日志组存储日志时，必须为每个日志源建立足够的生命周期，以优化存储和检索成本。客户通常可以查询三个月到一年的日志，日志保留期长达七年。可用性和保留时长的选择应与您的安全要求以及法律法规和业务授权的综合因素相一致。

- 使用适当的保留时长和生命周期策略为每个 AWS 服务和应用程序启用日志记录：对于组织内的每个 AWS 服务或应用程序，请查找特定的日志记录配置指南：

  - 配置 AWS CloudTrail 跟踪
  - 配置 VPC 流日志
  - 配置 Amazon GuardDuty 调查发现导出
  - 配置 AWS Config 记录
  - 配置 AWS WAF Web ACL 流量
  - 配置 AWS Network Firewall 网络流量日志
  - 配置弹性负载均衡访问日志
  - 配置 Amazon Route 53 Resolver 查询日志
  - 配置 Amazon RDS 日志
  - 配置 Amazon EKS 控制面板日志
  - 为 Amazon EC2 实例和本地服务器配置 Amazon CloudWatch 代理

- 选择和实施日志查询机制：对于日志查询，可以使用 CloudWatch Logs Insights 对存储在 CloudWatch 日志组中的数据进行查询，使用 Amazon Athena 和 Amazon OpenSearch Service 对存储在 Amazon S3 中的数据进行查询。您还可以使用第三方查询工具，如安全信息和事件管理（SIEM）服务。

  选择日志查询工具的过程中，应考虑安全运营的人员、流程和技术方面。选择一款能够满足运营、业务和安全要求并可长期使用和维护的工具。请记住，当要扫描的日志数量保持在工具的限制范围内时，日志查询工具的工作状态最佳。由于成本或技术限制，拥有多款查询工具的情况并不罕见。

例如，您可能使用第三方安全信息和事件管理（SIEM）工具对过去 90 天的数据执行查询，但由于 SIEM 的日志提取成本较高，使用 Athena 来执行 90 天以上的查询。无论采用何种实施方式，都要验证您的方法能够尽可能地减少充分提高运营效率所需的工具数量，尤其在安全事件调查期间。

- 使用日志发出警报：AWS 通过多项安全服务提供警报功能：
  - AWS Config 监控和记录您的 AWS 资源配置，并允许您对照所需的配置自动执行评估和修复。
  - Amazon GuardDuty 是一项威胁检测服务，可持续监控恶意活动和未经授权的行为，以保护您的 AWS 账户和工作负载。GuardDuty 可从 AWS CloudTrail 管理和数据事件、DNS 日志、VPC 流日志和 Amazon EKS 审计日志等来源提取、聚合和分析信息。GuardDuty 可直接从 CloudTrail、VPC 流日志、DNS 查询日志和 Amazon EKS 提取独立的数据流。您无需管理 Amazon S3 存储桶策略，也无需修改日志的收集和存储方式。仍建议保留这些日志，以便您自己进行调查和遵守法规。
  - AWS Security Hub 集中聚合、组织和优先处理来自多个 AWS 服务和可选第三方产品的安全警报或调查发现，以使您全面了解安全警报和合规性状态。

  您也可以使用自定义警报生成引擎来处理这些服务未涵盖的安全警报或与您的环境相关的特定警报。有关构建这些警报和检测的信息，请参阅《AWS Security Incident Response Guide》中的 Detection。

资源

相关最佳实践：

- SEC04-BP02 在标准化位置收集日志、调查发现和指标
- SEC07-BP04 定义可扩展的数据生命周期管理
- SEC10-BP06 预部署工具

相关文档：

- 《AWS Security Incident Response Guide》
- Amazon Security Lake 入门
- 入门：Amazon CloudWatch Logs

相关视频：

- AWS re:Invent 2022 - Introducing Amazon Security Lake

相关示例：

- 专为 AWS 提供的 Assisted Log Enabler
- AWS Security Hub调查发现历史导出

SEC04-BP02 在标准化位置收集日志、调查发现和指标

安全团队依靠日志和调查发现来分析事件，找出可能表明出现了未经授权活动或意外更改的事件。为了简化这种分析过程，请将安全日志和调查发现收集到标准化位置。 这样就能将需要分析的数据点用于关联，并可以简化工具集成。

期望结果：您采用标准化的方法来收集、分析和可视化日志数据、调查发现和指标。安全团队可以高效地关联、分析和可视化不同系统的安全数据，以便发现潜在的安全事件并识别异常情况。集成了安全信息和事件管理（SIEM，Security Information and Event Management）系统或其它机制，用于查询和分析日志数据，以便及时响应、跟踪和上报安全事件。

常见反模式：

- 团队独立负责和管理日志记录及指标的收集，但是采取了与企业的日志记录策略不一致的方法。
- 团队没有采取足够的访问控制措施来限制所收集数据的可见性以及对数据的更改。
- 团队没有将其安全日志、调查发现和指标包括在数据分类策略中进行管理。
- 团队在配置数据收集时，忽略了数据主权和本地驻留要求。

建立此最佳实践的好处：采用标准化日志记录解决方案来收集和查询日志数据及事件，可以改善从所包含的信息中获得的见解。为收集的日志数据配置自动处理生命周期，可以降低日志存储产生的成本。您可以根据数据的敏感性和团队需要的访问模式，对收集的日志信息建立精细的访问控制。您可以集成工具，用于关联和可视化数据，以及从数据中发掘洞察。

在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

企业内 AWS 使用量的增长会导致分布式工作负载和环境数量的增加。由于这些工作负载和环境都会生成有关其内部活动的数据，因此在本地收集和存储这些数据对安全运营带来了挑战。安全团队使用安全信息和事件管理（SIEM，Security Information and Event Management）系统等工具，从分布式来源收集数据，并完成关联、分析和响应等工作流。这需要管理一组复杂权限以便访问各种数据来源，而且提取、转换、加载（ETL）流程的操作会带来额外开销。

要克服这些挑战，可以考虑将所有相关的安全日志数据来源汇总到一个日志归档账户中，如 Organizing Your AWS Environment Using Multiple Accounts 中所述。这包括您的工作负载的所有与安全相关的数据，以及各种 AWS 服务生成的日志，例如 AWS CloudTrail、AWS WAF、弹性负载均衡和 Amazon Route 53。使用合适的跨账户权限，在单独的 AWS 账户中的标准化位置收集这些数据可以带来多种好处。这种做法有助于防止受感染的工作负载和环境中发生日志篡改，提供可供其它工具使用的单一集成点，并为配置数据留存和生命周期提供更简化的模型。 评估数据主权、合规范围和其它法规的影响，确定是否需要多个安全数据存储位置和保留期。

为了简化日志和调查发现的收集和标准化工作，请评估在您的日志存档账户中是否适合使用 Amazon Security Lake。您可以将 Security Lake 配置为自动从各种常见事件源中摄取数据，例如 CloudTrail、Route 53、Amazon EKS 和 VPC 流日志。您也可以将 AWS Security Hub 配置作为传输到 Security Lake 的数据来源，这样您就可以将来自其它 AWS 服务（例如 Amazon GuardDuty 和 Amazon Inspector）的调查发现与您的日志数据相关联。 您还可以使用第三方数据来源集成，或配置自定义数据来源。所有集成都将您的数据标准化为开放网络安全架构框架（OCSF，Open Cybersecurity Schema Framework）格式，并作为 Parquet 文件存储在 Amazon S3 存储桶中，从而消除了 ETL 处理需求。

将安全数据存储在标准化位置可提供高级分析功能。AWS 建议您将 AWS 环境中运行的安全分析工具部署到安全工具账户中，而且该账户独立于您的日志存档账户。通过这种方法，您可以实施深入的控制措施，用来保护日志和日志管理流程的完整性和可用性，与访问这些日志的工具区分开。 考虑使用服务（例如 Amazon Athena）来运行关联多个数据来源的按需查询。还可以集成可视化工具，例如 QuickSight。人工智能驱动的解决方案的应用日益广泛，可以执行很多功能，例如将发现结果转化为人类可读的摘要并以自然语言进行交互。为查询设置标准化数据存储位置后，这些解决方案通常会更容易集成。

实施步骤

1. 创建日志存档账户和安全工具账户
    a. 使用 AWS Organizations，在安全组织单位下创建日志存档账户和安全工具账户。如果您使用 AWS Control Tower 来管理企业，则会自动为您创建日志存档账户和安全工具账户。您可以根据需要，配置用于访问和管理这些账户的角色和权限。
2. 配置标准化安全数据位置
    a. 确定创建标准化安全数据位置的策略。 为此，您可以使用通用数据湖架构方法、第三方数据产品或 Amazon Security Lake 等方案。AWS 建议您从您的账户选择加入的 AWS 区域中收集安全数据，即使这些区域并未处于活跃使用状态。
3. 将数据来源配置为发布到您的标准化位置

   a. 确定您的安全数据的来源，并将这些来源配置为发布到您的标准化位置。评估能够以所需格式自动导出数据的方案，而不是那些需要开发 ETL 流程的方案。使用 Amazon Security Lake，您可以从支持的 AWS 来源和集成的第三方系统收集数据。

4. 配置工具来访问您的标准化位置

   a. 配置 Amazon Athena、QuickSight 或第三方解决方案等工具，来获取您的标准化位置所要求的访问权限。 将这些工具配置为在安全工具账户之外运行，并在适用时，允许对日志存档账户的跨账户读取访问。在 Amazon Security Lake 上创建订阅用户，以便向这些工具提供对您数据的访问权限。

资源

相关最佳实践：

- SEC01-BP01 使用账户分隔工作负载
- SEC07-BP04 定义数据生命周期管理
- SEC08-BP04 强制实施访问控制
- OPS08-BP02 分析工作负载日志

相关文档：

- AWS Whitepapers: Organizing Your AWS Environment Using Multiple Accounts
- AWS Prescriptive Guidance: AWS Security Reference Architecture (AWS SRA)
- AWS Prescriptive Guidance: Logging and monitoring guide for application owners

相关示例：

- Aggregating, searching, and visualizing log data from distributed sources with Amazon Athena and QuickSight
- How to visualize Amazon Security Lake findings with QuickSight
- Generate AI powered insights for Amazon Security Lake using Amazon SageMaker AI Studio and Amazon Bedrock
- Identify cybersecurity anomalies in your Amazon Security Lake data using Amazon SageMaker AI
- Ingest, transform, and deliver events published by Amazon Security Lake to Amazon OpenSearch Service

- Simplify AWS CloudTrail log analysis with natural language query generation in CloudTrail Lake

相关工具：

- Amazon Security Lake
- Amazon Security Lake 合作伙伴集成
- 开放式网络安全架构框架（OCSF）
- Amazon Athena
- QuickSight
- Amazon Bedrock

SEC04-BP03 关联和扩充安全警报

出现意外的活动时，不同的来源可能会生成多个安全警报，此时需要进一步关联和扩充警报以便理解整个背景信息。实施自动化的安全警报关联和扩充，有助于更准确地识别和响应事件。

期望结果：当您的工作负载和环境中的活动生成了不同警报时，自动化机制会关联数据，并用其它信息扩充这些数据。通过这种预处理方法，您可以更详细地了解事件，这可以帮助调查人员确定事件的严重程度，以及是否构成了需要正式响应的事件。这个流程可减轻监控和调查团队的负担。

常见反模式：

- 除非职责分离要求另有规定，否则由不同团队的人员调查不同系统生成的调查发现和警报。
- 企业将所有安全调查发现和警报数据汇集到标准位置，但需要调查人员手动进行关联和扩充。
- 您完全依靠威胁检测系统的情报来报告调查发现并确定严重程度。

建立此最佳实践的好处：自动关联和扩充警报有助于减少调查人员的总体认知负荷和数据准备人工工作。这种做法可以缩短确定事件是否表示出现事故并启动正式响应所需的时间。其它背景信息还可以帮助您准确评测事件的真实严重性，因为其严重性可能会高于或低于任何警报所暗示的严重性。

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

安全警报可能来自 AWS 中的许多不同来源，包括：

- Amazon GuardDuty、AWS Security Hub、Amazon Macie、Amazon Inspector、AWS Config、AWS Identity and Access Management Access Analyzer 和网络访问分析器等服务

- 来自对 AWS 服务、基础设施和应用程序日志的自动分析的警报，例如来自 Amazon OpenSearch Service 的安全分析。
- 响应 Amazon CloudWatch、Amazon EventBridge 或 AWS Budgets 等来源中账单活动变化的警报。
- 第三方来源，例如来自 AWS Partner Network 的威胁情报源和安全合作伙伴解决方案
- AWS 信任与安全团队或其它来源（例如客户或内部员工）发起的联系事宜。

警报的最基本形式中包含谁（主体或身份）在对什么（受影响的资源）做了哪些事（操作）。对于每个这些来源，确定是否有办法可以为这些身份、操作和资源跨标识符创建映射，以此作为执行关联的基础。要想做到这一点，可以将警报来源与安全信息和事件管理（SIEM，Security Information and Event Management）工具集成在一起来为您执行自动关联，或者构建自己的数据管道和数据处理，也可将两种方法结合使用。

Amazon Detective 就是一个可以为您执行关联的服务示例。Detective 从各种 AWS 来源和第三方来源持续摄取警报，并使用不同形式的情报来描绘出这些关系的可视化图表，以协助进行调查。

虽然警报的初始严重程度可以用于协助排列优先顺序，但发生警报的具体环境决定了真正的严重程度。例如，Amazon GuardDuty 可能会发出警报，指出您工作负载中的某个 Amazon EC2 实例正在查询意外的域名。GuardDuty 可能会自行为此警报指定低严重程度。但是，通过自动关联发生警报时间前后的其它活动，可能会发现通过同一个身份部署了数百个 EC2 实例，这会增加总体运营成本。在这种情况下，这种相关的事件上下文将需要一个新的安全警报，且严重程度可能会被调整为高，这将加快进一步的行动。

实施步骤

1. 确定安全警报信息的来源。了解来自这些系统的警报所代表的身份、操作和资源，以便确定可能的关联性。
2. 建立一种机制来收集源自不同来源的警报。对于此目的，可以考虑使用 Security Hub、EventBridge 和 CloudWatch 等服务。
3. 确定用于数据关联和扩充的来源。示例来源包括 AWS CloudTrail、VPC 流日志、Route 53 Resolver 日志以及基础设施和应用程序日志。这些日志中的任何一个或所有日志都可以通过与 Amazon Security Lake 的单一集成来使用。
4. 将警报与您的数据关联和扩充来源集成，创建更详细的安全事件背景信息并确定严重程度。
    a. Amazon Detective、SIEM 工具或其它第三方解决方案可以自动执行一定级别的摄取、关联和扩展。

b. 您也可以使用 AWS 服务来构建自己的服务。例如，您可以调用一个 AWS Lambda 函数来对 AWS CloudTrail 或 Amazon Security Lake 运行 Amazon Athena 查询，并将结果发布到 EventBridge。

资源

相关最佳实践：

- SEC10-BP03 准备取证能力
- OPS08-BP04 创建可操作的警报
- REL06-BP03 发送通知（实时处理和报警）

相关文档：

- 《AWS Security Incident Response Guide》

相关示例：

- How to enrich AWS Security Hub findings with account metadata

相关工具：

- Amazon Detective
- Amazon EventBridge
- AWS Lambda
- Amazon Athena

SEC04-BP04 启动对不合规资源的修复

您的检测性控制措施可能会对不符合配置要求的资源发出警报。您可以手动或自动启动以编程方式定义的修复措施，用于修复这些资源并尽可能减少潜在影响。通过以编程方式定义的修复措施，您可以迅速采取一致的行动。

虽然利用自动化功能可以增强安全修复操作，但您应谨慎地实施和管理自动化功能。 您需要建立适当的监督和控制机制，确保自动化响应有效、准确且符合组织的策略和风险偏好。

期望结果：您定义了资源配置标准，以及在检测到资源不合规情况时应采取的修复措施。您尽可能以编程方式定义修复措施，以便手动或者自动启动这些措施。实施了检测系统，用于识别不合规的资源，并将警报发布到由您的安全人员监控的集中式工具。这些工具支持手动或自动运行您的程序化修复措施。采取了相应的监督和控制机制来管控自动修复措施的使用。

常见反模式：

- 您实施了自动化功能，但没有全面测试和验证修复措施。这可能会导致意外的后果，例如中断合法的业务运营或导致系统不稳定。
- 您利用自动化功能来改善响应时间和流程，但没有采取适当的监控和机制，以便在需要时进行人工干预和判断。
- 您完全依赖修复措施，而不是将修复措施作为一部分，融入到更全面的事件响应和恢复计划中。

建立此最佳实践的好处：面对错误配置，自动修复的响应速度比手动流程更快，这让您可以尽量减少潜在的业务影响，并减少出现意外使用情况的机会。以编程方式定义修复措施后，可以始终如一地应用这些措施，从而降低人为错误的风险。自动化功能还可以同时处理更大量的警报，这在大规模运行的环境中尤其重要。

在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

如 SEC01-BP03 识别并验证控制目标中所述，AWS Config 和 AWS Security Hub 等服务有助于您监控账户中资源的配置，以便符合您的要求。当检测到不合规的资源时，诸如 AWS Security Hub 之类的服务有助于相应地发送警报并进行补救。这些解决方案为安全调查人员提供了一个中心位置，可用来监控问题和采取纠正措施。

一些不合规资源的情况会非常独特，需要人工判断才能修复，不过其它情况可以采取以编程方式定义的标准响应。例如，对于 VPC 安全组的配置错误，标准响应可以是删除不允许的规则并通知负责人。可以在 AWS Lambda 函数、AWS Systems Manager Automation 文档中或者通过其它您常用的代码编写环境来定义响应措施。确保环境能够使用 IAM 角色在 AWS 上进行身份验证，并具有执行纠正措施所需的最低权限。

在定义了所需的修复措施之后，您就可以确定启动修复措施的首选方式。AWS Config 可以为您启动修复。如果您使用 Security Hub，则可以通过自定义操作来执行这个过程，自定义操作会将调查发现信息发布到 Amazon EventBridge。然后，EventBridge 规则启动修复措施。您可以通过 Security Hub 配置补救措施，使其自动或手动运行。

对于程序化的修复措施，我们建议您全面记录所采取的行动及其结果，并进行审计。查看和分析这些日志，以评测自动化流程的有效性，并确定需要改进的地方。将 Amazon CloudWatch Logs 中的日志和修复结果作为调查发现备注收集到 Security Hub 中。

首先，您可以考虑 AWS 上的自动化安全响应，其中预先构建了修复措施，用于解决常见的安全配置错误。

实施步骤

1. 分析警报并确定其优先级。
    a. 将来自各种 AWS 服务的安全警报整合到 Security Hub 中，以便在集中位置进行查看、划分优先级和执行修复措施。
2. 制定修复措施。
    a. 使用 Systems Manager 和 AWS Lambda 等服务来运行程序化的修复措施。
3. 配置启动修复的方式。
    a. 使用 Systems Manager，定义将调查发现发布到 EventBridge 的自定义操作。将这些操作配置为手动或自动启动。
    b. 如果需要，您还可以使用 Amazon Simple Notification Service（SNS）向相应的利益相关者（例如安全团队或事件响应团队）发送通知和警报，以便进行手动干预或上报。
4. 查看和分析修复日志，了解有效性并发现改进的机会。
    a. 将日志输出发送到 CloudWatch Logs。在 Security Hub 中将结果收集为调查发现备注。

资源

相关最佳实践：

- SEC06-BP03 减少人工管理工作和交互式访问

相关文档：

- 《AWS Security Incident Response Guide》- Detection

相关示例：

- AWS 上的自动化安全响应
- Monitor EC2 instance key pairs using AWS Config

- Create AWS Config custom rules by using AWS CloudFormation Guard policies
- Automatically remediate unencrypted Amazon RDS DB instances and clusters

相关工具：

- AWS Systems Manager Automation
- AWS 上的自动化安全响应

基础设施保护

问题

- SEC 5. 您如何保护您的网络资源？
- SEC 6. 您如何保护自己的计算资源？

SEC 5. 您如何保护您的网络资源？

任何以某种形式连接至网络（互联网或专用网络）的工作负载都需要多层防御，以帮助防御基于外部和内部网络的威胁。

最佳实践

- SEC05-BP01 创建网络层
- SEC05-BP02 控制网络层中的流量流动
- SEC05-BP03 实施基于检查的保护
- SEC05-BP04 自动执行网络保护

SEC05-BP01 创建网络层

根据工作负载组件的逻辑分组，按照数据敏感性和访问权限要求，将网络拓扑划分成不同的层。区分需要接受来自互联网的入站访问的组件（例如公有 Web 端点）与只需要进行内部访问的组件（例如数据库）。

期望结果：网络中的各个层是完整的深度防御安全方法的一部分，是工作负载身份验证和授权策略的有力补充。根据数据敏感性和访问权限要求进行了分层，并采用合适的流量流动和控制机制。

常见反模式：

- 您在单个 VPC 或子网中创建所有资源。
- 在构造网络层时，您没有考虑数据敏感性要求、组件行为或功能。
- 您使用 VPC 和子网的默认值，未考虑所有的网络层注意事项，而且也未考虑 AWS 托管服务对拓扑有何影响。

建立此最佳实践的好处：建立网络层是限制网络中不必要路径的第一步，尤其是在需要限制去往关键系统和数据的路径的情况下。通过这种方法，未经授权的操作者更难于访问您的网络，也更难导航到网络中的其它资源。彼此分隔的网络层带来的益处包括减少了检查系统的分析范围，例如进行入侵检测或恶意软件防御时。这可以减少误报的可能性和不必要的处理开销。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在设计工作负载架构时，一种常见的做法是根据组件的职责将组件分隔到不同的层中。例如，Web 应用程序可以具有表示层、应用层和数据层。在设计网络拓扑时，您可以采用类似的方法。底层网络控制措施可用于强制执行工作负载的数据访问权限要求。例如，在三层 Web 应用程序架构中，您可以将静态的表示层文件存储在 Amazon S3 中，并通过 Amazon CloudFront 等内容分发网络（CDN）来提供这些文件。应用层可以有公有端点，该端点由应用程序负载均衡器（ALB）在 Amazon VPC 公有子网（类似于非军事区，简称 DMZ）中提供服务，并将后端服务部署到私有子网中。数据层中托管数据库和共享文件系统等资源，可以与应用层的资源位于不同的私有子网中。在每个这些层的边界（CDN、公有子网、私有子网），您都可以部署控制措施，仅允许授权流量穿过这些边界。

类似于根据工作负载组件的功能用途对网络层进行建模，此时同样需要考虑所处理数据的敏感性。以 Web 应用程序为例，虽然您的所有工作负载服务可能都位于应用层内，但不同的服务会处理具有不同敏感性级别的数据。在这种情况下，根据您的控制要求，您可能需要针对不同的数据敏感性来划分应用层，例如使用多个私有子网、同一个 AWS 账户 中的不同 VPC，甚至是不同 AWS 账户 中的不同 VPC。

网络层的另一个注意事项是工作负载组件的行为一致性。仍旧以上例来说明，在应用层中，您的服务可能接受来自最终用户或外部系统集成的输入，这些输入本质上比其它服务的输入风险更大。例如文件上传、要运行的代码脚本、电子邮件扫描等。将这些服务放在各自的网络层中，可以在这些服务周围建立更可靠的隔离边界，并可以防止它们的独特行为在检查系统中产生误报提醒。

在设计过程中，请考虑使用 AWS 托管服务会对您的网络拓扑造成什么影响。探索 Amazon VPC Lattice 等服务如何有助于简化跨网络层的工作负载组件互操作性。使用 AWS Lambda 时，除非有特殊的原因，否则应将该服务部署在您的 VPC 子网中。对于限制访问互联网网关的安全策略，确定如何利用 VPC 端点和 AWS PrivateLink 来简化遵守这些策略所需的工作。

实施步骤

1. 查看您的工作负载架构。根据组件和服务提供的功能、所处理数据的敏感性及其行为，对组件和服务进行逻辑分组。
2. 如果组件需要响应来自互联网的请求，请考虑使用负载均衡器或其它代理来提供公有端点。探索通过使用 CloudFront、Amazon API Gateway、弹性负载均衡和 AWS Amplify 等托管服务来托管公有端点，从而转变安全控制模式。
3. 对于在计算环境中运行的组件，例如 Amazon EC2 实例、AWS Fargate 容器或 Lambda 函数，请根据您在第一步中的分组，将它们部署到私有子网中。
4. 对于完全托管式 AWS 服务，例如 Amazon DynamoDB、Amazon Kinesis 或 Amazon SQS，请考虑默认使用 VPC 端点来通过私有 IP 地址进行访问。

资源

相关最佳实践：

- REL02 计划网络拓扑
- PERF04-BP01 了解联网对性能的影响

相关视频：

- AWS re:Invent 2023 – AWS networking foundations

相关示例：

- VPC 示例
- 使用 AWS Fargate、AWS PrivateLink，和网络负载均衡器在 Amazon ECS 上私密访问容器应用程序
- 使用 Amazon CloudFront 通过 VPC 在 Amazon S3 存储桶中提供静态内容

SEC05-BP02 控制网络层中的流量流动

在网络的各层中，进一步对网络进行分段，从而将流量限制在对各个工作负载所必要的流动路径中。首先，将重点放在控制互联网或其它外部系统与工作负载和您环境之间的流量（南北向流量）上。然后，审查不同组件与系统之间的流量（东西向流量）。

期望结果：您只允许必要的网络流量流动，以便让工作负载的组件彼此通信，以及与其客户端和所依赖的任何其它服务进行通信。您在设计时充分考虑了各种因素，例如公有入口和出口与私有入口和出口的对比、数据分类、区域法规，以及协议要求。作为最低权限原则设计的一部分，您尽可能地使用点对点流动，而不是网络对等连接。

常见反模式：

- 您采用基于边界的方法来保护网络，并且仅在网络层的边界控制流量流动。
- 您假设一个网络层中的所有流量都经过了身份验证和授权。
- 您对入口流量或出口流量进行了控制，但没有对两者均进行控制。
- 您完全依靠工作负载组件和网络控制措施来对流量进行身份验证和授权。

建立此最佳实践的好处：这种做法有助于减少网络中未经授权移动的风险，并为您的工作负载增加了额外的授权层。通过执行流量流动控制，您可以限制安全事件的影响范围，同时加快检测和响应的速度。

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

虽然通过网络层，您可以围绕工作负载中具有相似功能、数据敏感性级别和行为的组件来划定边界，不过您可以利用各种技术，在这些层中遵循最低权限原则来进一步划分组件，从而创建更精细的流量控制级别。在 AWS 中，网络层主要根据 Amazon VPC 中的 IP 地址范围使用子网进行定义。您也可以使用不同的 VPC 来定义层，例如按业务域来对微服务环境进行分组。使用多个 VPC 时，使用 AWS Transit Gateway 来仲裁路由。虽然这种方法使用安全组和路由表在第 4 层上（IP 地址和端口范围）提供流量控制，不过您可以使用其它服务（例如 AWS PrivateLink、Amazon Route 53 Resolver DNS 防火墙、AWS Network Firewall 和 AWS WAF）实现进一步的控制。

了解工作负载的数据流动和通信要求并列出清单，这涉及到连接发起方、端口、协议和网络层等。评估可用于建立连接和传输数据的协议，从而选择符合您的保护要求的协议（例如，HTTPS 而不是 HTTP）。在网络边界和每个层中收集这些要求。确定这些要求后，探索仅允许所需流量流经每个连接点的选项。作为开始，一种很好的方法是在 VPC 中使用安全组，因为安全组可以连接到使用弹性网络接口（ENI）的资源，例如 Amazon EC2 实例、Amazon ECS 任务、Amazon EKS 容器组（pod）或 Amazon RDS 数据库。与第 4 层防火墙不同，安全组可以设定一条规则，按照标识符来允许其它安全组的流量，这样当组中的资源随着时间发生变化时，能尽可能减少更新工作。您还可以使用安全组，通过入站和出站规则来筛选流量。

当流量在 VPC 之间移动时，对于简单路由通常会使用 VPC 对等连接，对于复杂路由则使用 AWS Transit Gateway。使用这些方法，您可以在源网络和目标网络的 IP 地址范围之间，协调流量流

动。但是，如果您的工作负载只需要在不同 VPC 中的特定组件之间的流量流动，请考虑使用 AWS PrivateLink 进行点对点连接。为此，请确定哪些服务应充当产生器，哪些服务应充当使用器。为产生器部署兼容的负载均衡器，相应地启用 PrivateLink，然后接受使用器的连接请求。接下来，向产生器服务分配来自使用器 VPC 的私有 IP 地址，使用器可以使用该地址发出后续请求。这种方法减少了对网络对等连接的需求。评估 PrivateLink 时，请包括数据处理和负载均衡的成本。

虽然安全组和 PrivateLink 均可用于控制工作负载的组件之间的流量流动，不过还有另一个重要考虑因素，即如何控制允许您的资源访问哪些 DNS 域（如果有）。根据 VPC 的 DHCP 配置，您可以考虑使用两种不同的 AWS 服务来实现此目的。大多数客户使用默认的 Route 53 Resolver DNS 服务（也称为 Amazon DNS 服务器或 AmazonProvidedDNS），该服务可用于 VPC，地址为其 CIDR 范围 +2。通过这种方法，您可以创建 DNS 防火墙规则，然后将规则关联到 VPC，用来确定对您提供的域列表采取哪些操作。

如果您不使用 Route 53 Resolver，或者想在域筛选之外，利用更深入的检查和流量控制功能来补充 Resolver，请考虑部署 AWS Network Firewall。此服务使用无状态或有状态规则检查单独的数据包，以确定是拒绝还是允许流量。您可以使用类似的方法，通过 AWS WAF 来筛选流向公有端点的入站 Web 流量。有关这些服务的更多指导信息，请参阅《SEC05-BP03 实施基于检查的保护》。

实施步骤

1. 确定在工作负载的组件之间必需的数据流。
2. 对于入站和出站流量，采用深度防御方法应用多种控制措施，包括使用安全组和路由表。
3. 针对出入 VPC 和 VPC 之间的网络流量，使用防火墙定义精细的控制措施，例如 Route 53 Resolver DNS 防火墙、AWS Network Firewall 和 AWS WAF。考虑使用 AWS Firewall Manager 来集中配置和管理整个企业的防火墙规则。

资源

相关最佳实践：

- REL03-BP01 选择如何划分工作负载
- SEC09-BP02 在传输中执行加密

相关文档：

- VPC 的安全最佳实践
- AWS Network Optimization Tips
- Guidance for Network Security on AWS

- Secure your VPC's outbound network traffic in the AWS Cloud

相关工具：

- AWS Firewall Manager

相关视频：

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield
- AWS re:Inforce 2023: Firewalls and where to put them

SEC05-BP03 实施基于检查的保护

在各网络层之间设置流量检测点，确保传输中数据符合预期的类别和模式。 分析流量、元数据和模式，以便于更有效地识别、检测和响应事件。

期望结果：在各网络层之间穿行的流量均经过检查和授权。 允许和拒绝的决定基于明确的规则、威胁情报和偏离基线的行为。 流量越接近敏感数据，保护措施就越严格。

常见反模式：

- 仅依赖基于端口和协议的防火墙规则，而不利用智能系统。
- 根据当前可能发生变化的特定威胁模式制定防火墙规则。
- 只检查从私有子网传输到公有子网或从公有子网传输到互联网的流量。
- 没有网络流量基线视图，无法与异常行为对照。

建立此最佳实践的好处：检测系统允许您制定智能规则，例如仅当流量数据中存在特定条件时才允许或拒绝流量。根据最新的威胁情报，从 AWS 和合作伙伴提供的托管规则集获益，因为威胁状况会随着时间的推移而发生变化。 这减少了维护规则和研究折衷指标的开销，降低了误报的可能性。

在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 AWS Network Firewall 或 AWS Marketplace 上其它可部署在网关负载均衡器（GWLB）后面的防火墙和入侵防御系统（IPS，Intrusion Prevention Systems），对有状态和无状态网络流量进行细粒度控制。AWS Network Firewall 支持与 Suricata 兼容的开源 IPS 规范，有助于保护您的工作负载。

AWS Network Firewall 和使用 GWLB 的供应商解决方案都支持不同的内联检查部署模型。 例如，您可以逐个 VPC 执行检查，在所检查 VPC 内集中进行检查，或者以混合模式进行部署，即东西向流量流经检查 VPC，而互联网入口则逐个 VPC 进行检查。 另一个考虑因素是，解决方案是否支持解包传输层安全性协议（TLS），从而能够对任一方向启动的流量进行深度数据包检查。有关这些配置的更多信息和深入细节，请参阅《AWS Network Firewall 最佳实践指南》。

如果您使用的是执行带外检查的解决方案，例如对来自以混杂模式运行的网络接口的数据包数据进行 pcap 分析，则可以配置 VPC 流量镜像。镜像流量计入接口的可用带宽，与非镜像流量收取相同的数据传输费用。您可以查看 AWS Marketplace 上是否有这些设备的虚拟版本，它们可能支持在 GWLB 后面进行内联部署。

对于通过基于 HTTP 的协议进行事务处理的组件，应使用 Web 应用程序防火墙（WAF，Web Application Firewall）保护应用程序免受常见威胁。AWS WAF 是一种 Web 应用程序防火墙，可让您在将符合可配置规则的 HTTP(S) 请求发送到 Amazon API Gateway、Amazon CloudFront、AWS AppSync 或应用程序负载均衡器之前，监控并阻止这些请求。在评估 Web 应用程序防火墙的部署时，可以考虑深度数据包检查，因为有些防火墙要求在流量检查前终止 TLS。要开始使用 AWS WAF，您可以将 AWS 托管式规则 与自己的规则结合使用，也可以使用现有的合作伙伴集成。

您可以使用 AWS Firewall Manager，在整个 AWS 组织内集中管理 AWS WAF、AWS Shield Advanced、AWS Network Firewall 和 Amazon VPC 安全组。

实施步骤

1. 确定是可以通过检查 VPC 等方式宽泛地确定检查规则的范围，还是需要更细粒度的针对每个 VPC 的方法。
2. 对于内联检查解决方案：
   a. 如果使用 AWS Network Firewall，则创建规则、防火墙策略和防火墙本身。配置完这些后，就可以将流量路由到防火墙端点以便启用检查。
   b. 如果使用带有网关负载均衡器（GWLB）的第三方设备，请在一个或多个可用区内部署和配置设备。然后，创建 GWLB、端点服务、端点，并为流量配置路由。
3. 对于带外检查解决方案：
   1. 在应该对入站和出站流量进行镜像的接口上，启用 VPC 流量镜像功能。您可以使用 Amazon EventBridge 规则调用 AWS Lambda 函数，以便在创建新资源时在接口上启用流量镜像功能。将流量镜像会话指向在设备前面处理流量的网络负载均衡器。
4. 对于入站 Web 流量解决方案：

a. 要配置 AWS WAF，首先要配置 Web 访问控制列表（Web ACL）。Web ACL 是众多规则的集合，具有连续处理的默认操作（ALLOW 或 DENY），可定义 WAF 如何处理流量。您可以创建自己的规则和组，也可以在 Web ACL 中使用 AWS 托管规则组。

b. 配置好 Web ACL 后，将 Web ACL 与 AWS 资源（如应用程序负载均衡器、API Gateway REST API 或 CloudFront 分配）关联，即可开始保护 Web 流量。

资源

相关文档：

- What is Traffic Mirroring?
- Implementing inline traffic inspection using third-party security appliances
- AWS Network Firewall example architectures with routing
- Centralized inspection architecture with AWS Gateway Load Balancer and AWS Transit Gateway

相关示例：

- 部署网关负载均衡器的最佳实践
- TLS inspection configuration for encrypted egress traffic and AWS Network Firewall

相关工具：

- AWS Marketplace IDS/IPS

SEC05-BP04 自动执行网络保护

使用 DevOps 实践 [例如基础设施即代码（IaC）和 CI/CD 管道] 自动部署网络保护。 这些实践有助于您通过版本控制系统跟踪网络保护措施的变更，缩短部署变更所需的时间，并有助于检测网络保护措施是否偏离了您所需的配置。

期望结果：您可以使用模板来定义网络保护，并将模板提交到版本控制系统中。 当有新的变更时，自动管道就会启动，协调这些变更的测试和部署。 进行策略检查和其它静态测试，以便在部署之前验证变更。 您可以将变更部署到暂存环境中，以便验证控制措施是否按预期运行。 一旦控制措施获得批准，还可自动部署到生产环境中。

常见反模式：

- 依靠各个工作负载团队各自定义完整的网络堆栈、保护措施和自动化。 不集中发布网络堆栈和保护措施的标准内容，供工作负载团队使用。
- 依靠中央网络团队来定义网络、保护措施和自动化的所有方面。 不将网络堆栈和保护措施的特定工作负载方面委托给该工作负载的团队。
- 在网络团队和工作负载团队之间的集中化和委托之间取得适当平衡，但不在 IaC 模板和 CI/CD 管道中应用一致的测试和部署标准。 没有在检查模板是否符合要求的工具中捕获所需的配置。

建立此最佳实践的好处：使用模板来定义网络保护，可以通过版本控制系统跟踪和比较随时间发生的变更。 使用自动化功能来测试和部署变更，可实现标准化和可预测性，增加成功部署的机会，减少重复的手动配置。

在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

SEC05-BP02 控制网络层中的流量流动和 SEC05-BP03 实施基于检查的保护中描述的许多网络保护控制措施都带有可根据最新威胁情报自动更新的托管规则系统。 保护 Web 端点的示例包括 AWS WAF 托管规则和 AWS Shield Advanced 自动应用层 DDoS 缓解。使用 AWS Network Firewall 托管规则组，也可随时更新低声誉域列表和威胁特征。

除了托管规则外，我们还建议您使用 DevOps 实践来自动部署网络资源、保护措施和您指定的规则。 您可以在 AWS CloudFormation 或您选择的其它基础设施即代码（IaC）工具中捕获这些定义，将其提交到版本控制系统，并使用 CI/CD 管道进行部署。 使用这种方法可获得 DevOps 在管理网络控制方面的传统优势，如更具可预测性的发布、使用 AWS CloudFormation Guard 等工具进行自动测试，以及检测已部署环境与所需配置之间的偏差等。

根据您在 SEC05-BP01 创建网络层中做出的决定，您可以采用集中管理方法创建 VPC，专用于入口、出口和检查流。 如 AWS Security Reference Architecture（AWS SRA）所述，您可以在专用网络基础设施账户中定义这些 VPC。 您可以使用类似的技术，来集中定义其它账户中工作负载使用的 VPC、其安全组、AWS Network Firewall 部署、Route 53 Resolver 规则和 DNS Firewall 配置以及其它网络资源。 您可以通过 AWS Resource Access Manager 与其它账户共享这些资源。 通过这种方法，您可以将网络控制的自动测试和部署简化到网络账户中，只需管理一个目标即可。 您可以采用混合模式来实现这一点，即集中部署和共享某些控制措施，并将其它控制措施委托给各个工作负载团队及其各自的账户。

实施步骤

1. 确定网络和保护措施的哪些方面是集中定义的，哪些是工作负载团队可以维护的。

2. 创建环境来测试和部署对网络及其保护措施的变更。 例如，使用“网络测试”账户和“网络生产”账户。
3. 确定如何在版本控制系统中存储和维护模板。 将中央模板存储在有别于工作负载存储库的存储库中，而工作负载模板可存储在特定于该工作负载的存储库中。
4. 创建 CI/CD 管道来测试和部署模板。 定义测试方法，用于检查配置是否有误，以及模板是否符合公司标准。

资源

相关最佳实践：

- SEC01-BP06 自动部署标准安全控制措施

相关文档：

- AWS Security Reference Architecture - Network account

相关示例：

- AWS Deployment Pipeline Reference Architecture
- NetDevSecOps to modernize AWS networking deployments
- Integrating AWS CloudFormation security tests with AWS Security Hub and AWS CodeBuild reports

相关工具：

- AWS CloudFormation
- AWS CloudFormation Guard
- cfn_nag

SEC 6. 您如何保护自己的计算资源？

工作负载中的计算资源需要多层防御，以帮助抵御外部和内部威胁。计算资源包括 EC2 实例、容器、AWS Lambda 函数、数据库服务、IoT 设备等。

最佳实践

- SEC06-BP01 执行漏洞管理

- SEC06-BP02 从强化映像预置计算
- SEC06-BP03 减少人工管理工作和交互式访问
- SEC06-BP04 验证软件完整性
- SEC06-BP05 自动保护计算

SEC06-BP01 执行漏洞管理

频繁扫描和修补您的代码、依赖项和基础设施中的漏洞，以帮助防御新的威胁。

期望结果：您的解决方案可以持续扫描工作负载，来发现软件漏洞、潜在缺陷和意外的网络泄露。您已经制定了流程和过程，可以根据风险评测标准来识别这些漏洞、确定其优先级并对其进行修复。此外，您还为计算实例实施了自动补丁管理。您的漏洞管理程序已集成到软件开发生命周期中，并提供了在 CI/CD 管道期间扫描源代码的解决方案。

常见反模式：

- 未制定漏洞管理计划。
- 在不考虑严重性或风险规避的情况下执行系统修补。
- 使用已超过供应商提供的生命周期结束（EOL）日期的软件。
- 在分析安全问题之前，将代码部署到生产环境中。

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

漏洞管理是维护安全且稳健的云环境的一个关键环节。它涉及一个全面的流程，包括安全扫描、问题的识别和优先级排序，以及用于修复已识别漏洞的修补操作。自动化在此流程中起着举足轻重的作用，因为它有助于对工作负载进行持续扫描来发现潜在问题和意外的网络泄露，以及实施修复工作。

AWS 责任共担模式是支撑漏洞管理的基本概念。根据该模式，AWS 负责保护底层基础设施的安全，包括运行 AWS 服务的硬件、软件、网络和设施。相反，您负责保护与 Amazon EC2 实例和 Amazon S3 对象等服务关联的数据、安全配置和管理任务。

AWS 提供一系列服务来支持漏洞管理计划。Amazon Inspector 持续扫描 AWS 工作负载中是否存在软件漏洞和意外网络访问，而 AWS Systems Manager 补丁管理器则有助于管理跨 Amazon EC2 实例的修补工作。这些服务可以与 AWS Security Hub 这一云安全态势管理服务集成，该服务可自动执行 AWS 安全检查，集中安全警报，并提供组织安全态势的全面视图。此外，Amazon CodeGuru 安全防御工具使用静态代码分析，来识别 Java 和 Python 应用程序在开发阶段期间的潜在问题。

通过将漏洞管理实践纳入软件开发生命周期，您可以在漏洞引入生产环境之前主动解决漏洞，从而降低安全事件的风险，并最大限度地减少漏洞的潜在影响。

实施步骤

1. 了解责任共担模式：查看 AWS 责任共担模式，来了解您在云端保护工作负载和数据的责任。AWS 负责保护底层云基础设施，而您负责保护您的应用程序、数据和所使用的服务。
2. 实施漏洞扫描：配置漏洞扫描服务（例如 Amazon Inspector），以便自动扫描计算实例（例如虚拟机、容器或无服务器函数），来查找软件漏洞、潜在缺陷和意外的网络泄露。
3. 建立漏洞管理流程：定义用于识别漏洞、确定漏洞优先级和修复漏洞的流程和过程。这可能包括制定定期漏洞扫描计划、建立风险评估标准以及根据漏洞严重程度定义修补时间表。
4. 设置补丁管理：使用补丁管理服务，来自动执行为操作系统和应用程序修补计算实例的过程。您可以将服务配置为扫描实例中缺少的补丁，并按计划自动安装这些补丁。可以考虑使用 AWS Systems Manager 补丁管理器来提供此功能。
5. 配置恶意软件防护：实施相应的机制来检测环境中的恶意软件。例如，可以使用诸如 Amazon GuardDuty 之类的工具来分析、检测 EC2 和 EBS 卷中的恶意软件并发出警报。GuardDuty 还可以扫描新上传到 Amazon S3 的对象中是否存在潜在的恶意软件或病毒，并在它们被摄取到下游进程之前采取措施将其隔离。
6. 在 CI/CD 管道中集成漏洞扫描：如果您使用 CI/CD 管道进行应用程序部署，请将漏洞扫描工具集成到您的管道中。诸如 Amazon CodeGuru 安全防御工具和开源选项之类的工具可以扫描源代码、依赖项和构件，来发现潜在的安全问题。
7. 配置安全监控服务：设置安全监控服务（例如 AWS Security Hub），来全面了解您在多个云服务中的安全状况。该服务应从各种来源收集安全调查发现，并以标准化格式呈现它们，以便于确定优先级和进行补救。
8. 实施 Web 应用程序渗透测试：如果您的应用程序是 Web 应用程序，并且您的组织具有必要的技能或可以聘请外部协助，请考虑实施 Web 应用程序渗透测试，以识别应用程序中的潜在漏洞。
9. 利用基础设施即代码实现自动化：使用基础设施即代码（IaC）工具（例如 AWS CloudFormation）来自动部署和配置资源，包括前面提到的安全服务。这种做法有助于您在多个账户和环境中创建更加一致和标准化的资源架构。
10. 监控并持续改进：持续监控漏洞管理计划的有效性，并根据需要进行改进。审核安全调查发现，评测补救工作的有效性，并相应地调整您的流程和工具。

资源

相关文档：

- AWS Systems Manager
- AWS Lambda 安全性概述
- Amazon CodeGuru
- 使用新的 Amazon Inspector 改进了云工作负载的自动化漏洞管理
- 使用 Amazon Inspector 和 AWS Systems Manager 自动执行 AWS 中的漏洞管理和修复 – 第 1 部分

相关视频：

- 保护无服务器和容器服务
- Security best practices for the Amazon EC2 instance metadata service

SEC06-BP02 从强化映像预置计算

通过从强化映像部署运行时环境，减少意外访问运行时环境的机会。只从可信注册表获取运行时依赖项（例如容器映像和应用程序库），并验证其签名。创建自己的专用注册表来存储可信映像和库，供构建和部署流程使用。

期望结果：您的计算资源是从强化的基准映像预置的。您只从可信注册表检索外部依赖项（例如容器映像和应用程序库），并验证其签名。这些依赖项存储在专用注册表中，供构建和部署流程参考。您会定期扫描和更新映像和依赖项，以便于应对任何新发现的漏洞。

常见反模式：

- 从可信注册表获取映像和库，但在投入使用前不验证其签名或进行漏洞扫描。
- 强化映像，但没有定期测试映像是否存在新漏洞或更新到最新版本。
- 安装或不删除在映像预期生命周期内不需要的软件包。
- 仅依靠打补丁来保持生产计算资源的最新状态。随着时间的推移，仅靠打补丁仍会导致计算资源偏离强化标准。打补丁也可能无法清除威胁行为者在安全事件中安装的恶意软件。

建立此最佳实践的好处：强化映像有助于减少在运行时环境中，可能允许未经授权的用户或服务进行意外访问的路径数量。如果发生任何意外访问，强化映像还可以缩小影响范围。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要强化系统，请从最新版本的操作系统、容器映像和应用程序库开始。应用补丁以解决已知问题。删除所有不需要的应用程序、服务、设备驱动程序、默认用户和其它凭证，尽量减小系统。采取其它任何必要操作，例如禁用端口，以便创建一个只拥有工作负载所需资源和功能的环境。在此基础上，您可以安装必要的软件、代理或其它进程，以满足监控工作负载或管理漏洞等目的的需要。

您可以遵循可信来源提供的指导，如互联网安全中心（CIS，Center for Internet Security）以及美国国防信息系统局（DISA，Defense Information Systems Agency）的安全技术实施指南（STIG），从而减轻强化系统的负担。我们建议您从 AWS 或 APN 合作伙伴发布的亚马逊机器映像（AMI）开始，并使用 AWS EC2 Image Builder，以期综合使用 CIS 和 STIG 控制措施来自动配置。

虽然有可用的强化映像和 EC2 Image Builder 配方可应用 CIS 或 DISA STIG 建议，但您可能会发现，它们的配置会阻止您的软件成功运行。在这种情况下，您可以从未经强化的基础映像开始，安装软件，然后逐步应用 CIS 控制措施来测试其影响。对于任何阻止软件运行的 CIS 控制措施，请测试是否可以改为在 DISA 中实施更精细的强化建议。跟踪您能够成功应用的不同 CIS 控制措施和 DISA STIG 配置。在 EC2 Image Builder 中使用这些控制措施和配置，相应地定义映像强化配方。

对于容器化工作负载，Amazon Elastic Container Registry（ECR）公共存储库提供 Docker 的强化映像。您可以结合使用 EC2 Image Builder 与 AMI 来强化容器映像。

与操作系统和容器映像类似，您可以通过 pip、npm、Maven 和 NuGet 等工具，从公共存储库中获取代码包（或库）。我们建议您将私有存储库（例如在 AWS CodeArtifact 中）与可信的公共存储库进行集成，来管理代码包。这种集成可为您处理代码包的检索、存储和保持最新状态。然后，您的应用程序构建流程就可以使用一些技术 [例如软件组成分析（SCA，Software Composition Analysis）、静态应用程序安全测试（SAST，Static Application Security Testing）和动态应用程序安全测试（DAST，Dynamic Application Security Testing）等]，与您的应用程序一起获取和测试这些代码包的最新版本。

对于使用 AWS Lambda 的无服务器工作负载，可使用 Lambda 层简化对代码包依赖项的管理。使用 Lambda 层将不同函数之间共享的一组标准依赖项配置到独立的存档中。您可以通过自己的构建流程来创建和维护层，从而能够以集中方式使您的函数保持最新状态。

实施步骤

- 强化操作系统。使用来自可信来源的基础映像为基础，来构建强化的 AMI。使用 EC2 Image Builder 来帮助自定义安装在映像上的软件。
- 强化容器化资源。配置容器化资源以符合安全最佳实践。当使用容器时，在您的构建管道中对您的映像存储库定期实施 ECR 映像扫描，以便在您的容器中查找 CVE。

- 在使用 AWS Lambda 实现无服务器时，请使用 Lambda 层来隔离应用程序函数代码和共享的依赖项库。为 Lambda 配置代码签名，以便确保只有可信代码才能在您的 Lambda 函数中运行。

资源

相关最佳实践：

- OPS05-BP05 执行补丁管理

相关视频：

- Deep dive into AWS Lambda security

相关示例：

- Quickly build STIG-compliant AMI using EC2 Image Builder
- Building better container images
- Using Lambda layers to simplify your development process
- Develop & Deploy AWS Lambda Layers using Serverless Framework
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST and DAST tools

SEC06-BP03 减少人工管理工作和交互式访问

尽可能使用自动化方式来执行部署、配置、维护和调查任务。在紧急程序或安全（沙盒）环境中，如果自动化不可用，可以考虑手动访问计算资源。

期望结果：程序化脚本和自动化文档（运行手册）可捕获计算资源上的授权操作。这些运行手册可以通过变更检测系统自动启动，也可以在需要人工判断时手动启动。只有在无法实现自动化的紧急情况下，才允许直接访问计算资源。所有手动活动都会被记录下来并纳入审查流程，以便不断提高自动化能力。

常见反模式：

- 使用 SSH 或 RDP 等协议对 Amazon EC2 实例进行交互式访问。
- 维护个人用户登录信息，例如 `/etc/passwd` 或 Windows 本地用户。
- 多个用户共用一个密码或私钥来访问实例。
- 手动安装软件，手动创建或更新配置文件。

- 手动更新或修补软件。
- 登录实例来解决问题。

建立此最佳实践的好处：自动执行操作有助于降低意外更改和错误配置的操作风险。避免使用 Secure Shell（SSH）和远程桌面协议（RDP，Remote Desktop Protocol）进行交互式访问，可缩小计算资源的访问范围。这样可以消除一种执行未经授权操作的常见方式。可以在自动化文档和程序化脚本中捕获计算资源管理任务，这种机制以细粒度的方式定义和审计授权活动的全部范围。

在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

登录到实例上是一种传统的系统管理方法。安装服务器操作系统后，用户通常会手动登录，以便配置系统并安装所需的软件。在服务器的生命周期内，用户可能会登录服务器来更新软件、应用补丁、更改配置和解决问题。

然而，手动访问会带来一些风险。这需要一个能监听请求（如 SSH 或 RDP 服务）的服务器，这就可能为未经授权的访问提供潜在的路径。这还增加了与执行手动措施相关的人为出错风险。这些操作可能导致工作负载事件、数据损坏或毁坏或者其它安全问题。人工访问还需要防止共享凭证，从而增加了管理开销。

为了降低这些风险，您可以实施基于代理的远程访问解决方案，例如 AWS Systems Manager。AWS Systems ManagerAgent（SSM Agent）会启动一个加密通道，因此它不依赖于监听外部发起的请求。考虑配置 SSM Agent 以便通过 VPC 端点建立此通道。

利用 Systems Manager 可以精细控制您与托管实例进行交互的方式。您可以定义要运行的自动化操作、谁可以运行以及何时运行。Systems Manager 可以打补丁、安装软件和更改配置，而无需与实例进行交互式访问。Systems Manager 还可提供对远程 Shell 的访问，并将会话期间调用的每条命令及其输出记录到日志和 Amazon S3 中。AWS CloudTrail 会记录对 Systems Manager API 的调用，以供检查之用。

实施步骤

1. 在 Amazon EC2 实例上安装 AWS Systems Manager Agent（SSM Agent）。检查 SSM Agent 是否包含在基本 AMI 配置中并能够自动启动。
2. 验证与 EC2 实例配置文件相关联的 IAM 角色是否包含 `AmazonSSMManagedInstanceCore` 托管 IAM 策略。
3. 禁止在实例上运行 SSH、RDP 和其它远程访问服务。为此，您可以运行在启动模板的用户数据部分内配置的脚本，或者使用 EC2 Image Builder 等工具构建自定义 AMI。

4. 确保适用于 EC2 实例的安全组入口规则不允许访问端口 22/tcp（SSH）或端口 3389/tcp（RDP）。使用 AWS Config 等服务对配置错误的安全组实施检测和提醒。
5. 在 Systems Manager 中定义适当的自动化操作、运行手册和运行命令。使用 IAM 策略来定义谁可以执行这些操作以及允许执行这些操作的条件。请在非生产环境中彻底测试这些自动化操作。请尽可能调用这些自动化操作，而不是以交互方式访问实例。
6. 必要时，使用 AWS Systems Manager Session Manager 提供对实例的交互式访问。启用会话活动日志记录，以便在 Amazon CloudWatch Logs 或 Amazon S3 中保留审计跟踪记录。

资源

相关最佳实践：

- REL08-BP04 使用不可变基础设施进行部署

相关示例：

- Replacing SSH access to reduce management and security overhead with AWS Systems Manager

相关工具：

- AWS Systems Manager

相关视频：

- Controlling User Session Access to Instances in AWS Systems Manager Session Manager

SEC06-BP04 验证软件完整性

使用加密验证来验证工作负载使用的软件构件（包括映像）的完整性。 对软件进行加密签名，以防在计算环境中出现未经授权的更改。

期望结果：所有构件均从可信来源获得。供应商网站证书已通过验证。 下载的构件通过其签名进行加密验证。您自己的软件经过加密签名，并由您的计算环境进行验证。

常见反模式：

- 信任信誉良好的供应商网站，从中获取软件构件，但忽视证书过期通知。 在未确认证书有效的情况下就继续下载。

- 验证供应商网站证书，但是从这些网站下载的构件没有进行加密验证。
- 仅依靠摘要或哈希值来验证软件的完整性。 哈希值可用于确定构件未在原始版本的基础上进行修改，但不能证实其来源正确。
- 不签署您自己的软件、代码或库，即使它们仅用于自己的部署。

建立此最佳实践的好处：验证工作负载所依赖的构件是否完整，这有助于防止恶意软件进入计算环境。 对软件进行签名有助于防止未经授权的软件在计算环境中运行。 通过签署和验证代码，保护软件供应链。

在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

操作系统映像、容器映像和代码构件通常在分发时提供完整性检查，例如通过摘要或哈希值进行检查。 这样，客户端就可以通过计算自己的有效负载哈希值，并验证哈希值与发布的哈希值是否相同，来验证完整性。 虽然这些检查有助于验证有效负载是否未被篡改，但并不能证实有效负载来自原始来源（数据出处）。 验证数据出处时，需要有可信机构签发的证书对构件进行了数字签名。

如果在工作负载中使用下载的软件或构件，请检查提供商是否提供了用于验证数字签名的公钥。 以下这些示例说明 AWS 如何为我们发布的软件提供公钥和验证说明：

- EC2 Image Builder: Verify the signature of the AWSTOE installation download
- AWS Systems Manager：验证 SSM Agent 签名
- Amazon CloudWatch：验证 CloudWatch 代理软件包的签名

将数字签名验证过程纳入您用于获取和强化映像的流程中，如 SEC06-BP02 从强化映像预置计算中所述。

您可以使用 AWS Signer 来协助管理签名验证过程，以及您自己的软件和构件的代码签名生命周期。 AWS Lambda 和 Amazon Elastic Container Registry 均实现了与 Signer 的集成，能够验证代码和映像的签名。 您可以参考“资源”部分中的示例，将 Signer 纳入持续集成和持续交付（CI/CD，Continuous Integration and Delivery）管道，以便自动验证签名并签署自己的代码和映像。

资源

相关文档：

- Cryptographic Signing for Containers

- Best Practices to help secure your container image build pipeline by using AWS Signer
- Announcing Container Image Signing with AWS Signer and Amazon EKS
- 为 AWS Lambda 配置代码签名
- Best practices and advanced patterns for Lambda code signing
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys

相关示例：

- Automate Lambda code signing with Amazon CodeCatalyst and AWS Signer
- Signing and Validating OCI Artifacts with AWS Signer

相关工具：

- AWS Lambda
- AWS Signer
- AWS Certificate Manager
- AWS Key Management Service
- AWS CodeArtifact

SEC06-BP05 自动保护计算

自动执行保护计算的操作，减少人工干预的需要。使用自动扫描检测计算资源中是否可能存在问题，并通过自动程序化响应或实例集管理操作进行修复。 在您的 CI/CD 流程中融入自动化功能，以最新的依赖项来部署值得信赖的工作负载。

期望结果：由自动化系统对计算资源进行所有扫描和修补。您可以使用自动验证来检查软件映像和依赖项是否来自可信来源，以及是否被篡改。自动检查工作负载是否有最新的依赖项，并对工作负载进行签名，以便在 AWS 计算环境中建立信任。 一旦检测到不合规的资源，就会启动自动修复措施。

常见反模式：

- 遵循不可变基础设施的做法，但没有制定紧急修补或更换生产系统的解决方案。
- 使用自动化技术来修复配置错误的资源，但没有手动覆盖机制。 在某些情况下，您可能需要调整要求，并且在进行这些更改之前需要暂停自动化操作。

建立此最佳实践的好处：自动化操作可以降低未经授权访问和使用计算资源的风险。 这有助于防止错误配置对生产环境产生影响，检测错误配置，并在发生错误配置时对其进行修复。 自动化操作还有助于检测未经授权的访问和使用计算资源的情况，从而缩短响应时间。 这反过来又能够缩小问题的总体影响范围。

在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

您可以应用安全性支柱实践中描述的自动化操作来保护计算资源。SEC06-BP01 执行漏洞管理描述了如何在 CI/CD 管道中使用 Amazon Inspector，以及如何持续扫描运行时环境，查看是否存在已知的通用漏洞披露（CVE，Common Vulnerabilities and Exposures）。 您可以遵循自动化运行手册，使用 AWS Systems Manager 来应用补丁或者利用新映像重新部署，从而始终使用最新的软件和库来更新计算实例集。 使用这些技术可减少对人工流程和交互式访问计算资源的需求。 请参阅《SEC06-BP03 减少人工管理工作和交互式访问》，了解更多信息。

自动化操作在部署值得信赖的工作负载方面也发挥着作用，如《SEC06-BP02 从强化映像预置计算》和《SEC06-BP04 验证软件完整性》中所述。 您可以使用 EC2 Image Builder、AWS Signer、AWS CodeArtifact 和 Amazon Elastic Container Registry（ECR）等服务，来下载、验证、构造和存储经过强化和批准的映像和代码依赖项。 通过与 Inspector 配合使用，这些服务都可以在您的 CI/CD 流程中发挥作用，这样您的工作负载只有在确认其依赖项是最新的且来自可信来源时，才会进入生产阶段。 工作负载还经过签名，这样 AWS Lambda 和 Amazon Elastic Kubernetes Service（EKS）等 AWS 计算环境就能在允许工作负载运行之前，验证工作负载是否未被篡改。

除了这些预防性控制措施之外，您还可以在计算资源的检测性控制中运用自动化。 例如，AWS Security Hub 提供 NIST 800-53 Rev. 5 标准，其中包括 [EC2.8] EC2 实例应使用实例元数据服务版本 2（IMDSv2）等检查内容。 IMDSv2 使用会话验证、阻止包含 X-Forwarded-For HTTP 标头的请求，以及设置网络 TTL 为 1 等技术，来阻止来自外部来源的流量检索有关 EC2 实例的信息。Security Hub 中的这项检查可检测 EC2 实例何时使用 IMDSv1，并启动自动修复措施。参阅《SEC04-BP04 启动对不合规资源的修复》，了解有关自动化检测和修复的更多信息。

实施步骤

1. 利用 EC2 Image Builder 自动创建安全、合规且经过强化的 AMI。 您可以在基础 AWS 和 APN 合作伙伴映像中融入符合互联网安全中心（CIS，Center for Internet Security）基准或安全技术实施指南（STIG，Security Technical Implementation Guide）标准的控制措施，从而生成自己的映像。
2. 自动执行配置管理。使用配置管理服务或工具，在计算资源中自动实施和验证安全配置。
   a. 使用 AWS Config 自动执行配置管理
   b. 使用 AWS Security Hub 自动管理安全性和合规性态势

3. 自动修补或替换 Amazon Elastic Compute Cloud（Amazon EC2）实例。AWSSystems Manager Patch Manager 使用安全相关的更新和其它类型的更新自动执行修补托管实例的流程。您可以使用 Patch Manager 来应用操作系统和应用程序的补丁。
   a. AWS Systems Manager Patch Manager
4. 自动扫描计算资源以便查找通用漏洞披露（CVE，Common Vulnerabilities and Exposures），并在构建管道中嵌入安全扫描解决方案。
   a. Amazon Inspector
   b. ECR 映像扫描
5. 考虑使用 Amazon GuardDuty 自动检测恶意软件和威胁，以便保护计算资源。在 AWS 环境中调用 AWS Lambda 函数时，GuardDuty 还可以识别出潜在问题。
   a. Amazon GuardDuty
6. 考虑采用 AWS 合作伙伴解决方案。AWS合作伙伴提供业界领先的产品，这些产品与您的本地环境中的现有控制措施等效、相同或与之集成。这些产品对现有 AWS 服务起到补充作用，使您能够在云端和本地环境中部署全面的安全架构，进而实现无缝效果更好的体验。
   a. 基础设施安全性

资源

相关最佳实践：

- SEC01-BP06 自动部署标准安全控制措施

相关文档：

- Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure

相关视频：

- Security best practices for the Amazon EC2 instance metadata service

数据保护

问题

- SEC 7. 如何对数据进行分类？
- SEC 8. 您如何保护静态数据？

- SEC 9. 您如何保护传输中数据？

SEC 7. 如何对数据进行分类？

分类提供了一种基于重要程度和敏感度对数据进行分类的方法，以帮助您确定适当的保护和保留控制措施。

最佳实践

- SEC07-BP01 了解数据分类方案
- SEC07-BP02 根据数据敏感性应用数据保护控制措施
- SEC07-BP03 自动识别和分类
- SEC07-BP04 定义可扩展的数据生命周期管理

SEC07-BP01 了解数据分类方案

了解工作负载正在处理的数据的分类、数据处理要求、相关业务流程、数据存储位置以及数据所有者是谁。 您的数据分类和处理方案应考虑工作负载的适用法律和合规性要求，以及需要采取哪些数据控制措施。了解数据是数据分类之旅的第一步。

期望结果：您的工作负载中的数据类型已得到充分了解并记录在案。 根据敏感数据的分类，采取适当的控制措施来保护敏感数据。 这些控制措施要考虑的因素包括：谁被允许访问数据以及访问的目的、数据的存储位置、数据的加密策略以及加密密钥的管理方式、数据的生命周期及其留存要求、适当的销毁流程、备份和恢复流程以及访问审计。

常见反模式：

- 没有正式的数据分类策略来定义数据敏感性级别及其处理要求
- 没有充分了解工作负载中数据的敏感性级别，也没有在架构和操作文档中记录这些信息
- 未能按照数据分类和处理策略的规定，根据数据的敏感性和要求，对数据采取适当的控制措施
- 未能向策略所有者提供有关数据分类和处理要求的反馈。

建立此最佳实践的好处：这种实践消除了工作负载中数据适当处理方面的模糊性。 运用正式策略来定义组织中数据的敏感性级别及其所需的保护措施，这有助于您遵守法律法规以及其它网络安全证明和认证。 工作负载所有者可以放心地了解敏感数据的存储位置和保护控制措施。 将这些内容记录在文档中，有助于团队新成员更好地理解这些内容，并在任职初期保持控制。这些实践还有助于通过合理调整各类数据的控制措施来降低成本。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在设计工作负载时，您可能会直观地考虑保护敏感数据的方法。 例如，在多租户应用程序中，直观的做法是将每个租户的数据视为敏感数据，并采取保护措施，使一个租户无法访问另一个租户的数据。 同样，您也可以直观地设计访问控制措施，只有管理员可以修改数据，而其他用户只有读取级访问权限，或者根本没有访问权限。

通过在策略中定义和记录这些数据敏感性级别及其数据保护要求，您可以正式确定工作负载中存在哪些数据。然后，您可以确定是否制定了正确的控制措施，是否可以对控制措施进行审计，以及在发现数据处理不当时应采取哪些适当的应对措施。

为了有助于确定敏感数据在工作负载中的位置，请考虑使用数据目录。数据目录是一个数据库，用于映射组织中的数据、其位置、敏感度级别以及为保护这些数据而设置的控制措施。此外，如果可用，请考虑使用 resource tags。 例如，对于受保护健康信息（PHI），您可以应用标签键为 `Classification`、标签值为 `PHI` 的标签，以及标签键为 `Sensitivity`、标签值为 `High` 的标签。 然后，AWS Config 等服务可用于监控这些资源的更改，并在这些资源被修改导致其不符合保护要求（例如更改加密设置）时发出警报。 您可以使用标签策略（AWS Organizations 的一项功能），来获取标签键和可接受值的标准定义。建议标签键或标签值不要包含私有或敏感数据。

实施步骤

1. 了解组织的数据分类方案和保护要求。
2. 确定工作负载处理的敏感数据的类型。
3. 在数据目录中捕获数据，数据目录可提供数据在组织中的位置以及该数据的敏感度级别的单一视图。
4. 考虑在可用的情况下使用资源和数据级标记，来标记数据的敏感性级别以及其它有助于监控和响应事件的操作元数据。
   a. AWS Organizations 标签策略可用于执行标记标准。

资源

相关最佳实践：

- SUS04-BP01 实施数据分类策略

相关文档：

- Data Classification whitepaper
- 标记 AWS 资源的最佳实践

相关示例：

- AWS Organizations Tag Policy Syntax and Examples

相关工具

- AWS 标签编辑器

SEC07-BP02 根据数据敏感性应用数据保护控制措施

应用数据保护控制措施，为分类策略中定义的每一类数据提供适当水平的控制。 这种做法可以保护敏感数据，防止在未经授权的情况下访问和使用敏感数据，同时保持数据可用。

期望结果：您有一项分类策略，它定义了组织内数据的不同敏感性级别。 对于每个敏感性级别，您都有明确的指导原则，规定了经批准的存储和处理服务、位置及其所需的配置。 您可以根据所需的保护级别和相关成本，实施每个级别的控制措施。 如果数据出现在未经授权的位置、在未经授权的环境中处理、被未经授权的行为者访问，或者相关服务的配置变得不合规，您都能够进行监控并发出警报。

常见反模式：

- 对所有数据应用相同级别的保护控制措施。这可能导致为低敏感性数据预配过多的安全控制措施，或者对高敏感性数据保护不足。
- 在定义数据保护控制措施时，没有让安全、合规和业务团队的利益相关方参与进来。
- 忽视与实施和维护数据保护控制措施相关的运营开销和成本。
- 不定期进行数据保护控制措施审查，无法保持一直符合分类策略。
- 没有有关静态数据和传输中数据所在位置的完整清单。

建立此最佳实践的好处：贵组织通过根据数据分类级别调整控制措施，能够在需要时投资更高级别的控制措施。可能包括增加用于保护、监控、测量、修复和报告的资源。 在适合减少控制措施的情况下，您可以为员工、客户或成员提高数据的可访问性和完整性。 这种方法既能让贵组织在数据使用方面获得极大的灵活性，又能遵守数据保护要求。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

根据数据敏感性级别实施数据保护控制措施时，涉及几个关键步骤。首先，确定工作负载架构中不同的数据敏感性级别（如公开、内部、保密和受限），并评估在哪里存储和处理这些数据。接下来，根据数据的敏感性级别定义数据的隔离边界。我们建议您将数据分别放置到不同的 AWS 账户 中，使用服务控制策略（SCP）来限制每个数据敏感性级别所允许的服务和操作。这样，您就可以创建强大的隔离边界，并执行最低权限原则。

定义隔离边界后，根据数据敏感性级别实施适当的保护控制措施。参考保护静态数据和保护传输中数据中的最佳实践，实施加密、访问控制和审计等相关控制措施。考虑采用令牌化或匿名化等技术，来降低数据的敏感性级别。利用集中式令牌化和去令牌化系统，简化在整个企业中应用一致数据策略的过程。

持续监控和测试所实施控制措施的有效性。随着贵组织数据状况和威胁的变化，定期审查和更新数据分类方案、风险评估和保护控制措施。使所实施的数据保护控制措施符合相关行业法规、标准和法律要求。此外，培养安全意识和提供培训，让员工了解数据分类方案及其在处理和保护敏感数据方面的责任。

实施步骤

1. 确定工作负载中数据的分类和敏感性级别。
2. 规定每个级别的隔离边界，并确定执行策略。
3. 评估您定义的控制措施，这些控制措施管理访问、加密、审计、留存以及数据分类策略所要求的其它事项。
4. 评估在适当情况下降低数据敏感性级别的方案，例如采用令牌化或匿名化。
5. 自动测试和监控已配置资源来验证控制措施。

资源

相关最佳实践：

- PERF03-BP01 使用最能满足数据访问和存储要求的专用数据存储
- COST04-BP05 执行数据留存策略

相关文档：

- Data Classification whitepaper

- Best Practices for Security, Identify, & Compliance
- AWS KMS 最佳实践
- Encryption best practices and features for AWS services

相关示例：

- Building a serverless tokenization solution to mask sensitive data
- How to use tokenization to improve data security and reduce audit scope

相关工具：

- AWS Key Management Service (AWS KMS)
- AWS CloudHSM
- AWS Organizations

SEC07-BP03 自动识别和分类

自动识别和分类数据可帮助您实施正确的控制措施。使用自动化技术来增强人工判断，可降低人为出错和暴露的风险。

期望结果：您能够根据自己的分类和处理策略，来验证是否有适当的控制措施。自动化工具和服务有助于您识别数据的敏感性级别并加以分类。 自动化技术还有助于您持续监控环境，以便检测数据是否以未经授权的方式存储或处理，并发出警报，从而能够迅速采取纠正措施。

常见反模式：

- 完全依赖人工流程进行数据识别和分类，既容易出错又耗费时间。 这可能导致数据分类效率低下且不稳定，尤其是在数据量不断增长的情况下。
- 缺乏机制，无法跟踪和管理整个组织内的数据资产。
- 忽视了数据在组织内部移动和演变时，对数据进行持续监控和分类的需求。

建立此最佳实践的好处：数据识别和分类自动化可使数据保护控制措施的应用更加稳定和准确，从而降低人为出错的风险。 自动化技术还可以提供敏感数据访问和移动操作的可见性，有助于您检测到未经授权的处理并采取纠正措施。

在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

在工作负载的初始设计阶段，通常运用人工判断对数据进行分类，但作为一种预防性控制机制，应考虑建立若干系统，以期对测试数据进行自动识别和分类。例如，可以向开发人员提供工具或服务对代表性数据进行扫描，以便确定数据敏感性。 在 AWS 中，您可以将数据集上传至 Amazon S3，并使用 Amazon Macie、Amazon Comprehend 或 Amazon Comprehend Medical 对数据进行扫描。 同样，考虑在单元测试和集成测试中纳入数据扫描，以便检测哪里的敏感数据不在预期之内。如果在这一阶段对敏感数据发出警报，就能够在部署到生产环境之前突出保护方面的漏洞。AWS Glue、Amazon SNS 和 Amazon CloudWatch 中的敏感数据检测等其它功能也可用于检测 PII 并采取缓解措施。对于任何自动化工具或服务，都要了解其如何定义敏感数据，并根据需要使用其它人工或自动化解决方案来解决任何漏洞。

作为一种检测性控制措施，利用对环境的持续监控来检测敏感数据是否以不合规的方式存储。 这有助于检测一些情况，例如，敏感数据是否被发送到日志文件或者复制到数据分析环境，而没有进行适当的去标识化或修订。 可使用 Amazon Macie 对存储在 Amazon S3 中的数据进行持续监控，以便发现敏感数据。

实施步骤

1. 查看 SEC07-BP01 中介绍的组织内数据分类方案。
    a. 通过了解贵组织的数据分类方案，您可以建立与公司策略相一致的准确的自动识别和分类流程。
2. 对环境进行初步扫描，以便自动识别和分类。
    a. 对数据初步进行全面扫描有助于全面了解敏感数据在环境中的位置。如果最初不需要进行全面扫描，或者由于成本原因无法提前完成扫描，则应评估数据采样技术是否适合实现您的结果。例如，可以对 Amazon Macie 进行配置，以便在 S3 存储桶中执行广泛的自动敏感数据发现操作。 该功能利用采样技术，对敏感数据的位置进行初步分析，成本效益高。 然后，可以使用敏感数据发现作业对 S3 存储桶进行更深入的分析。其它数据存储也可以导出到 S3，由 Macie 扫描。
    b. 为扫描中识别的数据存储资源建立在 SEC07-BP02 中定义的访问控制。
3. 配置对环境的持续扫描。
    a. Macie 的自动敏感数据发现功能可用于对环境进行持续扫描。 可使用 Macie 中的允许列表排除已授权存储敏感数据的已知 S3 存储桶。
4. 将识别和分类纳入构建和测试流程。
    a. 确定开发人员可用于在开发工作负载时扫描数据敏感性的工具。 在集成测试过程中使用这些工具，在敏感数据意外出现时发出警报，并阻止继续部署。
5. 实施系统或运行手册，以便在未经授权的位置发现敏感数据时采取行动。

a. 使用自动修复功能来限制对数据的访问。例如，如果您使用基于属性的访问权限控制（ABAC），则可以将此数据移到访问受限的 S3 存储桶，或者为对象添加标签。此外，可以考虑在检测到数据时对其进行屏蔽。

b. 提醒您的数据保护和事件响应团队调查事件的根本原因。他们汲取的任何经验教训都有助于预防未来的事件。

资源

相关文档：

- AWS Glue：检测和处理敏感数据
- 在 Amazon SNS 中使用托管数据标识符
- Amazon CloudWatch Logs：通过屏蔽帮助保护敏感的日志数据

相关示例：

- Enabling data classification for Amazon RDS database with Macie
- Detecting sensitive data in DynamoDB with Macie

相关工具：

- Amazon Macie
- Amazon Comprehend
- Amazon Comprehend Medical
- AWS Glue

SEC07-BP04 定义可扩展的数据生命周期管理

了解您的数据生命周期要求，因为这些要求与不同的数据分类等级和处理方式密切相关。 这可能包括数据首次进入您的环境时的处理方式、数据转换方式以及数据销毁规则。必须考虑数据的留存期限、访问、审计和跟踪溯源等因素。

期望结果：您可以在尽可能接近摄取点和摄取时间的情况下对数据进行分类。当数据分类需要执行屏蔽、令牌化或其它降低敏感性级别的处理时，您可以在尽可能接近摄取点和摄取时间的情况下执行这些操作。

根据数据分类情况，当数据不再适合保留时，您可以按照策略删除数据。

常见反模式：

- 采用“一刀切”的数据生命周期管理方法，而不考虑不同的敏感性级别和访问权限要求。
- 仅从可用数据或备份数据的角度考虑生命周期管理，而不是两者兼顾。
- 在未确定数据价值或出处的情况下，就假定进入您工作负载的数据是有效的。
- 依赖数据持久性来替代数据备份和保护。
- 在数据超过其时效性和必要的留存期限之后，仍然保留数据。

建立此最佳实践的好处：明确定义且可扩展的数据生命周期管理策略有助于保持监管合规性，提高数据安全性，优化存储成本，并在保持适当控制的同时实现高效的数据访问和共享。

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

工作负载中的数据通常是动态变化的。 数据在进入工作负载环境时所采用的形式，可能不同于数据在业务逻辑、报告、分析或机器学习中存储或使用时的形式。 此外，数据的价值可能会随着时间的推移而变化。有些数据本质上具有时效性，会随着时间的推移而失去价值。 考虑数据的这些变化对于数据分类方案和相关控制措施下的评估有何影响。 尽可能使用自动生命周期机制（例如 Amazon S3 生命周期策略和 Amazon Data Lifecycle Manager），来配置数据留存、归档和过期流程。对于存储在 DynamoDB 中的数据，可以使用生存时间（TTL）功能来定义每个项目的过期时间戳。

区分可供使用的数据和作为备份存储的数据。 考虑使用 AWS Backup 来自动备份跨 AWS 服务的数据。 Amazon EBS 快照提供了一种使用 S3 功能（包括生命周期、数据保护和访问保护机制）复制 EBS 卷并存储 EBS 卷的方法。S3 对象锁定和 AWS Backup 保管库锁就是其中的两种保护机制，可以为您的备份提供额外的安全防御和控制。管理明确的职责分工和备份访问权限。在账户级别隔离所有备份，以便在事件发生期间与受影响环境保持隔离。

生命周期管理的另一个方面是记录数据在工作负载中进展的历史，即数据溯源跟踪。该功能可以让您确信，您知道数据来自何处、执行过哪些转换、更改是由哪位所有者或流程执行的以及在何时做的更改。 掌握这些历史记录后，有助于在潜在安全事件中解决问题和进行调查。 例如，您可以在 Amazon DynamoDB 表中记录有关转换的元数据。 在数据湖中，您可以针对每个数据管道阶段，在不同的 S3 存储桶中保存转换后数据的副本。在 AWS Glue Data Catalog 中存储架构和时间戳信息。 无论采用哪种解决方案，都需要考虑最终用户的需求，以便确定报告数据溯源情况所需的适当工具。 这样有助于您确定如何以合适的方式跟踪数据溯源情况。

实施步骤

1. 分析工作负载的数据类型、敏感性级别和访问权限要求，对数据进行分类，并制定适当的生命周期管理策略。
2. 设计并实施符合法律、监管和组织要求的数据留存策略及自动销毁流程。
3. 建立流程和自动化机制，以便能够根据工作负载要求和监管的变化，持续监控、审计和调整数据生命周期管理策略、控制措施及政策。
   a. 使用 AWS Config 检测未开启自动生命周期管理的资源

资源

相关最佳实践：

- COST04-BP05 执行数据留存策略
- SUS04-BP03 使用策略管理数据集的生命周期

相关文档：

- Data Classification Whitepaper
- AWS Blueprint for Ransomware Defense
- DevOps Guidance: Improve traceability with data provenance tracking

相关示例：

- How to protect sensitive data for its entire lifecycle in AWS
- Build data lineage for data lakes using AWS Glue, Amazon Neptune, and Spline

相关工具：

- AWS Backup
- Amazon Data Lifecycle Manager
- AWS Identity and Access Management Access Analyzer

SEC 8. 您如何保护静态数据？

通过实施多种控制措施来保护静态数据，以降低未经授权的访问或处理不当的风险。

最佳实践

- SEC08-BP01 实施安全密钥管理
- SEC08-BP02 强制实施静态加密
- SEC08-BP03 自动执行静态数据保护
- SEC08-BP04 强制实施访问控制

SEC08-BP01 实施安全密钥管理

安全密钥管理包括密钥材料的存储、轮换、访问控制和监控，这些都是保护工作负载的静态数据安全所必需的。

期望结果：您拥有一种可扩展、可重复且自动化的密钥管理机制。该机制对密钥材料强制实施最低权限访问，并在密钥可用性、机密性和完整性之间提供适当的平衡。您可以监控对密钥的访问权限，如果需要轮换密钥材料，则使用自动流程轮换密钥材料。您不让人工操作员访问密钥材料。

常见反模式：

- 由人类访问未加密的密钥材料。
- 创建自定义加密算法。
- 访问密钥材料的权限过于宽泛。

建立此最佳实践的好处：通过为您的工作负载建立安全的密钥管理机制，您可以帮助保护您的内容免遭未经授权的访问。此外，您可能需要遵守对数据进行加密的监管要求。有效的密钥管理解决方案可以提供符合这些法规的技术机制，进而保护密钥材料。

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

静态数据加密是一项基本的安全控制措施。为实施这种控制措施，工作负载需要一种机制，来安全地存储和管理用于加密静态数据的密钥材料。

AWS 提供的 AWS Key Management Service（AWS KMS）可为 AWS KMS 密钥提供持久、安全和冗余的存储。许多 AWS 服务都与 AWS KMS 集成来支持对您的数据进行加密。AWS KMS 使用经 FIPS

140-2 Level 3 验证的硬件安全模块来保护您的密钥。不存在以纯文本格式导出 AWS KMS 密钥的机制。

使用多账户策略部署工作负载时，应将 AWS KMS 密钥与使用这些密钥的工作负载保存在同一个账户中。This distributed model 将管理 AWS KMS 密钥的责任交给您的团队。在其它用例中，贵组织可以选择将 AWS KMS 密钥存储到集中式账户中。这种集中式结构需要额外的策略，以实现工作负载账户访问集中式账户中存储的密钥所需的跨账户访问，但可能更适用于多个 AWS 账户 共享单个密钥的用例。

无论密钥材料存储在哪里，都应通过使用 key policies 和 IAM 策略来严格控制对密钥的访问。密钥策略是控制对 AWS KMS 密钥的访问权限的主要方式。此外，AWS KMS 密钥授权可以提供对 AWS 服务的访问权限，从而代表您加密和解密数据。请查看 guidance for access control to your AWS KMS keys。

您应监控加密密钥的使用情况，以检测异常的访问规律。使用 AWS 托管密钥和 AWS KMS 中存储的客户自主管理型密钥执行的操作可以记录在 AWS CloudTrail 中，并应定期进行审查。特别注意监控密钥销毁事件。为了减少意外或恶意破坏密钥材料的情况，密钥销毁事件不会立即删除密钥材料。尝试删除 AWS KMS 中的密钥时会经历一个 waiting period，默认为 30 天且最少为 7 天，这让管理员有时间审核这些操作并在必要时回滚请求。

大多数 AWS 服务使用 AWS KMS 的方式对您来说都是透明的，您只需要决定是使用 AWS 托管密钥还是客户自主管理型密钥。如果您的工作负载需要直接使用 AWS KMS 来加密或解密数据，则应使用 envelope encryption 来保护您的数据。此 AWS 加密开发工具包可为您的应用程序提供客户端加密原语，来实施信封加密并与 AWS KMS 集成。

实施步骤

1. 为密钥确定合适的密钥管理选项（AWS 托管或客户自主管理）。
    a. 为便于使用，AWS 为大多数服务提供 AWS 自有和 AWS 托管密钥，这样，无需管理密钥材料或密钥策略，即可提供静态加密功能。
    b. 使用客户自主管理型密钥时，请考虑使用默认密钥存储，以便在敏捷性、安全性、数据主权和可用性之间取得最佳平衡。其他用例可能需要使用附带 AWS CloudHSM 的自定义密钥存储或使用外部密钥存储。
2. 查看您用于工作负载的服务列表，以了解 AWS KMS 如何与该服务集成。例如，EC2 实例可以使用加密的 EBS 卷，验证从这些卷创建的 Amazon EBS 快照是否也使用客户自主管理型密钥进行加密，并减少未加密快照数据的意外泄露。
    a. AWS 服务如何使用 AWS KMS

   b. 有关 AWS 服务提供的加密选项的详细信息，请参阅该服务的用户指南或开发人员指南中的“静态加密”主题。
3. 实施 AWS KMS：AWS KMS 使您可以轻松创建和管理密钥，并控制各种 AWS 服务和应用程序中的加密使用情况。
   a. 入门：AWS Key Management Service（AWS KMS）
   b. 请查看 AWS KMS 密钥访问控制的最佳实践。
4. 考虑使用 AWS Encryption SDK：当应用程序需要在客户端加密数据时，使用包含 AWS KMS 集成的 AWS Encryption SDK。
   a. AWS Encryption SDK
5. 启用 IAM Access Analyzer 以自动审查是否存在过于宽泛的 AWS KMS 密钥策略并相应地发出通知。
   a. 考虑使用 custom policy checks，来验证资源策略更新是否不会授予对 KMS 密钥的公有访问权限。
6. 启用 Security Hub 以便在密钥策略配置错误、计划删除密钥或存在未启用自动轮换的密钥时，接收通知。
7. 确定适合您的 AWS KMS 密钥的日志记录级别。由于对 AWS KMS 的调用（包括只读事件）会被记录下来，因此与 AWS KMS 关联的 CloudTrail 日志可能会变得非常庞大。
   a. 一些组织倾向于将 AWS KMS 日志活动分成单独的跟踪。有关更多详细信息，请参阅《AWS KMS 开发者指南》的使用 CloudTrail 记录 AWS KMS API 调用部分。

资源

相关文档：

- AWS Key Management Service
- AWS 加密服务和工具
- 利用加密保护 Amazon S3 数据
- 信封加密
- 数字主权承诺
- 揭开 AWS KMS 密钥操作的神秘面纱、自带密钥、自定义密钥库和加密文字可移植性
- AWS Key Management Service 加密详情

相关视频：

- AWS 中的加密原理
- 在 AWS 上保护您的数据块存储
- AWS data protection: Using locks, keys, signatures, and certificates

相关示例：

- 使用 AWS KMS 实施高级访问控制机制

SEC08-BP02 强制实施静态加密

加密私有静态数据，来保持机密性并提供额外的一层防护，以防止无意的数据泄露或外流。加密可保护数据，因此，如果不首先对数据进行解密，则无法读取或访问加密的数据。清点和控制未加密的数据，以降低与数据泄露关联的风险。

期望结果：您拥有在私有数据处于静态时默认加密这些数据的机制。这些机制有助于保持数据的机密性，并提供额外的一层防护，以防止无意的数据泄露或外流。您可以维护未加密数据的清单，并了解为保护这些数据而采取的控制措施。

常见反模式：

- 不使用默认加密配置。
- 提供对解密密钥过于宽松的访问权限。
- 不监控加密和解密密钥的使用。
- 未加密便存储数据。
- 对所有数据使用相同的加密密钥，而不考虑数据用途、类型和分类。

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

将加密密钥映射到工作负载中的数据分类。当对数据使用单个或非常少量的加密密钥时，这种方法有助于防止过于宽松的访问（请参阅 SEC07-BP01 了解数据分类方案）。

AWS Key Management Service（AWS KMS）与许多 AWS 服务集成，使加密静态数据更加轻松。例如，在 Amazon Elastic Compute Cloud（Amazon EC2）中，您可以对账户设置默认加密，以便自动加密新的 EBS 卷。使用 AWS KMS 时，请考虑需要对数据进行多严格的限制。默认和服务控制的 AWS KMS 密钥由 AWS 代表您进行管理和使用。对于需要对底层加密密钥进行精细访问的敏感数据，

请考虑使用客户自主管理型密钥（CMK）。您可以完全控制 CMK，包括通过使用密钥策略进行轮换和访问管理。

此外，诸如 Amazon Simple Storage Service（Amazon S3）之类的服务现在默认加密所有新对象。这种实施提供了增强的安全性，而不会对性能产生任何影响。

其它服务，例如 Amazon Elastic Compute Cloud（Amazon EC2）或 Amazon Elastic File System（Amazon EFS），则支持默认加密设置。还可以使用 AWS Config 规则 来自动检查您是否正在为贵组织内的 Amazon Elastic Block Store (Amazon EBS) volumes、Amazon Relational Database Service (Amazon RDS) instances、Amazon S3 buckets 和其它服务使用加密。

AWS 还提供客户端加密选项，使您能够在将数据上传到云之前对数据进行加密。AWS Encryption SDK 提供了一种使用信封加密对数据进行加密的方法。您提供包装密钥，AWS Encryption SDK 为它加密的每个数据对象生成一个唯一数据密钥。如果需要托管的单租户硬件安全模块（HSM），请考虑 AWS CloudHSM。AWS CloudHSM 使您可在通过 FIPS 140-2 Level 3 验证的 HSM 上生成、导入和管理加密密钥。AWS CloudHSM 的一些使用案例包括保护用于签发证书颁发机构（CA）的私有密钥，以及为 Oracle 数据库启用透明数据加密（TDE）。AWS CloudHSM 客户端开发工具包提供的软件使您可在将数据上传到 AWS 之前，使用存储在 AWS CloudHSM 中的密钥对客户端数据进行加密。Amazon DynamoDB Encryption Client 还使您可在将项目上传到 DynamoDB 表之前，对项目进行加密和签名。

实施步骤

- 为新的 Amazon EBS 卷配置默认加密：指定所有新创建的 Amazon EBS 卷要以加密形式创建，并选择使用 AWS 提供的默认密钥或您创建的密钥。
- 配置加密亚马逊机器映像（AMI）：通过复制已配置加密功能的现有 AMI，可自动加密根卷和快照。
- 配置 Amazon RDS 加密：通过使用加密选项，配置对您的 Amazon RDS 数据库集群和静态快照的加密。
- 使用策略创建和配置 AWS KMS 密钥，以限制对每个数据分类的相应主体的访问：例如，创建一个 AWS KMS 密钥用于加密生产数据，创建一个不同的密钥用于加密开发或测试数据。您还可以提供对其他 AWS 账户 的密钥访问权限。不妨考虑分开设立开发环境和生产环境的账户。如果您的生产环境需要解密开发账户中的构件，您可以编辑用于加密开发构件的 CMK 策略，使生产账户有能力解密这些构件。然后，生产环境可以摄取解密后的数据以用于生产。
- 在其它 AWS 服务中配置加密：对于您使用的其它 AWS 服务，请查看该服务的安全性文档，以确定该服务的加密选项。

资源

相关文档：

- AWS 加密工具
- AWS Encryption SDK
- AWS KMS 加密详情白皮书
- AWS Key Management Service
- AWS 加密服务和工具
- Amazon EBS Encryption
- Amazon EBS 卷的默认加密
- 加密 Amazon RDS 资源
- 如何为 Amazon S3 存储桶启用默认加密？
- 利用加密保护 Amazon S3 数据

相关视频：

- AWS 中的加密原理
- 在 AWS 上保护您的数据块存储

SEC08-BP03 自动执行静态数据保护

使用自动化技术来验证和执行静态数据控制。 使用自动扫描功能来检测数据存储解决方案的错误配置，并在可能的情况下通过自动程序化响应进行修复。 在 CI/CD 流程中融入自动化功能，以便在数据存储部署到生产环境之前检测出错误配置。

期望结果：自动化系统对数据存储位置进行扫描和监控，防止出现控制措施配置错误、未经授权的访问和意外使用。 检测到配置错误的存储位置后，自动修复措施就会启动。 自动化流程可创建数据备份，并在原始环境之外存储不可更改的副本。

常见反模式：

- 在支持的情况下，不考虑通过默认设置启用加密的选项。
- 在制定自动备份和恢复策略时，除操作事件外，不考虑安全事件。
- 不对存储服务执行公共访问设置。

- 不监控和审计保护静态数据的控制措施。

建立此最佳实践的好处：自动化有助于防止错误配置数据存储位置的风险。这有助于防止错误配置进入生产环境。这种最佳实践还有助于在发生错误配置时进行检测和修复。

在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

自动化是整个静态数据保护实践的主题。SEC01-BP06 自动部署标准安全控制措施介绍如何使用基础设施即代码（IaC）模板（例如 AWS CloudFormation）捕获资源配置。 这些模板已提交到版本控制系统，并用于通过 CI/CD 管道在 AWS 上部署资源。 这些技术同样适用于自动配置数据存储解决方案，如 Amazon S3 存储桶的加密设置。

您可以在 CI/CD 管道中使用 AWS CloudFormation Guard 中的规则，来检查您在 IaC 模板中定义的设置是否存在配置错误。 您可以使用 AWS Config 监控 CloudFormation 或其它 IaC 工具中尚未提供的设置，以防配置错误。 如 SEC04-BP04 启动对不合规资源的修复中所述，Config 针对错误配置生成的警报可自动修复。

将自动化功能融入权限管理策略，也是自动化数据保护不可或缺的组成部分。SEC03-BP02 授予最低访问权限和 SEC03-BP04 持续减少权限描述了如何配置最低权限访问策略，这些策略会受到 AWS Identity and Access Management Access Analyzer 的持续监控，以便得出何时可以减少权限的调查发现。 除了监控权限的自动化功能之外，您还可以配置 Amazon GuardDuty 以监控 EBS 卷（通过 EC2 实例）、S3 存储桶和受支持的 Amazon Relational Database Service 数据库的异常数据访问行为。

在检测敏感数据何时存储在未经授权的位置时，自动化功能也能发挥作用。SEC07-BP03 自动识别和分类描述了 Amazon Macie 如何监控您的 S3 存储桶中是否存在意外敏感数据，并生成可启动自动响应的警报。

按照 REL09 备份数据中的做法，制定自动数据备份和恢复策略。对于从安全事件中恢复和从操作事件中恢复，备份数据和恢复数据同样重要。

实施步骤

1. 在 IaC 模板中捕获数据存储配置。 在 CI/CD 管道中使用自动检查来检测错误配置。
    a. 您可以使用 AWS CloudFormation 来处理 IaC 模板，并使用 AWS CloudFormation Guard 来检查模板是否存在配置错误。
    b. 使用 AWS Config 以主动评估模式运行规则。使用此设置可在创建资源前，检查资源作为 CI/CD 管道中的一个步骤的合规性。

2. 监控资源中是否存在数据存储配置错误。
    a. 设置 AWS Config 以监控数据存储资源中控制措施配置的变化，并在检测到错误配置时生成警报，以便调用修复措施。
    b. 有关自动修复的更多指导，请参见 SEC04-BP04 启动对不合规资源的修复。
3. 通过自动化功能持续监控和减少数据访问权限。
    a. IAM Access Analyzer 可持续运行，在权限可能减少时发出警报。
4. 监控异常数据访问行为并发出警报。
    a. GuardDuty 可监控已知威胁特征，还监控 EBS 卷、S3 存储桶和 RDS 数据库等数据存储资源的基线访问行为偏差。
5. 对存储在意外位置的敏感数据进行监控并发出警报。
    a. 使用 Amazon Macie 持续扫描 S3 存储桶，查找敏感数据。
6. 自动对数据进行安全加密备份。
    a. AWS Backup 是一项托管服务，可为 AWS 上的各种数据来源创建加密和安全的备份。 弹性灾难恢复允许您复制完整的服务器工作负载，并保持持续的数据保护，恢复点目标（RPO）以秒为单位。 您可以配置这两项服务，使之协同工作，从而自动创建数据备份并复制到失效转移位置。 这有助于在受到操作或安全事件影响时保持数据可用。

资源

相关最佳实践：

- SEC01-BP06 自动部署标准安全控制措施
- SEC03-BP02 授予最低访问权限
- SEC03-BP04 持续减少权限
- SEC04-BP04 启动对不合规资源的修复
- SEC07-BP03 自动识别和分类
- REL09-BP02 保护并加密备份
- REL09-BP03 自动执行数据备份

相关文档：

- AWS Prescriptive Guidance: Automatically encrypt existing and new Amazon EBS volumes
- Ransomware Risk Management on AWS Using the NIST Cyber Security Framework (CSF)

相关示例：

- How to use AWS Config proactive rules and AWS CloudFormation Hooks to prevent creation of noncompliant cloud resources
- Automate and centrally manage data protection for Amazon S3 with AWS Backup
- AWS re:Invent 2023 - Implement proactive data protection using Amazon EBS snapshots
- AWS re:Invent 2022 - Build and automate for resilience with modern data protection

相关工具：

- AWS CloudFormation Guard
- AWS CloudFormation Guard 规则注册表
- IAM Access Analyzer
- Amazon Macie
- AWS Backup
- 弹性灾难恢复

SEC08-BP04 强制实施访问控制

为有助于保护静态数据，请使用隔离和版本控制等机制来强制实施访问控制。应用最低权限和条件访问控制。防止向公众授予访问您数据的权限。

期望结果：您验证只有获得授权的用户才能按照“需要知晓”的原则访问数据。您通过定期备份和版本控制来保护您的数据，以防止数据被有意或无意地修改或删除。您将关键数据与其它数据隔离，以保护其机密性和数据完整性。

常见反模式：

- 将具有不同敏感度要求或分类的数据存储在一起。
- 解密密钥的权限过于宽松。
- 数据分类不当。
- 不保留重要数据的详细备份。
- 提供对生产数据的持久访问。
- 未审计数据访问，也未定期检查权限。

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

保护静态数据对于维护数据的完整性、机密性以及符合监管要求非常重要。您可以实施多种控制措施来协助实现这一目标，包括访问控制、隔离、条件访问和版本控制。

您可以按照最低权限原则强制实施访问控制，该原则仅向用户和服务提供执行其任务所需的权限。这包括对于加密密钥的访问权限。查看您的 AWS Key Management Service (AWS KMS) policies，来验证您授予的访问权限级别是否适当，以及相关条件是否适用。

可以通过为每个分类级别使用不同的 AWS 账户，根据不同的分类级别分离数据，并使用 AWS Organizations 管理这些账户。这种隔离有助于防止未经授权的访问，并最大限度地降低数据泄露的风险。

定期审核在 Amazon S3 存储桶策略中授予的访问权限级别。除非绝对必要，否则请避免使用可公开读取或写入的存储桶。考虑使用 AWS Config 来检测可公开可用的存储桶，并使用 Amazon CloudFront 来提供 Amazon S3 中的内容。验证正确配置了应不支持公开访问的存储桶，以防止公开访问。

对存储在 Amazon S3 中的关键数据实施版本控制和对象锁定机制。Amazon S3 版本控制保留对象的先前版本，以便在意外删除或覆盖时恢复数据。Amazon S3 对象锁定为对象提供强制访问控制，从而防止对象在锁定到期之前被删除或覆盖，即使是根用户也是如此。此外，Amazon S3 Glacier Vault Lock 为存储在 Amazon S3 Glacier 中的归档提供了类似的功能。

实施步骤

1. 采用最低权限原则，强制实施访问控制：
   - 审核向用户和服务授予的访问权限，并验证他们是否只拥有执行其任务所需的权限。
   - 通过检查 AWS Key Management Service (AWS KMS) policies 来查看对加密密钥的访问权限。
2. 根据不同的分类级别将数据分开：
   - 对每个数据分类级别使用不同的 AWS 账户。
   - 使用 AWS Organizations 管理这些账户。
3. 审核 Amazon S3 存储桶和对象权限：
   - 定期审核在 Amazon S3 存储桶策略中授予的访问权限级别。
   - 除非绝对必要，否则请避免使用可公开读取或写入的存储桶。
   - 考虑使用 AWS Config 来检测公开可用的存储桶。
   - 使用 Amazon CloudFront 来提供 Amazon S3 中的内容。
   - 验证正确配置了应不支持公开访问的存储桶，以防止公开访问。

- 可以对使用 IAM 身份验证的数据库和任何其它数据来源（例如 SQS 或第三方数据存储）应用相同的审核流程。

4. 使用 AWS IAM Access Analyzer：
   - 您可以配置 AWS IAM Access Analyzer 来分析 Amazon S3 存储桶，并在 S3 策略向外部实体授予访问权限时生成调查发现。
5. 实施版本控制和对象锁定机制：
   - 使用 Amazon S3 版本控制来保留对象的先前版本，这样可以从意外删除或覆盖中恢复。
   - 使用 Amazon S3 对象锁定来为对象提供强制访问控制，从而防止对象在锁定到期之前被删除或覆盖，即使是根用户也是如此。
   - 将 Amazon S3 Glacier Vault Lock 用于存储在 Amazon S3 Glacier 中的归档。
6. 使用 Amazon S3 清单：
   - 可以使用 Amazon S3 清单来审计和报告 S3 对象的复制和加密状态。
7. 审核 Amazon EBS 和 AMI 共享权限：
   - 审核 Amazon EBS 和 AMI 共享的共享权限，来验证映像和卷是否未与工作负载外部的 AWS 账户共享。
8. 定期审核 AWS Resource Access Manager 共享：
   - 可以使用 AWS Resource Access Manager 来在 Amazon VPC 内共享资源，如 AWS Network Firewall 策略、Amazon Route 53 Resolver 规则和子网。
   - 定期审计共享的资源，并停止共享不再需要共享的资源。

资源

相关最佳实践：

- SEC03-BP01 定义访问要求
- SEC03-BP02 授予最低访问权限

相关文档：

- AWS KMS 加密详情白皮书
- 管理对 Amazon S3 资源的访问权限简介
- 管理对 AWS KMS 资源的访问权限概览
- AWS Config 规则

- Amazon S3 + Amazon CloudFront: A Match Made in the Cloud
- 使用版本控制
- 使用 Amazon S3 对象锁定以锁定对象
- 共享 Amazon EBS 快照
- 共享 AMI
- Hosting a single-page application on Amazon S3
- AWS 全局条件键
- Building a Data Perimeter on AWS

相关视频：

- 在 AWS 上保护您的数据块存储

SEC 9. 您如何保护传输中数据？

通过实施多种控制措施来保护传输中数据，以降低未经授权的访问或丢失的风险。

最佳实践

- SEC09-BP01 实施安全密钥和证书管理
- SEC09-BP02 在传输中执行加密
- SEC09-BP03 对网络通信进行身份验证

SEC09-BP01 实施安全密钥和证书管理

传输层安全性协议（TLS）证书用于保障网络通信的安全，确立网站、资源和工作负载在互联网上以及专用网络上的身份。

期望结果：一个安全的证书管理系统，可以在公钥基础设施（PKI，Public Key Infrastructure）中预置、部署、存储和续订证书。安全密钥和证书管理机制可防止证书私钥材料泄露，并定期自动续订证书。它还与其他服务集成，为工作负载内的计算机资源提供安全的网络通信和标识。密钥材料永远不应能够通过人员的身份来访问。

常见反模式：

- 在证书部署或续订流程中执行人工步骤。
- 在设计私有证书颁发机构（CA，Certificate Authority）时，对 CA 层次结构的关注不够。

- 对公共资源使用自签名证书。

建立此最佳实践的好处：

- 通过自动化的部署和续订流程简化证书管理
- 鼓励使用 TLS 证书对传输中数据进行加密
- 提高了证书颁发机构执行的证书操作的安全性和可审计性
- 在 CA 层次结构的不同层级上组织管理职责

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

现代化工作负载广泛使用通过 PKI 协议（如 TLS）进行加密的网络通信。PKI 证书管理可能很复杂，但是，通过自动化的证书预置、部署和续订机制，可以减少与证书管理相关的麻烦。

AWS 提供了两种服务用于管理通用 PKI 证书：AWS Certificate Manager 和 AWS Private Certificate Authority（AWS Private CA）。ACM 是客户用于预置、管理和部署证书的主要服务，适用于面向公众的工作负载以及私有 AWS 工作负载。ACM 使用 AWS Private CA 来颁发私有证书，并 integrates 许多其它 AWS 托管式服务，以便为工作负载提供安全的 TLS 证书。ACM 还可以从 Amazon Trust Services 颁发公开可信的证书。ACM 的公有证书可用于面向公众的工作负载，因为默认情况下，现代浏览器和操作系统信任这些证书。

利用 AWS Private CA，您可以建立自己的根证书颁发机构或从属证书颁发机构，并通过 API 颁发 TLS 证书。在 TLS 连接的客户端一侧控制和管理信任链的场景中，您可以使用这些类型的证书。除了 TLS 使用场景外，还可以使用 AWS Private CA 通过自定义模板向 Kubernetes 容器组（pod）、Matter 设备产品认证、代码签名和其他使用场景颁发证书。您还可以使用 IAM Roles Anywhere，向已经为其颁发了 X.509 证书（使用您的私有 CA 签名）的本地工作负载提供临时 IAM 凭证。

除了 ACM 和 AWS Private CA 之外，AWS IoT Core 针对为物联网设备预置、管理和部署 PKI 证书提供专业化支持。AWS IoT Core 提供专门的机制，用于大规模将物联网设备载入到您的公钥基础设施中。

某些 AWS 服务，例如 Amazon API Gateway 和弹性负载均衡，提供自己的使用证书保护应用程序连接的能力。例如，API Gateway 和应用程序负载均衡器（ALB）都支持使用客户端证书的双向 TLS（mTLS），而这些证书是使用 AWS Management Console、CLI 或 API 创建和导出的。

建立私有 CA 层次结构的注意事项

当您需要建立私有 CA 时，请务必重视预先正确设计 CA 层次结构。在创建私有 CA 层次结构时，最佳实践是将 CA 层次结构的每个级别部署到单独的 AWS 账户 中。这个有意而为的步骤可减少 CA 层次结构中每个级别的暴露范围，使得发现 CloudTrail 日志数据中的异常变得更加简单，并可在某个账户遭到未经授权的访问时，缩小访问或影响的范围。根 CA 应位于自己的独立账户中，并且只能用于发布一个或多个中间 CA 证书。

然后，在不同于根 CA 账户的账户中创建一个或多个中间 CA，为最终用户、设备或其他工作负载发布证书。最后，从您的根 CA 向中间 CA 颁发证书，后者随之向您的最终用户或设备颁发证书。有关规划 CA 部署和设计 CA 层次结构（包括弹性规划、跨区域复制、在组织中共享 CA 等）的更多信息，请参阅《Planning your AWS Private CA deployment》。

实施步骤

1. 确定您的使用场景所需的相关 AWS 服务：
   - 许多使用场景都可以利用现有的 AWS 公钥基础设施并使用 AWS Certificate Manager。ACM 可用于为 Web 服务器、负载均衡器或公共可信证书的其他用途部署 TLS 证书。
   - 在您需要建立自己的私有证书颁发机构层次结构或需要使用可导出证书时，请考虑 AWS Private CA。然后，可以使用 ACM 颁发多种类型的终端实体证书（使用 AWS Private CA）。
   - 对于必须为嵌入式物联网（IoT）设备大规模预置证书的使用场景，请考虑使用 AWS IoT Core。
   - 考虑在 Amazon API Gateway 或应用程序负载均衡器等服务中使用原生 mTLS 功能。
2. 尽可能实施自动证书续订：
   - 将 ACM 托管续订用于 ACM 颁发的证书以及集成的 AWS 托管服务。
3. 建立日志记录和审计跟踪：
   - 启用 CloudTrail 日志，以便跟踪对具有证书颁发机构的账户的访问。请考虑在 CloudTrail 中配置日志文件完整性验证，用于验证日志数据的真实性。
   - 定期生成和审查审计报告，列出您的私有 CA 已颁发或撤销的证书。这些报告可以导出到 S3 存储桶。
   - 部署私有 CA 时，您还需要创建一个 S3 存储桶，用于存储证书撤销列表（CRL，Certificate Revocation List）。有关根据工作负载要求配置此 S3 存储桶的指南，请参阅《Planning a certificate revocation list (CRL)》。

资源

相关最佳实践：

- SEC02-BP02 使用临时凭证

- SEC08-BP01 实施安全密钥管理
- SEC09-BP03 对网络通信进行身份验证

相关文档：

- How to host and manage an entire private certificate infrastructure in AWS
- How to secure an enterprise scale ACM Private CA hierarchy for automotive and manufacturing
- Private CA best practices
- How to use AWS RAM to share your ACM Private CA cross-account

相关视频：

- Activating AWS Certificate Manager Private CA（讲习会）

相关示例：

- Private CA workshop
- 物联网设备管理讲习会（包括设备预置）

相关工具：

- 使用 AWS Private CA 的 Kubernetes 证书管理器插件

SEC09-BP02 在传输中执行加密

实施您根据贵组织的政策、监管义务和标准定义的加密要求，以帮助满足组织、法律和合规性要求。如要在虚拟私有云（VPC）外部传输敏感数据，务必仅使用具有加密功能的协议。即使在不可信的网络中传输数据，加密也有助于保持数据的机密性。

期望结果：对资源与互联网之间的网络流量进行加密，以减少对数据的未经授权访问。根据您的安全需求，加密内部 AWS 环境中的网络流量。可以使用安全的 TLS 协议和密码套件对传输中数据进行加密。

常见反模式：

- 使用已弃用的 SSL、TLS 和密码套件组件版本（例如，SSL v3.0、1024 位 RSA 密钥和 RC4 密码）。

- 允许未加密的（HTTP）流量进出面向公众的资源。
- 未在 X.509 证书到期前监控和替换证书。
- 对 TLS 使用自签名 X.509 证书。

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

AWS 服务提供使用 TLS 的 HTTPS 端点进行通信，从而可以在与 AWS API 通信时提供传输中加密。通过使用安全组，可以在虚拟私有云（VPC）中审计和阻止不安全的 HTTP 协议。也可以在 Amazon CloudFront 中或应用程序负载均衡器上，将 HTTP 请求自动重定向到 HTTPS。可以使用 Amazon Simple Storage Service (Amazon S3) bucket policy 来限制通过 HTTP 上传对象的功能，从而有效地强制使用 HTTPS 将对象上传到存储桶。您可以完全控制计算资源，以便在整个服务中实施加密。您也可以利用 VPN 连接从外部网络或 AWS Direct Connect 连接到您的 VPC 中，以便于对流量进行加密。验证客户端是否使用至少 TLS 1.2 来调用 AWS API，因为 AWS has deprecated the use of earlier versions of TLS as of February 2024。我们建议您使用 TLS 1.3。如果您对传输中加密有特殊要求，可以在 AWS Marketplace 中找到可用的第三方解决方案。

实施步骤

- 实施传输中加密：您定义的加密要求应基于最新的标准和最佳实践，且仅允许使用安全协议。例如，配置一个安全组，仅允许通过 HTTPS 协议访问应用程序负载均衡器或 Amazon EC2 实例。
- 在边缘服务中配置安全协议：使用 Amazon CloudFront 配置 HTTPS，并使用适合您的安全状况和使用案例的安全配置文件。
- 将 VPN 用于外部连接：考虑使用 IPsec VPN 来保护点对点或网络对网络连接，以帮助实现数据隐私和完整性。
- 在负载均衡器中配置安全协议：选择一个安全策略，该策略提供受客户端支持且将要连接到侦听器的强大密码套件。为应用程序负载均衡器创建 HTTPS 侦听器。
- 在 Amazon Redshift 中配置安全协议：将集群配置为要求安全套接字层（SSL）或传输层安全性协议（TLS）连接。
- 配置安全协议：查看 AWS 服务文档，以确定传输中加密功能。
- 上传到 Amazon S3 存储桶时配置安全访问：使用 Amazon S3 存储桶策略控制措施执行对数据的安全访问。
- 不妨考虑使用 AWS Certificate Manager：ACM 允许您预置、管理和部署用于 AWS 服务的公有 TLS 证书。

- 不妨考虑使用 AWS Private Certificate Authority 满足私有 PKI 需求：AWS Private CA 允许您创建私有证书颁发机构（CA）层次结构，以签发可用于创建加密 TLS 通道的终端实体 X.509 证书。

资源

相关文档：

- 将 HTTPS 与 CloudFront 搭配使用
- 使用 AWS Virtual Private Network 将 VPC 连接到远程网络
- 为应用程序负载均衡器创建 HTTPS 侦听器
- 教程：在 Amazon Linux 2 上配置 SSL/TLS
- 使用 SSL/TLS 加密与数据库实例的连接
- 配置连接的安全选项

SEC09-BP03 对网络通信进行身份验证

使用传输层安全性协议（TLS）或 IPsec 等支持身份验证的协议来验证通信的身份。

将您的工作负载设计为在服务和应用程序之间通信或与用户通信时，使用经过身份验证的安全网络协议。使用支持身份验证和授权的网络协议，可以加强对网络流量的控制，并减少未授权访问的影响。

期望结果：工作负载具有明确定义的数据面板和控制面板，它们控制流量在服务之间的流动。在技术上可行的情况下，流量将使用经过身份验证和加密的网络协议。

常见反模式：

- 工作负载中存在未加密或未经身份验证的流量。
- 在多个用户或实体之间重用身份验证凭证。
- 仅依赖网络控制作为访问控制机制。
- 创建自定义身份验证机制，而不是依赖行业通用身份验证机制。
- VPC 中的服务组件或其它资源之间有过于宽松的流量流动。

建立此最佳实践的好处：

- 限制未经授权访问工作负载某一部分的影响范围。
- 提供更高级别的保障，即操作只能由经过身份验证的实体执行。

- 通过明确定义和强制执行预期的数据传输接口，改善服务的解耦。
- 通过请求归因和明确定义的通信界面，增强监控、日志记录和事件响应。
- 通过将网络控制与身份验证和授权控制相结合，为您的工作负载提供深度防御。

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

您的工作负载的网络流量模式可分为两类：

- 东西向流量代表构成工作负载的服务之间的流量。
- 南北向流量代表您的工作负载和使用器之间的流量。

对南北向流量进行加密是常见做法，而使用经过身份验证的协议保护东西向流量则不太常见。现代安全实践建议，仅靠网络设计并不足以在两个实体之间建立可信关系。当两项服务可能位于公共网络边界内时，极佳做法仍然是对这些服务之间的通信进行加密、身份验证和授权。

例如，无论请求来自哪个网络，AWS 服务 API 都使用 AWS 签名版本 4（SigV4）签名协议对调用方进行身份验证。这种身份验证可确保 AWS API 可以验证请求操作的身份，然后可以将该身份与策略结合起来，作出授权决策，以确定是否应该允许该操作。

Amazon VPC Lattice 和 Amazon API Gateway 等服务允许您使用相同的 SigV4 签名协议，为自己的工作负载中的东西向流量添加身份验证和授权。如果您的 AWS 环境之外的资源要与服务进行通信，而服务需要基于 SigV4 的身份验证和授权，则您可以对非 AWS 资源使用 AWS Identity and Access Management（IAM）Roles Anywhere 来获取临时 AWS 凭证。这些凭证可用于对使用 SigV4 的服务请求进行签名，以授权访问权限。

验证东西向流量的另一种常见机制是 TLS 双向身份验证（mTLS）。许多物联网（IoT）、企业对企业应用程序和微服务都使用 mTLS，通过使用客户端和服务器端 X.509 证书来验证 TLS 通信两端的身份。这些证书可以由 AWS Private Certificate Authority（AWS Private CA）颁发。可以使用 Amazon API Gateway 等服务，针对工作负载间或工作负载内的通信提供 mTLS 身份验证。应用程序负载均衡器还支持将 mTLS 用于内部或外部工作负载。虽然 mTLS 为 TLS 通信的两端提供身份验证信息，但它不提供授权机制。

最后，OAuth 2.0 和 OpenID Connect（OIDC）这两种协议通常用于控制用户对服务的访问，但如今在服务间流量中也变得越来越流行。API Gateway 提供了 JSON Web 令牌（JWT）授权方，允许工作负载使用 OIDC 或 OAuth 2.0 身份提供商颁发的 JWT 来限制对 API 路由的访问。可依据 OAuth2 范围来作出基本授权决策，但授权检查仍需要在应用层实现，仅靠 OAuth2 范围无法支持更复杂的授权需求。

实施步骤

- 定义并记录您的工作负载网络流：实施深度防御策略的第一步是定义工作负载的流量。
  - 创建数据流示意图，明确定义构成工作负载的不同服务之间如何传输数据。此示意图是强制这些流量通过经身份验证的网络渠道传输的第一步。
  - 在开发和测试阶段对您的工作负载进行检测，以验证数据流示意图是否准确反映了工作负载在运行时的行为。
  - 在执行威胁建模练习时，数据流示意图可能也很有用，如《SEC01-BP07 使用威胁模型识别威胁并确定缓解措施的优先级》中所述。
- 建立网络控制：考虑使用 AWS 功能建立起与数据流相符的网络控制。虽然网络边界不应该是唯一的安全控制措施，但它们在深度防御策略中提供了一个安全层来保护您的工作负载。
  - 使用安全组来建立、定义和限制资源之间的数据流。
  - 考虑使用 AWS PrivateLink 与 AWS 以及支持 AWS PrivateLink 的第三方服务通信。通过 AWS PrivateLink 接口端点发送的数据保留在 AWS 网络主干内，而不通过公共互联网传输。
- 在工作负载中实施跨服务的身份验证和授权：选择极其适合在工作负载中提供经过身份验证的加密流量的一组 AWS 服务。
  - 考虑使用 Amazon VPC Lattice 来保护服务间的通信。VPC Lattice 可以结合使用 Sigv4 身份验证与身份验证策略来控制服务间的访问。
  - 对于使用 mTLS 进行的服务间通信，请考虑使用 API Gateway 和应用程序负载均衡器。AWS Private CA 可用于建立私有 CA 层次结构，该层次结构能够颁发与 mTLS 结合使用的证书。
  - 与使用 OAuth 2.0 或 OIDC 的服务集成时，可以考虑使用 JWT 授权方的 API Gateway。
  - 对于工作负载和物联网设备之间的通信，可以考虑使用 AWS IoT Core，它提供多种网络流量加密和身份验证选项。
- 监控未经授权的访问：持续监控非预期的通信渠道、试图访问受保护资源的未授权主体以及其它不当访问模式。
  - 如果使用 VPC Lattice 来管理对服务的访问，请考虑启用和监控 VPC Lattice 访问日志。这些访问日志包括有关请求实体的信息、源和目的地 VPC 等网络信息以及请求元数据。
  - 考虑启用 VPC 流日志，以捕获网络流量的元数据并定期检查是否存在异常。
  - 有关规划、模拟和响应安全事件的更多指导，请参阅《AWS Security Incident Response Guide》和 AWS Well-Architected Framework 安全性支柱的“事件响应”部分。

资源

相关最佳实践：

- SEC03-BP07 分析公共和跨账户访问
- SEC02-BP02 使用临时凭证
- SEC01-BP07 使用威胁模型识别威胁并确定缓解措施的优先级

相关文档：

- Evaluating access control methods to secure Amazon API Gateway APIs
- Configuring mutual TLS authentication for a REST API
- How to secure API Gateway HTTP endpoints with JWT authorizer
- Authorizing direct calls to AWS services using AWS IoT Core credential provider
- 《AWS Security Incident Response Guide》

相关视频：

- AWS re:invent 2022: Introducing VPC Lattice
- AWS re:invent 2020: Serverless API authentication for HTTP APIs on AWS

相关示例：

- Amazon VPC Lattice Workshop
- Zero-Trust Episode 1 – The Phantom Service Perimeter workshop

事件响应

问题

- SEC 10. 如何预测、响应意外事件以及从意外事件中恢复？

SEC 10. 如何预测、响应意外事件以及从意外事件中恢复？

即使采用成熟的预防和检测性控制措施，您的组织也应实施机制来响应安全事件并缓解安全事件可能带来的影响。您的准备工作会极大地影响团队在意外事件发生期间采取有效行动、对问题进行隔离、遏制和取证并将运行状态恢复到已知良好状态的能力。在安全事件发生之前确保相关工具和访问权限部署到位，然后通过 GameDay 活动定期进行事件响应演练，这样有助于确保您有能力恢复并最大限度避免业务中断。

最佳实践

- SEC10-BP01 确定关键人员和外部资源
- SEC10-BP02 制定事件管理计划
- SEC10-BP03 准备取证能力
- SEC10-BP04 制定和测试安全事件响应行动手册
- SEC10-BP05 预置访问权限
- SEC10-BP06 预部署工具
- SEC10-BP07 运行模拟
- SEC10-BP08 建立从事件中吸取经验教训的框架

SEC10-BP01 确定关键人员和外部资源

确定内部和外部人员、资源和法律义务，来协助组织应对事件。

期望结果：您有一份关键人员名单、他们的联系信息以及他们在应对安全事件时扮演的角色。您可以定期审查这些信息并进行更新，以便分别从内部工具和外部工具的角度反映人事变动。在记录这些信息时，您需要考虑所有第三方服务提供商和供应商，包括安全合作伙伴、云提供商和软件即服务（SaaS，Software-as-a-Service）应用程序。在安全事件发生期间，有适当级别的责任、背景和访问权限的人员能够做出响应和恢复动作。

常见反模式：

- 在应对安全事件时，没有维护一份包含关键人员联系信息、角色和职责的最新关键人员名单。
- 在应对事件和从事件中恢复时，假设每个人都了解人员、依赖项、基础设施和解决方案。
- 没有代表关键基础设施或应用程序设计的文档或知识库。
- 没有为新员工制定适当的入职流程，无法有效参与安全事件响应，例如进行事件模拟。
- 没有制定当关键人员暂时无法到岗或者在安全事件中无法做出反应时，所需要的上报途径。

建立此最佳实践的好处：这种实践减少了事件发生期间用于确定合适人员及其角色的分流和响应时间。通过维护一份关键人员及其角色的最新名单，极大限度地减少事件发生期间的时间浪费，这样您就能够让合适的人员进行分流并从事件中恢复过来。

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

确定组织中的关键人员：维护一份贵组织内需要参与事件的人员的联系名单。在发生组织变革、晋升和团队变动等人事变动时，应定期审查和更新这些信息。这对于事件经理、事件响应者和沟通负责人等关键角色尤其重要。

- 事件经理：事件经理在事件响应期间拥有全面权力。
- 事件响应者：事件响应者负责调查和修复活动。这些人员可能因事件类型而异，但通常是负责受影响的应用程序的开发人员和运营团队。
- 沟通负责人：沟通负责人负责内外部沟通，特别是与公共机构、监管机构和客户的沟通。
- 入职流程：定期对新员工进行培训和入职，使他们具备必要的技能和知识，以便有效地为事件响应工作做出贡献。将模拟和动手练习作为入职流程的一部分，以协助他们做好准备。
- 主题专家（SME）：对于分布式自主团队，我们建议您为任务关键型工作负载确定一名 SME。主题专家有助于我们深入了解事件中涉及的关键工作负载的运行和数据分类。

示例表格式：

```
  | Role | Name | Contact Information | Responsibilities |
1 | ——- | ——- | ——- | ——- |
2 | Incident Manager | Jane Doe| jane.doe@example.com | Overall authority during
 response |
3 | Incident Responder | John Smith | john.smith@example.com | Investigation and
 remediation |
4 | Communications Lead | Emily Johnson | emily.johnson@example.com | Internal and
 external communications |
5 | Communications Lead | Michael Brown | michael.brown@example.com | Insights on
 critical workloads |
```

考虑使用 AWS Systems Manager Incident Manager 功能，来捕获关键联系人、制定响应计划、自动执行随时待命方案并制定上报计划。通过随时待命方案自动安排和轮换所有员工，使工作负载的责任由其所有者分担。这促进了良好的实践，例如发布相关指标和日志，以及定义与工作负载相关的警报阈值。

确定外部合作伙伴：企业运用独立软件供应商（ISV）、合作伙伴和分包商开发的工具，为客户构建差异化解决方案。让各方的这些关键人员参与进来，他们有助于应对事件并从事件中恢复。我们建议您注册相应级别的 支持，以便通过支持案例及时联系 AWS 主题专家。考虑与所有关键解决方案提供商就工作负载达成类似安排。有些安全事件要求上市企业向相关公共机构和监管机构通报事件及影响。请维护和更新相关部门和负责人的联系信息。

实施步骤

1. 设置事件管理解决方案。
   a. 考虑在您的安全工具账户中部署 Incident Manager。
2. 在事件管理解决方案中定义联系人。
   a. 为每位联系人至少定义两种联系渠道（例如短信、电话或电子邮件），以便确保事件发生期间能够联系上。
3. 制定响应计划。
   a. 确定事件发生时应接洽的最合适的联系人。根据参与人员的角色，而不是单个联系人，制定上报计划。考虑纳入可能负责通知外部实体的联系人，即使他们没有直接参与解决事件也是如此。

资源

相关最佳实践：

- OPS02-BP03 确定对运营活动绩效负责的责任人

相关文档：

- 《AWS Security Incident Response Guide》

相关示例：

- AWS 客户行动手册框架
- 准备和响应 AWS 环境中的安全事件

相关工具：

- AWS Systems Manager Incident Manager

相关视频：

- Amazon's approach to security during development

SEC10-BP02 制定事件管理计划

为事件响应制定的第一个文档是事件响应计划。事件响应计划旨在为您的事件响应计划和战略奠定基础。

建立此最佳实践的好处：要想成功实现可扩展的事件响应计划，制定全面且明确定义的事件响应流程是关键。在发生安全事件时，明确的步骤和工作流有助于您及时做出响应。您可能已经有事故响应流程。无论您当前的状态如何，定期更新、迭代和测试事件响应流程都很重要。

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

对于响应、缓解安全事件的潜在影响并从中恢复来说，事件管理计划是至关重要的。事件管理计划是一个结构化的过程，用于及时地确定、补救和响应安全事件。

云的许多操作角色和要求都与本地环境中的相同。在创建事件管理计划时，应考虑最符合业务成果和合规性要求的响应和恢复策略，这一点非常重要。例如，如果您在 AWS 中运行符合美国 FedRAMP 标准的工作负载，请遵守 NIST SP 800-61 Computer Security Handling Guide 中的建议。同样，在运行存储个人身份信息（PII）的工作负载时，请考虑如何防止和应对与数据驻留和使用相关的问题。

在为 AWS 中的工作负载制定事件管理计划时，请首先使用 AWS 责任共担模式，以便构建针对事件响应的深度防御方法。在此模式中，AWS 负责管理云本身的安全，云内部的安全则由您负责。这意味着您将保留控制权，并对选择实施的安全控制机制负责。《AWS Security Incident Response Guide》详细介绍了构建以云为中心的事件管理计划的关键概念和基本指南。

必须不断地迭代有效的事件管理计划，使其与您的云运营目标保持一致。在创建和改进事件管理计划时，请考虑使用下面详述的实施计划。

实施步骤

1. 定义组织内部用于处理安全事件的角色和职责。这应涉及不同部门的代表，包括：
    - 人力资源（HR）
    - 执行团队
    - 法务部门
    - 应用程序所有者和开发人员（主题专家或 SME）
2. 明确概述在事件发生期间谁负责、对谁问责、咨询谁以及通知谁（RACI）。创建 RACI 图表来促进快速和直接的沟通，并清楚地概述事件不同阶段的领导关系。

3. 在事件发生期间，让应用程序所有者和开发人员（SME）参与进来，因为他们可以提供有价值的信息和背景信息来帮助衡量影响。与这些 SME 建立关系，并在实际事件发生之前与他们练习事件响应场景。
4. 让值得信赖的合作伙伴或外部专家参与调查或响应过程，因为他们可以提供额外的专业知识和视角。
5. 使您的事件管理计划和角色与管理您组织的任何当地法规或合规要求保持一致。
6. 定期练习和测试您的事件响应计划，并涉及所有已定义的角色和职责。这有助于简化流程，并验证您对安全事件的响应井井有条且高效。
7. 定期审核和更新角色、职责和 RACI 图表，或者在组织结构或要求发生变化时进行审核和更新。

了解 AWS 响应团队和支持

- AWS 支持
  - 支持 包含一系列计划，这些计划旨在让您能够运用各种工具和专业知识来为成功部署和正常实施 AWS 解决方案提供支持。如果您需要技术支持及更多资源来规划、部署和优化 AWS 环境，则可以选择最符合 AWS 使用案例的支持计划。
  - 考虑将支持中心（在 AWS Management Console 中，需要登录）作为中心联系点，为影响您 AWS 资源的问题获取支持。对 支持 的访问由 AWS Identity and Access Management 控制。有关获取对 支持 功能的访问权限的更多信息，请参阅《Getting started with 支持》。
- AWS 客户事件响应团队（CIRT）
  - AWS 客户事件响应团队（CIRT）是一支专业的 AWS 全球团队，全天候向客户提供支持，协助客户解决根据 AWS 责任共担模式应由客户一方负责的安全事件。
  - 当 AWS CIRT 为您提供支持时，他们会为 AWS 上出现的安全事件提供分类和恢复方面的协助。他们可以使用 AWS 服务日志来协助分析根本原因，并为您提供恢复建议。他们还可以提供安全建议和最佳实践，从而让您以后能够避免出现安全事件。
  - AWS 客户要与 AWS CIRT 交流，可以开立 支持 案例。
- DDoS 响应支持
  - AWS 提供 AWS Shield，它提供了托管的分布式拒绝服务（DDoS）攻击保护服务，可保护在 AWS 上运行的 Web 应用程序。Shield 提供不间断检测和自动化内嵌缓解措施，可以最大限度地减少应用程序停机时间和延迟，因此无需与 支持 交流即可从 DDoS 保护中受益。Shield 分为两个级别：AWS Shield Standard 和 AWS Shield Advanced。要了解这两个级别之间的区别，请参阅《Shield 功能文档》。
- AWS Managed Services（AMS）

- AWS Managed Services（AMS）可持续管理您的 AWS 基础设施，让您可以专注于应用程序。AMS 实施最佳实践来维护您的基础设施，让您能够降低运营开销和风险。AMS 可以自动执行常见活动 (例如更改请求、监控、补丁管理、安全性和备份服务)，并可以提供全生命周期服务来预置、运行和支持您的基础设施。
- AMS 负责部署一套安全检测控制措施，并全天候提供对警报的第一线响应。启动警报后，AMS 遵循一组标准的自动和手动行动手册，验证是否有一致的响应。这些行动手册在功能部署期间与 AMS 客户共享，这样客户就能够开发并与 AMS 协调响应措施。

制定事件响应计划

事件响应计划旨在为您的事件响应计划和战略奠定基础。事件响应计划应包含在正式文档中。事件响应计划通常包括以下部分：

- 事件响应团队概述：概述事件响应团队的目标和职能。
- 角色和职责：列出事件响应利益相关者，并详细说明他们在发生事件时的角色。
- 沟通计划：详细介绍联系信息，以及在事件发生期间如何进行沟通。
- 后备沟通方法：此时的最佳实践是采用带外通信，作为事件沟通的后备。AWS Wickr 就是一个提供安全的带外通信渠道的应用程序示例。
- 事件响应阶段和应采取的行动：列举事件响应的各个阶段（例如，检测、分析、消除、遏制和恢复），包括在这些阶段中要采取的高级别操作。
- 事件严重性和优先级定义：详细说明如何对事件的严重性进行分类，如何确定事件的优先级，然后详细说明严重性定义对上报程序有何影响。

尽管这些内容部分在各种规模和行业的公司中很常见，但每个组织的事件响应计划都是独一无二的。您需要制定最适合贵组织的事件响应计划。

资源

相关最佳实践：

- SEC04 检测

相关文档：

- 《AWS Security Incident Response Guide》
- NIST：计算机安全事件处理指南

SEC10-BP03 准备取证能力

在发生安全事件之前，可以考虑构建取证能力来支持安全事件调查工作。

在未建立这种最佳实践的情况下暴露的风险等级：中

传统本地取证的概念适用于 AWS。有关开始在 AWS Cloud 中构建取证功能的关键信息，请参阅《Forensic investigation environment strategies in the AWS Cloud》。

设置好取证的环境和 AWS 账户 结构后，确定在以下四个阶段有效执行可靠取证方法所需的技术：

- 收集：收集相关的 AWS 日志，例如 AWS CloudTrail、AWS Config、VPC 流日志和主机级日志。收集受影响的 AWS 资源的快照、备份和内存转储（如果有）。
- 检查：通过提取和评测相关信息来检查收集到的数据。
- 分析：分析收集到的数据，以便了解事件并从中得出结论。
- 报告：提供分析阶段得出的信息。

实施步骤

准备取证环境

AWS Organizations 有助于您随着 AWS 资源的增长和扩展，集中管理和监管 AWS 环境。AWS 组织会整合您的 AWS 账户，这样您就能够将这些账户作为一个单元进行管理。您可以使用组织单元 (OU)，将账户分组到一起，作为一个单元管理。

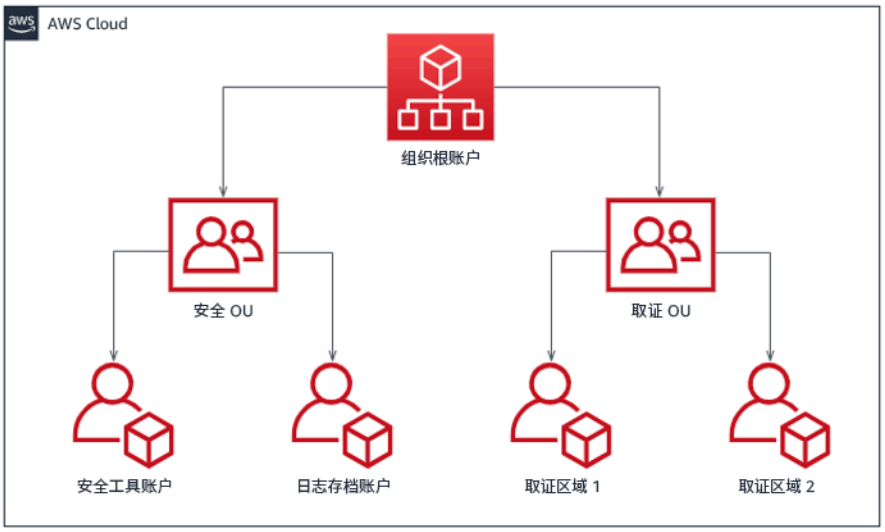
对于事件响应，拥有一个支持事件响应功能的 AWS 账户 结构（包括安全 OU 和取证 OU）会很有帮助。在安全 OU 中，您应该拥有以下账户：

- 日志存档：将日志聚合到权限有限的日志存档 AWS 账户 中。
- 安全工具：将安全服务集中在安全工具 AWS 账户 中。此账户以安全服务的委托管理员身份运行。

在取证 OU 中，您可以选择实施单一取证账户，也可以为您运营的每个区域实施账户，具体取决于哪种账户最适合您的业务和运营模式。如果为每个区域创建一个取证账户，就可以阻止在该区域之外创建 AWS 资源，降低资源被复制到非预期区域的风险。例如，如果您只在美国东部（弗吉尼亚州北部）区域（us-east-1）和 美国西部（俄勒冈州）（us-west-2）运营，那么您将在取证 OU 中拥有两个账户：一个用于 us-east-1，另一个用于 us-west-2。

可以为多个区域创建取证 AWS 账户。在将 AWS 资源复制到该账户时应小心谨慎，确保符合数据主权要求。由于预置新账户需要时间，因此必须在事件发生前创建和分析取证账户，以便响应人员能够做好准备，有效地使用这些账户进行响应。

下图显示了一个账户结构示例，其中包括一个取证 OU，涵盖了根据每个区域创建的取证账户：



用于响应事件而根据区域创建的账户结构

捕获备份和快照

为关键系统和数据库建立备份对于从安全事件中恢复和取证至关重要。有了备份，您就能够将系统恢复到以前的安全状态。在 AWS 上，您可以创建各种资源的快照。快照为您提供这些资源的时间点备份。有许多 AWS 服务能够在备份和恢复方面为您提供支持。有关这些服务以及备份和恢复方法的详细信息，请参阅《Backup and Recovery Prescriptive Guidance》以及《Use backups to recover from security incidents》。

特别是遇到勒索软件等情况时，妥善保护备份至关重要。有关保护备份的指导，请参阅《Top 10 security best practices for securing backups in AWS》。除了确保备份安全外，您还应当定期测试备份和还原流程，从而确保现有的技术和流程按预期运行。

自动取证

在安全事件期间，您的事件响应团队必须能够快速收集和分析证据，同时保持事件相关时间段的准确性（例如捕获与特定事件或资源相关的日志，或收集 Amazon EC2 实例的内存转储）。对于事件响应团队来说，手动收集相关证据既具有挑战性又很耗时，尤其是在存在大量实例和账户的情况下。此外，手动收集容易出现人为错误。出于这些原因，您应该尽可能开发和实现取证自动化功能。

AWS 提供了大量用于取证的自动化资源，这些资源在下面的“资源”部分中列出。这些资源是我们开发并由客户实施的取证模式示例。虽然这些资源可能是有用的参考架构，但可以考虑根据您的环境、要求、工具和取证流程对资源进行修改，或者创建新的取证自动化模式。

资源

相关文档：

- 《AWS Security Incident Response Guide》– Develop Forensics Capabilities
- 《AWS Security Incident Response Guide》– Forensics Resources
- AWS Cloud 中的取证调查环境策略
- 如何在 AWS 中自动实施取证磁盘收集
- AWS Prescriptive Guidance – 自动化事件响应和取证

相关视频：

- 自动化事件响应和取证

相关示例：

- 自动事件响应和取证框架
- Amazon EC2 的自动取证编排工具

SEC10-BP04 制定和测试安全事件响应行动手册

准备事件响应流程的关键环节是制定行动手册。事件响应行动手册提供了规范性指南和步骤，供发生安全事件时遵循。清晰的结构和步骤可简化响应，减少发生人为错误的可能性。

在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

应针对以下事件场景创建行动手册：

- 预期事件：应针对预期的事件创建行动手册。这包括拒绝服务（DoS）、勒索软件和凭证泄露等威胁。
- 已知的安全调查发现或警报：应该创建行动手册来应对已知的安全调查发现和警报，例如来自 Amazon GuardDuty 的调查发现和警报。当您收到 GuardDuty 调查发现时，行动手册应提供明确

的步骤，以防止错误处理或忽略警报。有关修复措施的更多详细信息和指南，请参阅 Remediating security issues discovered by GuardDuty。

行动手册应包含安全分析师需要完成的技术步骤，以便充分调查和应对潜在的安全事件。

实施步骤

行动手册中应包括的项目有：

- 行动手册概述：本行动手册针对哪些风险或事件场景？ 本行动手册的目标是什么？
- 先决条件：此事件场景需要哪些日志、检测机制和自动化工具？ 预期的通知是什么？
- 沟通和上报信息：谁参与其中，他们的联系信息是什么？ 每个利益相关方的责任是什么？
- 响应步骤：在事件响应的各个阶段，应采取哪些战术性措施？ 分析师应该进行哪些查询？ 应该运行什么代码才能达到预期的结果？
  - 检测：如何检测事件？
  - 分析：如何确定影响范围？
  - 控制：如何隔离事件来限制其影响范围？
  - 消除：如何从环境中消除威胁？
  - 恢复：受影响的系统或资源将如何恢复生产？
- 期望结果：运行查询和代码后，行动手册的期望结果是什么？

资源

相关的 Well-Architected 最佳实践：

- SEC10-BP02 – 制定事件管理计划

相关文档：

- 事件响应行动手册框架
- 制定自己的事件响应行动手册
- 事件响应行动手册样本
- 使用 Jupyter 行动手册和 CloudTrail Lake 构建 AWS 事件响应运行手册

SEC10-BP05 预置访问权限

确保事件响应者将正确的访问权限预置到 AWS 中，以缩短调查到恢复所需的时间。

常见反模式：

- 使用根账户进行事件响应。
- 变更现有账户。
- 在提供实时权限提升时直接操作 IAM 权限。

在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 建议尽可能减少或消除对长期有效凭证的依赖，转而使用临时凭证和实时权限提升机制。长期有效的凭证容易带来安全风险，并且会增加运营开销。对于大多数管理任务以及事件响应任务，建议您对管理访问实施身份联合验证以及临时上报。在此模型中，用户请求提升到更高级别的权限（例如事件响应角色），如果用户符合提升条件，则会向审批者发送请求。如果请求获得批准，用户将收到一组临时的 AWS 凭证，可用于完成用户任务。在这些凭证过期后，用户必须提交新的提升请求。

在大多数事件响应场景中，建议使用临时权限提升。执行此操作的正确方法是使用 AWS Security Token Service 和会话策略来限定访问范围。

在一些场景中，联合身份不可用，例如：

- 与被盗用的身份提供者（IdP）相关的中断。
- 导致联合访问管理系统损坏的错误配置或人为错误。
- 恶意活动，例如分布式拒绝服务（DDoS，Distributed Denial of Service）事件或导致系统不可用的活动。

在上述情况下，应配置紧急 Break Glass 访问，以允许调查事件并及时给予补救。我们建议您使用具有适当权限的用户、组或角色，来执行任务和访问 AWS 资源。请仅将根用户用于需要根用户凭证的任务。要确认事件响应者对 AWS 和其他相关系统是否具有正确的访问权限级别，建议预置专用的账户。账户需要特许的访问权限，并且必须受到严格的控制和监视。在构建账户时，必须使用执行必要任务所需的最少权限，并且访问级别应基于作为事件管理计划的一部分创建的行动手册。

最好使用专门构建的专用用户和角色。通过添加 IAM 策略来临时提升用户或角色的访问权限，既会导致无法清楚地了解用户在事件期间拥有哪些访问权限，又会带来无法撤销提升的权限的风险。

请务必删除尽可能多的依赖项，以确保能在尽可能多的故障场景中获得访问权限。为了支持此操作，可创建一个行动手册，验证是否在专用的安全账户中创建事件响应用户作为用户，而不是通过任何现有的联合身份验证或单点登录（SSO）解决方案管理他们。每个响应者都必须拥有自己的指定账户。账户配置必须实施强密码策略和多重身份验证（MFA）。如果事件响应行动手册仅需要对 AWS Management Console 的访问权限，则用户不应配置访问密钥，并且应明确禁止用户创建访问密钥。可以使用 IAM 策略或服务控制策略（SCP，Service Control Policy）进行此配置，如 AWS 安全最佳实践（适用于 AWS Organizations SCP）中所述。用户仅能够在其他账户中代入事件响应角色，而不应具有其他任何权限。

在事件处理期间，可能需要向其他内部或外部人员授予访问权限，以支持调查、补救或恢复活动。在这种情况下，可以使用前面提到的行动手册机制，并且必须创建一个流程，确保在事件结束后立即撤消其他任何访问权限。

要确保能正确地监控和审计对事件响应角色的使用，至关重要的一点是，为此目的创建的 IAM 账户不会在人员之间共享，并且不会使用 AWS 账户根用户，除非特定任务要求这样做。如果需要根用户（例如，对特定账户的 IAM 访问权限不可用），请使用单独的流程和可用的行动手册来验证根用户登录凭证和 MFA 令牌的可用性。

要为事件响应角色配置 IAM 策略，请考虑使用 IAM Access Analyzer 来生成基于 AWS CloudTrail 日志的策略。为此，请在非生产账户中向事件响应角色授予管理员访问权限，并运行行动手册。完成后，会创建一个策略，仅允许已执行的操作。之后，可以跨所有账户将此策略应用于所有事件响应角色。您可能希望为每个行动手册创建一个单独的 IAM 策略，以便更轻松地进行管理和审计。示例行动手册可能包括针对勒索软件、数据泄露、丢失生产访问权限和其他场景的响应计划。

使用事件响应用户账户可在其他 AWS 账户 中代入专用的事件响应 IAM 角色。必须将这些角色配置为仅可由安全账户中的用户代入，并且信任关系必须要求调用主体已使用 MFA 进行身份验证。角色必须使用严格界定的 IAM 策略来控制访问。确保这些角色的所有 `AssumeRole` 请求都记录在 CloudTrail 中并发出提醒，并确保记录使用这些角色执行的任何操作。

强烈建议清楚地命名 IAM 账户和 IAM 角色，以便在 CloudTrail 日志中轻松找到他们。例如，将 IAM 账户命名为 *`<USER_ID>`*`-BREAK-GLASS`，并将 IAM 角色命名为 `BREAK-GLASS-ROLE`。

CloudTrail 用于记录 AWS 账户中的 API 活动，并且应该用于配置关于使用事件响应角色的提醒。请参阅博文，了解有关配置使用根密钥时的提醒。可以修改说明以配置 Amazon CloudWatch 指标筛选条件，从而筛选 `AssumeRole` 事件（与事件响应 IAM 角色相关）：

```
{ $.eventName = "AssumeRole" && $.requestParameters.roleArn =
 "<INCIDENT_RESPONSE_ROLE_ARN>" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !
= "AwsServiceEvent" }
```

由于事件响应角色可能具有高级别的访问权限，因此，请务必将这些提醒转至广泛的群体，并及时采取适当的行动。

在事件处理期间，响应者可能需要访问不受 IAM 直接保护的系统。它们可能包括 Amazon Elastic Compute Cloud 实例、Amazon Relational Database Service 数据库或软件即服务（SaaS）平台。强烈建议不要使用 SSH 或 RDP 等本机协议，而是使用 AWS Systems Manager Session Manager 对 Amazon EC2 实例进行所有管理访问。可以使用安全且经过审计的 IAM 控制此访问。此外，还可以使用 AWS Systems Manager Run Command 文档自动实施行动手册的部分内容，这样可以减少用户出错的机会并缩短恢复时间。对于访问数据库和第三方工具，我们建议将访问凭证存储在 AWS Secrets Manager 中，并向事件响应者角色授予访问权限。

最后，事件响应 IAM 账户的管理应该添加到您的合并人员、移动人员和离开人员流程中，并定期进行检查和测试，以确认只允许预期访问。

资源

相关文档：

- 管理对 AWS 环境的临时提升的访问权限
- 《AWS Security Incident Response Guide》
- AWS Elastic Disaster Recovery
- AWS Systems Manager Incident Manager
- 为 IAM 用户设置账户密码策略
- 在 AWS 中使用多重身份验证（MFA）
- 使用 MFA 配置跨账户访问
- 使用 IAM Access Analyzer 生成 IAM 策略
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- 如何在使用 AWS 账户的根访问密钥时接收通知
- 使用 IAM 托管策略创建精细会话权限
- Break glass access

相关视频：

- 在 AWS 中自动化事件响应和取证
- 运行手册、事件报告和事件响应 DIY 指南
- 准备和响应 AWS 环境中的安全事件

SEC10-BP06 预部署工具

确保安全人员预部署了适当的工具，来缩短从调查到恢复的时间。

在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

要自动执行安全响应和操作功能，您可以使用 AWS 提供的一整套 API 和工具。您可以完全自动执行身份管理、网络安全、数据保护和监控功能，并使用您已采用的常见软件开发方法交付这些功能。当构建安全自动化时，您的系统可以监控、审核和启动响应，您不必安排人员监控您的安全位置并对事件做出人为响应。

如果您的事件响应团队继续以同样的方式响应警报，警报可能会让他们应接不暇。久而久之，团队对警报的敏感性可能会下降，并可能在处理正常情况时犯错或者错过异常警报。利用一些功能自动处理重复和正常的警报，并将敏感、特殊的事件交由人员来处理，这样有助于避免疲于应对警报。集成异常检测系统（例如 Amazon GuardDuty、AWS CloudTrail Insights 和 Amazon CloudWatch Anomaly Detection）可以减轻常见阈值警报的负担。

您可以通过编程方式自动执行此流程中的步骤，从而改进手动流程。为事件定义修复模式之后，您可以将此模式分解为可执行的逻辑，并编写代码以执行此逻辑。然后，响应人员可以运行该代码来修复问题。久而久之，您就可以自动化越来越多的步骤，并最终自动处理各类常见事件。

在安全调查期间，您需要能够查看相关日志，以便记录并了解事件的来龙去脉和时间线。生成警报时也需要日志，因为日志可以指示某些相关操作已经发生。选择、启用、存储、设置查询和检索机制以及设置警报至关重要。此外，提供工具来搜索日志数据的有效方法是 Amazon Detective。

AWS 提供 200 多种云服务和数千种功能。我们建议您检查可支持和简化事件响应策略的服务。

除日志记录外，还应当制定并实施标记策略。标记有助于提供有关 AWS 资源用途的背景信息。标记也可用于实现自动化。

实施步骤

选择并设置用于分析和报警的日志

请参阅以下关于配置事件响应日志记录的文档：

- 安全事件响应的日志记录策略
- SEC04-BP01 配置服务和应用程序日志记录

启用安全服务来支持检测和响应

AWS 提供了本机检测、预防和响应功能，而其他服务可用于构建自定义安全解决方案。有关与安全事件响应最相关的服务列表，请参阅《云功能定义》。

制定和实施标记策略

要获取围绕 AWS 资源的业务场景和相关内部利益相关方的背景信息，可能很困难。要做到这一点，可以采用标签的形式，标签为 AWS 资源分配元数据，并由用户定义的键和值组成。您可以创建标签，按照用途、所有者、环境、处理的数据类型以及您选择的其他标准对资源进行分类。

采用一致的标记策略可以加快响应速度，并通过快速识别和辨别 AWS 资源的背景信息，最大限度地减少在组织背景方面所花费的时间。标签还可以充当启动自动响应的机制。有关要标记的内容的详细信息，请参阅《标记 AWS 资源》。首先，您需要定义要在组织内实施的标签。之后，实施并强制执行这一标记策略。有关实施和强制执行的详细信息，请参阅《使用 AWS 标签策略和服务控制策略（SCP）实施 AWS 资源标记策略》。

资源

相关的 Well-Architected 最佳实践：

- SEC04-BP01 配置服务和应用程序日志记录
- SEC04-BP02 在标准化位置收集日志、调查发现和指标

相关文档：

- 安全事件响应的日志记录策略
- 事件响应云功能定义

相关示例：

- 使用 Amazon GuardDuty 和 Amazon Detective 进行威胁检测和响应
- Security Hub 讲习会
- 使用 Amazon Inspector 进行漏洞管理

SEC10-BP07 运行模拟

随着组织不断发展壮大，威胁形势也会不断变化，因此务必要持续评估组织的事件响应能力。运行模拟（也称为实际演练）是可用于执行这种评估的一种方法。模拟过程使用现实世界中的安全事件场景，旨

在模仿威胁主体采取的战术、技术和程序（TTP），让组织通过响应现实中可能发生的模拟网络事件，来练习和评估自己的事件响应能力。

建立此最佳实践的好处：模拟有多种好处：

- 检验网络准备情况，有助于事件响应人员树立信心。
- 测试工具和工作流程的准确性和有效性。
- 完善沟通和上报环节，使之与您的事件响应计划相吻合。
- 提供机会来应对不太常见的攻击载体。

在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

模拟主要分为三种类型：

- 桌面演练：桌面演练模拟方法是一种基于讨论的研讨会，让各个事件响应利益相关方参与进来，练习角色和职责，以及练习使用既定的沟通工具和行动手册。通常是用一整天的时间在虚拟场地和/或实地中协调完成演练。由于桌面演练以讨论为基础，因此侧重于流程、人员和协作。在讨论中，技术是必不可少的一部分，但事件响应工具或脚本的实际使用通常不包括在桌面演练中。
- 紫队演练：紫队演练可提高事件响应人员（蓝队）和模拟威胁主体（红队）之间的协作能力。蓝队由安全运营中心（SOC）的成员组成，但也可以包括在实际网络事件中会参与进来的其他利益相关方。红队由渗透测试团队或接受过攻击安全培训的关键利益相关方组成。在设计场景时，红队会与演练协调员相互协作，以确保场景的准确性与可行性。在紫队演练中，主要的关注点是支持事件响应工作的检测机制、工具和标准操作程序（SOP）。
- 红队演练：在红队演练中，进攻方（红队）模拟进行攻击，以在预定范围内实现特定目标或一系列目标。防御方（蓝队）不一定知道演练的范围和持续时间，如此，可以更真实地评估他们应对真实事件的能力。由于红队的演练可能是侵入性测试，因此务必谨慎行事，并实施控制措施，以确保该演练不会对环境造成实际破坏。

请考虑定期协调开展网络模拟。对于参与者和整个组织而言，每种演练类型都可以带来独特的好处，因此您可以选择从不太复杂的模拟类型（例如桌面演练）入手，然后再慢慢过渡到较为复杂的模拟类型（红队演练）。您应根据自身的安全成熟度、资源和期望结果选择模拟类型。由于红队演练的复杂性和成本，一些客户可能不会选择进行红队演练。

实施步骤

无论您选择哪种模拟类型，模拟通常都遵循以下实施步骤：

1. 定义核心演练要素：定义模拟场景和模拟要达成的目标。这两者都应该得到领导层的认同。
2. 确定关键利益相关方：演练至少需要演练协调员和参与者。根据具体的场景，可能会涉及其他利益相关方，例如法务、通信或行政等领域的领导层。
3. 构建和测试场景：如果有特定要素不可行，则可能需要在构建时重新定义该场景。本阶段的期望结果是最终确定的场景。
4. 协调开展模拟：采用的模拟类型决定了所需的协调工作（书面讨论场景对比高技术含量的模拟场景）。协调员应根据演练目标调整其协调战术，并应尽可能让所有演练参与者都参与进来，以实现最大利益。
5. 撰写事后报告（AAR）：确定哪些方面进展较为顺利、哪些方面需要改进以及可能存在的差距。AAR 应衡量模拟的有效性，并记录团队对模拟事件的响应情况，以便在将来的模拟中可以不断跟踪进度。

资源

相关文档：

- AWS 事件响应指南

相关视频：

- AWS 实际演练 – 安全版
- Running effective security incident response simulations

SEC10-BP08 建立从事件中吸取经验教训的框架

实现经验教训总结框架和根本原因分析能力不仅能够有助于提高事件响应能力，还有助于防止事件再次发生。通过从每次事件中吸取教训，您可以避免重复同样的错误、泄露或错误配置，这不仅可以改善您的安全态势，还可以最大限度地减少因可预防的情况而损失的时间。

在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

重要的是要实现一个经验教训总结框架，大体上确立并实现以下几点：

- 何时总结经验教训？
- 总结经验教训的过程涉及什么？

- 如何总结经验教训？
- 谁参与了这个过程，具体情况如何？
- 如何确定需要改进的领域？
- 如何确保有效跟踪和实施改进措施？

该框架不应关注或指责个人，而应侧重于改进工具和流程。

实施步骤

除了前面列出的大体上的成果外，重要的是要确保提出正确的问题，以便从流程中获得最大价值（可以带来切实可行的改进的信息）。请考虑以下问题，以便于您启动经验教训讨论：

- 发生了什么事件？
- 何时首次发现该事件？
- 是如何发现的？
- 哪些系统针对该活动发出了警报？
- 涉及哪些系统、服务和数据？
- 具体发生了什么？
- 哪些地方做得好？
- 哪些地方做得不好？
- 哪些流程或程序出现问题或未能扩展以应对事件？
- 以下方面有哪些地方有待改进：
  - 人员
    - 需要联系的人是否真的可以联系上，联系名单是否是最新名单？
    - 相应人员是否缺少有效应对和调查事件所需的培训或能力？
    - 相应的资源是否已就绪并随时可用？
  - 流程
    - 是否遵循了流程和程序？
    - 是否针对这种事件记录并提供了流程和程序？
    - 是否缺少必要的流程和程序？
    - 响应人员是否能够及时获得所需的信息来处理问题？
  - 技术
    - 现有警报系统是否能有效识别活动并发出警报？

- 我们如何将检测时间缩短 50%？
- 现有警报是否需要改进，或者是否需要针对这种事件设置新的警报？
- 现有工具是否允许对事件进行有效调查（搜索/分析）？
- 怎样才能更快地识别这种事件？
- 如何防止这种事件再次发生？
- 谁是改进计划的负责人，如何检验改进计划的执行情况？
- 实施和测试额外监控或预防性控制机制和流程的时间表是怎样的？

此列表并非详尽无遗，但旨在作为一个起点，确定组织和业务需求是什么，以及如何分析这些需求，以便最有效地从事件中吸取经验教训，并不断改进您的安全态势。最重要的是，该列表开始将经验教训作为事件响应流程、文档和利益相关方期望的标准组成部分。

资源

相关文档：

- 《AWS Security Incident Response Guide》– Establish a framework for learning from incidents
- NCSC CAF 指南：总结经验教训

应用程序安全性

问题

- SEC 11. 如何在整个设计、开发和部署生命周期中纳入并验证应用程序的安全属性？

SEC 11. 如何在整个设计、开发和部署生命周期中纳入并验证应用程序的安全属性？

开展员工培训、执行自动化测试、了解依赖项，并验证各种工具和应用程序的安全属性，有助于降低生产工作负载中出现安全问题的可能性。

最佳实践

- SEC11-BP01 应用程序安全性培训
- SEC11-BP02 在整个开发和发布生命周期中执行自动化测试
- SEC11-BP03 定期执行渗透测试
- SEC11-BP04 实施代码审查
- SEC11-BP05 集中管理服务，方便获取软件包和依赖项

- SEC11-BP06 以编程方式部署软件
- SEC11-BP07 定期评测管道的安全属性
- SEC11-BP08 建立规程，让工作负载团队负责安全领域

SEC11-BP01 应用程序安全性培训

为您的团队提供有关安全开发和操作实践的培训，这有助于他们构建安全、高质量的软件。这种做法有助于您的团队在开发生命周期的早期预防、检测和修复安全问题。考虑开展涵盖威胁建模、安全编码实践以及使用服务进行安全配置和操作等方面的培训。通过自助服务资源为您的团队提供培训机会，并定期收集他们的反馈以实现持续改进。

期望结果：您为团队配备必要的知识和技能，让他们从一开始设计和构建软件时就考虑安全性。通过开展威胁建模和安全开发实践方面的培训，您的团队对潜在的安全风险以及如何在软件开发生命周期（SDLC）中缓解此类风险有了深刻的了解。这种积极主动的实现安全性的方法是团队文化的一部分，您可以尽早发现和修复潜在的安全问题。因此，您的团队可以更高效地交付高质量、安全的软件和功能，从而加快整体交付时间。您的组织内部有一种协作和包容的安全文化，在这种文化中，安全的所有权由所有构建者共享。

常见反模式：

- 您等到安全审查阶段，才考虑系统的安全属性。
- 您将所有安全决策都交给中心安全团队。
- 您没有传达在 SDLC 中做出的决策如何与组织的总体安全期望或策略相关联。
- 您过迟参与安全审查过程。

建立此最佳实践的好处：

- 在开发周期的早期更好地了解组织对安全性的要求。
- 能够更快地识别和修复潜在的安全问题，从而更快地交付功能。
- 提高软件和系统的质量。

在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

要构建安全且高质量的软件，请向您的团队提供培训，使其了解有关安全开发和运行应用程序的常见实践。这种实践有助于您的团队在开发生命周期的早期预防、检测和修复安全问题，从而加快交付时间。

要实现这一实践，可以考虑使用 Threat Modeling Workshop 等 AWS 资源，对您的团队进行威胁建模培训。威胁建模有助于您的团队了解潜在的安全风险，并从一开始设计系统时就考虑安全性。此外，您还可以提供 AWS 培训和认证、行业或 AWS 合作伙伴就安全开发实践开展的培训。有关大规模设计、开发、保护和高效运营的全面方法的更多详细信息，请参阅 AWS DevOps Guidance。

明确定义并传达组织的安全审查流程，并概述您的团队、安全团队和其它利益相关方的职责。发布自助服务指南、代码示例和模板，演示如何满足您的安全要求。可以使用诸如 AWS CloudFormation、AWS Cloud Development Kit (AWS CDK) (AWS CDK) Constructs 和 Service Catalog 之类的 AWS 服务来提供预先批准的安全配置，并减少对自定义设置的需求。

定期收集团队对安全审查流程和培训体验的反馈，并利用这些反馈不断改进。开展 GameDay 或漏洞狂欢活动，以识别和解决安全问题，同时提高团队的技能。

实施步骤

1. 确定培训需求：通过调查、代码审查或与团队成员展开讨论，评测团队中有关安全开发实践的当前技能水平和知识差距。
2. 规划培训：根据确定的需求，制定涵盖诸如威胁建模、安全编码实践、安全测试和安全部署实践等相关主题的培训计划。利用 Threat Modeling Workshop、AWS 培训和认证以及行业或 AWS 合作伙伴培训计划等资源。
3. 安排和提供培训：为您的团队安排定期的培训课程或讲习会。这些课程可以由讲师指导或自定进度，具体取决于团队的偏好和空闲时间。鼓励动手练习和实际示例来增强学习效果。
4. 定义安全审查流程：与您的安全团队和其他利益相关者合作，来为您的应用程序明确定义安全审查流程。记录参与该流程的每个团队或个人的责任，包括您的开发团队、安全团队和其他相关的利益相关者。
5. 创建自助服务资源：开发自助服务指南、代码示例和模板，演示如何满足组织的安全要求。可以使用诸如 CloudFormation、AWS CDK Constructs 和 Service Catalog 之类的 AWS 服务来提供预先批准的安全配置，并减少对自定义设置的需求。
6. 沟通和社交化：有效地向您的团队传达安全审查流程和可用的自助服务资源。举办培训课程或讲习会，让他们熟悉这些资源，并验证他们是否了解如何使用这些资源。
7. 收集反馈并改进：定期收集团队对安全审查流程和培训体验的反馈。利用这些反馈来确定需要改进的领域，并不断完善培训材料、自助服务资源和安全审查流程。
8. 开展安全练习：组织 GameDay 或漏洞狂欢活动，以识别和解决应用程序中的安全问题。这些练习不仅有助于发现潜在的漏洞，还可以为您的团队提供实践学习机会，来增强他们在安全开发和运营方面的技能。
9. 持续学习和改进：鼓励您的团队及时了解最新的安全开发实践、工具和技术。定期审核和更新您的培训材料和资源，以反映不断变化的安全形势和最佳实践。

资源

相关最佳实践：

- SEC11-BP08 建立规程，让工作负载团队负责安全领域

相关文档：

- AWS 培训 和认证
- 如何看待云安全治理
- 如何处理威胁建模
- 加速培训 – AWS Skills Guild
- AWS DevOps Sagas

相关视频：

- 主动式安全性：注意事项和方法

相关示例：

- 有关威胁建模的讲习会
- 开发人员的行业意识

相关服务：

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)（AWS CDK）Constructs
- 服务目录

SEC11-BP02 在整个开发和发布生命周期中执行自动化测试

在整个开发和发布生命周期中自动测试安全属性。自动化使得在发布软件前，更加容易始终如一反复识别软件中可能存在的问题，进而减少所提供的软件中存在安全问题的风险。

期望结果：自动测试的目标是提供一种程序化方式，在整个开发生命周期中常常尽早检测潜在问题。自动执行回归测试时，您可以重新运行功能测试和非功能测试，确认以前测试过的软件在更改后仍按预期

执行。定义安全性单元测试以检查常见的错误配置（如身份验证中断或缺失）时，可以在开发过程的早期识别并修复这些问题。

测试自动化根据应用程序的要求和期望的功能，使用专门构建的测试用例进行应用程序验证。自动测试的结果基于将生成的测试输出与其各自的预期输出进行比较，从而加快整个测试生命周期。回归测试和单元测试套件等测试方法最适合自动化。自动执行安全属性的测试使构建者无需等待安全审查即可自动接收反馈。静态或动态代码分析形式的自动化测试可以提高代码质量，并帮助在开发生命周期的早期检测潜在的软件问题。

常见反模式：

- 不传达自动化测试的测试用例和测试结果。
- 就在发布之前执行自动化测试。
- 使用自动化测试用例来应对经常变化的需求。
- 未能就如何处理安全测试的结果提供指导。

建立此最佳实践的好处：

- 减少对评估系统安全属性的人员的依赖。
- 在多个工作流程中得到一致的结果可提高一致性。
- 降低在生产软件中引入安全问题的可能性。
- 由于及早发现软件问题，可以缩短检测和修复之间的时间。
- 增加多个工作流中的系统或重复行为的可见性，可用于促进组织范围内的改进。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

在构建软件时，采用各种软件测试机制，以确保根据应用程序的业务逻辑测试应用程序的功能要求和非功能要求（重点关注应用程序可靠性、性能和安全性）。

静态应用程序安全性测试（SAST）分析源代码是否存在异常安全模式，并指出容易出现缺陷的代码。SAST 依赖于文档（需求规范、设计文档和设计规范）和应用程序源代码等静态输入来测试一系列已知的安全问题。静态代码分析器可以帮助加快大量代码的分析。NIST Quality Group 对源代码安全性分析器进行了比较，包括针对字节码扫描器和二进制码扫描器的开源工具。

动态分析安全测试（DAST）方法针对正在运行的应用程序执行测试，以识别潜在的意外行为，能够对静态测试作出补充。动态测试可用于检测通过静态分析无法检测到的潜在问题。通过在代码存储库、构建和管道阶段进行测试，您可以检查出不同类型的潜在问题，防止这些问题进入到代码中。Amazon Q 开发者版在构建器的 IDE 中提供代码建议，包括安全扫描。Amazon CodeGuru 安全防御工具可以识别应用程序开发过程中的严重问题、安全问题以及难以发现的错误，并提供可提高代码质量的建议。提取软件物料清单（SBOM）还可让您提取一份正式记录，其中包含构建软件时使用的各个组件的详细信息和关系。这使您可以为漏洞管理提供信息，并快速识别软件或组件依赖关系以及供应链风险。

通过开发人员安全性讲习会，学会使用 AWS 开发人员工具（例如，AWS CodeBuild、AWS CodeCommit 和 AWS CodePipeline）来自动执行发布管道，包括 SAST 和 DAST 测试方法。

在 SDLC 中，建立一个迭代过程，其中包括与安全团队一起定期审查应用程序。在发布准备情况审查过程中，应该处理和验证从这些安全审查中收集到的反馈。这些审查建立了健壮的应用程序安全态势，并为构建者提供切实可行的反馈，以解决潜在问题。

实施步骤

- 实施一致的 IDE、代码审查和 CI/CD 工具，其中包括安全测试。
- 考虑在 SDLC 中的哪个阶段适合阻塞管道，而不仅仅是通知构建者需要修复问题。
- Automated Security Helper (ASH) 是开源代码安全扫描工具的一个示例。
- 使用自动化工具（例如，与开发人员 IDE 集成的 Amazon Q 开发者版，以及用于在提交时扫描代码的 Amazon CodeGuru 安全防御工具）执行测试或代码分析，有助于构建者适时获得反馈。
- 使用 AWS Lambda 构建应用程序时，您可以使用 Amazon Inspector 来扫描函数中的应用程序代码。
- 当 CI/CD 管道中包括自动化测试时，您应该使用工单系统来跟踪软件问题的通知和修正。
- 对于可能生成结果的安全测试，链接到补救指南可帮助构建者提高代码质量。
- 定期分析使用自动化工具获得的结果，以确定下一个自动化、构建者培训或认知宣传活动的优先级。
- 要提取 SBOM 作为 CI/CD 管道的一部分，请使用 Amazon Inspector SBOM Generator，来为 CycloneDX SBOM 格式的归档、容器映像、目录、本地系统以及编译后的 Go 和 Rust 二进制文件生成 SBOM。

资源

相关最佳实践：

- DevOps Guidance: DL.CR.3 Establish clear completion criteria for code tasks

相关文档：

- 持续交付和持续部署

- AWS DevOps 能力合作伙伴
- 适用于应用程序安全性的 AWS 安全能力合作伙伴
- 选择 Well-Architected CI/CD 方法
- Secrets detection in Amazon CodeGuru Security
- Amazon CodeGuru Security Detection Library
- 通过有效的治理加快 AWS 上的部署
- AWS 方法如何自动实现无需干预的安全部署
- How Amazon CodeGuru Security helps you effectively balance security and velocity

相关视频：

- 无需干预：在亚马逊自动实现持续交付管道
- 自动执行跨账户 CI/CD 管道
- The Software Development Prcess at Amazon
- Testing software and systems at Amazon

相关示例：

- 开发人员的行业意识
- Automated Security Helper (ASH)
- AWS CodePipeline Governance - Github

SEC11-BP03 定期执行渗透测试

定期对软件执行渗透测试。此机制有助于识别无法通过自动化测试或人工代码审查加以检测的潜在软件问题。它还有助于您了解检测控制的有效性。渗透测试应设法确定软件是否会以意外方式执行，例如公开应受保护的数据，或者授予比预期更广泛的权限。

期望结果：使用渗透测试来检测、修复和验证应用程序的安全属性。在软件开发生命周期（SDLC）中应定期执行计划的渗透测试。在发布软件之前应处理渗透测试的结果。您应该分析渗透测试的结果，以

确定是否存在使用自动化可以发现的问题。拥有包括主动反馈机制的定期且可重复渗透测试流程，有助于为构建者提供指导并提高软件质量。

常见反模式：

- 仅对已知或普遍存在的安全问题进行渗透测试。
- 未使用相关的第三方工具和库对应用程序执行渗透测试。
- 仅对软件包安全问题进行渗透测试，而不评估已实施的业务逻辑。

建立此最佳实践的好处：

- 在发布之前增强对软件安全属性的信心。
- 有机会确定首选的应用程序模式，从而提高软件质量。
- 获得一个反馈环路，在开发周期早期确定自动化或额外培训可以在哪些方面改进软件的安全属性。

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

渗透测试是一项结构化安全测试练习，让您可以运行计划的安全漏洞方案，以便检测、修复和验证安全控制机制。渗透测试从侦察开始，在这个过程中，根据应用程序的当前设计及其依赖项收集数据。生成并运行特定于安全方面的测试方案的精选列表。这些测试的主要目的是发现应用程序中的安全问题，有人会利用这些安全问题来获得对环境的非预期访问，或未经授权访问数据。当推出新功能时，或者应用程序的功能或技术实施方面发生重大变更时，您应该执行渗透测试。

您应该确定在开发生命周期的哪个阶段执行渗透测试最为合适。应当尽量晚些时候执行此测试，以便系统功能接近预期的发布状态，但也要留有足够的时间来修复任何问题。

实施步骤

- 采用结构化流程来确定渗透测试的范围，让这个流程基于威胁模型是保留场景相关性的好方法。
- 确定在开发周期的什么阶段执行渗透测试较为合适。这个阶段应该是在应用程序预期改动很细微，但仍留有足够时间进行修复的时候。
- 为构建者提供以下方面的培训：从渗透测试结果中可以期待获得什么，以及如何获得有关修复的信息。
- 使用工具自动执行常见或可重复的测试，从而加快渗透测试的速度。
- 分析渗透测试结果，以便确定系统性安全问题，并使用此数据为额外的自动化测试和正在进行的构建者培训提供信息。

资源

相关最佳实践：

- SEC11-BP01 应用程序安全性培训
- SEC11-BP02 在整个开发和发布生命周期中执行自动化测试

相关文档：

- AWS 渗透测试提供有关 AWS 上的渗透测试的详细指导
- 通过有效的治理加快 AWS 上的部署
- AWS 安全能力合作伙伴
- 使 AWS Fargate 上的渗透测试架构实现现代化改造
- AWS Fault injection Simulator

相关示例：

- 使用 AWS CodePipeline 自动执行 API 测试（GitHub）
- 自动安全助手（GitHub）

SEC11-BP04 实施代码审查

实施代码审查来协助验证正在开发的软件的质量和安全性。代码审查包括让除原始代码作者之外的团队成员审查代码是否存在潜在问题、漏洞，以及是否遵守编码标准和最佳实践。此过程有助于发现原始开发人员可能已忽略的错误、不一致和安全漏洞。运用自动化工具来协助进行代码审查。

期望结果：在开发过程中纳入代码审查，以提高正在编写的软件的质量。您可以通过在代码审查期间确定的经验教训，来提高团队中经验不足的成员的技能。您可以使用自动化工具和测试来发现自动化的机会并支持代码审查流程。

常见反模式：

- 您在部署前不执行代码审查。
- 您让同一个人编写和审查代码。
- 您不使用自动化和工具来协助或编排代码审查。
- 在构建者审查代码之前，您未对他们进行应用程序安全方面的培训。

建立此最佳实践的好处：

- 提高代码质量。
- 通过重复利用通用方法提高代码开发的一致性。
- 减少在渗透测试和后续阶段发现的问题。
- 改进团队内部的知识传授。

在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

代码审查有助于在开发过程中验证软件的质量和安全性。手动审查包括让除原始代码作者之外的团队成员审查代码是否存在潜在问题、漏洞，以及是否遵守编码标准和最佳实践。此过程有助于发现原始开发人员可能已忽略的错误、不一致和安全漏洞。

考虑使用 Amazon CodeGuru Security 来协助实施自动代码审查。CodeGuru 安全防御工具使用机器学习和自动推理来分析您的代码，并识别潜在的安全漏洞和编码问题。将自动化代码审查与现有代码存储库以及持续集成/持续部署（CI/CD）管道相集成。

实施步骤

1. 建立代码审查流程：
   - 定义何时应进行代码审查，例如，在将代码合并到主分支之前或部署到生产环境之前。
   - 确定谁应该参与代码审查流程，例如团队成员、高级开发人员和安全专家。
   - 决定代码审查方法，包括要使用的流程和工具。
2. 设置代码审查工具：
   - 评估并选择符合团队需求的代码审查工具，例如 GitHub 拉取请求或 CodeGuru 安全防御工具。
   - 将所选工具与您现有的代码存储库和 CI/CD 管道相集成。
   - 配置工具来强制实施代码审查要求，例如最少审查人员数量和审批规则。
3. 定义代码审查清单和指南：
   - 创建代码审查清单或指南，其中概述了应审查的内容。考虑诸如代码质量、安全漏洞、对编码标准的遵守情况以及性能等因素。
   - 与开发团队共享清单或指南，并确认每个人都了解预期结果。
4. 对开发人员进行代码审查最佳实践培训：
   - 为您的团队提供有关如何进行有效代码审查的培训。

- 让您的团队了解应用程序安全原则和在审查期间需要注意的常见漏洞。
- 鼓励开展知识共享和配对编程课程，以提高经验不足的团队成员的技能。

5. 实施代码审查流程：
   - 将代码审查步骤集成到开发工作流程中，例如创建拉取请求和分配审查人员。
   - 要求在合并或部署之前对代码更改进行代码审查。
   - 鼓励在审查过程中进行开放式沟通和提供建设性反馈。
6. 监控和改进：
   - 定期检查代码审查流程的有效性，并收集团队的反馈。
   - 确定实施自动化或工具改进的机会，以简化代码审查流程。
   - 根据经验教训和行业最佳实践，不断更新和完善代码审查清单或指南。
7. 培养代码审查文化：
   - 强调代码审查对于维护代码质量和安全性的重要性。
   - 庆祝代码审查流程取得成功和获得经验教训。
   - 鼓励营造一个协作和支持的环境，让开发人员能够舒适地提出和接受反馈。

资源

相关最佳实践：

- SEC11-BP02 在整个开发和发布生命周期中执行自动化测试

相关文档：

- DevOps Guidance: DL.CR.2 Perform peer review for code changes
- 关于 GitHub 中的拉取请求

相关示例：

- Automate code reviews with Amazon CodeGuru Security
- Automating detection of security vulnerabilities and bugs in CI/CD pipelines using Amazon CodeGuru Security CLI

相关视频：

- Continuous improvement of code quality with Amazon CodeGuru Security

SEC11-BP05 集中管理服务，方便获取软件包和依赖项

提供集中式服务，方便您的团队获取软件包和其它依赖项。通过采取这种做法，可以在将软件包纳入所编写的软件之前，对软件包进行验证；另外，还可以为分析贵组织所使用的软件提供数据来源。

期望结果：除了您编写的代码之外，还可以使用外部软件包来构建工作负载。这使您更容易实现重复使用的功能，例如 JSON 解析器或加密库。您将这些软件包和依赖项的来源集中在一起，以便安全团队可以在使用软件包和依赖项之前对其来源进行验证。您将此方法与手动和自动测试流程结合使用，来增强对所开发软件的质量的信心。

常见反模式：

- 您从互联网上的任意存储库中提取软件包。
- 您在将新软件包提供给构建者之前，未对其进行测试。

建立此最佳实践的好处：

- 更好地了解正在构建的软件中使用了哪些软件包。
- 了解谁使用了哪些软件包后，在需要更新软件包时，能够向工作负载团队发出通知。
- 降低软件中存在问题软件包的风险。

在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

以构建者易于使用的方式为软件包和依赖项提供集中管理服务。集中管理服务可以在逻辑上集中，而不用作为一个整体系统来实施。利用此方法，您可以通过满足构建者需求的方法来提供服务。您应该实施一种有效的方法：在发生更新或出现新需求时将软件包添加到存储库。AWS CodeArtifact 等 AWS 服务或类似的 AWS 合作伙伴解决方案提供了一种实现此功能的方法。

实施步骤

- 实施可在用于开发软件的所有环境中使用的逻辑集中式存储库服务。
- 在 AWS 账户 分配过程中包括对存储库的访问权限。
- 构建自动化以在存储库中发布软件包之前对其进行测试。

- 维护最常用软件包、语言和更改量最大的团队的指标。
- 为构建者团队提供一种自动化机制来请求新软件包和提供反馈。
- 定期扫描存储库中的软件包，以确定新发现的问题的潜在影响。

资源

相关最佳实践：

- SEC11-BP02 在整个开发和发布生命周期中执行自动化测试

相关文档：

- DevOps Guidance: DL.CS.2 Sign code artifacts after each build
- Supply chain Levels for Software Artifacts (SLSA)

相关示例：

- 通过有效的治理加快 AWS 上的部署
- 使用 CodeArtifact Package Origin Control 工具包加强软件包的安全性
- 多区域软件包发布管道（GitHub）
- 使用 AWS CodePipeline 在 AWS CodeArtifact 上发布 Node.js 模块（GitHub）
- AWS CDK Java CodeArtifact 管道示例（GitHub）
- 使用 AWS CodeArtifact 分发专用 .NET NuGet 包（GitHub）

相关视频：

- 主动式安全性：注意事项和方法
- AWS 安全理念（re:Invent 2017）
- 当安全、保障和紧迫性都很重要时：处理 Log4Shell

SEC11-BP06 以编程方式部署软件

尽可能以编程方式部署软件。通过采取这种做法，可以降低由于人为错误导致部署失败或引入意外问题的可能性。

期望结果：您测试的工作负载版本就是您部署的版本，每次都一致地执行部署。您可以将工作负载的配置外部化，这有助于您无需更改即可部署到不同的环境。您使用软件包的加密签名来验证环境之间没有任何变化。

常见反模式：

- 手动将软件部署到生产环境中。
- 手动对软件进行更改，以适应不同的环境。

建立此最佳实践的好处：

- 增强对软件发布过程的信心。
- 降低了失败的更改对业务功能造成影响的风险。
- 由于更改风险降低，从而加快了发布节奏。
- 针对部署过程中的意外事件的自动回滚功能。
- 能够以加密方式证明所测试的软件是部署的软件。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要维护稳健且可靠的应用程序基础设施，应实施安全和自动化的部署实践。这种做法涉及从生产环境中移除持久的人员访问权限，使用 CI/CD 工具进行部署，以及将特定于环境的配置数据外部化。通过采用这种方法，您可以增强安全性，降低人为错误的风险，并简化部署流程。

您可以构建自己的 AWS 账户结构，以便从生产环境中移除持久的人员访问权限。这种做法可以最大限度地降低未经授权的更改或意外修改的风险，从而提高生产系统的完整性。您可以使用 AWS CodeBuild 和 AWS CodePipeline 之类的 CI/CD 工具来执行部署，而不是使用直接人员访问权限。您可以使用这些服务来自动执行构建、测试和部署流程，从而减少手动干预并提高一致性。

为了进一步增强安全性和可追溯性，您可以在测试应用程序包后对其进行签名，并在部署期间验证这些签名。为此，请使用诸如 AWS Signer 或 AWS Key Management Service（AWS KMS）之类的加密工具。通过对软件包进行签名和验证，您可以确保仅将经过授权和验证的代码部署到您的环境中。

此外，您的团队可以适当地设计工作负载，以便从外部源（例如 AWS Systems Manager Parameter Store）获得特定于环境的配置数据。这种做法可将应用程序代码与配置数据分开，这有助于您独立管理和更新配置，而无需修改应用程序代码本身。

为简化基础设施的预置和管理，可以考虑使用基础设施即代码（IaC）工具，例如 AWS CloudFormation 或 AWS CDK。可以使用这些工具来定义基础设施即代码，从而提高不同环境中部署的一致性和可重复性。

考虑使用金丝雀部署来验证软件是否成功部署。金丝雀部署涉及在部署到整个生产环境之前，先对一部分实例或用户推出更改。然后，您可以监控更改的影响并在必要时进行回滚，从而最大限度地降低出现广泛问题的风险。

按照 Organizing Your AWS Environment Using Multiple Accounts 白皮书中列出的建议进行操作。本白皮书提供了有关将环境（例如开发、暂存和生产）分离到不同 AWS 账户的指导，这种划分可以进一步增强安全性和隔离性。

实施步骤

1. 设置 AWS 账户结构：
   - 按照 Organizing Your AWS Environment Using Multiple Accounts 白皮书中的指导，为不同的环境（例如开发、暂存和生产）创建单独的 AWS 账户。
   - 为每个账户配置适当的访问控制和权限，以限制人员对生产环境的直接访问权限。
2. 实施 CI/CD 管道：
   - 使用 AWS CodeBuild 和 AWS CodePipeline 之类的服务设置 CI/CD 管道。
   - 将管道配置为自动构建、测试应用程序代码，并将其部署到相应的环境。
   - 将代码存储库与 CI/CD 管道相集成，以实现版本控制和代码管理。
3. 签署并验证应用程序包：
   - 在应用程序包经过测试和验证后，使用 AWS Signer 或 AWS Key Management Service（AWS KMS）对其进行签名。
   - 在将应用程序包部署到目标环境之前，配置部署过程来验证应用程序包的签名。
4. 使配置数据外部化：
   - 将特定于环境的配置数据存储在 AWS Systems Manager Parameter Store 中。
   - 修改应用程序代码，以便在部署或运行时期间从 Parameter Store 检索配置数据。
5. 实施基础设施即代码（IaC）：
   - 使用 AWS CloudFormation 或 AWS CDK 之类的 IaC 工具来定义和管理基础设施即代码。
   - 创建 CloudFormation 模板或 CDK 脚本，来为应用程序预置和配置必要的 AWS 资源。
   - 将 IaC 与 CI/CD 管道集成，以便在应用程序代码更改的同时自动部署基础设施变更。
6. 实施金丝雀部署：

- 将您的部署流程配置为支持金丝雀部署，也即，在将更改部署到整个生产环境之前，先向一部分实例或用户推出这些更改。
- 使用诸如 AWS CodeDeploy 或 AWS ECS 之类的服务来管理金丝雀部署并监控更改的影响。
- 实施回滚机制，如果在金丝雀部署期间检测到问题，则还原到之前的稳定版本。

7. 监控和审计：
   - 设置监控和日志记录机制，来跟踪部署、应用程序性能和基础设施更改。
   - 使用诸如 Amazon CloudWatch 和 AWS CloudTrail 之类的服务来收集和分析日志和指标。
   - 实施审计和合规性检查，以验证对安全最佳实践和监管要求的遵守情况。
8. 持续改进：
   - 定期审查和更新您的部署实践，并纳入从以前的部署中吸取的反馈和经验教训。
   - 尽可能地实现部署过程自动化，以减少手动干预和潜在的人为错误。
   - 与跨职能团队（例如运营或安全）合作，来协调并持续改进部署实践。

通过执行这些步骤，可以在 AWS 环境中实施安全的自动化部署实践，从而增强安全性，降低人为错误的风险，并简化部署过程。

资源

相关最佳实践：

- SEC11-BP02 在整个开发和发布生命周期中执行自动化测试
- DL.CI.2 Trigger builds automatically upon source code modifications

相关文档：

- 通过有效的治理加快 AWS 上的部署
- 自动实现无需干预的安全部署
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys
- Code Signing, a Trust and Integrity Control for AWS Lambda

相关视频：

- 无需干预：在亚马逊自动实现持续交付管道

相关示例：

- Blue/Green deployments with AWS Fargate

SEC11-BP07 定期评测管道的安全属性

对您的管道运用 Well-Architected 安全性支柱原则，尤其注意权限分离。定期评测管道基础设施的安全属性。通过有效管理管道的安全性，可以确保通过管道的软件的安全性。

期望结果：用于构建和部署软件的管道遵循与环境中任何其它工作负载相同的建议做法。您在管道中实施的测试不可由使用这些测试的团队编辑。您只向管道授予它们使用临时凭证执行的部署所需的权限。您可以实施安全措施来防止将管道部署到错误的环境中。您可以将管道配置为发出状态，以便可以验证构建环境的完整性。

常见反模式：

- 构建者可以绕过安全测试。
- 用于部署管道的权限过于宽松。
- 未将管道配置为验证输入。
- 不定期审查与 CI/CD 基础设施关联的权限。
- 使用长期或硬编码凭证。

建立此最佳实践的好处：

- 对通过管道构建和部署的软件的完整性有了更大的信心。
- 在出现可疑活动时可以停止部署。

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

部署管道是软件开发生命周期的关键组成部分，应遵循与环境中任何其它工作负载相同的安全原则和做法。这包括实施适当的访问控制、验证输入，以及定期审查和审计与 CI/CD 基础设施关联的权限。

确认负责构建和部署应用程序的团队无法编辑或绕过在管道中实施的安全测试和检查。这种关注点分离有助于维护构建和部署过程的完整性。

首先，可以考虑使用 AWS Deployment Pipelines Reference Architecture。此参考架构为在 AWS 上构建 CI/CD 管道提供了安全且可扩展的基础。

此外，可以使用 AWS Identity and Access Management Access Analyzer 等服务为管道权限生成最低权限 IAM 策略，并作为管道中验证工作负载权限的一个步骤。这有助于验证管道和工作负载是否仅具有其特定功能所需的必要权限，从而降低未经授权的访问或操作的风险。

实施步骤

- 从 AWS 部署管道参考架构开始。
- 考虑使用 AWS IAM Access Analyzer 以编程方式生成管道的最低权限 IAM 策略。
- 将管道与监控和警报集成在一起，以便在发生意外或异常活动时您会得到通知，对于 AWS 托管服务，Amazon EventBridge 允许您将数据路由到目标，例如 AWS Lambda 或 Amazon Simple Notification Service（Amazon SNS）。

资源

相关文档：

- AWS 部署管道参考架构
- 监控 AWS CodePipeline
- AWS CodePipeline 的安全最佳实践

相关示例：

- DevOps 监控控制面板（GitHub）

SEC11-BP08 建立规程，让工作负载团队负责安全领域

建立规程或机制，使构建者团队能够针对创建的软件作出安全决策。这些决策仍然需要由安全团队通过审查加以验证，但让构建者团队负责安全领域可以构建速度更快、安全性更高的工作负载。此机制还可促进负责任文化，进而对所构建系统的运营产生积极影响。

期望结果：您的团队中嵌入了安全所有权和决策权。您要么已经对团队进行了如何考虑安全的培训，要么已通过内嵌或关联的安全人员增强了团队。因此，您的团队在开发周期的早期就做出了更高质量的安全决策。

常见反模式：

- 将所有安全设计决策交给安全团队。

- 在开发过程中没有及早满足安全要求。
- 没有从构建者和安全人员那里获得关于计划运营的反馈。

建立此最佳实践的好处：

- 缩短完成安全审查的时间。
- 减少等到安全审查阶段才检测到安全问题的情况。
- 提高所编写软件的整体质量。
- 有机会识别和了解系统性问题或高价值改进领域。
- 进行安全审查后，发现的问题可以在早期进行修复，从而减少所需的返工量。
- 提升对安全功能的认知。

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

从 SEC11-BP01 应用程序安全性培训 中的指导开始。然后确定您认为可能最适合您组织的计划的运营模式。两个主要模式是对构建者进行培训，或在构建者团队中加入安全人员。确定初始方法后，应使用单个或一小组工作负载团队进行试点，以证明该模式适用于您的组织。来自组织的构建者和安全团队的领导层支持有助于计划的成功交付。在构建此计划时，重要的是选择可以用来显示项目价值的指标。了解 AWS 如何解决这个问题是一个很好的学习经验。这个最佳实践非常注重组织变革和文化。您使用的工具应支持构建者和安全社区之间的协作。

实施步骤

- 首先对构建者进行应用程序安全性培训。
- 创建一个社区和入门培训计划来对构建者进行培训。
- 为计划选择一个名称。通常使用守护者、拥护者或倡导者。
- 确定要使用的模式：培训构建者、加入安全工程师或具有相关性安全角色。
- 从安全性、构建者和可能的其他相关团体中确定项目发起人。
- 跟踪参与计划的人数、审查所花时间以及来自构建者和安全人员的反馈等指标。使用这些指标来作出改进。

资源

相关最佳实践：

- SEC11-BP01 应用程序安全性培训
- SEC11-BP02 在整个开发和发布生命周期中执行自动化测试

相关文档：

- 如何处理威胁建模
- 如何看待云安全治理
- How AWS built the Security Guardians program, a mechanism to distribute security ownership
- How to build a Security Guardians program to distribute security ownership

相关视频：

- 主动式安全性：注意事项和方法
- AppSec tooling and culture tips from AWS and Toyota Motor North America

可靠性

可靠性支柱涵盖相关工作负载按照计划正确而稳定执行其预期功能的能力。有关具体实施的说明性指导，请参阅《可靠性支柱白皮书》。

最佳实践领域

- 基本原理
- 工作负载架构
- 变更管理
- 故障管理

基本原理

问题

- REL 1. 如何管理服务配额和约束？
- REL 2. 您如何规划网络拓扑？

REL 1. 如何管理服务配额和约束？

对于基于云的工作负载架构，存在服务配额（也称为服务限制）。这些限额的存在是为了防止意外调配超出所需的资源，并限制 API 操作的请求率，以保护服务不被滥用。还存在资源限制，例如光缆的比特传输速率或物理磁盘的存储量。

最佳实践

- REL01-BP01 了解服务配额和约束
- REL01-BP02 跨多个账户和区域管理服务配额
- REL01-BP03 通过架构适应固定服务配额和约束
- REL01-BP04 监控和管理配额
- REL01-BP05 自动管理配额
- REL01-BP06 确保当前配额与最大使用量之间存在足够的差距来应对失效转移

REL01-BP01 了解服务配额和约束

了解工作负载架构的默认配额并管理配额提高请求。了解哪些云资源约束（如磁盘或网络）可能会对您产生影响。

期望结果：客户可以通过实施适当的指导方针来监控关键指标、基础设施审查和自动化修复步骤，验证是否未达到可能导致服务性能下降或中断的服务配额和约束，从而防止其 AWS 账户 中的服务性能下降或中断。

常见反模式：

- 在不了解所用服务的硬配额或软配额及其限制的情况下部署工作负载。
- 在未分析和重新配置必要配额或未事先联系支持部门的情况下，部署替代工作负载。
- 假设云服务没有限制，并且认为可以在不考虑费率、限制、计数、数量的情况下使用服务。
- 假设配额会自动增加。
- 不了解配额请求的流程和时间表。
- 假设每个服务的默认云服务配额在不同区域都是相同的。
- 假设可以突破服务约束，并且系统会自动扩展或提高限制以超出资源约束。
- 没有在流量高峰期测试应用程序，以便对资源的利用率进行压力测试。
- 在没有分析所需资源规模的情况下配置资源。
- 通过选择远远超出实际需求或预期峰值的资源类型来过量配置容量。

- 在新的客户事件或部署新技术之前，不评测新流量水平的容量要求。

建立此最佳实践的好处：对服务配额及资源约束的监视和自动化管理可以主动减少故障。如果不遵循最佳实践，客户服务的流量模式变化可能会导致中断或性能下降。通过监视和管理所有区域和所有账户的这些值，应用程序可以在出现不利或意外事件时具有更好的韧性。

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

服务配额是一项 AWS 服务，可帮助您从一个位置管理超过 250 项 AWS 服务的配额。除了查找配额值，您还可以通过服务配额控制台或 AWS SDK 请求提高配额并跟踪配额。AWS Trusted Advisor 提供的服务配额检查功能会显示服务使用情况，以及某些服务在某些方面的配额。有关每项服务的默认服务配额，请查看相应服务的 AWS 文档（例如参阅 Amazon VPC 配额）。

通过配置使用计划，可在 Amazon API Gateway 内设置某些服务限制，例如节流 API 的速率限制。可通过配置相应服务进行设置的部分限制包括预调配 IOPS、已分配的 Amazon RDS 存储，以及 Amazon EBS 卷分配。Amazon Elastic Compute Cloud 有自己的服务限制控制面板，可帮助您管理实例、Amazon Elastic Block Store 和弹性 IP 地址限制。如果在某应用场景中，服务配额会对应用程序的性能造成影响，而且无法按需求进行调整，请联系 支持 了解是否有解决的办法。

服务配额可以是区域性的，也可以是全局性的。使用达到配额的 AWS 服务不会具有正常使用中预期的行为，并且可能会导致服务中断或性能下降。例如，服务配额限制了一个区域中使用的 DL Amazon EC2 实例的数量。在使用自动扩缩组（ASG）的流量扩缩事件期间，可能会达到该限制。

应定期评测每个账户服务配额的使用情况，确定适合该账户的适当服务限制。这些服务配额作为操作防护机制存在，可防止意外地配置超出所需数量的资源；也用于限制 API 操作的请求率，保护服务不被滥用。

服务约束与服务配额不同。服务约束代表由特定资源类型定义的该资源的限制，这些可能是存储容量（例如，gp2 的大小限制为 1 GB - 16 TB）或磁盘吞吐量。必须对资源类型的约束进行设计，并不断评测可能达到其限制的使用量。如果意外地达到约束条件，账户的应用程序或服务可能会降级或中断。

如果在某应用场景中，服务配额对应用程序的性能造成影响，而且无法根据必要需求进行调整，请联系支持 了解是否有解决办法。有关调整固定配额的更多详细信息，请参阅 REL01-BP03 通过架构适应固定服务配额和约束。

有一些 AWS 服务和工具可以帮助监视和管理服务配额。应该利用这些服务和工具来自动或手动检查配额水平。

- AWS Trusted Advisor 提供的服务配额检查功能会显示服务使用情况，以及某些服务在某些方面的配额。该工具可以帮助识别接近配额的服务。
- AWS Management Console 提供了显示服务配额值，管理、请求新配额，监视配额请求状态以及显示配额历史记录的方法。
- AWS CLI 和 CDK 提供了通过编程方式自动管理和监视服务配额水平和使用情况的方法。

实施步骤

服务配额：

- 请查看 AWS Service Quotas。
- 要了解现有的服务配额，请先确定使用的服务（如 IAM Access Analyzer）。大约有 250 项 AWS 服务由服务配额控制。然后，确定每个账户和区域内可能使用的具体服务配额名称。每个区域大约有 3000 个服务配额名称。
- 使用 AWS Config 来增强此配额分析，找出 AWS 账户 中使用的所有 AWS 资源。
- 使用 AWS CloudFormation 数据来确定使用的 AWS 资源。查看通过 AWS Management Console 或 `list-stack-resources` AWS CLI 命令创建的资源。您还可以查看配置为要在模板自身部署的资源。
- 通过查看部署代码来确定工作负载所需的所有服务。
- 确定适用的服务配额。通过 Trusted Advisor 和服务配额使用能够以编程方式访问的信息。
- 建立自动化的监控方法（参见 REL01-BP02 跨多个账户和区域管理服务配额和 REL01-BP04 监控和管理配额），以便在服务配额接近或已达到限制时发出提醒和通知。
- 建立自动化的程序化方法，用于检查同一账户内某项服务配额是否在某一区域发生变更，但在其他地区未发生变更（参见 REL01-BP02 跨多个账户和区域管理服务配额和 REL01-BP04 监控和管理配额）。
- 自动扫描应用程序日志和指标，确定是否存在任何配额或服务约束错误。如果存在这些错误，则向监控系统发送警报。
- 确定特定服务需要更高的配额后，制定工程设计步骤来计算所需的配额变动（参见 REL01-BP05 自动管理配额）。
- 创建一个配置和批准工作流，以请求更改服务配额。这应该包括在请求被拒绝或部分批准情况下的例外工作流。
- 创建一个工程设计方法，在配置和使用新的 AWS 服务之前，以及在推出到生产环境或加载的环境之前（例如，负载测试账户），审查服务配额。

服务约束：

- 建立监视和度量方法，用于提醒资源接近其资源约束。适当地利用 CloudWatch 进行指标或日志监视。
- 为每个具有对应用程序或系统有意义的约束的资源建立警报阈值。
- 创建工作流和基础设施管理程序，可在约束条件接近利用率时更改资源类型。该工作流应包括负载测试作为最佳实践，可验证新类型是否为新约束条件下的正确资源类型。
- 使用现有的程序和流程，将确定的资源迁移到推荐的新资源类型。

资源

相关最佳实践：

- REL01-BP02 跨多个账户和区域管理服务配额
- REL01-BP03 通过架构适应固定服务配额和约束
- REL01-BP04 监控和管理配额
- REL01-BP05 自动管理配额
- REL01-BP06 确保当前配额与最大使用量之间存在足够的差距来应对失效转移
- REL03-BP01 选择如何划分工作负载
- REL10-BP01 将工作负载部署到多个位置
- REL11-BP01 监控工作负载的所有组件以检测故障
- REL11-BP03 自动修复所有层
- REL12-BP04 使用混沌工程测试韧性

相关文档：

- AWS Well-Archited Framework 的可靠性支柱：可用性
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks（参见“Service limits”小节）
- AWS Answers 上的 AWS Limit Monitor
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase

- Service endpoints and quotas
- Service Quotas User Guide
- 适用于 AWS 的配额监控程序
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS 数据解决方案
- 什么是持续集成？
- 什么是持续交付？
- APN 合作伙伴：可帮助进行配置管理的合作伙伴
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

相关视频：

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

相关工具：

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK

- AWS Systems Manager
- AWS Marketplace

REL01-BP02 跨多个账户和区域管理服务配额

如果您目前使用多个账户或区域，请确保在运行生产工作负载的所有环境中都请求适当的配额。

期望结果：对于跨账户或区域的配置，或者具有使用区域或账户失效转移的韧性设计的配置，服务和应用程序不应受到服务配额耗尽的影响。

常见反模式：

- 允许一个隔离区域内的资源利用率增加，但没有相关机制保持其他隔离区域中的容量。
- 手动单独设置隔离区域中的所有配额。
- 没有考虑到在非主要区域出现性能下降期间，韧性架构（如主动或被动）对未来配额需求的影响。
- 没有定期评估配额，并且不在工作负载运行的每个区域和账户中进行必要更改。
- 不利用配额请求模板在多个地区和账户中请求提高配额。
- 不更新服务配额，因为错误地认为提高配额会像计算预留请求一样产生成本影响。

建立此最佳实践的好处：验证是否可以在区域服务不可用的情况下处理辅助区域或账户中的当前负载。这可以帮助降低在区域丢失期间发生的错误数量或性能下降水平。

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

每个账户的服务配额都可被跟踪。除非另有说明，否则每个配额都针对的是特定的 AWS 区域。除生产环境以外，还要管理所有适用的非生产环境中的配额，避免妨碍测试与开发。保持高度韧性需要持续评测服务配额（无论是自动还是手动）。

由于使用主动/主动、主动/被动 – 热、主动/被动 – 冷以及主动/被动 – 指示灯方法实施设计，跨区域的工作负载越来越多，因此了解所有区域和账户配额级别至关重要。如果服务配额设置正确，过去的流量模式并不总是一个好的指标。

同样重要的是，服务配额名称限制并非对每个区域都始终相同。在一个区域，该值可能是 5，而在另一个区域，该值可能是 10。必须跨越所有相同的服务、账户和区域来管理这些配额，以便在负载状态下提供一致的韧性。

协调不同区域（主动区域或被动区域）之间的所有服务配额差异，并创建流程来持续协调这些差异。被动区域失效转移的测试计划很少扩展到能够满足峰值主动容量，这意味着 GameDay 演练或桌面演练可能无法发现区域之间的服务配额差异，因此也无法保持正确的限制。

服务配额偏差，即某一特定命名配额的服务配额限制在一个区域发生变更但未在所有区域发生变更的情况，这对于跟踪和评测非常重要。应考虑更改在有流量或可能有流量的区域的配额。

- 根据您的服务要求、延迟、法规和灾难恢复（DR）要求选择相关账户和区域。
- 确定跨所有相关账户、区域和可用区的服务配额。限制的范围具体到账户和区域。应对这些值进行比较，了解差异情况。

实施步骤

- 审查可能已经超出风险使用水平的服务配额值。AWS Trusted Advisor 会在超出 80% 和 90% 阈值时提供警报。
- 审查任何被动区域（在主动/被动设计中）的服务配额值。验证在主区域发生故障时，负载能否在辅助区域成功运行。
- 自动评测同一账户的不同区域之间是否发生了任何服务配额偏移，并采取相应的行动来更改限制。
- 如果客户的组织单位（OU）以受支持的方式构建，则应更新服务配额模板，反映应该应用于多个区域和账户的任何配额的变化。
  - 创建模板并将区域与配额变化关联起来。
  - 审查所有现有的服务配额模板，看看是否需要进行任何更改（区域、限制和账户）。

资源

相关最佳实践：

- REL01-BP01 了解服务配额和约束
- REL01-BP03 通过架构适应固定服务配额和约束
- REL01-BP04 监控和管理配额
- REL01-BP05 自动管理配额
- REL01-BP06 确保当前配额与最大使用量之间存在足够的差距来应对失效转移
- REL03-BP01 选择如何划分工作负载
- REL10-BP01 将工作负载部署到多个位置
- REL11-BP01 监控工作负载的所有组件以检测故障

- REL11-BP03 自动修复所有层
- REL12-BP04 使用混沌工程测试韧性

相关文档：

- AWS Well-Archited Framework 的可靠性支柱：可用性
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks（参见“Service limits”小节）
- AWS Answers 上的 AWS Limit Monitor
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- 适用于 AWS 的配额监控程序
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS 数据解决方案
- 什么是持续集成？
- 什么是持续交付？
- APN 合作伙伴：可帮助进行配置管理的合作伙伴
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

相关视频：

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

相关服务：

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP03 通过架构适应固定服务配额和约束

了解不可更改的服务配额、服务约束和物理资源限制。为应用程序和服务设计架构，防止这些限制影响可靠性。

示例包括网络带宽、无服务器函数调用有效负载大小、API Gateway 的节流突发速率，以及并发用户连接至数据库。

期望结果：应用程序或服务在正常流量和高流量条件下按预期执行。它们被设计为在相应资源的固定约束或服务配额的限制范围内工作。

常见反模式：

- 选择使用一项服务资源的设计时，没有意识到设计存在约束，这些约束将导致扩展时设计失败。
- 执行不现实的基准测试，并且在测试期间将达到服务固定配额。例如，以突发限制运行测试，但运行时间较长。
- 选择的设计在超过固定服务配额时无法扩展或修改。例如，SQS 有效负载大小为 256 KB。
- 没有设计和实施可观测性，无法监控在高流量事件期间可能面临风险的服务配额阈值并发出警报。

建立此最佳实践的好处：确认应用程序会在所有预计的服务负载水平下运行，不会出现中断或性能下降。

在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

与软服务配额或替换为更高容量单位的资源不同，无法更改 AWS 服务的固定配额。这意味着，在应用程序设计中使用所有这些类型的 AWS 服务时，必须评估是否存在潜在的硬容量限制。

服务配额控制台中显示了硬限制。如果列显示 ADJUSTABLE = No，表示服务存在硬限制。一些资源配置页中也显示了硬限制。像是 Lambda 就具有无法调整的特定硬限制。

例如，在设计要在 Lambda 函数中运行的 Python 应用程序时，应评估应用程序，确定 Lambda 是否有可能运行超过 15 分钟。如果代码可能运行超过此服务配额限制，则必须考虑替代技术或设计。如果在生产部署后达到此限制，则应用程序将遭受性能下降和中断，直到可以补救为止。与软配额不同，即使出现严重性为 1 的紧急事件，也无法更改这些限制。

应用程序部署到测试环境之后，应使用策略来查明是否会达到任何硬限制。引入测试计划中应包括压力测试、负载测试和混沌测试。

实施步骤

- 查看可在应用程序设计阶段使用的 AWS 服务的完整列表。
- 查看所有这些服务的软配额限制和硬配额限制。并非所有限制都会在服务配额控制台中显示。有些服务在备用位置描述了这些限制。
- 在设计应用程序时，检查工作负载的业务和技术驱动因素，例如业务成果、应用场景、相依系统、可用性目标和灾难恢复对象。让业务和技术驱动因素指导流程，以确定适合工作负载的分布式系统。
- 分析各个区域和账户的服务负载。服务的许多硬限制基于区域，但有些限制基于账户。
- 分析韧性架构，了解在分区故障和区域故障期间的资源使用情况。在使用“主动/主动”、“主动/被动 – 热”、“主动/被动 – 冷”和“主动/被动 – 指示灯”方法的多区域设计过程中，这些故障情况会导致使用率更高。这会形成达到硬限制的潜在应用场景。

资源

相关最佳实践：

- REL01-BP01 了解服务配额和约束
- REL01-BP02 跨多个账户和区域管理服务配额
- REL01-BP04 监控和管理配额

- REL01-BP05 自动管理配额
- REL01-BP06 确保当前配额与最大使用量之间存在足够的差距来应对失效转移
- REL03-BP01 选择如何划分工作负载
- REL10-BP01 将工作负载部署到多个位置
- REL11-BP01 监控工作负载的所有组件以检测故障
- REL11-BP03 自动修复所有层
- REL12-BP04 使用混沌工程测试韧性

相关文档：

- AWS Well-Archited Framework 的可靠性支柱：可用性
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks（参见“Service limits”小节）
- AWS Answers 上的 AWS Limit Monitor
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- 适用于 AWS 的配额监控程序
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS 数据解决方案
- 什么是持续集成？
- 什么是持续交付？
- APN 合作伙伴：可帮助进行配置管理的合作伙伴
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

- Actions, resources, and condition keys for Service Quotas

相关视频：

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

相关工具：

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP04 监控和管理配额

评估可能的使用情况，并适当提高配额，支持使用量按计划增长。

期望结果：部署了可进行管理和监控的主动和自动化系统。这些操作解决方案可确保接近达到配额使用阈值。根据请求的配额更改主动修复这些问题。

常见反模式：

- 没有配置监控来检查服务配额阈值。
- 没有为硬限制配置监控，即使这些值不能更改。

- 假定请求和确立软配额变化所需的时间是即时或短时间。
- 配置警报在快达到服务配额时发出警报，但没有关于如何对警报做出响应的流程。
- 只为 AWS 服务配额支持的服务配置警报，不监控其他 AWS 服务。
- 不考虑多区域韧性设计（如“主动/主动”、“主动/被动 – 热”、“主动/被动 – 冷”和“主动/被动 – 指示灯”方法）的配额管理。
- 不评测区域之间的配额差异。
- 不评测每个区域对特定配额提高请求的需求。
- 不利用模板进行多区域配额管理。

建立此最佳实践的好处：自动跟踪 AWS 服务配额，并根据这些配额监控使用情况，以便了解何时会达到配额限制。您还可以使用此监控数据帮助限制由于配额耗尽而导致的性能下降。

在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

对于支持的服务，您可以配置各种能进行评测的不同服务，再通过发送提醒或警报来监控配额。这有助于监控使用情况，并可在接近配额时发出提醒。这些警报可以从 AWS Config、Lambda 函数、Amazon CloudWatch 或 AWS Trusted Advisor 调用。您还可以使用 CloudWatch Logs 上的指标筛选条件来搜索与提取日志中的模式，确定使用量是否快达到配额阈值。

实施步骤

监控：

- 获取当前资源使用情况（例如存储桶或实例）。使用 Amazon EC2 `DescribeInstances` API 等服务 API 操作来收集当前资源使用情况信息。
- 使用以下资源获得必要且适用于服务的当前配额：
  - AWS 服务限额
  - AWS Trusted Advisor
  - AWS 文档
  - AWS 服务特定页面
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- 使用 AWS 服务配额（一项 AWS 服务），帮助您从一个地方管理超过 250 项 AWS 服务的配额。

- 使用 Trusted Advisor 服务限制来监控在各种阈值下的当前服务限制。
- 使用服务配额历史记录（控制台或 AWS CLI）来检查区域增长情况。
- 如果需要，比较每个区域和每个账户中的服务配额变化，形成等效关系。

管理：

- 自动：设置 AWS Config 自定义规则以扫描各个区域的服务配额，并比较它们之间的差异。
- 自动：设置计划好的 Lambda 函数以扫描各个区域的服务配额，并比较它们之间的差异。
- 手动：通过 AWS CLI、API 或 AWS 控制台来扫描各个区域的服务配额，并比较它们之间的差异。报告差异。
- 如果在不同区域之间发现配额差异，则根据需要请求更改配额。
- 检查所有请求的结果。

资源

相关最佳实践：

- REL01-BP01 了解服务配额和约束
- REL01-BP02 跨多个账户和区域管理服务配额
- REL01-BP03 通过架构适应固定服务配额和约束
- REL01-BP05 自动管理配额
- REL01-BP06 确保当前配额与最大使用量之间存在足够的差距来应对失效转移
- REL03-BP01 选择如何划分工作负载
- REL10-BP01 将工作负载部署到多个位置
- REL11-BP01 监控工作负载的所有组件以检测故障
- REL11-BP03 自动修复所有层
- REL12-BP04 使用混沌工程测试韧性

相关文档：

- AWS Well-Archited Framework 的可靠性支柱：可用性
- AWS Service Quotas (formerly referred to as service limits)

- AWS Trusted Advisor Best Practice Checks（参见“Service limits”小节）
- AWS Answers 上的 AWS Limit Monitor
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- 适用于 AWS 的配额监控程序
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS 数据解决方案
- 什么是持续集成？
- 什么是持续交付？
- APN 合作伙伴：可帮助进行配置管理的合作伙伴
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

相关视频：

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

相关工具：

- AWS CodeDeploy

- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP05 自动管理配额

服务配额（在 AWS 服务中也称为限制）是您的 AWS 账户中资源的最大值。每项 AWS 服务都定义了一组配额及其默认值。为了让工作负载能够访问它所需的所有资源，您可能需要增加服务配额值。

如果超过配额，AWS 资源的工作负载消耗的增长可能会威胁工作负载的稳定性，并影响用户体验。实施相应的工具，以便在工作负载接近限制时发出警报，并考虑自动创建增加配额的请求。

期望结果：为在每个 AWS 账户和区域中运行的工作负载适当配置了配额。

常见反模式：

- 您未能适当考虑和调整配额来满足工作负载要求。
- 您使用可能过时的方法（例如电子表格）来跟踪配额和使用情况。
- 您只按定期计划更新服务限制。
- 您的组织缺乏操作流程，无法查看现有配额并在必要时请求增加服务配额。

建立此最佳实践的好处：

- 增强了工作负载韧性：您可以防止因超出 AWS 资源配额而导致的错误。
- 简化了灾难恢复：在另一个 AWS 区域中进行灾难恢复设置时，您可以重用在主区域中构建的自动配额管理机制。

在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

通过 AWS 服务配额控制台、AWS Command Line Interface（AWS CLI）和 AWS 等机制，查看当前配额并跟踪正在进行的配额消耗。还可以将配置管理数据库（CMDB）和 IT 服务管理（ITSM）系统与 AWS 服务配额 API 集成。

如果配额使用量达到您定义的阈值，则自动生成警报，并定义在收到警报时提交配额增加请求的过程。如果底层工作负载对您的业务至关重要，则可以自动提出配额增加请求，但要仔细测试此项自动功能，以避免出现失控操作的风险，例如增长反馈循环。

较小的配额增加通常会自动获得批准。较大的配额请求可能需要由 AWS 支持人员手动处理，可能需要更多时间来审核和处理。留出额外的时间来处理多个请求或大幅增加配额的请求。

实施步骤

- 对服务配额实施自动监控，并在工作负载的资源利用率增长接近配额限制时发出警报。例如，AWS 的 Quota Monitor 可以提供对服务配额的自动监控。此工具与 AWS Organizations 集成，并使用 Cloudformation StackSets 进行部署，以便在创建新账户时自动进行监控。
- 使用 Service Quotas request templates 或 AWS Control Tower 等功能简化新账户的服务配额设置。
- 构建控制面板来显示您当前在所有 AWS 账户和区域的服务配额使用情况，并在必要时参考这些控制面板来防止超过配额。Trusted Advisor Organizational (TAO) Dashboard 是 Cloud Intelligence Dashboards 的一部分，可让您快速开始使用此类控制面板。
- 跟踪服务限制提高请求。Consolidated Insights from Multiple Accounts(CIMA) 可以提供所有请求的组织级视图。
- 通过在非生产账户中设置较低的配额阈值，测试警报生成和任何自动发出配额增加请求的功能。请勿在生产账户中进行这些测试。

资源

相关最佳实践：

- OPS10-BP07 自动响应事件

相关文档：

- APN 合作伙伴：可帮助进行配置管理的合作伙伴
- AWS Marketplace：可帮助跟踪限制的 CMDB 产品
- AWS Service Quotas (formerly referred to as service limits)

- AWS Trusted Advisor Best Practice Checks（参见“Service limits”小节）
- 适用于 AWS 的配额监控程序解决方案 - AWS 解决方案
- What is Service Quotas?
- What is Service Quotas request templates?

相关视频：

- AWS Live re:Inforce 2019 - Service Quotas
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

相关工具：

- Quota Monitor for AWS

REL01-BP06 确保当前配额与最大使用量之间存在足够的差距来应对失效转移

本文介绍了如何在资源配额和使用量之间保留一定空间，以及这如何让组织受益。使用完资源后，该资源的使用量配额可能会继续计入该资源。这可能导致资源出现故障或无法访问。确认配额涵盖无法访问的资源及其替换资源的重叠部分，避免资源出现故障。在计算此差距时，应考虑网络故障、可用区故障或区域故障等应用场景。

期望结果：在当前服务阈值内可以覆盖资源或资源可访问性方面的或大或小的故障。在资源规划中已考虑到可用区故障、网络故障或甚至是区域故障。

常见反模式：

- 根据当前需求设置服务配额，而不考虑失效转移场景。
- 在计算服务的峰值配额时不考虑静态稳定性原则。
- 在计算每个区域所需的总配额时，不考虑可能无法访问资源的情况。
- 不考虑某些服务的 AWS 服务故障隔离边界及其可能的异常使用模式。

建立此最佳实践的好处：当服务中断事件影响应用程序可用性时，使用云实施相关策略，以便从这些事件中恢复。此类策略通常包括创建额外资源来替换无法访问的资源，在不耗尽服务限制的情况下应对失效转移条件。

在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

在评估配额限制时，考虑因性能下降可能导致的失效转移情况。考虑以下失效转移情况：

- VPC 中断或无法访问。
- 子网无法访问。
- 可用区性能下降，影响资源的可访问性。
- 系统阻止或更改了网络路由或入口点和出口点。
- 区域性能下降，影响资源的可访问性。
- 区域或可用区出现故障，影响资源子集。

因为业务影响可能会有很大差异，每种情况的失效转移决策都是独特的。在决定对应用程序或服务进行失效转移之前，请先考虑失效转移位置的资源容量规划和资源配额。

在检查每项服务的配额时，要考虑到高于正常水平的活动峰值。这些峰值可能与因网络或权限而无法访问但仍处于活动状态的资源有关。未终止的活动资源将计入服务配额限制。

实施步骤

- 在服务配额与最高使用量之间保留一定空间，以便应对失效转移或失去可访问性。
- 确定服务配额。要考虑典型的部署模式、可用性要求和使用量增长情况。
- 根据需要请求提高配额。预计配额提高请求需要等待的一段时间。
- 确定可靠性要求（也称为“X 个 9”）。
- 了解潜在的故障情况，例如组件、可用区或区域缺失。
- 确定部署方法（例如金丝雀部署、蓝绿部署、红黑部署或滚动部署）。
- 在当前配额限制中包含适当的缓冲区。例如设置 15% 的缓冲区。
- 在适当情况下包括静态稳定性（可用区和区域）的计算。
- 预计使用量增长情况并监控使用量趋势。
- 考虑静态稳定性对最关键工作负载的影响。评测所有区域和可用区中适应静态稳定系统的资源。
- 考虑使用按需容量预留，以便在发生任何失效转移之前安排容量。对于关键的业务计划来说，这是一种值得实施的有用策略，可以在失效转移期间获得正确数量和类型的资源，从而降低潜在的风险。

资源

相关最佳实践：

- REL01-BP01 了解服务配额和约束
- REL01-BP02 跨多个账户和区域管理服务配额
- REL01-BP03 通过架构适应固定服务配额和约束
- REL01-BP04 监控和管理配额
- REL01-BP05 自动管理配额
- REL03-BP01 选择如何划分工作负载
- REL10-BP01 将工作负载部署到多个位置
- REL11-BP01 监控工作负载的所有组件以检测故障
- REL11-BP03 自动修复所有层
- REL12-BP04 使用混沌工程测试韧性

相关文档：

- AWS Well-Archited Framework 的可靠性支柱：可用性
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks（参见“Service limits”小节）
- AWS Answers 上的 AWS Limit Monitor
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- 适用于 AWS 的配额监控程序
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS 数据解决方案
- 什么是持续集成？
- 什么是持续交付？
- APN 合作伙伴：可帮助进行配置管理的合作伙伴

- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

相关视频：

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

相关工具：

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL 2. 您如何规划网络拓扑？

工作负载通常存在于多个环境中。其中包括多个云环境（可公开访问和私有），可能还包括您现有的数据中心基础架构。相关计划必须涵盖网络注意事项，如系统内部和系统间连接、公有 IP 地址管理、私有 IP 地址管理以及域名解析。

最佳实践

- REL02-BP01 为工作负载公共端点使用高度可用的网络连接
- REL02-BP02 为云环境和本地环境之间的私有网络预置冗余连接
- REL02-BP03 确保 IP 子网分配考虑扩展和可用性
- REL02-BP04 轴辐式拓扑优先于多对多网格
- REL02-BP05 在互相连接的所有私有地址空间中强制实施非重叠的私有 IP 地址范围

REL02-BP01 为工作负载公共端点使用高度可用的网络连接

构建与工作负载的公共端点的高可用性网络连接有助于减少因连接丢失而导致的停机，并提高工作负载的可用性和改进 SLA。为此，需使用高度可用的 DNS、内容分发网络（CDN）、API Gateway、负载均衡或反向代理。

期望结果：为公共端点规划、构建和实施高度可用的网络连接至关重要。如果工作负载因连接中断而无法访问，即使工作负载正在运行且可用，客户也会看到系统处于关闭状态。通过将工作负载的公共端点的高可用性和弹性网络连接与工作负载本身的弹性架构结合起来，您可以为客户提供最佳的可用性和服务级别。

AWS Global Accelerator、Amazon CloudFront、Amazon API Gateway、AWS Lambda 函数 URL、AWS AppSync API 和弹性负载均衡（ELB）都提供了高度可用的公有端点。Amazon Route 53 为域名解析提供了高度可用的 DNS 服务，可确认公共端点地址是否可以解析。

您还可以评估 AWS Marketplace 软件设备是否适用于负载均衡和代理。

常见反模式：

- 在没有规划 DNS 和网络连接以实现高可用性的情况下设计高可用性工作负载。
- 在各个实例或容器中使用公有互联网地址并使用 DNS 管理与它们的连接。
- 使用 IP 地址而非域名来查找服务。
- 没有测试与公共端点的连接丢失的场景。
- 没有分析网络吞吐量需求和分发模式。
- 没有测试和规划与工作负载公共端点的互联网连接可能中断的情况。
- 为较大地理区域提供内容（如网页、静态资产或媒体文件），而不使用内容分发网络。
- 没有为分布式拒绝服务（DDoS）攻击制定计划。DDoS 攻击会引发关闭您的用户的合法流量并降低可用性的风险。

建立此最佳实践的好处：设计具有韧性的高可用性网络连接，确保用户可以访问和使用工作负载。

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

构建与公共端点的高可用性网络连接的核心是流量的路由。为了验证您的流量是否能够到达端点，DNS 必须能够将域名解析为其相应的 IP 地址。使用 Amazon Route 53 等高度可用且可扩展的域名系统（DNS）来管理域名的 DNS 记录。您还可以使用 Amazon Route 53 提供的运行状况检查。运行状况检查会确认应用程序可访问、可用且正常运行，并且支持以模仿用户行为的方式进行设置，例如请求网页或特定 URL。如果发生故障，Amazon Route 53 会响应 DNS 解析请求，并仅将流量定向到运行状况良好的端点。您还可以考虑使用 Amazon Route 53 提供的 Geo DNS 和基于延迟的路由功能。

要验证工作负载本身是否具有高可用性，请使用弹性负载均衡（ELB）。Amazon Route 53 可用于将流量定向到 ELB，而 ELB 会将流量分配到目标计算实例。您还可以将 Amazon API Gateway 与 AWS Lambda 结合使用，实现无服务器解决方案。客户也可以在多个 AWS 区域 中运行工作负载。借助多站点主动/主动模式，工作负载可以处理来自多个区域的流量。在多站点主动/被动模式下，工作负载为来自主动区域的流量提供服务，同时将数据复制到辅助区域，并在主区域出现故障时变为主动区域。然后，可以使用 Route 53 运行状况检查来控制从主区域中的任何端点到辅助区域的端点的 DNS 故障转移，从而验证工作负载是否可以访问并可供用户使用。

Amazon CloudFront 提供一个简单的 API，通过使用世界各地的边缘站点网络提供请求，从而分发具有低延迟和高数据传输速率的内容。内容分发网络（CDN）提供位于或缓存在用户附近位置的内容，从而为客户提供服务。随着内容负载从服务器转移到 CloudFront 的边缘站点，这也提高了应用程序的可用性。边缘站点和区域边缘高速缓存在靠近查看者的位置保存内容的缓存副本，以便可以快速检索并提高工作负载的可访问性和可用性。

对于用户在地理上分散的工作负载，AWS Global Accelerator 可帮助您提高应用程序的可用性和性能。AWS Global Accelerator 提供任播静态 IP 地址，它充当在一个或多个 AWS 区域 中托管的应用程序的固定入口点。这样就可以让流量在尽可能靠近用户的位置进入 AWS 全球网络，从而提高工作负载的可访问性和可用性。AWS Global Accelerator 还使用 TCP、HTTP 和 HTTPS 运行状况检查来监控应用程序端点的运行状况。端点的运行状况或配置发生任何更改，都会允许用户流量重定向到运行状况良好的端点，为用户提供最佳性能和可用性。此外，AWS Global Accelerator 采用故障隔离设计，使用由独立网络区域提供的两个静态 IPv4 地址，提高了应用程序的可用性。

为了帮助保护客户免受 DDoS 攻击，AWS 提供 AWS Shield Standard。Shield Standard 会自动开启，可防范 SYN/UDP 泛洪和反射攻击等常见基础设施（第 3 层和第 4 层）攻击，以便在 AWS 上支持应用程序的高可用性。要针对更复杂和更大规模的攻击（如 UDP 泛洪）、状态耗尽攻击（如 TCP SYN 泛洪）提供额外保护，以及要帮助保护 Amazon Elastic Compute Cloud（Amazon EC2）、弹性

负载均衡（ELB）、Amazon CloudFront、AWS Global Accelerator 和 Route 53 上运行的应用程序，可以考虑使用 AWS Shield Advanced。为了防范 HTTP POST 或 GET 泛洪等应用程序层攻击，请使用 AWS WAF。AWS WAF 可使用 IP 地址、HTTP 标头、HTTP 主体、URI 字符串、SQL 注入和跨站点脚本条件来确定是应该阻止还是允许请求。

实施步骤

1. 设置高度可用的 DNS：Amazon Route 53 是一种可用性高、可扩展性强的域名系统（DNS）Web 服务。Route 53 会将用户请求连接到在 AWS 或本地运行的互联网应用程序。有关更多信息，请参阅将 Amazon Route 53 配置为 DNS 服务。
2. 设置运行状况检查：当使用 Route 53 时，确认只有运行状况良好的目标才可解析。首先创建 Route 53 运行状况检查并配置 DNS 故障转移。在设置运行状况检查时，必须考虑以下方面：
   a. Amazon Route 53 如何确定运行状况检查是否正常
   b. 创建、更新和删除运行状况检查
   c. 监控运行状况检查状态和获取通知
   d. Amazon Route 53 DNS 的最佳实践
3. 将 DNS 服务连接到端点。
   a. 使用弹性负载均衡作为流量目标时，请使用 Amazon Route 53 创建指向负载均衡器区域端点的别名记录。在创建别名记录期间，将评估目标运行状况选项设置为“是”。
   b. 对于使用 API Gateway 时的无服务器工作负载或私有 API，请使用 Route 53 将流量路由到 API Gateway。
4. 决定内容分发网络。
   a. 要使用离用户更近的边缘站点分发内容，首先要了解 CloudFront 如何分发内容。
   b. 开始使用简单 CloudFront 分发。然后，CloudFront 知道您希望从何处分发内容，还知道有关如何跟踪和管理内容分发的详细信息。在设置 CloudFront 分发时，必须了解和考虑以下方面：
      i. 缓存如何用于 CloudFront 边缘站点
      ii. 增加直接从 CloudFront 缓存提供服务的请求的比例（缓存命中率）
      iii. 使用 Amazon CloudFront Origin Shield
      iv. 通过 CloudFront 源失效转移来优化高可用性
5. 设置应用程序层保护：AWS WAF 帮助您免受可能会影响可用性、危及安全性或消耗过多资源的常见 Web 漏洞和机器人的攻击。若要加深理解，请查看 How AWS WAF works；若准备好实施针对应用层 HTTP POST AND GET 泛洪的保护，请查看 Getting started with AWS WAF。要将 AWS WAF 与 CloudFront 结合使用，请参阅有关 How AWS WAF works with Amazon CloudFront features 的文档。

6. 设置额外的 DDoS 防护：默认情况下，所有 AWS 客户都可以免费使用 AWS Shield Standard 获得保护，防范针对您的网站或应用程序的常见、最常发生的网络和传输层 DDoS 攻击。要进一步保护在 Amazon EC2、弹性负载均衡、Amazon CloudFront、AWS Global Accelerator 和 Amazon Route 53 上运行的面向互联网的应用程序，可以考虑 AWS Shield Advanced 并查看 Examples of DDoS resilient architectures。要保护工作负载和公共端点免受 DDoS 攻击，请查看 Getting started with AWS Shield Advanced。

资源

相关最佳实践：

- REL10-BP01 将工作负载部署到多个位置
- REL11-BP04 恢复期间依赖于数据面板而不是控制面板
- REL11-BP06 当事件影响可用性时发送通知

相关文档：

- APN 合作伙伴：可帮助规划联网的合作伙伴
- AWS Marketplace for Network Infrastructure
- 什么是 AWS Global Accelerator？
- What is Amazon CloudFront?
- What is Amazon Route 53?
- 什么是 Elastic Load Balancing？
- Network Connectivity capability - Establishing Your Cloud Foundations
- 什么是 Amazon API Gateway？
- What are AWS WAF, AWS Shield, and AWS Firewall Manager?
- What is Amazon Application Recovery Controller?
- 配置针对 DNS 故障转移的自定义运行状况检查

相关视频：

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2020: Global traffic management with Amazon Route 53
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications

- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure
- AWS re:Invent 2022 - Building resilient networks

相关示例：

- Disaster Recovery with Amazon Application Recovery Controller (ARC)
- AWS Global Accelerator 讲习会

REL02-BP02 为云环境和本地环境之间的私有网络预置冗余连接

在云环境和本地环境中的专用网络之间实施冗余连接，以实现连接的韧性。这可以通过部署两条或更多链路和流量路径来实现，从而在网络出现故障时仍然保持连接。

常见反模式：

- 仅依赖一个网络连接，这会造成单点故障。
- 仅使用一个 VPN 隧道，或者使用多个隧道，但是多个隧道又连接到同一个可用区。
- 依赖一家互联网服务提供商（ISP）来提供 VPN 连接，这会导致在 ISP 中断期间彻底故障。
- 未实施像 BGP 这样的动态路由协议，这些协议对于在网络中断期间重新路由流量至关重要。
- 忽略了 VPN 隧道的带宽限制，高估了 VPN 隧道的备份能力。

建立此最佳实践的好处：通过在云环境和企业或本地环境之间实施冗余连接，两个环境之间的依赖服务就能够可靠通信。

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

使用 AWS Direct Connect 将您的本地网络连接到 AWS 时，如果使用不同的连接来连接到多个本地位置和多个 AWS Direct Connect 位置中的不同设备，则可以实现出色的网络韧性（SLA 为 99.99%）。这种拓扑结构可抵御设备故障、网络连接问题以及彻底的位置中断。或者，您可以通过使用两个单独的连接与多个位置相连（每个本地位置连接到一个 Direct Connect 位置）来实现较高的韧性（SLA 达到 99.9%）。这种方法可以防止因光纤中断或设备故障而导致的连接中断，并有助于缓解完全的位置故障。AWS Direct Connect 韧性工具包有助于您设计 AWS Direct Connect 拓扑。

您也可以考虑使用与 AWS Transit Gateway 相连的 AWS Site-to-Site VPN，以经济实惠的方式备份至主 AWS Direct Connect 连接。这种设置支持跨多个 VPN 隧道的等价多路径（ECMP）路由，即使每

个 VPN 隧道的吞吐量上限为 1.25 Gbps，也能实现高达 50 Gbps 的吞吐量。但值得注意的是，AWS Direct Connect 仍然是大幅减少网络中断并实现稳定连接的极佳选择。

在通过互联网使用 VPN 将您的云环境连接到本地数据中心时，将两个 VPN 隧道配置为单个 Site-to-Site VPN 连接的一部分。为了实现高可用性，每条隧道都应连接到不同的可用区，并使用冗余硬件来防止本地设备故障。此外，可以考虑使用不同互联网服务提供商（ISP）的多个互联网连接来连接到本地位置，避免因单个 ISP 中断而导致彻底中断 VPN 连接。选择具有不同路由和基础设施的 ISP，尤其是那些具有单独物理路径通往 AWS 端点的 ISP，可实现高连接可用性。

除了通过多个 AWS Direct Connect 连接和/或多个 VPN 隧道实现物理冗余外，实施边界网关协议（BGP）动态路由也至关重要。动态 BGP 可根据实时网络状况和配置的策略，自动将流量从一条路径重新路由到另一条路径。这种动态行为特别有助于在链路或网络出现故障时，保持网络可用性和服务连续性，进而快速选择替代的路径，增强网络的韧性和可靠性。

实施步骤

- 在 AWS 和本地环境之间获取高度可用的连接。
  - 在单独部署的专用网络之间使用多个 AWS Direct Connect 连接或 VPN 隧道。
  - 使用多个 AWS Direct Connect 位置来实现高可用性。
  - 如果使用多个 AWS 区域，请至少在其中两个区域中创建冗余。
- 如果可能，请使用 AWS Transit Gateway 终止 VPN 连接。
- 评估 AWS Marketplace 设备以终止 VPN 或将 SD-WAN 扩展到 AWS。如果您使用 AWS Marketplace 设备，请在不同的可用区中部署冗余实例以实现高可用性。
- 提供面向本地环境的冗余连接。
  - 您可能需要面向多个 AWS 区域 的冗余连接来满足可用性需求。
  - 使用 AWS Direct Connect 韧性工具包开始操作。

资源

相关文档：

- AWS Direct Connect Resiliency Recommendations
- Using Redundant Site-to-Site VPN Connections to Provide Failover
- Routing policies and BGP communities
- Active/Active and Active/Passive Configurations in AWS Direct Connect
- APN 合作伙伴：可帮助规划联网的合作伙伴

- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options 白皮书
- 构建可扩展的安全多 VPC AWS 网络基础设施
- Using redundant Site-to-Site VPN connections to provide failover
- Using the AWS Direct Connect Resiliency Toolkit to get started
- VPC 端点和 VPC 端点服务（AWS PrivateLink）
- 什么是 Amazon VPC？
- What is a transit gateway?
- 什么是 AWS Site-to-Site VPN ？
- Working with Direct Connect gateways

相关视频：

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs

REL02-BP03 确保 IP 子网分配考虑扩展和可用性

Amazon VPC IP 地址范围必须足够大，能够满足工作负载要求，包括考虑未来的扩展以及跨可用区为子网分配 IP 地址。这包括负载均衡器、EC2 实例和基于容器的应用程序。

当您规划网络拓扑时，第一步是定义 IP 地址空间本身。应（按照 RFC 1918 准则）为每个 VPC 分配私有 IP 地址范围。作为此流程的一部分，要满足以下要求：

- 在每个区域中为多个 VPC 留出 IP 地址空间。
- 在 VPC 内，为多个子网留出空间，这样您就可以跨多个可用区。
- 请考虑在 VPC 内保留未使用的 CIDR 块空间以用于未来扩展。
- 确保 IP 地址空间足以满足可能使用的任何 Amazon EC2 实例临时性队列的需求，如适用于机器学习的竞价型实例集、Amazon EMR 集群或 Amazon Redshift 集群。若是 Amazon Elastic Kubernetes Service（Amazon EKS）等 Kubernetes 集群，也应考虑这些因素。因为默认情况下，每个 Kubernetes 容器组（pod）都会从 VPC CIDR 块中分配一个可路由的地址。
- 注意，每个子网 CIDR 块中的前四个 IP 地址和最后一个 IP 地址将被预留，无法供您使用。
- 注意，最初被分配到您 VPC 的 VPC CIDR 块无法被更改或删除，但可以向 VPC 添加额外的非重叠的 CIDR 块。虽然无法更改子网 IPv4 CIDR，但可以更改 IPv6 CIDR。

- 可以使用的最大 VPC CIDR 块为 /16，最小为 /28。
- 考虑其他互联网络（VPC、本地部署或其他云提供商），并确保 IP 地址空间不重叠。有关更多信息，请参阅 REL02-BP05 在互相连接的所有私有地址空间中强制实施非重叠的私有 IP 地址范围。

期望结果：可扩展的 IP 子网有助于您适应未来的增长，并避免不必要的浪费。

常见反模式：

- 没有考虑未来的增长，导致 CIDR 块过小且需要重新配置，这可能会造成停机。
- 错误估计弹性负载均衡器可以使用的 IP 地址数量。
- 在相同子网中部署多个高流量负载均衡器。
- 使用自动扩缩机制，但未能监控 IP 地址使用情况。
- 定义过大的 CIDR 范围，远远超出未来的增长预期，这会导致地址范围重叠，难以与其他网络建立对等连接。

建立此最佳实践的好处：这可确保您能适应工作负载增长要求，并在纵向扩展过程中继续提供可用性。

在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

规划您的网络以适应增长、符合监管合规性以及实现与其他服务的集成。如果没有合理的规划，则增长可能会被低估、监管合规性可能会发生变化并且购置或私有网络连接可能难以实施。

- 根据您服务、延迟、法规和灾难恢复（DR）要求，选择相关 AWS 账户 和区域。
- 确定对区域 VPC 部署的需求。
- 确定 VPC 的大小。
  - 确定是否要部署多 VPC 连接。
    - What Is a Transit Gateway?
    - Single Region Multi-VPC Connectivity
  - 确定是否需要隔离网络来满足法规要求。
  - 使用适当大小的 CIDR 块创建 VPC，满足当前和未来的需求。
    - 如果增长预测不明朗，则可能需要偏向更大的 CIDR 块，降低未来重新配置的可能性
  - 在双堆栈 VPC 中，考虑对子网使用 IPv6 寻址。IPv6 非常适合用于包含大量临时实例集或临时容器集的私有子网，因此需要大量 IPv4 地址。

资源

相关的 Well-Architected 最佳实践：

- REL02-BP05 在互相连接的所有私有地址空间中强制实施非重叠的私有 IP 地址范围

相关文档：

- APN 合作伙伴：可帮助规划联网的合作伙伴
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options 白皮书
- 多数据中心 HA 网络连接
- Single Region Multi-VPC Connectivity
- 什么是 Amazon VPC？
- AWS 上的 IPv6
- IPv6 on reference architectures
- Amazon Elastic Kubernetes Service launches IPv6 support
- Recommendations for your VPC - Classic Load Balancers
- Availability Zone subnets - Application Load Balancers
- Availability Zones - Network Load Balancers

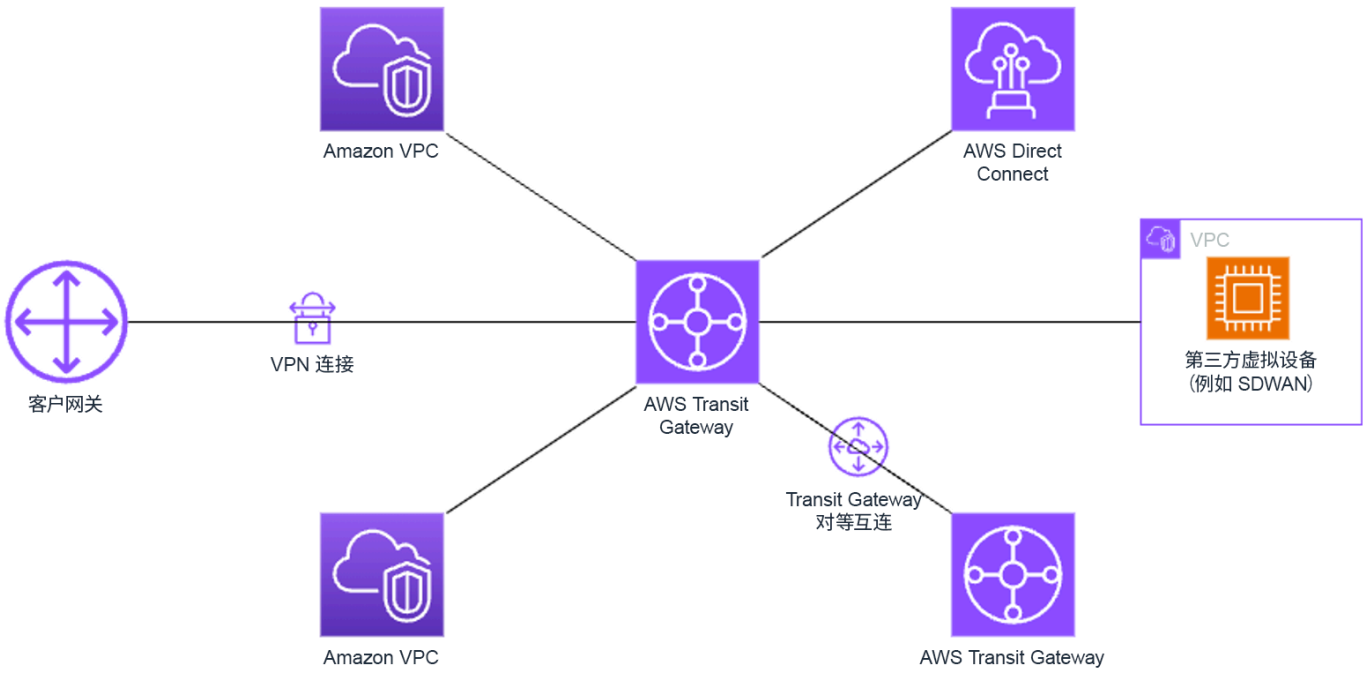
相关视频：

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)
- AWS re:Invent 2023: AWS Ready for what's next? Designing networks for growth and flexibility (NET310)

REL02-BP04 轴辐式拓扑优先于多对多网格

连接多个私有网络时，例如连接虚拟私有云（VPC）与本地网络，应优先选择轴辐式拓扑而不是网格拓扑。在网格拓扑中，每个网络直接连接到其他网络，这会增加复杂性和管理开销，而轴辐式拓扑则不同，这种拓扑架构通过单个中心枢纽来集中连接。这种集中式方法简化了网络结构，并且可以增强可操作性、可扩展性和控制能力。

AWS Transit Gateway 是一项托管服务，可扩展且能够提供高可用性，专为在 AWS 上构建轴辐式网络而设计。该服务充当网络的中心枢纽，提供网络分段、集中路由功能，并可简化与云端以及与本地环境的连接。下图说明了如何使用 AWS Transit Gateway 来构建轴辐式拓扑。



期望结果：您已通过中心枢纽连接虚拟私有云（VPC）和本地网络。可以通过此枢纽配置对等连接，该枢纽充当高度可扩展的云路由器。因为您不必处理复杂的对等关系，所以路由得以简化。网络之间的流量已加密，并且您可以隔离网络。

常见反模式：

- 您构建复杂的网络对等规则。
- 您在不应彼此通信的网络之间提供路由（例如，没有相互依赖关系的独立工作负载）。
- 枢纽实例的治理效率低下。

建立此最佳实践的好处：随着互联网络数量增加，网格连接的管理和扩展变得越来越有挑战性。网格架构会带来其它挑战，例如额外的基础设施组件、配置要求和部署注意事项。网格还为管理和监控数据面板和控制面板组件带来了额外的开销。您必须考虑如何提供网格架构的高可用性，如何监控网格运行状况和性能，以及如何处理网格组件的升级。

另一方面，轴辐式模型可在多个网络之间建立集中式流量路由。它提供了一种更简单的方法来管理和监控数据面板和控制面板组件。

在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

如果网络服务账户不存在，请创建一个此类账户。将枢纽置于组织的网络服务账户中。这种方法可让网络工程师对枢纽进行集中管理。

轴辐式模型的枢纽充当虚拟路由器，用于路由在虚拟私有云（VPC）和本地网络之间流动的流量。这种方法降低了网络复杂性，并且可以更轻松地对网络问题进行故障排除。

考虑您的网络设计，包括要互连的 VPC、AWS Direct Connect 和 Site-to-Site VPN 连接。

考虑为每个中转网关 VPC 连接使用单独的子网。对于每个子网，请使用小型 CIDR（例如 /28），以便您有更多地址空间用于计算资源。此外，创建一个网络 ACL，并将其与已和枢纽关联的所有子网相关联。确保网络 ACL 在入站和出站方向打开。

设计并实现路由表，以便仅在应进行通信的网络之间提供路由。忽略不应彼此通信的网络之间的路由（例如，在没有相互依赖关系的不同工作负载之间）。

实施步骤

1. 规划网络。确定要连接的网络，并确认它们不共享重叠的 CIDR 范围。
2. 创建 AWS Transit Gateway 并连接您的 VPC。
3. 根据需要，创建 VPN 连接或 Direct Connect 网关，并将其与 Transit Gateway 关联。
4. 通过配置 Transit Gateway 路由表，定义如何在连接的 VPC 和其他连接之间路由流量。
5. 使用 Amazon CloudWatch 进行监控并根据需要调整配置，从而优化性能和成本。

资源

相关最佳实践：

- REL02-BP03 确保 IP 子网分配考虑扩展和可用性
- REL02-BP05 在互相连接的所有私有地址空间中强制实施非重叠的私有 IP 地址范围

相关文档：

- What Is a Transit Gateway?
- 中转网关设计最佳实践
- 构建可扩展的安全多 VPC AWS 网络基础设施

- Building a global network using AWS Transit Gateway Inter-Region peering
- Amazon Virtual Private Cloud Connectivity Options
- APN 合作伙伴：可帮助规划联网的合作伙伴
- AWS Marketplace for Network Infrastructure

相关视频：

- AWS re:Invent 2023 – AWS networking foundations
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities

相关讲习会：

- AWS Transit Gateway Workshop

REL02-BP05 在互相连接的所有私有地址空间中强制实施非重叠的私有 IP 地址范围

当多个 VPC 对等连接、通过 Transit Gateway 连接或者通过 VPN 连接时，各个 VPC 的 IP 地址范围不得重叠。避免 VPC 与本地环境之间或者与所使用的其他云提供商之间出现 IP 地址冲突。您还必须能够在需要时分配私有 IP 地址范围。IP 地址管理（IPAM）系统有助于实现这一操作的自动化。

期望结果：

- VPC、本地环境或其他云提供商之间不存在 IP 地址范围冲突。
- 适当的 IP 地址管理支持更轻松地扩展网络基础设施来适应不断增长和变化的网络要求。

常见反模式：

- 在 VPC 中使用与本地、企业网络或者其他云提供商相同的 IP 范围。
- 不追踪用于部署工作负载的 VPC 的 IP 范围。
- 依赖手动 IP 地址管理流程，例如电子表格。
- CIDR 块过大或过小，这往往会导致 IP 地址浪费或地址空间不足以容纳您的工作负载。

建立此最佳实践的好处：主动规划网络可确保您不会遇到互连网络中多次出现相同 IP 地址的情况。这可防止使用不同应用程序的工作负载部分出现路由问题。

在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 IPAM（例如 Amazon VPC IP 地址管理器）来监控并管理 CIDR 使用情况。AWS Marketplace 也提供了几个 IPAM。评估您在 AWS 上的可能使用量、将 CIDR 范围添加到现有 VPC，并且在创建 VPC 时要考虑到计划的使用量增长情况。

实施步骤

- 捕获当前的 CIDR 使用量数据（例如，VPC 和子网）。
  - 使用服务 API 操作收集当前的 CIDR 使用量数据。
  - 使用 Amazon VPC IP 地址管理器来发现资源。
- 捕获当前的子网使用量数据。
  - 使用服务 API 操作在每个区域中按 VPC 收集子网。
  - 使用 Amazon VPC IP 地址管理器来发现资源。
- 记录当前使用量数据。
- 确定是否创建了任何重叠的 IP 范围。
- 计算备用容量。
- 确定重叠的 IP 范围。您可以迁移到新的地址范围，也可以考虑在需要连接重叠范围时使用私有 NAT 网关或 AWS PrivateLink 等技术。

资源

相关最佳实践：

- 保护网络

相关文档：

- APN 合作伙伴：可帮助规划联网的合作伙伴
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options 白皮书
- 多数据中心 HA 网络连接
- Connecting Networks with Overlapping IP Ranges
- 什么是 Amazon VPC？
- 什么是 IPAM？

相关视频：

- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs
- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2021 – {New Launch} Manage your IP addresses at scale on AWS

工作负载架构

问题

- REL 3. 如何设计工作负载服务架构？
- REL 4. 如何在分布式系统中设计交互以预防发生故障？
- REL 5. 如何在分布式系统中设计交互以缓解或承受故障？

REL 3. 如何设计工作负载服务架构？

使用服务导向型架构（SOA）或微服务架构构建高度可扩展的可靠工作负载。服务导向型架构（SOA）可通过服务接口使软件组件可重复使用。微服务架构则进一步让组件变得更小、更简单。

最佳实践

- REL03-BP01 选择如何划分工作负载
- REL03-BP02 构建专注于特定业务领域和功能的服务
- REL03-BP03 根据 API 提供服务合同

REL03-BP01 选择如何划分工作负载

在确定应用程序的弹性要求时，工作负载划分很重要。应尽量避免使用整体架构，仔细考虑哪些应用程序组件可以分解为多项微服务。应用程序具体需求各异，架构到最后往往会将服务导向型架构（SOA）与微服务架构两者结合起来。能够实现无状态的工作负载更容易部署为微服务。

期望结果：工作负载应该可支持、可扩展，并且尽量做到松耦合。

在选择如何划分工作负载时，要权衡其优点和复杂性。适用于即将首次发布的新产品的功能，有别于从一开始就构建用于扩展的工作负载的需求。重构一个现有的整体架构时，您需要考虑应用程序对无状态分解的支持程度。通过将服务分解为较小的部分，可以让职责明确的小型团队来开发和管理它们。然而，较小的服务会带来复杂性，包括可能会增加延迟，调试变得更复杂，而且加重运营负担。

常见反模式：

- 微服务 Death Star 是这样一种情况：原子组件变得高度相互依赖，牵一发而动全身，使组件像一块整体一样死板而又脆弱。

建立此最佳实践的好处：

- 更多特定分段可以提高敏捷性、组织灵活性和可扩展性。
- 减小了服务中断的影响。
- 应用程序组件可能有不同的可用性要求，可通过更加原子化的分段来满足这些要求。
- 支持工作负载的团队职责分明。

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

请根据工作负载的分段方式选择架构类型。选择 SOA 或微服务架构（极少数情况下是整体架构）。即便在刚开始选择整体架构，您必须确保它是模块化的，而且由于您的产品随着采用的用户增加而扩展，它最终也可转变成为 SOA 或微服务架构。SOA 和微服务各自提供较小的区段，它们是现代可扩展和可靠架构的首选，但您需要认真权衡利弊，尤其在部署微服务架构时。

一项主要的权衡是您现在使用的是分布式计算架构，可能更难实现用户延迟要求，还增加了调试和跟踪用户交互的复杂性。您可以使用 AWS X-Ray 来帮助解决此问题。需要考虑的另一个影响是，您管理的应用程序数量增加时，运营复杂性也会随之增加，这要求部署多个相互独立组件。

整体架构、服务导向型架构和微服务架构

实施步骤

- 确定构建或重构应用程序所需的适当架构。SOA 和微服务分别提供较小分段，这是现代可扩展的可靠架构的首选。要在实现较小分段的同时避免一些微服务复杂性，SOA 是很好的折中方案。有关更多详细信息，请参阅 Microservice Trade-Offs。
- 如果工作负载适合，并且组织可以支持，则应使用微服务架构来实现最佳敏捷性和可靠性。有关更多详细信息，请参阅在 AWS 上实施微服务。
- 考虑按照 Strangler Fig 模式将整体重构为较小的组件。这包括使用新的应用程序和服务逐步替换特定的应用程序组件。AWS Migration Hub Refactor Spaces 充当增量重构的起点。有关更多详细信息，请参阅 Seamlessly migrate on-premises legacy workloads using a strangler pattern。
- 实施微服务可能需要服务发现机制，让这些分布式服务间能够相互通信。AWS App Mesh 可用于服务导向型架构，提供对服务的可靠发现和访问。AWS Cloud Map 也可以用于基于 DNS 的动态服务发现。
- 如果要从整体架构迁移到 SOA，Amazon MQ 可以在重新设计云中的传统应用程序时作为服务总线帮助缩小差距。
- 对于具有单个共享数据库的现有整体架构，请选择将数据重组为较小分段的方式。可以按业务部门、访问模式或数据结构来划分。在重构过程的这一阶段，应选择是使用关系型还是非关系型（NoSQL）数据库来继续操作。有关更多详细信息，请参阅从 SQL 到 NoSQL。

实施计划的工作量级别：高

资源

相关最佳实践：

- REL03-BP02 构建专注于特定业务领域和功能的服务

相关文档：

- Amazon API Gateway：使用 OpenAPI 配置 REST API
- 什么是服务导向型架构？
- Bounded Context (a central pattern in Domain-Driven Design)
- 在 AWS 上实施微服务
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- AWS 上的微服务

- 什么是 AWS App Mesh ？

相关示例：

- Iterative App Modernization 讲习会

相关视频：

- Delivering Excellence with Microservices on AWS

REL03-BP02 构建专注于特定业务领域和功能的服务

服务导向型架构（SOA）采用按业务需求定义的、划分明确的功能来定义服务。微服务使用域模型和限界上下文，沿业务环境边界划定服务边界。通过将重点放在业务领域和功能上，有助于团队为其服务定义独立的可靠性要求。限界上下文隔离和封装业务逻辑，让团队能够更好地解释如何处理故障。

期望结果：工程师和业务利益相关方共同定义限界上下文，并使用它们将系统设计为实现特定业务功能的服务。这些团队使用事件风暴等既定的实践来定义需求。新的应用程序被设计为服务，具有明确定义的边界并采用松耦合。现有整体式架构被分解为限界上下文，而系统设计则朝着服务导向型架构或微服务架构转变。重构整体式架构时，会应用气泡上下文和整体式架构分解模式等既定方法。

面向领域的服务作为一个或多个进程执行，彼此之间不分享状态。这些进程独立应对需求的波动，并根据特定领域的要求处理故障情景。

常见反模式：

- 围绕特定技术领域（例如 UI 和 UX、中间件或数据库）组建团队，而不是根据特定的业务领域组建。
- 应用程序进行了领域职责划分。跨越限界上下文的服务可能更难于维护，需要更多测试工作，并且需要多个领域团队参与软件更新。
- 领域依赖关系（例如领域实体库）在服务之间共享，因此对一个服务领域进行更改也需要更改其他服务领域
- 服务合同和业务逻辑没有使用通用且一致的领域语言来描述实体，这会导致翻译层的存在，使得系统更加复杂并增加调试工作量。

建立此最佳实践的好处：应用程序被设计为独立的服务，按照业务领域确定界限，并使用通用的业务语言。服务可独立测试和部署。对于所实施的领域，服务满足领域特定的韧性要求。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

领域驱动型设计（DDD）是围绕业务领域设计和构建软件的基本方法。围绕业务领域构建服务时，使用现有框架会很有帮助。在使用现有的整体式应用程序时，您可以利用分解模式提供成熟的技术，将应用程序改造为现代化服务。

领域驱动型设计

实施步骤

- 团队可以举办事件风暴讲习会，以简便的便签格式，快速确定事件、命令、聚合和领域。
- 在领域上下文中形成领域实体和功能后，您可以使用限界上下文将领域划分为服务，将具有相似特征和属性的实体分成一组。通过将模型细分为不同的上下文，即可得到如何确定微服务边界的模板。
  - 以 Amazon.com 网站为例，实体可能包括包装、配送、时间表、价格、折扣和货币。
  - 包装、配送和时间表分组到运输上下文中，而价格、折扣和货币分组到定价上下文中。
- Decomposing monoliths into microservices 概述了重构微服务的模式。使用按照业务功能、子领域或事务进行分解的模式，与领域驱动型方法非常吻合。
- 利用气泡上下文等战术性技巧，您可以在现有或旧版应用程序中引入 DDD，无需预先重新编写和承诺完全转向 DDD。在气泡上下文方法中，使用服务映射和协调来建立小型限界上下文，或建立防损层来保护新定义的领域模型免受外部影响。

在团队进行了领域分析并定义实体和服务合同后，他们可以利用 AWS 服务，将自己的领域驱动型设计作为云端服务来实施。

- 通过定义执行领域业务规则的测试来开始开发。测试驱动型开发（TDD）和行为驱动型开发（BDD），有助于团队将服务重点放在解决业务问题上。
- 选择最符合业务领域要求和微服务架构的 AWS 服务：
  - AWS 上的无服务器服务让团队可以将精力集中于具体的领域逻辑上，而不是管理服务器和基础设施。

  - AWS 上的容器可简化基础设施的管理，让您可以将精力集中于领域要求上。
  - 专用数据库有助于将领域要求与最适合的数据库类型相匹配。
- Building hexagonal architectures on AWS 中概述了一个框架，即从业务领域出发，采用逆向工作方法，将业务逻辑构建到服务中来满足功能要求，然后连接集成适配器。采用将接口详细信息从业务逻辑与 AWS 服务之间分离的模式，让团队能够专注于研究领域功能并提高软件质量。

资源

相关最佳实践：

- REL03-BP01 选择如何划分工作负载
- REL03-BP03 根据 API 提供服务合同

相关文档：

- AWS 微服务
- 在 AWS 上实施微服务
- How to break a Monolith into Microservices
- Getting Started with DDD when Surrounded by Legacy Systems
- Domain-Driven Design: Tackling Complexity in the Heart of Software
- Building hexagonal architectures on AWS
- Decomposing monoliths into microservices
- Event Storming
- Messages Between Bounded Contexts
- Microservices
- Test-driven development
- Behavior-driven development

相关示例：

- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop)

相关工具：

- AWS Cloud 数据库
- AWS 上的无服务器
- AWS 上的容器

REL03-BP03 根据 API 提供服务合同

服务合同是 API 生产者与使用者之间的书面协议，采用机器可读的 API 定义形式进行定义。合同版本控制策略让使用者能够继续使用现有的 API，并在更新的 API 准备就绪时，将其应用程序迁移到更新的 API。只要遵守合同，生产者部署可随时进行。服务团队可以使用自己选择的技术堆栈来满足 API 合同要求。

期望结果：使用服务导向型架构或微服务架构所构建的应用程序能够独立运行，但具有集成的运行时系统依赖项。当双方都遵循共同的 API 合同时，向 API 使用者或生产者部署更改并不会影响整个系统的稳定性。通过服务 API 进行通信的组件能够独立执行功能发布、升级到运行时系统依赖项或者失效转移到灾难恢复（DR）站点，而彼此之间的影响很小，或者根本没有影响。此外，离散服务能够独立扩展来满足资源需求，无需统一扩展其他服务。

常见反模式：

- 创建不使用强类型架构的服务 API。这样，API 不能用来生成无法通过编程方式验证的 API 绑定和有效负载。
- 不采用版本控制策略，会迫使 API 使用者在服务合同变化时进行更新和发布，否则会出现故障。
- 错误消息会泄露基础服务的实施细节，而不是按照域上下文和语言描述集成故障。
- 不使用 API 合同开发测试用例和模拟 API 实施，以便对服务组件进行独立测试。

建立此最佳实践的好处：分布式系统由通过 API 服务合同进行通信的组件组成，可以提高可靠性。开发人员可以在开发过程的早期发现潜在问题，在编译期间进行类型检查，来验证请求和响应是否符合 API 合同以及是否存在必填字段。API 合同为 API 提供了清晰的自描述接口，并在不同的系统和编程语言之间提供了更好的互操作性。

在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

确定业务领域并确定工作负载划分后，即可开发服务 API。首先，为 API 定义机器可读的服务合同，然后实施 API 版本控制策略。在准备好通过 REST、GraphQL 等常见协议或异步事件来集成服务时，您可以将 AWS 服务整合到架构中，从而将组件与强类型的 API 合同集成。

面向服务 API 合同的 AWS 服务

将包括 Amazon API Gateway、AWS AppSync 和 Amazon EventBridge 在内的 AWS 服务整合到架构中，以便在应用程序中使用 API 服务合同。Amazon API Gateway 有助于直接与原生 AWS 服务和其他网络服务集成。API Gateway 支持 OpenAPI 规范和版本控制。AWS AppSync 属于 GraphQL 托管端点，您可以通过定义 GraphQL 架构来配置该端点，定义用于查询、突变和订阅的服务接口。Amazon EventBridge 使用事件架构来定义事件，为事件生成代码绑定。

实施步骤

- 首先，为您的 API 定义一个合同。合同将说明 API 的功能，并为 API 的输入和输出定义强类型的数据对象和字段。
- 在 API Gateway 中配置 API 时，您可以导入和导出端点的 OpenAPI 规范。
  - 导入 OpenAPI 定义可简化 API 的创建过程，并可与 AWS 基础设施即代码工具（例如 AWS Serverless Application Model 和 AWS Cloud Development Kit (AWS CDK)）集成。
  - 导出 API 定义可简化与 API 测试工具的集成，并为服务使用者提供集成规范。
- 您可以通过定义 GraphQL 架构文件，使用 AWS AppSync 来定义和管理 GraphQL API，从而生成合同接口，并简化与复杂 REST 模型、多个数据库表或传统服务的交互。
- AWS Amplify 项目与 AWS AppSync 集成后，会生成强类型的 JavaScript 查询文件供应用程序使用，还会生成 AWS AppSync GraphQL 客户端库供 Amazon DynamoDB 表使用。
- 当您使用来自 Amazon EventBridge 的服务事件时，事件遵循架构注册表中已有的架构或您使用 OpenAPI 规范定义的架构。通过在注册表中定义架构，您还可以从架构合同生成客户端绑定，以将代码与事件集成。
- 扩展 API 或者实施 API 版本控制。在添加可以配置为可选字段的字段时，或者为必填字段添加默认值时，扩展 API 是一种相对比较简单的选项。
  - 对于 REST 和 GraphQL 等协议，基于 JSON 的合同可能非常适合合同扩展。
  - 对于 SOAP 等协议，基于 XML 的合同应与服务使用者一起进行测试，来确定合同扩展的可行性。
- 对 API 进行版本控制时，可以考虑实施代理版本控制，其中使用 Facade 模式来支持版本，这样就能够在单个代码库中维护逻辑。
  - 借助 API Gateway，您可以使用请求和响应映射，通过建立 Facade 模式为新字段提供默认值，或者从请求或响应中除去已删除的字段，从而简化接受合同变更的过程。通过这种方法，底层服务可以维护单个代码库。

资源

相关最佳实践：

- REL03-BP01 选择如何划分工作负载
- REL03-BP02 构建专注于特定业务领域和功能的服务
- REL04-BP02 实施松耦合的依赖关系
- REL05-BP03 控制与限制重试调用
- REL05-BP05 设置客户端超时

相关文档：

- 什么是 API（应用程序编程接口）？
- 在 AWS 上实施微服务
- Microservice Trade-Offs
- Microservices – a definition of this new architectural term
- AWS 上的微服务
- 使用基于 OpenAPI 的 API Gateway 扩展
- OpenAPI-Specification
- GraphQL: Schemas and Types
- Amazon EventBridge code bindings

相关示例：

- Amazon API Gateway：使用 OpenAPI 配置 REST API
- Amazon API Gateway to Amazon DynamoDB CRUD application using OpenAPI
- Modern application integration patterns in a serverless age: API Gateway Service Integration
- Implementing header-based API Gateway versioning with Amazon CloudFront
- AWS AppSync: Building a client application

相关视频：

- Using OpenAPI in AWS SAM to manage API Gateway

相关工具：

- Amazon API Gateway

- AWS AppSync
- Amazon EventBridge

REL 4. 如何在分布式系统中设计交互以预防发生故障？

分布式系统依靠通信网络来互连组件，例如服务器或服务。即使这些网络中出现数据丢失或延迟情况，您的工作负载也必须可靠运行。分布式系统组件的运行方式不得对其他组件或工作负载产生负面影响。这些最佳实践可以防止故障并缩短平均故障间隔时间 (MTBF)。

最佳实践

- REL04-BP01 确定所依赖的分布式系统的类型
- REL04-BP02 实施松耦合的依赖关系
- REL04-BP03 持续工作
- REL04-BP04 使变异操作幂等

REL04-BP01 确定所依赖的分布式系统的类型

分布式系统可以是同步系统、异步系统或批处理系统。同步系统必须尽可能快地处理请求，并使用 HTTP/S、REST 或远程过程调用（RPC，Remote Procedure Call）协议，同步地发出请求和响应调用，来彼此通信。异步系统在彼此通信时通过中间服务来异步交换数据，无需将各个系统耦合在一起。批处理系统则会接收大量输入数据，无需人工干预即可运行自动数据处理，并生成输出数据。

期望结果：设计能够与同步、异步和批处理依赖项进行有效交互的工作负载。

常见反模式：

- 工作负载无限期地等待其依赖项的响应，这可能导致工作负载客户端超时，不知道其请求是否已被接收。
- 工作负载使用会同步调用彼此的依赖系统链。这种模式要求每个系统都可用并能成功处理请求，整个链才能成功运行，这导致很容易出现崩溃行为，影响到整体可用性。
- 工作负载与其依赖项异步通信，并依赖于保证消息传递且仅传递一次的概念，但仍然会经常收到重复的消息。
- 工作负载没有使用正确的批处理调度工具，导致允许并行执行相同的批处理作业。

建立此最佳实践的好处：对于给定的工作负载，通常能够在同步、异步和批处理之间实施一种或多种通信方式。此最佳实践可帮助您确定，选择每种通信方式所面对的不同权衡，以便让您的工作负载能够承受其任意依赖项中断所带来的干扰。

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

以下各节针对每种依赖项，介绍了一般性实施指导和具体实施指导。

一般指导

- 确保您的依赖项提供的性能和可靠性服务级别目标（SLO，Service-Level Objective），能够满足工作负载的性能和可靠性要求。
- 使用 AWS 可观测性服务来监控响应时间和错误率，确保依赖项提供的服务达到工作负载所需的水平。
- 确定您的工作负载在与其依赖项进行通信时，可能会遇到的潜在挑战。分布式系统面临着诸多挑战，可能会增加架构的复杂性、运营负担和成本。常见的挑战包括延迟、网络中断、数据丢失、可扩展性和数据复制延迟。
- 实施强大的错误处理和日志记录，在依赖项遇到问题时帮助排查问题。

同步依赖项

在同步通信中，工作负载会向其依赖项发送请求并停止操作来等待响应。当其依赖项收到请求时，依赖项会尝试尽快处理请求并将响应发送回工作负载。同步通信面临的一个巨大挑战是它会导致时间耦合，这要求您的工作负载及其依赖项同时可用。当您的工作负载需要与其依赖项以同步方式通信时，请考虑以下指南：

- 工作负载在执行单个功能时，不应依赖多个同步依赖项。这种依赖项链会导致整体更加脆弱，因为要想完成请求，路径中的所有依赖项都必须可用。
- 请确定当依赖项运行状况不佳或不可用时，您采用的错误处理和重试策略。避免使用双模态行为。双模态行为是指工作负载在正常模式和故障模式下表现出不同的行为。有关双模态行为的更多详细信息，请参阅 REL11-BP05 使用静态稳定性来防止双模态行为。
- 请记住，快速失效机制比让工作负载等待要好。例如，《AWS Lambda 开发人员指南》描述了在调用 Lambda 函数时如何处理重试和失败。
- 设置工作负载调用其依赖项时的超时。这种技术可以避免等待太长时间或无限期等待响应。有关此主题的实用讨论，请参阅 Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications。

- 在完成单个请求时，尽可能减少从工作负载对依赖项进行的调用次数。在它们之间进行的频繁调用会增加耦合和延迟。

异步依赖项

要想临时将工作负载与其依赖项解耦，就应该采取异步通信。使用异步方法，您的工作负载无需等待其依赖项或依赖项链发送响应，即可继续进行任何其他处理。

当您的工作负载需要与其依赖项以异步方式通信时，请考虑以下指南：

- 根据应用场景和要求，确定是使用消息收发还是事件流。消息收发可让工作负载通过消息代理发送和接收消息，从而与其依赖项进行通信。事件流可让工作负载及其依赖项利用流服务来发布和订阅以连续数据流形式交付且需尽快处理的事件。

- 消息收发和事件流以不同的方法来处理消息，因此您需要根据以下因素作出权衡决策：
  - 消息优先级：消息代理可以先处理高优先级消息，再处理普通消息。在事件流中，所有消息具有相同的优先级。
  - 消息使用：消息代理确保使用者能够收到消息。事件流使用器必须跟踪所读取的每一条消息。
  - 消息排序：除非对消息收发使用先进先出（FIFO）方法，否则无法保证按照发送消息的确切顺序接收消息。事件流则始终保留数据生成的顺序。
  - 消息删除：使用消息收发时，使用者必须在处理消息后将其删除。事件流服务则将消息附加到流并一直保留在流中，直到消息的保留期到期。这种删除策略让事件流非常适合重放消息。
- 定义如何让工作负载在其依赖项完成工作时了解这一信息。例如，当工作负载异步调用 Lambda 函数时，Lambda 将请求置于队列中并返回成功响应，而不返回其他信息。处理完成后，Lambda 函数可以将结果发送到目标，该目标可根据成功或失败进行配置。
- 利用幂等性，构建工作负载以处理重复的消息。幂等性意味着，即使您的工作负载针对同一消息多次生成结果，这些结果也会保持不变。需要指出的是，如果发生网络故障或未收到确认，则消息收发或流服务将重新发送消息。
- 如果您的工作负载没有从其依赖项获得响应，则需要重新提交请求。请考虑限制重试次数，以保留工作负载的 CPU、内存和网络资源用于处理其他请求。AWS Lambda 文档介绍了如何处理异步调用错误。
- 利用合适的可观测性、调试和跟踪工具，来管理和操作工作负载与其依赖项的异步通信。您可以使用 Amazon CloudWatch 来监控消息收发和事件流服务。您还可以使用 AWS X-Ray 检测工作负载，快速获得洞察来排查问题。

批处理依赖项

批处理系统获取输入数据，启动一系列作业来处理数据，然后生成一些输出数据，整个过程无需人工干预。根据数据大小，作业的运行时间可能只需要几分钟，而在某些情况下，也可能长达数天。当您的工作负载与其批处理依赖项通信时，请考虑以下指南：

- 定义工作负载应运行批处理作业的时间窗口。工作负载可以设置重复模式来调用批处理系统，例如每小时或每月月底。
- 确定数据输入的位置，以及处理后数据输出的位置。选择一种能让工作负载大规模读取和写入文件的存储服务，例如 Amazon Simple Storage Service（Amazon S3）、Amazon Elastic File System（Amazon EFS）和适用于 Lustre 的 Amazon FSx。
- 如果工作负载需要调用多个批处理作业，则可以利用 AWS Step Functions 来简化在 AWS 内部或本地运行的批处理作业的编排。此示例项目演示了使用 Step Functions、AWS Batch 和 Lambda 编排批处理作业。
- 监控批处理作业以发现异常情况，例如作业完成用时超过了应有的时间。您可以使用 CloudWatch Container Insights 之类的工具来监控 AWS Batch 环境和作业。在这种情况下，工作负载会从头停止下一个作业，并向相关人员通知异常情况。

资源

相关文档：

- AWS Cloud 运维：监控和可观测性
- Amazon Builders' Library：分布式系统相关挑战
- REL11-BP05 使用静态稳定性来防止双模态行为
- AWS Lambda 开发人员指南：AWS Lambda 中的错误处理和自动重试
- Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications
- AWS 消息收发
- 什么是流数据？
- AWS Lambda 开发人员指南：异步调用
- Amazon Simple Queue Service 常见问题：FIFO 队列
- Amazon Kinesis Data Streams Developer Guide: Handling Duplicate Records
- Amazon Simple Queue Service Developer Guide: Available CloudWatch metrics for Amazon SQS
- Amazon Kinesis Data Streams Developer Guide: Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch

- AWS X-Ray Developer Guide: AWS X-Ray concepts
- AWS Samples on GitHub: AWS Step functions Complex Orchestrator App
- AWS Batch User Guide: AWS Batch CloudWatch Container Insights

相关视频：

- AWS Summit SF 2022 – Full-stack observability and application monitoring with AWS (COP310)

相关工具：

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Service（Amazon S3）
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx for Lustre
- AWS Step Functions
- AWS Batch

REL04-BP02 实施松耦合的依赖关系

队列系统、流系统、工作流和负载均衡器等依赖关系是松耦合的。松耦合有助于隔离某个组件的行为与依赖于它的其他组件的行为，从而提升韧性和敏捷性。

解耦依赖关系（例如排队系统、流系统和工作流程），有助于最大限度地减少更改或故障对系统的影响。这种分离将组件的行为与依赖该组件的其他行为隔离开来，从而提高了韧性和敏捷性。

在紧耦合的系统中，更改一个组件可能导致需要更改依赖该组件的其他组件，从而使所有组件的性能均降低。松耦合会打破这种依赖关系，使存在依赖关系的组件只需了解经过版本控制而且已发布的接口。在依赖项之间实施松耦合将隔离一个组件中的故障，防止对其他组件造成影响。

松耦合允许您修改代码或向组件添加功能，同时最大限度地降低依赖于该组件的其他组件的风险。它还允许在组件级别实现精细的韧性，让您可以横向扩展或甚至改变依赖项的底层实施。

要通过松耦合进一步提升韧性，在可能的情况下采用异步组件交互。若确定对请求进行注册已足够，则此模型适用于无需立即响应的任何交互。它包含一个生成事件的组件和另外一个使用事件的组件。

两个组件不会通过直接点对点交互，但通常经由中间持久存储层集成，例如 Amazon SQS 队列或是 Amazon Kinesis 或 AWS Step Functions 这样的就流数据平台。

图 4：队列系统和负载均衡器等依赖关系是松散耦合的

Amazon SQS 队列和 AWS Step Functions 只是为松耦合增加中间层的两种方式。您还可以使用 Amazon EventBridge 在 AWS Cloud 中构建事件驱动型架构，而 Amazon EventBridge 可从其依赖的服务（事件使用器）中提取客户端（事件产生器）。如果需要进行高吞吐量、基于推送的多对多消息收发，Amazon Simple Notification Service（Amazon SNS）是可供选择的高效解决方案。通过 Amazon SNS 主题，您的发布者系统可以将消息扇出到大量订阅用户端点以便进行并行处理。

虽然队列具有多项优点，但在大多数硬性实时系统中，早于阈值时间（通常为秒）的请求应被视为过时（客户端已放弃而且不再等待响应）而不被处理。因此，较新（而且可能依然有效）的请求会被处理。

期望结果：实施松耦合依赖项可以将故障的影响范围最大限度地缩小到组件级别，有助于诊断和解决问题。松耦合还简化了开发周期，允许团队在模块级别实施更改，而不会影响依赖它的其他组件的性能。这种方法能够根据资源需求以及有助于提高成本效益的组件利用率，在组件层面进行横向扩展。

常见反模式：

- 部署整体工作负载。
- 直接在工作负载层之间调用 API，不具备失效转移或异步处理请求的功能。
- 使用共享数据进行紧密耦合。松耦合的系统应避免通过共享数据库或其他形式的紧密耦合数据存储共享数据，这可能会重新引入紧耦合并阻碍可扩展性。
- 忽略背压。当组件无法以相同速度处理传入数据时，您的工作负载应该能够减慢或停止传入数据。

建立此最佳实践的好处：松耦合有助于隔离某个组件的行为与依赖于它的其他组件的行为，从而提升韧性和敏捷性。组件中的故障相互隔离。

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

实施松耦合的依赖关系。可利用多种解决方案来构建松耦合的应用程序。其中包括用于实施全面托管的队列、自动化工作流、对事件的反应以及 API 等内容的服务，这些服务有助于将组件的行为相互隔离，从而提高韧性和敏捷性。

- 构建事件驱动型架构：Amazon EventBridge 有助于构建松耦合和分布式的事件驱动架构。

- 在分布式系统中实施队列：可以使用 Amazon Simple Queue Service（Amazon SQS）集成或解耦分布式系统。
- 将组件容器化为微服务：微服务有助于团队构建由小型独立组件组成且通过明确定义的 API 进行通信的应用程序。Amazon Elastic Container Service（Amazon ECS）和 Amazon Elastic Kubernetes Service（Amazon EKS）有助于快速使用容器。
- 使用 Step Functions 管理工作流程：Step Functions 有助于将多个 AWS 服务协调为灵活的工作流程。
- 利用发布-订阅（pub/sub）消息收发架构：Amazon Simple Notification Service（Amazon SNS）提供从发布者到订阅用户（也称为生产者和使用者）的消息传输。

实施步骤

- 事件驱动型架构中的组件由事件启动。事件是系统中发生的操作，例如用户将商品添加到购物车。操作成功后，将生成一个激活系统的下一个组件的事件。
  - Building Event-driven Applications with Amazon EventBridge
  - AWS re:Invent 2022 – Designing Event-Driven Integrations using Amazon EventBridge
- 分布式消息收发系统主要有三个部分，需要为基于队列的架构实施这些部分。它们包括分布式系统的组件、用于解耦的队列（分布在 Amazon SQS 服务器上）以及队列中的消息。典型的系统有将消息发送到队列中的产生器和从队列接收消息的使用器。该队列将消息存储在多台 Amazon SQS 服务器上以实现冗余。
  - Basic Amazon SQS architecture
  - 使用 Amazon Simple Queue Service 在分布式应用程序间发送消息
- 由于松耦合的组件由独立团队管理，因此微服务如果得到充分利用，可以增强可维护性并提高可扩展性。它还允许在发生变化时将行为隔离到单个组件。
  - 在 AWS 上实施微服务
  - Let's Architect! Architecting microservices with containers
- 借助 AWS Step Functions，您可以构建分布式应用程序、实现流程自动化、编排微服务等。将多个组件编排到一个自动化工作流程中，让您可以将应用程序中的依赖关系解耦。
  - 使用 AWS Step Functions 和 AWS Lambda 创建无服务器工作流程
  - AWS Step Functions 入门

资源

相关文档：

- Amazon EC2: Ensuring Idempotency
- Amazon Builders' Library：分布式系统相关挑战
- Amazon Builders' Library：Reliability, constant work, and a good cup of coffee
- 什么是 Amazon EventBridge？
- What Is Amazon Simple Queue Service?
- Break up with your monolith
- Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS
- Basic Amazon SQS architecture
- Queue-Based Architecture

相关视频：

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda
- AWS re:Invent 2022 – Designing Event-Driven Integrations using Amazon EventBridge
- AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices

REL04-BP03 持续工作

系统会在负载中存在剧烈快速更改时失败。例如，如果您的工作负载执行的一项运行状况检查监控着数千个服务器的运行状况，每次都应发送相同大小的有效负载（当前状态的完整快照）。无论是否有服务器或有多少服务器发生故障，运行状况检查系统都会持续工作，而不会有剧烈、快速的变动。

例如，如果运行状况检查系统正在监控 10 万台服务器，在通常较低的服务器故障率下，它的负载是正常的。但如果发生重大事件让一半的服务器运行状况不佳，则运行状况检查系统会因为尝试更新通知系统以及向其客户端传送状态而变得不堪重负。因此，运行状况检查系统每次都应发送当前状态的完整快照，10 万台服务器的运行状况状态（每个状态都用一位表示）仅占 12.5 KB 的有效负载。无论是没有服务器发生故障还是所有服务器都发生故障，运行状况检查系统都会持续工作，而大幅度骤变也不会威

胁到系统的稳定性。这实际上就是 Amazon Route 53 处理对端点（例如 IP 地址）运行状况检查的方式，从而确定最终用户如何路由到这些端点。

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

- 持续工作，这样系统就不会在负载发生骤变时出现故障。
- 实施松耦合的依赖关系。队列系统、流系统、工作流和负载均衡器等依赖关系是松耦合的。松耦合有助于隔离某个组件的行为与依赖于它的其他组件的行为，从而提升韧性和敏捷性。
  - Amazon Builders' Library：Reliability, constant work, and a good cup of coffee
  - AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
    - 以监控 10 万台服务器的运行状况检查系统为例，对工作负载进行设计，确保无论成功或失败次数如何，有效负载大小都保持不变。

资源

相关文档：

- Amazon EC2: Ensuring Idempotency
- Amazon Builders' Library：分布式系统相关挑战
- Amazon Builders' Library：Reliability, constant work, and a good cup of coffee

相关视频：

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP04 使变异操作幂等

幂等服务承诺每个请求只确切处理一次，因此发起多个相同请求与发起单个请求的效果相同。这使客户端可以更轻松地进行重试，而不必担心多次错误地处理请求。要执行此操作，客户端可以发出具有幂等性令牌的 API 请求，每当重复该请求时都会使用此令牌。幂等服务 API 使用令牌来返回响应，该响应与首次完成请求时返回的响应相同，即使系统的底层状态已经改变也是如此。

在分布式系统中，至多（客户端仅发起一个请求）或至少（持续发起请求直到客户端收到成功确认）执行某项操作一次相对简单。难就难在要保证某项操作确切执行一次，从而使发出多个相同的请求和发出单个请求具有一样的效果。在 API 中使用幂等性令牌，服务可以一次或多次收到变异请求，而不需要创建重复的记录或产生副作用。

期望结果：您有一个一致、有据可查且广泛采用的方法来确保跨所有组件和服务的幂等性。

常见反模式：

- 您不加选择地应用幂等性，即使不需要也是如此。
- 您引入过于复杂的逻辑来实现幂等性。
- 您使用时间戳作为幂等性的密钥。由于时钟偏差或多个客户端使用相同的时间戳来应用更改，这可能会导致不准确。
- 您存储整个有效载荷以保持幂等性。在这种方法中，您为每个请求保存完整的数据有效载荷，并对于每个新请求将其覆盖。这可能会降低性能并影响可扩展性。
- 您在不同服务之间以不一致的方式生成密钥。如果密钥不一致，服务可能无法识别重复的请求，从而导致意想不到的结果。

建立此最佳实践的好处：

- 提高了可扩展性：系统可以处理重试和重复的请求，而无需执行额外的逻辑或复杂的状态管理。
- 增强了可靠性：幂等性有助于服务以一致的方式处理多个相同的请求，从而降低意外副作用或重复记录的风险。这在分布式系统中尤其重要，在此类系统中，网络故障和重试很常见。
- 提高了数据一致性：由于同一个请求会产生相同的响应，因此幂等性有助于保持分布式系统间的数据一致性。这对于维护事务和操作的完整性至关重要。
- 错误处理：幂等性令牌使错误处理变得更加简单。如果客户端由于问题而未收到响应，则它可以使用相同的幂等性令牌安全地重新发送请求。
- 操作透明度：通过幂等性，可以更好地进行监控和日志记录。服务可以使用其幂等性令牌记录请求，这可以更轻松地跟踪和调试问题。
- 简化了 API 合约：它可以简化客户端和服务器端系统之间的合约，并减少对错误数据处理的担忧。

在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

在分布式系统中，至多（客户端仅发起一个请求）或至少（客户端持续发起请求直到确认成功）执行某项操作一次相对简单。然而，确切一次 实施此行为则具有挑战性。为了实现这一点，客户端应该为每个请求生成并提供幂等性令牌。

通过使用幂等性令牌，服务可以区分新请求和重复的请求。当服务收到带有幂等性令牌的请求时，它会检查该令牌是否已被使用。如果该令牌已被使用，则该服务会检索并返回存储的响应。如果令牌是新的，则服务会处理请求，将响应与令牌一起存储，然后返回响应。这种机制使所有响应都是幂等的，从而提高了分布式系统的可靠性和一致性。

幂等性也是事件驱动型架构的一项重要行为。这些架构通常由消息队列提供支持，例如 Amazon SQS、Amazon MQ、Amazon Kinesis Streams 或 Amazon Managed Streaming for Apache Kafka（MSK）。在某些情况下，只发布一次的消息可能会意外地传送多次。当发布者生成幂等性令牌并将其包含在消息中时，它要求对收到的任何重复消息的处理不会导致对同一消息执行重复的操作。使用者应跟踪收到的每个令牌，并忽略包含重复令牌的消息。

服务和使用者还应将收到的幂等性令牌传递给它调用的任何下游服务。处理链中的每个下游服务都同样负责确保实施幂等性，以避免多次处理消息的副作用。

实施步骤

1. 确定幂等操作

   确定哪些操作需要幂等性。这些操作通常包括 POST、PUT 和 DELETE HTTP 方法以及数据库插入、更新或删除操作。不变异状态的操作（例如只读查询）通常不需要幂等性，除非它们有副作用。

2. 使用唯一标识符

   在发送者发送的每个幂等操作请求中包含一个唯一的令牌，既可以直接在请求中，也可以作为其元数据的一部分（例如，HTTP 标头）。这可让接收者识别和处理重复的请求或操作。通常用于令牌的标识符包括 Universally Unique Identifiers (UUIDs) 和 K-Sortable Unique Identifiers (KSUIDs)。

3. 跟踪和管理状态

   维护工作负载中每个操作或请求的状态。这可以通过将幂等性令牌和相应的状态（例如待处理、已完成或失败）存储在数据库、缓存或其它持久存储中来实现。此状态信息可让工作负载识别和处理重复的请求或操作。

如果需要，可通过使用适当的并发控制机制（例如锁定、事务或乐观并发控制）来保持一致性和原子性。这包括记录幂等令牌以及运行与为请求提供服务相关联的所有变异操作的过程。这有助于防止竞争条件并验证幂等操作是否正确运行。

定期从数据存储中移除旧的幂等性令牌来管理存储和性能。如果存储系统支持此操作，请考虑对数据使用过期时间戳（通常称为生存时间或 TTL 值）。重用幂等性令牌的可能性会随时间推移而降低。

通常用于存储幂等性令牌和相关状态的常见 AWS 存储选项包括：

- Amazon DynamoDB：DynamoDB 是一项 NoSQL 数据库服务，可提供低延迟性能和高可用性，因此非常适合存储与幂等性相关的数据。DynamoDB 的键值和文档数据模型支持高效存储和检索幂等性令牌和关联的状态信息。如果应用程序在插入幂等性令牌时设置了 TTL 值，则 DynamoDB 也可以自动使这些令牌过期。
- Amazon ElastiCache：ElastiCache 能够以高吞吐量、低延迟和低成本存储幂等性令牌。如果应用程序在插入幂等性令牌时设置了 TTL 值，ElastiCache（Redis）和 ElastiCache（Memcached）也可以自动使这些令牌过期。
- Amazon Relational Database Service（RDS）：可以使用 Amazon RDS 存储幂等性令牌和相关状态信息，特别是在应用程序已经将关系数据库用于其它用途的情况下。
- Amazon Simple Storage Service（S3）：Amazon S3 是一项高度可扩展和耐用的对象存储服务，可用于存储幂等性令牌和相关元数据。S3 的版本控制功能对于维护幂等操作的状态特别有用。存储服务的选择通常取决于诸如因素，例如：与幂等性相关的数据量、所需的性能特征、对耐久性和可用性的需求，以及幂等性机制如何与整体工作负载架构集成。

4. 实施幂等操作

将 API 和工作负载组件设计为幂等的。将幂等性检查纳入工作负载组件。在处理请求或执行操作之前，请检查是否已经处理了唯一标识符。如果已处理，则返回之前的结果，而不是再次执行该操作。例如，如果客户端发送一个创建用户的请求，请检查是否已存在具有相同唯一标识符的用户。如果该用户存在，则应返回现有用户信息，而不是创建新用户。同样，如果队列使用者收到带有重复幂等性令牌的消息，则该使用者应忽略该消息。

创建全面的测试套件来验证请求的幂等性。它们应涵盖各种各样的场景，例如成功的请求、失败的请求和重复的请求。

如果工作负载利用 AWS Lambda 函数，请考虑使用 Powertools for AWS Lambda。Powertools for AWS Lambda 是一个开发人员工具包，有助于在使用 AWS Lambda 函数时实施无服务器最佳实践

并提高开发人员速度。特别是，它提供了一个实用程序，可将 Lambda 函数转换为可以安全重试的幂等操作。

5. 清晰地传达幂等性

   记录 API 和工作负载组件，以清晰地传达操作的幂等性质。这有助于客户了解预期行为以及如何可靠地与工作负载进行交互。

6. 监控和审计

   实施监控和审计机制，以检测与响应的幂等性相关的任何问题，例如意外的响应变化或过多的重复请求处理。这有助于您检测和调查工作负载中的任何问题或意外行为。

资源

相关最佳实践：

- REL05-BP03 控制与限制重试调用
- REL06-BP01 为工作负载监控全部组件（生成）
- REL06-BP03 发送通知（实时处理和报警）
- REL08-BP02 将功能测试作为部署的一部分进行集成

相关文档：

- The Amazon Builders' Library: Making retries safe with idempotent APIs
- Amazon Builders' Library：分布式系统相关挑战
- Amazon Builders' Library：Reliability, constant work, and a good cup of coffee
- Amazon Elastic Container Service: Ensuring idempotency
- 如何让我的 Lambda 函数保持幂等性？
- Ensuring idempotency in Amazon EC2 API requests

相关视频：

- Building Distributed Applications with Event-driven Architecture - AWS Online Tech Talks
- AWS re:Invent 2023 - Building next-generation applications with event-driven architecture
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge

- AWS re:Invent 2018 - Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019 - Moving to event-driven architectures (SVS308)

相关工具：

- Idempotency with AWS Lambda Powertools (Java)
- Idempotency with AWS Lambda Powertools (Python)
- AWS Lambda Powertools GitHub page

REL 5. 如何在分布式系统中设计交互以缓解或承受故障？

分布式系统依赖于通信网络实现组件（例如服务器或服务）的互联。尽管这些网络中存在数据丢失或延迟，但是您的工作负载必须可靠运行。分布式系统组件的运行方式不得对其他组件或工作负载产生负面影响。这些最佳实践使工作负载能够承受压力或故障，从中更快地恢复，并且降低此类损坏的影响。其结果是缩短平均恢复时间（MTTR）。

最佳实践

- REL05-BP01 实施优雅降级以将适用的硬依赖关系转换为软依赖关系
- REL05-BP02 限制请求
- REL05-BP03 控制与限制重试调用
- REL05-BP04 快速失效机制和限制队列
- REL05-BP05 设置客户端超时
- REL05-BP06 尽可能使系统为无状态
- REL05-BP07 实施紧急杠杆

REL05-BP01 实施优雅降级以将适用的硬依赖关系转换为软依赖关系

即使依赖项不可用，应用程序组件也应继续执行其核心功能。应用程序组件可以提供稍微陈旧的数据、替代数据，甚至没有数据。这可确保在提供核心业务价值的同时，将局部故障对整体系统功能造成的障碍减至最少。

期望结果：某个组件的依赖项运行状况不佳时，该组件仍可在性能降低的条件下运行。组件的故障模式应视为正常运行。工作流在设计时，应确保此类故障不会导致完全失败，或者至少实现可预测和可恢复的状态。

常见反模式：

- 未确定所需的核心业务功能。即使在依赖项故障期间也不测试组件是否正常运行。
- 不论是出错时，还是当多个依赖项中只有一个不可用且仍可以返回部分结果时，不提供任何数据。
- 在事务部分失败时造成不一致的状态。
- 没有替代方法用于访问中央 Parameter Store。
- 在刷新失败时，使本地状态失效或清空，而没有考虑这样做的后果。

建立此最佳实践的好处：优雅降级可以提高整个系统的可用性，即使在故障期间也能保持最重要功能的功能。

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

实施优雅降级有助于最大限度地减少依赖项故障对组件功能的影响。理想情况下，组件检测依赖项故障，并以对其他组件或客户影响最小的方式解决这些故障。

为优雅降级设计架构意味着在依赖项设计期间，需要考虑潜在的故障模式。对于每种故障模式，都要有办法向调用方或客户提供组件的大部分功能，或者至少提供最关键的功能。这些注意事项可以作为额外的要求进行测试和验证。理想情况下，即使一个或多个依赖项出现故障，一个组件也能够以可接受的方式执行其核心功能。

这既是商业议题，也是技术议题。所有业务要求都很重要，应尽可能满足。但是，确定在无法满足所有要求时会出现什么情况，这种做法同样很有意义。系统可以设计为具备可用性和一致性，但是如果必须放弃一个要求，那么哪个要求更重要？ 对于付款处理而言，可能一致性更重要。对于实时应用程序，可能可用性更重要。对于面向客户的网站，这个回答可能取决于客户的期望值。

其具体的意义取决于组件的要求，以及将什么功能视为其核心功能。例如：

- 在登录页面上，电子商务网站可能会显示来自多个不同系统的数据，例如个性化推荐、最热销的产品和客户订单状态。当一个上游系统出现故障时，合理的做法是显示其余的所有信息，而不是向客户显示一个错误页面。
- 对于执行批量写入的组件，如果某个单独的操作失败，它仍应继续执行批处理。实施重试机制应该很简单。要实施重试机制，可以向调用方返回哪些操作成功、哪些操作失败的信息，或者将失败的请求放入死信队列中来实施异步重试。同时还应记录有关失败操作的信息。
- 处理事务的系统必须确保，要么执行了所有更新，要么未执行任何更新。对于分布式事务，在同一个事务后面的操作失败时，可以使用 Sega 模式来回滚先前的操作。这里的核心功能是保持一致性。

- 时间关键型系统应能够处理未及时响应的依赖项。在这些情况下，可以使用断路器模式。当来自依赖项的响应开始超时时，系统可以切换为关闭状态，不进行额外的调用。
- 应用程序可以从 Parameter Store 中读取参数。创建具有默认参数集的容器镜像，并在 Parameter Store 不可用时使用这些镜像，这种做法会很有用。

请注意，需要对组件出现故障时所采取的途径进行测试，而且这一途径应该比主要途径简单得多。通常，应避免使用回退策略。

实施步骤

确定外部和内部依赖项。考虑这些依赖项可能出现什么样的故障。思考能在故障期间尽力减少对上游和下游系统以及客户的负面影响的方法。

以下是依赖项列表以及在依赖项故障时如何优雅降级：

1. 依赖项部分故障：一个组件可以向下游系统发出多个请求，这可以是向一个系统发出多个请求，也可以是向多个系统发出一个请求。根据具体的业务环境，可能需要采用不同的处理方式（有关更多详细信息，请参阅“实施指导”中的示例）。
2. 下游系统因高负载无法处理请求：如果对下游系统的请求持续失败，则继续重试就没有意义了。这可能会对已经过载的系统造成额外负载，使得系统更难于恢复。这时可以使用断路器模式，该模式监视对下游系统的失败调用。如果大量调用失败，它会停止向下游系统发送更多请求，仅不定期让调用通过，以测试下游系统是否再次可用。
3. Parameter Store 不可用：要转换 Parameter Store，可以使用容器或机器映像中包含的软依赖项缓存或合理默认值。请注意，这些默认值需要保持为最新并包含在测试套件中。
4. 监控服务或其他非功能性依赖项不可用：如果某个组件间歇性地无法将日志、指标或跟踪发送到中央监控服务，通常最好还是正常执行业务功能。长时间静默地不进行日志记录或不推送指标通常不可接受。此外，某些应用场景可能需要完整的审核条目才能满足合规性要求。
5. 关系数据库的主实例可能不可用：与几乎所有关系数据库一样，Amazon Relational Database Service 只能有一个主写入器实例。对于写入工作负载，这会造成单点故障，并增加扩缩的难度。使用多可用区配置来实现高可用性，或者使用 Amazon Aurora Serverless 无服务器架构来实现更好的扩展能力，可以部分缓解这种情况。对于非常高的可用性要求，完全不依赖于主写入器是有意义的。对于只读查询，可以使用只读副本，这提供了冗余和横向扩展能力，而不仅仅是纵向扩展。对写入操作可以进行缓冲，例如缓冲在 Amazon Simple Queue Service 队列中，这样即使主写入器暂时不可用，仍然可以接受来自客户的写入请求。

资源

相关文档：

- Amazon API Gateway：限制对 API 的请求以提高吞吐量
- CircuitBreaker (summarizes Circuit Breaker from “Release It!” book)
- Error Retries and Exponential Backoff in AWS
- Michael Nygard “Release It! Design and Deploy Production-Ready Software”
- Amazon Builders' Library：避免在分布式系统中回退
- Amazon Builders' Library：避免无法克服的队列积压
- Amazon Builders' Library：缓存挑战和策略
- Amazon Builders' Library：超时、重试和抖动回退

相关视频：

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL05-BP02 限制请求

限制请求，防范因需求意外增加而导致的资源耗尽情况。系统将处理未超过限制速率的请求，而超过所定义限制的请求将被拒绝，并返回一条消息，指出请求已受限制。

期望结果：使用请求限制可以缓解客户流量突增、泛洪攻击或重试风暴所造成的大量容量峰值情况，让工作负载能够继续正常处理支持的请求量。

常见反模式：

- 未实施 API 端点限制，或者未考虑预期容量即保留默认值。
- API 端点未经过负载测试，也未测试节流限制。
- 限制请求速率而未考虑请求大小或复杂性。
- 测试最大请求速率或最大请求大小，但未同时测试两者。
- 资源预置的限制与测试中确定的限制不同。
- 尚未为应用程序到应用程序的（A2A）API 使用者配置或考虑使用量计划。
- 横向扩展的队列使用者没有配置最大并发设置。
- 没有基于每个 IP 地址实施速率限制。

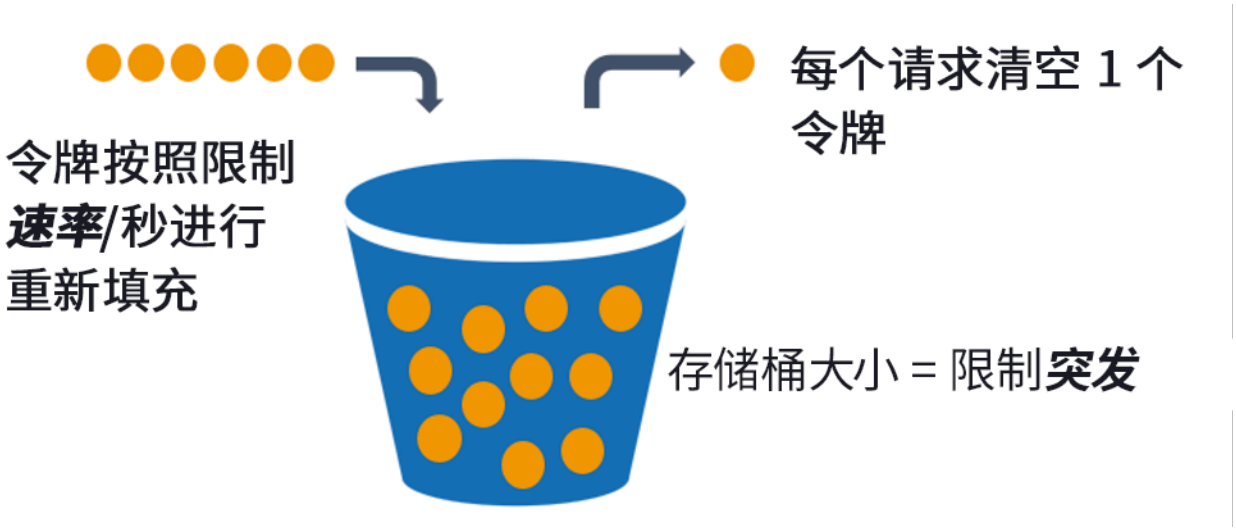
建立此最佳实践的好处：在遇到意外的容量峰值时，设置了节流限制的工作负载能够正常运行，并成功处理已接受的请求负载。API 和队列上突然或持续出现的请求峰值会受到限制，不会耗尽请求处理资源。速率限制会限制单独的请求者，这样来自单个 IP 地址或 API 使用者的大量流量就不会耗尽资源，从而不会影响其他使用者。

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

服务应设计为处理已知的请求容量；这种容量可以通过负载测试来确立。如果请求到达速率超过限制，则会发出相应的响应，表示请求已被限制。这让使用者可以处理错误并稍后重试。

当您的服务需要实施节流时，可以考虑实施令牌存储桶算法，每个令牌对应于一个请求。令牌按照每秒的限制速率重新填充，并按照每个请求一个令牌的模式异步清空。



令牌存储桶算法。

Amazon API Gateway 根据账户和区域限制实施令牌存储桶算法，可通过使用量计划为每个客户端配置。此外，Amazon Simple Queue Service（Amazon SQS）和 Amazon Kinesis 可以缓冲请求来稳定请求速率，并允许对可以处理的请求实施更高的节流速率。最后，您可以使用 AWS WAF 实施速率限制，限制产生异常高负载的特定 API 使用者。

实施步骤

您可以为 API 配置 API Gateway 节流限制，并在超过限制时返回 `429 Too Many Requests` 错误。您可以将 AWS WAF 与 AWS AppSync 和 API Gateway 端点结合使用，根据各个 IP 地址来启用速率限制。此外，如果系统能够接受异步处理，则可以将消息放入队列或流中，借此加快对服务客户端的响应，这样便可以突增到更高的限制速率。

采用异步处理，在将 Amazon SQS 配置为 AWS Lambda 的事件源时，您可以配置最大并发数，避免高事件速率消耗工作负载或账户中其他服务所需的可用账户并发执行配额。

虽然 API Gateway 提供了令牌存储桶的托管实施，但在无法使用 API Gateway 的情况下，您可以针对服务利用具体语言的令牌存储桶开源实施（参见“资源”中的相关示例）。

- 了解每个区域的账户级别、每个阶段的 API 和每个使用计划级别的 API 密钥的 API Gateway 节流限制，并进行配置。
- 对 API Gateway 和 AWS AppSync 端点应用 AWS WAF 速率限制规则，防范泛洪并阻止恶意 IP。对于 A2A 使用者，也可以在 AWS AppSync API 密钥上配置速率限制规则。
- 对于 AWS AppSync API，请考虑所需节流控制是否超过速率限制；如果超过，则在 AWS AppSync 端点前面配置 API Gateway。
- 在将 Amazon SQS 队列设置为 Lambda 队列使用者的触发器时，请将最大并发数设置为足以满足服务级别目标，但不会消耗会影响其他 Lambda 函数的并发限制的值。通过 Lambda 使用队列时，请考虑为相同账户和区域中的其他 Lambda 函数设置预留并发度。
- 将 API Gateway 与 Amazon SQS 或 Kinesis 的原生服务集成结合使用可缓冲请求。
- 如果无法使用 API Gateway，请查看具体语言的库，以便为工作负载实施令牌存储桶算法。查看示例部分，然后自行研究找出合适的库。
- 对您计划设置的限制或者您计划允许增加的限制进行测试，并记录测试后的限制值。
- 不要将限制值提高到超出您在测试中确立的限制值。增加限制时，请先确认预置的资源是否已经等于或大于测试场景中预置的资源，然后再进行增加。

资源

相关最佳实践：

- REL04-BP03 持续工作
- REL05-BP03 控制与限制重试调用

相关文档：

- Amazon API Gateway：限制对 API 的请求以提高吞吐量
- AWS WAF: Rate-based rule statement
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source
- AWS Lambda: Maximum Concurrency

相关示例：

- The three most important AWS WAF rate-based rules
- Java Bucket4j
- Python token-bucket
- Node token-bucket
- .NET System Threading Rate Limiting

相关视频：

- Implementing GraphQL API security best practices with AWS AppSync

相关工具：

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF
- AWS 上的虚拟等候室

REL05-BP03 控制与限制重试调用

使用指数回退来重试请求，每次重试之间的间隔会逐渐延长。在两次重试之间引入抖动来随机调整重试间隔。限制最大重试次数。

期望结果：分布式软件系统中的常见组件包括服务器、负载均衡器、数据库和 DNS 服务器。在正常运行期间，这些组件对请求的响应可能是临时错误或者受限制错误，也能是无论如何重试都会持续存在的错误。当客户端向服务发出请求时，请求会消耗资源，包括内存、线程、连接、端口或任何其他有限的资源。控制和限制重试策略用于释放资源并最大限度地减少资源消耗，这样就可以避免承受压力的系统组件不堪重负。

当客户端请求超时或收到错误响应时，客户端应决定是否重试。如果进行重试，则会按照最大重试次数值，采用指数回退和抖动方法进行重试。因此，后端服务和进程可以缓解负载并缩短自我修复时间，从而加快恢复速度并成功处理服务请求。

常见反模式：

- 实施重试，但没有添加指数回退、抖动和最大重试次数值。指数回退和抖动有助于避免因无意间按照相同间隔协调进行重试，从而导致的人为流量峰值。
- 实施重试但没有测试重试的效果，或者假设 SDK 中已经内置了重试而不测试重试场景。
- 无法理解依赖项发布的错误代码，导致重试所有错误，包括那些有明确原因的错误，这些错误指出缺乏权限、配置错误或其他预计需要手动干预才能解决的情况。
- 没有解决可观测性实践，包括监控反复出现的服务故障并发出警报，以便了解和解决潜在问题。
- 在内置或第三方重试功能便已足够时，开发自定义重试机制。
- 在应用程序堆栈的多层进行重试，而重试方法导致重试尝试复杂化，进一步加剧了重试风暴中的资源消耗。一定要了解这些错误对您的应用程序以及所依赖的依赖项有何影响，然后仅在一个级别实施重试。
- 重试非幂等的服务调用，导致意外的副作用，例如重复的结果。

建立此最佳实践的好处：重试有助于客户端在请求失败时获得预期的结果，但也会消耗更多的服务器时间来获取所需的成功响应。当故障比率很低或者是临时性故障时，重试效果很好。当故障是由资源过载导致时，重试会导致情况进一步恶化。通过在客户端重试中添加指数回退和抖动，服务器可以从因资源过载导致的故障中恢复。抖动可避免请求同时出现造成峰值，指数回退可以减少因在正常请求负载中增添重试而导致的负载上升。最后，务必要配置最大重试次数或用时，避免产生导致亚稳态故障的积压。

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

控制与限制重试调用。在逐渐延长的间隔以后使用指数回退进行重试。引入抖动来随机调整重试间隔，并限制重试的最大次数。

默认情况下，一些 AWS SDK 实施重试和指数回退。在适用于工作负载的情况下，使用这些内置 AWS 实施。在调用幂等性服务时，以及重试可以提高客户端可用性时，在工作负载中实施类似的逻辑。根据应用场景确定超时以及何时停止重试。为重试应用场景构建和演练测试场景。

实施步骤

- 对于应用程序所依赖的服务，确定应用程序堆栈中最适合实施重试的层。
- 请注意，现有的 SDK 会针对您选择的语言，实施采用了指数回退和抖动方法的成熟重试策略，相比您自己编写重试实施，使用这些实施方法会更好。

- 请先验证服务是否具有幂等性，再实施重试。实施重试后，请确保在生产环境中进行测试，并定期进行演练。
- 调用 AWS 服务 API 时，请使用 AWS SDK 和 AWS CLI，并了解重试配置选项。确定默认值是否适用于应用场景，进行测试，并根据需要进行调整。

资源

相关最佳实践：

- REL04-BP04 使变异操作幂等
- REL05-BP02 限制请求
- REL05-BP04 快速失效机制和限制队列
- REL05-BP05 设置客户端超时
- REL11-BP01 监控工作负载的所有组件以检测故障

相关文档：

- Error Retries and Exponential Backoff in AWS
- Amazon Builders' Library：超时、重试和抖动回退
- Exponential Backoff and Jitter
- Making retries safe with idempotent APIs

相关示例：

- Spring Retry
- Resilience4j Retry

相关视频：

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

相关工具：

- AWS SDKs and Tools: Retry behavior
- AWS Command Line Interface: AWS CLI retries

REL05-BP04 快速失效机制和限制队列

当服务无法成功响应请求时，可采用快速失效机制。这样可释放与请求关联的资源，并允许该服务在资源不足的情况下进行恢复。快速失效机制一种成熟的软件设计模式，可用于在云端构建高度可靠的工作负载。队列也是一种成熟的企业集成模式，其能够实现平稳的负载，并在能够容忍异步处理的情况下，使得客户端能够释放资源。如果某个服务在正常条件下能够成功响应，但在请求速率过高时失败，请使用队列来缓冲请求。不过，不要允许出现较长的队列积压，否则可能导致处理已被客户端放弃的过时请求。

期望结果：当系统遇到资源争用、超时、异常或灰色故障等情况，导致无法实现服务等级目标时，快速失效机制策略可以加快系统恢复速度。如果系统必须承受流量峰值并能够适应异步处理，就可以使用队列来缓冲对后端服务的请求，让客户端可以快速释放请求，从而提高可靠性。在将请求缓冲到队列中时，系统将实施队列管理策略来避免出现无法克服的积压。

常见反模式：

- 实施消息队列，但不配置死信队列（DLQ）或 DLQ 数量警报来检测系统出现故障的时间。
- 不测量队列中消息的时限（关于延迟的度量）来了解队列使用者何时落后或者出错并导致重试。
- 当业务不需要再存在时，处理积压的消息没有任何价值，但不从队列中清除这些消息。
- 当后进先出（LIFO）队列可以更好地满足客户端需求时，配置先进先出（FIFO）队列，例如，在不需要严格排序并且处理积压内容会延误所有新的和注重时效性的请求时，先进先出队列会导致所有客户端出现违反服务等级协议的情况。
- 向客户端公开内部队列，而不是公开那些管理工作摄入并将请求放入内部队列的 API。
- 将过多的工作请求类型合并到一个队列中，这会导致在一个队列中分配对多种请求类型的资源需求，进而会加剧积压情况。
- 在同一个队列中处理复杂请求和简单请求，但这些请求具有不同的监控、超时和资源分配需求。
- 不验证输入，也不使用断言在软件中实施快速失效机制，这些机制可将异常上报到更高级别的组件来轻松处理错误。
- 不从请求路由中移除出现故障的资源，尤其是在由于崩溃和重启、间歇性依赖项故障、容量减少或网络数据包丢失，导致同时出现成功和失败的灰色故障时。

建立此最佳实践的好处：采用快速失效机制的系统更容易调试和修复，通常在将版本发布到生产环境之前，在编码和配置阶段就会暴露问题。采用有效排队策略的系统在面对流量高峰和间歇性系统故障的情况时，能够提供更出色的韧性和可靠性。

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

快速失效机制策略能够通过编码形式构建到软件解决方案中，也能够在基础设施中配置。除了快速失效机制之外，还可通过队列这种简单而强大的架构技术来解耦系统组件，实现平稳的负载。Amazon CloudWatch：提供监控故障和发出警报的功能。在确定系统出现故障时，可以调用缓解策略，包括从受损资源进行失效转移。当系统使用 Amazon SQS 和其他队列技术实施队列来实现平稳的负载时，须考虑如何管理队列积压以及消息使用故障。

实施步骤

- 在软件中实施编程式断言或特定指标，并将其用于明确发出关于系统问题的警报。Amazon CloudWatch 有助于根据应用程序日志模式和 SDK 检测工具来创建指标和警报。
- 使用 CloudWatch 指标和警报从受损资源进行故障转移，这些受损资源会增加处理延迟或者在处理请求时反复失败。
- 要使用异步处理，您可以设计 API 来接受请求，并使用 Amazon SQS 将请求附加到内部队列，然后向生成消息的客户端发送成功消息，这样客户端就可以释放资源，并继续处理其他工作，同时后端队列使用者可以处理请求。
- 在每次从队列中删除消息时，通过将现在的时间戳与消息时间戳进行比较来生成 CloudWatch 指标，从而测量和监控队列处理延迟。
- 如果故障导致无法成功处理消息，或者有大量的流量高峰无法按照服务等级协议的要求处理，则将较旧或过多的流量转到溢出队列。这样便可以优先处理新的工作，在有可用容量时再处理较早的工作。这种技术与 LIFO 处理有相似之处，使得可以对所有新工作进行正常的系统处理。
- 使用死信或再驱动队列，将无法处理的消息从积压中移至另一个位置，供以后研究和解决
- 您可以重试，或者在允许的情况下，通过将现在的时间戳与消息时间戳进行比较，丢弃与发出请求的客户端不再相关的消息，以此来删除旧消息。

资源

相关最佳实践：

- REL04-BP02 实施松耦合的依赖关系
- REL05-BP02 限制请求
- REL05-BP03 控制与限制重试调用
- REL06-BP02 定义与计算指标（聚合）
- REL06-BP07 对系统中的请求进行端到端跟踪监控

相关文档：

- 避免无法克服的队列积压
- Fail Fast
- 如何防止我的 Amazon SQS 队列中日益积压的消息？
- Elastic Load Balancing: Zonal Shift
- Amazon Application Recovery Controller: Routing control for traffic failover

相关示例：

- Enterprise Integration Patterns: Dead Letter Channel

相关视频：

- AWS re:Invent 2022 – Operating highly available Multi-AZ applications

相关工具：

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 设置客户端超时

您应适当设置连接和请求的超时，对其进行系统性验证，不要依赖默认值，因为默认值并不了解具体的工作负载情况。

期望结果：客户端超时应考虑当完成请求需要超长时间时，与等待请求相关的客户端、服务器和工作负载成本。由于无法知晓任何超时的确切原因，因此客户端必须运用对服务的了解，预测可能的原因和相应的超时时间。

客户端会根据配置的值连接超时。遇到超时后，客户端会决定回退并重试，或者打开断路器。这些模式可避免发出会加剧底层错误状况的请求。

常见反模式：

- 不了解系统超时或默认超时。
- 不了解正常的请求完成时间。
- 不了解完成请求需要超长时间的可能原因，也不了解与等待完成这些请求相关的客户端、服务器或工作负载性能成本。
- 不了解网络受损只要在达到超时后就可能会导致请求失败，也不了解未采用更短的超时时间而招致的客户端和工作负载性能成本。
- 未针对连接和请求来测试超时场景。
- 将超时设置得过高，这会导致等待时间过长并增加资源使用。
- 将超时设置得过低，会导致人为故障。
- 忽略处理远程调用超时错误的模式，例如断路器和重试。
- 不考虑监控服务调用错误率、延迟的服务等级目标和延迟异常值。这些指标能够提供关于过长或不合理超时的洞察信息

建立此最佳实践的好处：配置了远程调用超时，且系统在设计上可以轻松处理超时，这样在远程调用响应异常缓慢时能够节省资源，且服务客户端可以轻松处理超时错误。

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

针对所有服务依赖项调用以及一般情况下的所有跨流程调用，设置连接超时和请求超时。许多框架具有内置超时功能，但仍需谨慎，因为一些超时的默认值为无限值，或者高于您的服务目标可以接受的值。过高的值会降低超时的实用性，因为客户端等待超时发生时，系统会继续消耗资源。过低的值可能会重试请求过多次，因而导致后端流量增加以及延迟变长。在有些情况下，由于要对全部请求进行重试，从而可能导致完全中断。

在确定超时策略时，请考虑以下几点：

- 由于请求的内容、目标服务受损或网络分区故障，处理请求所需的时间可能比正常时间要长。
- 请求如果具有成本异常高的内容，就可能会消耗不必要的服务器和客户端资源。在这种情况下，让这些请求超时而不进行重试可以节省资源。服务还应利用限制和服务器端超时，来保护自身免受成本异常高的内容的侵害。
- 如果由于服务受损而导致请求用时超长，则可以使请求超时并进行重试。请求和重试的服务成本需要考虑在内，但如果原因是局部受损，则重试的成本可能不会太高，而且会减少客户端资源消耗。根据损害的性质，超时还可能会释放服务器资源。

- 如果由于网络未能传输请求或响应，导致请求完成用时很长，则可以使请求超时并进行重试。由于未能传输请求或响应，因此无论超时时间多长，结果都是失败。在这种情况下，超时不会释放服务器资源，但可以释放客户端资源并提高工作负载性能。

利用重试和断路器等成熟的设计模式，可轻松地处理超时并支持快速失效机制方法。AWSSDK 和 AWS CLI 允许配置连接和请求超时，以及使用指数回退和抖动进行重试。AWS Lambda 函数支持超时配置，所以借助 AWS Step Functions，您可以利用与 AWS 服务和 SDK 的预构建集成，以低代码方式构建断路器。AWS App MeshEnvoy 提供超时和断路器功能。

实施步骤

- 配置远程服务调用的超时，并利用内置语言超时功能或开源超时库。
- 当您的工作负载使用 AWS SDK 进行调用时，请查看文档以了解具体语言的超时配置。
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- 在工作负载中使用 AWS SDK 或 AWS CLI 命令时，可通过设置 `connectTimeoutInMillis` 和 `tlsNegotiationTimeoutInMillis` 的 AWS 配置默认值来配置默认超时值。
- 应用命令行选项 `cli-connect-timeout` 和 `cli-read-timeout` 来控制对 AWS 服务的一次性 AWS CLI 命令。
- 监控远程服务调用的超时，并针对持续性错误设置警报，这样您就能够主动处理错误情况。
- 实施对调用错误率、延迟的服务等级目标和延迟异常值的 CloudWatch Metrics 和 CloudWatch 异常检测，有助于深入了解如何管理过长或不合理的超时。
- 配置 Lambda 函数的超时时间。
- 处理超时的时候，API Gateway 客户端必须实施自己的重试。对于下游集成，API Gateway 支持 50 毫秒到 29 秒的集成超时，并且不会在集成请求超时时重试。
- 实施断路器模式，避免在超时时进行远程调用。打开断路器避免调用失败，并在调用响应正常时关闭断路器。

- 对于基于容器的工作负载，请查看 App Mesh Envoy 功能，以便充分利用内置的超时和断路器。
- 使用 AWS Step Functions，以低代码方式为远程服务调用构建断路器，尤其是在调用 AWS 原生 SDK 和所支持的 Step Functions 集成时，以此简化工作负载。

资源

相关最佳实践：

- REL05-BP03 控制与限制重试调用
- REL05-BP04 快速失效机制和限制队列
- REL06-BP07 对系统中的请求进行端到端跟踪监控

相关文档：

- AWS SDK: Retries and Timeouts
- Amazon Builders' Library：超时、重试和抖动回退
- Amazon API Gateway 配额和重要说明
- AWS Command Line Interface: Command line options
- AWS SDK for Java 2.x: Configure API Timeouts
- AWS Botocore using the config object and Config Reference
- 适用于 .NET 的 AWS SDK: Retries and Timeouts
- AWS Lambda：配置 Lambda 函数选项

相关示例：

- Using the circuit breaker pattern with AWS Step Functions and Amazon DynamoDB
- Martin Fowler: CircuitBreaker

相关工具：

- AWS SDK
- AWS Lambda
- Amazon SQS
- AWS Step Functions

- AWS Command Line Interface

REL05-BP06 尽可能使系统为无状态

系统应该不需要状态，或者在不同的客户端请求之间卸载状态，磁盘上和内存中本地存储的数据不存在依赖关系。从而支持任意替换服务器，而且不会对可用性产生影响。

当用户或服务与应用程序进行交互，它们通常会执行一系列交互并构成一次会话。对于用户来说，会话是他们在使用应用程序时持续存在于请求之间的特殊数据。无状态应用程序是无需掌握之前交互而且不会存储会话信息的应用程序。

若采用无状态设计，则您可以使用无服务器计算服务，如 AWS Lambda 或 AWS Fargate。

除了服务器替换，无状态应用程序的另一项优点是，由于任何可用的计算资源（如 EC2 实例和 AWS Lambda 函数）都可以处理任何请求，因此它们可以进行横向扩展。

建立此最佳实践的好处：设计为无状态的系统更适合水平扩缩，因为可以根据流量和需求的波动情况增加或删除容量。此类系统还固有故障恢复能力，为应用程序开发提供了灵活性和敏捷性。

在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

让应用程序无状态。无状态应用程序支持水平扩缩，并且可以承受单个节点故障。分析并了解在架构内维持状态的应用程序组件。这有助于您评测过渡到无状态设计的潜在影响。无状态架构会解耦用户数据并分流会话数据。这为独立扩展每个组件提供了灵活性，可以满足不同的工作负载需求，并优化资源利用率。

实施步骤

- 确定并了解应用程序中的有状态组件。
- 将用户数据与核心应用程序逻辑分离开来进行管理，以此来解耦数据。
  - Amazon Cognito 可以使用身份池、用户池和 Amazon Cognito Sync 等功能，将用户数据与应用程序代码解耦。
  - 您可以通过将密钥存储在安全的集中位置来使用 AWS Secrets Manager 解耦用户数据。这意味着应用程序代码不需要存储密钥，这会令其更加安全。
  - 考虑采用 Amazon S3 存储大型非结构化数据，例如图像和文档。应用程序可以在需要时检索这些数据，无需将数据存储在内存中。
  - 使用 Amazon DynamoDB 存储用户资料等信息。您的应用程序可以近乎实时地查询这些数据。

- 将会话数据分流到数据库、缓存或外部文件。
  - Amazon ElastiCache、Amazon DynamoDB、Amazon Elastic File System（Amazon EFS）和 Amazon MemoryDB 都是可用于分流会话数据的 AWS 服务示例。
- 在确定需要将哪些状态和用户数据保留在所选存储解决方案中之后，设计无状态架构。

资源

相关最佳实践：

- REL11-BP03 自动修复所有层

相关文档：

- Amazon Builders' Library：避免在分布式系统中回退
- Amazon Builders' Library：避免无法克服的队列积压
- Amazon Builders' Library：缓存挑战和策略
- AWS 上无状态 Web 层的最佳实践

REL05-BP07 实施紧急杠杆

紧急杠杆是可帮助您的工作负载减轻可用性影响的快速流程。

紧急杠杆的工作原理是使用已知且经过测试的机制，禁用、节流或更改组件或依赖项的行为。这可以缓解因需求意外增加导致资源耗尽而造成的工作负载损失，并减少工作负载中非关键组件故障的影响。

期望结果：通过实施应急杠杆，您可以建立已知良好的流程，以保持工作负载中关键组件的可用性。在激活紧急杠杆期间，工作负载应进行优雅降级，并继续执行其关键业务功能。有关优雅降级的更多详细信息，请参阅 REL05-BP01 实施优雅降级以将适用的硬依赖关系转换为软依赖关系。

常见反模式：

- 非关键依赖关系的故障会影响核心工作负载的可用性。
- 在非关键组件受损时，不测试或验证关键组件的行为。
- 没有为紧急杠杆的激活或停用定义明确的标准。

建立此最佳实践的好处：实施紧急杠杆可以为解析器提供既定的流程来应对意外的需求激增或非关键依赖关系的故障，从而提高工作负载中关键组件的可用性。

在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

- 识别工作负载中的关键组件。
- 设计和构建工作负载中的关键组件，使其能够承受非关键组件的故障。
- 进行测试以验证关键组件在非关键组件出现故障期间的行为。
- 定义和监控相关指标或触发器，以启动紧急杠杆程序。
- 定义构成紧急杠杆的（手动或自动）程序。

实施步骤

- 识别工作负载中的关键业务组件。
  - 工作负载中的每个技术组件都应与相关业务职能相对应，并评定为关键组件或非关键组件。有关 Amazon 关键和非关键功能的示例，请参阅 Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second。
  - 这既是技术决策又是业务决策，而且因组织和工作负载而异。
- 设计和构建工作负载中的关键组件，使其能够承受非关键组件的故障。
  - 在分析依赖项期间，考虑所有潜在的故障模式，并验证您的紧急杠杆机制是否为下游组件提供了关键功能。
- 进行测试以验证关键组件在紧急杠杆激活期间的行为。
  - 避免双模态行为。有关更多详细信息，请参阅 REL11-BP05 使用静态稳定性来防止双模态行为。
- 定义和监控相关指标并针对指标发出警报，以便启动紧急杠杆程序。
  - 根据工作负载，找到要监控的正确指标。例如，这些指标可以是延迟，或者是对依赖项请求失败的次数。
- 定义构成紧急杠杆的手动或自动程序。
  - 这可能包括卸除负载、限制请求或实施优雅降级等机制。

资源

相关最佳实践：

- REL05-BP01 实施优雅降级以将适用的硬依赖关系转换为软依赖关系
- REL05-BP02 限制请求

- REL11-BP05 使用静态稳定性来防止双模态行为

相关文档：

- 自动实现无需干预的安全部署
- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

相关视频：

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

变更管理

问题

- REL 6. 如何监控工作负载资源？
- REL 7. 如何设计工作负载以适应需求变化？
- REL 8. 您如何实施更改？

REL 6. 如何监控工作负载资源？

日志和指标是深入了解工作负载运行状况的强大工具。您可以将工作负载配置为监控日志和指标，并在超过阈值或发生重大事件时发送通知。通过监控，您的工作负载可以发现超出低性能阈值和发生故障的情形，从而自动恢复以做出响应。

最佳实践

- REL06-BP01 为工作负载监控全部组件（生成）
- REL06-BP02 定义与计算指标（聚合）
- REL06-BP03 发送通知（实时处理和报警）
- REL06-BP04 自动响应（实时处理和警报）
- REL06-BP05 分析日志
- REL06-BP06 定期审核监控范围和指标
- REL06-BP07 对系统中的请求进行端到端跟踪监控

REL06-BP01 为工作负载监控全部组件（生成）

使用 Amazon CloudWatch 或第三方工具监控工作负载组件。使用 AWS Health 控制面板监控 AWS 服务。

应监控工作负载的全部组件，包括前端、业务逻辑和存储层。定义关键指标，描述如何将其从日志中提取出来（如有必要），并为对应的警报事件设置阈值。确保指标与工作负载的关键性能指标（KPI）相关，并使用指标和日志来识别服务性能下降的早期预警信号。例如，与业务成果相关的指标（例如每分钟成功处理的订单数）可以比技术指标（例如 CPU 利用率）更快地表明工作负载问题。使用 AWS Health 控制面板可提供 AWS 资源底层的 AWS 服务性能和可用性的个性化视图。

云中监控创造新的机会。大多数云提供商都开发了可自定义的挂钩，可以提供见解来帮助您监控多层工作负载。Amazon CloudWatch 等 AWS 服务应用统计和机器学习算法来持续分析系统和应用程序的指标，只需最少的用户干预即可确定正常基线和表面异常。异常检测算法将指标的季节性变化和趋势变化考虑在内。

AWS 提供了大量的监控和日志信息，这些信息可用于定义工作负载特定的指标、需求变化流程，也助于采用机器学习技术，无论是否具备机器学习专业知识如何。

此外，还可监控所有外部端点，确保这些端点独立于基本实施。这种主动监控可通过综合事务实现（有时被称为用户金丝雀，但与金丝雀部署不同），这些事务会按照工作负载的客户端所执行的操作，定期执行许多常见任务。确保这些任务的持续时间较短，不要让工作负载在测试期间过载。Amazon CloudWatch Synthetics 让您可以创建 Synthetics 金丝雀，以便对端点和 API 进行监控。您还可以整合 Synthetics 金丝雀客户端节点和 AWS X-Ray 控制台，精确定位哪些 Synthetics 金丝雀遇到错误、故障，或对指定时段的速率进行限制的问题。

期望结果：

收集并使用来自工作负载所有组件的关键指标，确保工作负载的可靠性和最佳的用户体验。若能检测到工作负载无法实现业务成果，可快速宣布发生灾难并从事件中恢复。

常见反模式：

- 仅监控连接到工作负载的外部接口。
- 不生成任何工作负载特定的指标，只依靠工作负载使用的 AWS 服务所提供的指标。
- 仅使用工作负载中的技术指标，而不监控与工作负载带来的非技术性 KPI 相关的任何指标。
- 依靠生产流量和简单的运行状况检查来监控并评估工作负载状态。

建立此最佳实践的好处：在工作负载的各个层级进行监控，便于更快地预测并解决构成工作负载的组件中的问题。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

1. 启用日志记录（如适用）。应从工作负载的所有组件获取监控数据。启用其他日志记录（例如 S3 访问日志），让工作负载记录工作负载特定的数据。收集来自 Amazon ECS、Amazon EKS、Amazon EC2、弹性负载均衡、AWS Auto Scaling 和 Amazon EMR 等服务的 CPU、网络 I/O 和磁盘 I/O 平均值的指标。有关向 CloudWatch 发布指标的 AWS 服务列表，请参阅发布 CloudWatch 指标的 AWS 服务。
2. 审查所有默认指标并探究任何数据收集欠缺。每项服务都会生成默认指标。通过收集默认指标，您可以更好地了解工作负载组件之间的依赖关系，以及组件的可靠性和性能对工作负载的影响。您可使用 AWS CLI 或 API 向 CloudWatch 发布自定义指标。
3. 评估所有指标，决定要针对工作负载中每项 AWS 服务的哪些指标发出警报。您可以选择对工作负载可靠性有重大影响的指标子集。关注关键指标和阈值有助于细化警报数量，也助于尽量减少误报。
4. 定义警报以及在调用警报之后工作负载的恢复流程。通过定义警报，您可以快速通知、上报和执行必要的步骤，以便从事件中恢复并达到规定的恢复时间目标（RTO）。您可以使用 Amazon CloudWatch 警报调用自动工作流，并根据定义的阈值启动恢复程序。
5. 探索使用综合事务来收集有关工作负载状态的相关数据。综合监控遵循相同的路线并执行与客户相同的操作，让您能够持续验证客户体验，即使应用程序中没有任何客户流量。使用综合事务，您可以早于客户先行发现问题。

资源

相关最佳实践：

- REL11-BP03 自动修复所有层

相关文档：

- Getting started with your AWS Health Dashboard – Your account health
- 发布 CloudWatch 指标的 AWS 服务
- Access Logs for Your Network Load Balancer
- Access logs for your application load balancer
- 访问 AWS Lambda 的 Amazon CloudWatch Logs
- Amazon S3 服务器访问日志记录

- Enable Access Logs for Your Classic Load Balancer
- Exporting log data to Amazon S3
- 在 Amazon EC2 实例上安装 CloudWatch 代理
- 发布自定义指标
- Using Amazon CloudWatch Dashboards
- 使用 Amazon CloudWatch 指标
- 使用金丝雀（Amazon CloudWatch Synthetics）
- What are Amazon CloudWatch Logs?

用户指南：

- 创建跟踪
- Monitoring memory and disk metrics for Amazon EC2 Linux instances
- 将 CloudWatch Logs 与容器实例结合使用
- VPC 流日志
- What is Amazon DevOps Guru?
- 什么是 AWS X-Ray ？

相关博客：

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

相关示例：

- Amazon Builders' Library：检测分布式系统的运营可见性
- Observability 讲习会

REL06-BP02 定义与计算指标（聚合）

从工作负载组件收集指标和日志，并从中计算相关的聚合指标。这些指标为工作负载提供了广泛而深入的可观测性，并可以显著提高韧性态势。

可观测性不仅仅是从工作负载组件中收集指标以及能够查看指标和针对指标发出警报。其最终目的是对工作负载的行为进行全面的了解。此类行为信息来自工作负载中的所有组件，包括它们所依赖的云服

务、精心制定的日志和指标。这些数据使您能够监督工作负载的整体行为，并可以非常详细地了解每个组件与每个工作单元的交互情况。

期望结果：

- 可以从工作负载组件和 AWS 服务依赖关系中收集日志，然后将其发布到一个便于访问和处理的中心位置。
- 日志包含高保真和准确的时间戳。
- 日志包含有关处理上下文的相关信息，例如跟踪标识符、用户或账户标识符以及远程 IP 地址。
- 可以从日志中创建聚合指标，这些指标从高层次视角表示工作负载的行为。
- 可以查询聚合的日志，以获得有关工作负载的深入和相关的见解，并确定实际和潜在的问题。

常见反模式：

- 您未从运行工作负载的计算实例或工作负载使用的云服务中收集相关日志或指标。
- 您忽略了与业务关键绩效指标（KPI）相关的日志和指标的收集。
- 您单独分析与工作负载相关的遥测数据，而没有采用聚合和关联。
- 您让指标和日志过快过期，这会阻碍趋势分析和识别反复出现的问题。

建立这些最佳实践的好处：您可以检测更多异常情况，并使工作负载的不同组件之间的事件和指标相关联。您可以根据日志中包含的信息，从工作负载组件中创建见解，而这些信息通常仅在指标中不可用。通过大规模查询日志，您可以更快地确定失败原因。

在未建立这些最佳实践的情况下暴露的风险等级：高

实施指导

确定与您的工作负载及其组件相关的遥测数据来源。这些数据不仅来自发布指标的组件，例如您的操作系统（OS）和应用程序运行时（例如 Java），还来自应用程序和云服务日志。例如，Web 服务器通常会记录每个请求以及诸如时间戳、处理延迟、用户 ID、远程 IP 地址、路径和查询字符串等详细信息。这些日志中的详细程度有助于您执行详细的查询，并生成原本可能无法得到的指标。

使用适当的工具和流程收集指标和日志。在 Amazon EC2 实例上运行的应用程序生成的日志可以由 Amazon CloudWatch 代理等代理收集，并发布到 Amazon CloudWatch Logs 等中央存储服务。AWS Lambda 和 Amazon Elastic Container Service 等 AWS 托管式计算服务会自动将日志发布到 CloudWatch Logs。为工作负载使用的 AWS 存储和处理服务启用日志收集，如 Amazon CloudFront、Amazon S3、弹性负载均衡和 Amazon API Gateway。

使用维度 丰富遥测数据，维度有助于您更清楚地看到行为规律，并将相关问题隔离到相关组件组中。添加后，您可以更详细地观察组件行为，检测相关的故障，并采取适当的补救措施。有用维度的示例包括可用区、EC2 实例 ID 和容器任务或容器组（pod）ID。

收集指标和日志后，您可以编写查询并从中生成聚合指标，从而为正常和异常行为提供有用的见解。例如，您可以使用 Amazon CloudWatch Logs Insights 从应用程序日志中得出自定义指标，使用 Amazon CloudWatch Metrics Insights 大规模查询您的指标，使用 Amazon CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合和汇总指标和日志，或者，如果您使用的是 AWS Lambda 函数，则可以使用 Amazon CloudWatch Lambda 洞察。要创建聚合错误率指标，可以在每次在组件日志中发现错误响应或消息时递增计数器，或者计算现有错误率指标的聚合值。可以使用这些数据来生成显示尾部行为 的直方图，例如性能最差的请求或进程。还可以使用 CloudWatch Logs anomaly detection 等解决方案实时扫描这些数据，来发现异常规律。这些见解可以放在控制面板上，以便根据您的需求和偏好进行整理。

查询日志有助于您了解工作负载组件如何处理特定的请求，并揭示影响工作负载韧性的请求规律或其它上下文。根据您对应用程序和其它组件行为的了解，提前研究和准备查询可能很有用，这样您就可以更轻松地根据需要运行它们。例如，使用 CloudWatch Logs Insights，您能够以交互方式搜索和分析存储在 CloudWatch Logs 中的日志数据。还可以使用 Amazon Athena 查询来自多个来源（包括许多 AWS 服务）的日志，数据量可达 PB 级。

在定义日志保留策略时，请考虑历史日志的价值。历史日志有助于确定工作负载性能的长期使用情况和行为规律、回归以及改进领域。永久删除的日志以后无法分析。然而，历史日志的价值往往会随着时间推移而减少。选择的策略应能够适当平衡您的需求，并符合您可能需要遵守的任何法律或合同要求。

实施步骤

1. 为您的可观测性数据选择收集、存储、分析和显示机制。
2. 在工作负载的适当组件上安装和配置指标和日志收集器（例如，在 Amazon EC2 实例上和边车容器中）。将这些收集器配置为在意外停止时自动重新启动。为收集器启用磁盘或内存缓冲，这样，临时发布失败就不会影响应用程序或导致数据丢失。
3. 在您用作工作负载一部分的 AWS 服务上启用日志记录，并在需要时将这些日志转发到您选择的存储服务。有关详细说明，请参阅相应服务的用户或开发人员指南。
4. 定义与基于遥测数据的工作负载相关的操作指标。这些指标可能基于从工作负载组件发出的直接指标，其中可能包括与业务 KPI 相关的指标，也可能基于聚合计算的结果，例如总和、比率、百分位数或直方图。使用日志分析器计算这些指标，并根据需要将其放在控制面板上。
5. 根据需要准备相应的日志查询，来分析工作负载组件、请求或事务行为。
6. 为组件日志定义并启用日志保留策略。当日志的时间超过策略允许的时间时，定期删除日志。

资源

相关最佳实践：

- REL06-BP01 为工作负载监控全部组件（生成）
- REL06-BP03 发送通知（实时处理和报警）
- REL06-BP04 自动响应（实时处理和警报）
- REL06-BP05 分析日志
- REL06-BP06 定期审核监控范围和指标
- REL06-BP07 对系统中的请求进行端到端跟踪监控

相关文档：

- 说明 Amazon CloudWatch 的工作原理
- Amazon Managed Prometheus
- Amazon Managed Grafana
- Analyzing log data with CloudWatch Logs Insights
- Amazon CloudWatch Lambda 洞察
- Amazon CloudWatch Container Insights
- 使用 CloudWatch Metrics Insights 查询您的指标
- 适用于 OpenTelemetry 的 AWS Distro
- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- Searching and Filtering Log Data
- Sending Logs Directly to Amazon S3
- Amazon Builders' Library：检测分布式系统的运营可见性

相关讲习会：

- One Observability 讲习会

相关工具：

- AWS Distro for OpenTelemetry (GitHub)

REL06-BP03 发送通知（实时处理和报警）

当组织检测到潜在问题时，会向相应的人员和系统发送实时通知和警报，以便快速有效地处理这些问题。

期望结果：通过根据服务和应用程序指标配置相关警报，可对运维事件做出快速响应。当超出警报阈值时，相应的人员和系统会收到通知，以便解决潜在的问题。

常见反模式：

- 配置的警报阈值过高，导致无法发送重要通知。
- 配置的警报阈值过低，导致通知过多，而重要警报无法得到处理。
- 使用情况发生变化时不更新警报及其阈值。
- 对于最好通过自动操作来处理的警报，向人员发送通知而不是生成自动操作，导致发送的通知过多。

建立此最佳实践的好处：通过向相应的人员和系统发送实时通知和警报，可以及早发现问题并快速处理运维方面的意外事件。

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

工作负载应配备实时处理和报警功能，从而更及时地检测到可能影响应用程序可用性的问题，并充当自动响应的触发器。组织可以通过使用定义的指标创建警报来执行实时处理和报警，以便在发生重大事件或指标超过阈值时收到通知。

Amazon CloudWatch 有助于您使用基于静态阈值、异常检测和其他标准的 CloudWatch 警报创建指标和复合警报。有关可以使用 CloudWatch 配置的警报类型的更多详细信息，请参阅 CloudWatch 文档的警报部分。

您可以使用 CloudWatch 控制面板为团队自定义 AWS 资源指标和警报的视图。通过 CloudWatch 控制台中的可自定义主页，您可以在单一视图中监控多个区域的资源。

警报可执行一项或多项操作，例如向 Amazon SNS 主题发送通知、执行 Amazon EC2 操作或 Amazon EC2 Auto Scaling 操作，或在 AWS Systems Manager 中创建 OpsItem 或事件。

Amazon CloudWatch 使用 Amazon SNS 在警报状态发生变化时发送通知，提供从发布者（生产者）到订阅用户（使用者）的信息传递。要了解有关设置 Amazon SNS 通知的更多信息，请参阅 Configuring Amazon SNS。

每当 CloudWatch 警报被创建、更新、删除或警报状态发生变化时，CloudWatch 都会发送 EventBridge 事件。您可以将 EventBridge 与这些事件结合使用来创建执行操作的规则，例如，每当警报状态发生变化时通知您，或者使用 Systems Manager 自动化在账户中自动触发事件。

随时了解 AWS Health 的最新信息。AWS Health 是有关 AWS Cloud资源运行状况的权威信息来源。使用 AWS Health 可获取任何已确认的服务事件的通知，以便您可以快速采取措施来减轻任何影响。通过 AWS 用户通知服务 创建要发送到电子邮件和聊天渠道且契合目标的 AWS Health 事件通知，并以编程方式通过 Amazon EventBridge 与监控和警报工具集成。如果您使用 AWS Organizations，则跨账户汇总 AWS Health 事件。

EventBridge 和 Amazon SNS 的使用时机

EventBridge 和 Amazon SNS 都可用于开发事件驱动型应用程序，您可以根据自己的具体需求进行选择。

如果想要构建能够对自己的应用程序、SaaS 应用程序和 AWS 服务中的事件做出反应的应用程序，建议使用 Amazon EventBridge。EventBridge 是唯一直接与第三方 SaaS 合作伙伴集成的基于事件的服务。EventBridge 还可以自动从 200 多项 AWS 服务中提取事件，无需开发者在其账户中创建任何资源。

EventBridge 使用已定义的基于 JSON 的事件结构，有助于您创建应用于整个事件主体的规则，以便选择要转发到目标的事件。EventBridge 目前支持 20 多项 AWS 服务作为目标，包括 AWS Lambda、Amazon SQS、Amazon SNS、Amazon Kinesis Data Streams 和 Amazon Data Firehose。

对于需要高扇出（数千或数百万个端点）的应用程序，建议使用 Amazon SNS。常见的一种模式是，客户将 Amazon SNS 用作规则的目标，来筛选所需的事件并扇出到多个端点。

消息是非结构化的，可以采用任意格式。Amazon SNS 支持将消息转发到六种不同类型的目标，包括 Lambda、Amazon SQS、HTTP/S 端点、短信、移动推送和电子邮件。Amazon SNS 典型延迟不超过 30 毫秒。有许多 AWS 服务通过配置服务来发送 Amazon SNS 消息（超过 30 项服务，包括 Amazon EC2、Amazon S3 和 Amazon RDS）。

实施步骤

1. 使用 Amazon CloudWatch 警报创建警报。
    a. 指标警报可监控单个 CloudWatch 指标或依赖于 CloudWatch 指标的表达式。这种警报会根据在若干时间间隔内，指标或表达式的值与阈值的比较结果，启动一项或多项操作。操作可以是向 Amazon SNS 主题发送通知、执行 Amazon EC2 操作或 Amazon EC2 Auto Scaling 操作，或在 AWS Systems Manager 中创建 OpsItem 或事件。

   b. 复合警报由一个规则表达式组成，该规则表达式考虑了您创建的其他警报的警报条件。只有满足所有规则条件，复合警报才会进入警报状态。复合警报的规则表达式中指定的警报可以包括指标警报和其他复合警报。复合警报可以在改变状态时发送 Amazon SNS 通知，并且可以在进入警报状态时创建 Systems Manager OpsItems 或事件，但无法执行 Amazon EC2 Auto Scaling 操作。

2. 设置 Amazon SNS 通知。创建 CloudWatch 警报时，可以包括 Amazon SNS 主题，以便在警报状态发生变化时发送通知。
3. 在 EventBridge 中创建与指定的 CloudWatch 警报相匹配的规则。每条规则都支持多个目标，包括 Lambda 函数。例如，您可以定义一个会在可用磁盘空间不足时启动的警报，通过 EventBridge 规则触发 Lambda 函数来清理空间。有关 EventBridge 目标的更多详细信息，请参阅 EventBridge targets。

资源

相关的 Well-Architected 最佳实践：

- REL06-BP01 为工作负载监控全部组件（生成）
- REL06-BP02 定义与计算指标（聚合）
- REL12-BP01 使用行动手册调查故障

相关文档：

- Amazon CloudWatch
- CloudWatch Logs Insights
- 使用 Amazon CloudWatch 告警
- Using Amazon CloudWatch dashboards
- 使用 Amazon CloudWatch 指标
- 设置 Amazon SNS 通知
- CloudWatch 异常检测
- CloudWatch Logs data protection
- Amazon EventBridge
- Amazon Simple Notification Service

相关视频：

- re:Invent 2022 observability 视频
- AWS re:Invent 2022 – Observability best practices at Amazon

相关示例：

- One Observability 讲习会
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

REL06-BP04 自动响应（实时处理和警报）

检测到事件后，利用自动化功能执行操作；例如，更换故障组件。

实施警报的自动实时处理，以便系统可以快速采取纠正措施，并在触发警报时尝试防止故障或服务降级。警报的自动响应可能包括更换故障组件，调整计算容量，将流量重定向到运行状况良好的主机、可用区或其他区域，以及通知操作员。

期望结果：识别实时警报，并设置警报的自动处理，以便调用适当措施来维护服务级别目标和服务水平协议（SLA）。自动处理的范围可以是单个组件的自我修复活动，也可以是全站点的失效转移。

常见反模式：

- 没有明确的关键实时警报的清单或目录。
- 关键警报没有自动响应（例如，当计算资源即将耗尽时自动进行扩展）。
- 警报响应操作相互矛盾。
- 操作员在收到警报通知时没有任何标准操作程序（SOP）可以遵循。
- 不监控配置更改，因为未检测到的配置更改可能会导致工作负载停机。
- 没有撤消意外配置更改的策略。

建立此最佳实践的好处：自动处理警报可以提高系统的韧性。系统会自动采取纠正措施，从而减少手动操作，而手动操作往往是容易出错的人工干预。工作负载的运行符合可用性目标，并减少服务中断。

在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

为了有效地管理警报并自动进行响应，请根据警报的严重程度和影响对警报进行分类，记录响应程序，并在对任务进行评级之前制定好响应计划。

确定需要特定操作的任务（通常在运行手册中详细说明），并检查所有运行手册和行动手册，判断哪些任务可以自动执行。如果操作可以定义，通常就可以实现自动化。如果操作无法自动化，请在 SOP 中记录手动步骤并对操作员进行培训。不断挑战手动流程，寻找自动化机会，以便制定和维护自动响应警报的计划。

实施步骤

1. 创建警报清单：要获取所有警报的列表，您可以使用 Amazon CloudWatch 命令 `describe-alarms` 来使用 AWS CLI。根据设置的警报数量，您可能需要使用分页来检索每个呼叫的警报子集，或者也可以使用 AWS SDK 通过 API 调用获取警报。
2. 记录所有警报操作：更新包含所有警报及其操作的运行手册，无论它们是手动还是自动的。AWS Systems Manager 提供预定义的运行手册。有关运行手册的更多信息，请参阅使用运行手册。有关如何查看运行手册内容的详细信息，请参阅 View runbook content。
3. 设置和管理警报操作：对于任何需要操作的警报，请使用 CloudWatch SDK 指定自动操作。例如，您可以通过创建和启用警报操作或禁用警报操作，根据 CloudWatch 警报自动更改 Amazon EC2 实例的状态。

   您还可以使用 Amazon EventBridge 自动响应系统事件，例如应用程序可用性问题或资源更改。您可以创建规则来指示要关注的事件，以及在事件匹配规则时要执行的操作。可以自动启动的操作包括调用 AWS Lambda 函数、调用 Amazon EC2 Run Command、将事件中继到 Amazon Kinesis Data Streams 以及查看使用 EventBridge 自动执行 Amazon EC2。
4. 标准操作程序（SOP）：根据应用程序组件，AWS Resilience Hub 会推荐多个 SOP 模板。您可以使用这些 SOP 来记录在出现警报时操作员应遵循的所有流程。您还可以根据韧性监测中心的建议构造 SOP，前提是有一个具有相关韧性策略的 Resilience Hub 应用程序，以及针对该应用程序的历史韧性评测。针对 SOP 的建议由韧性评测生成。

   韧性监测中心与 Systems Manager 结合使用，通过提供大量可用作这些 SOP 基础的 SSM 文档，自动执行 SOP 的步骤。例如，韧性监测中心可能会根据现有的 SSM 自动化文档推荐用于添加磁盘空间的 SOP。
5. 使用 Amazon DevOps Guru 执行自动操作：可以使用 Amazon DevOps Guru 自动监控应用程序资源的异常行为并提供针对性的建议，缩短识别问题和进行修复所需的时间。借助 DevOps Guru，您可以近乎实时地监控来自多个来源的运营数据流，包括 Amazon CloudWatch 指标、AWS Config、AWS CloudFormation 和 AWS X-Ray。您还可以使用 DevOps Guru 在 OpsCenter 中自动创建 OpsItems，并将事件发送到 EventBridge 实现更多自动化操作。

资源

相关最佳实践：

- REL06-BP01 为工作负载监控全部组件（生成）
- REL06-BP02 定义与计算指标（聚合）
- REL06-BP03 发送通知（实时处理和报警）
- REL08-BP01 对部署等标准活动使用运行手册

相关文档：

- AWS Systems Manager 自动化
- Creating an EventBridge Rule That Triggers on an Event from an AWS Resource
- One Observability 讲习会
- Amazon Builders' Library：检测分布式系统的运营可见性
- What is Amazon DevOps Guru?
- 使用自动化文档（行动手册）

相关视频：

- AWS re:Invent 2022 – Observability best practices at Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction to AWS Resilience Hub
- Create Custom Ticket Systems for Amazon DevOps Guru Notifications
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru

相关示例：

- Amazon CloudWatch and Systems Manager 讲习会

REL06-BP05 分析日志

收集日志文件和指标历史记录并加以分析，获得更全面的趋势和工作负载见解。

Amazon CloudWatch Logs Insights 支持简单但强大的查询语言，可用于分析日志数据。Amazon CloudWatch Logs 还支持订阅，允许数据无缝流动到 Amazon S3（可在其中使用数据）或 Amazon Athena（可在其中查询数据）。该服务支持查询多种格式。请参阅《Amazon Athena 用户指南》，了解支持的 SerDes 和数据格式。针对大型日志文件集的分析，您可以运行 Amazon EMR 集群以执行 PB 级分析。

AWS 合作伙伴和第三方提供了许多用于聚合、处理、存储和分析的工具。这些工具包括 New Relic、Splunk、Loggly、Logstash、CloudHealth 和 Nagios。但是，系统和应用程序日志之外的生成对于每个云提供商，甚至每个服务来说都是独一无二的。

监控过程中常常被忽视的部分是数据管理。您需要确定数据监控的保留要求，然后相应地应用生命周期策略。Amazon S3 支持 S3 存储桶级别的生命周期管理。此生命周期管理可以通过不同的方式应用到存储桶中的不同路径。您可以在生命周期临近结束时，将数据转移到 Amazon S3 Glacier 进行长期存储，然后在保留期结束后让它们过期。S3 Intelligent-Tiering 存储类旨在通过将数据自动移动到最具成本效益的访问层来优化成本，却不会对性能或运营开销产生影响。

在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

- 您可以使用 CloudWatch Logs Insights，通过交互方式搜索并分析 Amazon CloudWatch Logs 中的日志数据。
  - Analyzing Log Data with CloudWatch Logs Insights
  - Amazon CloudWatch Logs Insights Sample Queries
- 使用 Amazon CloudWatch Logs 将日志发送到 Amazon S3 以供使用，或发送到 Amazon Athena 以查询数据。
  - 如何使用 Athena 分析我的 Amazon S3 服务器访问日志？
    - 为服务器访问日志存储桶创建 S3 生命周期策略。配置生命周期策略以定期删除日志文件。这样做可以减少 Athena 为每个查询分析的数据量。
      - 如何为 S3 存储桶创建生命周期策略？

资源

相关文档：

- Amazon CloudWatch Logs Insights Sample Queries
- Analyzing Log Data with CloudWatch Logs Insights

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- 如何为 S3 存储桶创建生命周期策略？
- 如何使用 Athena 分析我的 Amazon S3 服务器访问日志？
- One Observability 讲习会
- Amazon Builders' Library：检测分布式系统的运营可见性

REL06-BP06 定期审核监控范围和指标

经常审核工作负载监控的实施情况，并根据工作负载及其架构的发展进行更新。定期审计监控有助于降低遗漏或忽视故障指标的风险，并进一步协助工作负载实现其可用性目标。

有效的监控以关键业务指标为基础，这些指标会随着业务优先级变化而变化。监控审核过程应强调服务级别指标（SLI），并纳入来自基础设施、应用程序、客户和用户的见解。

期望结果：您拥有有效的监控策略，该策略会定期进行审核和更新，并在发生任何重大事件或变更后进行更新。随着工作负载和业务需求发生变化，您可以验证关键的应用程序运行状况指标是否仍然相关。

常见反模式：

- 您仅收集默认指标。
- 您设置了监控策略，但从不对其进行审核。
- 部署重大更改时，您不讨论监控。
- 您信任过时的指标来确定工作负载运行状况。
- 由于指标和阈值过时，误报的警报让您的运营团队不堪重负。
- 您对未受监控的应用程序组件缺乏可观测性。
- 在监控中，您只关注低级技术指标，而不关注业务指标。

建立这种最佳实践的好处：当您定期审核监控时，您可以预测潜在的问题，并验证自己是否有能力发现这些问题。它还可让您找出之前的审核中可能错过的盲点，从而进一步提高您发现问题的能力。

在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 operational readiness review (ORR) 过程中审核监控指标和范围。按照一致的时间表定期进行运营准备情况审查，以评估您当前的工作负载与您配置的监控之间是否存在任何差距。定期开展运营性能审

查和知识共享，有助于增强运营团队提高绩效的能力。验证现有的警报阈值是否仍然适合，并检查运营团队是否收到误报的警报，或者是否未监控应用程序的应受监控的各个方面。

Resilience Analysis Framework 提供了有用的指导，有助于您驾驭整个过程。该框架的重点是确定潜在的故障模式，以及可用于减轻其影响的预防和纠正控制措施。这些知识有助于您确定要监控和发出警报的正确指标和事件。

实施步骤

1. 计划并执行工作负载控制面板常规检查。您可能对检查深度具有不同的安排。
2. 检查指标中的趋势。对比指标值与历史值，了解是否有趋势表明需要调查某些情况。这种情况的示例包括延迟增加、主要业务功能减少以及故障响应增加。
3. 检查指标中是否存在离群值和异常值，这些值可能会被平均值或中位数掩盖。查看时间范围内的最高值和最低值，并调查观测结果远超正常范围的原因。随着您持续消除这些原因，您可以收紧预期的指标范围，以提高工作负载性能的一致性。
4. 查找清晰的行为变化。指标数量或方向的立即更改可能表示应用程序已发生变化，或者出现了需要添加额外指标进行跟踪的外部因素。
5. 审核当前的监控策略是否仍然与应用程序保持相关。根据对先前事件的分析（或韧性分析框架），评测该应用程序中是否还有其它方面应纳入监控范围。
6. 查看您的真实用户监控（RUM）指标，以确定应用程序功能覆盖范围是否存在任何差距。
7. 审查您的更改管理流程。如有必要，请更新相关过程，来包括应在批准更改之前执行的监控分析步骤。
8. 实施监控审核，以此作为运营准备情况审查和错误更正流程的一部分。

资源

相关最佳实践

- REL06-BP01 为工作负载监控全部组件（生成）
- REL06-BP02 定义与计算指标（聚合）
- REL06-BP07 对系统中的请求进行端到端跟踪监控
- REL12-BP02 执行事后分析
- REL12-BP06 定期进行 GameDay 活动

相关文档：

- Why you should develop a correction of error (COE)
- Using Amazon CloudWatch Dashboards
- 构建控制面板以获取操作可见性
- Advanced Multi-AZ Resilience Patterns - Gray failures
- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- One Observability 讲习会
- Amazon Builders' Library：检测分布式系统的运营可见性
- Using Amazon CloudWatch Dashboards
- AWS Observability Best Practices
- Resilience Analysis Framework
- Resilience Analysis Framework - Observability
- Operational Readiness Review - ORR

REL06-BP07 对系统中的请求进行端到端跟踪监控

跟踪各个服务组件的请求处理情况，这样产品团队便能够更轻松地分析和调试问题并提高性能。

期望结果：针对所有组件全面跟踪工作负载，实现轻松调试，进而通过简化发现错误根本原因的过程，缩短错误的平均解决时间（MTTR）和延迟。采用端到端的跟踪方式，有助于更快地发现受影响的组件，并详细深入地了解造成错误或延迟的根本原因。

常见反模式：

- 只针对部分组件而不是全部组件进行跟踪。例如，如果不跟踪 AWS Lambda，团队可能无法清楚地了解高峰工作负载中冷启动所造成的延迟。
- Synthetics 金丝雀或真实用户监控（RUM）未配置跟踪功能。没有金丝雀或 RUM，跟踪分析中会忽略客户端交互遥测数据，这样得出的性能概况就不够完整。
- 混合工作负载包括云原生跟踪工具和第三方跟踪工具，但尚未采取措施来选择并完全集成单个跟踪解决方案。根据所选跟踪解决方案，应使用云原生跟踪 SDK 来检测非云原生组件，或者应将第三方工具配置为摄取云原生跟踪遥测数据。

建立此最佳实践的好处：当开发团队收到问题提醒时，能够查看系统组件交互情况的全貌，包括各个组件在日志记录、性能和故障方面的相关性。由于跟踪有助于直观且轻松地找出根本原因，因此调查根本原因所花费的时间得以减少。在解决问题时，团队如果能详细了解组件的交互情况，就可以更快地做出

更好的决策。分析系统跟踪数据有助于改进多种决策，例如何时调用灾难恢复（DR）失效转移，或者在何处实施自我修复策略最合适等，最终势必能够提高客户对服务的满意度。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

团队在运行分布式应用程序时，能够借助跟踪工具来建立关联标识符、收集请求跟踪数据，以及构建互联组件的服务地图。请求跟踪中应该涵盖所有应用程序组件，包括服务客户端、中间件网关和事件总线、计算组件以及存储（包括键/值存储和数据库）。在端到端跟踪配置中，纳入 Synthetics 金丝雀和真实用户监控来衡量远程客户端交互情况和延迟，这样您就能够根据服务水平协议和目标准确地评估系统性能。

您可以使用 AWS X-Ray 和 Amazon CloudWatch 应用程序监控检测服务，在请求通过应用程序时提供请求的完整视图。X-Ray 会收集应用程序遥测，有助于跨有效负载、函数、跟踪、服务、API 对其进行可视化和筛选，并且可以通过无代码或低代码的方式系统组件启用。CloudWatch 应用程序监控包括 ServiceLens，可将跟踪与指标、日志和警报集成。CloudWatch 应用程序监控还包括用于监控端点和 API 的 Synthetics，以及用于检测 Web 应用程序客户端的真实用户监控。

实施步骤

- 在所有支持的本机服务上使用 AWS X-Ray，例如 Amazon S3、AWS Lambda 和 Amazon API Gateway。这些 AWS 服务可使用基础设施即代码、AWS SDK 或 AWS Management Console 来启用 X-Ray。
- 检测应用程序（适用于 OpenTelemetry 的 AWS Distro 和 X-Ray）或第三方收集代理。
- 查看《AWS X-Ray Developer Guide》，了解编程语言特定的实施。这些文档部分详细介绍了如何检测 HTTP 请求、SQL 查询和应用程序编程语言特定的其他进程。
- 使用适用于 Amazon CloudWatch Synthetics 金丝雀 和 Amazon CloudWatch RUM 的 X-Ray 追踪，对最终用户客户端通过下游 AWS 基础设施的请求路径进行分析。
- 根据资源运行状况和金丝雀遥测数据来配置 CloudWatch 指标和警报，这样团队就能够快速收到问题提醒，然后使用 ServiceLens 深入探究跟踪数据和服务地图。
- 如果使用第三方工具作为主要的追踪解决方案，则将 X-Ray 与 Datadog、New Relic 或 Dynatrace 等第三方追踪工具集成。

资源

相关最佳实践：

- REL06-BP01 为工作负载监控全部组件（生成）

- REL11-BP01 监控工作负载的所有组件以检测故障

相关文档：

- 什么是 AWS X-Ray ？
- Amazon CloudWatch：应用程序监控
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- Amazon Builders' Library：检测分布式系统的运营可见性
- Integrating AWS X-Ray with other AWS services
- AWS Distro for OpenTelemetry and AWS X-Ray
- Amazon CloudWatch：使用综合监控
- Amazon CloudWatch：使用 CloudWatch RUM
- Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm
- Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS

相关示例：

- One Observability 讲习会

相关视频：

- AWS re:Invent 2022 – How to monitor applications across multiple accounts
- How to Monitor your AWS Applications

相关工具：

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

REL 7. 如何设计工作负载以适应需求变化？

可扩展工作负载提供了自动添加或删除资源的弹性，因此资源在任何给定时间点都非常符合当前需求。

最佳实践

- REL07-BP01 在获取或扩展资源时利用自动化
- REL07-BP02 在检测到对工作负载的破坏时获取资源
- REL07-BP03 在检测到某个工作负载需要更多资源时获取资源
- REL07-BP04 对工作负载进行负载测试

REL07-BP01 在获取或扩展资源时利用自动化

云端可靠性的基石是基础设施和资源的程序化定义、预置和管理。自动化有助于您简化资源预置，促进一致和安全的部署，并在整个基础设施中扩展资源。

期望结果：管理基础设施即代码（IaC）。您可以在版本控制系统（VCS）中定义和维护基础设施代码。您可以将预置 AWS 资源的过程委托给自动机制，并利用托管式服务，例如应用程序负载均衡器（ALB）、网络负载均衡器（NLB）和自动扩缩组。您可以使用持续集成/持续交付（CI/CD）管道来预置资源，以便代码更改自动启动资源更新，包括对自动扩缩配置的更新。

常见反模式：

- 您使用命令行或在 AWS Management Console中（也称为单击操作）手动部署资源。
- 您将应用程序组件或资源紧密耦合，从而创建了不灵活的架构。
- 您实施的扩展策略不灵活，无法适应不断变化的业务需求、流量规律或新的资源类型。
- 您手动估算容量来满足预期需求。

建立这种最佳实践的好处：基础设施即代码（IaC）支持以编程方式定义基础设施。这有助于您在与应用程序更改相同的软件开发生命周期中管理基础设施更改，从而提高一致性和可重复性，并降低手动、易出错任务的风险。通过使用自动交付管道实施 IaC，可以进一步简化预置和更新资源的流程。您无需手动干预，即可可靠、高效地部署基础设施更新。在扩展资源以满足不断变化的需求时，这种敏捷性尤其重要。

您可以结合使用 IaC 和交付管道来实现动态、自动的资源扩展。通过监控关键指标和应用预定义的扩展策略，自动扩缩可以根据需要自动预置或取消预置资源，从而提高性能和成本效益。这样可以减少因应用程序或工作负载要求发生变化而导致手动错误或延迟的可能性。

IaC、自动交付管道和自动扩缩相结合，有助于组织放心地预置、更新和扩展其环境。这种自动化对于维护响应迅速、富有韧性和高效管理的云基础设施至关重要。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要为 AWS 架构的 CI/CD 管道和基础设施即代码（IaC）设置自动化，请选择版本控制系统（例如 Git）来存储 IaC 模板和配置。这些模板可以使用诸如 AWS CloudFormation 之类的工具编写。首先，请在这些模板中定义基础设施组件（例如 AWS VPC、Amazon EC2 Auto Scaling 组和 Amazon RDS 数据库）。

接下来，将这些 IaC 模板与 CI/CD 管道集成，以实现部署过程的自动化。AWS CodePipeline 提供了无缝的 AWS 原生解决方案，您也可以使用其它第三方 CI/CD 解决方案。创建一个在版本控制存储库发生更改时激活的管道。将管道配置为包括以下各个阶段：整理和验证 IaC 模板、将基础设施部署到暂存环境、运行自动测试以及最终部署到生产环境。必要时纳入审批步骤，以保持对变更的控制。这种自动化的管道不仅加快了部署速度，而且促进了跨环境的一致性和可靠性。

在 IaC 中配置诸如 Amazon EC2 实例、Amazon ECS 任务和数据库副本等资源的自动扩缩，以根据需要提供自动横向扩展和横向缩减。这种方法通过根据需求动态调整资源，来增强应用程序可用性和性能并优化成本。有关支持的资源列表，请参阅 Amazon EC2 Auto Scaling 和 AWS Auto Scaling。

实施步骤

1. 创建并使用源代码存储库，来存储控制基础设施配置的代码。提交对此存储库的更改，来反映您要进行的任何持续更改。
2. 选择基础设施即代码解决方案（例如 AWS CloudFormation），以使您的基础设施保持最新状态并检测与预期状态的不一致性（偏差）。
3. 将 IaC 平台与 CI/CD 管道集成来实现部署自动化。
4. 确定并收集用于自动扩缩资源的适当指标。
5. 使用适用于工作负载组件的横向扩展和横向缩减策略，来配置资源的自动扩缩。考虑使用计划的扩展来实现可预测的使用规律。
6. 监控部署来检测故障和回归。在 CI/CD 平台中实施回滚机制，以便在必要时还原更改。

资源

相关文档：

- AWS Auto Scaling: How Scaling Plans Work
- AWS Marketplace: products that can be used with auto scaling
- 使用 DynamoDB Auto Scaling 自动管理吞吐能力

- Using a load balancer with an Auto Scaling group
- What Is AWS Global Accelerator?
- What Is Amazon EC2 Auto Scaling?
- 什么是 AWS Auto Scaling ？
- What is Amazon CloudFront?
- What is Amazon Route 53?
- 什么是 Elastic Load Balancing？
- 什么是网络负载均衡器？
- 什么是应用程序负载均衡器？
- Integrating Jenkins with AWS CodeBuild and AWS CodeDeploy
- Creating a four stage pipeline with AWS CodePipeline

相关视频：

- Back to Basics: Deploy Your Code to Amazon EC2
- AWS Supports You | Starting Your Infrastructure as Code Solution Using AWS CloudFormation Templates
- Streamline Your Software Release Process Using AWS CodePipeline
- Monitor AWS Resources Using Amazon CloudWatch Dashboards
- Create Cross Account & Cross Region CloudWatch Dashboards | Amazon Web Services

REL07-BP02 在检测到对工作负载的破坏时获取资源

如果可用性受到影响，在必要时被动扩展资源，从而还原工作负载的可用性。

首先，您必须配置运行状况检查和关于此类检查的标准，表示在什么时候可用性会因缺少资源而受到影响。然后，通知适当的人员手动扩展资源，或启动自动化以对其进行自动扩展。

可以根据工作负载手动调整资源规模（例如，通过 AWS Management Console 或 AWS CLI 更改自动扩缩组中 EC2 实例的数量，或者修改 DynamoDB 表的吞吐量）。但是，应尽量采用自动化技术（请参阅在获取或扩展资源时利用自动化）。

期望结果：在检测到故障或客户体验降级时，启动扩缩活动（自动或手动）来恢复可用性。

在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

对工作负载中的所有组件实施可观测性和监控，以监控客户体验并检测故障。定义用于扩展所需资源的手动或自动程序。有关更多信息，请参阅 REL11-BP01 监控工作负载的所有组件以检测故障。

实施步骤

- 定义扩展所需资源的手动或自动程序。
  - 扩缩程序取决于工作负载中不同组件的设计方式。
  - 扩缩程序还因所使用的底层技术而异。
    - 使用 AWS Auto Scaling 的组件可以使用扩缩计划来配置一组用于扩展资源的指令。如果使用 AWS CloudFormation 或向 AWS 资源添加标签，则可以根据应用程序为不同的资源集设置扩缩计划。Auto Scaling 提供了针对每个资源的定制扩展策略建议。创建扩缩计划后，Auto Scaling 结合了动态扩缩和预测性扩缩方法来支持扩缩策略。有关更多详细信息，请参阅 How scaling plans work。
    - Amazon EC2 Auto Scaling 可验证您具有正确数量的 Amazon EC2 实例来处理应用程序负载。您可创建 EC2 实例的集合，称为自动扩缩组。您可以指定每个自动扩缩组中的最小和最大实例数，Amazon EC2 Auto Scaling 会确保您的组永远不会低于或超过这些限制。有关更多详细信息，请参阅 What is Amazon EC2 Auto Scaling?
    - Amazon DynamoDB Auto Scaling 会根据实际的流量模式，使用 Application Auto Scaling 服务代表您动态调整预置的吞吐容量。这样表或全局二级索引可以增加预置读取和写入容量处理突增流量，不会受到限制。有关更多详细信息，请参阅使用 DynamoDB Auto Scaling 自动管理吞吐能力。

资源

相关最佳实践：

- REL07-BP01 在获取或扩展资源时利用自动化
- REL11-BP01 监控工作负载的所有组件以检测故障

相关文档：

- AWS Auto Scaling: How Scaling Plans Work
- 使用 DynamoDB Auto Scaling 自动管理吞吐能力

- What Is Amazon EC2 Auto Scaling?

REL07-BP03 在检测到某个工作负载需要更多资源时获取资源

云计算最有价值的功能之一是能够动态预置资源。

在传统的本地计算环境中，您必须提前确定和预置足够的容量来满足峰值需求。这的确是个问题，因为很昂贵，如果您低估了工作负载的峰值容量需求，则会对可用性构成风险。

在云中，您不必这样做。相反，您可以根据需要预置计算、数据库和其它资源容量，以满足当前和预测的需求。Amazon EC2 Auto Scaling 和 Application Auto Scaling 等自动化解决方案可以根据您指定的指标在线为您提供资源。这可以使扩展过程变得更加轻松和可预测，还可以通过确保始终有足够的可用资源来显著提高工作负载的可靠性。

期望结果：您可以配置计算资源和其它资源的自动扩缩来满足需求。您在扩展策略中提供了充足的余量，可以在额外资源上线时应对流量爆增。

常见反模式：

- 您预置固定数量的可扩展资源。
- 您选择的扩展指标与实际需求不相关。
- 您未能在扩展计划中提供足够的余量来应对需求爆增。
- 您的扩展策略添加容量的时间过晚，这会导致容量耗尽和服务降级，同时使额外的资源上线。
- 您未能正确地配置最小和最大资源计数，这会导致扩展失败。

建立此最佳实践的好处：拥有足够的资源来满足当前需求，对于提供工作负载的高可用性和遵守您定义的服务级别目标（SLO）至关重要。自动扩缩可让您提供工作负载所需的适量计算资源、数据库资源和其它资源，以满足当前和预测的需求。您无需确定峰值容量需求，也无需静态分配资源来满足此需求。相反，随着需求增长，您可以分配更多资源来满足需求，在需求下降后，您可以停用资源以降低成本。

在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

首先，确定工作负载组件是否适合自动扩缩。这些组件之所以称为可水平扩展，是因为它们提供的资源相同，行为也相同。水平可扩展组件的示例包括以类似方式配置的 EC2 实例、Amazon Elastic Container Service（ECS）任务以及在 Amazon Elastic Kubernetes Service（EKS）上运行的容器组（pod）。这些计算资源通常位于负载均衡器后面，称为副本。

其它复制的资源可能包括数据库只读副本、Amazon DynamoDB 表以及 Amazon ElastiCache（Redis OSS）集群。有关支持的资源的完整列表，请参阅 AWS services that you can use with Application Auto Scaling。

对于基于容器的架构，可能需要通过两种不同的方式进行扩展。首先，您可能需要扩展可提供水平可扩展服务的容器。其次，您可能需要扩展计算资源，以便为新容器腾出空间。每个层都有不同的自动扩缩机制。要扩展 ECS 任务，可以使用 Application Auto Scaling。要扩展 Kubernetes 容器组（pod），可以使用 Pod 水平自动扩缩控制器（HPA）或 Kubernetes Event-driven Autoscaling (KEDA)。要扩展计算资源，对于 ECS，可以使用容量提供程序，或者对于 Kubernetes，可以使用 Karpenter 或集群自动扩缩器。

接下来，选择您将如何执行自动扩缩。有三个主要选项：基于指标的扩展、计划扩展和预测式扩展。

基于指标的扩展

基于指标的扩展根据一个或多个扩展指标 的值来预置资源。扩展指标是与工作负载的需求相对应的指标。确定适当扩展指标的一个好方法是在非生产环境中执行负载测试。在负载测试期间，请将可扩展资源的数量保持为固定，并缓慢增加需求（例如，吞吐量、并发性或模拟用户数）。然后，寻找随着需求增长而增加（或减少）的指标，以及相反，即随着需求下降而减少（或增加）的指标。典型的扩展指标包括 CPU 利用率、工作队列深度（例如 Amazon SQS 队列）、活跃用户数量和网络吞吐量。

> **Note**
>
> AWS 观察到，对于大多数应用程序，内存利用率会随着应用程序预热而增加，然后达到稳定的值。当需求减少时，内存利用率通常保持较高水平，而不是并行下降。因为内存利用率与两个方向的需求不符（即随需求而增长和下降），因此在选择此指标进行自动扩缩之前，请慎重考虑。

基于指标的扩展是一种潜在操作。利用率指标可能需要几分钟才能传播到自动扩缩机制，而这些机制通常要等到明确的需求增加信号后才会做出反应。然后，当自动扩缩器创建新资源时，这些资源可能需要更多时间才能全面投入使用。因此，重要的是不要将扩展指标目标设置为过于接近完全利用率（例如，90% 的 CPU 利用率）。这样做有可能在额外容量上线之前耗尽现有资源容量。为了获得最佳可用性，典型的资源利用率目标可介于 50-70% 之间，具体取决于需求规律以及预置额外资源所需的时间。

计划扩展

计划扩展会根据日历或一天中的时间预置或移除资源。它通常用于需求可预测的工作负载，例如工作日工作时间或销售活动期间的峰值利用率。Amazon EC2 Auto Scaling 和 Application Auto Scaling 都支持计划扩展。KEDA 的 cron scaler 支持 Kubernetes 容器组（pod）的计划扩展。

预测式扩展

预测式扩展使用机器学习来根据预期的需求自动扩缩资源。预测式扩展分析您提供的利用率指标的历史值，并持续预测其未来值。然后，使用预测值来纵向扩展或缩减资源。Amazon EC2 Auto Scaling 可以执行预测式扩展。

实施步骤

1. 确定工作负载组件是否适合自动扩缩。
2. 确定哪种扩展机制最适合工作负载：基于指标的扩展、计划扩展或预测式扩展。
3. 为组件选择适当的自动扩缩机制。对于 Amazon EC2 实例，请使用 Amazon EC2 Auto Scaling。对于其它 AWS 服务，请使用 Application Auto Scaling。对于 Kubernetes 容器组（pod）（例如在 Amazon EKS 集群中运行的容器），可以考虑水平容器组（pod）自动扩缩器（HPA）或 Kubernetes 事件驱动的自动扩缩（KEDA）。对于 Kubernetes 或 EKS 节点，可以考虑 Karpenter 和集群自动扩缩器（CAS）。
4. 对于指标或计划扩展，请进行负载测试，以确定工作负载的相应扩展指标和目标值。对于计划扩展，请确定在您选择的日期和时间所需的资源数量。确定为应对预期的峰值流量所需的最大资源数量。
5. 根据上面收集的信息配置自动扩缩器。有关详细信息，请参阅自动扩缩服务的文档。验证最大和最小扩展限制是否配置正确。
6. 验证扩展配置是否按预期发挥作用。在非生产环境中执行负载测试，观察系统的反应，并根据需要进行调整。在生产环境中启用自动扩缩时，请配置适当的警报来向您通知任何意外行为。

资源

相关文档：

- What Is Amazon EC2 Auto Scaling?
- AWS Prescriptive Guidance: Load testing applications
- AWS Marketplace: products that can be used with auto scaling
- 使用 DynamoDB Auto Scaling 自动管理吞吐能力
- Predictive Scaling for EC2, Powered by Machine Learning
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Telling Stories About Little's Law

REL07-BP04 对工作负载进行负载测试

采用负载测试方法来衡量扩展活动能否满足工作负载要求。

持续开展负载测试，这一点很重要。负载测试用于发现工作负载的断点并测试工作负载的性能。利用 AWS，您可以轻松设置能够模拟生产工作负载规模的临时测试环境。在云中，您可以根据需要创建一套生产规模等级的测试环境，完成测试，然后停用资源。由于测试环境只需在运行时付费，您模拟真实环境的成本仅为本地测试成本的一小部分。

生产中的负载测试还应该被视为 GameDay 活动的一部分，因为在客户使用量降低的那几个小时内，在场的所有员工都忙于解读结果与处理任何出现的问题，生产系统承受着很大的压力。

常见反模式：

- 对与生产采用不同配置的部署执行负载测试。
- 仅对单个工作负载分段（而非整个工作负载）执行负载测试。
- 使用请求子集，而不是具有代表性的实际请求集执行负载测试。
- 对超出预期负载的较小安全系数执行负载测试。

建立此最佳实践的好处：您知道架构中哪些组件会在负载下失败，而且能够确定要监控哪些可指示您即将达到该负载的指标，从而及时解决问题，防止故障影响。

在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

- 执行负载测试，确定工作负载的哪些方面表明您必须添加或移除容量。负载测试应具有您在生产中接收的流量类似的代表性流量。增加负载，同时监视所有已检测指标，以便确定哪种指标指示何时必须添加或移除资源。
  - AWS 上的分布式负载测试：模拟数千个连接的用户
    - 确定请求组合。您可能拥有不同的请求组合，因此应当在确定流量组合时查看不同的时间范围。
    - 实施负载驱动程序。您可以使用自定义代码、开源或商用软件来实施负载驱动程序。
    - 最初使用小容量进行负载测试。通过将负载降低到较小容量（可能小到一个实例或容器），可能会有立竿见影的效果。
    - 针对更大的容量进行负载测试。分布式负载的效果会有所不同，因此您必须对尽量接近生产环境的目标进行测试。

资源

相关文档：

- AWS 上的分布式负载测试：模拟数千个连接的用户
- 加载测试应用程序

相关视频：

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing

REL 8. 您如何实施更改？

要部署新功能，必须对更改加以控制，以确保工作负载和操作环境正在运行已知的软件，并以可预测的方式进行修补和替换。如果这些更改不受控制，那么就很难预测这些更改的影响，也很难解决由此产生的问题。

最佳实践

- REL08-BP01 对部署等标准活动使用运行手册
- REL08-BP02 将功能测试作为部署的一部分进行集成
- REL08-BP03 将韧性测试作为部署的一部分进行集成
- REL08-BP04 使用不可变基础设施进行部署
- REL08-BP05 使用自动化功能部署更改

REL08-BP01 对部署等标准活动使用运行手册

运行手册是用来实现特定结果的预定义程序。使用运行手册执行标准活动，无论这些活动是手动还是自动执行。示例包括部署工作负载，对其进行修补或修改 DNS。

例如，实施流程以确保部署期间安全回滚。确保您可以为客户进行部署回滚而不会出现中断，这是保证服务可靠的关键。

针对运行手册程序，从一个有效的手动流程开始，用代码进行实施，并在适当的情况下调用流程自动运行。

即使是高度自动化的复杂工作负载，运行手册仍可用于开展 GameDay 活动或满足严格的报告和审核要求。

请注意，行动手册可用于对特定事件做出响应，运行手册则用来达成特定的结果。通常，运行手册适用于例行活动，而行动手册则用于对非例行事件做出响应。

常见反模式：

- 对生产中的配置执行计划外更改。
- 为加快部署速度而跳过计划中的步骤，导致部署失败。
- 在未测试反向更改的情况下做出更改。

建立此最佳实践的好处：有效的更改计划有助于成功执行更改，因为您知道所有受影响的系统。在测试环境中验证更改能够增强您的信心。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

- 通过在运行手册中记录程序，实现对为人熟知的事件的一致且及时的响应。
  - AWS Well-Architected Framework：概念：运行手册
- 使用基础设施即代码的原则定义基础设施。通过使用 AWS CloudFormation（或受信任的第三方）来定义基础设施，您可以使用版本控制软件管理并跟踪更改。
  - 使用 AWS CloudFormation（或受信任的第三方提供商）定义基础设施。
    - 什么是 AWS CloudFormation ？
  - 使用良好的软件设计原则创建单个解耦模板。
    - 确定实施的权限、模板和责任方。
      - 使用 AWS Identity and Access Management 控制访问权限
    - 使用基于 Git 等流行技术的托管源代码管理系统，来存储源代码和基础设施即代码（IaC）配置。

资源

相关文档：

- APN 合作伙伴：可帮助创建自动化部署解决方案的合作伙伴
- AWS Marketplace：可用于自动实施部署的产品
- AWS Well-Architected Framework：概念：运行手册
- 什么是 AWS CloudFormation ？

相关示例：

- 根据行动手册和运行手册自动完成操作

REL08-BP02 将功能测试作为部署的一部分进行集成

使用单元测试和集成测试等技术来验证所需功能。

单元测试是测试代码的最小功能单元来验证其行为的过程。集成测试旨在验证每个应用程序功能是否符合软件要求。虽然单元测试侧重于以隔离方式测试应用程序的一部分，但集成测试会考虑副作用（例如，通过变异操作更改数据所带来的影响）。无论是哪种情况，都应将测试集成到部署管道中，若未能满足成功条件，则相关管道会中止或回滚。这些测试在预生产环境中运行，该环境会在管道中的生产开始前被暂存。

如果这些测试作为构建和部署措施的一部分自动运行，则您可以获得最佳的结果。例如，使用 AWS CodePipeline，开发人员将更改提交到源代码存储库，而 CodePipeline 会自动在其中检测这些更改。构建应用程序，并运行单元测试。在单元测试获得通过之后，将构建的代码部署到暂存服务器来进行测试。CodePipeline 会从暂存服务器运行更多测试，例如集成或负载测试。在成功完成此类测试以后，CodePipeline 会将经过测试并获得批准的代码部署到生产实例。

期望结果：您使用自动化功能来执行单元测试和集成测试，以验证您的代码是否按预期运行。这些测试集成到部署过程中，而测试失败会中止部署。

常见反模式：

- 您在部署过程中忽略或绕过测试失败和测试计划，以加快部署时间表。
- 在部署管道之外手动执行测试。
- 您跳过自动化流程中的测试步骤，而采用手动应急工作流程。
- 您在与生产环境不太相似的环境中运行自动测试。
- 您构建的测试套件不够灵活，难以随应用程序发展来进行维护、更新或扩展。

建立这种最佳实践的好处：在部署过程中进行自动测试可以及早发现问题，从而降低发布到生产环境时出现错误或意外行为的风险。单元测试可验证代码是否按预期运行，以及 API 合约是否得到遵守。集成测试可验证系统是否按照指定的要求运行。这些类型的测试可验证诸如用户界面、API、数据库和源代码等组件是否按预期正常运行。

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

采用测试驱动型开发（TDD）方法编写软件，从而通过开发测试用例来指定和验证代码。首先，为每个函数创建测试用例。如果测试失败，则编写新的代码来通过测试。这种方法有助于您验证每个函数的预期结果。在将代码提交到源代码存储库之前，请运行单元测试并验证测试获得通过。

执行单元测试和集成测试，作为 CI/CD 管道的构建、测试和部署阶段的一部分。自动执行测试，并在准备好部署新版本的应用程序时自动启动测试。若未满足成功条件，则相关管道会中止或回滚。

如果应用程序是 Web 或移动应用程序，请在多个桌面浏览器或实际设备上执行自动的集成测试。这种方法对于验证移动应用程序在各种设备上的兼容性和功能性特别有用。

实施步骤

1. 在编写功能代码（测试驱动型开发 或 TDD）之前，先编写单元测试。制定代码指南，使编写和运行单元测试成为非功能性编码要求。
2. 创建一套自动化集成测试，其中涵盖已确定的可测试功能。这些测试应模拟用户互动并验证预期结果。
3. 创建运行集成测试所需的测试环境。这可能包括与生产环境非常相似的暂存环境或预生产环境。
4. 使用 AWS CodePipeline 控制台或 AWS Command Line Interface（CLI）设置源代码以及构建、测试和部署各个阶段。
5. 在代码构建和测试完毕后部署应用程序。AWS CodeDeploy 可以将其部署到您的暂存（测试）环境和生产环境。这些环境可能包括 Amazon EC2 实例、AWS Lambda 函数或本地服务器。应使用相同的部署机制将应用程序部署到所有环境。
6. 监控管道的进度以及每个阶段的状态。使用质量检查，根据测试的状态来阻止管道。此外，您可以接收有关任何管道阶段故障或管道完成的通知。
7. 持续监控测试结果，并查找规律、回归或需要更多关注的领域。使用这些信息来改进测试套件，确定应用程序中需要更稳健测试的领域，并优化部署过程。

资源

相关最佳实践：

- REL07-BP04 对工作负载进行负载测试
- REL08-BP03 将韧性测试作为部署的一部分进行集成
- REL12-BP04 使用混沌工程测试韧性

相关文档：

- AWS Prescriptive Guidance: Test automation
- 持续交付和持续集成
- Indicators for functional testing
- Monitoring pipelines
- Use AWS CodePipeline with AWS CodeBuild to test code and run builds
- AWS Device Farm

REL08-BP03 将韧性测试作为部署的一部分进行集成

集成韧性测试功能，通过在系统中特意引入故障来衡量系统在遇到破坏性情景时的功能。韧性测试不同于通常集成在部署周期中的单元和功能测试，因为它们侧重于识别系统中的意外故障。虽然在预生产环境中集成韧性测试是安全的，但您要设定一个目标，将这些测试作为 GameDay 活动的一部分在生产环境中实施。

期望结果：韧性测试有助于建立信心，确信系统能够承受生产环境中的性能降级。通过实验来找出可能导致故障的薄弱环节，这可以帮助您改进系统，从而自动有效地减少故障和性能降级的情况。

常见反模式：

- 部署流程中缺乏可观测性和监控能力
- 依靠人工来解决系统故障
- 糟糕的质量分析机制
- 只看到系统中的已知问题，缺少通过实验来发现未知问题的手段
- 识别故障，但没有解决
- 没有关于调查发现和运行手册的文档

建立最佳实践的好处：部署中集成的韧性测试有助于识别系统中的未知问题，以防因忽视这些问题导致生产中断。识别系统中的这些未知问题可以协助您记录调查发现，将测试集成到 CI/CD 流程中，并制定运行手册，从而通过高效、可重复的机制简化缓解措施。

在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

在系统部署中，可以集成的最常见的韧性测试形式是灾难恢复和混沌工程。

- 对于任何重大部署，都要包括对灾难恢复计划和标准操作程序（SOP）的更新。
- 将可靠性测试集成到自动部署管道中。诸如 AWS Resilience Hub 的服务可以集成到 CI/CD 管道中，用于建立持续的韧性评测，这些评测将作为每次部署的一部分自动进行评估。
- 在 AWS Resilience Hub 中定义应用程序。韧性评测会生成代码片段，帮助您以 AWS Systems Manager 文档的形式为应用程序创建恢复程序，并提供推荐的 Amazon CloudWatch 监控和警报列表。
- 更新灾难恢复计划和标准操作程序后，完成灾难恢复测试以验证它们是否有效。灾难恢复测试可帮助您确定，在事件发生后是否可以恢复系统并恢复正常运行。您可以模拟各种灾难恢复策略，确定计划是否足以满足您的正常运行时间要求。常见的灾难恢复策略包括备份和恢复、指示灯、冷备用、温备用、热备用和主动-主动模式，这些策略的成本和复杂性各不相同。在执行灾难恢复测试之前，建议先定义恢复时间目标（RTO）和恢复点目标（RPO），简化模拟策略的选择。AWS 提供 AWS Elastic Disaster Recovery 等灾难恢复工具，可帮助您开始规划和测试。
- 混沌工程实验会在系统中引入中断，例如网络中断和服务故障。通过模拟受控的故障，您可以发现系统的漏洞，同时控制注入故障的影响。与其他策略一样，使用诸如 AWS Fault Injection Service 的服务在非生产环境中运行受控故障模拟，以便在生产环境中部署之前增强信心。

资源

相关文档：

- Experiment with failure using resilience testing to build recovery preparedness
- Continually assessing application resilience with AWS Resilience Hub and AWS CodePipeline
- Disaster recovery (DR) architecture on AWS, part 1: Strategies for recovery in the cloud
- Verify the resilience of your workloads using Chaos Engineering
- 混沌工程原则
- Chaos Engineering 讲习会

相关视频：

- AWS re:Invent 2020: Testing Resilience using Chaos Engineering
- Improve Application Resilience with AWS Fault Injection Service
- Prepare & Protect Your Applications From Disruption With AWS Resilience Hub

REL08-BP04 使用不可变基础设施进行部署

不可变基础设施模式要求在生产工作负载上不会出现就地更新、安全补丁或配置更改。需要更改时，会在新的基础设施上构建架构，并将其部署到生产环境中。

遵循不可变基础设施部署策略，以提高工作负载部署的可靠性、一致性和可重复性。

期望结果：使用不可变基础设施，就禁止为了在工作负载中运行基础设施资源而进行就地修改。相反，当需要更改时，会将一组包含所有必要更改的新基础设施资源与现有资源并行部署。会自动验证此部署，如果成功，流量将逐渐转移到新的资源集。

此部署策略适用于软件更新、安全补丁、基础设施更改、配置更新和应用程序更新等。

常见反模式：

- 对正在运行的基础设施资源实施就地更改。

建立此最佳实践的好处：

- 提高环境间的一致性：由于环境间的基础设施资源没有差异，可以提高一致性并简化测试。
- 减小配置偏差：通过使用已知且版本受控的配置替换基础设施资源，基础设施被设置为已知经过测试的可信状态，避免配置偏差。
- 采用可靠的原子部署：部署要么成功完成，要么没有任何更改，从而提高部署过程的一致性和可靠性。
- 简化部署：由于无需支持升级，部署得到简化。升级即意味着新的部署。
- 采用快速回滚和恢复流程实现更安全的部署：由于之前运行的版本未发生更改，部署变得更安全。您可以在检测到错误时进行回滚。
- 增强安全态势：通过不允许更改基础设施，可以禁用远程访问机制（例如 SSH）。这样做可以减少攻击向量，改善组织的安全态势。

在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

自动化

在定义不可变基础设施部署策略时，建议尽可能使用自动化功能来提高可重复性，并最大限度地减少出现人为错误的可能性。有关更多详细信息，请参阅 REL08-BP05 使用自动化功能部署更改和自动实现无需干预的安全部署。

借助基础设施即代码（IaC），基础设施预置、编排和部署步骤以编程、描述性和声明性的方式进行定义，并存储在源代码控制系统中。利用基础设施即代码可以更轻松地自动化基础设施部署，并有助于实现基础设施的不可变性。

部署模式

当需要更改工作负载时，不可变基础设施部署策略要求部署一组新的基础设施资源，包括所有必要的更改。这组新资源必须遵循可最大限度地减少对用户影响的推出模式，这一点非常重要。此部署有两种主要策略：

金丝雀部署：将少量客户引导到新版本的做法，通常在单个服务实例（金丝雀）上运行。然后，您可以深入检查生成的任何行为更改或错误。如果遇到了严重问题，您可以将金丝雀中的流量删除，并将用户发回到以前的版本。如果部署成功，您可以继续以期望的速度进行部署，同时监控更改以便发现错误，直到所有部署完成。AWS CodeDeploy 的部署配置可以配置为允许金丝雀部署。

蓝绿部署：与金丝雀部署类似，只是会并行部署一整套应用程序。您可以在两个堆栈（蓝和绿）之间轮流部署。同样，您可以将流量发送到新版本中，如果发现部署中存在问题，可以对其进行故障恢复，然后送回旧版本中。通常来说，所有流量会被一次性切换，但您也可以通过 Amazon Route 53 的加权 DNS 路由功能向每个版本发送部分流量，加快采用新版本的速度。AWS CodeDeploy 和 AWS Elastic Beanstalk 的部署配置可以配置为允许蓝绿部署。

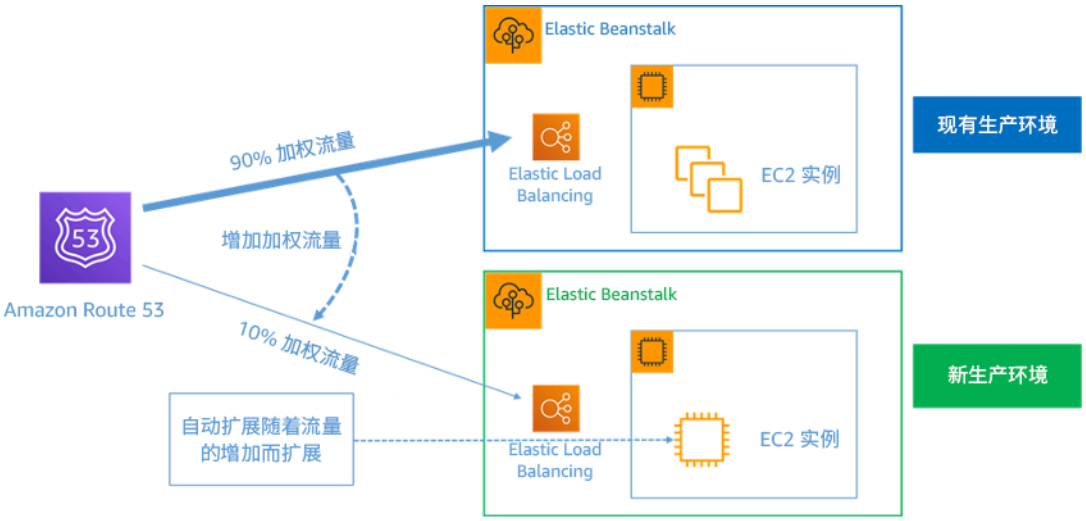


图 8：使用 AWS Elastic Beanstalk 和 Amazon Route 53 进行蓝绿部署

偏差检测

偏差定义为导致基础设施资源的状态或配置与预期不同的任何更改。任何类型非受管配置更改都与不可变基础设施的概念背道而驰，因此应加以检测和修复，以便成功实施不可变基础设施。

实施步骤

- 禁止就地修改正在运行的基础设施资源。
  - 您可以使用 AWS Identity and Access Management（IAM）来指定可以访问 AWS 中服务和资源的对象，集中管理精细权限，并分析访问权限来细化跨 AWS 的权限。
- 自动部署基础设施资源，以提高可重复性并最大限度地减少出现人为错误的可能性。
  - 正如《AWS DevOps 简介》白皮书中所述，自动化是 AWS 服务的基石，并且在所有服务、功能和产品中均得到内部支持。
  - 预烘焙亚马逊机器映像（AMI）可以加快启动速度。EC2 Image Builder 是一项完全托管的 AWS 服务，可帮助您自动创建、维护、验证、共享和部署自定义、安全且最新的 Linux 或 Windows 自定义 AMI。
  - 一些支持自动化的服务包括：
    - AWS Elastic Beanstalk 是一项服务，用于在熟悉的服务器（例如 Apache、NGINX、Passenger 和 IIS）上部署和扩展使用 Java、.NET、PHP、Node.js、Python、Ruby、GO 和 Docker 开发的 Web 应用程序。
    - AWS Proton 让平台团队能够连接和协调开发团队进行基础设施预置、代码部署、监控和更新所需的所有不同工具。AWS Proton 可实现自动化基础设施即代码预置，以及无服务器和基于容器的应用程序的部署。
  - 利用基础设施即代码可以轻松实现基础设施部署的自动化，并有助于实现基础设施的不可变性。AWS 提供的服务可用于以编程、描述性和声明性的方式创建、部署和维护基础设施。
    - AWS CloudFormation 帮助开发人员以有序且可预测的方式创建 AWS 资源。资源使用 JSON 或 YAML 格式写入文本文件。模板需要特定的语法和结构，具体取决于创建和管理的资源类型。可以使用任何代码编辑器以 JSON 或 YAML 格式创作资源，将其签入版本控制系统，然后 AWS CloudFormation 会以安全、可重复的方式构建指定的服务。
    - AWS Serverless Application Model（AWS SAM）是一个开源框架，可用于在 AWS 上构建无服务器应用程序。AWS SAM 与其他 AWS 服务集成，是 AWS CloudFormation 的扩展。
    - AWS Cloud Development Kit (AWS CDK) 是一个开源软件开发框架，可使用熟悉的编程语言对云应用程序资源进行建模和预置。您可以使用 AWS CDK 通过 TypeScript、Python、Java 和 .NET 对应用程序基础设施进行建模。AWS CDK 在后台使用 AWS CloudFormation，以安全、可重复的方式预置资源。
    - AWS 云端控制 API 引入了一组通用的创建、读取、更新、删除和列出（CRUDL）API，帮助开发人员以简单一致的方式管理其云基础设施。Cloud Control API 通用 API 允许开发人员统一管理 AWS 和第三方服务的生命周期。
- 实施能够最大限度减少对用户的影响的部署模式。

  - 金丝雀部署：
    - 设置 API Gateway 金丝雀版本部署
    - 使用 AWS App Mesh 为 Amazon ECS 创建具有金丝雀部署的管道
  - 蓝绿部署：《Blue/Green Deployments on AWS》白皮书介绍了实施蓝绿部署策略的示例技术。
- 检测配置或状态偏差。有关更多详细信息，请参阅检测堆栈和资源的非托管配置更改。

资源

相关最佳实践：

- REL08-BP05 使用自动化功能部署更改

相关文档：

- 自动实现无需干预的安全部署
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- 基础设施即代码
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

相关视频：

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

REL08-BP05 使用自动化功能部署更改

自动执行部署与修补来消除负面影响。

对许多组织来说，对生产系统进行变更是风险最大的工作之一。除了软件解决的业务问题外，我们认为部署也是亟待解决的首要问题。如今，这意味着根据实际情况在操作中使用自动化，包括测试和部署更改、添加或删除容量以及迁移数据。

期望结果：通过广泛的预生产测试、自动回滚和错开生产部署，将自动化部署安全性融入发布流程。这种自动化尽可能地减少了部署失败对生产造成的潜在影响，开发人员不再需要主动关注部署到生产的情况。

常见反模式：

- 手动执行更改。
- 跳过自动化流程中的步骤，采用手动应急工作流。
- 不遵循既定的计划和流程，急于求成。
- 在不预留烘焙时间的情况下，快速执行了后续部署。

建立此最佳实践的好处：当使用自动化来部署所有更改时，可以消除引入人为错误的可能性，并提供在更改生产环境之前进行测试的能力。在生产推送之前执行此流程，以便验证您的计划是否能完成。此外，自动回滚到发布流程可以识别生产问题，并将您的工作负载恢复到以前的正常工作运行状态。

在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

实现部署管道的自动化。借助部署管道，您可以调用自动化测试和异常检测，并且能够在生产部署前的某个步骤停止管道，或自动回滚更改。其中必不可少的是采用持续集成和持续交付/部署（CI/CD）文化，这样一来，提交或代码更改会经过各种自动化阶段（完成构建和测试，并最终部署至生产环境中）。

虽然传统观点认为，您应该让人来处理循环中最困难的操作程序，但出于相同的原因，我们建议您将最困难的程序自动化。

实施步骤

您可以按照以下步骤实现自动化部署，从而消除手动操作：

- 设置代码存储库以安全地存储您的代码：使用基于 Git 等流行技术的托管源代码管理系统，来存储源代码和基础设施即代码（IaC）配置。
- 配置持续集成服务来编译源代码、运行测试和创建部署构件：要为此目的设置构建项目，请参阅 Getting started with AWS CodeBuild using the console。
- 设置部署服务来自动执行应用程序部署并处理复杂的应用程序更新，无需依赖容易出错的人工部署过程：AWS CodeDeploy 可自动将软件部署到各种计算服务，例如 Amazon EC2、AWS Fargate、AWS Lambda 和本地服务器。要配置这些步骤，请参阅 Getting started with CodeDeploy。
- 设置持续交付服务，实现发布管道的自动化，从而带来更快、更可靠的应用程序和基础设施更新：请考虑使用 AWS CodePipeline 来帮助您实现发布管道的自动化。有关更多详细信息，请参阅 CodePipeline tutorials。

资源

相关最佳实践：

- OPS05-BP04 使用构建和部署管理系统
- OPS05-BP10 完全自动化集成和部署
- OPS06-BP02 测试部署
- OPS06-BP04 自动测试和回滚

相关文档：

- Continuous Delivery of Nested AWS CloudFormation Stacks Using AWS CodePipeline
- APN 合作伙伴：可帮助创建自动化部署解决方案的合作伙伴
- AWS Marketplace：可用于自动实施部署的产品
- Automate chat messages with webhooks.
- Amazon Builders' Library：确保部署期间安全回滚
- Amazon Builders' Library：采用持续交付，加速交付进度
- 什么是 AWS CodePipeline？
- What Is CodeDeploy?
- AWS Systems Manager Patch Manager
- What is Amazon SES?
- What is Amazon Simple Notification Service?

相关视频：

- AWS Summit 2019: CI/CD on AWS

故障管理

问题

- REL 9. 您如何备份数据？
- REL 10. 如何使用故障隔离来保护工作负载？
- REL 11. 如何将工作负载设计为可承受组件故障的影响？

- REL 12. 如何测试可靠性？
- REL 13. 如何规划灾难恢复（DR）？

REL 9. 您如何备份数据？

备份数据、应用程序和配置，以满足您对恢复时间目标 (RTO) 和恢复点目标 (RPO) 的要求。

最佳实践

- REL09-BP01 识别并备份需要备份的所有数据或从源复制数据
- REL09-BP02 保护并加密备份
- REL09-BP03 自动执行数据备份
- REL09-BP04 定期执行数据恢复以验证备份完整性和流程

REL09-BP01 识别并备份需要备份的所有数据或从源复制数据

了解并使用工作负载所用的数据服务和资源的备份功能。大多数服务提供了备份工作负载数据的功能。

期望结果：数据来源已确定，并根据重要性进行了分类。然后，根据 RPO 为数据恢复建立了策略。此策略涉及到备份这些数据来源，或者能够从其他来源复制数据。在出现数据丢失的情况下，所实施的策略可以在定义的 RPO 和 RTO 内实现数据的恢复或复制。

云成熟度阶段：基础

常见反模式：

- 不了解工作负载的所有数据来源及其重要性。
- 没有对关键数据来源进行备份。
- 仅对部分数据来源进行备份，但没有考虑重要性标准。
- 没有定义 RPO，或者备份频率无法满足 RPO。
- 没有评估备份是否必需或者是否可以从其他来源复制数据。

建立此最佳实践的好处：确定需要备份的位置并实施某种机制来创建备份，或者具备从外部来源复制数据的能力，这样可以提高在停机期间还原和恢复数据的能力。

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

所有 AWS 数据存储均提供备份功能。Amazon RDS 和 Amazon DynamoDB 等服务还额外地支持可实现时间点故障恢复（PITR）的自动备份，这使您可以将备份恢复到距当前时间不超过五分钟的任意时间点。许多 AWS 服务提供了将备份复制到其他 AWS 区域的功能。AWS Backup 工具向您提供了在不同 AWS 服务中集中实现自动化数据保护的能力。AWS Elastic Disaster Recovery 使您可以从本地、跨可用区或跨区域复制完整的服务器工作负载并保持连续数据保护，恢复点目标（RPO）以秒为单位。

Amazon S3 可用作自行管理数据来源和 AWS 托管数据来源的备份目标。Amazon EBS、Amazon RDS、和 Amazon DynamoDB 等 AWS 服务具有可用于创建备份的内置功能。此外，也可使用第三方备份软件。

可以使用 AWS Storage Gateway 或 AWS DataSync 将本地数据备份到 AWS Cloud。Amazon S3 存储桶可用于在 AWS 中存储此数据。Amazon S3 提供多个存储层（例如 Amazon S3 Glacier 或 S3 Glacier Deep Archive），可用于降低数据存储的成本。

您可以从其他来源复制数据，以此来满足数据恢复需求。例如，Amazon ElastiCache 副本节点或 Amazon RDS 只读副本可用于在主来源丢失时复制数据。如果像这样的来源可用于满足恢复点目标（RPO）和恢复时间目标（RTO）要求，您可能不需要备份。在另一个例子中，如果使用 Amazon EMR，只要可以将数据从 Amazon S3 复制到 Amazon EMR 中，则可能不需要备份 HDFS 数据存储。

在选择备份策略时，请考虑恢复数据所用的时间。恢复数据所需的时间取决于备份的类型（在采用备份策略时）或数据复制机制的复杂性。此时间应该符合工作负载的 RTO。

实施步骤

1. 确定工作负载的所有数据来源。数据可以存储在多种资源中，例如数据库、卷、文件系统、日志记录系统和对象存储。请参阅资源部分，查找有关存储数据的不同 AWS 服务的相关文档，以及这些服务提供的备份功能。
2. 根据重要性对数据来源进行分类。对于工作负载，不同数据集具有不同的重要程度，因此对韧性具有不同的要求。例如，一些数据可能会非常重要，要求接近于零的 RPO，而另一些数据则不那么重要，可以承受较高的 RPO 和某种程度的数据丢失。与此类似，不同数据集也可能会有不同的 RTO 要求。
3. 使用 AWS 或第三方服务来创建数据的备份。AWS Backup 是一项托管服务，支持在 AWS 上创建各种数据来源的备份。AWS Elastic Disaster Recovery 处理到 AWS 区域的自动亚秒级数据复制。大多数 AWS 服务还具有原生的创建备份功能。AWS Marketplace 有许多解决方案同样提供了这些功能。请参阅下面所列的资源，了解有关如何从不同 AWS 服务创建数据备份的信息。

4. 为没有备份的数据建立数据复制机制。您可能会出于各种原因，不对可从其他来源复制的数据进行备份。您可能会遇到一种情况，在需要时从来源复制数据的成本相比创建备份更低，因为可能会有与存储备份相关的成本。另一个例子是从备份进行还原的时间比从来源复制数据用时更长，使得备份不符合 RTO 要求。在此类情况下请做出权衡，并建立明确定义的流程，确定在需要进行恢复时如何从这些来源复制数据。例如，若从 Amazon S3 将数据加载到数据仓库（如 Amazon Redshift）或 MapReduce 集群（如 Amazon EMR），以便对此类数据进行分析，这就算是从其他来源复制数据的例子。只要此类分析的结果被存储在某位置或者可重现，您就不会因为数据仓库或 MapReduce 集群故障而承受数据丢失风险。其他可从数据来源复制数据的例子包括缓存（如 Amazon ElastiCache）或 RDS 只读副本。
5. 制定备份数据的频率。创建数据来源的备份是一个定期执行的流程，其频率取决于 RPO。

实施计划的工作量级别：中

资源

相关最佳实践：

REL13-BP01 定义停机和数据丢失的恢复目标

REL13-BP02 使用定义的恢复策略来实现恢复目标

相关文档：

- 什么是 AWS Backup？
- What is AWS DataSync?
- What is Volume Gateway?
- APN 合作伙伴：可帮助进行备份的合作伙伴
- AWS Marketplace：可用于备份的产品
- Amazon EBS Snapshots
- Backing Up Amazon EFS
- Backing up Amazon FSx for Windows File Server
- Backup and Restore for ElastiCache for Redis
- Creating a DB Cluster Snapshot in Neptune
- 创建数据库快照
- Creating an EventBridge Rule That Triggers on a Schedule

- 使用 Amazon S3 进行跨区域复制
- EFS 到 EFS AWS Backup
- Exporting Log Data to Amazon S3
- 对象生命周期管理
- DynamoDB 的按需备份和还原
- DynamoDB 的时间点恢复
- Working with Amazon OpenSearch Service Index Snapshots
- 什么是 AWS Elastic Disaster Recovery？

相关视频：

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP02 保护并加密备份

使用身份验证和授权功能来控制并检测对备份的访问。使用加密功能防止备份的数据完整性遭到破坏，以及检测其完整性是否遭到损坏。

常见反模式：

- 对备份和还原自动化的访问权限与对数据的访问权限相同。
- 不加密备份。

建立此最佳实践的好处：保护备份安全可防止篡改数据，而加密数据可防止数据意外暴露时遭到访问。

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

使用 AWS Identity and Access Management（IAM）等身份验证和授权服务来控制并检测对备份的访问。使用加密功能防止备份的数据完整性遭到破坏，以及检测其完整性是否遭到损坏。

Amazon S3 支持多种对静态数据进行加密的方式。借助服务器端加密功能，Amazon S3 以未加密数据的形式接受对象，然后在存储此类数据时进行加密。若采用客户端加密，工作负载应用程序需要负

责在将数据发送到 Amazon S3 之前加密数据。这两种方式都让您可以使用 AWS Key Management Service（AWS KMS）创建并存储数据密钥，或者您也可以提供自己的密钥并自行对其负责。使用 AWS KMS，您可以通过 IAM 设置策略，决定谁可以、谁不可以访问数据密钥与解密数据。

针对 Amazon RDS，如果您已选择对数据库进行加密，那么备份也会被加密。DynamoDB 备份始终加密。使用 AWS Elastic Disaster Recovery 时，加密所有传输中数据和静态数据。借助弹性灾难恢复，可以使用默认的 Amazon EBS 加密“卷加密密钥”或自定义的客户自主管理型密钥来加密静态数据。

实施步骤

1. 对每个数据存储使用加密。如果源数据已加密，则备份也将被加密。
   - 在 Amazon RDS 中使用加密。当您创建 RDS 实例时，可以使用 AWS Key Management Service 配置静态加密。
   - 在 Amazon EBS 卷上使用加密。您可以配置默认加密或在创建卷时指定唯一密钥。
   - 使用所需的 Amazon DynamoDB 加密。DynamoDB 可加密所有静态数据。您可以使用 AWS 拥有的 AWS KMS 密钥或者 AWS 托管式 KMS 密钥，指定存储在账户中的密钥。
   - 加密 Amazon EFS 中存储的数据。在创建文件系统时配置加密。
   - 在源和目标区域中配置加密。您可以使用 KMS 中存储的密钥在 Amazon S3 中配置静态加密，但这些密钥是区域特定密钥。您在配置复制时可以指定目标密钥。
   - 选择是为弹性灾难恢复使用默认还是自定义 Amazon EBS 加密。使用此选项会加密暂存区域子网磁盘和复制磁盘上的已复制静态数据。
2. 实施用于访问备份的最低权限。请遵循最佳实践，根据安全最佳实践来限制对备份、快照和副本的访问。

资源

相关文档：

- AWS Marketplace：可用于备份的产品
- Amazon EBS Encryption
- Amazon S3：利用加密来保护数据
- CRR 附加配置：复制通过存储在 AWS KMS 中的加密密钥、使用服务器端加密（SSE）创建的对象
- DynamoDB 静态加密
- 加密 Amazon RDS 资源
- Encrypting Data and Metadata in Amazon EFS

- Encryption for Backups in AWS
- 管理加密表
- 安全性支柱 - AWS Well-Architected Framework
- 什么是 AWS Elastic Disaster Recovery？

REL09-BP03 自动执行数据备份

将备份配置为根据遵循恢复点目标（RPO）的定期计划自动备份，或者在数据集发生更改时自动备份。具有低数据丢失要求的关键数据集，需要频繁地自动备份；而可以接受一定丢失的较不关键的数据，备份频率可以更低。

期望结果：按照确定的节奏创建数据来源备份的自动流程。

常见反模式：

- 手动执行备份。
- 使用具有备份功能的资源，但不包括自动化中的备份。

建立此最佳实践的好处：自动化备份可以确保按照 RPO 定期执行备份，并在未备份时发出警报。

在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 Backup 可用于创建各种 AWS 数据来源的自动数据备份。Amazon RDS 实例可以按照五分钟的频率进行几乎连续的备份，Amazon S3 对象可以按照十五分钟的频率进行几乎连续的备份，提供可恢复到备份历史记录中的特定时间点的时间点故障恢复（PITR）功能。对于其他 AWS 数据来源（如 Amazon EBS 卷、Amazon DynamoDB 表或 Amazon FSx 文件系统），AWS Backup 最快可以按每小时的频率运行自动备份。这些服务还提供了原生备份功能。以下 AWS 服务提供了具备时间点故障恢复的自动备份功能：Amazon DynamoDB、Amazon RDS 和 Amazon Keyspaces（Apache Cassandra 兼容）；这些备份可以恢复到备份历史记录中的特定时间点。大部分其他 AWS 数据存储服务提供了计划定期备份的功能，频率最快为每小时一次。

Amazon RDS 和 Amazon DynamoDB 提供支持时间点恢复的持续备份。一旦启用，Amazon S3 版本控制即会自动工作。Amazon Data Lifecycle Manager 可用于自动创建、复制和删除 Amazon EBS 快照。它还可以自动创建、复制、弃用和取消注册 Amazon EBS 支持的亚马逊机器映像（AMI）及其底层 Amazon EBS 快照。

AWS Elastic Disaster Recovery 提供从源环境（本地或 AWS）到目标恢复区域的持续、块级复制。服务会自动创建和管理时间点 Amazon EBS 快照。

针对您的备份自动化和历史的集中式视图，AWS Backup 提供完全托管的基于策略的备份解决方案。它会使用 AWS Storage Gateway 将云端和本地的多项 AWS 服务的数据备份集中在一起并自动处理。

除了版本控制，Amazon S3 还具有复制功能。整个 S3 存储桶都可自动复制到相同或不同 AWS 区域中的其他存储桶。

实施步骤

1. 确定当前在手动备份的数据来源。有关更多详细信息，请参阅REL09-BP01 识别并备份需要备份的所有数据或从源复制数据。
2. 确定工作负载的 RPO。有关更多详细信息，请参阅REL13-BP01 定义停机和数据丢失的恢复目标。
3. 使用自动化备份解决方案或托管服务。AWS Backup 是一项完全托管式服务，可让您在云端和本地对不同 AWS 服务中的数据保护实现集中化和自动化。使用 AWS Backup 中的备份计划，创建规则来定义要备份的资源，以及创建这些备份的频率。此频率应遵循在第 2 步中确定的 RPO。有关如何使用 AWS Backup 创建自动备份的动手实践指导，请参阅 Testing Backup and Restore of Data。用于存储数据的大多数 AWS 服务提供了原生备份功能。例如，可以利用 RDS 来实现支持时间点故障恢复（PITR）的自动备份。
4. 对于自动备份解决方案或托管服务不支持的数据来源（如本地数据来源或消息队列），请考虑使用受信任的第三方解决方案来创建自动备份。或者，您可以使用 AWS CLI 或开发工具包创建自动化过程来完成此操作。您可以使用 AWS Lambda 函数或 AWS Step Functions 来定义创建数据备份中涉及的逻辑，并使用 Amazon EventBridge 按照基于 RPO 确定的频率来调用它。

实施计划的工作量级别：低

资源

相关文档：

- APN 合作伙伴：可帮助进行备份的合作伙伴
- AWS Marketplace：可用于备份的产品
- Creating an EventBridge Rule That Triggers on a Schedule
- 什么是 AWS Backup？
- 什么是 AWS Step Functions？

- 什么是 AWS Elastic Disaster Recovery？

相关视频：

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP04 定期执行数据恢复以验证备份完整性和流程

通过执行恢复测试，验证备份流程实施是否满足恢复时间目标（RTO）和恢复点目标（RPO）要求。

期望结果：使用明确定义的机制定期从备份恢复数据，确认可以按照为工作负载确定的恢复时间目标（RTO）来恢复数据。验证从备份进行还原可以得到包含原始数据的资源，而不会造成数据损坏或无法访问数据，并且数据丢失在恢复点目标（RPO）之内。

常见反模式：

- 还原备份，但未查询或检索任何数据以确认还原操作可用。
- 假定备份存在。
- 假定系统的备份完全正常运行，并且可从中恢复数据。
- 假定从备份还原或恢复数据的时间满足工作负载的 RTO。
- 假定备份中包含的数据符合工作负载的 RPO
- 需要时进行还原，没有使用运行手册或者没有按照确定的自动程序执行。

建立此最佳实践的好处：测试备份的恢复过程可以确认在需要时能够将数据还原，不必担心数据可能丢失或损坏，可以按照工作负载要求的 RTO 还原和恢复，并且任何数据丢失都符合工作负载的 RPO。

在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

测试备份和还原功能可树立信心，确信能够在出现中断时执行这些操作。定期将备份还原到新的位置，并运行测试以验证数据的完整性。应该执行一些常见的测试，以核实所有数据是否均可用、未损坏、可访问且任何数据丢失都符合工作负载的 RPO。此类测试还可以帮助确定恢复机制是否足够快以满足工作负载的 RTO 要求。

使用 AWS，您可以构建一个测试环境，还原您的备份以评测 RTO 和 RPO 功能，并且对数据的内容和完整性执行测试。

此外，Amazon RDS 和 Amazon DynamoDB 还允许时间点故障恢复（PITR）。您可以使用持续备份将您的数据集还原到其在指定日期与时间所处的状态。

数据是否可用、没有损坏、是否可以访问并且任意数据丢失都符合工作负载的 RPO。此类测试还可以帮助确定恢复机制是否足够快以满足工作负载的 RTO 要求。

AWS Elastic Disaster Recovery 提供 Amazon EBS 卷的持续时间点恢复快照。复制源服务器时，根据配置的策略记录一段时间内的时间点状态。弹性灾难恢复可以启动实例用于测试和演练，而不重定向流量，从而帮助您验证这些快照的完整性。

实施步骤

1. 确定当前备份的数据来源以及存储这些备份的位置。有关实施指导，请参阅 REL09-BP01 识别并备份需要备份的所有数据或从源复制数据。
2. 为每个数据来源建立数据验证标准。不同类型的数据具有不同的属性，这可能需要不同的验证机制。在确信可将此数据用于生产之前，请考虑可以如何验证此数据。一些验证数据的常见方法包括使用数据和备份属性，例如数据类型、格式、校验和、大小，或者将这些属性与自定义的验证逻辑结合使用。例如，可以将所恢复资源的校验和值，与创建备份时数据来源的校验和值进行比较。
3. 设立 RTO 和 RPO，根据数据重要性来还原数据。有关实施指导，请参阅 REL13-BP01 定义停机和数据丢失的恢复目标。
4. 评测恢复能力。检查备份和还原策略，了解是否可以满足 RTO 和 RPO，再根据需要调整策略。使用 AWS 韧性监测中心，可对工作负载运行评估。该评测根据韧性策略评估应用程序配置，报告是否能够满足 RTO 和 RPO 目标。
5. 使用当前为生产环境中数据还原所确立的流程执行测试还原。这些流程依赖于对原始数据来源进行备份的方法，备份本身的格式和存储位置，或者数据是否从其他来源复制。例如，若使用的是 AWS Backup 等托管服务，则此流程可能就是简单地将备份还原到新的资源。如果使用的是 AWS Elastic Disaster Recovery，则可以启动恢复演练。
6. 根据您之前为数据验证确立的标准，从还原后的资源验证数据恢复。还原和恢复的数据是否包含备份时的最新记录或项目？ 此数据是否在工作负载的 RPO 之内？
7. 测量还原和恢复所需的时间，并与确立的 RTO 进行比较。此流程是否符合工作负载的 RTO？ 例如，比较还原流程开始时的时间戳以及恢复验证完成时的时间戳，由此计算此流程的用时。所有 AWS API 调用均有时间戳，此信息在 AWS CloudTrail 中提供。虽然此信息可以提供还原流程何时开始的详细信息，但验证完成时的结束时间戳应该由验证逻辑来记录。如果使用自动流程，则 Amazon DynamoDB 等服务可用于存储此信息。此外，许多 AWS 服务提供了事件历史记录，其中可提供发生特定操作时的时间戳信息。在 AWS Backup 中，备份和还原操作称为作业，这些作业在其元数据中包含时间戳信息，可用于测量还原和恢复所需的时间。

8. 如果数据验证失败，或者如果还原和恢复所需的时间超过了为工作负载设定的 RTO，则通知利益相关方。在实施自动化以完成此操作时（例如在本实验中），可以使用 Amazon Simple Notification Service（Amazon SNS）等服务将推送通知（例如电子邮件或短信）发送给利益相关方。这些消息还可以发布到消息传递应用程序，例如 Amazon Chime、Slack 或 Microsoft Teams，或用于使用 AWS Systems Manager OpsCenter 来创建 OpsItems 等任务。
9. 自动执行此流程以便定期运行。例如，AWS Lambda 等服务或 AWS Step Functions 中的状态机可用于自动完成还原和恢复流程，Amazon EventBridge 可用于定期调用此自动工作流，如以下架构图所示。了解如何使用 AWS Backup 自动完成数据恢复验证。此外，这个 Well-Architected Lab提供动手实践体验，可用于练习针对此处的多个步骤实现自动化的方法。

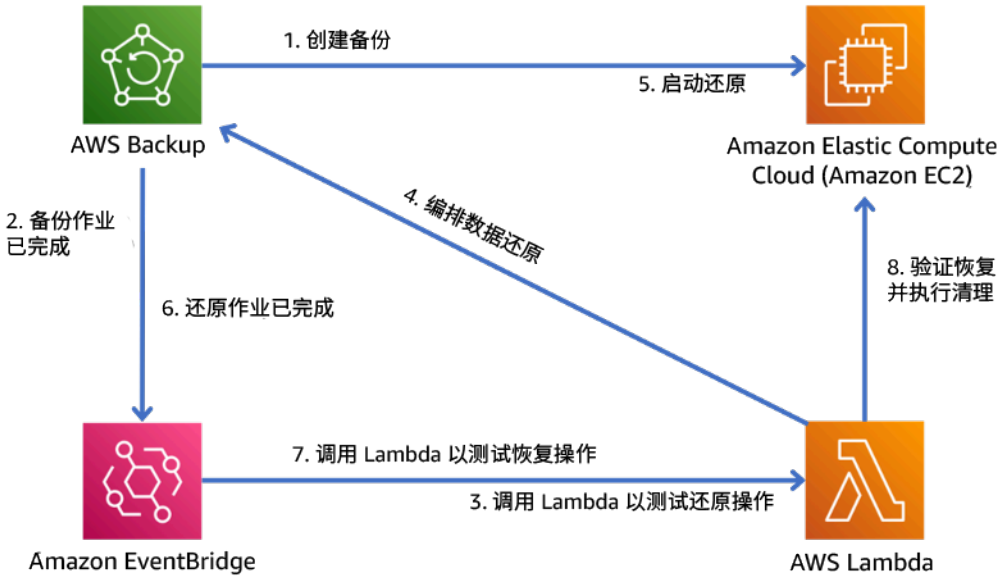


图 9：自动化的备份和还原流程

实施计划的工作量级别：中到高，具体取决于验证标准的复杂性。

资源

相关文档：

- Automate data recovery validation with AWS Backup
- APN 合作伙伴：可帮助进行备份的合作伙伴
- AWS Marketplace：可用于备份的产品
- Creating an EventBridge Rule That Triggers on a Schedule
- DynamoDB 的按需备份和还原

- 什么是 AWS Backup？
- 什么是 AWS Step Functions？
- 什么是 AWS Elastic Disaster Recovery
- AWS Elastic Disaster Recovery

REL 10. 如何使用故障隔离来保护工作负载？

故障隔离可将组件或系统故障的影响限制在定义的界限内。通过适当的隔离，界限之外的组件不受故障影响。跨多个故障隔离界限运行工作负载，可以提高工作负载对故障的韧性。

最佳实践

- REL10-BP01 将工作负载部署到多个位置
- REL10-BP02 组件的自动恢复受限于单个位置
- REL10-BP03 采用隔板架构来限制影响范围

REL10-BP01 将工作负载部署到多个位置

将工作负载数据和资源分布到多个可用区，或在必要时分布到多个 AWS 区域。

AWS 中服务设计的一个基本原则是避免单点故障，包括底层物理基础设施。AWS 在全球多个地理位置（称为 Regions）提供云计算资源和服务。每个区域在物理和逻辑上都是独立的，由三个或更多 Availability Zones (AZs) 组成。可用区在地理上彼此接近，但在物理上是分开和隔离的。当您将工作负载分布于各个可用区和区域之间时，可以降低火灾、洪水、与天气相关的灾难、地震和人为错误等威胁的风险。

制定位置策略，以提供适合您的工作负载的高可用性。

期望结果：生产工作负载分布于多个可用区（AZ）或区域之间，以实现容错和高可用性。

常见反模式：

- 您的生产工作负载只存在于单个可用区中。
- 您在多可用区架构满足业务要求时实施多区域架构。
- 您的部署或数据变得不同步，这会导致配置偏差或数据复制不足。
- 当应用程序组件之间的韧性和多位置要求不同时，您未考虑这些组件之间的依赖关系。

建立此最佳实践的好处：

- 您的工作负载更能抵御意外事件，例如电源或环境控制故障、自然灾害、上游服务故障或影响可用区或整个区域的网络问题。
- 在启动特定 EC2 实例类型时，您可以访问更广泛的 Amazon EC2 实例清单，并降低出现 InsufficientCapacityExceptions（ICE）的可能性。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在区域的至少两个可用区（AZ）中部署和运行所有生产工作负载。

使用多个可用区

可用区是资源托管位置，它们在物理上彼此分开，以避免由于火灾、洪水和龙卷风等风险而导致的相关故障。每个可用区都有独立的物理基础设施，包括市电连接、备用电源、机修服务和网络连接。这种安排方式可将其中任何组件的故障限制在受影响的可用区内。例如，如果可用区范围的事件导致 EC2 实例在受影响的可用区中不可用，则您在其它可用区的实例仍保持可用。

尽管同一 AWS 区域中的可用区在物理上是分开的，但它们之间的距离足够近，可以提供高吞吐量、低延迟（个位数毫秒）的联网。您可以在可用区之间同步复制大多数工作负载的数据，而不会显著影响用户体验。这样一来，您便能以主动/主动或主动/备用配置使用区域中的可用区。

与工作负载关联的所有计算均应分布于多个可用区中。这包括 Amazon EC2 实例、AWS Fargate 任务和 VPC 连接的 AWS Lambda 函数。AWS 计算服务，包括 EC2 Auto Scaling、Amazon Elastic Container Service（ECS）和 Amazon Elastic Kubernetes Service（EKS），为您提供了跨可用区启动和管理计算的方法。将它们配置为根据需要在不同的可用区中自动替换计算以保持可用性。要将流量定向到可用的可用区，请在计算前放置一个负载均衡器，例如应用程序负载均衡器或网络负载均衡器。如果某个可用区受到损害，AWS 负载均衡器可以将流量重新路由到可用的实例。

您还应该为工作负载复制数据，并使其在多个可用区中可用。某些 AWS 托管式数据服务，例如 Amazon S3、Amazon Elastic File Service（EFS）、Amazon Aurora、Amazon DynamoDB、Amazon Simple Queue Service（SQS）和 Amazon Kinesis Data Streams，默认情况可在多个可用区中复制数据，并且可以抵御可用区损害。对于其它 AWS 托管式数据服务，例如 Amazon Relational Database Service（RDS）、Amazon Redshift 和 Amazon ElastiCache，您必须启用多可用区复制。启用后，这些服务会自动检测可用区损害，将请求重定向到可用的可用区，并在恢复后根据需要重新复制数据，而无需客户干预。熟悉您使用的每项 AWS 托管式数据服务的用户指南，以了解其多可用区的功能、行为和操作。

如果您使用的是自行管理的存储，例如 Amazon Elastic Block Store（EBS）卷或 Amazon EC2 实例存储，则必须自行管理多可用区复制。

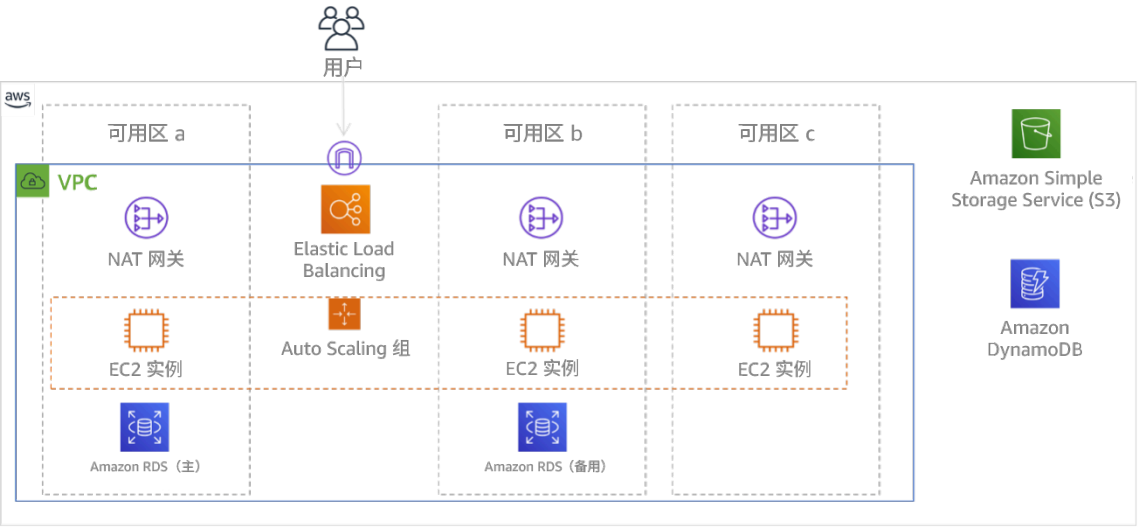


图 9：跨三个可用区部署的多层架构。请注意，Amazon S3 和 Amazon DynamoDB 始终会自动部署到多个可用区。而 ELB 也会被部署到所有三个区。

使用多个 AWS 区域

如果工作负载需要极高的韧性（如关键基础设施、与运行状况相关的应用程序或具有严格的客户或强制可用性要求的服务），您可能需要超出单个 AWS 区域所能提供的额外可用性。在这种情况下，您应该在至少两个 AWS 区域中部署和运行工作负载（假设数据驻留要求支持这么做）。

AWS 区域位于世界各地和多个大洲的不同地理区域。AWS 区域与单独的可用区相比，其物理分离和隔离程度甚至更高。除了少数例外情况，AWS 服务利用这种设计在不同区域之间完全独立运行（也称为区域服务）。AWS 区域服务的故障设计为不影响其它区域中的服务。

当您在多个区域中运行工作负载时，应考虑其它要求。由于不同区域中的资源彼此分开且独立，因此必须在每个区域中复制工作负载的组件。除计算服务和数据服务外，这还包括基本的基础设施，例如 VPC。

注意：在考虑多区域设计时，请验证工作负载能够在单个区域中运行。如果您在区域之间创建依赖关系，其中一个区域中的组件依赖于另一个区域中的服务或组件，则可能会增加故障风险并显著削弱可靠性状况。

为了简化多区域部署并保持一致性，AWS CloudFormation StackSets 可以跨多个区域复制整个 AWS 基础设施。AWS CloudFormation 还可以检测配置偏差，并在某个区域中的 AWS 资源不同步时通知您。许多 AWS 服务为重要的工作负载资产提供多区域复制。例如，例如，EC2 Image Builder

可以在每次构建后将 EC2 机器映像（AMI）发布到您使用的每个区域。Amazon Elastic Container Registry（ECR）可以将容器映像复制到选定的区域。

您还必须跨您所选的每个区域复制数据。许多 AWS 托管式数据服务提供跨区域复制功能，包括 Amazon S3、Amazon DynamoDB、Amazon RDS、Amazon Aurora、Amazon Redshift、Amazon Elasticache 和 Amazon EFS。Amazon DynamoDB 全局表接受在任何受支持的区域中进行写入，并将在所有其它配置的区域间复制数据。对于其它服务，您必须指定一个主区域进行写入，因为其它区域包含只读副本。对于工作负载使用的每项 AWS 托管式数据服务，请参阅其用户指南和开发人员指南，以了解其多区域功能和局限性。请特别注意必须将写入定向到何处、事务处理功能和限制、如何执行复制以及如何监控区域之间的同步。

AWS 还能够非常灵活地将请求流量路由到区域部署。例如，可以使用 Amazon Route 53 配置 DNS 记录，来将流量定向到离用户最近的可用区域。或者，可以在主动/备用配置中配置 DNS 记录，在这种配置中，您将一个区域指定为主区域，仅当主区域变得运行状况不正常时，才回退到区域副本。您可以配置 Route 53 health checks 来检测运行状况不正常的端点并执行自动失效转移，此外，还可以使用 Amazon 应用程序恢复控制器（ARC）来提供高度可用的路由控制，以便根据需要手动重新路由流量。

即使您选择不在多个区域中运行来实现高可用性，也可以将多个区域视为灾难恢复（DR）策略的一部分。如果可能，请在辅助区域中以热备用 或指示灯 配置来复制工作负载的基础设施组件和数据。在此设计中，您从主区域复制基准基础设施，如 VPC、自动扩缩组、容器编排工具和其它组件，但您将备用区域中的可变大小组件（如 EC2 实例和数据库副本的数量）配置为最小可操作大小。您还可以安排从主区域到备用区域的连续数据复制。如果发生事件，则可以横向扩展或增加备用区域中的资源，然后将其提升为主区域。

实施步骤

1. 与业务利益相关者和数据驻留专家合作，来确定哪些 AWS 区域可用来托管您的资源和数据。
2. 与业务和技术利益相关者合作，来评估您的工作负载，并确定其韧性需求是否可以通过多可用区方法（单个 AWS 区域）来满足，或者是否需要多区域方法（如果允许多个区域）。使用多个区域可以提高可用性，但可能会增加复杂性和成本。评估时考虑以下因素：
   a. 业务目标和客户要求：如果在可用区或区域中发生影响工作负载的事件，允许有多少停机时间？评估恢复点目标，如 REL13-BP01 定义停机和数据丢失的恢复目标中所述。
   b. 灾难恢复（DR）要求：您想要确保自行抵御哪种潜在灾难？ 考虑数据丢失或长期不可用的可能性，影响范围涉及单个可用区到整个区域。如果您跨可用区复制数据和资源，而单个可用区持续出现故障，则可以在另一个可用区中恢复服务。如果您跨区域复制数据和资源，则可以在另一个区域中恢复服务。
3. 将计算资源部署到多个可用区中。

a. 在 VPC 中，在不同的可用区中创建多个子网。将每个子网配置为足够大，以容纳处理工作负载所需的资源，即使在发生事件期间也是如此。有关更多详细信息，请参阅 REL02-BP03 确保 IP 子网分配考虑扩展和可用性。

b. 如果您使用的是 Amazon EC2 实例，请使用 EC2 Auto Scaling 来管理实例。在创建自动扩缩组时，指定在上一步中选择的子网。

c. 如果您正在对 Amazon ECS 或 Amazon EKS 使用 AWS Fargate 计算，请在创建 ECS 服务、启动 ECS 任务或为 EKS 创建 Fargate 配置文件时，选择您在第一步中选择的子网。

d. 如果您使用的 AWS Lambda 函数需要在 VPC 中运行，请在创建 Lambda 函数时，选择您在第一步中选择的子网。对于任何没有 VPC 配置的函数，AWS Lambda 自动为您管理可用性。

e. 将流量定向器（例如负载均衡器）放在计算资源之前。如果启用了跨区域负载均衡，AWS 应用程序负载均衡器和网络负载均衡器会检测何时由于可用区受损而无法访问 EC2 实例和容器等目标，并将流量重新路由到正常运行的可用区中的目标。如果您禁用跨区域负载均衡，请使用 Amazon 应用程序恢复控制器（ARC）来提供可用区转移功能。如果您使用的是第三方负载均衡器或已实施了自己的负载均衡器，请为它们配置跨不同可用区的多个前端。

4. 跨多个可用区复制工作负载的数据。

a. 如果您使用的是 AWS 托管式数据服务，例如 Amazon RDS、Amazon ElastiCache 或 Amazon FSx，请研读其用户指南以了解其数据复制和韧性功能。必要时启用跨可用区复制和失效转移。

b. 如果您使用 AWS 托管式存储服务，例如 Amazon S3、Amazon EFS 和 Amazon FSx，请避免对需要高耐久性的数据使用单可用区或单区配置。对这些服务使用多可用区配置。查看相应服务的用户指南，以确定默认情况下是否启用了多可用区复制，或者是否必须启用它。

c. 如果您运行的是自行管理的数据库、队列或其它存储服务，请根据应用程序的说明或最佳实践安排进行多可用区复制。熟悉应用程序的失效转移过程。

5. 配置您的 DNS 服务以检测可用区受损，并将流量重新路由到运行状况正常的可用区。Amazon Route 53 与弹性负载均衡器结合使用时，可以自动执行此操作。还可以为 Route 53 配置失效转移记录，这些记录使用运行状况检查来响应仅具有正常运行的 IP 地址的查询。对于用于失效转移的任何 DNS 记录，请指定较短的生存时间（TTL）值（例如，60 秒或更短），以协助防止记录缓存阻碍恢复（Route 53 别名记录为您提供相应的 TTL）。

使用多个 AWS 区域时的额外步骤

1. 跨所选区域复制工作负载使用的所有操作系统（OS）和应用程序代码。如有必要，可以使用 Amazon EC2 Image Builder 等解决方案复制 EC2 实例使用的亚马逊机器映像（AMI）。使用 Amazon ECR 跨区域复制等解决方案复制存储在注册表中的容器映像。为用于存储应用程序资源的任何 Amazon S3 存储桶启用区域复制。

2. 将您的计算资源和配置元数据（例如，存储在 AWS Systems Manager Parameter Store 中的参数）部署到多个区域。使用前面步骤中介绍的相同过程，但请为您要用于工作负载的每个区域复制配置。使用 AWS CloudFormation 等基础设施即代码解决方案在区域之间统一复制配置。如果您在指示灯配置中使用辅助区域进行灾难恢复，您可以将计算资源的数量减少到最小值以节省成本，同时相应地增加恢复时间。
3. 将数据从主区域复制到辅助区域。
   a. Amazon DynamoDB 全局表提供数据的全局副本，可以从任何支持的区域写入这些副本。对于其它 AWS 托管式数据服务，例如 Amazon RDS、Amazon Aurora 和 Amazon Elasticache，您可以指定主（读/写）区域和副本（只读）区域。有关区域复制的详细信息，请参阅相应服务的用户和开发人员指南。
   b. 如果您运行的是自行管理的数据库，请根据应用程序的说明或最佳实践安排进行多区域复制。熟悉应用程序的失效转移过程。
   c. 如果工作负载使用 AWS EventBridge，则可能需要将选定的事件从主区域转发到辅助区域。为此，请将辅助区域中的事件总线指定为主区域中匹配事件的目标。
4. 考虑是否以及在多大程度上希望跨区域使用相同的加密密钥。平衡安全性和易用性的典型方法是使用区域范围的密钥来处理区域本地数据和身份验证，并使用全球范围的密钥来对在不同区域间复制的数据进行加密。AWS Key Management Service（KMS）支持 multi-region keys，以便安全地分发和保护跨区域共享的密钥。
5. 考虑使用 AWS Global Accelerator，通过将流量定向到包含正常运行的端点的区域，来提高应用程序的可用性。

资源

相关最佳实践：

- REL02-BP03 确保 IP 子网分配考虑扩展和可用性
- REL11-BP05 使用静态稳定性来防止双模态行为
- REL13-BP01 定义停机和数据丢失的恢复目标

相关文档：

- AWS 全球基础设施
- White paper: AWS Fault Isolation Boundaries
- Resilience in Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling: Example: Distribute instances across Availability Zones

- How EC2 Image Builder works
- Amazon ECS 如何将任务放置在容器实例上（包括 Fargate）
- AWS Lambda 中的故障恢复能力
- Amazon S3：复制对象概述
- Amazon ECR 中的私有映像复制
- 全局表：使用 DynamoDB 的多区域复制
- Amazon Elasticache for Redis OSS: Replication across AWS 区域 using global datastores
- Amazon RDS 中的弹性
- 使用 Amazon Aurora Global Database
- AWS Global Accelerator Developer Guide
- Multi-Region keys in AWS KMS
- Amazon Route 53: Configuring DNS failover
- Amazon Application Recovery Controller (ARC) Developer Guide
- Sending and receiving Amazon EventBridge events between AWS 区域
- Creating a Multi-Region Application with AWS Services 系列博客文章
- 开启 AWS 灾难恢复（DR）架构，第一部分：云端恢复策略
- Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby

相关视频：

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure

REL10-BP02 组件的自动恢复受限于单个位置

如果工作负载的组件只能在单个可用区或本地数据中心内运行，则必须利用相关功能在定义的恢复目标内彻底重建工作负载。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

如果由于技术约束无法使用将工作负载部署到多个位置的最佳实践，则必须实施其他的韧性路径。在这种情况下，必须让重建必要基础设施、重新部署应用程序和重建必要数据的操作实现自动化。

例如，Amazon EMR 会为相同可用区内的特定集群启动全部节点，因为在相同区内运行集群可以改善作业流的性能，提高数据访问速率。如果这是工作负载韧性所需的必要组件，则必须设法重新部署集群及其数据。同样对于 Amazon EMR，您还应该通过除多可用区以外的方式对冗余进行预置。您可以预置多个节点。使用 EMR 文件系统（EMRFS），EMR 中的数据可存储在 Amazon S3 中，进而可以跨多个可用区或 AWS 区域进行复制。

同理，对于 Amazon Redshift，默认会在您选择的 AWS 区域内随机选择可用区，然后对其中的集群进行预置。所有集群节点在同一区域中配置。

对于部署到本地数据中心基于服务器的有状态工作负载，您可以使用 AWS Elastic Disaster Recovery 来保护 AWS 中的工作负载。如果已经在 AWS 托管中，则可以使用弹性灾难恢复将工作负载保护到其他可用区或区域。弹性灾难恢复在轻量级暂存区域中使用持续的块级复制，以便快速可靠地恢复本地应用程序和基于云的应用程序。

实施步骤

1. 实施自我修复。尽可能使用自动扩缩部署实例或容器。如果不能使用自动扩缩，则使用 EC2 实例的自动恢复功能，或者基于 Amazon EC2 或 ECS 容器生命周期事件实现自我修复自动化。
   - 将 Amazon EC2 Auto Scaling 组用于对单个实例 IP 地址、私有 IP 地址、弹性 IP 地址和实例元数据没有要求的实例和容器工作负载。
     - 启动模板用户数据可以用于实现自动化，让大多数工作负载可以自我修复。
   - 将 Amazon EC2 实例的自动恢复功能用于需要单个实例 ID 地址、私有 IP 地址、弹性 IP 地址和实例元数据的工作负载。
     - 自动恢复功能会在检测到实例故障时，向 SNS 主题发送恢复状态提醒。
   - 在无法使用自动扩缩或 EC2 恢复的情况下，请使用Amazon EC2 实例生命周期事件或 Amazon ECS 事件实现自我修复自动化。
     - 使用这些事件调用自动化，该自动化将根据您需要的流程逻辑来修复组件。
   - 使用 AWS Elastic Disaster Recovery 保护仅限于单个位置的有状态工作负载。

资源

相关文档：

- Amazon ECS 事件
- Amazon EC2 Auto Scaling 生命周期挂钩
- 恢复实例。
- 服务自动扩缩

- What Is Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP03 采用隔板架构来限制影响范围

实施隔板架构（也称为基于单元的架构），将工作负载中故障的影响限制于有限数量的组件。

期望结果：基于单元的架构使用多个独立的工作负载实例，其中每个实例称为单元。单元彼此独立，不与其他单元共享状态，并且处理整个工作负载请求的子集。这样减少了故障（例如，错误的软件更新）对单个单元及其正在处理的请求的潜在影响。如果某个工作负载使用 10 个单元来服务 100 个请求，则在发生故障时，总请求中有 90% 不会受故障的影响。

常见反模式：

- 让单元无限制地发展。
- 同时将代码更新或部署到所有单元。
- 在不同单元之间共享状态或组件（路由器层除外）。
- 向路由器层添加复杂的业务或路由逻辑。
- 没有尽量减少跨单元交互。

建立此最佳实践的好处：借助基于单元的架构，许多常见类型的故障控制在单元本身，从而实现了额外的故障隔离。若出现难以控制的故障类型（例如不成功的代码部署，或者是受损或调用特定故障模式的请求，也称为毒丸请求），这些故障边界可以提供韧性。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在船舶上，隔板确保船体裂口控制在船体的一个区段内。在复杂的系统中，通常会复制此模式以实现故障隔离。故障隔离边界可将一个工作负载内的故障影响限制于有限数量的组件。边界之外的组件不受故障影响。通过使用多个故障隔离边界，您可以限制对工作负载的影响。在 AWS 中，客户可以使用多个可用区和区域来实现故障隔离，但故障隔离的概念也可以扩展到工作负载的架构。

整个工作负载是分区的单元，按分区键进行划分。这个键需要与服务的粒度，或者与使用最少的跨单元交互来细分服务工作负载的自然方式保持一致。分区键的示例包括客户 ID、资源 ID 或可以在大多数 API 调用中轻松访问的任何其他参数。单元路由层根据分区键将请求分发到单个单元，并向客户端提供单个端点。

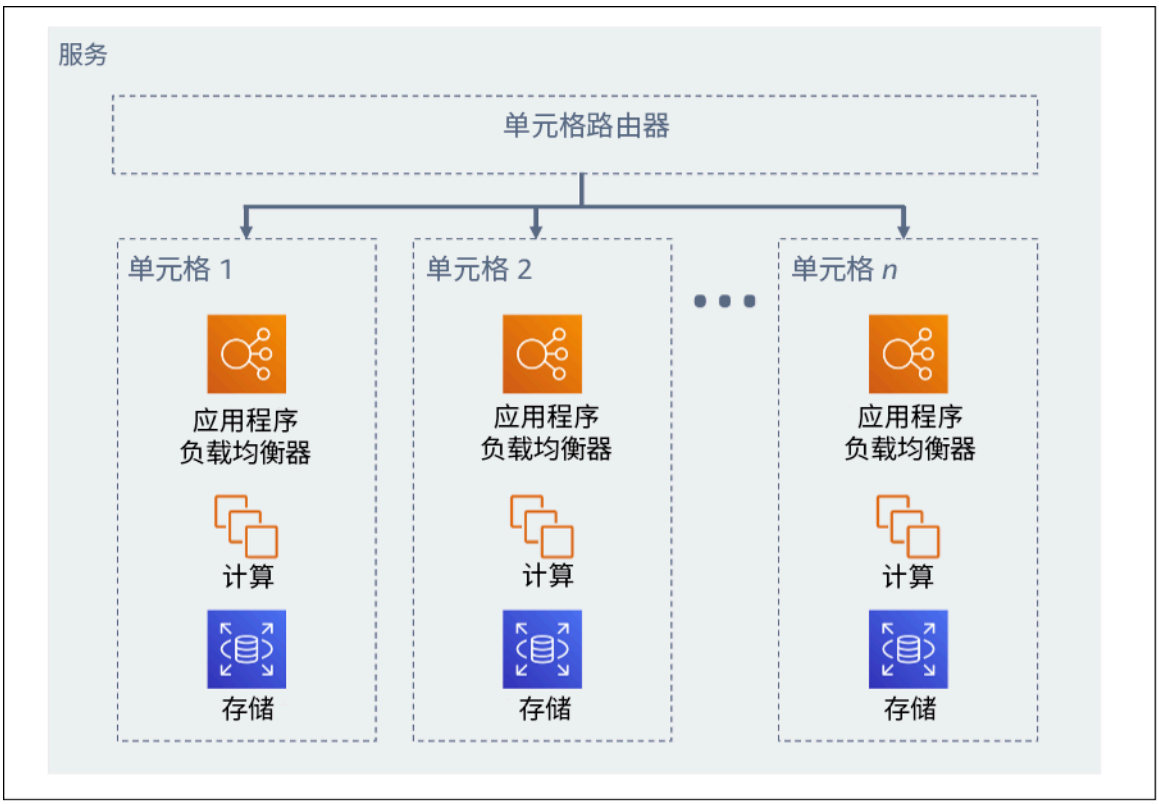


图 11：基于单元的架构

实施步骤

在设计基于单元的架构时，需要考虑几个设计注意事项：

1. 分区键：选择分区键时应考虑一些特别事项。
   - 分区键应与服务的粒度，或者与使用最少的跨单元交互来细分服务工作负载的自然方式保持一致。示例包括 `customer ID` 或 `resource ID`。
   - 在所有请求中都必须提供分区键，要么直接提供，要么通过很容易由其他参数确定地推断出来的方式提供。
2. 持久单元映射：上游服务在其资源的生命周期内应只与单个单元交互。
   - 根据工作负载，可能需要使用单元迁移策略将数据从一个单元迁移到另一个单元。可能需要进行单元迁移的一种情况是：工作负载中的一个特定用户或资源变得太大，要求它具有专用的单元。
   - 单元之间不应该共享状态或组件。
   - 因此，应该避免或尽量减少跨单元交互，因为这些交互会在单元之间产生依赖关系，从而削弱故障隔离的改进。
3. 路由器层：路由层是单元之间的共享组件，因此无法遵循与单元相同的分隔策略。

- 建议路由器层使用分区映射算法以一种计算效率高的方式将请求分发到各个单元，例如结合加密哈希函数和模块化算法，将分区键映射到单元。
- 为避免产生多单元影响，路由层必须尽可能保持简单和可横向扩展，这就需要在此层中避免出现复杂的业务逻辑。这还有一个额外的好处，即始终可以轻松地理解其预期行为，从而实现彻底的可测试性。正如 Colm MacCárthaigh 在《Reliability, constant work, and a good cup of coffee》一文中所说，简单的设计和持续工作模式可产生可靠的系统并降低抗脆弱性。

4. 单元大小：单元大小应具有上限，不得发展到超过这个值。
   - 应通过执行彻底的测试来确定大小上限，直至达到临界点并建立安全的运营边际。有关如何实施测试实践的更多详细信息，请参阅 REL07-BP04 对工作负载进行负载测试
   - 总体工作负载增长时应增加额外的单元，使得工作负载能够随着需求的增加而扩展。
5. 多可用区或多区域策略：应利用多层韧性来防止出现不同的故障域。
   - 要实现韧性，应使用可构建防御层的方法。其中一层使用多可用区，通过构建高度可用的架构，防止出现较小规模、更常见的中断。另一个防御层用于防御很少发生的事件，例如大范围的自然灾害和区域级别的中断。这个第二层涉及到设计应用程序的架构来跨越多个 AWS 区域。为工作负载实施多区域策略，有助于防御影响到某个国家/地区中较大地理面积的大范围自然灾害，或者区域范围的技术故障。请注意，实施多区域架构会有很高的复杂性，对于大部分工作负载通常来说都是不必要的。有关更多详细信息，请参阅REL10-BP01 将工作负载部署到多个位置。
6. 代码部署：交错的代码部署策略应该优于同时将代码更改部署到所有单元。
   - 这有助于最大限度地减少因部署不当或人为错误而导致多个单元出现故障。有关更多详细信息，请参阅自动实现无需干预的安全部署。

资源

相关最佳实践：

- REL07-BP04 对工作负载进行负载测试
- REL10-BP01 将工作负载部署到多个位置

相关文档：

- Reliability, constant work, and a good cup of coffee
- AWS and Compartmentalization
- 使用随机分区进行工作负载隔离
- 自动实现无需干预的安全部署

相关视频：

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

REL 11. 如何将工作负载设计为可承受组件故障的影响？

在构建具有高可用性和较短平均恢复时间（MTTR）要求的工作负载时必须考虑到韧性。

最佳实践

- REL11-BP01 监控工作负载的所有组件以检测故障
- REL11-BP02 失效转移到运行状况良好的资源
- REL11-BP03 自动修复所有层
- REL11-BP04 恢复期间依赖于数据面板而不是控制面板
- REL11-BP05 使用静态稳定性来防止双模态行为
- REL11-BP06 当事件影响可用性时发送通知
- REL11-BP07 构建产品以满足可用性目标和正常运行时间服务水平协议（SLA）

REL11-BP01 监控工作负载的所有组件以检测故障

持续监控工作负载的运行状况，以便您和您的自动化系统立即发现任何故障或性能下降情况。监控基于商业价值的关键性能指标（KPI）。

所有恢复和修复机制必须从快速检测问题的能力入手。首先，应该检测技术故障并加以解决。不过，可用性基于工作负载创造商业价值的能力，因此衡量它的关键性能指标（KPI）需要成为检测和补救策略的一部分。

期望结果：独立监控工作负载的重要组成部分，对故障发生的时间和位置进行检测，再根据检测结果发出警报。

常见反模式：

- 未配置警报，因此不会在发生中断时进行通知。
- 虽然配置了警报，但只有在达到阈值时才会发出警报，导致没有足够的响应时间。

- 收集指标的频率不够高，无法满足恢复时间目标（RTO）。
- 仅主动监控工作负载中面向客户的接口。
- 只收集技术指标，不收集业务功能指标。
- 没有衡量工作负载用户体验的指标。
- 创建的监控太多。

建立此最佳实践的好处：如果在所有层都设置了适当的监控，则可以通过减少检测时间来缩短恢复时间。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

确定将接受审查以决定是否监控的所有工作负载。确定工作负载中所有需要监控的组件后，就需要确定监控间隔。根据检测故障所花费的时间，监控间隔将直接影响启动恢复的速度。平均检测时间（MTTD）是从故障发生到修复操作开始之间的时间。服务列表应广泛而完整。

监控必须覆盖应用程序堆栈的所有层，包括应用程序、平台、基础设施和网络。

监控策略应考虑灰色故障的影响。有关灰色故障的更多详细信息，请参阅《Advanced Multi-AZ Resilience Patterns》白皮书中的 Gray failures。

实施步骤

- 监控间隔取决于必须以多快的速度恢复。恢复时间取决于恢复所需的时间，因此在确定收集频率时，必须考虑此时间和恢复时间目标（RTO）。
- 为组件和托管服务配置详细监控。
  - 确定详细监控对于 EC2 实例和自动扩缩来说是否有必要。详细监控以 1 分钟为间隔提供指标，默认监控以 5 分钟为间隔提供指标。
  - 确定是否需要为 RDS 设置增强监控。增强监控使用 RDS 实例上的代理，获取关于不同进程或线程的有用信息。
  - 确定以下各项的关键无服务器组件的监控要求：Lambda、API Gateway、Amazon EKS、Amazon ECS，以及所有类型的负载均衡器。
  - 确定以下各项的存储组件的监控要求：Amazon S3、Amazon FSx、Amazon EFS 和 Amazon EBS。
- 创建自定义指标来测量业务关键性能指标（KPI）。工作负载会实现关键业务功能，这些功能应用作 KPI 来协助在发生间接问题时予以识别。

- 使用用户金丝雀来监控用户的故障体验。可运行并模拟客户行为的综合事务测试（又称为“金丝雀测试”，但与金丝雀部署不同）是一项重要的测试流程。从不同的远程位置针对工作负载端点持续地运行此类测试。
- 创建跟踪用户体验的自定义指标。如果您可以衡量客户体验，就可以确定发生了客户体验下降。
- 设置警报，在检测到工作负载的任何部分未正常运行时发出警报，并指示什么时候自动扩展资源。警报可以直观地显示在控制面板上，通过 Amazon SNS 或电子邮件发送警报，并与自动扩缩功能结合使用来纵向扩展或缩减工作负载资源。
- 创建控制面板，以可视化形式呈现指标。可以使用控制面板直观地查看趋势、离群值和表示其他潜在问题的指标，或者提供您可能需要调查的问题的指示。
- 为服务创建分布式跟踪监控。使用分布式监控，您可以了解应用程序及其底层服务的运行情况，以便确定和诊断性能问题及错误的根本原因。
- 在单独的区域和账户中创建监控系统（使用 CloudWatch 或 X-Ray）控制面板和数据收集。
- 随时了解 AWS Health 的服务降级情况。通过 AWS 用户通知服务 创建要发送到电子邮件和聊天渠道且契合目标的 AWS Health 事件通知，并以编程方式通过 Amazon EventBridge 与监控和警报工具集成。

资源

相关最佳实践：

- 可用性定义
- REL11-BP06 当事件影响可用性时发送通知

相关文档：

- 使用 Amazon CloudWatch Synthetics 创建用户金丝雀
- 为实例启用或禁用详细监控
- 增强监控
- Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch
- 发布自定义指标
- 使用 Amazon CloudWatch 警报
- 使用 CloudWatch 控制面板
- 使用跨区域跨账户的 CloudWatch 控制面板
- 使用跨区域跨账户的 X-Ray 跟踪

- Understanding availability

相关视频：

- Mitigating gray failures

相关示例：

- One Observability Workshop: Explore X-Ray

相关工具：

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 失效转移到运行状况良好的资源

如果资源发生故障，运行状况良好的资源应继续为请求提供服务。对于位置受损（如可用区或 AWS 区域 受损），确保拥有适当的系统，可失效转移到未受损位置内运行状况良好的资源。

设计服务时，应在资源、可用区或区域之间分配负载。因此，可以通过将流量转移到运行状况良好的剩余资源，缓解单个资源的故障或损坏。考虑在出现故障时如何发现服务并将流量路由到相应服务。

在设计服务时要考虑故障恢复。在 AWS，我们设计服务时会尽量缩短从故障恢复的时间并降低对数据的影响。我们的服务主要使用的数据存储，只有在数据持久存储在一个区域中的多个副本之后，才会确认请求。它们经构建为使用基于单元的隔离，并使用可用区提供的故障隔离功能。我们在自己的运营过程中广泛使用自动化。我们还将替换和重新启动功能优化为可从中断快速恢复。

允许失效转移的模式和设计因各项 AWS 平台服务而异。许多 AWS 原生托管服务（如 Lambda 或 API Gateway）本质上是跨多个可用区部署的服务。其他 AWS 服务（如 EC2 和 EKS）需要特定的最佳实践设计，才能支持跨可用区的资源或数据存储的失效转移。

应设置监控功能，检查失效转移资源的运行状况，跟踪资源失效转移的进度，并监控业务流程的恢复情况。

期望结果：系统能够自动使用新资源从降级中恢复，或手动恢复。

常见反模式：

- 规划和设计阶段未考虑如何应对故障。

- 未设立 RTO 和 RPO。
- 监控不足，无法检测出故障的资源。
- 适当隔离故障域。
- 不考虑多区域失效转移。
- 在决定是否进行失效转移时，故障检测过于敏感或过于激进。
- 未测试或验证失效转移设计。
- 执行自动修复，但不通知需要进行该修复。
- 缺少缓冲期，无法避免太快进行失效自动恢复。

建立此最佳实践的好处：可以构建更具韧性的系统，在遇到故障时，通过优雅降级和快速恢复来保持可靠性。

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

AWS 服务（例如弹性负载均衡和 Amazon EC2 Auto Scaling）有助于跨资源和可用区分配负载。因此，可以通过将流量转移到运行状况良好的剩余资源，缓解单个资源（例如 EC2 实例）的故障或可用区的损坏。

对于多区域工作负载，设计就比较复杂。例如，跨区域只读副本允许您将数据部署到多个 AWS 区域。但是，仍需要进行失效转移才能将只读副本提升为主副本，然后将流量指向新的端点。Amazon Route 53、Amazon 应用程序恢复控制器 (ARC)、Amazon CloudFront 和 AWS Global Accelerator 可以协助跨 AWS 区域路由流量。

Amazon S3、Lambda、API Gateway、Amazon SQS、Amazon SNS、Amazon SES、Amazon Pinpoint、Amazon ECR、AWS Certificate Manager、EventBridge 或 Amazon DynamoDB 等 AWS 服务将通过 AWS 自动部署到多个可用区。如果出现故障，这些 AWS 服务会自动将流量路由到运行状况良好的位置。数据在多个可用区中进行冗余存储，并保持可用。

对于 Amazon RDS、Amazon Aurora、Amazon Redshift、Amazon EKS 或 Amazon ECS，多可用区是一个配置选项。如果启动了失效转移，AWS 可以将流量引导到运行状况良好的实例。此失效转移操作可由 AWS 执行，或根据客户要求执行

对于 Amazon EC2 实例、Amazon Redshift、Amazon ECS 任务或 Amazon EKS 容器组，您要选择部署到哪些可用区。对于某些设计，弹性负载均衡会提供解决方案来检测运行状况不佳的可用区内的实例，并将流量路由至运行状况良好的可用区。弹性负载均衡还可以将流量路由至本地数据中心内的组件。

对于多区域流量失效转移，重新路由可以利用 Amazon Route 53、Amazon 应用程序恢复控制器、AWS Global Accelerator、Route 53 Private DNS for VPC 或 CloudFront 提供一种方法，来定义互联网域并分配路由策略（包括运行状况检查），从而将流量路由到运行状况良好的区域。AWS Global Accelerator 提供静态 IP 地址，这些地址充当应用程序的固定入口点，然后使用 AWS 全球网络而不是互联网来路由到您选择的 AWS 区域中的端点，由此获得更高的性能和可靠性。

实施步骤

- 为所有相应的应用程序和服务创建失效转移设计。隔离每个架构组件，为每个组件创建符合 RTO 和 RPO 的失效转移设计。
- 使用失效转移计划所需的所有服务配置底层环境（例如开发或测试）。使用基础设施即代码（IaC）部署解决方案，确保可重复性。
- 配置恢复站点（例如第二个区域）来实施并测试失效转移设计。如有必要，可以临时配置测试资源来限制额外成本。
- 确定哪些失效转移计划由 AWS 自动执行，哪些可以通过 DevOps 流程自动执行，哪些可能需要手动执行。记录并测量每项服务的 RTO 和 RPO。
- 创建失效转移行动手册，包括对每个资源、应用程序和服务进行失效转移的所有步骤。
- 创建失效自动恢复行动手册，包括对每个资源、应用程序和服务进行失效自动恢复的所有步骤（包含时机）
- 制定计划来启动行动手册并加以演练。使用模拟和混沌测试来测试行动手册步骤和自动化功能。
- 对于位置受损（如可用区或 AWS 区域 受损），确保拥有适当的系统，可失效转移到未受损位置内运行状况良好的资源。在测试失效转移之前，请检查配额、自动扩展级别和正在运行的资源。

资源

相关的 Well-Architected 最佳实践：

- REL13 – 制定灾难恢复计划
- REL10 – 使用故障隔离来保护工作负载

相关文档：

- 设立 RO 和 RPO 目标
- 使用 Route 53 加权路由进行失效转移
- Disaster Recovery with Amazon Application Recovery Controller

- 带自动扩缩功能的 EC2
- EC2 部署 – 多可用区
- ECS 部署 – 多可用区
- Switch traffic using Amazon Application Recovery Controller
- 使用应用程序负载均衡器和失效转移的 Lambda
- ACM 复制和失效转移
- Parameter Store 复制和失效转移
- ECR 跨区域复制和失效转移
- Secrets Manager 跨区域复制配置
- 为 EFS 和失效转移启用跨区域复制
- EFS 跨区域复制和失效转移
- 网络失效转移
- 使用 MRAP 的 S3 端点失效转移
- 为 S3 创建跨区域复制
- Guidance for Cross Region Failover and Graceful Failback on AWS
- 使用多区域全球加速器进行失效转移
- 使用 DRS 进行失效转移

相关示例：

- Disaster Recovery on AWS
- Elastic Disaster Recovery on AWS

REL11-BP03 自动修复所有层

在检测到故障时，使用自动化功能执行修复操作。降级可能会通过内部服务机制自动修复，也可能需要通过补救措施重启或移除资源。

对于自我管理型应用程序和跨区域修复，可以从现有最佳实践中获取恢复设计和自动修复流程。

重启或移除资源是修复故障的重要方法。最佳实践是尽可能使服务为无状态。这可以防止重启资源时数据丢失或可用性受损。在云中，作为重启的一部分，您可以（而且在一般情况下也应该）替换完整的资源（例如计算实例或无服务器函数）。重启本身是从故障恢复的简单而可靠的方法。工作负载中会发生很多不同类型的故障。故障可能发生在硬件、软件、通信和操作上。

重启或重试也适用于网络请求。向网络超时以及依赖项返回错误的依赖性故障应用相同的恢复方法。这两个事件对系统具有类似的影响，可应用类似的采用指数回退和抖动的有限重试策略，而不是尝试将各个事件当作特例进行处理。重启功能是面向恢复的计算和高可用性集群架构的特色恢复机制。

期望结果：执行自动操作来修复检测到的故障。

常见反模式：

- 预置资源，而不进行自动扩缩。
- 在实例或容器中单独部署应用程序。
- 在不使用自动恢复的情况下，部署无法部署到多个位置的应用程序。
- 手动修复自动扩缩和自动恢复无法修复的应用程序。
- 缺乏对数据库进行失效转移的自动化功能。
- 缺乏将流量重新路由到新端点的自动化方法。
- 缺乏存储复制功能。

建立此最佳实践的好处：自动修复可以缩短平均恢复时间，并提高可用性。

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

Amazon EKS 或其他 Kubernetes 服务的设计应包括最小和最大副本集或有状态集，以及最小集群和节点组大小。这些机制提供了最低数量的持续可用处理资源，同时使用 Kubernetes 控制面板自动修复任何故障。

使用计算集群通过负载均衡器访问的设计模式应利用自动扩缩组。弹性负载均衡（ELB）自动在一个或多个可用区（AZ）中的多个目标和虚拟设备之间分配传入的应用程序流量。

对于不使用负载平衡的基于集群计算的设计，其规模至少应能承受一个节点的中断。这将允许服务在恢复新节点时，使自身以可能降低的容量维持运行状态。示例服务有 Mongo、DynamoDB Accelerator、Amazon Redshift、Amazon EMR、Cassandra、Kafka、MSK-EC2、Couchbase、ELK 和 Amazon OpenSearch Service。其中许多服务都可以设计带有额外的自动修复功能。某些集群技术必须在节点丢失时生成警报，从而触发自动或手动工作流程来重新创建新节点。该工作流程可以使用 AWS Systems Manager 自动执行，从而快速修复问题。

Amazon EventBridge 可用于监控和筛选事件，例如 CloudWatch 警报或其他 AWS 服务中的状态更改。然后，其可根据事件信息调用 AWS Lambda、Systems Manager Automation 或其他目标，对工作负载运行自定义修复逻辑。可以配置 Amazon EC2 Auto Scaling 来检查 EC2 实例的运行状况。若实

例处于正在运行以外的任何状态，或系统状态受损，Amazon EC2 Auto Scaling 会认为实例的运行状况不佳，继而启动替换实例。针对大规模替换（例如整个可用区丢失），静态稳定性更适合用来实现高可用性。

实施步骤

- 使用自动扩缩组在工作负载中部署层。自动扩缩可以对无状态应用程序执行自我修复，以及添加或移除容量。
- 对于前面提到的计算实例，可使用负载均衡，然后选择合适的负载均衡器类型。
- 考虑 Amazon RDS 修复。对于备用实例，请配置自动失效转移到备用实例。针对 Amazon RDS 只读副本，需要自动化工作流程，使只读副本成为主副本。
- 在 EC2 实例上实施自动恢复，这些实例中部署的应用程序无法部署到多个位置，但在故障后允许重新启动。当无法将应用程序部署到多个位置时，可以使用自动恢复来替换发生故障的硬件并重新启动实例。实例元数据和关联的 IP 地址，以及 EBS 卷和 Amazon Elastic File System 或 适用于 Lustre 的文件系统和适用于 Windows 的文件系统的挂载点，都会得到保留。使用 AWS OpsWorks 时，您可以在层级别配置 EC2 实例的自动修复。
- 当无法使用自动扩缩或自动恢复，或者自动恢复出故障时，可以使用 AWS Step Functions 和 AWS Lambda 实施自动恢复。当无法使用自动扩缩，并且无法使用自动恢复或自动恢复失败时，可以使用 AWS Step Functions 和 AWS Lambda 进行自动修复。
- Amazon EventBridge 可用于监控和筛选事件，例如 CloudWatch 警报或其他 AWS 服务中的状态更改。根据事件信息，该服务可以调用 AWS Lambda（或其他目标），在工作负载上运行自定义修复逻辑。

资源

相关最佳实践：

- 可用性定义
- REL11-BP01 监控工作负载的所有组件以检测故障

相关文档：

- AWS Auto Scaling 的工作原理
- Amazon EC2 Automatic Recovery
- Amazon Elastic Block Store（Amazon EBS）
- Amazon Elastic File System (Amazon EFS)

- What is Amazon FSx for Lustre?
- What is Amazon FSx for Windows File Server?
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances
- 什么是 AWS Step Functions ？
- 什么是 AWS Lambda ？
- 什么是 Amazon EventBridge？
- 使用 Amazon CloudWatch 警报
- Amazon RDS 失效转移
- SSM – Systems Manager Automation
- 韧性架构最佳实践

相关视频：

- Automatically Provision and Scale OpenSearch Service
- Amazon RDS Failover Automatically

相关示例：

- Amazon RDS 失效转移讲习会

相关工具：

- CloudWatch
- CloudWatch X-Ray

REL11-BP04 恢复期间依赖于数据面板而不是控制面板

控制面板提供用于创建、读取、描述、更新、删除和列出（CRUDL）资源的管理 API，而数据面板则处理日常服务流量。在对可能影响韧性的事件实施恢复或缓解响应时，着眼于使用最少数量的控制面板操作，实现对服务的恢复、重新扩缩、恢复、修复或失效转移。在这些降级事件期间，数据面板操作应凌驾于任何活动之上。

例如，以下是所有控制面板操作：启动新的计算实例、创建数据块存储和描述队列服务。启动计算实例时，控制面板必须执行多项任务，例如查找具有容量的物理主机、分配网络接口、准备本地数据块存储卷、生成凭证以及添加安全规则。控制面板往往是复杂的编排。

期望结果：当资源进入受损状态时，将流量从受损资源转移到运行状况良好的资源，能够自动或手动恢复系统。

常见反模式：

- 依赖于通过更改 DNS 记录来重新路由流量。
- 由于预置资源不足，依赖控制面板扩展操作来替换受损组件。
- 依靠广泛、多服务、多 API 控制面板操作来修复任何类别的受损情况。

建立此最佳实践的好处：提高自动修复的成功率可以缩短平均恢复时间，并提高工作负载的可用性。

在未建立这种最佳实践的情况下暴露的风险等级：中。对于某些类型的服务降级，控制面板会受到影响。依赖于大量使用控制面板进行修复，可能会增加恢复时间（RTO）和平均恢复时间（MTTR）。

实施指导

要限制数据面板操作，请评测每项服务，了解恢复服务所需的操作。

利用 Amazon 应用程序恢复控制器来转移 DNS 流量。这些功能会持续监控应用程序从故障中恢复的能力，让您能够跨多个 AWS 区域、可用区和本地部署控制应用程序恢复。

Route 53 路由策略使用控制面板，因此不要依赖控制面板进行恢复。Route 53 数据面板会回复 DNS 查询，并执行和评估运行状况检查。它们分布在全球各地，专为 100% 可用性的服务水平协议（SLA）而设计。

用于创建、更新和删除 Route 53 资源的 Route 53 管理 API 和控制台是在控制面板上运行，而这些控制面板设计用于优先考虑在管理 DNS 时所需的强一致性和持久性。为了实现这一点，控制面板位于单个区域“美国东部（弗吉尼亚州北部）”中。虽然这两个系统都非常可靠，但控制面板不包含在 SLA 中。在极少数情况下，数据面板的韧性设计允许自身保持可用性，而控制面板做不到。对于灾难恢复和失效转移机制，使用数据面板功能可提供尽可能高的可靠性。

将计算基础设施设计为静态稳定，以避免在事故发生期间使用控制面板。例如，如果您使用的是 Amazon EC2 实例，请避免手动配置新实例或指示自动扩缩组添加实例作为响应。要获得最高级别的韧性，请在集群中预置足够的容量用于失效转移。如果必须限制此容量阈值，请在整个端到端系统上设置限制，安全地限制到达有限资源集的总流量。

对于诸如 Amazon DynamoDB、Amazon API Gateway、负载均衡器和 AWS Lambda 无服务器之类的服务，使用这些服务时可以利用数据面板。但是，创建新函数、负载均衡器、API Gateway 或 DynamoDB 表是一项控制面板操作，应在降级之前完成，以便为事件和演练失效转移操作做准备。对于 Amazon RDS，数据面板操作允许访问数据。

有关数据面板、控制面板以及 AWS 如何构建服务来满足高可用性目标的更多信息，请参阅使用可用区的静态稳定性。

了解哪些操作位于数据面板，哪些位于控制面板。

实施步骤

对于降级事件后需要恢复的每个工作负载，请评估失效转移运行手册、高可用性设计、自动修复设计或高可用性（HA）资源恢复计划。确定可能被视为控制面板操作的每个操作。

考虑将控制面板操作更改为数据面板操作：

- 自动扩缩（控制面板）到预扩展 Amazon EC2 资源（数据面板）
- Amazon EC2 实例扩展（控制面板）到 AWS Lambda 扩展（数据面板）
- 评测任何使用 Kubernetes 的设计以及控制面板操作的性质。在 Kubernetes 中，添加容器组是一项数据面板操作。操作应仅限于添加容器组而不是添加节点。使用预置过度的节点是限制控制面板操作的首选方法

考虑允许数据面板操作影响相同修复措施的其他方法。

- Route 53 记录更改（控制面板）或 Amazon 应用程序恢复控制器（数据面板）
- Route 53 运行状况检查，可获取更多自动更新

如果是任务关键型服务，可考虑使用辅助区域中的某些服务，以便在不受影响的区域内进行更多控制面板和数据面板操作。

- 主区域中的 Amazon EC2 Auto Scaling 或 Amazon EKS 与辅助区域中的 Amazon EC2 Auto Scaling 或 Amazon EKS 相比，以及将流量路由到辅助区域（控制面板操作）
- 在辅助区域中创建只读副本或在主区域中尝试相同的操作（控制面板操作）

资源

相关最佳实践：

- 可用性定义
- REL11-BP01 监控工作负载的所有组件以检测故障

相关文档：

- APN 合作伙伴：可帮助实现容错自动化的合作伙伴
- AWS Marketplace：可支持容错的产品
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Amazon DynamoDB API（控制面板和数据面板）
- AWS Lambda 执行（拆分为控制面板和数据面板）
- AWS Elemental MediaStore Data Plane
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack
- 使用 Amazon Route 53 创建灾难恢复机制
- What is Amazon Application Recovery Controller
- Kubernetes 控制面板和数据面板

相关视频：

- Back to Basics - Using Static Stability
- Building resilient multi-site workloads using AWS global services

相关示例：

- Introducing Amazon Application Recovery Controller
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack
- 使用可用区的静态稳定性

相关工具：

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 使用静态稳定性来防止双模态行为

工作负载应具有静态稳定性，并且仅在单一正常模式下运行。双模态行为是指工作负载在正常模式和故障模式下表现出不同的行为。

例如，您可能会尝试在不同的可用区中启动新实例，以便从可用区故障中恢复。在故障模式下，这可能会导致双模态响应。您应该构建静态稳定的工作负载，并且仅在一个模式下运行。在此示例中，这些实例应在出现故障之前，已经在第二个可用区中预置。这种静态稳定性设计可确保工作负载仅在单一模式下运行。

期望结果：在正常模式和故障模式下，工作负载不会表现出双模态行为。

常见反模式：

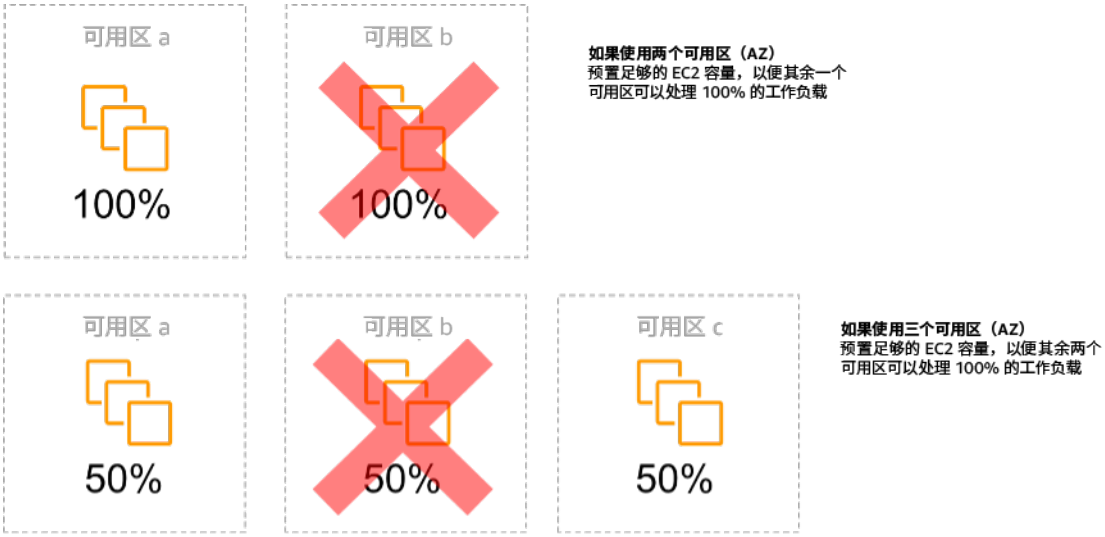
- 假设无论故障范围如何，始终可以预置资源。
- 尝试在故障期间动态获取资源。
- 在出现故障之前，没有跨可用区或跨区域预置足够的资源。
- 仅为计算资源考虑了静态稳定设计。

建立此最佳实践的好处：采用静态稳定设计运行的工作负载，在正常情况和故障事件期间能够得到可预测的结果。

在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

当工作负载在正常模式和故障模式下展现出不同的行为时，这就是双模态行为（例如，在可用区发生故障时依赖于启动新的实例）。双模态行为的一个例子是，稳定的 Amazon EC2 设计在每个可用区中预置了足够的实例，用于处理在移除了一个可用区时的工作负载。弹性负载均衡或 Amazon Route 53 运行状况将进行检查，将负载从受损实例上移开。在流量转移以后，使用 AWS Auto Scaling 异步替换故障可用区的实例，并在运行状况良好的可用区中启动。适用于计算部署（如 EC2 实例或容器）的静态稳定性将提供最高水平的可靠性。



跨可用区的 EC2 实例的静态稳定性

您必须就这一模型的成本以及在所有情况下保持工作负载韧性的业务价值进行权衡。预置较少的计算容量并依赖于在出现故障时启动新实例，这种方法的成本较低，但是对于大规模故障（例如可用区或区域受损），这种方法效果较差，因为它既依赖于运营层面，也依赖未受影响的可用区或区域中有足够的可用资源。

您的解决方案还需要在工作负载的可靠性和成本需求之间做出取舍。静态稳定性架构适用于各种架构，包括跨多个可用区的计算实例、数据库只读副本设计、Kubernetes（Amazon EKS）集群设计和多区域失效转移架构。

您还可以在每个可用区中使用更多资源，从而实施具有更好的静态稳定性的设计。通过添加更多可用区，您可以减少实现静态稳定性所需的额外计算容量。

双模态行为的一个例子是，网络超时可能会导致系统尝试刷新整个系统的配置状态。这会向另一个组件添加意外负载，进而可能导致该组件出现故障，引发其他意外后果。此负面的反馈环路会影响工作负载的可用性。相反，您可以构建静态稳定的系统，并且仅在一个模式下运行。静态稳定的设计会持续工作，并且始终定期刷新配置状态。当调用失败时，工作负载将使用先前缓存的值并启动警报。

双模态行为的另一个示例是允许客户端在故障发生时绕过工作负载缓存。这看起来似乎是可以满足客户端需求的解决方案，但它会明显改变工作负载的需求，而且很有可能导致故障。

评测关键工作负载，确定哪些工作负载需要这种韧性设计。对于被视为关键的工作负载，必须审核每个应用程序组件。需要静态稳定性评估的服务类型示例包括：

- 计算：Amazon EC2、EKS-EC2、ECS-EC2、EMR-EC2
- 数据库：Amazon Redshift、Amazon RDS、Amazon Aurora

- 存储：Amazon S3（单区）、Amazon EFS（挂载）、Amazon FSx（挂载）
- 负载均衡器：在某些设计下

实施步骤

- 构建静态稳定的系统，并且仅在一个模式下运行。在这种情况下，应在每个可用区或区域中预置足够的实例，以便在移除了一个可用区或区域时处理工作负载容量。您可以使用多种服务来路由到运行状况良好的资源，例如：
  - 跨区域 DNS 路由
  - MRAP Amazon S3 多区域路由
  - AWS Global Accelerator
  - Amazon Application Recovery Controller
- 配置数据库只读副本，应对单个主实例或只读副本丢失的情况。如果流量由只读副本提供服务，则每个可用区和每个区域中的资源数量，应等于可用区或区域出现故障时的总体需求量。
- 在 Amazon S3 存储中配置关键数据，设计为在可用区出现故障时可确保所存储数据的静态稳定性。如果使用 Amazon S3 One Zone-IA 存储类，则不应将其视为静态稳定，因为丢失该可用区会将对所存储数据的可访问性降到最低。
- 负载均衡器有时服务于特定可用区，这可能是因为配置不正确，也可能是有意设计成这样。在这种情况下，静态稳定的设计可能需要采用更复杂的设计，将工作负载分布到多个可用区。出于安全、延迟或成本方面的考虑，可能会使用最初的设计来减少区域间流量。

资源

相关的 Well-Architected 最佳实践：

- 可用性定义
- REL11-BP01 监控工作负载的所有组件以检测故障
- REL11-BP04 恢复期间依赖于数据面板而不是控制面板

相关文档：

- Minimizing Dependencies in a Disaster Recovery Plan
- Amazon Builders' Library：使用可用区的静态稳定性
- Fault Isolation Boundaries

- 使用可用区的静态稳定性
- 多可用区 RDS
- Minimizing Dependencies in a Disaster Recovery Plan
- 跨区域 DNS 路由
- MRAP Amazon S3 多区域路由
- AWS Global Accelerator
- Amazon Application Recovery Controller
- 单区 Amazon S3
- Cross Zone Load Balancing

相关视频：

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL11-BP06 当事件影响可用性时发送通知

在检测到突破阈值时发送通知，即使导致问题的事件已自动解决。

自动修复使您的工作负载变得可靠。不过，它也可能会掩盖需要处理的潜在问题。实施适当的监控和措施，以便检测问题的模式，包括那些被自动修复的问题，以便从根本上解决问题。

韧性系统经过精心设计，可以立即将性能下降事件传达给相应的团队。这些通知应通过一个或多个通信渠道发送。

期望结果：在突破阈值 [例如错误率、延迟或其他重要的关键绩效指标（KPI）] 时，系统会立即向运营团队发送警报，以便尽快解决这些问题，避免或最大限度地减少对用户的影响。

常见反模式：

- 发送的警报过多。
- 发送不可操作的警报。
- 将警报阈值设置得太高（过于敏感）或太低（不够敏感）。
- 不针对外部依赖项发送警报。
- 在设计监控和警报时不考虑灰色故障。

- 执行自动修复，但未通知相应的团队需要进行该修复。

建立此最佳实践的好处：恢复通知可以让运营和业务团队了解到服务性能下降的情况，这样他们就可以立即做出反应，尽可能地缩短平均检测时间（MTTD）和平均修复时间（MTTR）。恢复事件通知还可确保您不会忽略不经常发生的问题。

在未建立这种最佳实践的情况下暴露的风险等级：中。未能实施适当的监控和事件通知机制，会导致未能检测到出现的问题模式，包括那些被自动修复的问题。只有当用户联系客户服务或者在偶然的情况下，团队才会了解到系统性能下降的情况。

实施指导

在定义监控策略时，触发的警报是常见事件。此事件可能包含警报的标识符、警报状态（例如 `IN ALARM` 或 `OK`），以及触发该警报的对象的详细信息。在许多情况下，系统应该检测到警报事件并发送电子邮件通知。这是对警报执行操作的示例。警报通知对于可观测性至关重要，因为它会告知相关人员所存在的问题。不过，当您的可观测性解决方案能够针对事件采取合理的操作时，它可以自动修复问题，而无需人工干预。

建立 KPI 监控警报后，在超过阈值时，应向相应的团队发送警报。这些警报还可用于触发自动流程，尝试修复性能下降问题。

对于更复杂的阈值监控，应考虑使用复合警报。复合警报使用多个 KPI 监控警报，根据运营业务逻辑创建警报。CloudWatch 警报可被配置为发送电子邮件，或使用 Amazon SNS 集成或 Amazon EventBridge 将事件记录到第三方事件跟踪系统。

实施步骤

根据监控工作负载的方式，创建各种类型的警报，例如：

- 使用应用程序警报，检测工作负载的任何部分未正常工作的情况。
- 基础设施警报指明何时扩展资源。警报可以直观地显示在控制面板上，通过 Amazon SNS 或电子邮件发送警报，并与 Auto Scaling 结合使用来横向扩展或缩减工作负载资源。
- 可以创建简单的静态警报，用于监控在指定数量的评估周期内，指标突破静态阈值的情况。
- 复合警报可以处理来自多个来源的复杂警报。
- 创建警报后，请创建相应的通知事件。您可以直接调用 Amazon SNS API 来发送通知，并关联任何自动化功能进行修复或通信。
- 随时了解 AWS Health 的服务降级情况。通过 AWS 用户通知服务 创建要发送到电子邮件和聊天渠道且契合目标的 AWS Health 事件通知，并以编程方式通过 Amazon EventBridge 与监控和警报工具集成。

资源

相关的 Well-Architected 最佳实践：

- 可用性定义

相关文档：

- 根据静态阈值创建 CloudWatch 警报
- 什么是 Amazon EventBridge？
- What is Amazon Simple Notification Service?
- 发布自定义指标
- 使用 Amazon CloudWatch 警报
- 设置 CloudWatch 复合警报
- What's new in AWS Observability at re:Invent 2022

相关工具：

- CloudWatch
- CloudWatch X-Ray

REL11-BP07 构建产品以满足可用性目标和正常运行时间服务水平协议（SLA）

构建产品以满足可用性目标和正常运行时间服务水平协议（SLA）。如果您公开或私下同意可用性目标或正常运行时间 SLA，请确保架构和运营流程的设计支持它们。

期望结果：每个应用程序的性能指标都有明确的可用性和 SLA 目标，可以监控和维护目标，实现业务成果。

常见反模式：

- 在不设置任何 SLA 的情况下设计和部署工作负载。
- 在没有理由或业务需求的情况下将 SLA 指标设置得很高。
- 在设置 SLA 时不考虑依赖项及其基础 SLA。
- 设计应用程序时不考虑韧性的责任共担模式。

建立此最佳实践的好处：根据关键韧性目标设计应用程序，有助于实现业务目标并满足客户期望。这些目标有助于推动评估不同技术并考虑各种权衡的应用程序设计过程。

在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

应用程序设计必须考虑因业务、运营和财务目标而产生的各种要求。根据运营要求，工作负载需要具有特定的韧性指标目标，以便进行监控并予以支持。不得在部署工作负载之后才设置或引出韧性指标。这些指标应该在设计阶段得到定义，有助于指导作出各种决策和权衡。

- 每个工作负载都应有自己的一组韧性指标。这些指标可能与其他业务应用程序的指标不同。
- 减少依赖项可以对可用性产生积极影响。每个工作负载都应考虑自己的依赖项及其 SLA。一般来说，选择可用性目标等于或大于工作负载目标的依赖项。
- 若可行，尽量考虑采用松耦合设计，让工作负载可以在依赖关系受损的情况下正常运行。
- 减少控制面板依赖项，特别是在恢复或降级期间。评估可确保任务关键型工作负载静态稳定性的设计。使用资源节约来提高工作负载中这些依赖项的可用性。
- 可观测性和检测能力对于通过减少平均检测时间（MTTD）和平均修复时间（MTTR）来实现 SLA 至关重要。
- 降低故障频率（提高 MTBF）、缩短故障检测时间（缩短 MTTD）和缩短修复时间（缩短 MTTR）是提高分布式系统可用性的三项因素。
- 设立并满足工作负载的韧性指标，这是所有有效设计的基础。这些设计必须在设计复杂性、服务依赖项、性能、扩展和成本之间作出权衡。

实施步骤

- 检查并记录工作负载设计，考虑以下问题：
  - 在工作负载中的什么地方使用控制面板？
  - 工作负载如何实施容错？
  - 扩展、自动扩展、冗余和高可用性组件的设计模式是什么？
  - 数据一致性和可用性的要求是什么？
  - 是否考虑了资源节约或资源静态稳定性？
  - 有哪些服务依赖项？
- 与利益相关方合作时，根据工作负载架构定义 SLA 指标。考虑工作负载使用的所有依赖项的 SLA。
- 设立了 SLA 目标后，优化架构来满足 SLA。

- 确定了满足 SLA 的设计后，实施侧重于减少 MTTD 和 MTTR 的运营更改、流程自动化和运行手册。
- 部署之后，监控和报告 SLA。

资源

相关最佳实践：

- REL03-BP01 选择如何划分工作负载
- REL10-BP01 将工作负载部署到多个位置
- REL11-BP01 监控工作负载的所有组件以检测故障
- REL11-BP03 自动修复所有层
- REL12-BP04 使用混沌工程测试韧性
- REL13-BP01 定义停机和数据丢失的恢复目标
- 了解工作负载运行状况

相关文档：

- Availability with redundancy
- 可靠性支柱 – 可用性
- 衡量可用性
- AWS Fault Isolation Boundaries
- 韧性的责任共担模式
- 使用可用区的静态稳定性
- AWS 服务水平协议（SLA）
- Guidance for Cell-based Architecture on AWS
- AWS infrastructure
- Advanced Multi-AZ Resilience Patterns whitepaper

相关服务：

- Amazon CloudWatch
- AWS Config

- AWS Trusted Advisor

REL 12. 如何测试可靠性？

在为工作负载采用韧性设计以应对生产压力以后，测试是确保其按设计预期运行，并且提供所预期韧性的唯一方式。

最佳实践

- REL12-BP01 使用行动手册调查故障
- REL12-BP02 执行事后分析
- REL12-BP03 测试可扩展性和性能要求
- REL12-BP04 使用混沌工程测试韧性
- REL12-BP05 定期进行 GameDay 活动

REL12-BP01 使用行动手册调查故障

通过在行动手册中记录调查流程，对并不十分了解的故障场景实现一致且及时的响应。行动手册是在确定哪些因素导致故障场景时要执行的预定义步骤。所有流程步骤的结果都将用于确定要采取的后续步骤，直到问题得到确定或上报。

行动手册是您必须要执行的主动计划，以便有效采取被动措施。当在生产中遇到行动手册未涉及的故障场景时，首先要解决问题（灭火）。然后回过头来思考您在解决问题时采取的措施，并将这些措施作为新条目添加到行动手册中。

请注意，行动手册可用于对特定事件做出响应，运行手册则用来达成特定的结果。通常，运行手册适用于例行活动，而行动手册则用于对非例行事件做出响应。

常见反模式：

- 计划在以下情况下部署工作负载：不清楚诊断问题或响应事件的流程。
- 关于在对事件进行调查时从哪些系统收集日志和指标的计划外的决定。
- 指标和事件保留的时间不够长，无法检索到数据。

建立此最佳实践的好处：使用行动手册可确保始终如一地遵循流程。编写行动手册可以减少手动操作导致的错误。通过实现行动手册自动化，可以消除团队成员干预的需要，或者在他们开始干预时便向他们提供更多信息，从而缩短事件响应时间。

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

- 使用行动手册来发现问题。行动手册是用于调查问题的书面程序。在行动手册中记录流程，实现对故障场景的一致而及时的响应。行动手册必须包含所需的信息和指导，让足够熟练的员工能够收集适用信息、确定故障的潜在来源、隔离故障，并确定成因（即执行事后分析）。
  - 以代码形式实施行动手册。为行动手册编写脚本，以代码形式执行运营，确保一致性并减少由手动流程引起的错误。行动手册可以由代表不同步骤的多个脚本组成，这些步骤可能是确定问题成因所必需的。系统可能会在运行手册活动过程中调用或执行行动手册活动，也可能针对响应发现的事件而提示执行行动手册活动。
    - 使用 AWS Systems Manager 自动执行运营行动手册
    - AWS Systems Manager Run Command
    - AWS Systems Manager Automation
    - 什么是 AWS Lambda ？
    - 什么是 Amazon EventBridge？
    - 使用 Amazon CloudWatch 警报

资源

相关文档：

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- 使用 AWS Systems Manager 自动执行运营行动手册
- 使用 Amazon CloudWatch 警报
- 使用金丝雀（Amazon CloudWatch Synthetics）
- 什么是 Amazon EventBridge？
- 什么是 AWS Lambda ？

相关示例：

- 根据行动手册和运行手册自动完成操作

REL12-BP02 执行事后分析

审核影响客户的事件，确定这些事件的成因和预防措施。利用这些信息来制定缓解措施，限制或防止再次发生同类事件。制定程序，以便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。如果需要，可将这些原因告知他人。

评测为什么现有测试找不到问题。如果还没有，为此案例增设测试。

期望结果：您的团队采用一致且一致的方法来处理事后分析。一种机制是错误更正（COE）流程。COE 流程有助于您的团队识别、理解和解决事件的根本原因，同时还可以建立防护机制，以限制同一事件再次发生的可能性。

常见反模式：

- 查找事件成因，但不继续深入探究其他潜在问题和缓解问题的方法。
- 只找出人为错误原因，但不提供任何培训或可防止人为错误的自动化功能。
- 只注重追究责任，而不去了解根本原因，营造恐惧文化，阻碍开诚布公的交流
- 见解分享不畅，事件分析结果仅限于一小群人知道，其他人无法从中吸取经验教训
- 没有收集制度性知识的机制，因此无法以最新最佳实践的形式保存经验教训，从而失去宝贵见解，导致根本原因相同或相似的事件反复发生

建立此最佳实践的好处：如果其他工作负载实施了相同的成因，执行事后分析并共享分析结果可帮助缓解这些工作负载的故障风险，让它们能够在事件发生之前实施缓解或自动恢复措施。

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

有效的事后分析让您有机会针对系统中其他地方使用的架构模式存在的问题，提出常见的解决方案。

COE 流程的基础是记录和解决问题。建议定义一种标准化的方法来记录关键的根本原因，并确保其得到分析和处理。为事后分析流程分配明确的责任人。指派负责的团队或个人来监督事件调查和后续行动。

鼓励注重学习和改进而不是互相推脱责任的文化。强调目标是防止将来发生事件，而不是惩罚某些人。

为执行事后分析制定明确定义的程序。这些程序应概述要采取的步骤、要收集的信息以及在分析期间要解决的关键问题。彻底调查事件，除直接原因外，还要找出根本原因和成因。使用诸如五个为什么之类的技巧来深入研究潜在问题。

维护从事件分析中吸取的经验教训的存储库。这些制度性知识可以作为未来的事件和预防工作的参考。分享在事后分析中发现的结果和洞察，并考虑召开公开的事后审查会议，讨论经验教训。

实施步骤

- 在执行事后分析时，确保整个过程不是以追究责任为目的。这使事件中涉及到的人员能够冷静地看待建议的纠正措施，并促进诚实的自我评测和团队间的协作。
- 定义记录关键问题的标准化方法。此类文档的示例结构如下所示：
  - 发生了什么？
  - 客户和您的业务受到了什么影响？
  - 根本原因是什么？
  - 您有哪些数据来支持这一点？
    - 例如，指标和图表
  - 对关键支柱有什么影响，特别是在安全方面？
    - 在构造工作负载时，您需要基于业务环境在各个支柱之间做出权衡。这些业务决策可以推动您的工程优先事务。在开发环境中，您可能会通过降低可靠性来降低成本；而对于任务关键型解决方案，您可能会通过增加成本来提高可靠性。安全始终是头等大事，因为您必须保护您的客户。
  - 您获得了哪些经验教训？
  - 您采取了哪些纠正措施？
    - 操作项
    - 相关术语
- 为执行事后分析制定明确定义的标准操作程序。
- 设置标准化事件报告流程。全面记录所有事件，包括最初的事件报告、日志、通信和事件期间采取的行动。
- 请记住，并不是发生了停机才叫做事件。这可能是未遂事件，也可能是系统能够履行其业务功能，但以意外的方式运行。
- 根据反馈和经验教训，持续改进事后分析流程。
- 在知识管理系统中记录关键调查发现，并考虑应添加到开发人员指南或部署前清单中的任何模式。

资源

相关文档：

- Why you should develop a correction of error (COE)

相关视频：

- Amazon’s approach to failing successfully
- AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon

REL12-BP03 测试可扩展性和性能要求

使用负载测试等技术来验证工作负载是否满足扩展和性能要求。

在云中，可以按需为工作负载创建生产规模测试环境。可以使用云来预置一个与预期生产环境非常接近的测试环境，而不是依赖于缩减的测试环境（这可能会导致对生产行为的预测不准确）。此环境有助于您更准确地模拟应用程序面临的现实世界条件来进行测试。

除了性能测试工作外，还务必验证基础资源、扩展设置、服务配额和韧性设计在负载之下是否按预期运行。这种整体方法验证应用程序可根据需要可靠地扩展和执行，即使在最苛刻的条件下也不例外。

期望结果：即使在峰值负载下，工作负载也会保持其预期的行为。您可以主动解决随着应用程序发展和演变而可能出现的任何与性能有关的问题。

常见反模式：

- 您使用的测试环境与生产环境不太匹配。
- 您将负载测试视为单独的一次性活动，而不是部署持续集成（CI）管道不可或缺的部分。
- 您没有定义明确且可衡量的性能要求，例如响应时间、吞吐量和可扩展性目标。
- 您在不切实际或负载不足的情况下执行测试，并且无法针对峰值负载、突然激增和持续高负载进行测试。
- 您没有通过超出预期的负载限制来对工作负载进行压力测试。
- 您使用了不充分或不适当的负载测试和性能分析工具。
- 您缺乏全面的监控和警报系统来跟踪性能指标和检测异常情况。

建立此最佳实践的好处：

- 负载测试有助于您在系统投入生产之前识别其潜在的性能瓶颈。在模拟生产级流量和工作负载时，您可以确定系统可能难以处理负载的领域，例如响应时间慢、资源限制或系统故障。
- 当您在各种负载条件下测试系统时，可以更好地了解支持工作负载所需的资源需求。这些信息有助于您在资源分配方面做出明智的决策，并防止资源过度配置或配置不足。

- 要识别潜在的故障点，您可以观察工作负载在高负载条件下的性能。这些信息有助于您通过酌情实施容错机制、失效转移策略和冗余措施，来提高工作负载的可靠性和韧性。
- 您可以尽早发现并解决性能问题，这有助于避免系统中断、响应时间缓慢和用户不满意所带来的代价高昂的后果。
- 在测试期间收集的详细性能数据和分析信息有助于您排查生产环境中可能出现的与性能相关的问题。这可以加快事件响应和解决速度，从而减少对用户和组织运营的影响。
- 在某些行业，主动性能测试有助于工作负载达到合规标准，从而降低受处罚或出现法律问题的风险。

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

第一步是定义全面的测试策略，该策略涵盖扩展和性能要求的各个方面。首先，根据业务需求（例如吞吐量、延迟直方图和错误率），明确定义工作负载的服务级别目标（SLO）。接下来，设计一套测试来模拟各种负载场景，范围涵盖从平均使用量到突然激增和持续的峰值负载，并验证工作负载的行为是否符合 SLO。这些测试应自动执行，并集成到持续集成和部署管道中，以便在开发过程的早期阶段发现性能回归情况。

要有效地测试扩展和性能，请投资购买正确的工具和基础设施。这包括可以生成真实用户流量的负载测试工具、用于识别瓶颈的性能分析工具以及用于跟踪关键指标的监控解决方案。重要的是，您应该验证测试环境在基础设施和环境条件方面是否与生产环境紧密匹配，以使您的测试结果尽可能准确。为了更轻松且可靠地复制和扩展类似于生产环境的设置，请使用基础设施即代码和基于容器的应用程序。

扩展和性能测试是一个持续的过程，而不是一次性活动。实施全面的监控和警报来跟踪应用程序在生产环境中的性能，并使用这些数据来不断完善测试策略和优化工作。定期分析性能数据来识别新出现的问题，测试新的扩展策略，并实施优化以提高应用程序的效率和可靠性。当您采用迭代方法并不断从生产数据中学习时，可以验证应用程序是否能够适应不断变化的用户需求，并随着时间的推移保持韧性和最佳性能。

实施步骤

1. 制定明确且可衡量的性能要求，例如响应时间、吞吐量和可扩展性目标。这些要求应基于工作负载的使用规律、用户预期和业务需求。
2. 选择并配置负载测试工具，该工具可以准确地模仿生产环境中的负载规律和用户行为。
3. 设置与生产环境（包括基础设施和环境条件）紧密匹配的测试环境，来提高测试结果的准确性。
4. 创建涵盖各种场景的测试套件，范围从平均使用规律到峰值负载、快速激增和持续的高负载。将测试集成到持续集成和部署管道中，以便在开发过程的早期阶段发现性能回归情况。

5. 开展负载测试来模拟真实的用户流量，并了解应用程序在不同负载条件下的行为。要对应用程序进行压力测试，请超出预期负载并观察其行为，例如响应时间降级、资源耗尽或系统故障，这有助于确定应用程序的突破点并为扩展策略提供信息。通过逐步增加负载来评估工作负载的可扩展性，并衡量性能影响，来确定扩展限制并规划未来的容量需求。
6. 实施全面的监控和警报，来跟踪性能指标，检测异常，并在超过阈值时启动扩展操作或通知。
7. 持续监控和分析性能数据，来确定需要改进的领域。对测试策略和优化工作进行迭代。

资源

相关最佳实践：

- REL01-BP04 监控和管理配额
- REL06-BP01 为工作负载监控全部组件（生成）
- REL06-BP03 发送通知（实时处理和报警）

相关文档：

- 加载测试应用程序
- AWS 上的分布式负载测试
- 应用程序性能监控
- Amazon EC2 Testing Policy

相关示例：

- Distributed Load Testing on AWS (GitHub)

相关工具：

- Amazon CodeGuru Profiler
- Amazon CloudWatch RUM
- Apache JMeter
- K6
- Vegeta
- Hey

- ab
- wrk
- AWS 上的分布式负载测试

REL12-BP04 使用混沌工程测试韧性

在生产环境中或尽可能接近生产的环境中定期运行混沌试验，了解系统如何应对不利条件。

期望结果：

除了在事件期间验证已知预期工作负载行为的韧性测试之外，还可以通过以故障注入实验或注入意外负载的形式应用混沌工程，定期验证工作负载的韧性。将混沌工程和韧性测试结合起来，这可以让您提升信心，相信工作负载能够经受组件故障，并可从意外中断中恢复，而影响极小甚至没有影响。

常见反模式：

- 进行韧性设计，但不验证故障发生时工作负载如何作为一个整体运行。
- 从不在真实环境和预期负载下进行试验。
- 不将实验视为代码，也不在整个开发周期中维护实验。
- 不将混沌实验作为 CI/CD 管道的一部分，也不在部署之外运行。
- 在确定要对哪些故障进行试验时，没有想到使用过去的事件后分析。

建立此最佳实践的好处：注入故障来验证工作负载的韧性，这可以让您提升信心，相信韧性设计的恢复程序会在真正发生故障的情况下发挥作用。

在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

利用混沌工程，您的团队能够在服务提供商、基础设施、工作负载和组件级别，以可控的方式不断注入真实世界的干扰（模拟），而对客户的影响极小甚至没有影响。其可让团队从故障中学习，观察、测量和提高工作负载的韧性，并验证在发生事件时，系统会发出警报并通知团队。

当持续执行时，混沌工程可突出工作负载中的缺陷，这些缺陷若不加以解决，可能会对可用性和运营产生负面影响。

> **Note**
>
> 混沌工程是在系统上进行实验的学科，目的是建立对系统抵御生产环境中失控条件的能力以及信心。– 混沌工程原则

如果系统能够经受住这些干扰，则应将混沌实验作为自动回归测试来加以维护。这样一来，应将混沌实验作为系统开发生命周期（SDLC）的一部分，以及作为 CI/CD 管道的一部分来执行。

为了确保工作负载能够经受住组件故障，请在实验中注入实际事件。例如，对 Amazon EC2 实例的丢失或主 Amazon RDS 数据库实例的失效转移进行试验，并验证工作负载没有受到影响（或影响极小）。使用组件故障的组合来模拟可能因可用区中断而引起的事件。

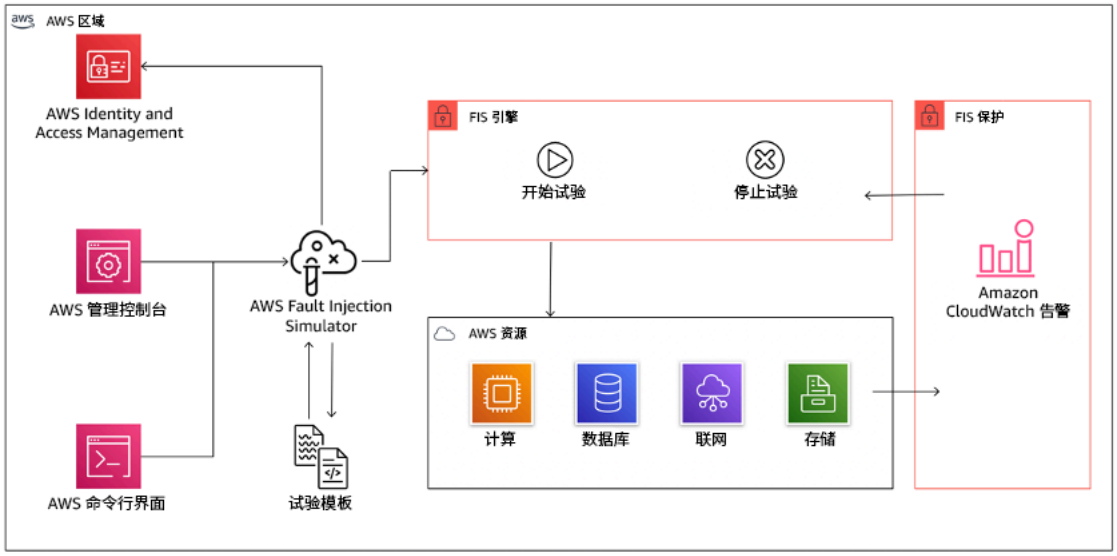
对于应用程序级故障（如崩溃），您可以从内存和 CPU 耗尽等压力源开始。

为了验证因间歇性网络中断而引发的外部依赖项的回退或失效转移机制，组件应通过在指定时间段（从几秒到几小时不等）内阻止对第三方提供商的访问来模拟此类事件。

其他降级模式可能会影响功能的使用并降低响应速度，这通常会导致服务中断。性能下降的常见原因是，关键服务的延迟增加以及网络通信不可靠（丢包）。对于这些故障（包括延迟、丢弃的消息和 DNS 故障等网络效应）的实验，可能包括无法解析名称、无法访问 DNS 服务或无法建立与依赖服务的连接。

混沌工程工具：

AWS Fault Injection Service（AWS FIS）是一项完全托管式服务，用于运行故障注入实验，而这些实验可用作 CD 管道的一部分，或在管道之外使用。AWS FIS 是在混沌工程 GameDay 活动期间使用的一个不错选择。该服务支持在不同类型的资源中同时引入故障，包括 Amazon EC2、Amazon Elastic Container Service（Amazon ECS）、Amazon Elastic Kubernetes Service（Amazon EKS）和 Amazon RDS 等资源。这些故障包括终止资源、强制失效转移、对 CPU 或内存施加压力、节流、延迟和数据包丢失。由于该服务已与 Amazon CloudWatch 警报集成，您可以设置停止条件作为防护机制，在实验导致意外影响时回滚。



AWS Fault Injection Service 与 AWS 资源集成，让您能够为工作负载运行故障注入实验。

故障注入实验也有多种第三方选项。其中包括开源工具（例如 Chaos Toolkit、Chaos Mesh 和 Litmus Chaos），以及商用工具（例如 Gremlin）。为了扩大可在 AWS 上注入的故障范围，AWS FIS 与 Chaos Mesh 和 Litmus Chaos 集成，让您能够在多个工具之间协调故障注入工作流程。例如，您可以使用 Chaos Mesh 或 Litmus 故障对容器组的 CPU 运行压力测试，同时使用 AWS FIS 故障操作终止随机选择的集群节点百分比。

实施步骤

1. 确定哪些故障要用于实验。

   评测工作负载的设计是否具有韧性。这种设计使用 Well-Architected Framework 的最佳实践创建，考虑到了基于关键依赖项、以往事件、已知问题以及合规性要求的风险。列出每个旨在保持韧性的设计元素及其旨在缓解的故障。有关创建此类列表的更多信息，请参阅《Operational Readiness Review》白皮书，了解如何创建流程来防止以往事件再次发生。故障模式与影响分析（FMEA）流程提供了一个框架，可对故障及其对工作负载的影响执行组件级分析。Adrian Cockcroft 在《Failure Modes and Continuous Resilience》中更详细地概述了 FMEA。

2. 为每个故障指定一个优先级。

   先进行粗略的分类，如高、中或低。要评测优先级，请考虑故障的频率和故障对整体工作负载的影响。

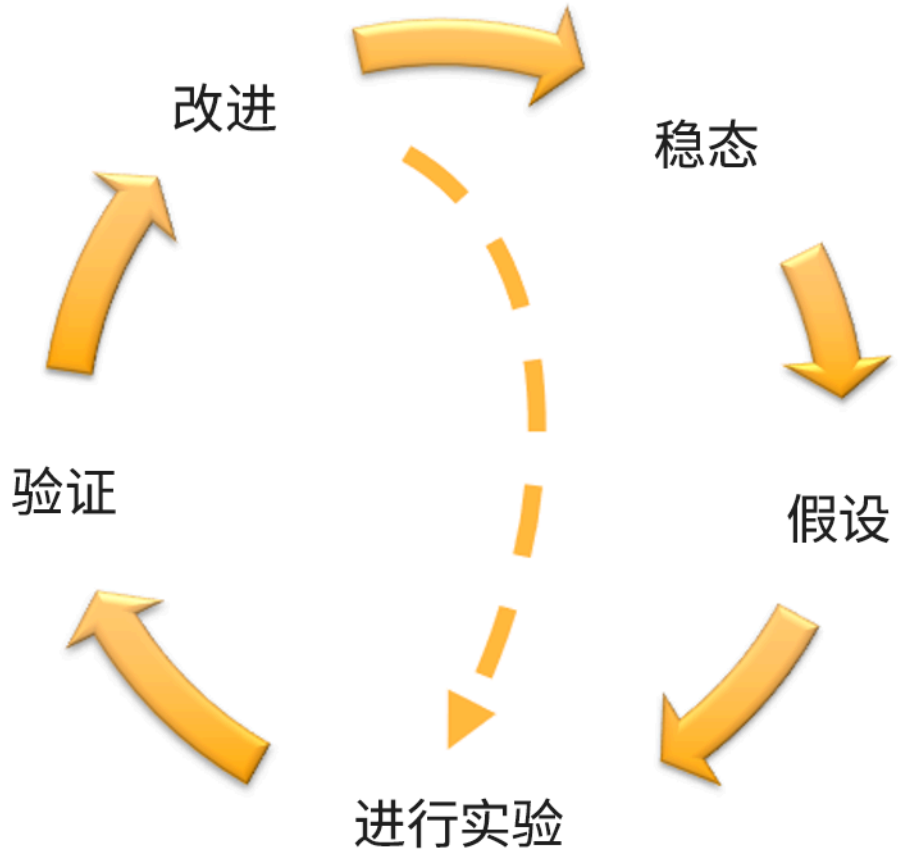
   考虑给定故障的频率时，请分析此工作负载的以往数据（如有）。如果没有以往数据，则使用在类似环境中运行的其他工作负载的数据。

考虑给定故障的影响时，故障的范围越大，影响通常也越大。还要考虑工作负载设计和目的。例如，访问源数据存储的能力对于进行数据转换和分析的工作负载至关重要。在这种情况下，您要确定访问故障以及节流访问和延迟插入等实验的优先级。

事件后分析是了解故障模式的频率和影响的良好数据来源。

使用指定的优先级来确定首先对哪些故障进行实验，以及开发新的故障注入实验的顺序。

3. 对于执行的每项实验，请遵循下图中的混沌工程和持续韧性飞轮。



Adrian Hornsby 采用科学方法制作的混沌工程和持续韧性飞轮。

a. 将稳定状态定义为指示正常行为的工作负载的一些可测量输出。

如果工作负载运行可靠且符合预期，则显示为稳定状态。因此，定义稳定状态之前，请验证工作负载是否运行状况良好。稳定状态并不一定意味着故障发生时对工作负载没有影响，因为一定百

分比的故障可能在可接受的范围内。稳定状态是您将在实验期间观察到的基线，如果下一步中定义的假设结果不符合预期，则会突出显示异常。

例如，可以将某个支付系统的稳定状态定义为处理 300 TPS，成功率为 99%，且往返时间为 500 毫秒。

b. 形成一个关于工作负载如何应对故障的假设。

一个好的假设是基于工作负载预计如何缓解故障来保持稳定状态。该假设指出，如果发生特定类型的故障，系统或工作负载将继续保持稳定状态，因为该工作负载在设计时就有特定缓解措施。应在假设中具体说明特定的故障类型和缓解措施。

假设可以使用以下模板（但其他措辞也可以接受）：

> **Note**
>
> 如果发生####，则######工作负载将########，以此维持#########。

例如：

- 如果 Amazon EKS 节点组中 20% 的节点出现故障，则 Transaction Create API 将在不到 100 毫秒的时间内继续处理 99% 的请求（稳定状态）。Amazon EKS 节点将在五分钟内恢复，容器组将在实验开始后八分钟内得到调度并处理流量。警报将在三分钟内发出。
- 如果单个 Amazon EC2 实例发生故障，订单系统的弹性负载均衡运行状况检查会让弹性负载均衡仅向剩余的运行状况良好的实例发送请求，而 Amazon EC2 Auto Scaling 将替换故障实例，从而保持服务器端（5xx）错误增长率低于 0.01%（稳定状态）。
- 如果主 Amazon RDS 数据库实例发生故障，则供应链数据收集工作负载将失效转移并连接到备用 Amazon RDS 数据库实例，以保持不到 1 分钟的数据库读写错误（稳定状态）。

c. 通过注入故障来进行实验。

默认情况下，实验应具有故障保护机制，可承受工作负载。如果知道工作负载将发生故障，则不要进行实验。混沌工程应该用于寻找已知的不确定因素或未知的不确定因素。已知的不确定因素是您知道但不完全理解的东西，而未知的不确定因素是您既不知道也不完全理解的东西。对已知发生了故障的工作负载进行实验，并不会为带来新的见解。您应该仔细规划实验，明确影响范围，并提供一种可在出现意外动荡时应用的回滚机制。如果尽职调查表明工作负载应该能经受住实验，才继续这项实验。有几种注入故障的选项。对于 AWS 上的工作负载，AWS FIS 提供了许多称为操作的预定义故障模拟。您还可以定义在 AWS FIS 中运行的自定义操作（使用 AWS Systems Manager 文档）。

我们不鼓励使用自定义脚本进行混沌实验，除非这些脚本能够了解工作负载的当前状态，能够发出日志，并在可能的情况下提供回滚和停止条件的机制。

支持混沌工程的有效框架或工具集应跟踪实验的当前状态，发出日志，并提供回滚机制以支持实验的受控执行。从 AWS FIS 这样的成熟服务开始，该服务支持您在明确定义的范围内和安全机制下进行实验，可在实验引入了意外动荡的情况下回滚实验。要了解更多使用 AWS FIS 的实验，另请参阅 Resilient and Well-Architected Apps with Chaos Engineering 实验室。此外，AWS Resilience Hub 将分析工作负载，并创建您可以选择在 AWS FIS 中实施和执行的实验。

> **Note**
>
> 对于每项实验，要清楚地了解其范围及影响。建议首先在非生产环境中模拟故障，再在生产环境中运行。

应使用实际负载，通过金丝雀部署在生产环境中进行实验，尽可能同时启动控制和实验系统部署。在非高峰时间进行实验是一种很好的做法，可以减小首次在生产环境中试验时的潜在影响。此外，如果使用实际的客户流量会带来太大的风险，您可以在生产基础设施上针对控制和实验部署使用合成流量进行实验。当不能使用生产环境时，在尽可能接近生产环境的预生产环境中进行实验。

您必须建立和监控防护机制，确保实验对生产流量或其他系统的影响不会超过可接受的限度。建立停止条件，以便在实验达到您定义的防护机制指标的阈值时停止实验。这应该包括工作负载的稳定状态指标，以及针对您要注入故障的组件的指标。综合监控（也称为用户金丝雀）是一个通常应作为用户代理包含的指标。AWS FIS 的停止条件应纳入实验模板中，每个模板最多可以有五个停止条件。

混沌的原则之一是尽量缩小实验范围并减小其影响：

虽然必须考虑到一些短期负面影响，但混沌工程师有责任和义务确保实验产生的影响极小且可控。

验证范围和潜在影响的一种方法是首先在非生产环境中进行实验，确认停止条件的阈值是否在实验期间按预期激活，以及是否具有可观测性来捕获异常，而不是直接在生产环境中进行实验。

运行故障注入实验时，确保所有责任方均知情。与适当的团队（如运营团队、服务可靠性团队和客户支持团队）沟通，让他们知道实验将在何时运行以及预期会发生什么。为这些团队提供沟通工具，以便在他们看到任何不利影响时通知进行实验的人员。

必须将工作负载及其底层系统恢复到最初的已知良好状态。通常，工作负载的韧性设计会自我修复。但一些故障设计或失败实验可能会让工作负载处于意外的失败状态。在实验结束时，您必须意识到这一点，并恢复工作负载和系统。使用 AWS FIS，您可以在操作参数中设置回滚配置（也称为后期操作）。后置操作可将目标返回到操作运行之前的状态。无论是自动执行（如使用 AWS FIS）还是手动执行，这些后期操作都应包含在描述如何检测和处理故障的行动手册中。

d. 验证假设。

混沌工程原则为如何验证工作负载的稳定状态提供了以下指导：

关注系统的可测量输出，而不是系统的内部属性。短时间内对该输出的测量构成了系统稳态的代理。整个系统的吞吐量、错误率和延迟百分比都可以是代表稳态行为的相关指标。通过关注实验过程中的系统行为模式，混沌工程验证系统确实在工作，而不是试图验证它如何工作。

在之前的两个示例中，我们包括了服务器端（5xx）错误增长率低于 0.01% 和数据库读写错误持续时间不到 1 分钟的稳态指标。

5xx 错误是一个很好的指标，因为此类错误是工作负载客户端会直接经历的故障模式的结果。数据库错误测量适合作为故障的直接结果，但是还应补充一个客户端影响测量，例如失败的客户请求或向客户端显示的错误。此外，在工作负载客户端直接访问的任何 API 或 URI 上包含一个综合监控（也称为用户金丝雀）。

e. 改进工作负载设计，提高韧性。

如果未保持稳定状态，则调查如何改进工作负载设计来缓解故障，并应用 AWS Well-Architected 可靠性支柱的最佳实践。可以在 AWS Builder's Library 中找到其他指导和资源，其中包含有关如何改进运行状况检查或在应用程序代码中结合采用重试与回退的文章，等等。

实施这些更改后，再次进行实验（如混沌工程飞轮中的虚线所示），确定更改的效果。如果验证步骤表明假设成立，则工作负载将处于稳定状态，循环将继续。

4. 定期进行实验。

混沌实验是一个循环，作为混沌工程的一部分，应定期进行实验。在工作负载满足实验的假设后，实验应实现自动化，作为 CI/CD 管道的回归部分持续运行。要了解如何做到这一点，请参阅关于如何使用 AWS CodePipeline 进行 AWS FIS 实验的博客。这个关于反复在 CI/CD 管道中进行 AWS FIS 实验的实验室让您能够动手实践。

故障注入实验也是 GameDay 活动的一部分（请参阅 REL12-BP05 定期进行 GameDay 活动）。GameDay 活动会模拟故障或事件，以便验证系统、流程和团队的响应。其目的是实际执行团队在发生意外事件时会执行的操作。

5. 捕获和存储实验结果。

   必须捕获并持久保存故障注入实验的结果。包括所有必要的数据（如时间、工作负载和条件），以便以后能够分析实验结果和趋势。结果示例可能包括控制面板的屏幕截图、从指标数据库进行的 CSV 转储，或实验中事件和观察结果的手写记录。使用 AWS FIS 进行实验记录可作为这种数据捕获的一部分。

资源

相关最佳实践：

- REL08-BP03 将韧性测试作为部署的一部分进行集成
- REL13-BP03 测试灾难恢复实施以验证实施效果

相关文档：

- 什么是 AWS Fault Injection Service ？
- 什么是 AWS Resilience Hub ？
- 混沌工程原则
- Chaos Engineering: Planning your first experiment
- Resilience Engineering: Learning to Embrace Failure
- 混沌工程案例
- 避免在分布式系统中回退
- 用于混沌实验的金丝雀部署

相关视频：

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301)

相关工具：

- AWS Fault Injection Service
- AWS Marketplace：Gremlin 混沌工程平台

- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP05 定期进行 GameDay 活动

安排 GameDay 来定期练习旨在应对影响工作负载的事件和损害的过程。让负责处理生产场景的团队参与进来。这些练习有助于强制实施相关措施，来防止生产事件对用户造成影响。当您在现实条件下实践响应过程时，可以在实际事件发生之前发现并解决任何差距或弱点。

GameDay 活动会模拟类似于生产的环境中的事件，以便测试系统、流程和团队的响应。其目的是执行团队在实际发生事件时会执行的相同操作。这些练习有助于您了解可以从哪些方面作出改进，并有助于培养组织在处理各种事件和损害方面的经验。这些练习应该定期开展，这样，团队就知道如何建立根深蒂固的应对习惯。

GameDay 可让团队做好准备，以便更充满信心地处理生产事件。经过良好练习的团队更有能力快速检测和应对各种场景。这可以显著改善就绪状态和韧性态势。

期望结果：您在一致、有计划的基础上运行韧性 GameDay。这些 GameDay 被视为业务运营中正常和预期的组成部分。您的组织已经建立了备灾文化，当出现生产问题时，团队已经做好了充分的准备，可以有效地做出响应，高效地解决问题并减轻对客户的影响。

常见反模式：

- 您记录过程，但从不练习这些过程。
- 您不让业务决策者参与测试练习。
- 您开展了 GameDay，但没有通知所有相关的利益相关者。
- 您只关注技术故障，但不涉及业务利益相关者。
- 您未将从 GameDay 中吸取的经验教训纳入恢复过程。
- 您将失败或错误归咎于团队。

建立此最佳实践的好处：

- 增强响应技能：在 GameDay，团队在模拟的事件中练习其职责并测试其沟通机制，从而在生产环境中做出更加协调和高效的响应。

- 识别和解决依赖关系：复杂的环境通常涉及各种系统、服务和组件之间错综复杂的依赖关系。GameDay 有助于您识别和解决这些依赖关系，并验证运行手册过程是否正确涵盖了关键系统和服务，以及是否可以及时纵向扩展或恢复这类系统和服务。
- 培养韧性文化：GameDay 有助于培养组织内部的韧性思维。当您让跨职能团队和利益相关者参与时，这些练习可以提高整个组织对韧性重要性的认识、协作和共同理解。
- 持续改进和适应：定期的 GameDay 有助于您不断评测和调整韧性策略，从而使这些策略在不断变化的环境中保持相关性和有效性。
- 增强对系统的信心：成功的 GameDay 有助于您树立信心，确信系统能够承受中断并从中恢复。

在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

设计并实施了必要的韧性措施后，请开展 GameDay 来验证生产中的一切是否按计划进行。GameDay，尤其是第一个 GameDay，应让所有团队成员都参与，并应事先向所有利益相关者和参与者告知日期、时间和模拟场景。

在 GameDay 期间，参与的团队会根据规定的过程模拟各种事件和潜在的场景。参与者密切监控和评测这些模拟事件的影响。如果系统按设计运行，则应激活自动检测、扩展和自我修复机制，且对用户几乎没有影响。如果团队观察到任何负面影响，他们就会回滚测试，并通过相应运行手册中记载的自动手段或手动干预来纠正已发现的问题。

要持续提高韧性，记录和吸取经验教训至关重要。该过程是一个反馈循环，它系统化地从 GameDay 捕获见解，并使用这些见解来增强系统、流程和团队能力。

为协助您重现系统组件或服务可能意外出现故障的现实场景，请将模拟故障作为 GameDay 练习注入。团队可以在受控的环境中测试其系统的韧性和容错能力，并模拟其事件响应和恢复流程。

借助 AWS，可以使用基础设施即代码，通过生产环境的副本来开展 GameDay。通过此过程，可以在与生产环境非常相似的安全环境中进行测试。考虑使用 AWS Fault Injection Service来创建不同的故障场景。使用诸如 Amazon CloudWatch 和 AWS X-Ray 之类的服务来监控 GameDay 期间的系统行为。使用 AWS Systems Manager 来管理和运行行动手册，并使用 AWS Step Functions 来编排重复出现的 GameDay 工作流程。

实施步骤

- 制定 GameDay 计划：制定结构化计划来定义 GameDay 的频率、范围和目标。让关键利益相关者和主题专家参与规划和实施这些练习。

- 为 GameDay 做好准备：
  1. 确定主要的业务关键服务，这些服务是 GameDay 的重点。对支持这些服务的人员、流程和技术进行编目和映射。
  2. 制定 GameDay 的日程，让相关团队做好参与事件的准备。准备好自动化服务来模拟计划的场景并运行相应的恢复流程。诸如 AWS Fault Injection Service、AWS Step Functions 和 AWS Systems Manager 等 AWS 服务有助于您自动实施 GameDay 的各个方面，例如注入故障和启动恢复操作。
- 运行模拟：在 GameDay，运行计划的场景。观察并记录人员、流程和技术对模拟事件的反应。
- 开展练习后回顾：GameDay 结束后，召开回顾会议来回顾所吸取的教训。确定需要改进的领域以及改善运营韧性所需的任何措施。记录您的调查发现，并跟踪任何必要的更改，来增强韧性策略和完成准备工作。

资源

相关最佳实践：

- REL12-BP01 使用行动手册调查故障
- REL12-BP04 使用混沌工程测试韧性
- OPS04-BP01 确定关键绩效指标
- OPS07-BP03 使用运行手册执行程序
- OPS10-BP01 使用流程来管理事件、意外事件和问题

相关文档：

- 什么是 AWS GameDay？
- AWS Well-Architected 概念 - GameDay

相关视频：

- AWS re:Invent 2023 - Practice like you play: How Amazon scales resilience to new heights

相关示例：

- AWS Workshop - Navigate the storm: Unleashing controlled chaos for resilient systems
- Build Your Own Game Day to Support Operational Resilience

REL 13. 如何规划灾难恢复（DR）？

拥有适当的备份和冗余工作负载组件是灾难恢复策略的开始。RTO 和 RPO 是您恢复工作负载的目标。根据业务需求设置这些目标。通过实施策略来实现这些目标，同时考虑工作负载资源和数据的位置和功能。中断概率和恢复成本也是关键因素，有助于了解为工作负载提供灾难恢复的业务价值。

最佳实践

- REL13-BP01 定义停机和数据丢失的恢复目标
- REL13-BP02 使用定义的恢复策略来实现恢复目标
- REL13-BP03 测试灾难恢复实施以验证实施效果
- REL13-BP04 管理灾难恢复站点或区域的配置偏差
- REL13-BP05 自动执行恢复

REL13-BP01 定义停机和数据丢失的恢复目标

故障可通过多种方式影响您的业务。首先，故障可能导致服务中断（停机时间）。其次，故障可能导致数据丢失、不一致或过时。为了指导您如何应对故障和从故障中恢复，请为每个工作负载定义恢复时间目标（RTO）和恢复点目标（RPO）。恢复时间目标（RTO）是指服务中断和恢复服务之间可接受的最大延迟。恢复点目标（RPO）是指在上一个数据恢复点之后可接受的最长时间。

期望结果：根据技术考虑和业务影响，每个工作负载都有指定的 RTO 和 RPO。

常见反模式：

- 您尚未指定恢复目标。
- 您选择任意的恢复目标。
- 您选择的恢复目标过于宽松并且不符合业务目标。
- 您尚未评估停机时间和数据丢失的影响。
- 您选择不切实际的恢复目标，如零恢复时间或零数据丢失，这对工作负载配置而言可能无法实现。
- 您选择的恢复目标比实际业务目标更苛刻。这会迫使实施恢复的成本和复杂度超出工作负载所需。
- 您选择的恢复目标与相关工作负载的恢复目标不兼容。
- 您没有考虑监管和合规要求。

建立此最佳实践的好处：在为工作负载设置 RTO 和 RPO 时，可以根据业务需求制定明确且可衡量的恢复目标。一旦设定了这些目标，就可以创建专为实现这些目标而量身定制的灾难恢复（DR）计划。

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

构造一个矩阵或工作表，来协助指导灾难恢复规划。在矩阵中，创建不同的工作负载类别或层级，所依据的是这些类别或层级的业务影响（例如严重、高、中和低），以及每个工作负载类别或层级关联的目标 RTO 和 RPO。以下矩阵提供了您可以遵循的示例（请注意，RTO 和 RPO 值可能有所不同）：

灾难恢复矩阵						
		恢复点目标				
		少于 1 分钟	少于 1 小时	少于 6 小时	少于 1 天	多于 1 天
恢复时间目标	少于 10 分钟	严重	严重	高	中	中
	少于 2 小时	严重	高	中	中	低
	少于 8 小时	高	中	中	低	低
	少于 24 小时	中	中	低	低	低
	多于 24 小时	中	低	低	低	低

灾难恢复矩阵示例

对于每个工作负载，调查和了解停机时间和数据丢失对业务的影响。影响通常会随着停机时间和数据丢失而增加，但影响的形式可能会因工作负载类型而异。例如，长达一小时的停机时间可能影响很小，但在一小时之后，影响可能会迅速加剧。影响可能以多种形式呈现，包括财务影响（如收入损失）、声誉影响（包括失去客户信任）、运营影响（如错过工资发放或生产率下降）和监管风险。完成后，将工作负载分配到相应的层级。

分析故障所产生的影响时，请考虑以下问题：

1. 在对业务产生不可接受的影响之前，工作负载可以保持不可用的最长时间是多少？
2. 工作负载中断将对业务产生多大影响以及会产生什么样的影响？ 考虑各种影响，包括财务、声誉、运营和监管。
3. 在对业务造成不可接受的影响之前，可以丢失或无法恢复的最大数据量是多少？
4. 能否从其它来源重新创建丢失的数据（也称为派生的 数据）？ 如果是，还要考虑用于重新创建工作负载数据的所有源数据的 RPO。
5. 此工作负载所依赖的工作负载（下游）的恢复目标和可用性期望是什么？ 根据工作负载下游依赖关系的恢复能力，工作负载的目标必须是可实现的。考虑可能的下游依赖关系解决办法或缓解措施，以提高此工作负载的恢复能力。

6. 依赖于此工作负载的工作负载（上游）的恢复目标和可用性期望是什么？ 上游工作负载目标可能要求此工作负载具有比最初看起来更严格的恢复能力。
7. 根据事件的类型，是否有不同的恢复目标？ 例如，根据事件影响的是一个可用区还是整个区域，您可能有不同的 RTO 和 RPO。
8. 恢复目标在一年中的某些事件或时期是否会发生变化？ 例如，在假日购物季、体育赛事、特别促销和新产品发布期间，您可能有不同的 RTO 和 RPO。
9. 恢复目标如何与您可能拥有的任何业务线和组织灾难恢复策略保持一致？
10. 是否需要考虑法律或合同后果？ 例如，根据合同，您是否有义务提供具有给定 RTO 或 RPO 的服务？ 不履行这些义务会受到什么处罚？
11. 您是否需要维护数据完整性来满足监管或合规性要求？

以下工作表有助于您评估每项工作负载。您可以修改此工作表来满足具体的需求，例如添加附加问题。

步骤 2：主要问题	适用于工作负载？	工作负载 RTO	工作负载 RPO	RTO 调整。	RPO 调整。	说明
[1] 工作负载可以停止的最长时间						以从中断开始到恢复的时间进行衡量
[2] 可以丢失的最大数据量						以从最后一个已知的可恢复数据集算起的时间进行衡量
[3a] 上游依赖关系						输入最严格的上游恢复目标
[3b] 下游依赖关系						输入最不严格的下游恢复目标
[3a] 协调后的上游依赖关系						如果上游值小于当前值，而下游值大于当前值，则使用依赖关系进行协调，
[3b] 协调后的下游依赖关系						并在此处输入协调后的值
[3] 依赖关系						降低值以满足上游依赖关系，或者根据下游依赖关系的能力提高值
步骤 2：其他问题						指出问题是否适用。如果问题不适用，则跳过
基本 RTO/RPO						将 RTO 和 RPO 值从上方向下移到此处
[4] 中断类型	[ ]Y / [ ]N					输入要求最严格的事件类型的恢复目标
[5] 基于时间的具体目标	[ ]Y / [ ]N					输入要求最严格的恢复时间目标
[6] 客户中断	[ ]Y / [ ]N					根据停机时间或数据丢失情况绘制受影响客户的图表。 根据客户影响输入允许的最大的 RTO 和 RPO
[7] 声誉影响	[ ]Y / [ ]N					与业务部门合作，根据对声誉的影响确定最大的 RTO 和 RPO
[8] 运营影响	[ ]Y / [ ]N					根据运营影响输入最大的 RTO 和 RPO
[9] 组织一致性	[ ]Y / [ ]N					根据 LOB 和组织要求，输入此类工作负载的最大 RTO 和 RPO
[10] 合同义务	[ ]Y / [ ]N					根据合同义务输入最大的 RTO 和 RPO
[11] 监管合规性	[ ]Y / [ ]N					根据适用的监管合规性输入最大的 RTO 和 RPO
基于其他问题的目标						从问题 4-11 中取最小值（更严格的值）并在此处输入
调整后的目标						如果无法满足上述目标，请与利益相关者一起放松约束， 并在此处输入新的最小值
调整后的 RTO/RPO						输入基本 RPO/RTO 值或调整后的目标值，以较低者为准
步骤 3						
映射到预定义的类别或层						将这两个值向下调整（更严格），以符合最接近的定义层

工作表

实施步骤

1. 确定业务利益相关方和负责每个工作负载的技术团队，并与他们互动。

2. 对组织中的工作负载影响创建严重性类别或层级。类别示例包括：严重、高、中和低。对于每个类别，请选择反映您的业务目标和要求的 RTO 和 RPO。
3. 将您在上一步创建的影响类别之一分配给每个工作负载。要决定工作负载如何映射到某个类别，请考虑工作负载对业务的重要性，以及中断或数据丢失的影响，并使用上述问题来提供指导。这将为每个工作负载生成一个 RTO 和 RPO。
4. 考虑上一步中确定的每个工作负载的 RTO 和 RPO。让工作负载的业务和技术团队参与进来，以确定是否应调整目标。例如，业务利益相关者可能确定需要更严格的目标。或者，技术团队可能决定应修改目标，以使这些目标能够在可用的资源和技术限制下得以实现。

资源

相关最佳实践：

- REL09-BP04 定期执行数据恢复以验证备份完整性和流程
- REL12-BP01 使用行动手册调查故障
- REL13-BP02 使用定义的恢复策略来实现恢复目标
- REL13-BP03 测试灾难恢复实施以验证实施效果

相关文档：

- AWS Architecture Blog: Disaster Recovery 系列博客文章
- AWS 上工作负载的灾难恢复：云中的恢复（AWS 白皮书）
- Managing resiliency policies with AWS Resilience Hub
- APN 合作伙伴：可帮助进行灾难恢复的合作伙伴
- AWS Marketplace：可用于灾难恢复的产品

相关视频：

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS 上工作负载的灾难恢复

REL13-BP02 使用定义的恢复策略来实现恢复目标

定义可满足工作负载恢复目标的灾难恢复（DR）策略。选择一种策略，例如备份和还原、备用（主动/被动）或主动/主动。

期望结果：每个工作负载都有已定义且实施的灾难恢复策略，可让该工作负载实现灾难恢复目标。工作负载之间的灾难恢复策略利用可重用模式（例如前面所述的策略）。

常见反模式：

- 为具有类似灾难恢复目标的工作负载实施不一致的恢复过程。
- 在发生灾难时临时实施灾难恢复策略。
- 没有针对灾难恢复的计划。
- 恢复期间依赖于控制面板操作。

建立此最佳实践的好处：

- 通过定义恢复策略，您可以使用常用工具和测试步骤。
- 使用定义的恢复策略可改善团队之间的知识共享情况，并在他们自己的工作负载上实施灾难恢复。

在未建立这种最佳实践的情况下暴露的风险等级：高。若没有经过计划、实施和测试的灾难恢复策略，发生灾难时就不太可能实现恢复目标。

实施指导

灾难恢复策略依赖于在主位置无法运行工作负载的情况下，在恢复站点中支持工作负载的能力。最常见的恢复目标是 RTO 和 RPO，如 REL13-BP01 定义停机和数据丢失的恢复目标中所述。

跨单个 AWS 区域内的多个可用区（AZ）的灾难恢复策略可以缓解火灾、洪水和重大停电等灾难事件造成的影响。如果需要实施保护措施，为工作负载无法在给定 AWS 区域中运行这种不太可能发生的事件提供保护，您可以使用跨多个区域的灾难恢复策略。

在跨多个区域构建灾难恢复策略时，您应该选择以下某个策略。这些策略按成本和复杂性升序排列，按 RTO 和 RPO 降序排列。恢复区域指的是 AWS 区域，而不是用于工作负载的主要区域。

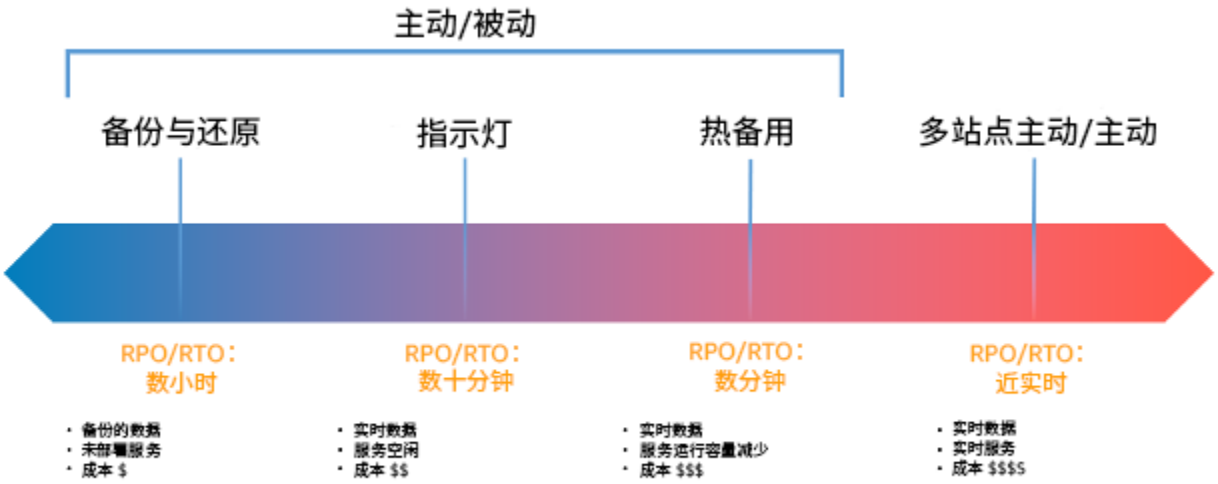


图 17：灾难恢复（DR）策略

- 备份和还原（RPO 以小时为单位，RTO 为 24 小时或更短时间为单位）：将数据和应用程序备份到恢复区域。使用自动或连续备份可以实现时间点故障恢复（PITR），在某些情况下，可以将 RPO 降低到 5 分钟。在发生灾难的情况下，您将部署基础设施（使用基础设施即代码来减少 RTO）、部署代码并还原备份的数据，以便在恢复区域从灾难中恢复。
- 指示灯（RPO 以分钟为单位，RTO 以数十分钟为单位）：在恢复区域中预置核心工作负载基础设施的副本。将数据复制到恢复区域并在其中创建数据备份。支持数据复制和备份所需的资源（如数据库和对象存储）始终处于启用状态。其他元素（如应用程序服务器或无服务器计算）未部署，但可以在需要时使用必要的配置和应用程序代码创建。
- 温备用（RPO 以秒为单位，RTO 以分钟为单位）：保证在恢复区域中始终运行缩减但功能齐全版本的工作负载。业务关键型系统是完全重复且始终可用的系统，只是其队列的规模经过缩减。数据在恢复区域中复制并留存。在需要恢复时，系统会快速纵向扩展来处理生产负载。温备用的规模越大，RTO 和控制面板依赖度就越低。当完全扩展时，这称为热备用。
- 多区域（多站点）主动-主动（RPO 接近于零，RTO 可能为零）：工作负载部署到多个 AWS 区域，并且主动处理来自这些区域的流量。此策略要求您跨区域同步数据。必须避免或处理在两个不同区域副本中写入同一记录可能引起的冲突，因为这种情况很复杂。数据复制对于数据同步非常有用，并且可以防止某些类型的灾难，但是它不能防止数据损坏或破坏，除非您的解决方案还包含时间点故障恢复选项。

> **Note**
>
> 指示灯和温备用之间的差异有时难以区分。两者都在恢复区域中包含一个环境，其中具有主区域资产的副本。区别在于，如果不先采取额外措施，指示灯无法处理请求，而温备用可以立即处理流量（容量级别降低）。指示灯将要求您启用服务器，可能需要部署额外的（非核心）基础设施并纵向扩展，而温备用只需要您纵向扩展（所有内容都已部署并运行）。根据 RTO 和 RPO 需求在两者之间进行选择。
>
> 如果需要考虑成本，并且希望实现与温备用策略中定义的类似 RPO 和 RTO 目标时，您可以考虑云原生解决方案（例如 AWS Elastic Disaster Recovery），此类解决方案会采用指示灯方法并提供改进的 RPO 和 RTO 目标。

实施步骤

1. 确定可满足此工作负载恢复要求的灾难恢复策略。

   选择灾难恢复策略是在减少停机时间和数据丢失（RTO 和 RPO）与策略实施的成本和复杂性之间进行权衡。应该避免实施比所需策略更严格的策略，因为这会产生不必要的成本。

   例如，在下图中，企业已经确定了自己允许的最大 RTO 以及可在服务恢复策略上花费的费用限额。考虑到企业目标，指示灯或温备用这样的灾难恢复策略将同时满足 RTO 和成本标准。

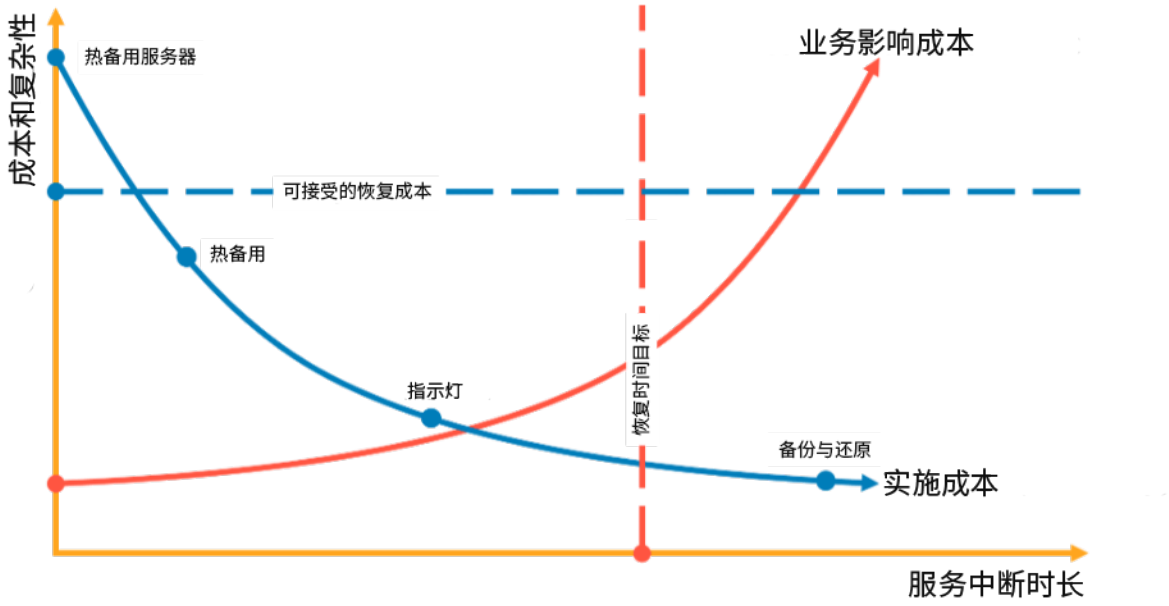


图 18：根据 RTO 和成本选择灾难恢复策略

如需了解更多信息，请参阅业务连续性计划（BCP）。

2. 查看如何实施所选灾难恢复策略的模式。

这一步是了解如何实施所选策略。这些策略可以解释为使用多个 AWS 区域作为主要站点和恢复站点。不过，您也可以选择使用单个区域内的多个可用区作为灾难恢复策略，这将利用多个策略的元素。

在后续步骤中，您可以对特定的工作负载应用策略。

备份和还原

备份和还原是实施起来最简单的策略，但需要更多时间和工作来恢复工作负载，导致更高的 RTO 和 RPO。最好的做法是，始终备份数据并将数据备份复制到另一个站点（例如另一个 AWS 区域）。

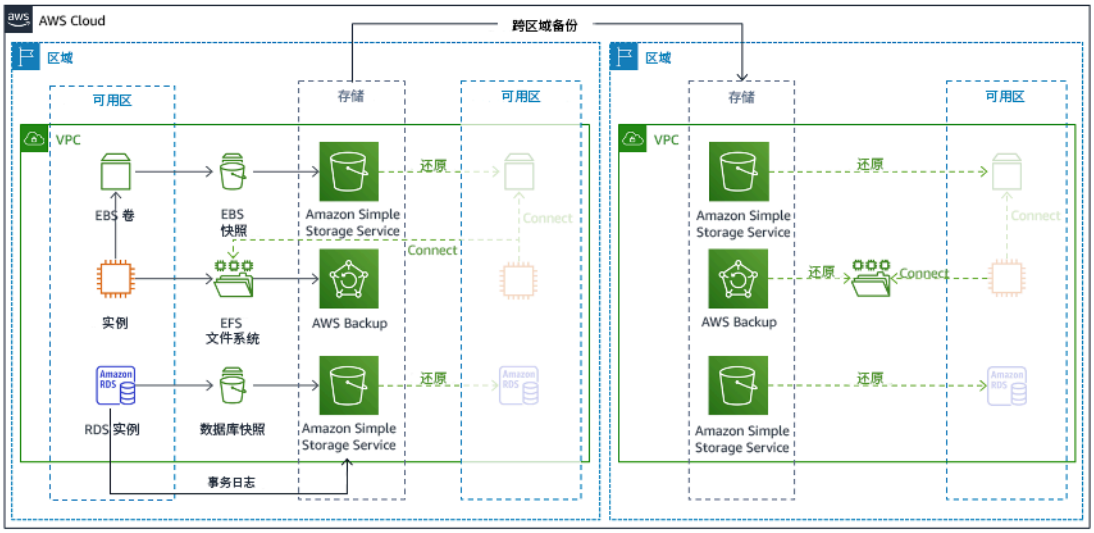


图 19：备份和还原架构

有关此策略的更多详细信息，请参阅《Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery》。

指示灯

利用指示灯方法，您可以将数据从主要区域复制到恢复区域。用于工作负载基础设施的核心资源部署在恢复区域中，但仍需要额外的资源和所有依赖项，才能让此恢复区域成为功能堆栈。例如，图 20 中就没有部署计算实例。

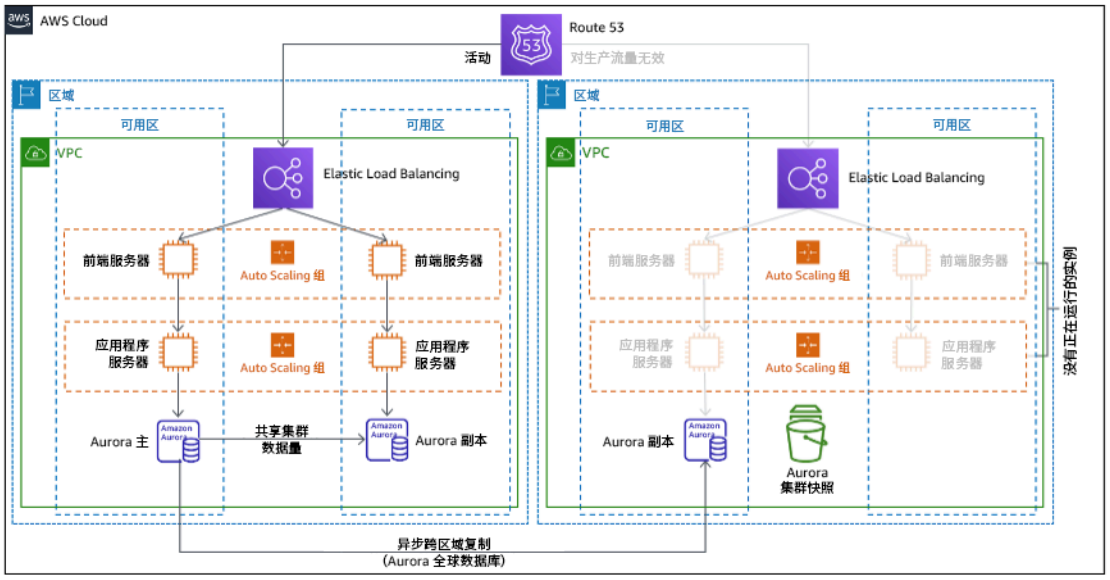


图 20：指示灯架构

有关此策略的更多详细信息，请参阅《Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby》。

温备用

温备用方法涉及到确保在另一个区域中存在生产环境的规模缩减但功能齐全的副本。这种方法扩展了指示灯概念并减少了恢复时间，因为工作负载始终在另一个区域中运行。如果以全部容量部署恢复区域，这种方式就称为热备用。

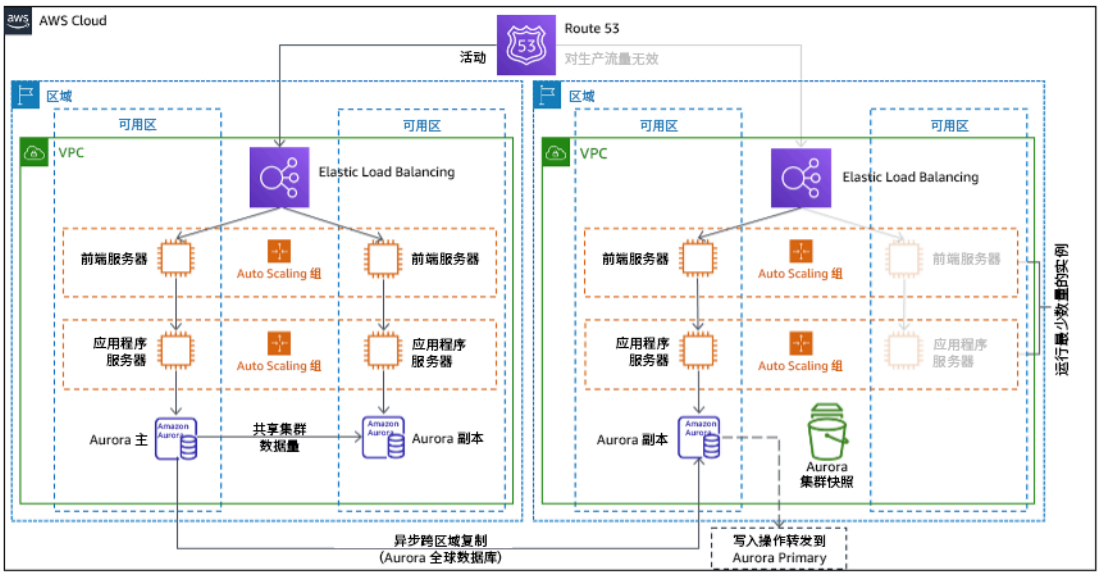


图 21：温备用架构

使用温备用或指示灯需要纵向扩展恢复区域中的资源。为确保在需要时有可用的容量，请考虑使用 EC2 实例的容量预留。如果使用 AWS Lambda，那么预置并发可以提供运行时环境，以便这些环境准备好立即响应函数的调用。

有关此策略的更多详细信息，请参阅《Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby》。

多站点主动/主动

作为多站点主动/主动策略的一部分，您可以在多个区域中同时运行工作负载。多站点主动/主动策略处理来自其部署到的所有区域的流量。客户可能会出于灾难恢复以外的原因选择此策略。此策略可以用于提高可用性，或者在向全球受众部署工作负载（使端点更靠近用户和/或部署针对该区域受众的本地化堆栈）时使用此策略。作为一种灾难恢复策略，如果工作负载在部署此策略的某个 AWS 区域中不能得到支持，那么该区域将被撤出，使用其余区域维持可用性。多站点主动/主动策略是灾难恢复策略中操作最复杂的策略，只有在业务要求需要时才应选择它。

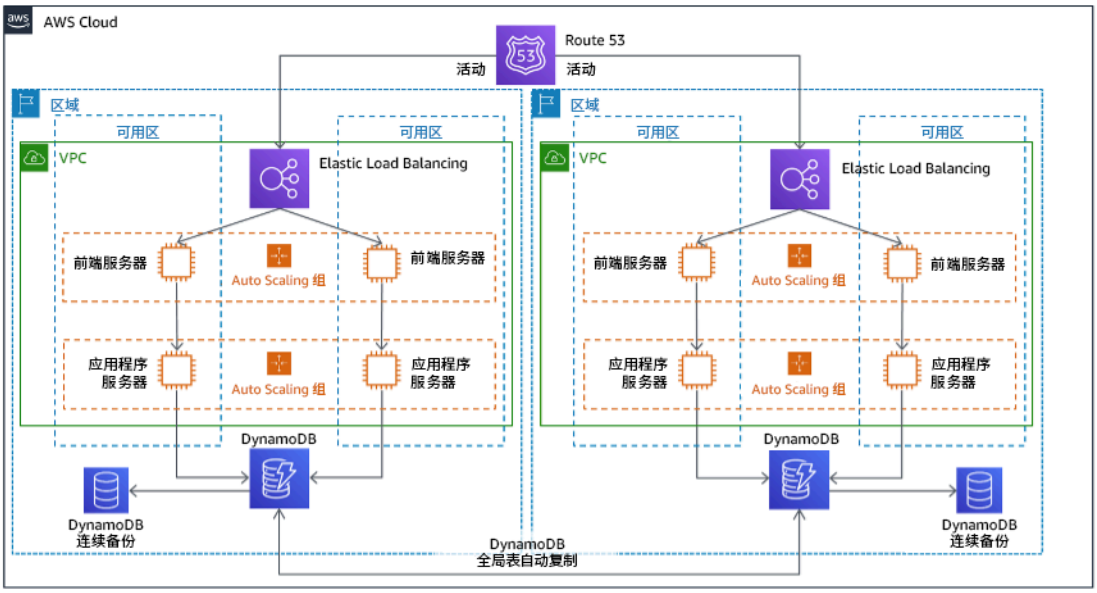


图 22：多站点主动/主动架构

有关此策略的更多详细信息，请参阅《Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active》。

AWS Elastic Disaster Recovery

如果您正考虑为灾难恢复使用指示灯或温备用策略，AWS Elastic Disaster Recovery 可以提供一种带来更多好处的替代方法。弹性灾难恢复可以提供类似于温备用方法的 RPO 和 RTO 目标，同时保持指示灯方法的低成本。弹性灾难恢复将数据从主区域复制到恢复区域，使用持续数据保护来实现以秒为单位的 RPO 和以分钟为单位的 RTO。在恢复区域中仅部署复制数据所需的资源，从而降低成本，类似于指示灯策略。使用弹性灾难恢复时，如果在失效转移或演练过程中启动，则服务会协调并编排计算资源的恢复。

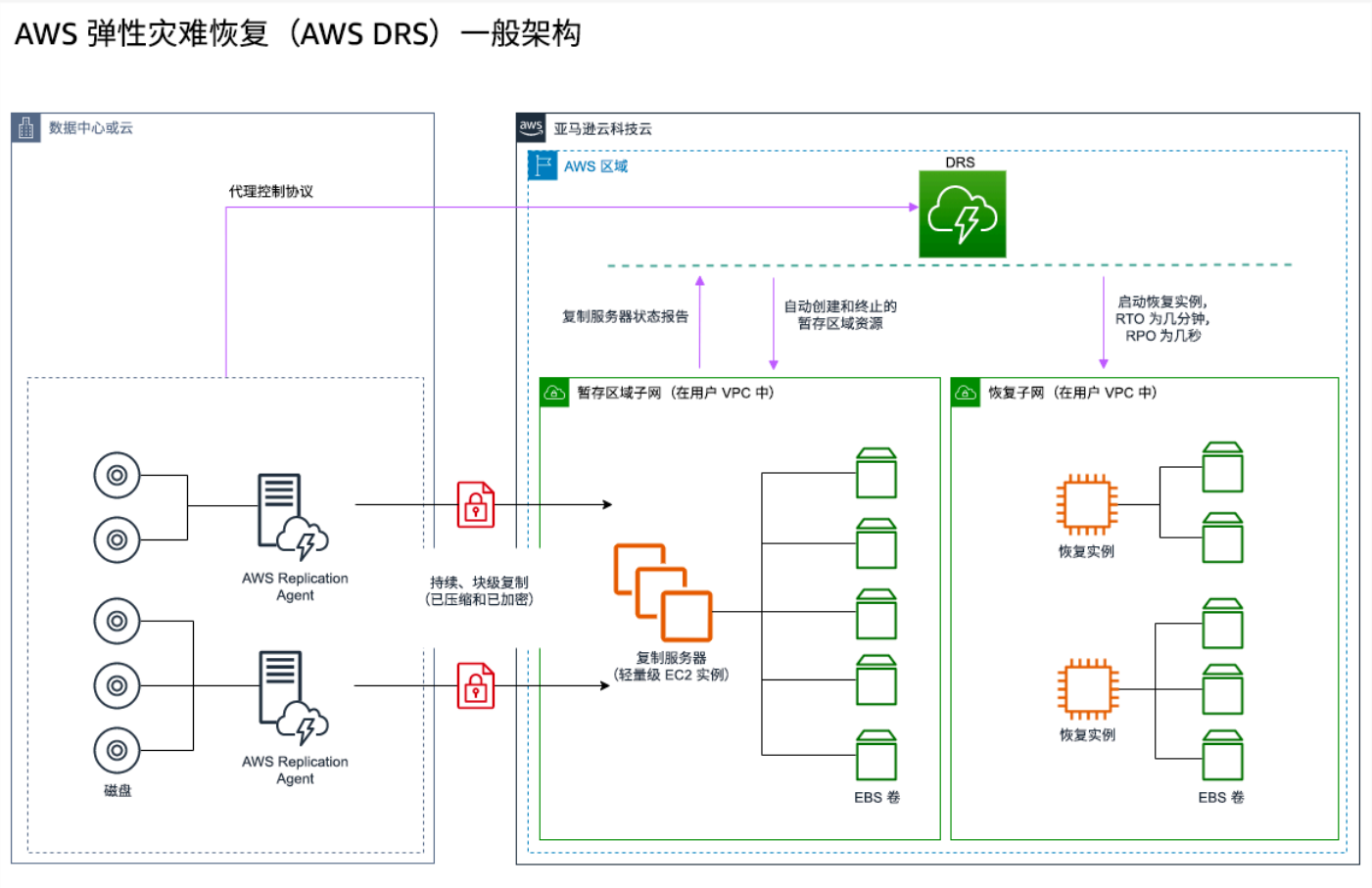


图 23：AWS Elastic Disaster Recovery 架构

其他保护数据的实践

对于所有这些策略，您还必须减轻数据灾难的影响。持续的数据复制可以防止某些类型的灾难，但它可能无法防止数据损坏或破坏，除非您的策略还包括存储数据的版本控制或用于时间点故障恢复的选项。除了副本之外，您还必须备份恢复站点中的复制数据以创建时间点备份。

使用单个 AWS 区域内的多个可用区（AZ）

使用单个区域内的多个可用区时，实施灾难恢复会用到上述策略的多个元素。首先，您必须使用多个可用区创建一个高可用性（HA）架构，如图 23 所示。此架构使用多站点主动/主动方法，因为 Amazon EC2 实例和弹性负载均衡器在多个可用区中部署了资源，主动处理请求。此架构还演示了热备用方法，如果主 Amazon RDS 实例出现故障（或可用区本身出现故障），则备用实例将提升为主实例。

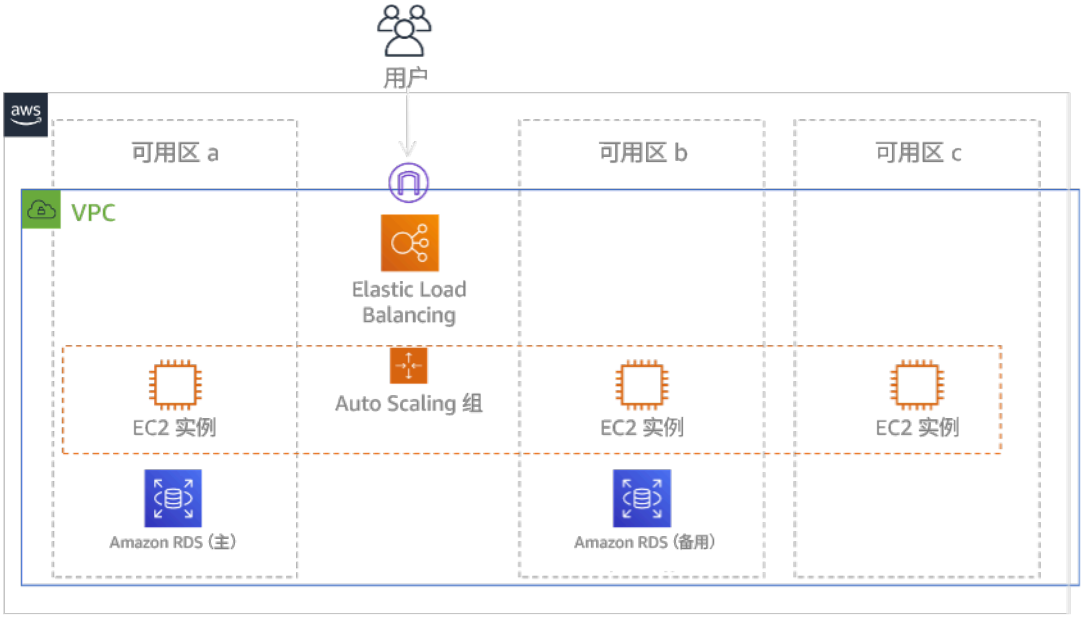


图 24：多可用区架构

除了这种高可用性架构，您还需要添加运行工作负载所需的所有数据的备份。这对于限制在单个区的数据尤其重要，例如 Amazon EBS 卷或 Amazon Redshift 集群。如果一个可用区发生故障，您需要将这些数据恢复到另一个可用区。如果可能，您还应该将数据备份复制到另一个 AWS 区域，提供另一层保护。

下面的博客文章中介绍了一种不太常见的单区域多可用区灾难恢复的替代方法：Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack。这里的策略是尽可能保持可用区之间的隔离，就像区域的运作方式一样。使用这种替代策略，您可以选择主动/主动或主动/被动方法。

> **Note**
>
> 某些工作负载具有数据驻留法规要求。如果这适用于当前只有一个 AWS 区域的位置的工作负载，那么多区域将不适合您的业务需求。多可用区策略可以很好地抵御大多数灾难。

3. 评测工作负载的资源，以及失效转移之前（正常操作期间）恢复区域中的资源配置。

对于基础设施和 AWS 资源，使用基础设施即代码功能（如 AWS CloudFormation）或第三方工具（如 Hashicorp Terraform）。要使用单个操作跨多个账户和区域部署，您可以使用 AWS CloudFormation StackSets。对于多站点主动/主动和热备用策略，恢复区域中部署的基础设施具有与主区域相同的资源。对于指示灯和温备用策略，部署的基础设施需要采取额外操作，才可用于生产。使用 CloudFormation 参数和条件逻辑，您可以通过单个模板控制部署的堆栈处于活动状态还是备用状态。使用弹性灾难恢复时，服务会复制并编排应用程序配置和计算资源的还原。

所有灾难恢复策略都要求在 AWS 区域内备份数据源，然后将这些备份复制到恢复区域。AWS Backup 提供了一个集中视图，可供您在其中配置、调度和监控这些资源的备份。对于指示灯、温备用和多站点主动/主动方法，您还应该将数据从主区域复制到恢复区域中的数据资源，例如 Amazon Relational Database Service（Amazon RDS）数据库实例或 Amazon DynamoDB 表。因此，这些数据资源处于活动状态，可以随时处理恢复区域中的请求。

要了解更多关于 AWS 服务如何跨区域运行的信息，请参阅博客系列文章《Creating a Multi-Region Application with AWS Services》。

4. 确定并实施措施，让恢复区域在需要时（在灾难事件期间）可以进行失效转移。

   对于多站点主动/主动策略，失效转移意味着撤离一个区域，并依赖剩余的活动区域。通常，这些区域已准备好接受流量。对于指示灯和温备用策略，恢复操作需要部署缺失的资源（如图 20 中的 EC2 实例），以及任何其他缺失的资源。

   对于上述所有策略，您可能需要将数据库的只读实例提升为主读/写实例。

   对于备份和还原，从备份中还原数据时会为该数据创建资源，例如 EBS 卷、RDS 数据库实例和 DynamoDB 表。您还需要还原基础设施并部署代码。您可以使用 AWS Backup 来还原恢复区域中的数据。有关更多信息，请参阅 REL09-BP01 识别并备份需要备份的所有数据或从源复制数据。重建基础设施包括创建资源，例如 EC2 实例以及所需的 Amazon Virtual Private Cloud（Amazon VPC）、子网和安全组。您可以自动执行大部分还原过程。要了解具体方法，请参阅这篇博客文章。

5. 确定并实施措施，在需要时（在灾难事件期间）可以重新路由流量进行失效转移。

   此失效转移操作可以自动或手动启动。应谨慎使用基于运行状况检查或警报自动启动的失效转移，因为不必要的失效转移（误报）会产生不可用和数据丢失等成本。因此，通常会使用手动启动的失效转移。在这种情况下，您仍然应该自动执行失效转移步骤，这样手动启动就像按一下按钮一样简单。

   在使用 AWS 服务时，需要考虑几个流量管理选项。一个选项是使用 Amazon Route 53。使用 Amazon Route 53，您可以将一个或多个 AWS 区域中的多个 IP 端点与一个 Route 53 域名相关

联。要实施手动启动的失效转移，您可以使用 Amazon 应用程序恢复控制器，利用其提供的高可用性数据面板 API 将流量重新路由到恢复区域。实施失效转移时，使用数据面板操作并避免控制面板操作，如此部分所述：REL11-BP04 恢复期间依赖于数据面板而不是控制面板。

要了解有关此选项及其他选项的更多信息，请参阅灾难恢复白皮书中的此部分。

6. 设计工作负载的失效自动恢复计划。

失效自动恢复是指在灾难事件消除后将工作负载运营恢复到主区域。向主区域预置基础设施和代码通常遵循最初使用的相同步骤，依赖于基础设施即代码和代码部署管道。失效自动恢复面临的挑战是还原数据存储，并确保它们与运行中的恢复区域保持一致。

在失效转移状态下，恢复区域中的数据库处于活动状态，并且具有最新数据。然后，目标是从恢复区域重新同步到主区域，确保主区域处于最新状态。

某些 AWS 服务会自动执行此操作。如果使用 Amazon DynamoDB 全局表，即使主区域中的表不可用，只要重新联机，DynamoDB 就会继续传播任何挂起的写操作。如果使用 Amazon Aurora Global Database 并使用托管的计划失效转移，则维护 Aurora 全局数据库的现有复制拓扑。因此，主区域中以前的读/写实例将成为副本，并从恢复区域接收更新。

如果这不是自动执行的，则需要在主区域中重新建立数据库，作为恢复区域中数据库的副本。在许多情况下，这将涉及删除旧的主数据库，然后创建新的副本。

失效转移后，如果可以继续在恢复区域中运行，请考虑将此区域设为新的主区域。您仍然需要执行上述所有步骤，将以前的主区域变成恢复区域。有些组织会进行定期轮换，定期交换其主区域和恢复区域（例如每三个月一次）。

失效转移和失效自动恢复所需的所有步骤都应保存在行动手册中且可供所有团队成员使用，并定期接受审查。

使用弹性灾难恢复时，服务会协助编排并自动执行失效自动恢复流程。有关更多详细信息，请参阅 Performing a failback。

实施计划的工作量级别：高

资源

相关最佳实践：

- the section called “REL09-BP01 识别并备份需要备份的所有数据或从源复制数据”
- the section called “REL11-BP04 恢复期间依赖于数据面板而不是控制面板”

- the section called “REL13-BP01 定义停机和数据丢失的恢复目标”

相关文档：

- AWS Architecture Blog: Disaster Recovery 系列博客文章
- AWS 上工作负载的灾难恢复：云中的恢复（AWS 白皮书）
- 云中的灾难恢复选项
- Build a serverless multi-region, active-active backend solution in an hour
- Multi-region serverless backend — reloaded
- RDS：跨区域复制只读副本
- Route 53: Configuring DNS Failover
- S3：跨区域复制
- 什么是 AWS Backup？
- What is Amazon Application Recovery Controller?
- AWS 弹性灾难恢复
- HashiCorp Terraform: Get Started - AWS
- APN 合作伙伴：可帮助进行灾难恢复的合作伙伴
- AWS Marketplace：可用于灾难恢复的产品

相关视频：

- AWS 上工作负载的灾难恢复
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

REL13-BP03 测试灾难恢复实施以验证实施效果

定期测试到恢复站点的失效转移，验证是否在正常运作，以及是否满足 RTO 和 RPO。

常见反模式：

- 从不在生产环境中进行失效转移演练。

建立此最佳实践的好处：定期测试灾难恢复计划，验证计划在需要时能否正常发挥作用，以及团队是否知道如何执行策略。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要避免的模式是制定了恢复路径但很少测试。例如，您可能有一个用于只读查询的辅助数据存储。在写入某个数据存储，却发现主存储故障时，您可能希望失效转移到辅助数据存储。如果不经常测试此失效转移，您可能会发现自己关于辅助数据存储容量的假设是错误的。辅助数据存储容量在上次测试时可能是足够的，但可能无法再容纳这次情况下的负载。根据我们的经验，唯一有效的错误恢复路径是您经常测试的路径。因此，最好只制定几条恢复路径。您可以建立恢复模式并定期对其进行测试。如果恢复路径比较复杂或至关重要，您仍需定期在生产环境中测试该故障，确保恢复路径有效。在我们刚才讨论的示例中，您应该定期将故障转移到备用存储，无论是否有需要。

实施步骤

1. 为灾难恢复设计工作负载。定期测试恢复路径。面向恢复的计算可识别系统中能够增强恢复功能的特性：隔离和冗余，系统范围回滚更改的能力，监控并确定运行状况的能力，提供诊断、自动恢复、模块化设计的能力，以及重启的能力。对恢复路径进行演练，确认可以在指定时间内恢复到指定状态。在此恢复过程中使用运行手册来记录问题，并在下一次测试之前找到问题的解决方案。
2. 对于基于 Amazon EC2 的工作负载，使用 AWS Elastic Disaster Recovery 为灾难恢复策略实施和启动演练实例。AWS Elastic Disaster Recovery 可以高效地运行演练，帮助您为失效转移事件做好准备。您还可以使用弹性灾难恢复频繁地启动实例进行测试和演练，无需重定向流量。

资源

相关文档：

- APN 合作伙伴：可帮助进行灾难恢复的合作伙伴
- AWS Architecture Blog: Disaster Recovery 系列博客文章
- AWS Marketplace：可用于灾难恢复的产品
- AWS Elastic Disaster Recovery
- AWS 上工作负载的灾难恢复：云中的恢复（AWS 白皮书）
- AWS Elastic Disaster Recovery 为失效转移做准备
- The Berkeley/Stanford recovery-oriented computing project
- What is AWS Fault Injection Simulator?

相关视频：

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS

REL13-BP04 管理灾难恢复站点或区域的配置偏差

为了成功执行灾难恢复（DR）过程，一旦灾难恢复环境上线，工作负载就必须能够及时恢复正常操作，而不会丢失相关的功能或数据。要实现这一目标，务必在灾难恢复环境和主环境之间保持一致的基础设施、数据和配置。

期望结果：灾难恢复站点的配置和数据与主站点相当，这有助于在需要时进行快速而完整的恢复。

常见反模式：

- 当对主位置进行更改时，您未能更新恢复位置，这导致配置过时，从而阻碍恢复工作。
- 您未考虑潜在的限制，例如主位置和恢复位置之间的服务差异，这些限制可能会在失效转移期间导致意外故障。
- 您依赖手动流程来更新和同步灾难恢复环境，这会增加人为错误和不一致的风险。
- 您未能检测到配置偏差，这会导致在事件发生之前错误地感知灾难恢复站点就绪状态。

建立此最佳实践的好处：灾难恢复环境和主环境之间的一致性可显著提高事件发生后成功恢复的可能性，并降低恢复过程失败的风险。

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

全面的配置管理和失效转移就绪方法有助于您验证灾难恢复站点是否持续更新，并准备好在主站点出现故障时进行接管。

要实现主环境和灾难恢复（DR）环境之间的一致性，请验证您的交付管道是否同时将应用程序分发到主站点和灾难恢复站点。在适当的评估期后推出对灾难恢复站点的更改（也称为错开部署），以检测主站点的问题，并在问题蔓延之前停止部署。实施监控以检测配置偏差，并跟踪环境中的更改和合规性。在灾难恢复站点中执行自动修复，以使其保持完全一致，并做好准备在发生事件时立即接管。

实施步骤

1. 验证灾难恢复区域包含成功执行灾难恢复计划所需的 AWS 服务和功能。

2. 使用基础设施即代码（IaC）。保持生产基础设施和应用程序配置模板的准确性，并定期将其应用于灾难恢复环境。AWS CloudFormation 可以检测 CloudFormation 模板指定的内容与实际部署内容之间的偏差。
3. 配置 CI/CD 管道来将应用程序和基础设施更新部署到所有环境，包括主站点和灾难恢复站点。诸如 AWS CodePipeline 等 CI/CD 解决方案可以自动执行部署过程，从而降低配置偏差的风险。
4. 在主环境和灾难恢复环境之间错开部署。这种方法支持在主环境中对更新进行初始部署和测试，这样可以在问题传播到灾难恢复站点之前，将其隔离在主站点中。这种方法可以防止同时将缺陷推送到生产和灾难恢复站点，并保持灾难恢复环境的完整性。
5. 持续监控主环境和灾难恢复环境中的资源配置。诸如 AWS Config 之类的解决方案有助于强制实施配置合规性并检测偏差，这有助于在不同环境中保持一致的配置。
6. 实施警报机制，以跟踪和通知任何配置偏差或数据复制中断或滞后。
7. 自动修复检测到的配置偏差。
8. 安排定期审计和合规性检查，以验证主配置和灾难恢复配置之间的持续一致性。定期审核可帮助您保持对既定规则的遵守，并确定需要解决的任何差异。
9. 检查 AWS 预置容量、服务配额、节流限制以及配置和版本差异是否存在不匹配。

资源

相关最佳实践：

- REL01-BP01 了解服务配额和约束
- REL01-BP02 跨多个账户和区域管理服务配额
- REL01-BP04 监控和管理配额
- REL13-BP03 测试灾难恢复实施以验证实施效果

相关文档：

- 按照 AWS Config 规则修正不合规 AWS 资源
- AWS Systems Manager Automation
- AWS CloudFormation：检测堆栈和资源的非托管配置更改
- AWS CloudFormation：在整个 CloudFormation 堆栈上检测偏差
- AWS Systems Manager Automation
- AWS 上工作负载的灾难恢复：云中的恢复（AWS 白皮书）
- 如何在 AWS 上实施基础设施配置管理解决方案？

- 按照 AWS Config 规则修正不合规 AWS 资源

相关视频：

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

相关示例：

- AWS CloudFormation 注册表
- 适用于 AWS 的配额监控程序
- Implement automatic drift remediation for AWS CloudFormation using Amazon CloudWatch and AWS Lambda
- AWS Architecture Blog: Disaster Recovery 系列博客文章
- AWS Marketplace：可用于灾难恢复的产品
- 自动实现无需干预的安全部署

REL13-BP05 自动执行恢复

实施可靠、可观测、可重复且经过测试的自动化恢复机制，来降低故障的风险和业务影响。

期望结果：您已经为恢复流程实施了有据可查、标准化且经过全面测试的自动化工作流程。恢复自动化功能会自动纠正导致数据丢失或不可用（低风险）的小问题。您可以针对严重事件快速调用恢复流程，在恢复流程运行时观察补救行为，并在观测到危险情况或故障时结束这些流程。

常见反模式：

- 您依赖于处于故障或已降级状态的组件或机制，作为恢复计划的一部分。
- 恢复流程需要手动干预，例如控制台访问（也称为单击操作）。
- 您在存在数据丢失或不可用高风险的情况下自动启动恢复过程。
- 您没有包含一个机制来中止不起作用或带来额外风险的恢复过程（如暗灯 或红色的大型停止按钮）。

建立此最佳实践的好处：

- 提高了恢复操作的可靠性、可预测性和一致性。

- 能够实现要求更高的恢复目标，包括恢复时间目标（RTO）和恢复点目标（RPO）。
- 降低了事件期间恢复失败的可能性。
- 降低了与容易出现人为错误的手动恢复过程相关的故障风险。

在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

要实施自动恢复，您需要一种使用 AWS 服务和最佳实践的全面方法。首先，请确定工作负载中的关键组件和潜在故障点。开发自动化流程，无需人工干预即可从故障中恢复工作负载和数据。

使用基础设施即代码（IaC）原则开发恢复自动化功能。这使您的恢复环境与源环境保持一致，并支持对恢复过程进行版本控制。要编排复杂的恢复工作流程，可以考虑诸如 AWS Systems Manager 自动化或 AWS Step Functions 等解决方案。

恢复过程自动化可带来显著的好处，有助于您更轻松地实现恢复时间目标（RTO）和恢复点目标（RPO）。但是，此类功能可能会遇到意外情况，而可能会导致失败或造成新的风险，例如额外的停机时间和数据丢失。要降低这种风险，请提供快速停止正在进行的恢复自动化的功能。一旦停止，您就可以进行调查并采取纠正措施。

对于支持的工作负载，可以考虑使用 AWS 弹性灾难恢复（AWS DRS）等解决方案来提供自动失效转移。AWSDRS 会持续将计算机（包括操作系统、系统状态配置、数据库、应用程序和文件）复制到目标 AWS 账户和首选区域中的暂存区。如果发生事件，AWS DRS 会自动将复制的服务器转换为 AWS 上恢复区域中完全预置的工作负载。

维护和改进自动恢复是一个持续的过程。根据经验教训不断测试和完善恢复过程，并随时了解可以增强恢复能力的新 AWS 服务和功能。

实施步骤

1. 规划自动恢复
    a. 对您的工作负载架构、组件和依赖关系进行全面审核，来确定和规划自动恢复机制。将工作负载的依赖关系分类为硬 依赖关系和软 依赖关系。硬依赖关系是指工作负载运行所必须具备且无法替代的依赖关系。软依赖关系是工作负载通常使用的依赖关系，但此类依赖关系可以用临时替代系统或流程替换，或者可以通过优雅降级进行处理。
    b. 建立识别和恢复丢失或已损坏数据的流程。
    c. 定义在完成恢复操作后确认已恢复的稳定状态的步骤。
    d. 考虑为使恢复的系统准备好提供全面服务所需的任何操作，例如预热和填充缓存。

e. 考虑在恢复过程中可能遇到的问题以及如何检测和修复这些问题。

f. 考虑无法访问主站点及其控制面板的场景。验证可以在不依赖主站点的情况下独立执行恢复操作。考虑使用诸如 Amazon 应用程序恢复控制器（ARC）之类的解决方案来重定向流量，而无需手动更改 DNS 记录。

2. 开发自动恢复流程

a. 实施自动化的故障检测和失效转移机制，来实现免手动恢复。构建控制面板（例如使用 Amazon CloudWatch）来报告自动恢复过程的进度和运行状况。包括验证成功恢复的过程。提供一种机制来中止正在进行的恢复。

b. 对于无法自动恢复的故障，编写行动手册作为后备流程，并考虑制定灾难恢复计划。

c. 测试恢复过程，如 REL13-BP03 中所述。

3. 为恢复做好准备

a. 评估恢复站点的状态并提前为其部署关键组件。有关更多详细信息，请参阅 REL13-BP04。

b. 为恢复操作定义明确的角色、职责和决策过程，涉及整个组织的有关利益相关者和团队。

c. 定义启动恢复过程的条件。

d. 制定计划来还原恢复过程，需要时或在认为安全之后回退到主站点。

资源

相关最佳实践：

- REL07-BP01 在获取或扩展资源时利用自动化
- REL11-BP01 监控工作负载的所有组件以检测故障
- REL13-BP02 使用定义的恢复策略来实现恢复目标
- REL13-BP03 测试灾难恢复实施以验证实施效果
- REL13-BP04 管理灾难恢复站点或区域的配置偏差

相关文档：

- AWS Architecture Blog: Disaster Recovery 系列博客文章
- AWS 上工作负载的灾难恢复：云中的恢复（AWS 白皮书）
- Orchestrate Disaster Recovery Automation using Amazon Route 53 ARC and AWS Step Functions
- Build AWS Systems Manager Automation runbooks using AWS CDK
- AWS Marketplace: Products That Can Be Used for Disaster Recovery

- AWS Systems Manager Automation
- AWS 弹性灾难恢复
- 使用弹性灾难恢复进行失效转移和失效自动恢复
- AWS 弹性灾难恢复资源
- APN 合作伙伴：有助于进行灾难恢复的合作伙伴

相关视频：

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2022: AWS On Air ft. AWS Failback for AWS Elastic Disaster Recovery

性能效率

性能效率支柱涉及高效地使用云资源以满足性能要求的能力，以及在需求变化和技术发展时保持该效率的能力。有关具体实施的说明性指导，请参阅《性能效率支柱白皮书》。

最佳实践领域

- 架构选择
- 计算和硬件
- 数据管理
- 网络和内容分发
- 流程和文化

架构选择

问题

- PERF 1. 如何为工作负载选择合适的云资源和架构？

PERF 1. 如何为工作负载选择合适的云资源和架构？

针对特定工作负载的最佳解决方案各不相同，而且解决方案通常会结合多种方法。Well-Architected 工作负载会使用多种解决方案，并且允许使用各种不同的功能来提高性能。

最佳实践

- PERF01-BP01 了解并掌握可用的云服务和功能
- PERF01-BP02 使用云提供商或合适的合作伙伴提供的指导来了解架构模式和最佳实践
- PERF01-BP03 制定架构决策时考虑成本因素
- PERF01-BP04 评估权衡机制对客户和架构效率的影响
- PERF01-BP05 使用策略和参考架构
- PERF01-BP06 使用基准测试来推动制定架构决策
- PERF01-BP07 使用数据驱动的方法进行架构选择

PERF01-BP01 了解并掌握可用的云服务和功能

不断了解和发现可用的服务和配置，这些服务和配置有助于作出更好的架构决策，并提高工作负载架构的性能效率。

常见反模式：

- 将云用作联合数据中心。
- 迁移到云后，没有对应用程序进行现代化改造。
- 仅使用一种存储类型来存储所有需要继续保留的内容。
- 使用的实例类型最接近当前标准，但有时候需要使用更大的实例。
- 部署和管理作为托管服务提供的技术。

建立此最佳实践的好处：通过考虑采用新的服务和配置，可以大大提高性能、降低成本并减少维护工作负载所需的工作量。还有助于缩短支持云的产品的价值实现时间。

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

AWS 不断发布新的服务和功能，可提高性能并降低云工作负载的成本。及时了解这些新服务和功能对于保持云的性能效率至关重要。对工作负载架构进行现代化改造还有助于提高工作效率、推动创新并解锁更多增长机会。

实施步骤

- 盘点相关服务的工作负载软件和架构。决定要进一步了解哪一类产品。
- 探索 AWS 产品，确定并了解有助于提高性能、降低成本和运营复杂性的相关服务和配置选项。

  - Amazon Web Services Cloud
  - AWS Academy
  - AWS 的新功能
  - AWS 博客
  - AWS Skill Builder
  - AWS 活动和网络研讨会
  - AWS 培训 和认证
  - AWS YouTube 频道
  - AWS 讲习会
  - AWS 社区
- 使用 Amazon Q 获取有关服务的相关信息和建议。
- 使用沙盒（非生产）环境来了解和试验新服务，且不会产生额外费用。
- 不断了解新的云服务和功能。

资源

相关文档：

- Overview of Amazon Web Services
- Amazon EC2 功能
- 通过 AWS 合作伙伴学习计划逐步学习
- AWS 培训和认证
- My learning path to become an AWS solutions architect
- AWS 架构中心
- AWS Partner Network
- AWS 解决方案库
- AWS Knowledge Center
- 在 AWS 上构建现代应用程序

相关视频：

- AWS re:Invent 2023 – What's new with Amazon EC2

- AWS re:Invent 2022 - Reduce your operational and infrastructure costs with Amazon ECS
- AWS re:Invent 2023 - Build with the efficiency, agility & innovation of the cloud with AWS
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- 这是我的架构

相关示例：

- AWS 示例
- AWS SDK 示例

PERF01-BP02 使用云提供商或合适的合作伙伴提供的指导来了解架构模式和最佳实践

利用云服务公司提供的资源（如文档、解决方案架构师、专业服务或合适的合作伙伴）来指导您制定架构决策。这些资源有助于您审查并改进架构，从而实现最佳性能。

常见反模式：

- 您将 AWS 视为普通的云提供商。
- 您没有按 AWS 服务的既定用途使用这些服务。
- 您在遵循所有指导时没有考虑到业务环境。

建立此最佳实践的好处：使用云提供商或合适的合作伙伴提供的指导，有助于您为工作负载选择合适的架构，让您对自己的决策充满信心。

在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 提供广泛的指导、文档和资源，有利于您构建和管理高效的云工作负载。AWS 文档提供了代码示例、教程和详细的服务说明。除文档外，AWS 还提供培训和认证计划、解决方案架构师和专业服务，可协助客户探索云服务的不同方面，并在 AWS 上实施高效的云架构。

利用这些资源深入了解宝贵的知识和最佳实践，节省时间，并在 AWS Cloud 中取得更好的成果。

实施步骤

- 查看 AWS 文档和指导并遵循最佳实践。这些资源有助于您高效地选择和配置服务来实现更好的性能。

  - AWS 文档（例如用户指南和白皮书）
  - AWS 博客
  - AWS 培训 和认证
  - AWS YouTube 频道
- 参加 AWS 合作伙伴活动（如 AWS 全球峰会、AWS re:Invent、用户群组和讲习会），向 AWS 专家学习关于使用 AWS 服务的最佳实践。
  - 通过 AWS 合作伙伴学习计划逐步学习
  - AWS 活动和网络研讨会
  - AWS 讲习会
  - AWS 社区
- 如需其他指导或产品信息，请联系 AWS 获取帮助。AWS解决方案架构师和AWS 专业服务提供关于实施解决方案的指导。AWS合作伙伴提供 AWS 专业知识，可帮助您实现业务敏捷性和创新能力。
- 如需技术支持来高效利用服务，请使用 支持。我们的支持计划旨在为您提供理想的工具组合以及获取专业知识的渠道，让您可以在优化性能、管理风险和控制成本的同时，使用 AWS 取得成功。

资源

相关文档：

- AWS 架构中心
- AWS Partner Network
- AWS 解决方案库
- AWS Knowledge Center
- AWS Enterprise Support

相关视频：

- 这是我的架构
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 – 在 AWS 上实施分布式设计模式
- AWS re:Invent 2023 – 应用程序架构即代码

相关示例：

- AWS 示例
- AWS SDK 示例
- AWS 分析参考架构

PERF01-BP03 制定架构决策时考虑成本因素

制定架构决策时考虑成本因素，以便提高云工作负载的资源利用率和性能效率。意识到云工作负载的成本影响时，就更有可能充分利用有效资源，减少浪费。

常见反模式：

- 只使用一个系列的实例。
- 没有对照开源解决方案对许可的解决方案进行评估。
- 没有定义存储生命周期策略。
- 没有查看 AWS Cloud 的新服务和功能。
- 只使用数据块存储。

建立此最佳实践的好处：通过在制定决策时考虑成本因素，可以让您使用更有效的资源，并探索其他投资方式。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

针对成本优化工作负载能够提高资源利用率，避免在云工作负载中出现浪费。要在制定架构决策时考虑成本因素，通常包括合理调整工作负载组件的大小和实现弹性，从而提高云工作负载的性能效率。

实施步骤

- 制定成本目标，如云工作负载的预算限额。
- 确定会增加工作负载成本的关键组件（如实例和存储）。可以使用 AWS 定价计算器 和 AWS Cost Explorer 来确定工作负载中的关键成本驱动因素。
- 了解云中的定价模式，例如按需型实例、预留实例、节省计划和竞价型实例。
- 使用 Well-Architected 成本优化最佳实践来优化这些关键组件的成本。
- 持续监控和分析成本，发现工作负载中的成本优化机会。

- 使用 AWS Budgets，针对无法接受的成本获取相关提醒。
- 使用 AWS Compute Optimizer 或 AWS Trusted Advisor 获取成本优化建议。
- 使用 AWS 成本异常检测自动进行成本异常检测和根本原因分析。

资源

相关文档：

- What is AWS Billing and Cost Management?
- 借助 AWS 实现成本优化
- Choosing an AWS cost management strategy
- A Beginner's Guide to AWS Cost Management
- A Detailed Overview of the Cost Intelligence Dashboard
- AWS 架构中心
- AWS 解决方案库
- AWS Knowledge Center

相关视频：

- 这是我的架构
- AWS re:Invent 2023 - What's new with AWS cost optimization
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

相关示例：

- AWS Compute Optimizer 演示代码
- Cost Optimization 讲习会
- Cloud Financial Management Technical Implementation Playbooks
- Startup optimization: Tuning application performance for maximum efficiency
- Serverless Optimization 讲习会（Performance and Cost）

- Scaling cost effective architectures

PERF01-BP04 评估权衡机制对客户和架构效率的影响

在评估与性能相关的改进时，确定哪些选择会对客户和工作负载效率产生影响。例如，如果使用键值数据存储可以提高系统性能，则评估这种更改的最终一致性对客户的影响就非常重要。

常见反模式：

- 您认为即便需要实施一些权衡机制，也要实现所有性能收益。
- 在性能问题已经非常严重时，只评估对工作负载的更改。

建立此最佳实践的好处：评估与性能相关的潜在改进时，必须决定更改时所采用的权衡机制是否符合工作负载要求。在某些情况下，可能必须实施额外的控制来补偿权衡机制。

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

根据性能和客户影响确定架构中的关键领域。确定可以如何进行改进、这些改进带来的利弊，并确定改进对系统和用户体验的影响。例如，缓存数据有助于大幅提高性能，但需要就如何以及何时更新缓存的数据或使其变得无效而制定明确的策略，以便防止产生不正确的系统行为。

实施步骤

- 了解工作负载要求和 SLA。
- 明确定义评估因素。这些因素可能与工作负载的成本、可靠性、安全性和性能有关。
- 选择可以满足要求的架构和服务。
- 开展试验工作并执行概念验证（POC），评估权衡因素以及对客户和架构效率的影响。高度可用、高性能和安全的工作负载往往会消耗更多的云资源，但同时也会提供更好的客户体验。了解工作负载的复杂性、性能和成本之间的权衡因素。通常，重视其中两个因素会以牺牲第三个因素为代价。

资源

相关文档：

- Amazon Builders' Library
- QuickSight KPIs

- Amazon CloudWatch RUM
- X-Ray 文档
- Understand resiliency patterns and trade-offs to architect efficiently in the cloud

相关视频：

- Optimize applications through Amazon CloudWatch RUM
- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

相关示例：

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP05 使用策略和参考架构

在选择服务和配置时使用内部策略和现有参考架构，可提高设计和实施工作负载时的效率。

常见反模式：

- 允许使用各种各样的技术，而这些技术可能会影响公司的管理开销。

建立此最佳实践的好处：制定架构、技术和供应商选择策略，有助于快速作出决策。

在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

在选择资源和架构时需要制定内部策略，这样在进行架构方面的选择时，可以提供应遵循的标准和指导方针。在选择合适的云服务时，这些指导方针可以简化决策过程，并提高性能效率。使用策略或参考架构部署工作负载。将服务集成到云部署中，然后使用性能测试来验证是否能继续满足性能要求。

实施步骤

- 清楚了解云工作负载的要求。
- 审查内部和外部策略，找出最有效的策略。

- 使用 AWS 提供的合适参考架构或行业最佳实践。
- 创建一个连续体，其中包含策略、标准、参考架构和针对常见情况的规范性指南。这样做可以让团队更快地开展工作。请酌情为垂直行业量身定制资产。
- 在沙盒环境中，为工作负载验证这些策略和参考架构。
- 随时了解行业标准和 AWS 更新，确保策略和参考架构有助于优化云工作负载。

资源

相关文档：

- AWS 架构中心
- AWS Partner Network
- AWS 解决方案库
- AWS Knowledge Center
- AWS Architecture Blog

相关视频：

- 这是我的架构
- AWS re:Invent 2022 - Accelerate value for your business with SAP & AWS reference architecture

相关示例：

- AWS 示例
- AWS SDK 示例

PERF01-BP06 使用基准测试来推动制定架构决策

对现有工作负载的性能进行基准测试，了解工作负载在云中的表现情况，并根据这些数据推动制定架构决策。

常见反模式：

- 启用普通的基准测试，而这些基准测试并不能反映出工作负载的特征。
- 将客户反馈和看法作为唯一的基准。

建立此最佳实践的好处：对当前实现进行基准测试可以衡量性能改进。

在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

结合使用基准测试与综合测试，评测工作负载组件的性能。相比负载测试，基准测试通常可以更快速地设置，并且适用于评估特定组件的技术。基准测试通常在新项目开始时进行，因为此时您还没有用于进行负载测试的完整解决方案。

您可以构建自己的自定义基准测试，也可以使用行业标准测试（如 TPC-DS），对工作负载进行基准测试。行业基准测试适用于比较不同的环境。对于在架构中想要执行的特定类型操作，自定义基准测试十分有用。

进行基准测试时，为了确保获得有效的结果，预热测试环境尤为重要。多次运行同一基准测试，确保捕获一段时间内的所有差异。

由于基准测试运行速度通常比负载测试快，它们可以在部署管道的早期使用，并能更快地提供有关性能偏差的反馈。评估组件或服务的重要更改时，可以使用基准测试快速了解是否有合理的理由来执行更改。结合使用基准测试与负载测试这一点很重要，因为负载测试能告知工作负载在生产环境中的表现如何。

实施步骤

- 规划和定义：
  - 为基准测试定义目标、基准、测试场景、指标（如 CPU 利用率、延迟或吞吐量）和 KPI。
  - 关注用户在用户体验方面的要求，以及响应时间和可访问性等因素。
  - 确定适用于工作负载的基准测试工具。可以使用与工作负载兼容的 Amazon CloudWatch 等 AWS 服务或第三方工具。
- 配置和检测：
  - 设置环境并配置资源。
  - 实施监控和日志记录来捕获测试结果。
- 基准测试和监控：
  - 执行基准测试并在测试期间监控指标。
- 分析和记录：
  - 记录基准测试过程和测试结果。
  - 对结果进行分析，确定瓶颈、趋势和需要改进的方面。

    - 利用测试结果制定架构决策并调整工作负载。这可能包括更改服务或采用新功能。
- 优化并重复：
    - 根据基准测试调整资源配置和分配。
    - 调整后重新测试工作负载，验证改进情况。
    - 记录经验教训，并重复该过程，确定其他需要改进的方面。

资源

相关文档：

- AWS 架构中心
- AWS Partner Network
- AWS 解决方案库
- AWS Knowledge Center
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Genomics workflows, Part 5: automated benchmarking
- Benchmark and optimize endpoint deployment in Amazon SageMaker AI JumpStart

相关视频：

- AWS re:Invent 2023 - Benchmarking AWS Lambda cold starts
- Benchmarking stateful services in the cloud
- 这是我的架构
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

相关示例：

- AWS 示例
- AWS SDK 示例
- 分布式负载测试
- Measure page load time with Amazon CloudWatch Synthetics

- Amazon CloudWatch RUM Web Client

PERF01-BP07 使用数据驱动的方法进行架构选择

为架构选择确定清晰的数据驱动方法，确保使用合适的云服务和配置来满足特定业务需求。

常见反模式：

- 您认为当前的架构是静态的，不应随着时间的推移而更新。
- 选择架构时基于猜测和假设。
- 不断对架构进行更改，而不提供正当理由。

建立此最佳实践的好处：通过使用明确定义的方法来选择架构，可以利用数据来优化工作负载设计，在未来作出明智的决策。

在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

利用内部经验和云知识或外部资源（如已发布的应用场景或白皮书）来选择架构中的资源和服务。应制定一个明确定义的流程，该流程支持对可能会用于工作负载的不同服务进行试验和基准测试。

关键工作负载的积压工作不仅应包括用户案例（提供与业务和用户相关的功能），还应包括技术案例（创建工作负载的架构跑道）。该跑道依托于技术和新服务领域新的改进，并根据数据和适当的理由采用这些改进。这可以确保架构经得起未来考验，不会停滞不前。

实施步骤

- 与关键利益相关方一起确定工作负载要求，包括性能、可用性和成本方面的考量。考虑诸如用户数量和工作负载使用模式之类的因素。
- 创建架构跑道或技术积压工作，统筹确定它们与功能型待办事项的优先级。
- 评估和评测不同的云服务（有关详细信息，请参阅 PERF01-BP01 了解并掌握可用的云服务和功能）。
- 探索满足性能要求的不同架构模式，如微服务或无服务器（有关详细信息，请参阅 PERF01-BP02 使用云提供商或合适的合作伙伴提供的指导来了解架构模式和最佳实践）。

- 咨询其他团队、架构图和资源，例如 AWS 解决方案架构师、AWS 架构中心和 AWS Partner Network，协助为工作负载选择合适的架构。

- 定义吞吐量和响应时间等性能指标，以便于评估工作负载的性能。
- 进行试验并使用定义的指标来验证所选架构的性能。
- 持续监控并根据需要进行调整，从而使架构保持最佳性能。
- 记录所选架构和决策，作为将来更新和学习的参考。
- 根据经验教训、新技术以及可表明当前方法需要进行更改或存在问题的指标，不断审查和更新架构选择方法。

资源

相关文档：

- AWS 解决方案库
- AWS Knowledge Center
- Architectural Patterns to Build End-to-End Data Driven Applications on AWS

相关视频：

- 这是我的架构
- AWS re:Invent 2021 - Data-driven enterprise: Going from vision to value
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results

相关示例：

- AWS 示例
- AWS SDK 示例

计算和硬件

问题

- PERF 2. 如何在工作负载中选择和使用计算资源？

PERF 2. 如何在工作负载中选择和使用计算资源？

适合特定工作负载的最佳计算方案会因应用程序设计、使用模式和配置设置而有所不同。架构可能会使用不同的计算方案来支持各种组件，并允许使用不同的功能来提高性能。为架构选择错误的计算方案可能会降低性能效率。

最佳实践

- PERF02-BP01 为工作负载选择最佳计算方案
- PERF02-BP02 了解可用的计算配置和功能
- PERF02-BP03 收集与计算相关的指标
- PERF02-BP04 配置计算资源并合理调整资源规模
- PERF02-BP05 动态扩展计算资源
- PERF02-BP06 使用基于硬件的优化型计算加速器

PERF02-BP01 为工作负载选择最佳计算方案

通过为工作负载选择最合适的计算方案，可以提高性能，减少不必要的基础设施成本以及维护工作负载所需的运营工作。

常见反模式：

- 使用本地所用的计算方案。
- 对云计算方案、功能和解决方案以及这些解决方案可以如何提高计算性能缺乏认识。
- 为了满足扩展或性能要求，过度预置现有计算方案，而使用替代计算方案可以更准确地满足工作负载特征需求。

建立此最佳实践的好处：通过确定计算要求并对可用方案进行评估，可以让工作负载更高效地利用资源。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

为了提高性能效率而优化云工作负载时，请务必根据应用场景和性能要求选择最合适的计算方案。AWS 提供了多种计算方案，可满足云中不同工作负载的需求。例如，您可以使用 Amazon EC2 启动和管理虚拟服务器，使用 AWS Lambda 运行代码而不必预置或管理服务器，使用 Amazon ECS 或

Amazon EKS 运行和管理容器，或者使用 AWS Batch 并行处理大量数据。应根据自己的规模和计算需求，选择和配置最适合自己情况的计算解决方案。也可以考虑在单个工作负载中使用多种类型的计算解决方案，因为每种解决方案都有自己的优缺点。

以下步骤将指导您根据自身工作负载的特征和性能要求，选择合适的计算方案。

实施步骤

- 了解工作负载计算要求。需要考虑的关键要求包括：处理需求、流量模式、数据访问模式、扩展需求和延迟要求。
- 了解适用于工作负载的不同 AWS 计算服务。有关更多信息，请参阅 PERF01-BP01 了解并掌握可用的云服务和功能。以下介绍了一些关键的 AWS 计算方案、这些方案的特征和常见应用场景：

AWS 服务	主要特性	常见使用案例
Amazon Elastic Compute Cloud (Amazon EC2)	具有硬件专用选项、许可证要求，多种不同实例系列、处理器类型和计算加速器可供选择	直接迁移、整体式应用程序、混合环境、企业应用程序
Amazon Elastic Container Service（Amazon ECS）、Amazon Elastic Kubernetes Service（Amazon EKS）	轻松部署、环境一致、可扩展	微服务、混合环境
AWS Lambda	无服务器计算服务运行代码来响应事件并自动管理底层计算资源。	微服务、事件驱动的应用程序
AWS Batch	高效、动态地预置和扩展 Amazon Elastic Container Service（Amazon ECS）、Amazon Elastic Kubernetes Service（Amazon EKS）和 AWS Fargate 计算资源，并可根据任务要求选择使用按需型实例或竞价型实例	HPC，训练机器学习模型

AWS 服务	主要特性	常见使用案例
Amazon Lightsail	预先配置的 Linux 和 Windows 应用程序，用于运行小型工作负载	简单的 Web 应用程序、自定义网站

- 评估与每种计算方案相关的成本（如每小时费用或数据传输）和管理开销（如修补和扩展）。
- 在非生产环境中进行试验和基准测试，确定哪种计算方案最能满足工作负载要求。
- 试验并确定新的计算解决方案后，规划迁移并验证性能指标。
- 使用 Amazon CloudWatch 等 AWS 监控工具和 AWS Compute Optimizer 等优化服务，根据实际使用模式持续优化计算资源。

资源

相关文档：

- 使用 AWS 进行云计算
- Amazon EC2 实例类型
- Amazon EKS 容器：Amazon EKS Worker 节点
- Amazon ECS 容器：Amazon ECS 容器实例
- 函数：Lambda 函数配置
- Prescriptive Guidance for Containers
- Prescriptive Guidance for Serverless

相关视频：

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AMS
- AWS re:Invent 2023 - What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Powering next-gen Amazon Elastic Compute Cloud: Deep dive on the Nitro System
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute

- AWS re:Invent 2019 - Amazon Elastic Compute Cloud foundations
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- Amazon EC2 foundations
- 部署 ML 模型，以便进行高性能和低成本的推理

相关示例：

- Migrating the Web application to containers
- 运行无服务器程序“Hello World”
- Amazon EKS 研讨会
- Amazon EC2 讲习会
- Efficient and Resilient Workloads with Amazon Elastic Compute Cloud Auto Scaling
- Migrating to AWS Graviton with Container Services

PERF02-BP02 了解可用的计算配置和功能

了解计算服务的可用配置选项和功能，帮助预置适量的资源并提高性能效率。

常见反模式：

- 没有依据工作负载特征评估计算方案或可用的实例系列。
- 过度预置计算资源来满足高峰需求。

建立此最佳实践的好处：熟悉 AWS 计算功能和配置，以便使用经过优化的计算解决方案，满足工作负载特征和需求。

在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

每种计算解决方案都有独特的配置和功能，可支持不同的工作负载特征和需求。了解这些方案如何完善工作负载，并确定哪些配置选项最适合您的应用程序。这些选项的示例包括实例系列、规模、功能（GPU、I/O）、突增、超时、函数大小、容器实例和并发。如果工作负载已经使用同一计算方案超过四周，并且预计这些特征在未来将保持不变，则可以使用 AWS Compute Optimizer 从 CPU 和内存角度查明当前计算方案是否适合工作负载。

实施步骤

- 了解工作负载要求（如 CPU 需求、内存和延迟）。
- 查看 AWS 文档和最佳实践，了解有助于提高计算性能的推荐配置选项。以下是一些需要考虑的关键配置选项：

配置选项	示例
实例类型	• 计算优化型实例非常适合需要较高 vCPU 与内存比的工作负载。 • 内存优化型实例提供大量内存来支持内存密集型工作负载。 • 存储优化型实例专为需要对本地存储进行大量顺序读写访问（IOPS）的工作负载而设计。
定价模型	• 按需型实例允许按小时或按秒使用计算容量，而无需做出长期承诺。这些实例非常适合超出性能基准需求的突增情况。 • 相比按需型实例，节省计划能够节省大量成本，从而换取在一年或三年内使用特定数量计算能力的承诺。 • 竞价型实例允许以折扣价将未使用的实例容量用于无状态的容错工作负载。
Auto Scaling	使用自动扩缩配置来让计算资源与流量模式相匹配。
调整大小	• 借助 Compute Optimizer，可以就哪种计算配置最符合计算特征，获得基于机器学习的建议。 • 使用 AWS Lambda Power Tuning 为 Lambda 函数选择最佳配置。
基于硬件的计算加速器	• 与基于 CPU 的替代方案相比，加速型计算实例执行图形处理或数据模式匹配等功能的效率更高。

配置选项	示例
	• 对于机器学习工作负载，请利用特定于工作负载的专用硬件，例如 AWS Trainium、AWS Inferentia 和 Amazon EC2 DL1

资源

相关文档：

- 使用 AWS 进行云计算
- Amazon EC2 实例类型
- Amazon EC2 实例的处理器状态控制
- Amazon EKS 容器：Amazon EKS Worker 节点
- Amazon ECS 容器：Amazon ECS 容器实例
- 函数：Lambda 函数配置

相关视频：

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations
- AWS re:Invent 2022 – Optimizing Amazon EKS for performance and cost on AWS

相关示例：

- Compute Optimizer 演示代码
- Amazon EC2 竞价型实例讲习会
- Efficient and Resilient Workloads with Amazon EC2 AWS Auto Scaling
- Graviton 开发人员讲习会

- AWS for Microsoft workloads immersion day
- AWS for Linux workloads immersion day
- AWS Compute Optimizer 演示代码
- Amazon EKS 讲习会

PERF02-BP03 收集与计算相关的指标

记录和跟踪与计算相关的指标，以便更好地了解计算资源的表现情况，并提高计算资源的性能和利用率。

常见反模式：

- 只手动搜索日志文件来查找指标。
- 只使用由监控软件记录的默认指标。
- 只在出现问题时审查指标。

建立此最佳实践的好处：收集与性能相关的指标有助于您根据业务要求调整应用程序性能，从而确保满足工作负载需求。收集指标还有利于您持续提高工作负载中的资源性能和利用率。

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

云工作负载会生成大量数据，例如指标、日志和事件。在 AWS Cloud 中，收集指标是提高安全性、成本效率、性能和可持续性的关键步骤。AWS 使用监控服务（如 Amazon CloudWatch）提供各种与性能相关的指标，从而为您提供宝贵的洞察。CPU 利用率、内存利用率、磁盘 I/O 以及网络入站和出站等指标有助于您深入了解利用率水平或性能瓶颈。将这些指标用作数据驱动方法的一部分，以便主动调整和优化工作负载的资源。 理想情况下，您应该在单一平台上收集与计算资源相关的所有指标，并实施留存策略以支持成本目标和运营目标。

实施步骤

- 确定哪些与性能相关的指标与您的工作负载相关。您应该收集有关资源利用率和云工作负载运行方式的指标（例如响应时间和吞吐量）。
  - Amazon EC2 默认指标
  - Amazon ECS 默认指标
  - Amazon EKS 默认指标

    - Lambda 默认指标
    - Amazon EC2 内存和磁盘指标
- 为工作负载选择并设置合适的日志记录和监控解决方案。
    - AWS native Observability
    - 适用于 OpenTelemetry 的 AWS Distro
    - Amazon Managed Service for Prometheus
- 根据工作负载要求为指标确定所需的筛选和聚合。
    - Quantify custom application metrics with Amazon CloudWatch Logs and metric filters
    - Collect custom metrics with Amazon CloudWatch strategic tagging
- 为指标配置数据留存策略，从而符合安全目标和运营目标。
    - CloudWatch 指标的默认数据留存
    - CloudWatch Logs 的默认数据留存
- 如有需要，可为指标创建警报和通知，协助您主动应对与性能相关的问题。
    - Create alarms for custom metrics using Amazon CloudWatch anomaly detection
    - Create metrics and alarms for specific web pages with Amazon CloudWatch RUM
- 使用自动化技术来部署指标和日志聚合代理。
    - AWS Systems Manager 自动化
    - OpenTelemetry Collector

资源

相关文档：

- 监控和可观测性
- Best practices: implementing observability with AWS
- Amazon CloudWatch 文档
- 使用 CloudWatch 代理从 Amazon EC2 实例和本地部署服务器中收集指标和日志
- 访问 AWS Lambda 的 Amazon CloudWatch Logs
- 将 CloudWatch Logs 与容器实例结合使用
- 发布自定义指标
- AWS Answers：集中式日志记录
- 发布 CloudWatch 指标的 AWS 服务

- Monitoring Amazon EKS on AWS Fargate

相关视频：

- AWS re:Invent 2023 – [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 – Implementing application observability
- AWS re:Invent 2023 – Building an effective observability strategy
- AWS re:Invent 2023 – Seamless observability with AWS Distro for OpenTelemetry
- Application Performance Management on AWS

相关示例：

- AWS for Linux Workloads Immersion Day- Amazon CloudWatch
- Monitoring Amazon ECS clusters and containers
- Monitoring with Amazon CloudWatch dashboards
- Amazon EKS 讲习会

PERF02-BP04 配置计算资源并合理调整资源规模

配置计算资源并合理调整资源规模，使其满足您工作负载的性能要求，避免资源利用不足或过度利用。

常见反模式：

- 忽略工作负载性能要求，导致计算资源预置过度或预置不足。
- 只选择适用于所有工作负载的最大或最小实例。
- 为了便于管理，只使用一个实例系列。
- 忽略来自 AWS Cost Explorer 或 Compute Optimizer 的关于合理调整规模的建议。
- 没有重新评估新实例类型是否适合工作负载。
- 只为组织认证少量实例配置。

建立此最佳实践的好处：合理调整计算资源的规模后，可避免资源预置过度和预置不足，从而确保资源在云端以最佳方式运行。适当调整计算资源的规模通常可以提高性能和改进客户体验，同时还可以降低成本。

在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

合理调整规模使组织能够以经济高效的方式运营云基础设施，同时满足业务需求。云资源预置过度可能会导致额外成本，而预置不足可能导致性能和客户体验不佳。AWS 提供 AWS Compute Optimizer 和 AWS Trusted Advisor 之类的工具，这些工具使用历史数据为计算资源提供合理调整规模的建议。

实施步骤

- 选择最能满足您需求的实例类型：
  - 如何为我的工作负载选择适当的 Amazon EC2 实例类型？
  - Amazon EC2 Fleet 的基于属性的实例类型选择
  - 使用基于属性的实例类型选择创建自动扩缩组
  - Optimizing your Kubernetes compute costs with Karpenter consolidation
- 分析您的工作负载的各种性能特性，以及这些特性与内存、网络和 CPU 使用率之间的关系。根据这些数据选择最符合您的工作负载情况和性能目标的资源。
- 使用 AWS 监控工具（如 Amazon CloudWatch）监控资源使用情况。
- 为计算资源选择合适的配置。
  - 对于临时工作负载，请评估实例 Amazon CloudWatch 指标（例如 `CPUUtilization`），确定实例是利用不足还是利用过度。
  - 对于稳定工作负载，请定期检查 AWS 合理调整规模工具（如 AWS Compute Optimizer 和 AWS Trusted Advisor），从而挖掘优化计算资源和合理调整计算资源规模的机会。
- 在实际环境中实施之前，先在非生产环境中测试配置更改。
- 持续重新评估新的计算产品/服务，并与工作负载的需求进行比较。

资源

相关文档：

- 使用 AWS 进行云计算
- Amazon EC2 实例类型
- Amazon ECS 容器：Amazon ECS 容器实例
- Amazon EKS 容器：Amazon EKS Worker 节点
- 函数：Lambda 函数配置
- Amazon EC2 实例的处理器状态控制

相关视频：

- Amazon EC2 foundations
- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

相关示例：

- AWS Compute Optimizer 演示代码
- Amazon EKS 讲习会
- 合理调整规模建议

PERF02-BP05 动态扩展计算资源

利用云的弹性根据需求动态增减计算资源，避免为工作负载预置的容量过多或者不足。

常见反模式：

- 通过手动增加容量来对警报做出反应。
- 使用本地所用的规模调整指南（通常是静态基础设施）。
- 在扩展事件之后保留增加的容量，而不是缩减容量。

建立此最佳实践的好处：配置和测试计算资源的弹性将有助于您节省资金、维护性能基准，以及在流量变化时提高可靠性。

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

AWS 让您能够通过各种扩展机制灵活地动态扩展或缩减资源，以便满足不断变化的需求。动态扩展结合计算相关的指标，可使工作负载自动响应变化，并利用一系列最优的计算资源来实现目标。

您可以使用大量不同方法来实现资源的供需匹配。

- 目标跟踪方法：监控您的扩缩指标，并根据需要自动增加或减少容量。
- 预测性扩缩：根据每日和每周的趋势进行扩缩。
- 基于计划的方法：根据可预测的负载变化设置自己的扩缩计划。
- 服务扩缩：选择可根据设计自动扩缩的服务（如无服务器）。

您必须确保工作负载部署可以处理扩展事件和缩减事件。

实施步骤

- 计算实例、容器和函数都能够与自动扩缩服务相结合或作为此服务的一项功能来提供可实现弹性的机制。以下是自动扩缩机制的一些示例：

自动扩缩机制	使用情形
Amazon EC2 Auto Scaling	确保您具有正确数量的 Amazon EC2 实例，可用于处理应用程序负载。
Application Auto Scaling	自动扩缩 Amazon EC2 以外的各项 AWS 服务的资源，例如 AWS Lambda 函数或 Amazon Elastic Container Service（Amazon ECS）服务。
Kubernetes Cluster Autoscaler/Karpenter	自动扩缩 Kubernetes 集群。

- 扩缩通常与计算服务（如 Amazon EC2 实例或 AWS Lambda 函数）相关。此外，务必考虑配置非计算服务（如 AWS Glue）来满足需求。
- 验证扩缩指标是否与正在部署的工作负载的特性相匹配。如果您正在部署一个视频转码应用程序，预计 CPU 利用率为 100%，但不应将此作为主要指标，而应使用转码作业队列的深度。如果需要，可以对扩缩策略使用自定义指标。要选择正确的指标，请考虑以下关于 Amazon EC2 的指导：
  - 指标应该是有效的利用率指标，并描述实例的繁忙程度。
  - 指标值必须随着自动扩缩组中的实例数按比例增加或减少。
- 确保对自动扩缩组使用动态扩缩而不是手动扩缩。我们还建议在动态扩缩中使用目标跟踪扩缩策略。
- 确认工作负载部署可以同时处理扩展事件和缩减事件。例如，可以使用活动历史记录来验证自动扩缩组的扩缩活动。

- 评估工作负载，得出可预测的模式，从而在预期需求会发生预测性的计划内变化时主动扩缩。预测性扩缩可以避免过度预置容量。有关更多详细信息，请参阅 Predictive Scaling with Amazon EC2 Auto Scaling。

资源

相关文档：

- 使用 AWS 进行云计算
- Amazon EC2 实例类型
- Amazon ECS 容器：Amazon ECS 容器实例
- Amazon EKS 容器：Amazon EKS Worker 节点
- 函数：Lambda 函数配置
- Amazon EC2 实例的处理器状态控制
- Deep Dive on Amazon ECS Cluster Auto Scaling
- Introducing Karpenter – An Open-Source High-Performance Kubernetes Cluster Autoscaler

相关视频：

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

相关示例：

- Amazon EC2 Auto Scaling Group Examples
- Amazon EKS 研讨会
- Scale your Amazon EKS workloads by running on IPv6

PERF02-BP06 使用基于硬件的优化型计算加速器

与基于 CPU 的替代方案相比，使用硬件加速器可以更高效地执行某些功能。

常见反模式：

- 在工作负载中，没有对照性能更高和成本更低的专用实例，对通用实例进行基准测试。
- 使用基于硬件的计算加速器执行任务，而使用基于 CPU 的替代方案能更高效地完成这些任务。
- 不监控 GPU 使用情况。

建立此最佳实践的好处：通过使用基于硬件的加速器 [如图形处理单元（GPU）和现场可编程门阵列（FPGA）]，可以更高效地执行某些处理功能。

在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

加速型计算实例提供对基于硬件的计算加速器（如 GPU 和 FPGA）的访问。这些硬件加速器能够比基于 CPU 的替代方案更有效地执行某些功能，例如图形处理或数据模式匹配。许多加速工作负载（如渲染、转码和机器学习）在资源使用方面变化很大。仅在需要时运行此硬件，并在不需要时自动将其停用，从而提高整体性能效率。

实施步骤

- 确定可以满足要求的加速型计算实例。
- 对于机器学习工作负载，请利用针对工作负载的专用硬件，例如 AWS Trainium、AWS Inferentia 和 Amazon EC2 DL1。AWSInf2 实例等 Inferentia 实例相比同类 Amazon EC2 实例，性能功耗比提升了 50%。
- 收集加速型计算实例的使用情况指标。例如，按照使用 Amazon CloudWatch 收集 NVIDIA GPU 指标所述，使用 CloudWatch 代理收集 GPU 的 `utilization_gpu` 和 `utilization_memory` 等指标。
- 优化硬件加速器的代码、网络运营和设置，确保底层硬件得到充分利用。
  - 优化 GPU 设置
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker AI
- 使用最新的高性能库和 GPU 驱动程序。

- 使用自动化功能在不使用 GPU 实例时将其释放。

资源

相关文档：

- 在 Amazon ECS 上使用 GPU
- GPU 实例
- 使用 AWS Trainium 的实例
- 使用 AWS Inferentia 的实例
- Let's Architect! Architecting with custom chips and accelerators

- 加速计算
- Amazon EC2 VT1 Instances
- 如何为我的工作负载选择适当的 Amazon EC2 实例类型？
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

相关视频：

- AWS re:Invent 2021 - How to select Amazon Elastic Compute Cloud GPU instances for deep learning
- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances
- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

相关示例：

- Amazon SageMaker AI and NVIDIA GPU Cloud (NGC)
- Use SageMaker AI with Trainium and Inferentia for optimized deep learning training and inferencing workloads
- Optimizing NLP models with Amazon Elastic Compute Cloud Inf1 instances in Amazon SageMaker AI

数据管理

问题

- PERF 3. 如何存储、管理和访问工作负载中的数据？

PERF 3. 如何存储、管理和访问工作负载中的数据？

针对特定系统的最佳数据管理解决方案往往取决于数据类型（数据块、文件或对象）、访问模式（随机或连续）、所需吞吐量、访问频率（在线、离线、归档）、更新频率（WORM、动态）以及可用性与持久性限制等因素。Well-Architected 工作负载使用专门构建的数据存储，这些存储允许使用不同的功能来提高性能。

最佳实践

- PERF03-BP01 使用最能满足数据访问和存储要求的专用数据存储
- PERF03-BP02 评估数据存储的可用配置选项
- PERF03-BP03 收集和记录数据存储性能指标
- PERF03-BP04 实施可提高数据存储查询性能的策略
- PERF03-BP05 实施利用缓存的数据访问模式

PERF03-BP01 使用最能满足数据访问和存储要求的专用数据存储

了解数据特性（如数据的可共享性、大小、缓存大小、访问模式、延迟、吞吐量和持久性），为工作负载选择合适的专用数据存储（存储或数据库）。

常见反模式：

- 由于内部对某种特定类型的数据库解决方案具备相关经验且比较了解，因此坚持使用一种数据存储。
- 认为所有工作负载都有类似的数据存储和访问要求。
- 没有实施数据目录来清点数据资产。

建立此最佳实践的好处：了解数据特性和要求，有助于确定效率最高、性能最高的存储技术来满足工作负载需求。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

选择和实施数据存储时，要确保查询、扩展和存储特性支持工作负载数据要求。AWS 提供多种数据存储和数据库技术，包括数据块存储、对象存储、流式存储、文件系统、关系数据库、键值数据库、文档数据库、内存数据库、图形数据库、时间序列数据库和分类账数据库等。每种数据管理解决方案都有可供您使用的选项和配置，可支持应用场景和数据模型。通过了解数据特性和要求，您可以摆脱单一存储技术以及有很多局限性的一刀切方法，专注于合理管理数据。

实施步骤

- 清点工作负载中存在的各种数据类型。
- 了解并记录数据特性和要求，包括：
  - 数据类型（非结构化、半结构化、关系型）
  - 数据量和增长
  - 数据持久性：持久、短暂、瞬时
  - ACID（原子性、一致性、隔离性、持久性）要求
  - 数据访问模式（读取密集型或写入密集型）
  - 延迟
  - 吞吐量
  - IOPS（每秒输入/输出操作数）
  - 数据留存期
- 了解可用于 AWS 工作负载的不同数据存储（存储和数据库服务），这些存储可以满足您的数据特性要求（如 PERF01-BP01 了解并掌握可用的云服务和功能中所述）。AWS 存储技术及其关键特性的一些示例包括：

类型	AWS 服务	主要特性
对象存储	Amazon S3	无限的可扩展性、高可用性以及多种可访问性选项。要在 Amazon S3 内外传输和访问对象，可以使用传输加速或接入点等服务来支持位置、安全需求和访问模式。
存档存储	Amazon S3 Glacier	专为数据存档而打造。

类型	AWS 服务	主要特性
流媒体存储	Amazon Kinesis Amazon Managed Streaming for Apache Kafka（Amazon MSK）	高效摄取和存储流媒体数据。
共享文件系统	Amazon Elastic File System (Amazon EFS)	可装载文件系统，可由多种类型的计算解决方案访问。
共享文件系统	Amazon FSx	基于最新 AWS 计算解决方案而构建，支持四种常用文件系统：NetApp ONTAP、OpenZFS、Windows File Server 和 Lustre。Amazon FSx 延迟、吞吐量和 IOPS 因文件系统而不同，因此，在为您的工作负载需求选择合适的文件系统时应考虑这些因素。
数据块存储	Amazon Elastic Block Store（Amazon EBS)	可扩展、高性能的数据块存储服务，专为 Amazon Elastic Compute Cloud（Amazon EC2）设计。Amazon EBS 包括用于事务型、IOPS 密集型工作负载的 SSD 支持型存储，以及用于吞吐量密集型工作负载的 HDD 支持型存储。
关系数据库	Amazon Aurora、Amazon RDS、Amazon Redshift。	旨在支持 ACID（原子性、一致性、隔离性、持久性）事务，并保持参照完整性和数据强一致性。许多传统应用程序、企业资源规划（ERP）、客户关系管理（CRM）和电子商务都使用关系数据库来存储其数据。

类型	AWS 服务	主要特性
键值数据库	Amazon DynamoDB	已针对常见的访问模式进行优化，通常用于存储和检索大量数据。键值数据库的典型使用案例包括高流量 Web 应用程序、电子商务系统和游戏应用程序。
文档数据库	Amazon DocumentDB	旨在将半结构化数据存储为类似 JSON 的文档。这些数据库可帮助开发人员快速构建和更新应用程序，例如内容管理、目录和用户配置文件。
内存数据库	Amazon ElastiCache、适用于 Redis 的 Amazon MemoryDB	用于需要实时访问数据、最低延迟和最高吞吐量的应用程序。您可以将内存数据库用于应用程序缓存、会话管理、游戏排行榜、低延迟机器学习特征存放区、微服务消息传送系统和高吞吐量流式传输机制。
图形数据库	Amazon Neptune	用于需要大规模以毫秒延迟在高度连接的图形数据集之间浏览和查询数百万关系的应用程序。许多公司将图形数据库用于欺诈检测、社交网络和推荐引擎。
时间序列数据库	Amazon Timestream	用于高效收集、合成数据，并从不断变化的数据中获得见解。IoT 应用程序、DevOps 和工业遥测可以利用时间序列数据库。

类型	AWS 服务	主要特性
宽列	Amazon Keyspaces (for Apache Cassandra)	使用表、行和列，但是与关系数据库不同的是，同一个表中各行的列名称和格式可能会有所不同。宽列存储常见于用于设备维护、队列管理和路线优化的大规模工业应用程序。
分类账	Amazon Quantum Ledger Database (Amazon QLDB)	提供可信中央机构，以维护每个应用程序的可扩展、不可变和允许以加密方式进行验证的交易记录。分类账数据库用于记录系统、供应链、注册甚至银行交易。

- 若要构建数据平台，可利用 AWS 上的现代数据架构来集成数据湖、数据仓库和专用数据存储。
- 为工作负载选择数据存储时需要考虑的关键问题如下：

问题	需要考虑的事项
数据结构如何？	• 若是非结构化数据，可以考虑使用对象存储（例如 Amazon S3）或 NoSQL 数据库（例如 Amazon DocumentDB） • 若是键值数据，可以考虑 DynamoDB、与 Redis OSS 兼容的 Amazon ElastiCache 或 Amazon MemoryDB
需要什么级别的参照完整性？	• 对于外键约束，Amazon RDS 和 Aurora 等关系数据库可以提供这种级别的完整性。 • 通常，在 NoSQL 数据模型中，您可以将数据去规范化到单个文档或文档集合，以便在单个请求中进行检索，而不是跨各文档或各表联接。

问题	需要考虑的事项
是否要求符合 ACID（原子性、一致性、隔离性、持久性）？	• 如果需要与关系数据库关联的 ACID 属性，请考虑使用关系数据库，例如 Amazon RDS 和 Aurora。 • 如果 NoSQL 数据库需要强一致性，则可以在 DynamoDB 中使用强一致性读取。
存储要求将如何随时间变化？ 这对可扩展性有何影响？	• Dynamo DB 和 Amazon Quantum Ledger Database（Amazon QLDB）等无服务器数据库会动态扩展。 • 关系数据库的预置存储空间设有上限，一旦达到这些限制，往往必须使用分片等机制进行水平分区。
读查询与写查询的比例是多少？ 缓存有可能提高性能吗？	• 读取密集型工作负载可以从缓存层中受益，例如 ElastiCache 或 DAX（若数据库为 DynamoDB）。 • 读取操作也可以通过关系数据库（如 Amazon RDS）分流到只读副本。
存储和修改（OLTP – Online Transaction Processing，联机事务处理）还是检索和报告（OLAP – Online Analytical Processing，联机分析处理）具有更高的优先级？	• 对于高吞吐量的按原样读取事务处理，可以考虑使用 NoSQL 数据库，例如 DynamoDB。 • 对于具有一致性的高吞吐量和复杂的读取模式（如联接），请使用 Amazon RDS。 • 对于分析查询，可以考虑使用 Amazon Redshift 等列存数据库，或者将数据导出到 Amazon S3 后使用 Athena 或 Amazon QuickSight 进行分析。

问题	需要考虑的事项
数据需要什么级别的持久性？	• Aurora 自动在一个区域内的三个可用区复制数据，这意味着数据具有高度的持久性，丢失的可能性较小。 • DynamoDB 自动跨多个可用区复制，提供高可用性和数据持久性。 • Amazon S3 提供 11 个 9 的持久性。许多数据库服务（如 Amazon RDS 和 DynamoDB）支持将数据导出到 Amazon S3，以便进行长期留存和存档。
是否希望摆脱商用数据库引擎或许可成本？	• 考虑使用 Amazon RDS 或 Aurora 上的开源引擎，如 PostgreSQL 和 MySQL。 • 利用 AWS Database Migration Service 和 AWS Schema Conversion Tool 执行从商用数据库引擎到开源引擎的迁移
对数据库的运维有什么期望？ 迁移到托管服务是主要的关注点吗？	• 利用 Amazon RDS 而不是 Amazon EC2，以及利用 DynamoDB 或 Amazon DocumentDB 而不是自行托管 NoSQL 数据库，可以减少运维开销。
当前如何访问数据库？ 是只有应用程序访问，还是有商业智能（BI）用户和其他互联的现成应用程序？	• 如果依赖外部工具，可能需要保持与这些工具支持的数据库的兼容性。Amazon RDS 与其支持的差异引擎版本完全兼容，包括 Microsoft SQL Server、Oracle、MySQL 和 PostgreSQL。

- 在非生产环境中进行试验和基准测试，确定哪种数据存储可以满足工作负载要求。

资源

相关文档：

- Amazon EBS 卷类型
- Amazon EC2 存储

- Amazon EFS：Amazon EFS 性能
- 适用于 Lustre 的 Amazon FSx 性能
- 适用于 Windows File Server 的 Amazon FSx 性能
- Amazon S3 Glacier：S3 Glacier 文档
- Amazon S3：请求速率和性能注意事项
- 使用 AWS 进行云存储
- Amazon EBS I/O 特性
- AWS 云数据库
- AWS 数据库缓存
- DynamoDB Accelerator
- Amazon Aurora 最佳实践
- Amazon Redshift 性能
- Amazon Athena 十大性能技巧
- Amazon Redshift Spectrum 最佳实践
- Amazon DynamoDB 最佳实践
- 在 Amazon EC2 和 Amazon RDS 之间进行选择
- 实施 Amazon ElastiCache 的最佳实践

相关视频：

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimizing storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2022: Building modern data architectures on AWS
- AWS re:Invent 2022: Building data mesh architectures on AWS
- AWS re:Invent 2023: Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023: Advanced data modeling with Amazon DynamoDB
- AWS re:Invent 2022: Modernize apps with purpose-built databases
- Amazon DynamoDB deep dive: Advanced design patterns

相关示例：

- AWS Purpose Built Databases 讲习会
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS
- Amazon S3 示例
- Optimize Data Pattern using Amazon Redshift Data Sharing
- Database Migrations
- MS SQL Server - AWS Database Migration Service (AWS DMS) Replication Demo
- Database Modernization Hands On 讲习会
- Amazon Neptune 示例

PERF03-BP02 评估数据存储的可用配置选项

了解并评估数据存储的各种可用功能和配置选项，从而优化工作负载的存储空间和性能。

常见反模式：

- 对所有工作负载都只使用一种存储类型，例如 Amazon EBS。
- 对所有工作负载都使用预调配 IOPS，而没有对所有存储层进行真实测试。
- 不了解所选数据管理解决方案的配置选项。
- 只依赖于增加实例大小，而没有考虑其他可用的配置选项。
- 没有测试数据存储的扩展特性。

建立此最佳实践的好处：通过探索和试用数据存储选项，也许能够降低基础设施成本、提高性能并减少维护工作负载所需的工作量。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

根据数据存储和访问要求，一个工作负载能够使用一个或多个数据存储。要优化性能效率和成本，必须评估数据访问模式来确定适当的数据存储配置。在研究数据存储选项时，要考虑存储选项、内存、计算、只读副本、一致性要求、连接池和缓存选项等各个方面。尝试使用这些不同的配置选项来改进性能效率指标。

实施步骤

- 了解数据存储的当前配置（如实例类型、存储大小或数据库引擎版本）。
- 查看 AWS 文档和最佳实践，了解有助于提高数据存储性能的推荐配置选项。需要考虑的关键数据存储选项如下：

配置选项	示例
分流读取操作（例如只读副本和缓存）	• 对于 DynamoDB 表，您可以使用 DAX 缓存功能来分流读取操作。 • 您可以创建一个 Amazon ElastiCache（Redis OSS）集群，并将应用程序配置为首先从缓存中读取，并在请求的项目不存在时使用数据库。 • 关系数据库（如 Amazon RDS 和 Aurora）以及预置的 NoSQL 数据库（如 Neptune 和 Amazon DocumentDB）全部支持添加只读副本，以便分流工作负载的读取部分。 • DynamoDB 等无服务器数据库将自动扩展。确保您预置了足够的读取容量单位（RCU）来处理工作负载。
扩展写入（例如分区键分片或引入队列）	• 对于关系数据库，您可以增加实例的大小来适应增加的工作负载，或增加预调配 IOPS 来增加底层存储的吞吐量。 • 您还可以在数据库前面引入队列，而不是直接写入数据库。借助此模式，您可以将摄取操作与数据库解耦，并控制流量，这样数据库就不会过载。 • 对写入请求进行批处理，而不是创建许多短期事务，这样有助于提高有大量写入的关系数据库的吞吐量。 • 像 DynamoDB 这样的无服务器数据库可以自动扩展写入吞吐量，也可以根据容量模式调整预置的写入容量单位（WCU）。

配置选项	示例
	• 当达到给定分区键的吞吐量限制时，仍然会遇到热分区问题。这可以通过选择更均匀分布的分区键或对分区键进行写分片来缓解。
用于管理数据集生命周期的策略	• 您可以使用 Amazon S3 生命周期在对象的整个生命周期中对其进行管理。如果您的访问模式未知、变化或不可预测，则可以使用 Amazon S3 Intelligent-Tiering，它能够监控访问模式并自动将尚未访问的对象移动到成本较低的访问层。您可以利用 Amazon S3 Storage Lens 存储统计管理工具指标来识别生命周期管理中的优化机会和差距。 • Amazon EFS 生命周期管理会自动为文件系统管理文件存储。
连接管理和连接池	• Amazon RDS 代理可与 Amazon RDS 和 Aurora 结合使用来管理与数据库的连接。 • DynamoDB 等无服务器数据库没有与之关联的连接，但会考虑根据预置容量和自动扩展策略来处理负载峰值。

- 在非生产环境中进行试验和基准测试，确定哪种配置选项可以满足工作负载要求。
- 试验完成后，规划迁移并验证性能指标。
- 使用 AWS 监控工具（如 Amazon CloudWatch）和优化工具（如 Amazon S3 Storage Lens 存储统计管理工具），在实际使用模式下持续优化数据存储。

资源

相关文档：

- 使用 AWS 进行云存储
- Amazon EBS 卷类型
- Amazon EC2 存储
- Amazon EFS：Amazon EFS 性能
- 适用于 Lustre 的 Amazon FSx 性能

- 适用于 Windows File Server 的 Amazon FSx 性能
- Amazon S3 Glacier：S3 Glacier 文档
- Amazon S3：请求速率和性能注意事项
- Amazon EBS I/O 特性
- AWS 云数据库
- AWS 数据库缓存
- DynamoDB Accelerator
- Amazon Aurora 最佳实践
- Amazon Redshift 性能
- Amazon Athena 十大性能技巧
- Amazon Redshift Spectrum 最佳实践
- Amazon DynamoDB 最佳实践

相关视频：

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimize storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2023: What's new with AWS file storage
- AWS re:Invent 2023: Dive deep into Amazon DynamoDB

相关示例：

- AWS Purpose Built Databases 讲习会
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Amazon EBS Autoscale
- Amazon S3 示例
- Amazon DynamoDB 示例
- AWS 数据库迁移示例
- Database Modernization 讲习会

- Working with parameters on your Amazon RDS for Postgress DB

PERF03-BP03 收集和记录数据存储性能指标

跟踪并记录数据存储的相关性能指标，了解数据管理解决方案的执行情况。这些指标有助于您优化数据存储，验证是否满足工作负载要求，并清晰地概述工作负载的表现情况。

常见反模式：

- 只手动搜索日志文件来查找指标。
- 只将指标发布到团队使用的内部工具，而没有全面了解工作负载。
- 只使用由自己选定的监控软件记录的默认指标。
- 只在出现问题时审查指标。
- 只监控系统级指标，而不捕获数据访问或使用情况指标。

建立此最佳实践的好处：建立性能基准有助于了解工作负载的正常行为和要求。可以更快地识别和调试异常模式，从而提高数据存储的性能和可靠性。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

要监控数据存储的性能，必须记录一段时间的多项性能指标。这样您便可以检测异常并根据业务指标衡量性能，确保满足您的工作负载需求。

指标既应包括支持数据存储的底层系统指标，也应包括数据库指标。底层系统指标可能包括 CPU 利用率、内存、可用磁盘存储、磁盘 I/O、缓存命中率以及网络入站和出站指标，而数据存储指标可能包括每秒事务数、最多的查询、平均查询速率、响应时间、索引使用情况、表锁定、查询超时和打开的连接数。这些数据对于了解工作负载的表现情况以及数据管理解决方案的使用方式至关重要。在数据驱动方法中使用这些指标，以便调整和优化工作负载的资源。

使用各种工具、库和系统来记录与数据库性能相关的性能测量值。

实施步骤

- 确定要跟踪的数据存储关键性能指标。
  - Amazon S3 指标与维度
  - 监控 Amazon RDS 实例中的指标
  - 在 Amazon RDS 上使用性能详情监控数据库负载

  - 增强监测概述
  - DynamoDB 指标与维度
  - 监控 DynamoDB Accelerator
  - 使用 Amazon CloudWatch 监控 Amazon MemoryDB
  - 应监控哪些指标？
  - 监控 Amazon Redshift 集群性能
  - Timestream 指标与维度
  - Amazon Aurora 的 Amazon CloudWatch 指标
  - 在 Amazon Keyspaces（Apache Cassandra 兼容）中记录和监控
  - 监控 Amazon Neptune 资源
- 使用经批准的日志记录和监控解决方案来收集这些指标。Amazon CloudWatch 可以收集架构中各种资源的指标。您也可以收集和发布自定义指标，用于显示业务指标或派生指标。使用 CloudWatch 或第三方解决方案来设置超出阈值时显示的警报。
- 检查数据存储监控，确定其能否受益于可检测性能异常的机器学习解决方案。
  - Amazon DevOps Guru for Amazon RDS 会显示性能问题，并提出纠正措施的建议。
- 在监控和日志记录解决方案中配置数据留存，从而满足您的安全和运营目标。
  - CloudWatch 指标的默认数据留存
  - CloudWatch Logs 的默认数据留存

资源

相关文档：

- AWS 数据库缓存
- Amazon Athena 十大性能技巧
- Amazon Aurora 最佳实践
- DynamoDB Accelerator
- Amazon DynamoDB 最佳实践
- Amazon Redshift Spectrum 最佳实践
- Amazon Redshift 性能
- AWS 云数据库
- Amazon RDS 性能详情

相关视频：

- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Database Performance Monitoring and Tuning with Amazon DevOps Guru for Amazon RDS
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- Best Practices for Monitoring Redis Workloads on Amazon ElastiCache

相关示例：

- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS Monitorng 讲习会
- AWS Purpose Built Databases 讲习会

PERF03-BP04 实施可提高数据存储查询性能的策略

实施可优化数据和改进数据查询的策略，从而提高工作负载的可扩展性和性能效率。

常见反模式：

- 没有对数据存储中的数据进行分区。
- 在数据存储中只以一种文件格式存储数据。
- 没有在数据存储中使用索引。

建立此最佳实践的好处：优化数据和查询性能可以提高效率、降低成本并改善用户体验。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

数据优化和查询调整是提高数据存储性能效率的关键环节，因为这会影响整个云工作负载的性能和响应能力。如果查询未经优化，则会耗用更多的资源并产生更多的瓶颈，从而降低数据存储的整体效率。

数据优化会涵盖多种技术，旨在确保高效的数据存储和访问，同时还有助于改进在数据存储中的查询性能。关键策略包括数据分区、数据压缩和数据去规范化，这有助于针对存储和访问优化数据。

实施步骤

- 了解并分析在数据存储中执行的关键数据查询。
- 识别数据存储中运行速度较慢的查询，并使用查询计划了解当前状态。
  - 在 Amazon Redshift 中分析查询计划
  - 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE
- 实施可提高查询性能的策略。一些关键策略包括：
  - 使用列式文件格式（如 Parquet 或 ORC）。
  - 压缩数据存储中的数据，减少存储空间和 I/O 操作。
  - 进行数据分区，将数据分割成更小的部分，减少数据扫描时间。
    - 在 Athena 中对数据进行分区
    - 分区和数据分发
  - 对查询中的常用列编制数据索引。
  - 使用实体化视图频繁地进行查询。
    - 了解实体化视图
    - 在 Amazon Redshift 中创建实体化视图
  - 为查询选择合适的联接操作。联接两个表时，请在联接的左侧指定较大的表，在联接的右侧指定较小的表。
  - 实施分布式缓存解决方案，从而缩短延迟并减少数据库 I/O 操作次数。
  - 执行定期维护，例如 vaccum 操作、重新索引和进行统计。
- 在非生产环境中试验和测试策略。

资源

相关文档：

- Amazon Aurora 最佳实践
- Amazon Redshift 性能
- Amazon Athena 十大性能技巧
- AWS 数据库缓存

- 实施 Amazon ElastiCache 的最佳实践
- 在 Athena 中对数据进行分区

相关视频：

- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Optimize Amazon Athena Queries with New Query Analysis Tools

相关示例：

- AWS Purpose Built Databases 讲习会

PERF03-BP05 实施利用缓存的数据访问模式

实施可从缓存数据受益的访问模式，以便快速检索经常访问的数据。

常见反模式：

- 缓存经常变化的数据。
- 依赖缓存的数据，就好像这些数据是持久存储的，并且始终可用。
- 不考虑缓存数据的一致性。
- 不监控缓存实现方案的效率。

建立此最佳实践的好处：将数据存储在缓存中可以改善读取延迟、读取吞吐量、用户体验和整体效率，还可以降低成本。

在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

缓存是一种软件或硬件组件，旨在存储数据，以便将来可以更快或更高效地处理对相同数据的请求。如果存储在缓存中的数据丢失，则可以通过重复先前的计算或从其他数据存储中获取数据进行重建。

数据缓存可能是提高应用程序整体性能和减轻底层主数据源负担的最有效策略之一。数据可以在应用程序的多个级别上缓存，例如在进行远程调用的应用程序内缓存（称作客户端缓存），或者使用快速辅助服务来存储数据（称作远程缓存）。

客户端缓存

借助客户端缓存，每个客户端（查询后端数据存储的应用程序或服务）都可以在本地将特定查询的结果存储指定的时间。可以通过先检查本地客户端缓存，减少通过网络向数据存储发出的请求数量。如果结果不存在，则应用程序可以查询数据存储并将这些结果存储在本地。这种模式允许每个客户端将数据存储在尽可能近的位置（客户端本身），从而尽可能降低延迟。当后端数据存储不可用时，客户端还可以继续支持某些查询，从而提高整个系统的可用性。

这种方法的一个缺点是，当涉及多个客户端时，它们可能会在本地存储相同的缓存数据。这会导致这些客户端之间存在重复的存储使用情况和数据不一致性。一个客户端可能刚缓存查询结果，而一分钟后，另一个客户端可能运行相同的查询并得到不同的结果。

远程缓存

为了解决客户端之间的重复数据问题，可以使用快速的外部服务或远程缓存来存储查询的数据。在查询后端数据存储之前，每个客户端都将检查远程缓存，而不是检查本地数据存储。这种策略可在客户端之间实现更加一致的响应、更高的存储数据效率以及更高的缓存数据量，因为存储空间可独立于客户端进行扩展。

远程缓存的缺点是整个系统的延迟可能会更高，因为需要额外的网络跃点数来检查远程缓存。客户端缓存可以与远程缓存一起使用，形成多级缓存来缩短延迟。

实施步骤

- 确定可以从缓存中受益的数据库、API 和网络服务。读取工作负载繁重、读写比率高或扩展成本高昂的服务适合使用缓存。
  - 数据库缓存
  - 启用 API 缓存以增强响应能力
- 确定最适合您的访问模式的适当缓存策略类型。
  - 缓存策略
  - AWS 缓存解决方案
- 遵循数据存储的缓存最佳实践。
- 为所有数据配置缓存失效策略，例如生存时间（TTL），以平衡数据的时效性并减轻后端数据存储的压力。
- 启用诸如自动连接重试、指数回退、客户端超时和客户端连接池等功能（如果有），因为它们可以提高性能和可靠性。
  - Best practices: Redis clients and Amazon ElastiCache for Redis

- 监控缓存命中率，目标为 80% 或更高。低于此值可能表示缓存大小不足，或访问模式无法从缓存中受益。
  - Which metrics should I monitor?
  - Best practices for monitoring Redis workloads on Amazon ElastiCache
  - Monitoring best practices with Amazon ElastiCache (Redis OSS) using Amazon CloudWatch
- 实施数据复制，将读取操作分流到多个实例，以提高数据读取性能和可用性。

资源

相关文档：

- Using the Amazon ElastiCache Well-Architected Lens
- Monitoring best practices with Amazon ElastiCache (Redis OSS) using Amazon CloudWatch
- 应监控哪些指标？
- Performance at Scale with Amazon ElastiCache 白皮书
- 缓存挑战和策略

相关视频：

- Amazon ElastiCache Learning Path
- Design for success with Amazon ElastiCache best practices
- AWS re:Invent 2020 - Design for success with Amazon ElastiCache best practices
- AWS re:Invent 2023 - [LAUNCH] Introducing Amazon ElastiCache Serverless
- AWS re:Invent 2022 - 5 great ways to reimagine your data layer with Redis
- AWS re:Invent 2021 - Deep dive on Amazon ElastiCache (Redis OSS)

相关示例：

- 使用 Amazon ElastiCache for Redis 提升 MySQL 数据库性能

网络和内容分发

问题

- PERF 4. 如何在工作负载中选择和配置网络资源？

PERF 4. 如何在工作负载中选择和配置网络资源？

适合某个工作负载的最佳网络解决方案会因延迟、吞吐量要求、抖动和带宽而有所不同。物理约束（例如用户资源或本地资源）决定位置选项。这些约束可以通过边缘站点或资源置放来抵消。

最佳实践

- PERF04-BP01 了解联网对性能的影响
- PERF04-BP02 评估可用的联网功能
- PERF04-BP03 为工作负载选择合适的专用连接或 VPN
- PERF04-BP04 使用负载平衡在多个资源之间分配流量
- PERF04-BP05 选择网络协议以提高性能
- PERF04-BP06 根据网络要求选择工作负载的位置
- PERF04-BP07 根据指标优化网络配置

PERF04-BP01 了解联网对性能的影响

分析并了解与网络相关的决策如何影响您的工作负载，从而提供更高的性能和更好的用户体验。

常见反模式：

- 所有流量都会流经现有的数据中心。
- 通过中央防火墙路由所有流量，而不是使用云原生网络安全工具。
- 在不了解实际使用要求的情况下预置 AWS Direct Connect 连接。
- 在确立联网解决方案时，未考虑工作负载特性和加密开销。
- 将本地概念和策略用于云中的联网解决方案。

建立此最佳实践的好处：通过了解联网如何影响工作负载性能，有助于您识别潜在的瓶颈、改善用户体验、提高可靠性并在工作负载发生变化时减少运营维护。

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

网络负责应用程序组件、云服务、边缘网络和本地数据之间的连接，因此，它会严重影响工作负载性能。除了工作负载性能之外，用户体验也会受到网络延迟、带宽、协议、位置、网络拥塞、抖动、吞吐量和路由规则的影响。

清楚记录工作负载的联网要求列表，包括延迟、数据包大小、路由规则、协议和支持的流量模式。查看可用的联网解决方案，并确定哪种服务与您的工作负载联网特性相符。基于云的网络可以快速重建，因此有必要随着时间的推移改进网络架构，以提高性能效率。

实施步骤：

- 定义和记录网络性能要求，包括网络延迟、带宽、协议、位置、流量模式（峰值和频率）、吞吐量、加密、检查和路由规则等指标。
- 了解关键 AWS 联网服务，例如 VPC、AWS Direct Connect、弹性负载均衡（ELB）以及 Amazon Route 53。
- 捕获以下关键网络特性：

特性	工具和指标
基础网络特性	• VPC 流日志 • AWS Transit Gateway 流日志 • AWS Transit Gateway 指标 • AWS PrivateLink 指标
应用程序网络特性	• Elastic Fabric Adapter • AWS App Mesh 指标 • Amazon API Gateway 指标
边缘网络特性	• Amazon CloudFront 指标 • Amazon Route 53 指标 • AWS Global Accelerator 指标
混合网络特性	• AWS Direct Connect 指标 • AWS Site-to-Site VPN 指标 • AWS Client VPN 指标 • AWS Cloud WAN 指标
安全网络特性	• AWS Shield、AWS WAF 和 AWS Network Firewall 指标
跟踪特性	• AWS X-Ray

特性	工具和指标
	• VPC Reachability Analyzer • 网络访问分析器 • Amazon Inspector • Amazon CloudWatch RUM

- 对网络性能进行基准测试和其他测试：
  - 对网络吞吐量进行基准测试，因为当实例位于同一 VPC 中时，一些因素可能会影响 Amazon EC2 网络性能。测量同一 VPC 中的 Amazon EC2 Linux 实例之间的网络带宽。
  - 执行负载测试以试用各种联网解决方案和选项。

资源

相关文档：

- 应用程序负载均衡器
- Linux EC2 上的增强联网功能
- Windows EC2 上的增强联网功能
- EC2 置放群组
- 在 Linux 实例上启用弹性网络适配器（ENA）增强联网功能
- Network Load Balancer
- AWS 联网产品
- Transit Gateway
- Transitioning to latency-based routing in Amazon Route 53
- VPC 端点

相关视频：

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - What can networking do for your application?
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2023 - A developer’s guide to cloud networking
- AWS re:Invent 2019 - Connectivity to AWS and hybrid AWS network architectures

- AWS re:Invent 2019 - Optimizing Network Performance for Amazon EC2 Instances
- AWS Summit Online - Improve Global Network Performance for Applications
- AWS re:Invent 2020 - Networking best practices and tips with the Well-Architected Framework
- AWS re:Invent 2020 - AWS networking best practices in large-scale migrations

相关示例：

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 讲习会
- Hands-on Network Firewall 讲习会
- Observing and Diagnosing your Network on AWS
- Finding and addressing Network Misconfigurations on AWS

PERF04-BP02 评估可用的联网功能

评估云中可能提高性能的联网功能。借助测试、指标和分析来衡量这些功能的影响。例如，利用可用的网络级功能来减少延迟、网络距离或抖动。

常见反模式：

- 一直待在一个区域，因为这是总部实际所在的区域。
- 使用防火墙而不是安全组来过滤流量。
- 中断 TLS 来进行流量检查，而不是依赖安全组、端点策略和其他云原生功能。
- 只使用基于子网的分段，而不是安全组。

建立此最佳实践的好处：评估所有服务功能和选项可以提高您的工作负载性能，降低基础设施的成本，减少维护工作负载所需的工作量，并提升您的整体安全态势。您可以利用 AWS 的全球主干网，为客户提供出色的联网体验。

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

AWS 提供有助于提高网络性能的 AWS Global Accelerator 和 Amazon CloudFront 等服务，而大多数 AWS 服务都具有用于优化网络流量的产品功能（例如 Amazon S3 Transfer Acceleration 功能）。

查看您可以使用哪些与网络相关的配置选项，以及这些配置选项对您的工作负载有何影响。要想优化性能，需要了解这些选项如何与您的架构进行交互，以及它们将对测得的性能和用户体验产生的影响。

实施步骤

- 创建工作负载组件列表。
  - 在构建统一全球网络时，考虑使用 AWS Cloud WAN 来构建、管理和监控您组织的网络。
  - 使用 Amazon CloudWatch Logs 指标监控您的全球与核心网络。利用 Amazon CloudWatch RUM，它提供了有助于识别、理解和增强用户的数字体验的见解。
  - 查看 AWS 区域 和可用区之间以及每个可用区内的聚合网络延迟，使用 AWS Network Manager 深入了解应用程序性能与 AWS 底层网络性能的关系。
  - 使用现有的配置管理数据库（CMDB）工具或 AWS Config 等服务创建工作负载清单及其配置方式。
- 如果这是一个现有的工作负载，请确定并记录性能指标的基准，重点关注瓶颈和需要改进之处。受业务要求和工作负载特性的影响，与性能相关的联网指标会因工作负载而异。首先，对于您的工作负载，检查带宽、延迟、数据包丢失、抖动和重传等指标可能很重要。
- 如果这是新的工作负载，请执行负载测试来确定性能瓶颈。
- 对于识别的性能瓶颈，请查看解决方案的配置选项，以确定性能改进机会。查看以下主要联网选项和功能：

改进机会	解决方案
网络路径或路由	使用网络访问分析器来确定路径或路由。
网络协议	请参阅 PERF04-BP05 选择网络协议以提高性能。
网络拓扑	当连接多个账户时，请评估 VPC 对等连接与 AWS Transit Gateway 之间的运营和性能权衡。AWS Transit Gateway 可简化所有 VPC 之间的互连，这些 VPC 可以跨越数千个 AWS 账户 并接入您的本地网络。使用 AWS Resource Access Manager 在多个账户之间共享您的 AWS Transit Gateway。

改进机会	解决方案
	请参阅 PERF04-BP03 为工作负载选择合适的专用连接或 VPN。
网络服务	AWS Global Accelerator 是一项网络服务，使用 AWS 全球网络基础设施，可将用户流量的性能提高多达 60%。 Amazon CloudFront 可在全球范围内提高工作负载内容分发性能并减少延迟。 使用 Lambda@Edge 运行一些函数，这些函数可自定义 CloudFront 在离用户更近的位置提供的内容、减少延迟并提高性能。 Amazon Route 53 提供基于延迟的路由、地理位置路由、地理位置邻近度路由和基于 IP 的路由选项，有助于提高面向全球受众的工作负载性能。如果工作负载分布在全球，通过查看工作负载流量和用户位置，确定哪种路由选项可以优化工作负载性能。
存储资源功能	Amazon S3 Transfer Acceleration 功能可让外部用户在向 Amazon S3 传输数据时通过 CloudFront 的网络优化获益。这就提高了将大量数据从没有专用连接的远程位置传输到 AWS Cloud 的能力。 Amazon S3 多区域接入点将内容复制到多个区域，并通过提供一个接入点简化了工作负载。使用多区域接入点时，您可以使用标识最低延迟存储桶的服务向 Amazon S3 请求或写入数据。

改进机会	解决方案
计算资源功能	Amazon EC2 实例、容器和 Lambda 函数使用的弹性网络接口（ENA）按流进行限制。查看置放群组以优化 EC2 联网吞吐量。为避免每个流上出现瓶颈，请将应用程序设计为使用多个流。要监控和查看与计算相关的联网指标，请使用 CloudWatch 指标和 ethtool。`ethtool` 命令包含在 ENA 驱动程序中，并公开了其他与网络相关的指标，这些指标可作为自定义指标发布到 CloudWatch。 Amazon 弹性网络适配器（ENA）为集群置放群组中的实例提供更大的网络吞吐量，实现进一步优化。 Elastic Fabric Adapter（EFA）是 Amazon EC2 实例的网络接口，使您能够在 AWS 上运行要求大规模高级别节点间通信的应用程序。 Amazon EBS 优化实例使用经过优化的配置堆栈，可以针对增加的 Amazon EBS I/O 提供额外的专用容量。

资源

相关文档：

- 应用程序负载均衡器
- Linux EC2 上的增强联网功能
- Windows EC2 上的增强联网功能
- EC2 置放群组
- 在 Linux 实例上启用弹性网络适配器（ENA）增强联网功能
- Network Load Balancer
- AWS 联网产品
- Transitioning to Latency-Based Routing in Amazon Route 53

- VPC 端点
- VPC 流日志

相关视频：

- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2018 – Optimizing Network Performance for Amazon EC2 Instances
- AWS Global Accelerator

相关示例：

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 讲习会
- Observing and diagnosing your network
- Finding and addressing network misconfigurations on AWS

PERF04-BP03 为工作负载选择合适的专用连接或 VPN

当需要混合连接来连接本地资源和云资源时，请预置足够的带宽以满足您的性能要求。估算混合工作负载的带宽和延迟要求。这些数字将推动您的规模需求。

常见反模式：

- 仅根据网络加密要求评估 VPN 解决方案。
- 不评估备用或冗余连接选项。
- 没有确定全部工作负载要求（加密、协议、带宽和流量需求）。

建立此最佳实践的好处：选择和配置适当的连接解决方案将会提高工作负载的可靠性，并最大限度地提高性能。通过确定工作负载要求、提前规划和评估混合解决方案，您可以最大限度地减少成本高昂的物理网络变更和运营开销，同时加快实现价值的速度。

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

根据带宽要求开发混合网络架构。可使用 AWS Direct Connect 将本地网络与 AWS 私密地连接。这适用于需要高带宽、低延迟，同时实现一致性能的情况。VPN 连接通过互联网建立安全连接。在以下情况下可使用 VPN 连接：只需要临时连接、需要考虑成本因素，或者在使用 AWS Direct Connect 的情况下等待建立弹性物理网络连接时作为应急措施。

如果您的带宽要求很高，则可以考虑使用多种 AWS Direct Connect 或 VPN 服务。可以在服务之间对流量进行负载平衡，但由于延迟和带宽差异，我们不建议在 AWS Direct Connect 和 VPN 之间进行负载平衡。

实施步骤

- 估计现有应用程序的带宽和延迟要求。
  - 对于迁移到 AWS 的现有工作负载，利用来自内部网络监控系统的数据。
  - 对于新工作负载或您没有监控数据的现有工作负载，请咨询产品所有者，以确定足够的性能指标并提供良好的用户体验。
- 选择专用连接或 VPN 作为连接选项。根据所有工作负载要求（加密、带宽和流量需求），您可以选择 AWS Direct Connect 或 AWS VPN（或两者）。下图可协助您选择适当的连接类型。
  - AWS Direct Connect 使用专用连接或托管连接，提供指向 AWS 环境的专用连接，速度从 50 Mbps 到 100 Gbps 不等。这样一来，延迟得到管理和控制，并且拥有预置带宽，让您的工作负载能够高效地连接到其他环境。使用 AWS Direct Connect 合作伙伴，您可以从多个环境获得端到端连接，从而提供具有一致性能的扩展网络。AWS 使用原生 100 Gbps、链接聚合组（LAG）或 BGP 同等成本多路径（ECMP）提供扩展 Direct Connect 连接带宽。
  - AWS Site-to-Site VPN 提供支持互联网协议安全（IPsec）的托管服务。创建 VPN 连接时，每个 VPN 连接包括两条隧道以实现高可用性。
- 按照 AWS 文档选择合适的连接选项：
  - 如果决定使用 AWS Direct Connect，请为连接选择合适的带宽。
  - 如果要在多个位置使用 AWS Site-to-Site VPN 连接到 AWS 区域，则使用加速 Site-to-Site VPN 连接，以便有机会提高网络性能。
  - 如果网络设计包含通过 AWS Direct Connect 进行 IPsec VPN 连接，则考虑使用私有 IP VPN 来提高安全性并实现分段。AWS私有 IP Site-to-Site VPN 部署在中转虚拟接口（VIF）上。
  - AWS Direct Connect SiteLink 通过绕过 AWS 区域 在 AWS Direct Connect 位置之间以最快路径发送数据，从而在全球的数据中心之间创建低延迟和冗余连接。

- 在部署到生产环境之前，验证您的连接设置。执行安全和性能测试，确保其满足您的带宽、可靠性、延迟和合规性要求。
- 定期监控您的连接性能和使用情况，并在需要时进行优化。

确定性性能流程图

资源

相关文档：

- AWS 联网产品
- AWS Transit Gateway
- VPC 端点
- 构建可扩展的安全多 VPC AWS 网络基础设施
- Client VPN

相关视频：

- AWS re:Invent 2023 – Building hybrid network connectivity with AWS
- AWS re:Invent 2023 – Secure remote connectivity to AWS
- AWS re:Invent 2022 – Optimizing performance with Amazon CloudFront
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2020 – AWS Transit Gateway Connect

相关示例：

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 讲习会

PERF04-BP04 使用负载平衡在多个资源之间分配流量

跨多个资源或服务分配流量，以便让工作负载能够利用云提供的弹性。您也可以使用负载均衡机制来分流加密终端，以便提高性能和可靠性，并有效管理和路由流量。

常见反模式：

- 在选择负载均衡器类型时不考虑工作负载要求。
- 不利用负载均衡器功能进行性能优化。
- 工作负载直接暴露给互联网，而不使用负载均衡器。
- 通过现有负载均衡器来路由所有互联网流量。
- 使用通用 TCP 负载均衡，并让每个计算节点处理 SSL 加密。

建立此最佳实践的好处：负载均衡器可在单个可用区内或多个可用区之间处理应用程序不断变化的流量负载，并实现高可用性、自动扩展和更高的工作负载利用率。

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

负载均衡器充当工作负载的接入点，从这里将流量分发到后端目标（例如计算实例或容器）来提高利用率。

优化架构的第一步是选择适合的负载均衡器类型。首先列出工作负载特性，例如协议（如 TCP、HTTP、TLS 或 WebSocket）、目标类型（如实例、容器或无服务器）、应用程序要求（如长时间运行的连接、用户身份验证或粘性）和置放（如区域、Local Zone、Outpost 或分区隔离）。

AWS 为应用程序提供多种模型来使用负载均衡。应用程序负载均衡器最适合 HTTP 和 HTTPS 流量的负载均衡，面向交付包括微服务和容器在内的现代应用程序架构，提供高级请求路由功能。

若要对需要极高性能的 TCP 流量进行负载均衡，网络负载均衡器是最佳选择。网络负载均衡器每秒能够处理数百万请求，同时能保持超低延迟，还针对处理突发和不稳定的流量模式进行了优化。

弹性负载均衡提供集成的证书管理和 SSL/TLS 解密，使您可以灵活地集中管理负载均衡器的 SSL 设置，并从工作负载中分流占用大量 CPU 的工作。

选择适合的负载均衡器之后，您可以开始利用其功能来减少后端为提供流量所付出的工作量。

例如，您可以使用应用程序负载均衡器（ALB）和网络负载均衡器（NLB）执行 SSL/TLS 加密分流，借此机会避免目标完成 CPU 密集型 TLS 握手，同时还可以改进证书管理。

当您在负载均衡器中配置 SSL/TLS 分流时，它负责加密进出客户端的流量，同时将未加密的流量传输到您的后端，释放后端资源和缩短客户端的响应时间。

应用程序负载均衡器还可以提供 HTTP/2 流量，无需在您的目标上支持它。因为 HTTP/2 可以更高效地使用 TCP 连接，所以这个简单的决定可以缩短应用程序的响应时间。

在定义架构时应考虑工作负载延迟要求。例如，如果您有延迟敏感型应用程序，则可以决定使用提供极低延迟的网络负载均衡器。您也可以决定通过在 AWS Local Zones 甚至 AWS Outposts 中利用应用程序负载均衡器，让工作负载更接近客户。

延迟敏感型工作负载的另一个考虑因素是跨可用区负载均衡。借助跨可用区负载均衡，每个负载均衡器节点在所有允许的可用区中的已注册目标之间分配流量。

使用与负载均衡器集成的自动扩缩功能。高效性能系统的其中一个关键方面与合理调整后端资源的规模有关。为此，您可以为后端目标资源使用负载均衡器集成。使用负载均衡器与自动扩缩组的集成，根据需要在负载均衡器中添加或删除目标，以应对传入流量。对于容器化工作负载，负载均衡器也可与 Amazon ECS 和 Amazon EKS 集成。

- Amazon ECS – 服务负载均衡器
- Amazon EKS 上的应用程序负载均衡
- Amazon EKS 上的网络负载均衡器

实施步骤

- 定义您的负载均衡要求，包括流量、可用性和应用程序可扩展性。
- 为您的应用程序选择正确的负载均衡器类型。
  - 为 HTTP/HTTPS 工作负载使用应用程序负载均衡器。
  - 为在 TCP 或 UDP 上运行的非 HTTP 工作负载使用网络负载均衡器。
  - 若想利用这两种产品的功能，请将两者结合使用（ALB 作为 NLB 的目标）。例如，若想将 NLB 的静态 IP 与 ALB 基于 HTTP 标头的路由结合使用，或者想将 HTTP 工作负载向 AWS PrivateLink 公开，就可以这样做。
  - 有关负载均衡器的完整比较，请参阅 ELB 产品比较。
- 如果可能，请使用 SSL/TLS 分流。
  - 配置 HTTPS/TLS 侦听器，同时使用集成了 AWS Certificate Manager 的应用程序负载均衡器和网络负载均衡器。
  - 请注意，出于合规性原因，有些工作负载可能需要端到端加密。在这种情况下，必须允许在目标上启用加密。
  - 有关安全最佳实践，请参阅 SEC09-BP02 执行传输中加密。
- 选择适合的路由算法（仅限 ALB）。

  - 路由算法会影响后端目标的使用情况，从而决定它们对性能的影响。例如，ALB 提供了两个路由算法选项：
  - 最少未完成请求：用于在应用程序的请求复杂程度不同或目标处理能力不同的情况下，实现更好的后端目标负载分布。
  - 轮询：当请求和目标类似，或需要在目标之间平均分配请求时使用。
- 考虑跨可用区或分区隔离。
  - 使用关闭的跨可用区（分区隔离）来改善延迟和分区故障域。在 NLB 中，该选项默认处于关闭状态；在 ALB 中，该选项可按目标组关闭。
  - 使用开启的跨可用区来提高可用性和灵活性。在 ALB 中，该选项默认处于打开状态；在 NLB 中，该选项可按目标组开启。
- 为 HTTP 工作负载开启 HTTP 保持活动（仅限 ALB）。借助此功能，在保持活动超时到期之前，负载均衡器可以重复使用后端连接，从而改进 HTTP 请求和缩短响应时间，还可以降低后端目标的资源利用率。有关如何为 Apache 和 Nginx 执行此操作的详细信息，请参阅使用 Apache 或 NGINX 作为 ELB 后端服务器的最佳设置是什么？
- 为您的负载均衡器开启监控。
  - 打开应用程序负载均衡器和网络负载均衡器的访问日志。
  - 主要考虑 ALB 的 `request_processing_time`、`request_processing_time` 和 `response_processing_time`。
  - 主要考虑 NLB 的 `connection_time` 和 `tls_handshake_time`。
  - 准备好在需要时查询日志。可以使用 Amazon Athena 来查询 ALB 日志和 NLB 日志。
  - 为性能相关指标（例如 ALB 的 `TargetResponseTime`）创建警报。

资源

相关文档：

- ELB 产品比较
- AWS 全球基础设施
- Improving Performance and Reducing Cost Using Availability Zone Affinity
- Step by step for Log Analysis with Amazon Athena
- 查询应用程序负载均衡器日志
- Monitor your Application Load Balancers
- Monitor your Network Load Balancer

- 使用弹性负载均衡跨自动扩缩组中的实例分配流量

相关视频：

- AWS re:Invent 2023: What can networking do for your application?
- AWS re:Inforce 2022: How to use Elastic Load Balancing to enhance your security posture at scale
- AWS re:Invent 2018: Elastic Load Balancing: Deep Dive and Best Practices
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads

相关示例：

- Gateway Load Balancer
- 使用 Amazon Athena 进行日志分析的 CDK 和 AWS CloudFormation 示例

PERF04-BP05 选择网络协议以提高性能

根据对工作负载性能的影响，做出有关系统与网络之间的通信协议的决策。

延迟和带宽之间的关系可以实现高吞吐量。如果文件传输使用传输控制协议（TCP），则延迟越高，整体吞吐量很可能越低。有一些方法可以使用 TCP 调整和优化的传输协议来解决此问题，但一种解决方案是使用用户数据报协议（UDP）。

常见反模式：

- 无论有怎样的性能要求，您都可以为所有工作负载使用 TCP。

建立此最佳实践的好处：确认已为用户和工作负载组件之间的通信使用适当的协议，有助于改善应用程序的整体用户体验。例如，无连接 UDP 虽然允许较高速度，但不提供重新传输或高可靠性。TCP 虽然是一个功能全面的协议，但它在处理这些数据包时需要较高的开销。

在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

如果您能够为应用程序选择不同的协议，并且具有该领域的专业知识，请使用不同的协议来优化您的应用程序和最终用户体验。请注意，这种方法难度很大，只有在先用其他方法优化了应用程序后才能尝试。

提高工作负载性能的主要考虑因素是了解延迟和吞吐量要求，然后选择可优化性能的网络协议。

何时考虑使用 TCP

TCP 提供可靠的数据传输，并可用于工作负载组件之间的通信，在这种情况下，可靠性和有保证的数据传输很重要。许多基于 Web 的应用程序依赖基于 TCP 的协议（例如 HTTP 和 HTTPS）来打开 TCP 套接字，以便在应用程序组件之间进行通信。电子邮件和文件数据传输也是使用 TCP 的常见应用，因为它是应用程序组件之间简单可靠的传输机制。在 TCP 之上使用 TLS 会增加一些通信开销，进而会导致延迟增加和吞吐量降低，但它也有安全方面的优势。该开销主要来自握手过程（需要多次往返才能完成）的额外开销。握手完成后，加密和解密数据的开销相对较小。

何时考虑使用 UDP

UDP 是一种面向无连接的协议，因此适用于需要快速、高效传输的应用，例如日志、监控和 VoIP 数据。此外，如果您的工作负载组件要响应来自大量客户端的小型查询，以确保工作负载实现最佳性能，则可考虑使用 UDP。数据报传输层安全性（DTLS）是传输层安全性协议（TLS）的 UDP 等效项。在 UDP 上使用 DTLS 时，因为简化了握手过程，所以开销来自加密和解密数据。因为 DTLS 包括额外的字段，用于指明安全性参数和检测篡改，所以它也会给 UDP 数据包增加少量的开销。

何时考虑使用 SRD

可扩展的可靠数据报（SRD）是一种针对高吞吐量工作负载而优化的网络传输协议，因为它能够跨多条路径对流量进行负载均衡，并能在发生丢包或链路故障时快速恢复。因此，SRD 非常适合在计算节点之间需要高吞吐量、低延迟通信的高性能计算（HPC）工作负载。这可能包括并行处理任务（例如，涉及在节点间进行大量数据传输的模拟、建模和数据分析）。

实施步骤

- 使用 AWS Global Accelerator 和 AWS Transfer Family 服务提高在线文件传输应用程序的吞吐量。AWS Global Accelerator 服务帮助您在客户端设备与 AWS 上的工作负载之间实现更低延迟。借助 AWS Transfer Family，您可以使用基于 TCP 的协议 [安全外壳文件传输协议（SFTP）和基于 SSL 的文件传输协议（FTPS）] 安全地扩展和管理发送到 AWS 存储服务的文件传输。
- 使用网络延迟来确定 TCP 是否适合工作负载组件之间的通信。如果客户端应用程序和服务器之间的网络延迟很高，则 TCP 三次握手需要一些时间，因而会影响应用程序的响应能力。可以使用第一个字节的时间（TTFB）和往返时间（RTT）等指标来衡量网络延迟。如果工作负载向用户提供动态内容，则考虑使用 Amazon CloudFront，因为该服务会为动态内容建立到每个源的持久连接，以便减少连接设置时间，避免减慢每个客户端请求的速度。
- 由于在 TCP 或 UDP 上使用 TLS 会影响加密和解密，从而导致工作负载的延迟增加和吞吐量降低。对于此类工作负载，请考虑使用弹性负载均衡上的 SSL/TLS 分流，使负载均衡器能够处理 SSL/TLS

加密和解密过程，而不是让后端实例来处理，从而提高工作负载性能。这可以帮助降低后端实例上的 CPU 利用率，进而可以提高性能并增加容量。

- 使用网络负载均衡器（NLB）来部署依赖 UDP 协议的服务（例如，身份验证和授权、日志记录、DNS、IoT 和串流媒体），以便提高工作负载的性能和可靠性。NLB 在多个目标之间分配传入的 UDP 流量，使您可以横向扩展工作负载、提高容量和减少单个目标的开销。
- 对于高性能计算（HPC）工作负载，可以考虑使用弹性网络适配器（ENA）Express 功能。该功能使用 SRD 协议，通过为 EC2 实例之间的网络流量提供更高的单流带宽（25 Gbps）和更低的尾部延迟（99.9%）来提高网络性能。
- 使用应用程序负载均衡器（ALB）对工作负载组件之间或 gRPC 客户端与服务之间的 gRPC（远程过程调用）流量进行路由和负载均衡。gRPC 使用基于 TCP 的 HTTP/2 协议进行传输，也可带来性能优势，例如更少的网络占用、压缩、高效的二进制序列化、支持众多语言以及双向流式传输。

资源

相关文档：

- How to route UDP traffic into Kubernetes
- 应用程序负载均衡器
- Linux EC2 上的增强联网功能
- Windows EC2 上的增强联网功能
- EC2 置放群组
- 在 Linux 实例上启用弹性网络适配器（ENA）增强联网功能
- Network Load Balancer
- AWS 联网产品
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC 端点

相关视频：

- AWS re:Invent 2022 – Scaling network performance on next-gen Amazon Elastic Compute Cloud instances
- AWS re:Invent 2022 – Application networking foundations

相关示例：

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 讲习会

PERF04-BP06 根据网络要求选择工作负载的位置

评估资源置放选项，以便减少网络延迟和提高吞吐量，通过缩短页面加载和数据传输时间来提供最佳的用户体验。

常见反模式：

- 将所有工作负载资源整合到一个地理位置中。
- 选择的是离自己位置最近的区域，而不是离工作负载最终用户最近的区域。

建立此最佳实践的好处：用户与应用程序之间的延迟会极大地影响用户体验。通过使用适当的 AWS 区域和 AWS 专用全球网络，您可以减少延迟，为远程用户提供更好的体验。

在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 EC2 实例等资源被放置在 AWS 区域、AWS Local Zones、AWS Outposts 或 AWS Wavelength 区域内的可用区中。选择此位置会影响给定用户位置的网络延迟和吞吐量。Amazon CloudFront 和 AWS Global Accelerator 等边缘服务也可用于在边缘站点缓存内容或为用户提供通过 AWS 全球网络到达工作负载的最佳路径，从而提高网络性能。

Amazon EC2 为联网提供置放群组。置放群组是实例的逻辑分组，可以减少延迟。使用具有支持的实例类型和弹性网络适配器（ENA）的置放群组，可使工作负载参与低延迟、低抖动的 25Gbps 网络。建议将置放群组用于可受益于低网络延迟和/或高网络吞吐量的工作负载。

延迟敏感型服务使用 AWS 全球网络在边缘站点交付，例如 Amazon CloudFront。这些边缘站点通常提供内容分发网络（CDN）和域名系统（DNS）等服务。通过在边缘交付这些服务，工作负载可以低延迟响应内容或 DNS 解析请求。这些服务还提供地理定位服务，例如内容地理定位（基于最终用户位置提供不同内容），或基于延迟的路由（将最终用户引导至最近的区域以实现最小延迟）。

可使用边缘服务来减少延迟并启用内容缓存。为 DNS 和 HTTP/HTTPS 正确配置缓存控制，以便通过这些方式获得最大优势。

实施步骤

- 捕获有关传入和传出网络接口的 IP 流量的信息。

  - 使用 VPC 流日志记录 IP 流量
  - 如何将客户端 IP 地址保留在 AWS Global Accelerator 中
- 分析工作负载中的网络访问模式，以便确定用户如何使用应用程序。
  - 使用 Amazon CloudWatch 和 AWS CloudTrail 等监控工具收集有关网络活动的数据。
  - 分析数据以确定网络访问模式。
- 请根据以下关键元素，为您的工作负载部署选择区域：
  - 数据所在位置：对于数据密集型应用程序（如大数据和机器学习），应用程序代码的运行应尽量接近数据。
  - 用户所在位置：对于面向用户的应用程序，选择接近您工作负载用户的一个或多个区域。
  - 其他约束：考虑成本和合规性等约束因素，如 What to Consider when Selecting a Region for your Workloads 中所述。
- 使用 AWS Local Zones 运行视频渲染等工作负载。Local Zones 使计算和存储资源更接近终端用户，从而使您受益。
- 将 AWS Outposts 用于需要保留在本地的工作负载，在此您希望该工作负载与 AWS 中的其他工作负载一起无缝运行。
- 高分辨率实时视频流、高保真度音频和增强现实或虚拟现实（AR/VR）等应用要求 5G 设备具有超低延迟。对于此类应用程序，请考虑使用 AWS Wavelength。AWS Wavelength 在 5G 网络内嵌入 AWS 计算和存储服务，为开发、部署和扩展超低延迟应用提供移动边缘计算基础设施。
- 对常用资产使用本地缓存或 AWS 缓存解决方案，以提高性能，减少数据移动并减小对环境的影响。

服务	何时使用
Amazon CloudFront	用于缓存静态内容（如图像、脚本和视频）以及动态内容（如 API 响应或 Web 应用程序）。
Amazon ElastiCache	用于缓存 Web 应用程序的内容。
DynamoDB Accelerator	用于将内存中加速添加到 DynamoDB 表。

- 使用有助于您在更接近工作负载用户的位置运行代码的服务，例如：

服务	何时使用
Lambda@Edge	用于执行计算密集型操作，当对象不在缓存中时启动这些操作。
Amazon CloudFront Functions	用于处理简单应用场景，如 HTTP(S) 请求或响应操作，这些操作可由短期运行的函数启动。
AWS IoT Greengrass	用于为互联设备运行本地计算、消息收发和数据缓存。

- 有些应用程序需要固定入口点，或通过减少第一个字节延迟和抖动以及提高吞吐量来提高性能。在边缘站点提供静态任播 IP 地址和 TCP 终止的联网服务可以让这些应用程序受益。AWS Global Accelerator 可以将应用程序的性能提高多达 60%，并为多区域架构提供快速失效转移。AWS Global Accelerator 提供静态任播 IP 地址，可作为一个或多个 AWS 区域 中托管的应用程序的固定入口点。这些 IP 地址使流量在尽可能靠近用户的位置进入 AWS 全球网络。AWS Global Accelerator 在客户端和最靠近客户端的 AWS 边缘站点之间建立 TCP 连接，从而缩短初始连接设置时间。检查 AWS Global Accelerator 的使用情况，以提高 TCP/UDP 工作负载的性能，并为多区域架构提供快速失效转移。

资源

相关最佳实践：

- COST07-BP02 根据成本选择区域
- COST08-BP03 实施服务以便降低数据传输成本
- REL10-BP01 将工作负载部署到多个位置
- REL10-BP02 为多位置部署选择合适的位置
- SUS01-BP01 根据业务要求和可持续性目标选择区域
- SUS02-BP04 根据其联网要求优化工作负载的地理位置
- SUS04-BP07 最大限度地减少跨网络的数据移动

相关文档：

- AWS 全球基础设施

- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- 置放群组
- AWS Local Zones
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- AWS Site-to-Site VPN
- Amazon Route 53

相关视频：

- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2023 - A migration strategy for edge and on-premises workloads
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

相关示例：

- AWS Global Accelerator Custom Routing 讲习会
- Handling Rewrites and Redirects using Edge Functions

PERF04-BP07 根据指标优化网络配置

使用收集和分析的数据做出有关优化网络配置的明智决策。

常见反模式：

- 认为所有性能相关的问题都与应用程序有关。
- 只从距离已部署工作负载很近的位置测试网络性能。
- 为所有网络服务使用默认配置。
- 过度预置网络资源来提供充足的容量。

建立此最佳实践的好处：收集 AWS 网络的必要指标并实施网络监控工具，使您可以了解网络性能和优化网络配置。

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

监控进出 VPC、子网或网络接口的流量，这对于了解如何利用 AWS 网络资源以及如何优化网络配置至关重要。通过使用以下 AWS 网络工具，您可以进一步检查有关流量使用、网络访问和日志的信息。

实施步骤

- 确定要收集的关键性能指标，例如延迟或丢包。AWS 提供了多种工具，可以协助您收集这些指标。通过使用以下工具，您可以进一步检查有关流量使用、网络访问和日志的信息：

AWS 工具	使用情形
Amazon VPC IP 地址管理器	使用 IPAM 来规划、跟踪和监控 AWS 与本地工作负载的 IP 地址。这是优化 IP 地址使用和分配的最佳实践。
VPC 流日志	使用 VPC 流日志来捕获有关进出 VPC 中网络接口的流量的详细信息。借助 VPC 流日志，您可以诊断过于严格或过于宽松的安全组规则，并确定进出网络接口的流量的方向。
AWS Transit Gateway 流日志	使用 AWS Transit Gateway 流日志捕获有关进出中转网关的 IP 流量的信息。
DNS 查询日志记录	有关 Route 53 收到的公有或私有 DNS 查询的日志信息。借助 DNS 日志，您可以了解请求的

AWS 工具	使用情形
	域或子域，或了解响应 DNS 查询的 Route 53 边缘站点，从而优化 DNS 配置。
Reachability Analyzer	Reachability Analyzer 有助于分析并调试网络可达性。作为一种配置分析工具，Reachability Analyzer 能够在 VPC 中的源资源和目标资源之间执行连接测试。此工具可帮助确认网络配置是否符合预期连接。
网络访问分析器	您可以使用网络访问分析器来了解对资源的网络访问。您可以使用网络访问分析器来指定网络访问需求，然后确定不能满足指定要求的潜在网络路径。通过优化相应的网络配置，您可以了解和验证网络的状态，并证明 AWS 中的网络满足您的合规性要求。
Amazon CloudWatch	使用 Amazon CloudWatch 并启用适当的网络选项指标。确保为工作负载选择适合的网络指标。例如，您可以为 VPC 网络地址使用、VPC NAT Gateway、AWS Transit Gateway、VPN 隧道、AWS Network Firewall、弹性负载均衡和 AWS Direct Connect 启用指标。为了观测和了解网络状态和使用情况，以及便于您根据观测结果优化网络配置，持续监控指标是一种好方法。
AWS Network Manager	为了实现运营和规划目的，您可以使用 AWS Network Manager 监控 AWS 全球网络的实时和历史性能。Network Manager 提供 AWS 区域 和可用区之间以及每个可用区内的聚合网络延迟，让您能够更好地了解应用程序性能与 AWS 底层网络性能的关系。
Amazon CloudWatch RUM	使用 Amazon CloudWatch RUM 收集指标，以便为您提供洞察，协助您识别、理解和改善用户体验。

- 使用 VPC 和 AWS Transit Gateway 流日志识别主要贡献者和应用程序流量模式。
- 评测和优化您当前的网络架构，包括 VPC、子网和路由。例如，您可以评估不同的 VPC 对等互连或 AWS Transit Gateway 如何让您能够改善架构中的联网。
- 评测网络中的路由路径，以验证目的地之间是否始终使用最短路径。网络访问分析器可以帮助实现此目的。

资源

相关文档：

- Public DNS query logging
- 什么是 IPAM？
- What is Reachability Analyzer?
- What is Network Access Analyzer?
- VPC 的 CloudWatch 指标
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format
- Monitoring your global and core networks with Amazon CloudWatch metrics
- Continuously monitor network traffic and resources

相关视频：

- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2020 – Networking best practices and tips with the AWS Well-Architected Framework
- AWS re:Invent 2020 – Monitoring and troubleshooting network traffic

相关示例：

- AWS Networking 讲习会

- AWS Network Monitoring
- Observing and diagnosing your network on AWS
- Finding and addressing network misconfigurations on AWS

流程和文化

问题

- PERF 5. 组织实践和文化如何助力提高工作负载的性能效率？

PERF 5. 组织实践和文化如何助力提高工作负载的性能效率？

在最初构建工作负载时，您可以采用一些原则和实践，协助您更好地运行高效、高性能的云工作负载。要采用能提高云工作负载性能效率的文化，请考虑以下关键原则和实践：

最佳实践

- PERF05-BP01 建立关键性能指标（KPI）来衡量工作负载运行状况和性能
- PERF05-BP02 使用监控解决方案了解性能最为关键的方面
- PERF05-BP03 制定流程来提高工作负载性能
- PERF05-BP04 对工作负载进行负载测试
- PERF05-BP05 使用自动化技术主动修复与性能相关的问题
- PERF05-BP06 让工作负载和服务保持最新状态
- PERF05-BP07 定期检查指标

PERF05-BP01 建立关键性能指标（KPI）来衡量工作负载运行状况和性能

确定用于定量和定性地衡量工作负载性能的 KPI。KPI 有助于您衡量与业务目标相关的工作负载的运行状况和性能。

常见反模式：

- 只监控系统级指标来深入了解工作负载，却不了解这些指标对业务的影响。
- 认为 KPI 已作为标准指标数据发布和共享。
- 没有定义可量化、可衡量的 KPI。
- KPI 与业务目标或策略不符。

建立此最佳实践的好处：确定可反映工作负载运行状况和性能的具体 KPI，有助于调整团队的工作重点，并确定成功的业务成果。与所有部门共享这些指标可让所有人了解并一致认可阈值、期望值和业务影响。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

利用 KPI，业务和工程团队可在衡量目标和策略以及如何将这些因素结合来取得业务成果方面达成共识。例如，网站工作负载可能会将页面加载时间用作总体性能指示。该指标将是用来衡量用户体验的多个数据点之一。除了确定页面加载时间阈值之外，您还应记录未达到理想性能要求时的预期成果或业务风险。较长的页面加载时间会直接影响最终用户的体验，降低他们的用户体验评分，并会导致客户流失。在定义 KPI 阈值时，请结合考虑行业基准和最终用户期望。例如，如果当前行业基准是两秒内加载网页，而您的最终用户希望网页在一秒内加载，那么您在建立 KPI 时应考虑这两个数据点。

您的团队必须使用实时的精细数据和历史数据作为参考来评估工作负载 KPI，并创建控制面板来对 KPI 数据执行指标计算，从而获得运维和利用率方面的洞察。应记录 KPI，包括支持业务目标和策略的阈值，并且应与所监控的指标对应起来。当业务目标、策略或最终用户需求发生变化时，应重新审视 KPI。

实施步骤

- 确定利益相关方：确定并记录关键的业务利益相关方，包括开发和运营团队。
- 定义目标：与这些利益相关方合作，定义并记录工作负载目标。考虑工作负载的关键性能方面（例如吞吐量、响应时间和成本），以及业务目标（例如用户满意度）。
- 查看行业最佳实践：查看行业最佳实践，确定与工作负载目标相一致的相关 KPI。
- 确定指标：确定与工作负载目标一致且有助于衡量绩效和业务目标的指标。根据这些指标建立 KPI。示例指标包括平均响应时间或并发用户数量等衡量指标。
- 定义并记录 KPI：使用行业最佳实践和工作负载目标为工作负载 KPI 设定目标。使用这些信息设置 KPI 阈值的严重性或警报级别。确定并记录未满足 KPI 时带来的风险和影响。
- 实施监控：使用 Amazon CloudWatch 或 AWS Config 等监控工具收集指标并衡量 KPI。
- 直观地传达 KPI：使用 Amazon QuickSight 等控制面板工具来可视化 KPI，并就此与利益相关方沟通。
- 分析和优化：定期审查并分析 KPI，确定需要从哪些方面改进工作负载。与利益相关方协作实施这些改进。
- 重新审视和完善：定期审查指标和 KPI，评测其有效性，尤其是在业务目标或工作负载绩效发生变化时。

资源

相关文档：

- CloudWatch 文档
- AWS Partner 监控、日志记录和性能
- AWS observability tools
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations
- How to track your cost optimization KPIs with the KPI Dashboard
- X-Ray 文档
- Using Amazon CloudWatch dashboards
- QuickSight KPIs

相关视频：

- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health
- AWS re:Invent 2023 - Performance & efficiency at Pinterest: Optimizing the latest instances
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Creating an Effective Metrics Strategy for Your Business | AWS Events

相关示例：

- Creating a dashboard with QuickSight

PERF05-BP02 使用监控解决方案了解性能最为关键的方面

了解并确定在哪些方面提高工作负载性能，会对效率或客户体验产生积极的影响。例如，拥有大量客户交互的网站会因为使用边缘服务在距离客户更近的位置向客户分发内容而受益。

常见反模式：

- 认为标准计算指标（例如，CPU 利用率或内存压力）足够捕获性能问题。
- 只使用由自己选定的监控软件记录的默认指标。
- 只在出现问题时审查指标。

建立此最佳实践的好处：了解关键性能领域可以帮助工作负载负责人监控 KPI 并确定具有高影响力的优先改进。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

设置端到端的跟踪，用于确定流量模式、延迟和关键性能领域。针对速度缓慢的查询或性能欠佳的碎片和分区数据，监控数据访问模式。使用负载测试或监控来确定受约束的工作负载领域。

通过了解架构、流量模式和数据访问模式，提高性能效率，并确定延迟和处理时间。确定随着工作负载增长可能会影响客户体验的潜在瓶颈。在研究了这些方面之后，再看看可以通过部署哪项解决方案来解决这些性能问题。

实施步骤

- 设置端到端的监控，用于收集所有工作负载组件和指标。以下是 AWS 监控解决方案的示例。

服务	使用情形
Amazon CloudWatch 真实用户监控（RUM）	收集真实用户客户端和前端会话的应用程序性能指标。
AWS X-Ray	通过应用程序层跟踪流量，并确定组件间的延迟以及依赖关系。使用 X-Ray 服务地图查看工作负载组件之间的关系和延迟。
Amazon Relational Database Service Performance Insights	查看数据库性能指标并确定性能改进机会。
Amazon RDS 增强监控	查看数据库 OS 性能指标。
Amazon DevOps Guru	检测异常运营模式，以便您可以在运营问题影响客户之前发现它们。

- 执行测试以生成指标，确定流量模式、瓶颈和关键性能领域。以下是一些有关如何执行测试的示例：

  - 设置 CloudWatch Synthetics 金丝雀，使用 Linux cron 作业或 rate 表达式，通过编程方式模拟浏览器端的用户活动，从而生成一段时间内的稳定指标。
  - 使用 AWS 分布式负载测试解决方案生成峰值流量，或者在预期增长速率下测试工作负载。
- 评估指标和遥测数据，确定您的关键性能方面。与团队一起审查这些方面，讨论监控和解决方案以避免瓶颈。
- 试验性能改进，并利用数据来衡量这些更改。例如，使用 CloudWatch Evidently 测试新的改进以及对工作负载的性能影响。

资源

相关文档：

- What's new in AWS Observability at re:Invent 2023
- Amazon Builders' Library
- X-Ray 文档
- Amazon CloudWatch RUM
- Amazon DevOps Guru

相关视频：

- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 years of Amazon operational excellence
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Visual Monitoring of Applications with Amazon CloudWatch Synthetics

相关示例：

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

- 适用于 Python 的 X-Ray 开发工具包
- AWS 上的分布式负载测试

PERF05-BP03 制定流程来提高工作负载性能

制定相应流程，对推出的新服务、设计模式、资源类型和配置进行评估。例如，对新实例产品运行现有性能测试，确定其是否有潜力改进工作负载。

常见反模式：

- 认为当前架构是静态的，将来不会更新。
- 不断对架构进行更改，却不提供任何指标方面的依据。

建立此最佳实践的好处：通过制定架构更改流程，您可以使用所收集的数据来影响以后的工作负载设计。

在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

工作负载的性能会面临一些关键约束。记录这些约束，以便了解哪些创新可以改进工作负载的性能。在知道有新的服务或技术推出时，借助这些信息来确定消除约束或瓶颈的方法。

确定针对工作负载的关键性能约束。记录工作负载的性能约束，以便了解哪类创新可以提高工作负载的性能。

实施步骤

- 确定 KPI：如 PERF05-BP01 建立关键性能指标（KPI）来衡量工作负载运行状况和性能中所述，确定工作负载性能 KPI，为工作负载建立基准。
- 实施监控：使用 AWS 可观测性工具收集绩效指标并衡量 KPI。
- 执行分析：执行深入分析，确定工作负载中性能欠佳的方面（如配置和应用程序代码），如 PERF05-BP02 使用监控解决方案了解性能最为关键的方面中所述。使用分析和性能工具来确定性能改进策略。
- 验证改进：使用沙盒环境或预生产环境来验证策略的有效性。
- 实施变更：在生产环境中实施变更并持续监控工作负载的性能。记录改进内容，并将变更内容传达给利益相关方。

- 重新审视和完善：定期审查绩效改进流程，确定需要改进的领域。

资源

相关文档：

- AWS 博客
- AWS 的新功能
- AWS Skill Builder

相关视频：

- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance

相关示例：

- AWS GitHub

PERF05-BP04 对工作负载进行负载测试

对工作负载进行负载测试，从而验证工作负载能否处理生产负载，并找出任何性能瓶颈。

常见反模式：

- 对工作负载的各个部分进行单独负载测试，而不是测试整个工作负载。
- 在与生产环境不同的基础设施上进行负载测试。
- 只对预期负载而不对其他负载进行负载测试，来预测未来可能会出现问题的方面。
- 没有查阅 Amazon EC2 Testing Policy 并提交“模拟事件提交表”，就执行负载测试。这会导致您的测试无法运行，因为它看起来像是拒绝服务事件。

建立此最佳实践的好处：通过负载测试来衡量性能，可说明随着负载的增加，您将在哪些方面受到影响。这样您便可以在变更影响自己的工作负载之前，对所需进行的变更进行预测。

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

云端负载测试是在预期用户负载的实际条件下衡量云工作负载性能的过程。这一过程包括：预置类似于生产的云环境，使用负载测试工具生成负载，分析各个指标来评测工作负载处理实际负载的能力。必须使用生产数据的合成或净化版本（删除敏感信息或身份识别信息）运行负载测试。作为交付管道的一部分，自动执行负载测试，并将结果与预定义的 KPI 和阈值进行比较。这一过程有利于您持续实现所需的性能。

实施步骤

- 定义测试目标：确定待评估工作负载的性能方面，例如吞吐量和响应时间。
- 选择测试工具：选择并配置适合工作负载的负载测试工具。
- 设置环境：根据生产环境设置测试环境。您可以使用 AWS 服务来运行生产规模的环境，进而测试架构。
- 实施监控：使用 Amazon CloudWatch 等监控工具，收集架构中各个资源的指标。您还可以收集和发布自定义指标。
- 定义场景：定义负载测试场景和参数（如测试持续时间和用户数量）。
- 执行负载测试：大规模执行负载测试场景。利用 AWS Cloud 来测试工作负载，发现工作负载的哪些部分无法扩展或者是否以非线性方式扩展。例如，您可以使用竞价型实例以很低的成本生成负载，并在投入生产前发现瓶颈。
- 分析测试结果：对结果进行分析，确定性能瓶颈和需要改进的地方。
- 记录和分享调查发现：记录并报告调查发现和建议。与利益相关方共享此信息，协助他们就性能优化策略做出明智的决策。
- 持续迭代：应定期执行负载测试，尤其是在系统更改更新之后。

资源

相关文档：

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- AWS 上的分布式负载测试

相关视频：

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing
- AWS re:Invent 2022 - Scaling on AWS for your first 10 million users
- Solving with AWS Solutions: Distributed Load Testing

- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

相关示例：

- AWS 上的分布式负载测试

PERF05-BP05 使用自动化技术主动修复与性能相关的问题

使用关键性能指标（KPI）并结合监控和警报系统，主动解决与性能相关的问题。

常见反模式：

- 只允许运营人员对工作负载进行运营更改。
- 通过设置筛选条件将所有没有主动修复行为的警报发送给运营团队。

建立此最佳实践的好处：主动修复警报行为使支持人员能够集中精力处理那些无法自动完成的工作。这样一来，操作人员只需集中精力处理关键警报，从而避免因处理所有警报而变得应接不暇。

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

使用警报触发自动操作，以便在可能的情况下修复问题。如果无法实现自动响应，则将警报上报给能够响应的人员。例如，您的系统在关键性能指标（KPI）超出特定阈值时，能够预测预期 KPI 值并发出警报；或者您的工具在 KPI 超出预期值时，能够自动停止或回滚部署。

实施相应流程，让您在工作负载运行期间了解其性能。构建监控控制面板并确定预期性能基准，以确定工作负载的性能是否达到最佳。

实施步骤

- 确定修复工作流程：识别并了解可以自动修复的性能问题。使用 Amazon CloudWatch 或 AWS X-Ray 等 AWS 监控解决方案，帮助您更好地了解问题的根本原因。

- 定义自动化流程：创建可用于自动修复问题的分步修复流程。
- 配置启动事件：将事件配置为自动启动修复流程。例如，您可以定义一个触发器，以便在实例达到特定 CPU 利用率阈值时自动重启实例。
- 自动执行修复：使用 AWS 服务和技术自动执行修复流程。例如，AWS Systems Manager Automation 提供了一种安全且可扩展的方法来自动执行修复流程。如果更改未成功解决问题，请务必使用自我修复逻辑来还原更改。
- 测试工作流程：在预生产环境中测试自动修复流程。
- 实施工作流程：在生产环境中实施自动修复。
- 制定行动手册：制定行动手册并记录相关内容，概述修复计划的步骤，包括启动事件、修复逻辑和采取的行动。确保对利益相关方进行培训，协助他们有效应对自动修复事件。
- 审查和完善：定期评测自动修复工作流程的有效性。必要时调整启动事件和修复逻辑。

资源

相关文档：

- CloudWatch 文档
- AWS Partner Network 合作伙伴监控、日志记录和性能
- X-Ray 文档
- Using Alarms and Alarm Actions in CloudWatch
- Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- Automate your Amazon Redshift performance tuning with automatic table optimization

相关视频：

- AWS re:Invent 2023 - Strategies for automated scaling, remediation, and smart self-healing
- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2021 - Intelligently automating cloud operations
- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - Automating patch management and compliance using AWS

- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS
- AWS re:Invent 2021 -{New Launch} Automatically detect and resolve issues with Amazon DevOps Guru
- AWS re:Invent 2023 - Centralize your operations

相关示例：

- CloudWatch Logs Customize Alarms

PERF05-BP06 让工作负载和服务保持最新状态

随时了解新的云服务和功能，积极采用高效的功能，解决出现的问题并提高工作负载的整体性能效率。

常见反模式：

- 认为当前架构是静态的，将来不会更新。
- 没有任何系统或定期安排来评估更新后的软件和软件包是否与工作负载兼容。

建立此最佳实践的好处：通过建立流程来及时了解新服务和产品的最新情况，您可以采用新的特性和功能、解决问题并提高工作负载性能。

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

随着新的服务、设计模式和产品功能的推出，评估可提高性能的方法。通过评估、内部讨论或外部分析来确定哪些方法可以提高工作负载的性能或效率。制定相应流程，评估与工作负载相关的更新、新功能和服务。例如，使用新技术构建概念验证或咨询内部团队。在尝试新想法或新服务时，运行性能测试来衡量这些新想法或新服务对工作负载性能的影响。

实施步骤

- 清点工作负载：清点工作负载软件和架构，确定需要更新的组件。
- 确定更新资源：确定与工作负载组件相关的资讯和更新来源。例如，您可以订阅 AWS 的新功能博客，了解与工作负载组件相匹配的产品。您可以订阅 RSS 源或管理电子邮件订阅。
- 制定更新计划：制定计划来评估工作负载的新服务和新功能。

  - 您可以使用 AWS Systems Manager 清单从 Amazon EC2 实例中收集操作系统（OS）、应用程序和实例元数据，并快速了解哪些实例正在运行软件策略所需的软件和配置，以及哪些实例需要更新。
- 评测新更新：了解如何更新工作负载的组件。利用云中的敏捷性，快速测试新功能如何改善工作负载，从而提高性能效率。
- 采用自动化：采用自动化更新流程，减少部署新功能的工作量，并减少手动过程引起的错误。
  - 您可以使用 CI/CD 自动更新 AMI、容器映像以及其他与云应用程序相关的构件。
  - 您可以使用 AWS Systems Manager 补丁管理器等工具来自动执行系统更新流程，并使用 AWS Systems Manager Maintenance Windows 来安排活动。
- 记录流程：记录评估更新和新服务的流程。为负责人提供所需的时间和空间来研究、测试、试验和验证更新及新服务。回顾记录的业务要求和 KPI，帮助确定会对业务产生积极影响的更新的优先级。

资源

相关文档：

- AWS 博客
- AWS 的新功能
- Implementing up-to-date images with automated EC2 Image Builder pipelines

相关视频：

- AWS re:Inforce 2022 - Automating patch management and compliance using AWS
- All Things Patch: AWS Systems Manager | AWS Events

相关示例：

- Inventory and Patch Management
- One Observability 讲习会

PERF05-BP07 定期检查指标

作为例行维护的一部分或为了应对事件或意外事件，请检查收集到了哪些指标。通过这些检查，找出哪些指标对于解决问题至关重要，以及跟踪哪些其他指标会有助于发现、解决或预防问题。

常见反模式：

- 让指标长时间保持警报状态。
- 创建自动化系统无法操作的警报。

建立此最佳实践的好处：不断检查收集的指标，确认这些指标是否有助于正确地发现问题、解决问题或预防问题。如果让指标长时间保持警报状态，这些指标也会过时。

在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

不断改进指标收集和监控效果。在响应意外事件或事件的过程中，评估哪些指标有助于解决问题、哪些目前没有跟踪的指标会有助于解决问题。通过这种方法，您可以提高收集的指标的质量，从而预防或更快速地解决未来发生的意外事件。

在响应意外事件或事件的过程中，评估哪些指标有助于解决问题、哪些目前没有跟踪的指标会有助于解决问题。这样，您可以提高收集的指标的质量，从而预防或更快速地解决未来发生的意外事件。

实施步骤

- 定义指标：定义为实现工作负载目标而需要监控的关键性能指标，包括响应时间和资源利用率等指标。
- 建立基准：为每个指标设置基准和期望值。基准应提供参考点，用于确定偏差或异常。
- 建立定期机制：建立定期机制（例如每周或每月）来审核关键指标。
- 识别性能问题：在每次审核期间，评测趋势以及与基准值的偏差。找出任何性能瓶颈或异常情况。对于已发现的问题，深入分析根本原因，了解问题背后的主要原因。
- 确定纠正措施：利用分析结果来确定纠正措施。这可能包括调整参数、修复错误和扩展资源。
- 记录调查发现：记录调查发现，包括已确定的问题、根本原因和纠正措施。
- 迭代和改进：持续评测和改进指标审核流程。利用从之前审核中吸取的经验教训，不断改进流程。

资源

相关文档：

- CloudWatch 文档
- 使用 CloudWatch 代理从 Amazon EC2 实例和本地部署服务器中收集指标和日志
- 使用 CloudWatch Metrics Insights 查询您的指标

- AWS Partner Network 合作伙伴监控、日志记录和性能
- X-Ray 文档

相关视频：

- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS

相关示例：

- Creating a dashboard with QuickSight
- CloudWatch Dashboards

成本优化

成本优化支柱包括以最低价格运行系统来实现业务价值的能力。有关具体实施的说明性指导，请参阅《成本优化支柱白皮书》。

最佳实践领域

- 践行云财务管理
- 支出和使用情况意识
- 具有成本效益的资源
- 管理需求和供应资源
- 持续优化

践行云财务管理

问题

- COST 1. 如何实施云财务管理？

COST 1. 如何实施云财务管理？

实施云财务管理有助于组织在 AWS 上优化成本和使用情况并进行扩展，从而实现商业价值和财务成功。

最佳实践

- COST01-BP01 确立成本优化的责任归属模式
- COST01-BP02 在财务和技术人员之间建立合作关系
- COST01-BP03 建立云预算和预测
- COST01-BP04 在组织流程中落实成本意识
- COST01-BP05 报告和通知成本优化
- COST01-BP06 主动监控成本
- COST01-BP07 及时了解新发布的服务
- COST01-BP08 建立对成本敏感的文化
- COST01-BP09 量化通过成本优化实现的业务价值

COST01-BP01 确立成本优化的责任归属模式

创建一个团队（云业务办公室、云卓越中心或 FinOps 团队），负责在整个组织内建立并维护成本意识。成本优化的负责人可以是了解整个组织和云财务的个人或团队（需要来自财务、技术和业务团队的人员）。

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

这是云业务办公室（CBO）或云卓越中心（CCoE）部门或团队的介绍，此部门或团队负责建立并维护一种对云计算成本敏感的文化。此部门可以是一个现有的个人、组织内的一个团队，也可以是一个由整个组织的关键财务、技术和组织利益相关者组成的新团队。

此部门（个人或团队）会排定成本管理和成本优化活动的优先级，并根据需要为这些活动投入一定比率的时间。相对于较大型企业中的全职部门，小型组织的这一部门在此方面花费的时间可能更少。

此部门需要采取多学科方法，并具备项目管理、数据科学、财务分析和软件或基础设施开发的能力。此部门可通过三种不同的责任归属模式来执行成本优化，以提高工作负载的效率：

- 集中式：通过 FinOps 团队、云财务管理（CFM）团队、云业务办公室（CBO）或云卓越中心（CCoE）等指定团队，客户可以设计和实施治理机制，并在全公司范围内推广最佳实践。

- 分散式：由具有影响力的技术团队执行成本优化。
- 混合式：同时利用集中式和分散式团队，两者可以合作执行成本优化。

可以对照成本优化目标（例如工作负载效率指标）来衡量此部门的执行和交付能力。

您必须为此部门获得高管支持，这是取得成功的一个关键因素。支持者负责倡导注重成本效益的云消费理念，为成本优化团队提供升级上报支持，确保按组织确定的优先级开展成本优化活动。否则，相关方面会忽视指导意见，并且不会优先考虑节省成本的机会。支持者与团队共同确保组织有效利用云资源并创造业务价值。

如果您制定了商业、企业入门或企业支持计划，并需要帮助来建立此团队或部门，请通过您的客户团队联系云财务管理（CFM）专家。

实施步骤

- 明确主要成员：组织的所有相关部门都必须关注成本管理，并做出贡献。组织中的常见团队通常包括：财务、应用程序或产品负责人、管理和技术团队（DevOps）。有些是全职工作（财务或技术），有些则是根据需要定期工作。执行 CFM 的个人或团队需要掌握以下技能组合：
  - 软件开发：在构建脚本和自动化的情况下。
  - 基础设施工程：部署脚本，自动化流程，并了解如何预置服务或资源。
  - 运营敏锐性：CFM 的宗旨是通过衡量、监控、修改、规划和扩展对云的有效使用，在云上高效地运行。
- 明确目标和指标：此部门需要以不同方式为组织创造价值。相关目标已确定，并将随着组织的发展而不断完善。常见活动包括：在整个组织内创建并执行关于成本优化的培训计划、制定涉及整个组织的标准，例如监控和报告成本优化，以及设置关于优化的工作负载目标。此部门还需要定期向组织报告其成本优化能力。

  可以定义基于价值或成本的关键绩效指标（KPI）。定义 KPI 时，可以根据效率和预期业务成果计算预期成本。基于价值的 KPI 将成本和使用情况指标与业务价值驱动因素联系起来，有助于合理调整 AWS 支出。获得基于价值的 KPI 的第一步是跨组织协作，选择并商定一组标准 KPI。
- 建立定期沟通机制：该小组（财务、技术和业务团队）应该定期开会，以审查目标和指标。典型的定期沟通机制包括审查组织的状态、当前执行的任何计划以及整体财务和优化指标。随后，更详细地报告关键工作负载。

  在这些定期审查中，可以审查工作负载效率（成本）和业务成果。例如，工作负载的成本增加 20% 可能与客户使用情况的增加保持一致。在这种情况下，这 20% 的成本增加可以理解为一项投资。这些定期沟通通话有助于团队确定价值 KPI，为整个组织带来意义。

资源

相关文档：

- AWS CCOE 博客
- 创建云业务办公室
- CCOE – 云卓越中心

相关视频：

- Vanguard CCOE 成功案例

相关示例：

- 使用云卓越中心（CCoE）实现整个企业转型
- 构建 CCOE 以实现整个企业转型
- 构建 CCOE 时应避免的 7 个陷阱

COST01-BP02 在财务和技术人员之间建立合作关系

在云之旅的所有阶段，都让财务和技术团队参与成本和使用情况的讨论。团队定期开会，讨论组织目标、成本和使用情况的当前状态以及财务和会计实务等主题。

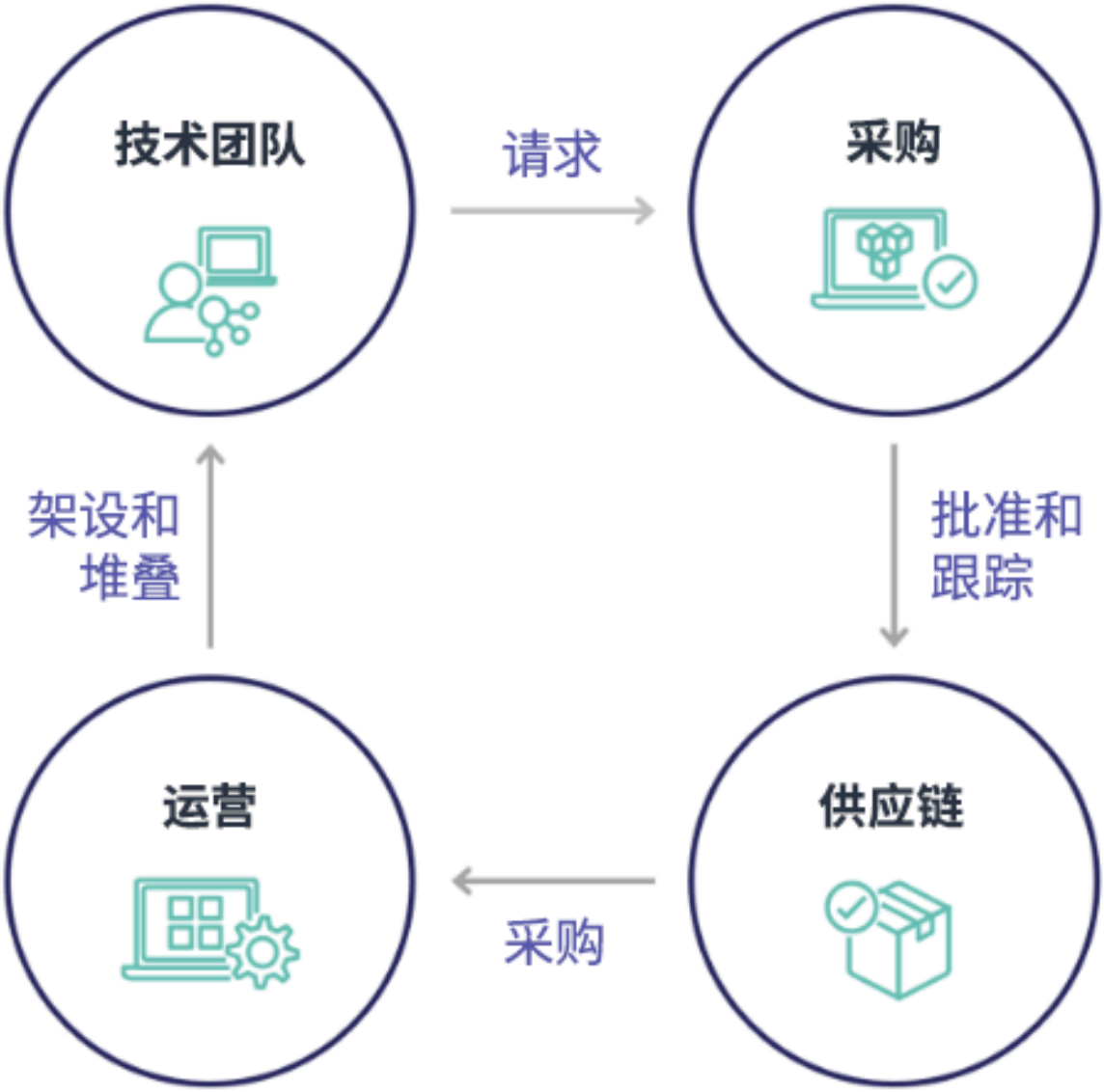
在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

由于缩短了审批、采购和基础设施部署周期，技术团队在云端的创新速度更快。这可能是对财务组织的一种调整，以前，他们习惯于运行耗时的资源密集型流程，以便在数据中心和本地环境中获取和部署资金，并且只在项目批准时进行成本分配。

从财务和采购组织的角度来看，资本预算编制、资本请求、审批、采购和安装物理基础设施这一流程是我们几十年来一直在学习和实现标准化的流程：

- 工程团队或 IT 团队通常是请求者
- 不同的财务团队充当审批者和采购者
- 运营团队架设、堆叠并移交可供使用的基础设施



随着云的采用，基础设施的采购和消费不再受制于一连串的依赖关系。在云模式下，技术和产品团队不再仅仅是构建者，还是产品的运营者和负责人，他们负责过去与财务和运营团队有关的大部分活动，包括采购和部署。

预置云资源真正需要的只是一个账户和一组正确的权限。这也有助于降低 IT 和财务风险，因为团队只需点击几下鼠标或进行几次 API 调用，就可以终止空闲或不必要的云资源。这也使技术团队能够更快创新，因为他们获得了启动实验以及之后拆除实验的敏捷性和能力。虽然从资本预算编制和预测的角度来看，云消费的可变性质可能会影响可预测性，但云为组织提供了降低过度预置成本的能力，以及降低与保守预置不足相关的机会成本的能力。

在关键的财务和技术利益相关方之间建立合作关系，让他们就组织目标达成共识，并开发在云计算的可变支出模型中获得财务成功的机制。组织内的相关团队必须在云之旅的各个阶段参与成本和使用情况讨论，包括：

- 财务领导：首席财务官、财务总监、财务规划师、业务分析师、采购、供应商开发人员和应付账款负责人必须了解消费、采购选项和每月开票流程的云模型。财务部门需要与技术团队合作创建有关 IT 价值的沟通内容并广泛传播，帮助业务团队了解技术支出与业务成果之间的联系。这样，技术支出就不会被视为成本，而会被视为投资。由于云运营（如使用情况的变化速率、即用即付定价模式、分级定价、定价模式以及详细的计费和使用情况信息）与本地运营之间存在根本差异，财务组织必须了解云的使用对业务方面的影响，包括采购流程、激励跟踪、成本分配和财务报表。
- 技术领导：技术领导（包括产品和应用程序拥有者）必须了解财务要求（如预算约束）和业务要求（如服务水平协议），如此才能实施工作负载并实现组织的预期目标。

财务与技术人员的合作可带来以下好处：

- 财务和技术团队几乎可以实时看到成本和使用情况。
- 财务和技术团队建立了标准操作程序来处理云支出差异。
- 在如何使用资本购买承诺折扣（例如预留实例或 AWS 节省计划）以及如何使用云来发展组织方面，财务利益相关方充当战略顾问。
- 将现有的应付账款和采购流程用于云部署。
- 财务和技术团队协作预测未来的 AWS 成本和使用情况，以调整和建立组织预算。
- 通过共通的语言以及对财务概念的一致理解，更好地进行跨组织沟通。

组织中应参与成本和使用情况讨论的其他利益相关方包括：

- 业务部门负责人：业务部门负责人必须了解云业务模式，以便为业务部门和整个公司提供指导。在需要预测增长和工作负载使用情况，以及评测不同购买选项（例如预留实例或节省计划）时，这方面的云知识至关重要。
- 工程团队：在财务和技术团队之间建立合作关系对于建立对成本敏感的文化至关重要，这可以鼓励工程师在云财务管理（CFM）上采取行动。CFM 或财务运营从业者和财务团队的一个常见问题是让工程师了解云上的整个业务，遵循最佳实践，并采取建议的行动。
- 第三方：如果组织使用第三方（例如顾问或工具），请确保他们与您的财务目标一致，并且可以通过参与模式和投资回报（ROI）证明一致性。第三方通常会帮助报告和分析其管理的任何工作负载，并且提供他们设计的任何工作负载的成本分析。

要想实施 CFM 并取得成功，需要财务、技术和业务团队之间彼此协作，并在整个组织内就如何转变云支出进行沟通和评估。将工程团队包括在内，让他们在所有阶段参与这些成本和使用情况的讨论；还要鼓励他们遵循最佳实践并采取相应的商定行动。

实施步骤

- 明确主要成员：确认财务和技术团队的所有相关成员都参与合作。相关财务成员将是与云账单进行交互的人员。他们通常是首席财务官、财务总监、财务规划师、业务分析师和采购员。技术成员通常是产品和应用程序负责人、技术经理和所有在云上执行构建的团队的代表。其他成员可能包括业务部门负责人（例如影响产品使用的营销部门）和第三方（例如顾问），以确保与您的目标和机制保持一致，并协助进行报告。
- 明确讨论主题：明确团队之间的共同主题，或者需要达成共识的主题。从生成成本之时跟踪成本，直到账单已付为止。请注意所涉及的任何成员，以及需要应用的组织流程。了解经过的每个步骤或流程以及相关信息，例如可用的定价模式、分级定价、折扣模型、预算编制和财务要求。

- 建立定期沟通机制：为了让财务和技术人员展开合作，建立定期沟通机制，以促进并保持一致。该小组需要定期开会，审查目标和指标。典型的定期沟通机制包括审查组织的状态、当前执行的任何计划以及整体财务和优化指标。然后，更详细地报告关键工作负载。

资源

相关文档：

- AWS 新闻博客

COST01-BP03 建立云预算和预测

调整现有的组织预算编制和预测流程，使之适应云成本和使用情况的易变特性。流程必须是动态的，可以使用基于趋势或基于业务驱动因素的算法，也可以将两者结合使用。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在传统的本地 IT 设置中，客户在规划偶有变动的固定成本时，经常会面临挑战。一般情况下，客户会购买新的 IT 硬件和服务来满足高峰需求，而这就会带来成本变动挑战。相反，AWS Cloud 采用了另一种方法，使客户只需为在满足他们实际的 IT 和业务需求时使用的资源付费。在云环境中，需求会每月、每天甚至每小时发生波动。

借助云，客户可以提升效率、速度和敏捷性，最终打造高度可变的成本和使用模式。随着工作负载效率的提高，或者新工作负载和功能的部署，成本可能会降低，有时也会提高。当工作负载扩展以便为不断扩大的客户群提供服务时，云的使用情况和成本也会相应上升，这是因为资源可访问性得到了提高。云服务的这种灵活性会延伸到成本和预测，从而带来一定的弹性。

您必须将这些不断变化的业务需求和需求驱动因素紧密结合起来，并尽可能准确地规划。传统的组织预算编制流程需要作出调整来适应这种可变性。

在预测新工作负载的成本时，可以考虑使用成本建模。通过成本建模，可以基本了解预期的云成本，这有助于您执行总拥有成本（TCO）、投资回报率（ROI）和其他财务分析，与利益相关方一起设定目标和期望，并确定成本优化机会。

组织应了解成本定义和可接受的分组。做出的预测的详细程度可能会因组织的结构和内部工作流程而异。选择符合特定要求和组织机构特点的粒度。请务必了解预测是在哪个级别进行的：

- 管理账户或 AWS Organizations 级别：管理账户是用于创建 AWS Organizations 的账户。默认情况下，Organizations 有一个管理账户。

- 关联账户或成员账户：Organizations 中的账户是标准 AWS 账户，其中包含您的 AWS 资源以及可以访问这些资源的身份。
- 环境：环境是运行应用程序版本的 AWS 资源的集合。可以使用多个关联账户或成员账户来创建环境。
- 项目：项目是在固定期限内要完成的既定目标或任务的组合。在预测期间，请务必考虑项目生命周期。
- AWS 服务：组或类别，例如计算或存储服务，可以在其中对 AWS 服务进行分组以进行预测。
- 自定义分组：可以根据组织需求创建自定义组，例如业务部门、成本中心、团队、成本分配标签、成本类别、关联账户或这些元素的任意组合。

确定可能影响使用成本的业务驱动因素，并分别针对每个驱动因素进行预测，以便提前计算出预期使用量。其中一些驱动因素可能与组织中的 IT 和产品团队有关。其他业务驱动因素（如营销活动、促销、地域扩张、合并和收购）则是销售、营销和业务领导者所了解的，因此与他们合作，将所有这些需求驱动因素考虑在内也很重要。

可以根据历史支出，使用 AWS Cost Explorer 对确定的未来时间范围进行基于趋势的预测。AWS Cost Explorer 的预测引擎会根据付费类型（例如预留实例）对历史数据进行细分，并结合使用机器学习和基于规则的模型来分别预测所有付费类型的支出。

建立预测流程并构建模型后，可以使用 AWS Budgets 设置精细的自定义预算，即通过指定时间段、循环或金额（固定或可变），及添加筛选条件（如服务、AWS 区域 和标标签）来实现。预算通常针对一年进行编制，并且保持固定，这将要求所有相关人员严格遵守。相比之下，预测更为灵活，可以在全年重新调整，并提供一年、两年或三年的动态预测。在确定各技术和业务利益相关方的财务预期时，预算编制和预测都起着至关重要的作用。预测和实施要保持准确，还需要直接负责预置成本的利益相关方承担责任，这也能提高他们的整体成本意识。

为了随时了解现有预算的执行情况，可以创建 AWS Budgets 报告并安排好时间表，定期以电子邮件的形式发送给您和您的利益相关方。您还可以根据实际成本（本质上是反应式的）或预测成本（可以留出时间缓解潜在的成本超支情况）创建 AWS Budgets 警报。您的成本或使用情况实际超出一定量，或预计将超出预算金额时，系统会向您发送警报。

使用基于趋势（将历史成本用作输入）的算法或者基于驱动因素（例如新产品发布、区域扩张或用于工作负载的新环境）的算法（适用于动态和可变支出环境），调整现有的预算编制和预测流程，使这些流程更为灵活。使用 Cost Explorer 或任何其他工具确定基于趋势的预测后，使用 AWS 定价计算器 即可估算您的 AWS 应用场景和基于预期使用情况（流量、每秒请求数或所需的 Amazon EC2 实例）的未来成本。

跟踪预测的准确性，因为预算的编制需要基于这些预测计算和估算值。监控集成云成本预测的准确性和有效性。定期针对预测审查实际支出，并根据需要作出调整来提高预测精度。跟踪预测差异，对报告的差异进行根本原因分析，以采取行动并调整预测。

如 COST01-BP02 在财务和技术人员之间建立合作关系 中所述，在 IT 部门、财务部门和其他利益相关方之间建立合作关系和沟通机制非常重要，以确保他们都使用相同的工具或流程来保持一致性。在预算可能需要更改的情况下，提高沟通频率，来更快地应对这些更改。

实施步骤

- 定义组织内的成本语言：在组织内创建一种具有多个维度和分组的通用 AWS 成本语言。确保利益相关方了解预测粒度、定价模式和成本预测级别。
- 分析基于趋势的预测：使用基于趋势的预测工具，例如 AWS Cost Explorer 和 Amazon Forecast。从服务、账户、标签和成本类别等多个维度分析使用成本。
- 分析基于驱动因素的预测：确定业务驱动因素对云使用情况的影响，并分别针对每个驱动因素进行预测，以预先计算预期的使用成本。与业务部门负责人和利益相关方密切合作，了解新驱动因素的影响，计算预期成本变化以确定准确的预算。
- 调整现有的预测和预算编制流程：根据所采用的预测方法（如基于趋势的预测方法、基于业务驱动因素的预测方法，或者结合使用两种预测方法），确定预测和预算编制流程。预算应经过计算得出且切合实际，并以预测为基础。
- 配置警报和通知：使用 AWS Budgets 警报和成本异常检测来获取警报和通知。
- 与关键利益相关方开展定期审查：例如，与 IT、财务、平台团队和其他业务领域的利益相关方就业务方向和使用情况方面的变化达成一致。

资源

相关文档：

- AWS Cost Explorer
- AWS 成本和使用情况报告
- 利用 Cost Explorer 进行预测
- QuickSight 预测
- AWS Budgets

相关视频：

- 如何使用 AWS Budgets 来跟踪我的支出和使用情况
- AWS 成本优化系列：AWS Budgets

相关示例：

- Understand and build driver-based forecasting
- How to establish and drive a forecasting culture
- How to improve your cloud cost forecasting
- Using the right tools for your cloud cost forecasting

COST01-BP04 在组织流程中落实成本意识

在影响使用情况的新流程或现有流程中落实成本意识、创建成本透明度和成本问责制，并利用现有流程提高成本意识。在员工培训中落实成本意识。

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

必须在新的和现有的组织流程中落实成本意识。它是其他最佳实践的基础性先决条件之一。建议在可能的情况下重用和修改现有流程，这可以最大限度地减少对敏捷性和速度的影响。向技术团队以及业务和财务团队的决策者报告云成本，以提高成本意识，并为财务和业务利益相关方建立效率关键性能指标（KPI）。以下建议有助于在工作负载中落实成本意识：

- 确认变更管理包含成本度量，以量化变更对财务的影响。这有助于主动解决与成本相关的问题，并强调成本节省。
- 确认成本优化是运营能力的核心组成部分。例如，可以利用现有的事件管理流程来调查和确定成本及使用情况异常或成本超支的根本原因。
- 通过自动化或工具加快节省成本和实现业务价值。在考虑实施成本时，请在对话中加入投资回报（ROI）信息，以证明投入时间或资金的合理性。
- 通过对云支出（包括在基于承诺的购买选项、共享服务和市场购买上的支出）实施 对账或扣款来分配云成本，以推动极具成本意识的云消费。
- 扩展现有的培训和发展计划，在整个组织中开展成本意识培训。建议在其中加入持续的培训和认证。这样有助于建立一个能够自我管理成本和使用情况的组织。
- 利用免费的 AWS 原生工具，例如 AWS Cost Anomaly Detection、AWS Budgets 和 AWS Budgets Reports。

当组织持续采用云财务管理（CFM）实践时，这些行为就会在工作和决策过程中落地生根。最终带来的是在从构建新的云原生应用程序的开发人员，到分析这些新的云投资的 ROI 的财务经理之间建立起更具有成本意识的企业文化。

实施步骤

- 确定相关的组织流程：每个组织单位审核自己的流程，并确定影响成本和使用情况的流程。任何导致资源创建或终止的流程都需要进行审核。查找能够在企业中支持成本意识的流程，例如事件管理和培训。
- 建立可自我维持的对成本敏感的文化：确保所有相关的利益相关方都认同变更原因和影响是一种成本，这样他们就能理解云成本。这将使贵组织能够在创新方面建立一种自我维持的对成本敏感的文化。
- 将成本意识融入流程：对每个流程进行修改以提高成本意识。流程可能需要执行额外的预检查，例如评估成本的影响，或执行后期检查，以验证成本和使用情况是否发生了预期变化。可以扩展支持流程（如培训和事件管理），以包括成本和使用情况项目。

要获得帮助，请通过您的客户团队与 CFM 专家联系，或浏览以下资源和相关文档。

资源

相关文档：

- AWS 云财务管理

相关示例：

- 高效云成本管理战略
- 成本控制博客系列 3：如何应对成本冲击
- AWS Cost Management 初学者指南

COST01-BP05 报告和通知成本优化

设置云预算并配置用于检测异常使用情况的机制。配置相关工具，根据预定义目标发出成本和使用情况警报，并在任何使用情况超过这些目标时接收通知。定期召开会议，分析工作负载的成本效益，提高成本意识。

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

必须定期报告组织内成本和使用情况的优化情况。可以安排专门的会议来讨论成本绩效，或者将成本优化纳入工作负载的常规运营报告周期。利用服务和工具定期监控成本绩效，并抓住机会实施成本节省措施。

使用 AWS Cost Explorer 按多种筛选条件和粒度查看成本和使用情况，此工具提供控制面板和报告，例如按服务或按账户统计的成本、每日成本或市场成本。通过 AWS Budgets Reports 跟踪成本和使用情况在所配置预算中的消耗进度。

使用 AWS Budgets 设置自定义预算以跟踪成本和使用情况，并在超过阈值时快速响应从电子邮件或 Amazon Simple Notification Service（Amazon SNS）通知收到的警报。将首选预算期设置为每日、每月、每季度或每年，并创建具体的预算限制，以随时了解实际或预测成本和使用情况相对于预算阈值的情况。还可以根据这些警报设置警报和操作，以便自动运行，或在超出预算目标时通过审批流程运行。

启用关于成本和使用情况的通知，确保在成本和使用情况发生意外变化时您能够迅速采取行动。AWS Cost Anomaly Detection 让您能够减少意外成本，加强控制，同时不放慢创新速度。AWS Cost Anomaly Detection 可识别异常支出并找出根本原因，这有助于降低出现账单意外的风险。只需简单三步，即可创建自己的情境化监控器，在检测到任何异常支出时接收警报。

还可以将 QuickSight 与 AWS 成本和使用情况报告（CUR）数据结合使用，以提供包含更精细数据的高度定制的报告。利用 QuickSight，可以安排报告，并定期接收关于历史成本和使用情况或成本节省机会的成本报告电子邮件。查看我们的 Cost Intelligence Dashboard（CID）解决方案，该解决方案基于 QuickSight 构建，可为您提供高级可见性。

使用 AWS Trusted Advisor，该工具可提供指导，以验证预置的资源是否符合 AWS 的成本优化最佳实践。

通过可视化图表，对照精细的成本和使用情况数据检查节省计划建议。每小时图表显示按需支出和建议的节省计划承诺，让用户深入了解估计的节省额、节省计划覆盖率和节省计划使用率。这有助于组织了解节省计划如何应用于每小时的支出，而不必投入时间和资源来构建支出分析模型。

定期创建报告，其中包含来自 AWS Cost Explorer 的节省计划、预留实例和 Amazon EC2 合理调整大小建议的提要，以开始降低与稳定状态工作负载、空闲和未充分利用的资源相关的成本。确定并收回与已部署资源的云浪费有关的支出。当创建的资源大小不正确，或者观察到不同于预期的使用模式时，就会发生云浪费。遵循 AWS 最佳实践以减少浪费，或者请客户团队和合作伙伴协助优化并节省您的云成本。

定期生成报告，为您的资源提供更好的购买选项，以降低工作负载的单位成本。诸如节省计划、预留实例或 Amazon EC2 竞价型实例等购买选项可为容错工作负载节省大量成本，并让利益相关方（业务负责人、财务和技术团队）能够参与有关这些承诺的讨论。

分享包含可能有助于降低云总拥有成本（TCO）的机会或新发布公告的报告。采用新的服务、区域、功能、解决方案或新的方式来进一步削减成本。

实施步骤

- 配置 AWS Budgets：在所有账户中为工作负载配置 AWS Budgets。通过使用标签设置账户总支出预算和工作负载预算。
  - Well-Architected Lab：成本和使用情况治理
- 报告成本优化：定期讨论和分析工作负载的效率。使用已确立的指标，报告实现的指标和实现成本。找出任何负面趋势，加以解决，并确定可以在整个组织中推广的正面趋势。报告的参与者应该包括应用程序团队和负责人、财务团队以及云支出方面的关键决策者的代表。

资源

相关文档：

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Budgets
- AWS 成本和使用情况报告
- AWS Budgets 最佳实践
- Amazon S3 分析

相关示例：

- 开始优化 AWS 云成本的关键方法

COST01-BP06 主动监控成本

利用工具和控制面板主动监控工作负载的成本。定期用已配置的工具或开箱即用的工具审核成本，不要只在收到通知时才查看成本和类别。主动监控和分析成本有助于确定正面趋势，让您能够在整个组织中推广这些趋势。

在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

建议在组织内部主动监控成本和使用情况，而不仅仅是在出现异常或意外时才监控。在整个办公室或工作环境中，一目了然的控制面板能确保关键人员可以访问所需信息，并凸显组织对成本优化的重视程度。通过可见的控制面板，可以积极推动成功的结果，并在整个组织中加以实施。

创建一个每日或经常性的例程，以使用 AWS Cost Explorer 或任何其他控制面板（如 Amazon QuickSight）来查看成本并主动分析。通过分组和筛选，在 AWS 账户级、工作负载级或特定 AWS 服务级分析 AWS 服务的使用情况和成本，并验证它们是否符合预期。使用小时级和资源级粒度和标签来筛选和识别主要资源所产生的成本。还可以使用 Cost Intelligence Dashboard（一种 Amazon QuickSight 解决方案，由 AWS 解决方案架构师构建）构建自己的报告，并将预算与实际成本和使用情况进行比较。

实施步骤

- 报告成本优化：定期讨论和分析工作负载的效率。使用已确立的指标，报告实现的指标和实现成本。找出任何负面趋势，加以解决，并确定正面趋势，以便在整个组织中推广。报告应该包括应用程序团队和负责人、财务团队和管理团队的代表。
- 为成本和使用情况创建并激活每日粒度 AWS Budgets，以便及时采取行动来防止任何潜在的成本超支情况：可以使用 AWS Budgets 配置警报通知，以便在任何预算类型超出预配置阈值时，都会收到通知。利用 AWS Budgets 的最佳方式是将预期成本和使用情况设置为您的限值，这样一来，任何超出预算的情况均视为超支。
- 创建 AWS Cost Anomaly Detection 用于成本监控：AWS Cost Anomaly Detection 使用先进的机器学习技术来识别异常支出和根本原因，以便您快速采取行动。可以利用它来配置成本监控，以定义您想要评估的支出部分（例如，各项 AWS 服务、成员账户、成本分配标签和成本类别），并设置何时、何地以及如何接收警报通知。对于每个监控器，为业务负责人和技术团队附加多个警报订阅，包括每个订阅的名称、成本影响阈值和警报频率（单个警报、每日总结、每周总结）。
- 使用 AWS Cost Explorer 或将 AWS 成本和使用情况报告（CUR）数据与 Amazon QuickSight 控制面板集成，使组织的成本可视化：AWS Cost Explorer 有一个易于使用的界面，让您能够可视化、理解和管理随时间变化的 AWS 成本和使用情况。Cost Intelligence Dashboard 是一个可定制且可访问的控制面板，可帮助创建您自己的成本管理和优化工具的基础。

资源

相关文档：

- AWS Budgets
- AWS Cost Explorer

- 每日成本和使用情况预算
- AWS Cost Anomaly Detection

相关示例：

- AWS Cost Anomaly Detection 警报与 Slack 集成

COST01-BP07 及时了解新发布的服务

定期咨询专家或 AWS 合作伙伴，以便确定哪些服务和功能的成本更低。查看 AWS Blog 和其他信息源。

在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 不断添加新功能，以便您可以利用前沿技术来更快地进行试验和创新。您或许可以实施新的 AWS 服务和功能，以提高工作负载的成本效率。请定期查阅 AWS 成本管理、AWS News 博客、AWS 成本管理博客和 AWS 的新功能，了解有关新服务和功能发布的信息。“新增功能”博客文章简要概述了 AWS 发布的所有服务、功能和区域扩展公告。

实施步骤

- 订阅博客：转到 AWS Blog 页面，订阅“新增功能”博客和其他相关博客。您可以在通信偏好页面上用您的电子邮件地址进行注册。
- 订阅 AWS News：请定期查阅 AWS News 博客和 AWS 的新功能，以了解有关新服务和功能发布的信息。订阅 RSS 源，或通过电子邮件关注公告和发布的信息。
- 关注 AWS 降价：我们会定期对各项服务进行降价，这是 AWS 将我们的规模所带来的经济效益传递给客户的一种标准方式。截至 2023 年 9 月 20 日，AWS 自 2006 年以来已降价 134 次。如果您有任何因价格问题而悬而未决的业务决策，可在降价和新服务整合后再次考虑这些决策。可以了解以前的降价情况，包括 Amazon Elastic Compute Cloud（Amazon EC2）实例（在 AWS News 博客的降价类别中了解这些情况）。
- AWS 活动和交流会：参加当地的 AWS 峰会，以及与当地其他组织的任何当地交流会。如果您不能亲自参加，请尝试参加虚拟活动，从 AWS 专家和其他客户的业务案例中了解更多信息。
- 与客户团队交流：定期与您的客户团队交流，讨论行业趋势和 AWS 服务。与您的客户经理、解决方案架构师和支持团队沟通。

资源

相关文档：

- AWS 成本管理
- AWS 新增功能
- AWS 新闻博客

相关示例：

- Amazon EC2 – 15 年来为您优化和节省 IT 成本
- AWS News 博客 – 降价

COST01-BP08 建立对成本敏感的文化

在整个组织中实施更改或计划，以建立对成本敏感的文化。建议先从小范围着手，然后随着能力的增强和组织对云的使用的增加，在更广泛的范围实施更大型的计划。

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

在对成本敏感的文化中，可以在整个组织中以有机和分散的方式执行最佳实践，从而扩展成本优化和云财务管理（财务运营、云卓越中心、云运营团队等）。与严格的自上而下的集中式方法相比，只需要很少的工作，成本意识就能让您在整个组织中培养起较高的能力水平。

建立云计算成本意识（尤其对于云计算中的主要成本驱动因素）可以让团队了解成本角度的任何更改的预期结果。访问云环境的团队应了解定价模式，以及传统的本地数据中心与云计算之间的区别。

对成本敏感的文化的主要好处是，技术团队会主动且持续地优化成本（例如，在构建新工作负载或更改现有工作负载时，优化成本会被视为一项非功能性需求），而不是根据需要被动地进行成本优化。

文化上的细微改变可对您当前和将来的工作负载的效率产生重大影响。这种情况的示例包括：

- 在工程团队中提供可见性并提高意识，以了解他们所做的事情，以及他们在成本方面的影响。
- 以游戏方式展示整个组织的成本和使用情况。这可以通过一个公开可见的控制面板或比较各个团队的规范化成本和使用情况的报告（例如，每个工作负载的成本和每笔交易的成本）来完成。
- 认可成本效率。公开或私下奖励自愿或主动实现的成本优化成果，并从错误中吸取教训，以免日后重蹈覆辙。

- 创建自上而下的组织要求，确保工作负载按预定义的预算运行。
- 询问变更的业务要求，以及所请求的架构基础设施或工作负载配置变更的成本影响，以确保您只需为所需的内容付费。
- 确保变更规划者了解对成本有影响的预期变更，并确保这些变更得到利益相关方的确认，以经济高效的方式实现业务成果。

实施步骤

- 向技术团队报告云成本：提高成本意识，为财务和业务利益相关方建立效率 KPI。
- 将计划变更告知利益相关方或团队成员：创建一个议程项目，在每周变更会议期间讨论计划变更及其对工作负载的成本效益影响。
- 与客户团队交流：建立与客户团队定期沟通的机制，讨论行业趋势和 AWS 服务。与您的客户经理、架构师和支持团队沟通。
- 分享成功案例：分享关于任何工作负载、AWS 账户 或组织的成本降低的成功案例，以便围绕成本优化树立积极的态度并给予鼓励。
- 培训：确保技术团队或团队成员接受了关于 AWS Cloud 资源成本意识方面的培训。
- AWS 活动和交流会：参加当地的 AWS 峰会，以及与当地其他组织的任何当地交流会。
- 订阅博客：转到 AWS Blog 页面，订阅新增功能博客和其他相关博客，以关注 AWS 分享的新发布、实施、示例和更改。

资源

相关文档：

- AWS 博客
- AWS 成本管理
- AWS 新闻博客

相关示例：

- AWS 云财务管理

COST01-BP09 量化通过成本优化实现的业务价值

通过量化成本优化带来的业务价值，可以了解组织获得的全部益处。由于成本优化是一项必要的投资，因此量化业务价值之后，您就可以向利益相关方说明投资回报。如果能够量化业务价值，在未来的成本优化投资中，就可以从利益相关方那里得到更多支持，并获得一个框架来衡量组织成本优化活动的成果。

在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

量化业务价值是指衡量企业从他们所采取的行动和作出的决策中获得的益处。业务价值可以是有形的（例如减少支出或增加利润），也可以是无形的（例如改善品牌声誉或提高客户满意度）。

量化成本优化带来的业务价值，意味着要确定您从提高支出效率的努力中获得了多少价值或益处。例如，如果一家公司花费 10 万美元在 AWS 上部署工作负载，然后对其进行优化，则在不牺牲质量或产出的情况下，新成本仅为 8 万美元。在这种情况下，成本优化带来的量化业务价值是节省 2 万美元。但是，除了节约成本外，企业还可以从缩短交付时间、提高客户满意度或成本优化工作带来的其他指标等方面量化价值。利益相关方需要就成本优化的潜在价值、优化工作负载的成本和回报价值作出决定。

除了报告通过成本优化节省的费用外，建议量化其创造的附加价值。成本优化的益处通常以每项业务成果降低的成本来量化。例如，当您购买节省计划时，可以量化 Amazon Elastic Compute Cloud（Amazon EC2）节省的成本，这可以降低成本并维持工作负载输出水平。当闲置的 Amazon EC2 实例移除，或者未连接的 Amazon Elastic Block Store（Amazon EBS）卷删除时，可以量化削减的 AWS 支出成本。

然而，成本优化带来的益处绝不仅仅在于降低或规避成本。考虑捕获其他数据来衡量效率提升值和业务价值。

实施步骤

- 评估业务效益：这是分析和调整 AWS Cloud 成本的过程，其方式可以最大限度地提高从花费的每一美元中获得的效益。与其专注于降低成本而忽略业务价值，不如考虑业务效益和成本优化的投资回报，这样可能会让您所花的钱产生更大价值。这就是要明智地花钱，在能产生最大回报的领域进行投资和支出。
- 分析预测 AWS 成本：预测可帮助财务利益相关方与其他内部和外部组织利益相关方设定预期，并可以提高组织的财务可预测性。AWS Cost Explorer 可用于预测成本和使用情况。

资源

相关文档：

- AWS Cloud 经济性
- AWS 博客
- AWS 成本管理
- AWS 新闻博客
- Well-Architected 可靠性支柱白皮书
- AWS Cost Explorer

相关视频：

- 在 AWS 上使用 Windows 挖掘业务价值

相关示例：

- 衡量和最大限度提升 Customer 360 的业务价值
- 采用亚马逊云科技托管式数据库的业务价值
- 亚马逊云科技为独立软件供应商带来的业务价值
- 云现代化的业务价值
- 向亚马逊云科技迁移的业务价值

支出和使用情况意识

问题

- COST 2. 如何管理使用情况？
- COST 3. 如何监控成本和使用情况？
- COST 4. 如何停用资源？

COST 2. 如何管理使用情况？

制定各种策略和机制，确保花费适当的成本来达到目标。采用制约与平衡方法，您可以在不超支的情况下进行创新。

最佳实践

- COST02-BP01 根据组织的要求制定各种策略
- COST02-BP02 实施方向性目标和执行性目标
- COST02-BP03 实施账户结构
- COST02-BP04 实施组和角色
- COST02-BP05 实施成本控制
- COST02-BP06 跟踪项目生命周期

COST02-BP01 根据组织的要求制定各种策略

制定策略，确定组织应该如何管理资源，并定期执行检查。策略应该涵盖资源和工作负载的成本，包括在资源生命周期内的创建、修改和停用。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

了解组织的成本和驱动因素，对于有效管理成本和使用情况以及找到降低成本的机会至关重要。在组织中，通常会有多个团队运行多个工作负载。这些团队可能在不同的组织单位，每个单位都有自己的收入来源。将资源成本分摊到工作负载、各个组织或产品负责人，这样既推动更高效的资源使用模式，又能减少浪费。准确的成本和使用情况监控有助于您了解如何优化工作负载，以及各部门和产品的盈利能力。利用这些知识，可以就在组织内的何处分配资源作出更明智的决策。组织中各层级的人员都了解使用情况是推动变化的关键，因为使用情况变化会导致成本变化。考虑采用多元方法来了解使用情况和支出情况。

执行治理的第一步是按照组织要求来针对云的使用制定策略。这些策略定义组织如何使用云以及如何管理资源。策略应涵盖与成本或使用情况有关的资源和工作负载的所有方面，包括资源生命周期内的创建、修改和停用。确认云环境中的任何更改都是遵循相应策略和程序实施的。在 IT 变更管理会议期间，提出问题以了解计划变更的成本影响（无论是增加还是减少）、业务理由以及预期结果。

策略应该简单易懂，能够在整个组织中有效实施。策略还需要易于遵守和解释（以便落实），并且要具体（不会在各团队之间造成误解）。此外，需要定期审查这些策略（例如我们的机制），并在客户业务状况或业务优先级发生变化时及时更新策略，以免导致策略过时。

从广泛的、高层级的策略开始，例如使用哪个地理区域，或者一天中应该运行资源的时间。逐步为各组织单位和工作负载细化策略。常见策略包括可以使用哪些服务和功能（例如，测试和开发环境中性能较低的存储），哪些类型的资源可供不同团队使用（例如，开发账户中最大规模的资源为中等大小），以及这些资源将使用多长时间（可能是临时使用、短期使用或在特定时间段内使用）。

策略示例

以下是侧重于成本优化的策略示例，可以参考该策略来创建自己的云治理策略。请确保根据组织和利益相关方的要求调整策略。

- 策略名称：定义清晰的策略名称，例如“资源优化和成本削减策略”。
- 目的：解释为什么应使用此策略以及预期结果如何。此策略的目标是确认，在为了满足业务要求而部署和运行所需工作负载方面，有最低的成本要求。
- 范围：明确定义谁应使用此策略以及何时应使用此策略，例如 DevOps X 团队，为美国东部的客户在 X 环境（生产或非生产）中使用此策略。

策略声明

1. 根据工作负载的环境和业务要求（开发、用户验收测试、预生产或生产），选择 us-east-1 或多个美国东部区域。
2. 安排 Amazon EC2 和 Amazon RDS 实例在早上 6 点到晚上 8 点 [美国东部标准时间（EST）] 之间运行。
3. 在处于不活动状态 8 小时之后，停止所有未使用的 Amazon EC2 实例，在处于不活动状态 24 小时之后，停止未使用的 Amazon RDS 实例。
4. 在非生产环境中处于不活动状态 24 小时之后，终止所有未使用的 Amazon EC2 实例。提醒 Amazon EC2 实例拥有者（根据标签）查看其生产环境中已停止的 Amazon EC2 实例，并告知他们，如果其 Amazon EC2 实例在 72 小时内未使用，将会被终止。
5. 使用通用实例系列和大小（如 m5.large），然后使用 AWS Compute Optimizer，根据 CPU 和内存利用率调整实例大小。
6. 优先使用自动扩缩，根据流量动态调整运行的实例数量。
7. 为非关键工作负载使用竞价型实例。
8. 查看容量需求，为可预测的工作负载使用节省计划或预留实例，并通知云财务管理团队。
9. 使用 Amazon S3 生命周期策略，将不经常访问的数据移动到更便宜的存储层。如果未定义保留策略，请使用 Amazon S3 Intelligent Tiering，自动将对象移动到存档层。
10. 使用 Amazon CloudWatch 监控资源利用率并设置警报来触发扩展事件。
11. 对于每个 AWS 账户，使用 AWS Budgets，根据成本中心和业务单位为您的账户设置成本和使用情况预算。
12. 使用 AWS Budgets 为账户设置成本和使用情况预算，有助于您控制支出和避免意外账单，从而更好地控制成本。

程序：提供实施此策略的详细过程，或参考介绍如何实施各个策略声明的其他文档。此部分应提供关于如何执行策略要求的分步说明。

要实施此策略，可以使用各种第三方工具或 AWS Config 规则来检查是否符合策略声明，并使用 AWS Lambda 函数触发自动修复操作。您也可以使用 AWS Organizations 来强制执行策略。此外，您应定期查看资源使用情况，并在必要时调整策略，以确认策略继续满足业务需求。

实施步骤

- 与利益相关方交流：要制定策略，让组织内的利益相关方（云业务办公室、工程师或实施策略的职能部门决策者）详细说明他们的要求并记录下来。采用迭代方法，首先大致进行，然后在每一步中不断细化到最小单元。团队成员包括与工作负载切身相关的人员（例如组织单位或应用程序负责人）以及支持小组（例如安全和财务团队）。
- 获得确认：确保团队就哪些人可以访问和部署到 AWS Cloud 的策略达成一致。确保这些人遵守组织的策略，并确认其资源创建符合商定的策略和程序。
- 创建入职培训课程：要求新组织成员完成入职培训课程，以建立成本意识并了解组织要求。他们可能会采取不同于以往经验的策略，或者根本不考虑这些策略。
- 定义运行工作负载的位置：定义工作负载的运行位置，包括国家/地区以及国家/地区中的区域。此信息用于映射到 AWS 区域 和可用区。
- 定义服务和资源并对其进行分组：定义工作负载所需的服务。对于每项服务，指定类型、大小和所需资源数量。按职能定义资源组，如应用程序服务器或数据库存储。资源可属于多个组。
- 定义用户并按职能对其进行分组：定义与工作负载交互的用户，侧重于用户的工作范畴及其使用工作负载的方式，而不是侧重于他们的身份或其在组织中的职位。将类似用户或职能分组在一起。可以使用 AWS 托管式策略作为指南。
- 定义操作：使用前面确定的位置、资源和用户，定义每项在其生命周期（开发、运行和停用）内实现工作负载成果所需的操作。根据每个位置的组（而不是组中的个别元素）确定操作。首先广泛读写，然后细化到每项服务的具体操作。
- 定义审核期：工作负载和组织要求可能会随着时间的推移而发生变化。定义工作负载审核计划，确保其与组织优先事项保持一致。
- 记录策略：确认已定义的策略是否可按组织的要求访问。这些策略用于实施、维护和审核对环境的访问。

资源

相关文档：

- 云中的变更管理

- 针对工作职能的 AWS 托管式策略
- AWS 多账户计费策略
- AWS 服务的操作、资源和条件键
- AWS 管理和治理
- 使用 IAM 策略控制对 AWS 区域 的访问
- 全球基础设施区域和可用区

相关视频：

- 大规模的 AWS 管理和治理

COST02-BP02 实施方向性目标和执行性目标

实施工作负载的成本和使用情况方向性目标和执行性目标。方向性目标为组织在预期结果方面指明了方向，而执行性目标则提供了要为工作负载实现的具体可衡量结果。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

为组织制定成本和使用情况方向性目标和执行性目标。作为一个在 AWS 上不断发展壮大的组织，针对成本优化设定方向性目标并进行跟踪，这一点非常重要。这些目标或关键绩效指标（KPI）可以是按需支出百分比等内容，也可以是采用某些优化服务（比如 AWS Graviton 实例或 gp3 EBS 卷类型）。设定可衡量且可实现的方向性目标，有助于您衡量效率的提高情况，这对于业务运营不可或缺。方向性目标为组织提供有关预期结果的指引和方向。

执行性目标则明确要实现的具体可衡量的结果。简言之，方向性目标是指您前进的方向，而执行性目标就是在这个方向上的进展情况，以及应何时实现这个目标（使用具体、可衡量、可分配、切合实际、适时的指导原则，即 SMART 指导原则）。方向性目标的一个示例是：在略微（非线性）增加成本的情况下，显著提升平台使用量。执行性目标的一个示例是：在成本增长不到 5% 的情况下，将平台使用量提升 20%。另一个常见的方向性目标是每 6 个月提高一次工作负载的效率。相应的执行性目标则是，需要每 6 个月将各项业务指标的成本缩减 5%。使用正确的指标，合理计算来为组织设定 KPI。可以从基本 KPI 开始，以后再根据业务需求进行改进。

就成本优化而言，其方向性目标是提高工作负载效率，这对应于逐步降低每项业务成果的工作负载成本。为所有工作负载实施此方向性目标，并设定执行性目标，例如每 6 至 12 个月将效率提高 5%。在云端，可以通过培养成本优化能力以及发布新服务和功能来做到这一点。

执行性目标是您为实现方向性目标而要达到的可量化基准，这些基准将您的实际结果与执行性目标进行比较。使用 KPI 为计算服务（例如竞价型实例采用、Graviton 采用、最新实例类型和按需覆盖范围）、存储服务（例如 EBS GP3 采用、过时的 EBS 快照和 Amazon S3 Standard 存储）或数据库服务使用情况（例如 RDS 开源引擎、Graviton 采用和按需覆盖范围）的单位成本建立基准。这些基准和 KPI 有助于验证您是否通过最经济高效的方式使用 AWS 服务。

下表提供了标准 AWS 指标列表以供参考。每个组织都可以为这些 KPI 设定不同的执行性目标值。

类别	KPI	描述
计算	EC2 使用情况覆盖率	使用 SP+RI+Spot 的 EC2 实例（以成本或小时计）与 EC2 实例的总计（以成本或小时计）的比较
计算	计算 SP/RI 使用率	使用的 SP 或 RI 小时数与可用 SP 或 RI 总小时数的比较
计算	EC2/小时成本	EC2 成本除以该小时内运行的 RDS 实例数量
计算	vCPU 成本	所有实例每个 vCPU 的成本
计算	最新一代实例	Graviton（或其他现代实例类型）上的实例百分比
数据库	RDS 覆盖率	使用 RI 的 RDS 实例（以成本或小时计）与 RDS 实例的总计（以成本或小时计）的比较
数据库	RDS 使用率	使用的 RI 小时数与可用 RI 总小时数的比较
数据库	RDS 正常运行时间	RDS 成本除以该小时内运行的 RDS 实例数量
数据库	最新一代实例	Graviton（或其他现代实例类型）上的实例百分比

类别	KPI	描述
存储	存储使用率	优化的存储成本（例如 Glacier、Deep Archive 或不频繁访问）除以总存储成本
标记	未标记的资源	Cost Explorer： 1. 筛选出服务抵扣金、折扣、税款、退款、商城，并复制最新的月度成本 2. 在 Cost Explorer 中选择仅显示未标记的资源 3. 将未标记资源中的数字除以每月成本。

此表包含执行性目标值或基准值，这些值应根据您的组织目标计算得出。您需要衡量业务的某些指标，并了解该工作负载的业务成果，以便确立准确、切合实际的 KPI。在评估组织内部的绩效指标时，要区分用于不同目的的各类指标。这些指标主要衡量技术基础设施的性能和效率，而不是直接衡量总体业务影响。例如，它们可能会跟踪服务器响应时间、网络延迟或系统正常运行时间。要评测基础设施能否有效支持组织的技术运营，这些指标必不可少。但是，您无法通过它们直接了解更广泛的业务目标，例如客户满意度、收入增长或市场份额。要全面了解业务绩效，请使用与业务成果直接相关的战略业务指标，补充这些效率指标。

您需要能够近乎实时地监控 KPI 及相关的成本节省机会，并不断跟踪进度。要着手定义和跟踪 KPI 目标，我们建议使用 Cloud Intelligence Dashboard（CID）中的 KPI 控制面板。根据来自成本和使用情况报告（CUR）的数据，KPI 控制面板提供了一系列建议的成本优化 KPI，能够设定自定义的方向性目标并不断跟踪进度。

如果通过其他解决方案来设定和跟踪 KPI 方向性目标，请务必让组织中的所有云财务管理利益相关方都采用这些方法。

实施步骤

- 定义预期使用量：首先，重点关注使用水平。与应用程序负责人、营销团队和更大的业务团队交流，了解工作负载的预期使用水平。客户需求可能如何随时间而变化？会因季节性增长或营销活动发生哪些变化？

- 定义工作负载资源和成本:定义使用水平后，量化满足这些使用量所需的工作负载资源变化。您可能需要增加工作负载组件的资源大小或数量，增加数据传输，或者在特定级别将工作负载组件更改为不同的服务。明确每个关键点的成本，并预测使用情况发生变化时的成本变化。
- 定义业务目标：从预期的使用情况和成本变化中获取输出，将其与预期的技术变化或正在开展的任何计划相结合，制定工作负载目标。方向性目标必须阐明使用情况和成本，以及两者之间的关系。方向性目标必须简明扼要，并能让人们了解企业对结果的期望（例如，确保未使用的资源保持在一定的成本水平以下）。您不需要为每种未使用的资源类型定义方向性目标，也不需要定义会导致方向性目标和执行性目标损失的成本。确认制定有组织计划（例如培训和教育等能力培养计划），以防成本呈预期变化，而使用情况无变化。
- 定义执行性目标：对于定义的每个方向性目标，指定一个可衡量的执行性目标。如果方向性目标是提高工作负载的效率，则执行性目标将量化改进量（通常为每一美元支出的业务产出）及获益时间。例如，可以设定一个方向性目标，最大限度地减少因过度预置而造成的浪费。有了此方向性目标，执行性目标可以是，因第一层生产工作负载中的计算过度预置而造成的浪费不应超过层级计算成本的 10%。此外，第二个执行性目标可以是，因第二层生产工作负载中的计算过度预置而造成的浪费不应超过层级计算成本的 5%。

资源

相关文档：

- 针对工作职能的 AWS 托管式策略
- AWS 多账户计费策略
- 使用 IAM 策略控制对 AWS 区域 的访问
- S.M.A.R.T. 目标
- How to track your cost optimization KPIs with the CID KPI Dashboard

相关视频：

- Well-Architected Lab：方向性目标和执行性目标（第 100 级）

相关示例：

- 什么是单位指标？
- 选择单位指标来支持您的业务
- 实践中的单位指标 – 经验教训

- 单位指标如何有助于在业务职能之间建立一致性

COST02-BP03 实施账户结构

实施与组织对应的账户结构。这有助于在整个组织内分摊和管理成本。

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

AWS Organizations 允许创建多个 AWS 账户，这有助于在扩展 AWS 上的工作负载时集中管理环境。可以通过在组织单位（OU）结构中分组 AWS 账户，并在每个 OU 下创建多个 AWS 账户，对组织层次结构进行建模。要创建账户结构，首先需要确定哪个 AWS 账户 将成为管理账户。之后，可以按照管理账户最佳实践和成员账户最佳实践，根据设计的账户结构创建新 AWS 账户 或选择现有账户作为成员账户。

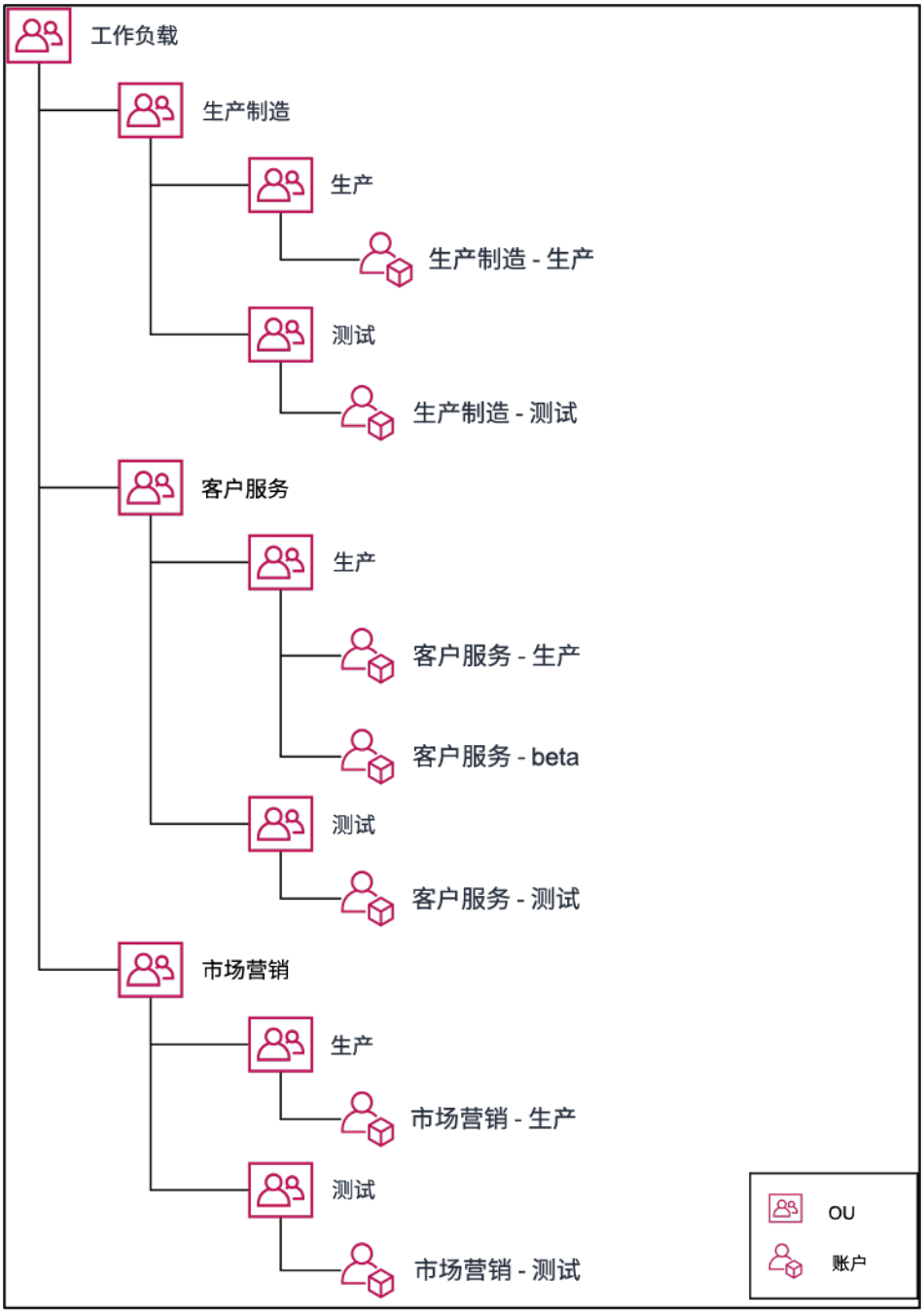
建议无论组织规模或使用情况如何，始终至少有一个管理账户和一个与之链接的成员账户。所有工作负载资源应仅驻留在成员账户中，不应在管理账户中创建任何资源。对于您应该拥有多少个 AWS 账户这一问题，没有标准答案。评测当前和未来的运营和成本模型，确保 AWS 账户 结构反映了组织的目标。有些公司出于业务原因会创建多个 AWS 账户，例如：

- 需要在组织部门、成本中心或特定工作负载之间实施管理或财务和计费隔离。
- AWS 服务限制设置为针对特定工作负载。
- 必须对工作负载和资源进行隔离和分离。

在 AWS Organizations 内，整合账单会在一个或多个成员账户与管理账户之间创建结构。通过成员账户，可以按组隔离和区分成本和使用情况。常见做法是每个组织部门（如财务、营销和销售）、每个环境生命周期（如开发、测试和生产）或每个工作负载（工作负载 a、b 和 c）具有单独的成员账户，然后使用整合账单将这些关联账户汇总在一起。

通过整合账单，可以将多个成员 AWS 账户 的付款整合至一个管理账户下，同时仍可查看每个关联账户的活动。由于成本和使用情况在管理账户中汇总，因此，可以最大限度地提高服务量折扣，并最大限度地利用承诺折扣（节省计划和预留实例）来获得最高折扣。

下图显示了如何对组织单位（OU）使用 AWS Organizations，以对多个账户进行分组，并在每个 OU 下放置多个 AWS 账户。建议将 OU 用于各种应用场景和工作负载，这提供了组织账户的模式。



在组织单位下将多个 AWS 账户 分组的示例。

AWS Control Tower 可以快速设置和配置多个 AWS 账户，确保治理符合组织要求。

实施步骤

- 定义分离要求：分离要求涉及多项因素，包括安全性、可靠性和财务结构。按顺序阐明每项因素，并详细说明工作负载或工作负载环境是否应与其他工作负载分开。安全性有助于满足访问和数据要求。可靠性管理限制，以便环境和工作负载不会影响其他项。定期查看 Well-Architected Framework 的

安全性和可靠性支柱，并遵循提供的最佳实践。财务结构创建严格的财务分离（不同的成本中心、工作负载所有权和问责制）。常见的分离示例包括在单独的账户中运行生产和测试工作负载，或使用单独的账户，以便可以将发票和账单数据提供给组织中的单个业务单位或部门，或拥有该账户的利益相关方。

- 定义分组要求：分组要求并不覆盖分离要求，而是用于协助管理。将无需分离的类似环境或工作负载分组在一起。例如，将一个或多个工作负载的多个测试或开发环境分组在一起。
- 定义账户结构：使用这些分离和分组，为每个组指定一个账户，并确保持续满足分离要求。这些账户有成员账户或关联账户。通过将这些成员账户分组到一个管理账户或付款人账户下，可以合并使用量，从而可以跨所有账户享有更大的批量折扣，这为所有账户提供一个账单。可以分离账单数据，并为每个成员账户提供其账单数据的单独视图。如果成员账户不能让任何其他账户看到自己的使用情况或账单数据，或者，如果需要 AWS 提供单独的账单，请定义多个管理账户或付款人账户。在这种情况下，每个成员账户都有自己的管理账户或付款人账户。资源应始终放置在成员账户或关联账户中。管理账户或付款人账户应只用于管理。

资源

相关文档：

- 使用成本分配标签
- 针对工作职能的 AWS 托管式策略
- AWS 多账户计费策略
- 使用 IAM 策略控制对 AWS 区域 的访问
- AWS Control Tower
- AWS Organizations
- 管理账户和成员账户的最佳实践
- 使用多个账户整理您的 AWS 环境
- 开启共享的预留实例和节省计划折扣
- 整合账单
- 整合账单

相关示例：

- 拆分 CUR 并共享访问权限

相关视频：

- AWS Organizations 简介
- 为 AWS Organizations 设置使用最佳实践的多账户 AWS 环境

相关示例：

- 为电信公司定义 AWS 多账户策略
- Best Practices for Optimizing AWS 账户
- Best Practices for Organizational Units with AWS Organizations

COST02-BP04 实施组和角色

实施与策略一致的组和角色，控制每个组中谁可以创建、修改或停用实例和资源。例如，实施开发组、测试组和生产组。这适用于 AWS 服务和第三方解决方案。

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

用户角色和组是设计和实施安全、高效的系统的基本构件。角色和组有助于组织在控制力需求与灵活性和生产率要求之间取得平衡，最终满足组织目标和用户需求。正如 AWS Well-Architected Framework 安全性支柱的身份与访问管理部分所建议的那样，您需要强大的身份管理和权限，以便在适当的条件下为合适的人员提供对正确资源的访问权限。用户仅获得完成任务所需的访问权限。这可最大限度地降低与未经授权访问或滥用相关的风险。

制定策略后，可以在组织内创建逻辑组和用户角色。这让您可以分配权限、控制使用情况，并有助于实施强健的访问控制机制，防止未经授权访问敏感信息。从大致的人员分组开始，这通常与组织单位和岗位角色（例如，IT 部门的系统管理员、财务主管或业务分析师）对应。这些组将执行相似任务并需要相似访问权限的人员划分在一起。角色定义组必须做什么。管理组和角色的权限比管理单个用户的权限更容易。角色和组可一致且系统性地为所有用户分配权限，防止出现错误和不一致。

当用户的角色发生变化时，管理员可以在角色或组级别上调整访问权限，而不是重新配置单个用户账户。例如，IT 部门的系统管理员需要创建所有资源的权限，而分析团队成员仅需要创建分析资源。

实施步骤

- 实施组：如有必要，请使用组织策略中定义的用户组实施相应的组。有关用户、组和身份验证的最佳实践，请参阅 AWS Well-Architected Framework 的安全性支柱。

- 实施角色和策略：使用组织策略中定义的操作，创建所需的角色和访问策略。有关角色和策略的最佳实践，请参阅 AWS Well-Architected Framework 的安全性支柱。

资源

相关文档：

- 针对工作职能的 AWS 托管式策略
- AWS 多账户计费策略
- AWS Well-Architected Framework 安全性支柱
- AWS Identity and Access Management（IAM）
- AWS Identity and Access Management 策略

相关视频：

- 为何使用身份与访问管理

相关示例：

- 使用 IAM 策略控制对 AWS 区域 的访问
- 开启您的云财务管理之旅：云成本运营

COST02-BP05 实施成本控制

根据组织策略以及定义的组和角色来实施控制。这样可以确保成本只根据组织要求的规定产生，例如，控制用户对区域或资源类型的访问。

在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

实施成本控制的第一步通常是进行相关通知设置，以便在发生成本或使用情况超出策略的事件时触发通知。可以迅速采取行动，并验证是否需要采取纠正措施，而不会限制工作负载或新活动，抑或是对它们产生负面影响。了解工作负载和环境限制后，就可以强制实施治理。AWS Budgets 允许您为 AWS 成本、使用情况和承诺折扣（节省计划和预留实例）设置通知和定义每月预算。可以在总成本级别（如所有成本）创建预算，也可以在更细粒度的级别创建预算，其中只包含特定的维度，如关联的账户、服务、标签或可用区。

使用 AWS Budgets 设置预算限制后，请使用 AWS Cost Anomaly Detection 来降低意外成本。AWS Cost Anomaly Detection 是一种成本管理服务，使用机器学习持续监控您的成本和使用情况，以检测异常支出。它可以帮助您识别异常支出和根本原因，以便您可以快速采取行动。首先，在 AWS Cost Anomaly Detection 中创建成本监控，然后通过设置美元阈值来选择提醒首选项（例如，对影响大于 1000 美元的异常情况发出提醒）。收到提醒后，可以分析造成异常情况的根本原因，以及对成本的影响。您还可以在 AWS Cost Explorer 中监控并执行自己的异常分析。

在 AWS 中通过 AWS Identity and Access Management 和 AWS Organizations 服务控制策略（SCP）强制执行治理策略。借助 IAM，可以安全地管理对 AWS 服务和资源的访问。可以使用 IAM 控制谁能创建或管理 AWS 资源、可创建的资源类型，以及可在何处创建。这最大限度地降低了在定义的策略之外创建资源的可能性。使用先前创建的角色和组，并分配 IAM 策略以强制实施正确的使用量。SCP 用于集中管控组织中所有账户的最大可用权限，从而确保账户始终在访问控制准则允许的范围内。SCP 仅在启用了所有功能的组织中可用，并且可以将 SCP 配置为默认情况下拒绝或允许对成员账户执行操作。有关实施访问管理的更多详细信息，请参阅《Well-Architected 安全性支柱白皮书》。

也可以通过 AWS 服务配额管理实施治理。通过确保为服务配额设置最低开销并进行准确维护，可以最大限度地减少组织要求以外的资源创建。要实现这一点，必须了解要求的改变速度、正在进行的项目（资源的创建和停用），并考虑配额更改的实施速度。必要时，服务配额可用于增加配额。

实施步骤

- 实施支出通知：使用定义的组织策略，创建 AWS Budgets 以在支出超出策略时收到通知。配置多个成本预算，每个账户一个，以通知您账户的总支出情况。在每个账户中为该账户内的较小单元配置额外的成本预算。这些单元因账户结构而异。一些常见示例包括 AWS 区域、工作负载（使用标签）或 AWS 服务。将电子邮件通讯组列表配置为通知的收件人，而不是个人的电子邮件账户。可以为超出金额的情况配置实际预算，或者使用预测预算通知预测使用情况。还可以预配置 AWS Budgets 操作，以强制实施特定 IAM 或 SCP 策略，或停止目标 Amazon EC2 或 Amazon RDS 实例。可以自动执行预算操作，也可以要求工作流程审批。
- 对异常支出实施通知：使用 AWS Cost Anomaly Detection 降低组织中的意外成本，并分析潜在异常支出的根本原因。创建成本监控来以指定粒度识别异常支出，并在 AWS Cost Anomaly Detection 中配置通知后，它会在检测到异常支出时向您发送提醒。这将让您能够分析导致异常的根本原因，并了解对成本的影响。配置 AWS Cost Anomaly Detection 时使用 AWS 成本类别来确定哪个项目团队或业务部门团队可以分析导致意外成本的根本原因，并及时采取必要的措施。
- 实施使用情况控制：使用定义的组织策略，实施 IAM 策略和角色，指定用户可执行和无法执行的操作。一个 AWS 策略中可能包含多个组织策略。采用定义策略时所用的方式，首先大致进行，然后在每一步施加更细粒度的控制。服务限制也是一种有效的使用情况控制措施。对所有账户实施正确的服务限制。

资源

相关文档：

- 针对工作职能的 AWS 托管式策略
- AWS 多账户计费策略
- 使用 IAM 策略控制对 AWS 区域 的访问
- AWS Budgets
- AWS Cost Anomaly Detection
- 控制您的 AWS 成本

相关视频：

- 如何使用 AWS Budgets 来跟踪我的支出和使用情况

相关示例：

- IAM 访问管理策略示例
- 服务控制策略示例
- AWS Budgets 操作
- 创建 IAM 策略以使用标签控制对 Amazon EC2 资源的访问权限
- 限制 IAM Identity 对特定 Amazon EC2 资源的访问权限
- Slack integrations for Cost Anomaly Detection using Amazon Q Developer in chat applications

COST02-BP06 跟踪项目生命周期

跟踪、衡量并审核项目、团队和环境的生命周期，以避免使用不必要的资源并为此付费。

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

通过有效地跟踪项目生命周期，组织可以加强规划、管理和资源优化，进而更好地控制成本。通过跟踪获得的见解对于作出明智的决策非常宝贵，有助于提高项目的成本效益和整体成功率。

跟踪工作负载的整个生命周期，有助于您了解何时不再需要工作负载或工作负载组件。现有的工作负载和组件可能看起来仍在使用中，但是当 AWS 发布新的服务或功能时，它们可能会被停用或采用。检查

工作负载的先前阶段。在工作负载进入生产之后，可以停用以前的环境或大幅降低其容量，直到再次需要它们为止。

可以用时间范围或提醒标记资源，以便确定工作负载的审核时间。例如，如果上一次审核开发环境是在几个月前，现在可能非常适合再次审核该环境，以探究是否能采用新服务或该环境是否在使用中。可以在 AWS 上使用 myApplications 对应用程序进行分组和标记，以管理和跟踪元数据，例如重要性、环境、上次审查时间和成本中心。您既可以跟踪工作负载的生命周期，也可以监控和管理应用程序的成本、运行状况、安全态势和性能。

AWS 提供了各种可用于实体生命周期跟踪的管理和治理服务。可以使用 AWS Config 或 AWS Systems Manager 提供一份详尽的 AWS 资源和配置清单。建议集成现有项目或资产管理系统，以跟踪组织内的活动项目和产品。将当前系统与 AWS 提供的丰富事件和指标集结合起来，您就可以构建重要生命周期事件的视图并主动管理资源，以减少不必要的成本。

与应用程序生命周期管理（ALM）类似，跟踪项目生命周期应涉及多个流程、工具、团队协同工作，例如设计和开发、测试、生产、支持及工作负载冗余。

通过仔细监控项目生命周期的每个阶段，组织可以获得重要的见解并增强掌控力，从而促进成功的项目规划、实施和完成。这种仔细的监督可以确保项目不仅符合质量标准，而且在预算范围内按时交付，从而提高总体成本效率。

有关实施实体生命周期跟踪的更多详细信息，请参阅《AWS Well-Architected 卓越运营支柱白皮书》。

实施步骤

- 建立项目生命周期监控流程：云卓越中心团队必须建立项目生命周期监控流程。建立结构化的系统化方法来监控工作负载，以改善项目的控制、可见性和性能。使监控过程透明化、协作化并注重持续改进，以最大限度地提高其有效性和价值。
- 执行工作负载审查：根据组织策略的定义，定期安排时间审核现有项目并执行工作负载审查。在审核方面投入的工作量应与组织的大致风险、价值或成本成比例。主要审核领域包括组织面临的意外事件或中断风险，或对组织所做的贡献（以收入或品牌声誉进行衡量）、工作负载的成本（以资源的总成本和运营成本进行衡量）和工作负载的使用情况（以单位时间的组织产出量进行衡量）。如果这些领域在生命周期内发生变化，则需要对工作负载进行调整，例如全部停用或部分停用。

资源

相关文档：

- Guidance for Tagging on AWS

- 什么是 ALM（应用程序生命周期管理）？
- 针对工作职能的 AWS 托管式策略

相关示例：

- 使用 IAM 策略控制对 AWS 区域 的访问

相关工具

- AWS Config
- AWS Systems Manager
- AWS Budgets
- AWS Organizations
- AWS CloudFormation

COST 3. 如何监控成本和使用情况？

建立策略和程序以便监控并适当分配您的成本。这让您能够衡量和改进此工作负载的成本效益。

最佳实践

- COST03-BP01 配置详细信息源
- COST03-BP02 在成本和使用情况中添加组织信息
- COST03-BP03 确定成本归属类别
- COST03-BP04 建立组织指标
- COST03-BP05 配置账单和成本管理工具
- COST03-BP06 根据工作负载指标分配成本

COST03-BP01 配置详细信息源

设置成本管理和报告工具，进一步分析成本和使用情况数据并提高透明度。配置工作负载以创建日志条目，便于跟踪和分割成本和使用情况。

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

利用成本管理工具中的详细账单信息（如每小时粒度），组织能够更详细地跟踪使用情况，并有助于组织找出成本增加的一些原因。这些数据来源最确切地反映了整个组织中的成本和使用情况。

可以使用 AWS Data Exports 创建 AWS 成本和使用情况报告（CUR）2.0 的导出。这是从 AWS 接收详细的成本和使用情况数据的新推荐方式。这种方法可以提供所有收费 AWS 服务的每日或每小时使用粒度、费率、成本和使用属性（与 CUR 的信息相同），以及一些改进。CUR 中包括所有可能的维度，例如标签、位置、资源属性和账户 ID。

根据要创建的导出的类型，有三种导出类型：标准数据导出、导出到集成 QuickSight 的成本与使用情况控制面板，或旧版数据导出。

- 标准数据导出：定期交付到 Amazon S3 的表的自定义导出。
- 成本与使用情况控制面板：导出到 QuickSight 和与之集成，用于部署预构建的成本与使用情况控制面板。
- 旧版数据导出：旧版 AWS 成本和使用情况报告（CUR）的导出。

可以使用以下自定义项创建数据导出：

- 包括资源 ID
- 拆分成本分配数据
- 每小时粒度
- 版本控制
- 压缩类型和文件格式

对于在 Amazon ECS 或 Amazon EKS 上运行容器的工作负载，启用拆分成本分配数据，以便可以根据容器工作负载使用共享计算和内存资源的方式，将容器成本分配给各个业务部门和团队。拆分成本分配数据将新的容器级资源的成本和使用情况数据引入 AWS 成本和使用情况报告。拆分成本分配数据通过计算集群上运行的各个 ECS 服务和任务的成本来计算。

成本与使用情况控制面板定期将成本与使用情况控制面板表导出到 S3 存储桶，并将预构建的成本与使用情况控制面板部署到 QuickSight。如果您想快速部署包含成本和使用情况数据的控制面板，且不需要自定义功能，请使用此选项。

如果需要，您仍然可以导出旧版 CUR，并可在其中集成其他处理服务（例如 AWS Glue）来准备数据以供分析，并使用 Amazon Athena 进行数据分析，通过 SQL 查询数据。

实施步骤

- 创建数据导出：使用所需数据创建自定义导出并控制导出的架构。使用基本 SQL 创建账单和成本管理数据导出，并通过集成 QuickSight 实现账单和成本管理数据的可视化。您还能以标准模式导出数据，以使用 Amazon Athena 等其他处理工具来分析数据。
- 配置成本和使用情况报告：使用账单控制台，配置至少一个成本和使用情况报告。配置采用每小时粒度的报告，以便包括所有标识符和资源 ID。还可以创建采用不同粒度的其他报告，以提供概括性摘要信息。
- 在 Cost Explorer 中配置每小时粒度：要以每小时粒度访问过去 14 天的成本和使用情况数据，请考虑在账单控制台中启用每小时和资源级别数据。
- 配置应用程序日志记录：确认应用程序记录所交付的每项业务成果，以便进行跟踪和衡量。确保该数据的粒度至少为每小时一次，以便与成本和使用情况数据匹配。有关日志记录和监控的更多详细信息，请参阅《Well-Architected 卓越运营支柱》。

资源

相关文档：

- AWS Data Exports
- AWS Glue
- QuickSight
- AWS 成本管理定价
- 为AWS资源添加标签
- 使用 Cost Explorer 分析成本
- 管理 AWS 成本和使用情况报告
- Well-Architected Operational Excellence Pillar

相关示例：

- AWS 账户设置
- Data Exports for AWS Billing and Cost Management
- AWS Cost Explorer 常见应用场景

COST03-BP02 在成本和使用情况中添加组织信息

根据组织、工作负载属性和成本分配类别定义标记方案，以便可以在成本管理工具中筛选和搜索资源或监控成本和使用情况。在可能的情况下，根据目的、团队、环境或与业务相关的其他标准，在所有资源中应用一致的标签。

在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 AWS 中应用标签以将组织信息添加到资源中，然后将该信息添加到成本和使用情况信息中。标签是键值对：键经过定义，必须在整个组织中唯一，值则对于一组资源唯一。键值对的一个示例是键为 `Environment`，值为 `Production`。生产环境中的所有资源都有这个键值对。借助标签，可以使用有意义、相关的组织信息对成本进行分类和跟踪。可以应用代表组织类别（例如成本中心、应用程序名称、项目或负责人）的标签，标识工作负载和工作负载的特征（例如测试或生产），在整个组织中对成本和使用情况进行归属。

当您将标签应用于 AWS 资源（如 Amazon Elastic Compute Cloud 实例或 Amazon Simple Storage Service 存储桶）并激活标签后，AWS 会将此信息添加到成本和使用情况报告中。可以在带标签和无标签的资源上运行报告并执行分析，以更好地遵守内部成本管理策略，并确保准确归属。

跨组织账户创建和实施 AWS 标签标准之后，您将能够一致且统一地管理和治理 AWS 环境。使用 AWS Organizations 中的标签策略定义有关如何在 AWS Organizations 账户的 AWS 资源上使用标签的规则。借助标签策略，可以采用标准化方法轻松标记 AWS 资源

AWS 标签编辑器可用于为多个资源添加、删除和管理标签。通过使用标签编辑器，您可以搜索要标记的资源，然后在搜索结果中管理这些资源的标签。

AWS Cost Categories 可用于向成本分配组织含义，而无需在资源上添加标签。可以将成本和使用情况信息映射到唯一的内部组织结构。可以定义类别规则，以使用账单维度（例如账户和标签）对成本进行映射和分类。除了标记之外，这还提供了另一个级别的管理能力。您还可以将特定账户和标签映射到多个项目。

实施步骤

- 定义标记方案：召集整个企业的所有利益相关方来定义方案。这通常包括担任技术、财务和管理角色的人员。定义所有资源必须具有的标签列表，以及资源应该具有的标签列表。确认标签名称和值在整个组织中是否一致。
- 标签资源：使用定义的成本归属类别，根据类别在工作负载中的所有资源上放置标签。使用 CLI、标签编辑器或 AWS Systems Manager 等工具提高效率。

- 实施 AWS Cost Categories：无需应用标签即可创建成本类别。Cost Categories 使用现有的成本和使用情况维度。根据方案创建类别规则，并在 Cost Categories 中加以实施。
- 实现标记自动化：为确保对所有资源保持高水平的标记，请自动进行标记，以便在创建资源时自动对其进行标记。使用诸如 AWS CloudFormation 之类的服务来验证资源在创建时是否已标记。还可以创建自定义解决方案以便使用 Lambda 函数实现自动标记，或使用微服务定期扫描工作负载并删除没有标记的任何资源，此方法非常适合测试和开发环境。
- 监控和报告标记：为确保在整个组织中保持高水平的标记，请跨工作负载报告和监控标签。可以使用 AWS Cost Explorer 查看带标签和无标签的资源的成本，也可以使用标签编辑器等服务。定期审核无标签的资源的数量，并执行操作添加标签，直到达到所需的标记级别。

资源

相关文档：

- 标记最佳实践
- AWS CloudFormation 资源标签
- AWS Cost Categories
- 为AWS资源添加标签
- 使用 AWS Budgets 分析成本
- 使用 Cost Explorer 分析成本
- 管理 AWS 成本和使用情况报告

相关视频：

- 如何标记 AWS 资源，以便按成本中心或项目划分账单
- 标记 AWS 资源

COST03-BP03 确定成本归属类别

确定组织类别，例如业务单位、部门或项目，这些类别可用于在组织中按照内部使用实体分配成本。利用这些类别来执行支出问责制，树立成本意识，并推动高效的使用行为。

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

成本分类过程在预算编制、会计、财务报告、决策、基准测试和项目管理中至关重要。通过对费用进行分级和分类，团队可以更好地了解自己整个云之旅中会产生的成本类型，从而有助于团队作出明智的决策并有效地管理预算。

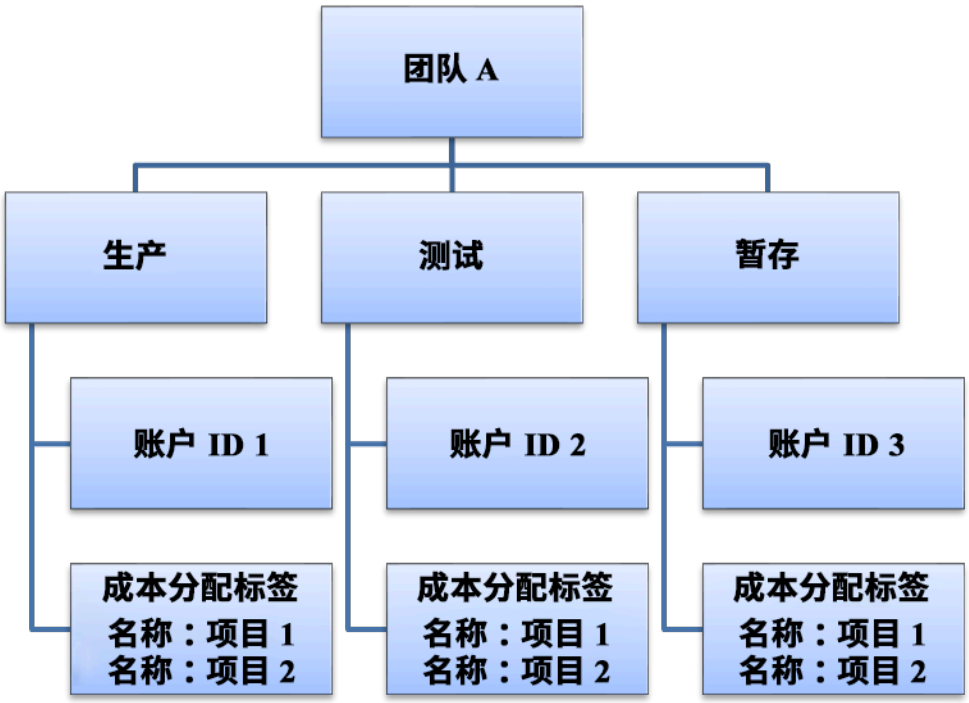
云支出问责制为严明的需求和成本管理提供了强有力的激励措施。最终，组织在将大部分云支出分配到使用这些资源的业务单位或团队后，可以显著地节省云成本。此外，通过分配云支出，有助于组织采用更多集中式云治理的最佳实践。

与财务团队和其他相关利益相关方合作，在定期沟通通话期间，了解必须如何在组织内部分配成本的要求。必须将工作负载成本分配至整个生命周期，包括开发、测试、生产和停用。了解组织如何对学习、员工培养和创意构思进行成本归类。这有助于将用于此目的的账户正确分配给培训和开发预算，而不是一般的 IT 成本预算。

与组织中的利益相关方一起定义成本归属类别后，使用 AWS Cost Categories 将成本和使用情况信息分组为 AWS Cloud 中有意义的类别，例如特定项目的成本、部门或业务部门的 AWS 账户。可以创建自定义类别，并根据使用各种维度（如账户、标签、服务或费用类型）定义的规则，将成本和使用情况信息映射到这些类别中。设置成本类别后，可以按这些类别查看成本和使用情况信息，从而让组织能够作出更好的战略和采购决策。这些类别也可在 AWS Cost Explorer、AWS Budgets 和 AWS 成本和使用情况报告 中查看。

例如，为业务部门（DevOps 团队）创建成本类别，并在每个类别下创建多个规则（每个子类别的规则），这些规则具有基于所定义分组的多个维度（AWS 账户、成本分配标签、服务或费用类型）。通过 Cost Categories，您可以使用基于规则的引擎来组织您的成本。您配置的规则按类别组织您的成本。在这些规则中，可以使用每个类别的多个维度进行筛选，例如特定 AWS 账户、AWS 服务或费用类型。可以在 AWS 账单与成本管理 和成本管理控制台中跨多个产品使用这些类别。这包括 AWS Cost Explorer、AWS Budgets、AWS 成本和使用情况报告 和 AWS Cost Anomaly Detection。

下图举例说明了如何通过多个团队（成本类别）、多个环境（规则）以及每个环境具有多个资源或资产（维度），对组织中的成本和使用情况信息进行分组。



成本和使用情况组织结构图

您也可以使用成本类别创建成本分组。创建成本类别后（为使用情况记录创建成本类别之后，允许在 24 小时内更新相应的值），这些类别将显示在 AWS Cost Explorer、AWS Budgets、AWS 成本和使用情况报告 和 AWS Cost Anomaly Detection 中。在 AWS Cost Explorer 和 AWS Budgets 中，成本类别显示为额外的计费维度。您可以使用此项筛选特定成本类别值，或者按成本类别分组。

实施步骤

- 定义组织类别：与内部利益相关方和业务部门会面，确定反映组织结构和要求的类别。这些类别应直接对应于现有财务类别的结构，例如业务部门、预算、成本中心或部门。了解云为您带来的业务成果（例如培训或教育），因为这些也是组织类别。
- 定义职能类别：与内部利益相关方和业务部门会面，确定反映业务所含职能的类别。这可以是工作负载名称或应用程序名称以及环境类型（例如生产、测试或开发）。
- 定义 AWS Cost Categories：使用 AWS Cost Categories 创建成本类别以整理成本和使用情况信息，并将 AWS 成本和使用情况映射到有意义的类别。可以将多个类别分配给一个资源，并且一个资源可以位于多个不同的类别中，因此可以根据需要定义任意数量的类别，以便可以使用 AWS Cost Categories 在分类的结构中管理成本。

资源

相关文档：

- 为AWS资源添加标签
- 使用成本分配标签
- 使用 AWS Budgets 分析成本
- 使用 Cost Explorer 分析成本
- 管理 AWS 成本和使用情况报告
- AWS Cost Categories
- 使用 AWS Cost Categories 管理成本
- 创建成本类别
- 标记成本类别
- 在成本类别中拆分费用
- AWS Cost Categories 的功能

相关示例：

- 使用 AWS Cost Categories 整理成本和使用情况数据
- 使用 AWS Cost Categories 管理成本

COST03-BP04 建立组织指标

建立此工作负载需要的组织指标。生成的客户报告或提供给客户的 Web 页面都属于工作负载指标。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

了解如何根据业务成功来衡量工作负载的输出。每个工作负载通常有一组表示性能的主要输出。如果您的工作负载复杂且包含许多组件，则可以对列表进行优先级排序，或者为每个组件定义和跟踪指标。与团队合作，了解要使用哪些指标。此部分将用于了解工作负载的效率，或每项业务产出的成本。

实施步骤

- 定义工作负载成果：与业务利益相关方召开会议，定义工作负载成果。这些主要用于衡量客户使用情况，因此必须是业务指标，而不是技术指标。每个工作负载应该有少量的概要指标（少于 5 个）。如果工作负载针对不同的应用场景产生多个结果，请将其分组为一个指标。
- 定义工作负载组件指标：如果工作负载大而复杂，或者可以轻松地将工作负载分为输入和输出定义明确的多个组件（例如微服务），则可以选择为每个组件定义指标。这项工作应反映组件的价值和成本。按照从大到小的顺序，从最大的组件开始，逐步处理较小的组件。

资源

相关文档：

- 为AWS资源添加标签
- 使用 AWS Budgets 分析成本
- 使用 Cost Explorer 分析成本
- 管理 AWS 成本和使用情况报告

COST03-BP05 配置账单和成本管理工具

根据组织的策略配置成本管理工具，以管理和优化云支出。这包括用于整理和跟踪成本和使用情况数据的服务、工具和资源，通过整合账单和访问权限增强控制，通过预算编制和预测改进规划，接收通知或提醒，并通过资源和定价优化降低成本。

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

为了建立强有力的问责制，首先将您的账户策略视为成本分配策略的一部分。做好这一点，可能会为您省下许多工作。否则就会出现意识不足的情况，造成更多痛点。

为了鼓励针对云支出建立问责制，应向用户授予权限，允许他们使用可查看其成本和使用情况的工具。AWS 建议您出于以下目的配置所有工作负载和团队：

- 整理：使用您自己的标记策略和分类方法，建立成本分配和治理基准。使用 AWS Control Tower 或 AWS Organization 等工具创建多个 AWS 账户。标记支持的 AWS 资源，并根据组织结构（业务部门、部门或项目）对资源进行有目的性的分类。为特定成本中心标记账户名称，并将其与 AWS Cost Categories 对应起来，以便为业务部门就其成本中心进行账户分组，这样业务部门负责人就可以在一个位置查看多个账户的使用情况。

- 访问权限：在整合账单中跟踪组织范围的账单信息。确认相应的利益相关方和业务负责人是否拥有访问权限。
- 控制：使用服务控制策略（SCP）、标签策略、IAM 策略和预算警报时，使用适当的防护机制建立有效的治理机制，以防止出现意外情况。例如，可以使用有效的控制机制，仅允许团队在首选区域内创建特定资源，并防止没有特定标签（例如成本中心）的资源创建。
- 当前状态：配置显示当前成本和使用情况水平的控制面板。该控制面板应位于工作环境中的显眼位置，类似于操作控制面板。可以导出数据，并使用 AWS 成本优化中心中的成本与使用情况控制面板或任何支持的产品来实现此可见性。您可能需要为不同的角色创建不同的控制面板。例如，管理器控制面板可能不同于工程控制面板。
- 通知：当成本或使用情况超过定义的限制并且 AWS Budgets 或 AWS 异常检测出现异常时，提供通知。
- 报告：汇总所有成本和使用情况信息。通过详细的、可归因的成本数据，提高对云支出的认识，加强问责制。创建报告（这些报告与其使用团队相关）并包含建议。
- 跟踪：对照配置的方向性目标或执行性目标显示当前的成本和使用情况。
- 分析：允许团队成员使用不同的筛选条件（资源、账户、标签等）执行自定义和深入分析，精确到每小时、每天或每月的粒度。
- 检查：随时了解最新的资源部署和成本优化机会。使用 Amazon CloudWatch、Amazon SNS 或 Amazon SES 获取组织级别的资源部署的通知。使用 AWS Trusted Advisor 或 AWS Compute Optimizer 审核成本优化建议。
- 趋势报告：以所需的粒度显示所需期限内成本和使用情况的变化。
- 预测：使用您创建的预测控制面板，显示估计的未来成本、估计资源使用情况和支出。

可以使用 AWS 成本优化中心来了解从集中位置整合的潜在成本节约机会，并创建数据导出以与 Amazon Athena 集成。还可以使用 AWS 成本优化中心来部署成本与使用情况控制面板，该控制面板利用 QuickSight 进行交互式成本分析和安全的成本洞察共享。

如果组织不具备必备技能或带宽，则可以与 AWS ProServ、AWS Managed Services（AMS）或 AWS 合作伙伴合作。您也可以使用第三方工具，但请务必验证价值主张。

实施步骤

- 允许对工具进行基于团队的访问：配置账户并创建组，这些组可以访问所需的成本和使用情况报告来了解其使用情况，并使用 AWS Identity and Access Management 控制对 AWS Cost Explorer 等工具的访问权限。这些组必须包括负责或管理应用程序的所有团队的代表。这证明每个团队都可以访问他们的成本和使用情况信息，以跟踪使用情况。

- 整理成本标签和类别：跨团队、业务部门、应用程序、环境和项目整理成本。使用资源标签来按成本分配标签整理成本。使用标签、账户、服务等，根据维度创建成本类别，以映射成本。
- 配置 AWS Budgets：在所有账户中为工作负载配置 AWS Budgets。使用标签和成本类别，设置账户总支出预算和工作负载预算。在 AWS Budgets 中配置通知，以便在您超出预算金额或估计成本超出预算时收到警报。
- 配置 AWS 成本异常检测：针对您的账户、核心服务或成本类别使用 AWS 成本异常检测，以监控成本和使用情况，并检测异常支出。可以在汇总的报告中单独收到警报，以及在电子邮件或 Amazon SNS 主题中收到警报，以便您分析和确定异常的根本原因，并确定导致成本增加的原因。
- 使用成本分析工具：为工作负载和账户配置 AWS Cost Explorer，实现成本数据可视化，以便进一步分析。为工作负载创建一个控制面板，用于跟踪总体支出、工作负载的关键使用指标，以及基于历史成本数据的未来成本预测。
- 使用成本节省分析工具：通过 AWS 成本优化中心利用量身定制的建议（包括删除未使用的资源、合理调整大小、节省计划和预留）和 Compute Optimizer 建议来发现可节省成本的机会。
- 配置高级工具：可以选择创建视觉对象以促进交互式分析和成本洞察的分享。借助 AWS 成本优化中心上的数据导出功能，可以为组织创建由 QuickSight 提供支持的成本与使用情况控制面板，从而提供更多详细信息和粒度。还可以通过使用 Amazon Athena 中的数据导出来实现高级分析功能以进行高级查询，并在 QuickSight 上创建控制面板。与 AWS 合作伙伴合作，将云管理解决方案用于整合的云账单监控和优化。

资源

相关文档：

- What is AWS 账单与成本管理 and Cost Management?
- 建立您的最佳实践 AWS 环境
- 标记 AWS 资源的最佳实践
- 标记您的 AWS 资源
- AWS Cost Categories
- 使用 AWS Budgets 分析成本
- 使用 AWS Cost Explorer 分析成本
- What is AWS Data Exports?

相关视频：

- Deploying Cloud Intelligence Dashboards

- Get Alerts on any FinOps or Cost Optimization Metric or KPI

相关示例：

- Cost and Usage Dashboard powered by QuickSight
- AWS Cost and Usage Governance 讲习会

COST03-BP06 根据工作负载指标分配成本

根据使用情况指标或业务成果分配工作负载的成本，以便衡量工作负载的成本效率。实施一个流程，使用分析服务来分析成本和使用情况数据，以便深入了解成本因素和退款功能。

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

成本优化是指以最低的价格实现业务成果，这只能通过按工作负载指标分配工作负载成本（按工作负载效率衡量）来实现。通过日志文件或其他应用程序监控来监控定义的工作负载指标。将此数据与工作负载的成本（可通过查看具有特定标签值或账户 ID 的成本获得）相结合。每小时进行一次此分析。如果有静态成本要素（例如，持续运行的后端数据库）且请求率不同（例如，使用量高峰在上午 9 点至下午 5 点，晚间的请求数量很少），则效率通常会变化。了解静态成本和可变成本之间的关系，有助于您专注于优化活动。

与 Amazon Elastic Container Service（Amazon ECS）和 Amazon API Gateway 上的容器化应用程序等资源相比，为共享资源创建工作负载指标可能并非易事。但是，您可以通过某些方法来分类使用情况并跟踪成本。如果您需要跟踪 Amazon ECS 和 AWS Batch 共享资源，则可以在 AWS Cost Explorer 中启用拆分成本分配数据。通过拆分成本分配数据，您可以了解并优化容器化应用程序的成本和使用情况，并根据共享计算和内存资源的使用情况，将应用程序成本分配给各个业务实体。

实施步骤

- 将成本分配到工作负载指标：使用定义的指标和配置的标签，创建结合工作负载输出和工作负载成本的指标。使用 Amazon Athena 和 Amazon QuickSight 等分析服务，为整个工作负载和任何组件创建效率控制面板。

资源

相关文档：

- 为AWS资源添加标签
- 使用 AWS Budgets 分析成本
- 使用 Cost Explorer 分析成本
- 管理 AWS 成本和使用情况报告

相关示例：

- 利用 AWS 拆分成本分配数据提高 Amazon ECS 和 AWS Batch 的成本可见性

COST 4. 如何停用资源？

在从项目开始到结束的过程中实施变更控制和资源管理。这可以确保您关闭或终止未使用的资源，以便减少浪费。

最佳实践

- COST04-BP01 在资源生命周期内跟踪资源
- COST04-BP02 实施停用流程
- COST04-BP03 停用资源
- COST04-BP04 自动停用资源
- COST04-BP05 执行数据留存策略

COST04-BP01 在资源生命周期内跟踪资源

制定和实施一种方法，在资源生命周期内跟踪资源及其与系统的关联。您可以使用标签来标识资源的工作负载或功能。

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

停用不再需要的工作负载资源。一个常见的示例是用于测试的资源：在测试完成后，可以将资源删除。使用标签跟踪资源（并对这些标签运行报告），可以帮助您识别要停用的资产，因为将不会使用这些资产，或者这些资产的许可证将过期。使用标签是跟踪资源的一种有效方法，它通过标记资源的功能或资源的已知可停用日期来跟踪资源。然后，可以对这些标签运行报告。功能标签的示例值是 `feature-X testing`，用于根据工作负载生命周期标识资源的用途。另一个例子是对资源使用 `LifeSpan` 或 `TTL`，例如待删除的标签键名称和值，用于定义停用的时间段或具体时间。

实施步骤

- 实施标记方案：实施标记方案，标识资源所属的工作负载，从而确保相应地标记工作负载中的所有资源。标签可以帮助您按用途、团队、环境或与业务相关的其他条件对资源进行分类。有关标签应用场景、策略和技术的更多详细信息，请参阅 AWS Tagging Best Practices。
- 实施工作负载吞吐量或输出监控：实施工作负载吞吐量监控或警报，在输入请求或输出完成时启动。将其配置为在工作负载请求或输出下降到零时发出通知，指示不再使用工作负载资源。如果在正常情况下，工作负载周期性地下降到零，则加入时间因素。有关未使用或未充分利用的资源的更多详细信息，请参阅 AWS Trusted Advisor Cost Optimization checks。
- 为 AWS 资源分组：为 AWS 资源创建组。可以使用 AWS Resource Groups 来整理和管理同一个 AWS 区域 中的 AWS 资源。可以将标签添加到大多数资源中，帮助标识资源并对组织内的资源进行排序。使用标签编辑器向支持的资源批量添加标签。考虑使用 AWS Service Catalog 创建、管理和向最终用户分发已批准的产品组合，并管理产品生命周期。

资源

相关文档：

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS Trusted Advisor Cost Optimization Checks
- 为AWS资源添加标签
- 发布自定义指标

相关视频：

- How to optimize costs using AWS Trusted Advisor

相关示例：

- 组织 AWS 资源
- 使用 AWS Trusted Advisor 优化成本

COST04-BP02 实施停用流程

实施一个流程来确定和停用未使用的资源。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在整个组织中实施标准化流程，以确定和删除未使用的资源。该流程应该定义执行搜索的频率以及删除资源的流程，以确认满足所有组织要求。

实施步骤

- 创建并实施停用流程：与工作负载开发人员和负责人合作，为工作负载及其资源构建停用流程。该流程应涵盖一种方法，验证工作负载是否正在使用以及每个工作负载资源是否正在使用。详细说明停用资源所需的步骤，将资源从服务中删除，同时确保符合所有法规要求。应包含任何关联的资源，例如许可证或附加的存储。向工作负载负责人发送已开启停用流程的通知。

  使用以下停用步骤来指导您确定在流程中应检查的内容：

  - 确定要停用的资源：确定 AWS Cloud 中符合停用条件的资源。记录所有必要的信息并安排停用。在您的时间表中，请务必考虑在此过程中是否（以及何时）会出现意外问题。
  - 协调和沟通：与工作负载负责人合作确认要停用的资源
  - 记录元数据并创建备份：如果生产环境中的资源需要或它们是关键资源，请记录元数据（例如公有 IP、区域、可用区、VPC、子网和安全组），并创建备份（例如 Amazon Elastic Block Store 快照或取得 AMI、密钥导出和证书导出）。
  - 验证基础设施即代码：确定资源是使用 AWS CloudFormation、Terraform、AWS Cloud Development Kit (AWS CDK) 还是任何其他基础设施即代码部署工具部署的，以便在必要时可以重新部署它们。
  - 阻止访问：在一段时间内应用限制性控制，以防止在确定是否需要资源时使用资源。验证资源环境是否可以在需要时恢复到其原始状态。
  - 遵循内部停用流程：遵循组织的管理任务和停用流程，例如从组织域中删除资源、删除 DNS 记录以及从配置管理工具、监控工具、自动化工具和安全工具中删除资源。

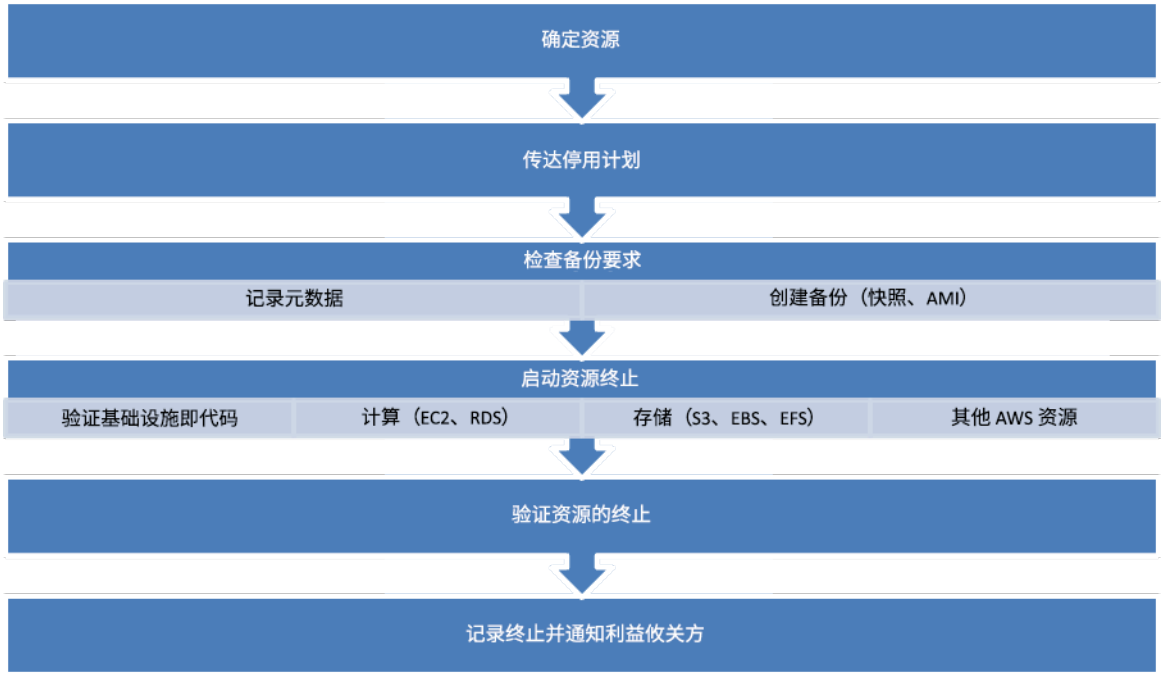
  如果资源是 Amazon EC2 实例，请参阅以下列表。有关更多详细信息，请参阅《如何删除或终止我的 Amazon EC2 资源？》

  - 停止或终止您的所有 Amazon EC2 实例和负载均衡器。Amazon EC2 实例终止后可短时间内在控制台中看到。您不需要为任何未处于运行状态的实例付费
  - 删除 Auto Scaling 基础设施。
  - 释放所有专属主机。
  - 删除所有 Amazon EBS 卷和 Amazon EBS 快照。
  - 释放所有弹性 IP 地址。

- 取消注册所有亚马逊机器映像（AMI）。
- 终止所有 AWS Elastic Beanstalk 环境。

如果资源是 Amazon S3 Glacier 存储中的对象，并且您在满足最短存储持续时间之前删除了存档，将按比例收取提前删除费用。Amazon S3 Glacier 的最短存储持续时间取决于所使用的存储类。有关每种存储类的最短存储持续时间摘要，请参阅跨 S3 存储类的性能。有关如何计算提前删除费用的详细信息，请参阅 Amazon S3 定价。

下面的简单停用过程流程图概述了停用步骤。在停用资源之前，请确认组织未使用已确定要停用的资源。



资源停用流程。

资源

相关文档：

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS CloudTrail

相关视频：

- 删除 CloudFormation 堆栈，但保留某些资源
- Find out which user launched Amazon EC2 instance

相关示例：

- 删除或终止 Amazon EC2 资源
- 了解哪个用户启动了 Amazon EC2 实例

COST04-BP03 停用资源

停用由定期审核或使用情况发生变化等事件启动的资源。停用通常定期执行，可以手动停用，也可以自动停用。

在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

搜索未使用资源的频率和工作量应反映潜在的节省额，因此，与成本较高的账户相比，对成本较低的账户进行分析的频率应该更低。搜索和停用事件可由工作负载中的状态更改启动，比如产品生命周期结束或产品被更换。搜索和停用事件也可由外部事件启动，如市场条件发生变化或产品终止。

实施步骤

- 停用资源：这是不再需要的 AWS 资源的折旧阶段或许可协议的终止。在进入处置阶段并停用资源之前完成所有最终检查，以防止任何意外中断，例如拍摄快照或备份。使用停用流程，停用每项被确定为未使用的资源。

资源

相关文档：

- AWS Auto Scaling
- AWS Trusted Advisor

COST04-BP04 自动停用资源

设计您的工作负载，使其在您确定并停用非关键资源、不需要的资源或使用率低的资源时妥善处理资源的终止。

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

使用自动化技术可以减少或消除停用流程中的相关成本。将工作负载设计为执行自动化停用，将减少工作负载在其整个生命周期内的总成本。您可以使用 Amazon EC2 Auto Scaling 或 Application Auto Scaling 来执行停用流程。还可以使用 API 或 SDK 来实施自定义代码，以自动停用工作负载资源。

现代应用程序是以“无服务器为先”理念构建的，这种策略优先考虑采用无服务器服务。AWS 为堆栈的所有三个层开发了无服务器服务：计算、集成和数据存储。使用无服务器架构将允许您在低流量期间通过自动纵向扩展和缩减来节省成本。

实施步骤

- 实施 Amazon EC2 Auto Scaling 或 Application Auto Scaling：对于支持的资源，请使用 Amazon EC2 Auto Scaling 或 Application Auto Scaling 来配置它们。这些服务可以帮助您在使用 AWS 服务时优化使用率和成本效率。当需求下降时，这些服务将自动删除任何多余的资源容量，以避免超支。
- 将 CloudWatch 配置为终止实例：可以将实例配置为使用 CloudWatch 警报终止。使用停用流程的指标，实施包含 Amazon Elastic Compute Cloud 操作的警报。在推出之前，在非生产环境中验证操作。
- 在工作负载中实现代码：可以使用 AWS SDK 或 AWS CLI 停用工作负载资源。在与 AWS 集成的应用程序中实现代码，并终止或删除不再使用的资源。
- 使用无服务器服务：优先在 AWS 上构建无服务器架构和事件驱动型架构，以构建和运行应用程序。AWS 提供多种无服务器技术服务，这些服务本质上可以自动优化资源利用率且具有自动停用功能（横向缩减和横向扩展）。通过无服务器应用程序，可自动优化资源利用率，您无需为过度预置付费。

资源

相关文档：

- Amazon EC2 Auto Scaling
- Getting Started with Amazon EC2 Auto Scaling
- Application Auto Scaling
- AWS Trusted Advisor
- AWS 无服务器

- 创建警报以停止、终止、重启或恢复实例
- 在 Amazon CloudWatch 警报中添加终止操作

相关示例：

- Scheduling automatic deletion of AWS CloudFormation stacks

COST04-BP05 执行数据留存策略

在支持的资源上定义数据留存策略，以根据组织要求处理对象的删除事宜。确定并删除非必要的或不再需要的孤立资源和对象。

在未建立这种最佳实践的情况下暴露的风险等级：中

使用数据留存策略和生命周期策略，降低停用过程的相关成本以及已确定资源的存储成本。定义数据留存策略和生命周期策略来执行自动存储类迁移和删除，将会降低其生命周期内的总体存储成本。可以使用 Amazon Data Lifecycle Manager 来自动创建和删除 Amazon Elastic Block Store 快照及基于 Amazon EBS 的亚马逊机器映像（AMI），并使用 Amazon S3 Intelligent-Tiering 或 Amazon S3 生命周期配置来管理 Amazon S3 对象的生命周期。还可以使用 API 或 SDK 实现自定义代码，为要自动删除的对象创建生命周期策略和策略规则。

实施步骤

- 使用 Amazon Data Lifecycle Manager：使用 Amazon Data Lifecycle Manager 上的生命周期策略，自动删除 Amazon EBS 快照和 Amazon EBS-backed AMI。
- 在存储桶上设置生命周期配置：根据业务要求，使用存储桶上的 Amazon S3 生命周期配置，定义 Amazon S3 在对象生命周期中要采取的操作以及对象生命周期结束时的删除操作。

资源

相关文档：

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- 如何在 Amazon S3 存储桶上设置生命周期配置

相关视频：

- Automate Amazon EBS Snapshots with Amazon Data Lifecycle Manager
- 使用生命周期配置规则清空 Amazon S3 存储桶

相关示例：

- 使用生命周期配置规则清空 Amazon S3 存储桶

具有成本效益的资源

问题

- COST 5. 您在选择服务时如何评估成本？
- COST 6. 在选择资源类型、规模和数量时，如何实现成本目标？
- COST 7. 如何使用定价模式来降低成本？
- COST 8. 如何规划数据传输费用？

COST 5. 您在选择服务时如何评估成本？

Amazon EC2、Amazon EBS 和 Amazon S3 属于基础 AWS 服务。托管服务（如 Amazon RDS 和 Amazon DynamoDB）属于更高级别或应用程序级别的 AWS 服务。通过选择适当的基础服务和托管服务，您可以优化此工作负载，从而降低成本。例如，使用托管服务，您可以节省或消除大部分管理和运营开销，让您有精力从事应用程序和业务相关活动。

最佳实践

- COST05-BP01 确定组织对成本的要求
- COST05-BP02 分析工作负载的所有组件
- COST05-BP03 对每个组件进行全面分析
- COST05-BP04 选择具有成本效益许可的软件
- COST05-BP05 选择此工作负载的组件，以便根据组织的优先事项优化成本
- COST05-BP06 对不同时间的不同使用情况执行成本分析

COST05-BP01 确定组织对成本的要求

与团队成员合作，为此工作负载确定成本优化与其他支柱（例如性能和可靠性）之间的平衡。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在大多数组织中，信息技术（IT）部门由多个小型团队组成，每个小团队都有自己的议程和重点领域，这反映了其团队成员的专长和技能。您需要了解组织的总体目标、优先事项、具体目标以及每个部门或项目如何为这些总体目标做出贡献。对所有基本资源（包括人员、设备、技术、材料和外部服务）进行分类，是实现组织目标和全面预算规划的重要一环。采用这种系统化的方法来确定和了解成本，对于组织制定切合实际且稳健的成本计划至关重要。

在为工作负载选择服务时，了解组织的优先要务至关重要。在成本优化和 AWS Well-Architected Framework 的其他支柱（例如性能和可靠性）之间取得平衡。这一过程应系统性地定期进行，以反映组织目标、市场条件和运营动态的变化。完全成本优化的工作负载是最符合组织要求的解决方案，但不一定是成本最低的。与组织内的所有团队（例如产品、业务、技术和财务）会面以收集信息。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作出明智的决策。

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，而不是迁移到针对数据类型优化的数据库和更新应用程序。

实施步骤

- 确定组织对成本的要求：与组织中的团队成员会面，这些成员包括产品管理、应用程序负责人、开发和运营团队、管理和财务人员。对此工作负载及其组件的 Well-Architected 支柱进行优先级排序，输出应该是一个按顺序排列的支柱列表。您还可以为每个支柱添加一个权重，指示该支柱体现的额外关注程度，或者两个支柱之间的关注点的相似程度。
- 解决技术债务并记录下来：在工作负载审核期间，解决技术债务。记录积压工作项以便将来重新审视工作负载，目标是重构或重新构建以进一步优化工作负载。必须向其他利益相关方明确说明所做的权衡取舍。

资源

相关最佳实践：

- REL11-BP07 构建产品以满足可用性目标和正常运行时间服务水平协议（SLA）
- OPS01-BP06 评估权衡

相关文档：

- AWS 总拥有成本（TCO）计算器
- Amazon S3 存储类
- 云产品

COST05-BP02 分析工作负载的所有组件

确认已分析工作负载的每个组件，无论当前大小或当前成本如何。审核工作应该体现可能带来的好处，例如当前成本和预期成本。

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

旨在为组织提供业务价值的工作负载组件可能包含各种服务。对于每个组件，组织可以选择特定的 AWS Cloud 服务来满足业务需求。对这些服务的熟悉程度，或以往的使用经验等因素，可能会影响这种选择。

按照 COST05-BP01 确定组织对成本的要求中的说明确定组织的要求后，对工作负载中的所有组件进行全面分析。根据当前和预计的成本和规模，分析每个组件。结合工作负载在其生命周期内可能实现的节省来考虑分析成本。分析此工作负载的所有组件所花费的精力，应与优化该特定组件预期能产生的潜在节省或改进相匹配。例如，如果拟议资源的成本为每月 10 美元，在预测的负载下不会超过每月 15 美元，那么为了将成本降低 50%（每月 5 美元）而耗费的一天时间，其价值可能已经超过系统使用寿命内的潜在收益。使用更快、更高效的基于数据的预估，可为该组件带来出色的总体结果。

工作负载可能会随时间变化，如果工作负载架构或使用情况发生变化，原本合适的服务集可能不再是最优之选。为甄选服务进行分析时，必须考虑工作负载当前和未来的状态以及使用水平。为将来的工作负载状态或使用情况实施服务，可以减少或消除未来进行更改所需的工作量，从而降低总体成本。例如，最初使用 EMR Serverless 可能是合适的选择。但是，随着该服务使用情况的增加，过渡到 EC2 上的 EMR 可以降低该工作负载组件的成本。

AWS Cost Explorer 和 AWS 成本和使用情况报告（CUR）可以分析概念验证（PoC）或运行环境的成本。您还可以使用 AWS 定价计算器 估算工作负载成本。

编写一个工作流程以供技术团队用来审核其工作负载。此工作流程不仅要简单，还要涵盖所有必要步骤，以确保团队了解工作负载的每个组件及其定价。之后，组织可以遵循此工作流程，并根据每个团队的具体需求自定义此工作流程。

1. 列出用于工作负载的每项服务：这是一个很好的起点。确定当前使用的所有服务以及成本的来源。

2. 了解这些服务的定价方式：了解每项服务的定价模式。不同的 AWS 服务具有不同的定价模式，具体取决于使用量、数据传输和功能特定的定价等因素。
3. 重点关注那些工作负载成本不符合预期且与您的预期使用情况和业务结果不一致的服务：找出成本与使用 AWS Cost Explorer 或 AWS 成本和使用情况报告 的价值或使用情况不成比例时的异常值或服务。将成本与业务成果关联起来以确定优化工作的优先顺序，这一点很重要。
4. 使用 AWS Cost Explorer、CloudWatch Logs、VPC 流日志和 Amazon S3 Storage Lens 存储统计管理工具了解这些高成本的根本原因：这些工具有助于诊断高成本。每项服务都提供一种不同的分析功能来查看和分析使用情况和成本。例如，Cost Explorer 有助于确定总体成本趋势，CloudWatch Logs 可提供运营见解，VPC 流日志可显示 IP 流量，Amazon S3 Storage Lens 存储统计管理工具有助于存储分析。
5. 使用 AWS Budgets 为服务或账户设置特定金额的预算：设置预算是一种主动管理成本的方法。使用 AWS Budgets 可以设置自定义预算阈值，并在成本超过这些阈值时接收警报。
6. 配置 Amazon CloudWatch 警报以发送账单和使用情况警报：设置成本和使用情况指标的监控和警报。CloudWatch 警报可在达到特定阈值时通知您，从而缩短干预响应时间。

对所有工作负载组件（无论其当前属性如何）进行战略审核，从而推动实现显著的改进和财务节省。应仔细考量审核过程中投入的精力，同时认真考虑可能实现的潜在优势。

实施步骤

- 列出工作负载组件：构建工作负载组件的列表。使用此列表来验证是否已分析每个组件。应基于工作负载对企业优先事项的重要程度，来投入相应的工作量。按功能将资源分组可以提高效率（例如，将多个数据库合并为生产数据库存储）。
- 确定组件列表的优先级：选择组件列表并按工作量顺序对其进行优先级排序。通常按照组件的成本从最昂贵到最便宜的顺序排列，或者按照组件对组织优先事项的重要程度排列。
- 执行分析：对于列表中的每个组件，检查可用的选项和服务，然后选择最符合组织优先事项的选项。

资源

相关文档：

- AWS 定价计算器
- AWS Cost Explorer
- Amazon S3 存储类
- AWS Cloud 产品

相关视频：

- AWS 成本优化系列：CloudWatch

COST05-BP03 对每个组件进行全面分析

分析组织为每个组件付出的总体成本。通过考虑运营和管理成本（尤其是在使用云提供商提供的托管服务时）来计算总拥有成本。审核工作量应该体现可能带来的好处（例如分析时间与组件成本成正比）。

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

利用节省下来的时间，您的团队将能够专注于解决技术债务、创新和增值功能，并打造企业的竞争优势。例如，您可能需要尽快将数据库从本地环境直接迁移（也称为重新托管）到云，然后再进行优化。值得探索的是，通过使用 AWS 上可消除或减少许可成本的托管服务，您可以节省多少成本。AWS 上的托管服务消除了维护服务的运营和管理负担（例如修补或更新操作系统），让您可以专注于创新和业务。

由于托管服务在云级运行，它们可以提供更低的单位事务或服务成本。您可以在不改变应用程序核心架构的情况下进行潜在优化，以获得一些切实的优势。例如，您可能希望通过迁移到 Amazon Relational Database Service（Amazon RDS）等数据库即服务平台或将应用程序迁移到完全托管式平台（例如 AWS Elastic Beanstalk），来减少管理数据库实例所花费的时间。

通常，可以设置托管服务的部分属性，以确保容量足够。您必须设置和监控这些属性，以便最大限度地减少多余容量，并最大限度地提高性能。您可以使用 AWS Management Console 或 AWS API 和 SDK 来修改 AWS Managed Services 的属性，使资源需求匹配不断变化的要求。例如，可以增加或减少 Amazon EMR 集群（或 Amazon Redshift 集群）上的节点数量，以横向扩展或缩减集群。

您还可以在 AWS 资源上打包多个实例，实现更高密度的使用情况。例如，您可以在单个 Amazon Relational Database Service（Amazon RDS）数据库实例上预置多个小型数据库。随着使用量的增长，您可以使用快照和还原过程，将其中一个数据库迁移到专用 Amazon RDS 数据库实例。

在托管服务上预置工作负载时，必须了解调整服务容量的要求。这些要求通常是时间、工作量和对正常工作负载运营的任何影响。预置的资源必须留出时间来进行任何更改，并预置所需开销以允许这样做。通过使用与系统和监控工具（如 Amazon CloudWatch）集成的 API 和软件开发工具包，可以将修改服务所需的持续工作量减少至接近零。

Amazon RDS、Amazon Redshift 和 Amazon ElastiCache 提供托管式数据库服务。Amazon Athena、Amazon EMR 和 Amazon OpenSearch Service 提供托管式分析服务。

AMS 是代表企业客户和合作伙伴运营 AWS 基础设施的服务。该服务提供了一个安全且合规的环境，您可以将工作负载部署到其中。AMS 使用具有自动化功能的企业云运营模型，让您可以满足组织要求，更快地迁移到云中并降低持续的管理成本。

实施步骤

- 执行全面的分析：使用组件列表，从最高优先级到最低优先级遍历每个组件。对于优先级较高且成本较高的组件，执行额外分析并评测所有可用选项及其长期影响。对于优先级较低的组件，评测使用情况的变化是否会更改组件的优先级，然后以适当的工作量进行分析。
- 比较托管资源和非托管资源：考虑您管理的资源的运营成本，并将其与 AWS 托管资源进行比较。例如，审查在 Amazon EC2 实例上运行的数据库，并与 Amazon RDS 选项（AWS 托管服务）进行比较，或将 Amazon EMR 与在 Amazon EC2 上运行 Apache Spark 进行比较。当从自我管理的工作负载迁移到 AWS 完全托管的工作负载时，请仔细研究您的选择。需要考虑的三个最重要的因素是您要使用的托管服务的类型、要用于迁移数据的流程以及了解 AWS 责任共担模式。

资源

相关文档：

- AWS 总拥有成本（TCO）计算器
- Amazon S3 存储类
- AWS Cloud 产品
- AWS 责任共担模式

相关视频：

- 为什么要迁移到托管数据库？
- 什么是 Amazon EMR，如何才能使用它来处理数据？

相关示例：

- 为什么要迁移到托管数据库？
- 使用 AWS DMS 将相同 SQL Server 数据库中的数据整合到单个 Amazon RDS for SQL Server 数据库中
- 将数据大规模传输到 Amazon Managed Streaming for Apache Kafka（Amazon MSK）
- 将 ASP.NET Web 应用程序迁移到 AWS Elastic Beanstalk

COST05-BP04 选择具有成本效益许可的软件

开源软件无需软件许可成本，从而大大节省了工作负载的成本。如果需要许可软件，应避免使用绑定到任意属性（如 CPU）的许可，而应使用绑定到输出或结果的许可。这些许可的成本与所提供的效益更为相当。

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

开源起源于软件开发，表示软件符合某些自由发布的标准。任何人都可以检查、修改和增强开源软件的源代码。根据业务要求、工程师技能、预测的使用情况或其他技术依赖关系，组织可以考虑在 AWS 上使用开源软件来最大限度地降低许可成本。换言之，使用开源软件可以降低软件许可成本。随着工作负载规模的扩展，这可能对工作负载成本产生重大影响。

将许可软件能够带来的好处与总成本进行比较，以优化工作负载。对许可的任何更改以及更改对工作负载成本的影响建模。如果供应商更改了数据库许可的成本，请调查这会如何影响工作负载的整体效率。考虑供应商的历史定价公告，了解其产品中的许可更改趋势。许可成本也可以独立于吞吐量或使用情况进行扩缩，例如按硬件扩缩的许可（CPU 绑定许可）。应避免使用这些许可，因为成本会迅速增加，而且无法取得相应的结果。

例如，与在 Windows 上运行 Amazon EC2 实例相比，使用 Linux 操作系统在 us-east-1 中运行另一个 Amazon EC2 实例可以将成本降低大约 45%。

AWS 定价计算器 提供了一种全面的方法来比较具有不同许可选项的各种资源的成本，例如 Amazon RDS 实例和不同的数据库引擎。此外，AWS Cost Explorer 为现有工作负载（尤其是附带不同许可的工作负载）的成本提供了宝贵的视角。对于许可管理，AWS License Manager 提供了一种简化的方法来监督和处理软件许可。客户可以在 AWS Cloud 中部署和运行他们首选的开源软件。

实施步骤

- 分析许可选项：查看可用软件的许可条款。查看具有所需功能的开源版本，以及许可软件提供的效益是否大于成本。优惠条款可确保软件成本与所提供的效益相符。
- 分析软件提供商：查看供应商的任何历史定价或许可更改。了解与成果不符的任何更改，例如在特定供应商硬件或平台上运行的惩罚性条款。此外，还要了解他们如何执行审核和处罚。

资源

相关文档：

- Open Source at AWS

- AWS 总拥有成本（TCO）计算器
- Amazon S3 存储类
- 云产品

相关示例：

- 开源博客
- AWS 开源博客
- 优化与许可评测

COST05-BP05 选择此工作负载的组件，以便根据组织的优先事项优化成本

为工作负载选择所有组件时考虑成本因素。这包括使用应用程序级服务和托管服务或无服务器、容器或事件驱动型架构，以降低总体成本。使用开源软件、没有许可费用的软件或替代方案来尽可能减少许可成本，从而减少开支。

在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

在选择所有组件时考虑服务和选项的成本。这包括使用应用程序级服务和托管服务（例如 Amazon Relational Database Service（Amazon RDS）、Amazon DynamoDB、Amazon Simple Notification Service（Amazon SNS）和 Amazon Simple Email Service（Amazon SES）来降低总体组织成本。

使用无服务器和容器进行计算，例如将 AWS Lambda 和 Amazon Simple Storage Service（Amazon S3）用于静态网站。尽可能将应用程序容器化，并使用 AWS 托管容器服务，例如 Amazon Elastic Container Service（Amazon ECS）和 Amazon Elastic Kubernetes Service（Amazon EKS）。

使用开源软件或没有许可费用的软件，尽可能减少许可成本：例如，对计算工作负载使用 Amazon Linux 或将数据库迁移到 Amazon Aurora。

可以使用无服务器服务或应用程序级服务，例如 Lambda、Amazon Simple Queue Service（Amazon SQS）、Amazon SNS 和 Amazon SES。这些服务消除了管理资源的需要，并提供代码执行、服务排队和消息传递功能。另一个好处是，这些服务可以根据使用情况横向缩减性能和成本，从而实现有效的成本分配和归属。

对于无服务器服务，也可以使用事件驱动型架构。事件驱动型架构是推送式的，所以当事件在路由器中出现时，一切都按需发生。这样，您就不会为检查事件的连续轮询而付费。这意味着更少的网络带宽消耗、更低的 CPU 利用率、更少的闲置实例集容量，以及更少的 SSL/TLS 握手次数。

有关无服务器的更多信息，请参阅《Well-Architected 无服务器应用程序剖析白皮书》。

实施步骤

- 选择每项服务以优化成本：使用经过优先级排序的列表和分析，选择最符合组织优先事项的每个选项。与其增加容量来满足需求，不如考虑其他可以提供更高性价比的选项。例如，您需要审查数据库在 AWS 上的预期流量，并考虑增加实例大小或使用 Amazon ElastiCache 服务（Redis 或 Memcached），为数据库提供缓存机制。
- 评估事件驱动型架构：通过使用无服务器架构，您还可以为基于微服务的分布式应用程序构建事件驱动型架构，这有助于您构建可扩展、有韧性、敏捷且具有成本效益的解决方案。

资源

相关文档：

- AWS 总拥有成本（TCO）计算器
- AWS 无服务器
- 什么是事件驱动型架构
- Amazon S3 存储类
- 云产品
- Amazon ElastiCache（Redis OSS）

相关示例：

- Getting started with event-driven architecture
- 事件驱动型架构
- How Statsig runs 100x more cost-effectively using Amazon ElastiCache (Redis OSS)
- Best practices for working with AWS Lambda functions

COST05-BP06 对不同时间的不同使用情况执行成本分析

工作负载可能会随时间而变化。某些服务或功能在不同的使用水平下更具成本效益。通过根据预期使用情况对一段时间内的每个组件执行分析，工作负载可在其生命周期内保持成本效益。

在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

随着 AWS 发布新的服务和功能，适用于工作负载的最佳服务可能会发生变化。所需的工作量应体现可能带来的好处。工作负载审核频率取决于组织要求。如果工作负载的成本很高，则尽早实施新服务可最大限度地节省成本，因此提高审核频率可能是有利的。审核的另一个启动因素是使用模式发生变化。使用情况发生重大变化可能表明备用服务更加理想。

如果您需要将数据移入 AWS Cloud，可以选择 AWS 提供的任何种类的服务和合作伙伴工具，帮助您迁移数据集，无论它们是文件、数据库、机器映像、数据块卷，还是磁带备份。例如，要将大量数据移入和移出 AWS，或者要在边缘处理数据，可以使用 AWS 专用设备之一，经济高效地离线移动 PB 级数据。再例如，对于更高的数据传输率，直接连接服务可能比 VPN 更便宜，而 VPN 可以为业务提供所需的稳定一致的连接。

根据一段时间内针对不同使用情况的成本分析，审查您的扩缩活动。分析结果，看看是否可以调整扩缩策略，以增加具有多种实例类型和购买选项的实例。审查设置，看看是否可以减小最小值，用较小的实例集大小满足用户请求，并增加更多资源来满足预期的高需求。

通过与组织中的利益相关方讨论，对不同使用情况执行成本分析，并使用 AWS Cost Explorer 预测功能来预测服务变更的潜在影响。使用 AWS Budgets、CloudWatch 计费警报和 AWS Cost Anomaly Detection 监控使用水平启动因素，以尽早确定和实施最具成本效益的服务。

实施步骤

- 定义预计使用模式：与组织中的相关人员（例如营销和产品负责人）合作，记录哪些预期和预计使用模式适用于工作负载。与业务利益相关方讨论历史和预测的成本及使用量的增加，并确保增加幅度与业务要求一致。确定预计会有更多用户使用您的 AWS 资源的日历日、周或月，这表明您应该增加现有资源的容量，或采用额外的服务来降低成本和提高性能。
- 对预测的使用情况执行成本分析：使用定义的使用模式，在每个点执行分析。分析工作量应该体现潜在的成果，例如，如果使用情况变化很大，应执行彻底分析，以验证任何成本和变化。换句话说，当成本增加时，企业的使用量也应该增加。

资源

相关文档：

- AWS 总拥有成本（TCO）计算器
- Amazon S3 存储类
- 云产品

- Amazon EC2 Auto Scaling
- 云数据迁移
- AWS Snow Family

相关视频：

- AWS OpsHub for Snow Family

COST 6. 在选择资源类型、规模和数量时，如何实现成本目标？

确保选择适合当前任务的资源规模和资源数量。选择最经济实惠的资源类型、规模和数量可以尽可能减少浪费。

最佳实践

- COST06-BP01 执行成本建模
- COST06-BP02 根据数据选择资源类型、规模和数量
- COST06-BP03 根据指标自动选择资源类型、规模和数量
- COST06-BP04 考虑使用共享资源

COST06-BP01 执行成本建模

确定组织要求（如业务需求和现有承诺），并对工作负载及其每个组件进行成本建模（总体成本）。对不同预计负载下的工作负载执行基准测试活动，并比较成本。建模工作应该体现可能带来的好处，例如花费的时间与组件成本成正比。

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

对工作负载及其每个组件执行成本建模，以了解资源之间的平衡，并在给定的具体性能水平下，确定工作负载中各项资源的适当规模。在评估计划的工作负载部署的价值实现结果时，了解成本考虑因素可为组织的业务案例和决策过程提供依据。

对不同预计负载下的工作负载执行基准测试活动，并比较成本。建模工作量应该体现可能带来的好处，例如花费的时间与组件成本或预计可节省的成本成正比。有关最佳实践，请参阅《AWS Well-Architected Framework 的性能效率支柱》的“审核”部分。

例如，要为由计算资源组成的工作负载创建成本模型，AWS Compute Optimizer 可以帮助实现对正在运行的工作负载的成本建模。它根据历史使用情况为计算资源提供合理调整大小的建议。确保将 CloudWatch 代理部署到 Amazon EC2 实例，以收集内存指标，这会通过 AWS Compute Optimizer 内更准确的建议为您提供帮助。这是计算资源的理想数据来源，因为它是一项免费服务，并且使用机器学习根据风险等级提出多项建议。

您可以使用多项服务将自定义日志作为数据来源，用于合理调整其他服务和工作负载组件（例如 AWS Trusted Advisor、Amazon CloudWatch 和 Amazon CloudWatch Logs）的操作的大小。AWS Trusted Advisor 会检查资源并标记利用率低的资源，这可以帮助您合理调整资源大小并创建成本模型。

以下是成本建模数据和指标的建议：

- 监控必须准确反映用户体验。为时间段选择正确的粒度，并仔细选择最大值或第 99 个百分位值而不是平均值。
- 为覆盖任何工作负载周期所需的分析时间段选择正确的粒度。例如，如果执行为期两周的分析，您可能会忽略高利用率的月度周期，这可能导致预置不足。
- 通过考虑现有的承诺、为其他工作负载选择的定价模式，以及更快地创新和专注于核心业务价值的能力，为计划的工作负载选择合适的 AWS 服务。

实施步骤

- 执行资源的成本建模：将工作负载或概念验证部署到具有特定资源类型和规模的单独账户，然后执行测试。使用测试数据运行工作负载，并记录输出结果以及测试运行时段的成本数据。然后，重新部署工作负载或更改资源类型和规模，并再次运行测试。在创建成本模型时，考虑您可能与这些资源一起使用的任何产品的许可费用，以及用于部署和管理这些资源的估计运营（劳动力或工程师）成本。考虑对一个时间段（每小时、每天、每月、每年或三年）进行成本建模。

资源

相关文档：

- AWS Auto Scaling
- 确定合理调整大小的机会
- Amazon CloudWatch 功能
- 成本优化：合理调整 Amazon EC2 的大小
- AWS Compute Optimizer

- AWS 定价计算器

相关示例：

- 执行数据驱动型成本建模
- 估算计划的 AWS 资源配置的成本
- 选择合适的 AWS 工具

COST06-BP02 根据数据选择资源类型、规模和数量

根据工作负载和资源特征的相关数据选择资源规模或类型，例如计算、内存、吞吐量或写入密集型资源。选择的依据通常是使用工作负载的上一个版本（本地版本）、文档或关于工作负载的其他信息源。

在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 EC2 提供许多实例类型，它们具有不同的 CPU、内存、存储和联网容量级别，适合不同的应用场景。这些实例类型具有不同的 CPU、内存、存储和联网容量组合，让您在为项目选择正确的资源组合时有更多的备选方案。每种实例类型都有多种大小，因此您可以根据工作负载的需求调整资源。要确定您需要哪种实例类型，请根据您计划在实例上运行的应用程序或软件，收集相关的系统要求详细信息。这些详细信息应包括以下各项：

- 操作系统
- CPU 核心数量
- GPU 核心数
- 系统内存（RAM）容量
- 存储类型和空间
- 网络带宽要求

确定计算要求的目的以及需要哪个实例，然后探索各种 Amazon EC2 实例系列。Amazon 提供以下实例类型系列：

- 通用
- 计算优化
- 内存优化

- 存储优化
- 加速计算
- HPC 优化型

要更深入地了解特定 Amazon EC2 实例系列可以实现的具体目的和应用场景，请参阅 AWS 实例类型。

想要选择最能满足需求的特定实例系列和实例类型，系统要求的收集至关重要。实例类型名称由系列名称和实例大小组成。例如，t2.micro 实例属于 T2 系列，而且是微型实例。

根据工作负载和资源特征（例如计算、内存、吞吐量或写入密集型）选择资源规模或类型。选择的依据通常是使用成本建模、工作负载的上一个版本（例如本地版本）、文档或关于工作负载的其他信息源（白皮书或发布的解决方案）。使用 AWS 定价计算器或成本管理工具有助于就实例类型、规模和配置作出明智的决策。

实施步骤

- 根据数据选择资源：使用成本建模数据选择预期的工作负载使用水平，然后选择指定的资源类型和规模。根据成本建模数据，确定虚拟 CPU 的数量、总内存（GiB）、本地实例存储卷（GB）、Amazon EBS 卷和网络性能级别，同时考虑实例所需的数据传输速率。务必根据详细的分析和准确的数据做出选择，这样不仅能够优化性能，还能有效管理成本。

资源

相关文档：

- AWS 实例类型
- AWS Auto Scaling
- Amazon CloudWatch 功能
- 成本优化：合理调整 EC2 的大小

相关视频：

- Selecting the right Amazon EC2 instance for your workloads
- Right size your service

相关示例：

- It just got easier to discover and compare Amazon EC2 instance types

COST06-BP03 根据指标自动选择资源类型、规模和数量

使用当前运行的工作负载的指标选择正确的规模和类型，从而优化成本。为计算、存储、数据和网络服务适当地预置吞吐量、规模和存储。这可以通过自动扩缩等反馈环路进行，也可以在工作负载中使用自定义代码来实现。

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

在工作负载中创建一个反馈环路，此循环使用正在运行的工作负载中的活动指标来对该工作负载进行更改。您可以使用托管服务（例如 AWS Auto Scaling），您配置该服务来为您执行合理调整大小的操作。AWS 还提供 API、SDK 以及各类功能，让您可以毫不费力地修改资源。您可以对工作负载进行编程以停止和启动 Amazon EC2 实例，从而允许更改实例大小或实例类型。这带来双重好处：既合理调整了大小，又几乎消除了进行更改所需的全部运营成本。

某些 AWS 服务内置了自动类型或大小选项，如 Amazon Simple Storage Service Intelligent-Tiering。Amazon S3 Intelligent-Tiering 会根据使用模式，自动在两个访问层之间移动数据：频繁访问和不频繁访问。

实施步骤

- 通过配置工作负载指标来提高可观测性：捕获工作负载的关键指标。这些指标指明了客户体验（例如工作负载输出），并适应资源类型和规模之间的差异（例如 CPU 和内存使用情况）。对于计算资源，请分析性能数据以合理调整 Amazon EC2 实例大小。确定空闲实例和未充分利用的实例。要查找的关键指标是 CPU 利用率和内存利用率（例如，在 90% 的时间内为 40% 的 CPU 利用率，如 Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled 中所述）。确定四周内最大 CPU 使用率和内存利用率低于 40% 的实例。这些是大小适当能够降低成本的实例。对于诸如 Amazon S3 之类的存储资源，可以使用 Amazon S3 Storage Lens 存储统计管理工具，它允许您在存储桶级别查看不同类别的 28 个指标，并在控制面板中默认查看 14 天的历史数据。可以按摘要和成本优化或事件筛选 Amazon S3 Storage Lens 存储统计管理工具控制面板，以分析特定指标。
- 查看合理调整大小建议：使用 AWS Compute Optimizer 中的合理调整大小建议和成本管理控制台中的 Amazon EC2 合理调整大小工具，或者查看 AWS Trusted Advisor 如何合理调整资源大小以调整工作负载。无论是 Amazon EC2 实例、AWS 存储类还是 Amazon RDS 实例类型，都必须使用正确的工具来调整不同资源的大小，并遵循合理调整大小指南。对于存储资源，可以使用 Amazon S3 Storage Lens 存储统计管理工具，它可以让您了解对象存储使用情况、活动趋势，并提出可行的建

议，以优化成本并应用数据保护最佳实践。使用 Amazon S3 Storage Lens 存储统计管理工具通过对整个组织指标的分析得出的上下文建议，您可以立即采取措施来优化存储。

- 根据指标自动选择资源类型和大小：使用工作负载指标，手动或自动选择工作负载资源。对于计算资源，配置 AWS Auto Scaling 或在应用程序中实现代码，可以减少频繁更改所需的工作量，而且实施更改的速度可能比手动操作更快。您可以启动并自动扩展单个 Auto Scaling 组中的一组按需实例和竞价型实例。除了享受使用竞价型实例的折扣外，您还可以使用预留实例或 Savings Plan 获得常规按需实例定价的折扣费率。将所有这些因素相结合，可帮助您优化 Amazon EC2 实例的成本节约，并确定应用程序所需的规模和性能。还可以在自动扩缩组（ASG）中使用基于属性的实例类型选择（ABS）策略，该策略允许您将实例要求表示为一组属性，例如 vCPU、内存和存储。您可以在发布新一代实例类型时自动使用它们，并通过 Amazon EC2 竞价型实例获得更广泛的容量。Amazon EC2 Fleet 和 Amazon EC2 Auto Scaling 会选择并启动符合指定属性的实例，无需手动选择实例类型。对于存储资源，可以使用 Amazon S3 Intelligent Tiering 和 Amazon EFS 不频繁访问功能，它们允许您自动选择存储类，以便在数据访问模式变更时自动节省存储成本，而不会影响性能或运营开销。

资源

相关文档：

- AWS Auto Scaling
- AWS 合理调整大小
- AWS Compute Optimizer
- Amazon CloudWatch 功能
- CloudWatch Getting Set Up
- CloudWatch Publishing Custom Metrics
- Getting Started with Amazon EC2 Auto Scaling
- Amazon S3 Storage Lens 存储统计管理工具
- Amazon S3 Intelligent-Tiering
- Amazon EFS 不频繁访问
- Launch an Amazon EC2 Instance Using the SDK

相关视频：

- Right Size Your Services

相关示例：

- 基于属性选择实例类型用于 Amazon EC2 Fleet 的自动扩缩
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive scaling with Amazon EC2 Auto Scaling
- 利用 Amazon S3 Storage Lens 存储统计管理工具优化成本并深入了解使用情况

COST06-BP04 考虑使用共享资源

对于已经在组织级别为多个业务部门部署的服务，可以考虑使用共享资源来提高利用率，并降低总拥有成本（TCO）。使用共享资源是一种经济实惠的选择，可以通过使用现有解决方案和/或共享组件，实现集中管理和成本优化。在账户边界内或专用账户中管理监控、备份和连接等常见功能。还可以通过实施标准化、减少重复和降低复杂性来降低成本。

在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

如果多个工作负载产生相同的功能，则使用现有解决方案和共享组件，改善管理和优化成本。考虑使用现有资源（尤其是共享资源），例如非生产数据库服务器或目录服务，通过遵循安全最佳实践和组织法规来降低云成本。为了更好地实现价值和提高效率，将成本分配（使用对账和扣款）给产生使用情况的相关业务领域至关重要。

对账是指将云成本分为可归属类别的报告，例如消费者、业务部门、总分类账账户或其他责任实体。对账的目标是向团队、业务部门或个人展示其使用的云资源的成本。

扣款是指根据适合特定财务管理流程的策略，将中央服务支出分配给成本单位。对于客户而言，扣款会将一个共享服务账户产生的成本，分别计入适用于客户报告流程的不同财务成本类别。通过建立扣款机制，您可以报告不同业务部门、产品和团队产生的成本。

工作负载可以分类为关键和非关键工作负载。根据此分类，使用常规配置的共享资源来处理不太关键的工作负载。为了进一步优化成本，请仅为关键工作负载预留专用服务器。跨多个账户共享或预置资源，对资源进行高效管理。即使在不同的开发、测试和生产环境中，安全共享也是可行的，不会破坏组织结构。

为了加强您的了解，并优化容器化应用程序的成本和使用情况，请使用拆分成本分配数据，采用这种做法，有助于您根据应用程序对共享计算和内存资源的使用情况，将成本分配给各个业务实体。拆分成本分配数据可帮助您在容器工作负载中实现任务级对账和扣款，这些工作负载在 Amazon Elastic Container Service（Amazon ECS）或 Amazon Elastic Kubernetes Service（Amazon EKS）上运行。

对于分布式架构，请构建共享服务 VPC，集中访问每个 VPC 中的工作负载所需的共享服务。这些共享服务可以包括目录服务或 VPC 端点等资源。为了减少管理开销和成本，请从中心位置共享资源，而不是在每个 VPC 中构建资源。

使用共享资源时，可以节省运营成本、大幅提高资源利用率并提高一致性。在多账户设计中，可以集中托管一些 AWS 服务，并在中心使用多个应用程序和账户访问这些服务，此举有助于节省成本。可以使用 AWS Resource Access Manager（AWS RAM）共享其它公用资源，如 VPC subnets and AWS Transit Gateway attachments、AWS Network Firewall 或 Amazon SageMaker AI pipelines。在多账户环境中，使用 AWS RAM 一次性创建资源并与其他账户共享。

组织应有效地标记共享成本，并确认其没有大量成本未标记或未分配。如果您未有效地分配共享成本，也没有人对共享成本管理负责，则共享云成本可能会急剧增加。您应当知道自己在资源、工作负载、团队或组织层面产生了哪些成本，因为这些信息有助于您进一步了解相较于实现的业务成果，在适当层面所实现的价值。最终，通过共享云基础设施，组织将实现成本节省。建议对共享云资源进行成本分配，以优化云支出。

实施步骤

- 评估现有资源：审核对您的工作负载使用类似服务的现有工作负载。根据工作负载的组件，如果业务逻辑或技术要求允许，考虑现有平台。
- 在 AWS RAM 中使用资源共享并进行相应的限制：使用 AWS RAM 与组织内的其他 AWS 账户共享资源。共享资源时，无需在多个账户中重复构建资源，这样可以大幅减少资源维护的运营负担。此过程还有助于您与账户中的角色和用户，以及与其他 AWS 账户 安全地共享已创建的资源。
- 标记资源：标记比较适合成本报告的资源，并将其在成本类别中归类。激活这些与成本相关的资源标签来进行成本分配，以实现 AWS 资源使用情况的可见性。专注于在成本和使用情况可见性方面创建适当的粒度级别，并通过成本分配报告和 KPI 跟踪来影响云使用行为。

资源

相关最佳实践：

- SEC03-BP08 在组织内安全地共享资源

相关文档：

- 什么是 AWS Resource Access Manager？
- AWS services that you can use with AWS Organizations
- Shareable AWS resources

- AWS Cost and Usage (CUR) Queries

相关视频：

- AWS Resource Access Manager - granular access control with managed permissions
- How to design your AWS cost allocation strategy
- AWS Cost Categories

相关示例：

- How-to chargeback shared services: An AWS Transit Gateway example
- How to build a chargeback/showback model for Savings Plans using the CUR
- Using VPC Sharing for a Cost-Effective Multi-Account Microservice Architecture
- Improve cost visibility of Amazon EKS with AWS Split Cost Allocation Data
- 利用 AWS 拆分成本分配数据提高 Amazon ECS 和 AWS Batch 的成本可见性

COST 7. 如何使用定价模式来降低成本？

使用最适合的资源定价模式可以尽可能减少支出。

最佳实践

- COST07-BP01 执行定价模式分析
- COST07-BP02 根据成本选择区域
- COST07-BP03 选择具有经济实惠条款的第三方协议
- COST07-BP04 针对此工作负载的所有组件实施定价模式
- COST07-BP05 在管理账户级别执行定价模式分析

COST07-BP01 执行定价模式分析

分析工作负载的每个组件。确定组件和资源是长时间运行（享受承诺折扣），还是短时间动态运行（采用竞价型或按需型实例定价）。使用成本管理工具中的建议对工作负载执行分析，并对这些建议应用业务规则以实现高回报。

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

AWS 有多种定价模式，您可以根据产品，以符合组织需求且最具成本效益的方式支付资源费用。与团队合作确定最适合的定价模式。定价模式通常包含多种选项的组合，根据供应情况来决定

按需型实例允许您只需要按小时或按秒（最低 60 秒）支付计算或数据库容量费用，具体取决于您运行的实例，无需做出长期承诺或预先付款。

节省计划是灵活的定价模式，支持以较低的 Amazon EC2、Lambda 和 AWS Fargate 使用价格换取一年或三年期的持续使用量承诺（以美元/小时为单位）。

竞价型实例是一种 Amazon EC2 定价机制，允许您以折扣小时费率（最高比按需价格低 90%）申请备用计算容量，无需预先承诺。

预留实例通过预付容量费用，允许您享受高达 75% 的折扣。有关更多详细信息，请参阅使用预留优化成本。

可以选择为与生产、质量和开发环境关联的资源纳入节省计划。另外，由于沙盒资源仅在需要时才会启用，因此您可以为该环境中的资源选择按需模式。使用 Amazon 竞价型实例降低 Amazon EC2 成本，或者使用计算类节省计划降低 Amazon EC2、Fargate 和 Lambda 成本。AWS Cost Explorer 建议工具提供通过节省计划获得承诺折扣的机会。

如果您过去曾购买过 Amazon EC2 的预留实例，或者组织内部已经建立成本分配实践，则可以暂时继续使用 Amazon EC2 预留实例。但是，我们建议制定一项战略，以便在未来使用节省计划，它是一种更灵活的成本节省机制。您可以在 AWS Cost Management 中刷新节省计划（SP）建议，以便随时生成新的节省计划建议。使用预留实例（RI）降低 Amazon RDS、Amazon Redshift、Amazon ElastiCache 和 Amazon OpenSearch Service 成本。节省计划和预留实例提供三个选项：全额预付、部分预付和无预付款。使用 AWS Cost Explorer RI 和 SP 购买建议中提供的建议。

要为竞价型实例工作负载寻找机会，请查看总体使用情况的小时视图，并确定使用情况或弹性的定期变化周期。您可以为各种容错和灵活的应用程序使用 Spot 实例。示例包括无状态 Web 服务器、API 端点、大数据和分析应用程序、容器化工作负载、CI/CD 及其他灵活工作负载。

分析 Amazon EC2 和 Amazon RDS 实例，确定在不使用时（下班后和周末）是否可以将其关闭。与 24/7 全天候使用这些实例相比，您可以使用这种方法将成本降低 70% 或更多。如果您的 Amazon Redshift 集群只需要在特定时间内可用，则可以暂停集群并在稍后恢复它。停止 Amazon Redshift 集群或 Amazon EC2 和 Amazon RDS 实例时，计算计费会停止，仅收取存储费用。

请注意，按需容量预留（ODCR）不是定价折扣。无论您是否在预留容量中运行实例，都按等同的按需费率为您计算容量预留费用。当您需要为计划运行的资源提供足够的容量时，应该考虑使用这种模

式。ODCR 不必做出长期承诺，因为当您不再需要时可以取消它们，但它们也可以享受节省计划或预留实例提供的折扣。

实施步骤

- 分析工作负载弹性：在 Cost Explorer 中使用每小时粒度或者使用自定义控制面板分析工作负载弹性。确定正在运行的实例数量的规律性变化。短期实例比较适合采用竞价型实例或竞价型实例集。
  - Well-Architected Lab：Cost Explorer
  - Well-Architected Lab：成本可视化
- 查看现有定价合同：查看当前合同或对长期需求的承诺。分析您目前拥有的承诺以及这些承诺中有多少正在使用。利用已有的合同折扣或企业协议。企业协议让客户可以选择定制最适合其需求的协议。对于长期承诺，请考虑预留定价折扣、特定实例类型的预留实例或节省计划、实例系列、AWS 区域和可用区。
- 执行承诺折扣分析：使用账户中的 Cost Explorer 查看节省计划和预留实例建议。要验证您是否实施了具有所需折扣和风险的正确建议，请按照 Well-Architected Lab 的说明操作。

资源

相关文档：

- Accessing Reserved Instance recommendations
- 实例购买选项
- AWS Enterprise

相关视频：

- Save up to 90% and run production workloads on Spot

相关示例：

- Well-Architected Lab：Cost Explorer
- Well-Architected Lab：成本可视化
- Well-Architected Lab：定价模式

COST07-BP02 根据成本选择区域

资源定价在每个区域中可能各不相同。确定区域成本差异，仅当需要满足延迟、数据驻留和数据主权要求时，才在成本较高的区域部署。考虑区域成本有助于您为此工作负载支付最低的总体费用。

在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 Cloud 基础设施是全球性的，托管在世界各地的多个地点，围绕 AWS 区域、可用区、Local Zones、AWS Outposts 和 Wavelength Zones 构建。区域是世界上的一个物理位置，每个区域都是一个独立的地理位置，而 AWS 有多个可用区。可用区是每个区域内多个相互隔离的位置，由一个或多个独立的数据中心组成，每个数据中心都拥有冗余的电力、联网和连接。

每个 AWS 区域 都在当地市场条件下运作，由于土地、光纤、电力和税收等成本的不同，每个区域的资源定价也不同。选择特定区域来运行解决方案组件或整个解决方案，以便可以在全球范围内以尽可能低的价格运行。使用 AWS Calculator 按位置类型（区域、Wavelength Zone 和 Local Zone）和区域搜索服务，估算工作负载在不同区域的成本。

在构建解决方案时，最佳实践是设法将计算资源放在更接近用户的位置，以提供更低的延迟和强大的数据主权。根据您的业务、数据隐私、性能和安全要求来选择地理位置。对于具有全球终端用户的应用程序，请使用多个位置。

如果您不存在数据隐私、安全和业务要求方面的义务，请使用 AWS 服务价格较低的区域来部署工作负载。例如，如果您的默认区域是亚太地区（悉尼）（`ap-southwest-2`），并且不存在使用其他区域的限制（例如数据隐私、安全），则在美国东部（弗吉尼亚州北部）（`us-east-1`）部署非关键（开发和测试）Amazon EC2 实例成本更少。

	合规性	延迟	成本	服务/功能
区域 1	✓	15 ms	$$	✓
区域 2	✓	20 ms	$$$	X
区域 3	✓	80 ms	$	✓
区域 4	✓	15 ms	$$	✓
区域 5	✓	20 ms	$$$	X
区域 6	✓	15 ms	$	✓
区域 7	✓	80 ms	$	✓
区域 8	✓	15 ms	$	X

区域特征矩阵表

前面的矩阵表显示，区域 6 是这种给定场景的最佳选择，因为与其他区域相比，其延迟较低，可以使用相应服务，而且是最便宜的区域。

实施步骤

- 查看 AWS 区域 定价：分析当前区域的工作负载成本。首先使用按服务和使用类型划分的最高成本，计算其他可用区域的成本。如果预测的节省额超过迁移组件或工作负载的成本，则迁移到新区域。
- 审查多区域部署的要求：分析您的业务要求和义务（数据隐私、安全或性能），确认是否存在任何会阻止您使用多个区域的限制。如果不存在要求您只能使用单个区域的限制，则使用多个区域。
- 分析所需的数据传输：在选择区域时考虑数据传输成本。让您的数据靠近客户和资源。在数据流动和数据传输非常少时，选择成本较低的 AWS 区域。根据数据传输业务要求，您可以使用 Amazon CloudFront、AWS PrivateLink、AWS Direct Connect 和 AWS Virtual Private Network 来降低联网成本、提高性能和增强安全性。

资源

相关文档：

- Accessing Reserved Instance recommendations
- Amazon EC2 定价
- 实例购买选项
- 区域表

相关视频：

- Save up to 90% and run production workloads on Spot

相关示例：

- Overview of Data Transfer Costs for Common Architectures
- Cost Considerations for Global Deployments
- What to Consider when Selecting a Region for your Workloads

COST07-BP03 选择具有经济实惠条款的第三方协议

经济实惠的协议和条款可确保这些服务的成本与所提供的效益相称。选择与可为组织带来额外效益相称的协议和定价。

在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

市场上有多种产品可用于管理云环境中的成本。根据客户需求，它们在功能方面可能有一些差异，例如有些侧重于成本治理或成本可见性，而有些则侧重于成本优化。想要得到有效的成本优化和治理，一个关键因素是使用具有必要功能的正确工具和正确的定价模式。这些产品有不同的定价模式。有些产品按每月账单的一定比例收费，有些则按所实现节省额的一定比例收费。理想情况下，您应该只为需要的资源付费。

当您在云中使用第三方解决方案或服务时，确保定价结构契合期望结果非常重要。定价应与其带来的结果和价值成比例。例如，可带来一定百分比节省额的软件中，节省额（结果）越高，其价格也就越高。许可协议规定，随着开支的增加，您需要支付更多的费用，这可能并不总是有利于您优化成本。但是，如果供应商为您的账单的所有部分都提供明显的效益，那么这种按比例收费可能是合理的。

例如，如果您使用的其他服务没有带来任何效益，提供 Amazon EC2 相关建议并按整体账单一定比例收取费用的解决方案将会变得更加昂贵。另一个示例是根据托管资源的成本，按一定百分比收费的托管

服务。实例越大并不一定意味着需要更多的管理工作，但会收取更多费用。验证这些服务定价安排是否在其服务中包含了成本优化计划或功能，以提高效率。

客户可能会发现市场上的这些产品更先进或更易于使用。您需要考虑这些产品的成本，并考虑长期潜在的成本优化结果。

实施步骤

- 分析第三方协议和条款：查看第三方协议中的定价。基于不同的使用水平执行建模，并将新成本（例如使用新服务，或当前服务由于工作负载增长导致使用量增加）纳入考量。确定额外成本能否为业务提供所需效益。

资源

相关文档：

- Accessing Reserved Instance recommendations
- 实例购买选项

相关视频：

- Save up to 90% and run production workloads on Spot

COST07-BP04 针对此工作负载的所有组件实施定价模式

永久运行的资源应利用预留容量，如节省计划或预留实例。短期容量配置为使用竞价型实例或竞价型实例集。按需型实例仅用于无法中断并且运行时长不足以使用预留容量的短期工作负载（根据具体的资源类型，时长介于 25% 到 75% 之间）。

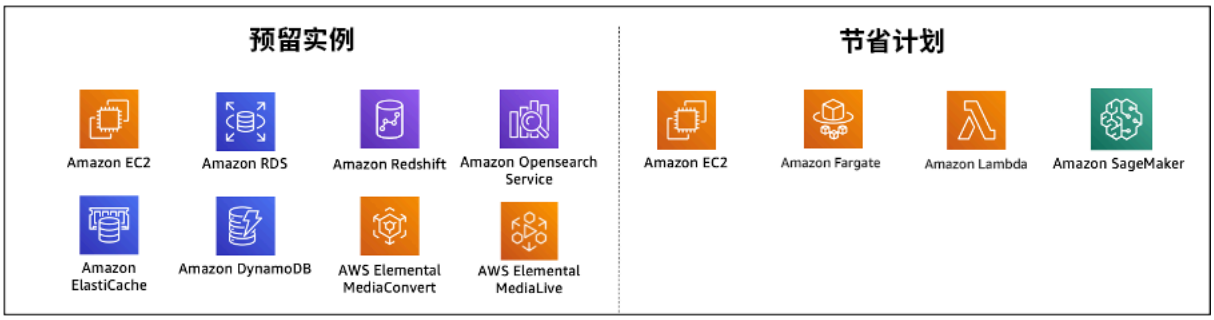
在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

为了提高成本效率，AWS 根据过去的使用情况提供多项承诺建议。您可以使用这些建议，了解可以节省的金额以及如何使用承诺。使用这些服务的方式可以是按需型实例、竞价型实例，或在一定时间内承诺使用量，通过预留实例（RI）和节省计划（SP）降低按需成本。您不仅需要了解每个工作负载组件和多项 AWS 服务，还需要了解这些服务的承诺折扣、购买选项和竞价型实例，以优化工作负载。

考虑工作负载组件的要求，并了解这些服务的不同定价模式。定义这些组件的可用性要求。确定工作负载中是否存在执行功能的多个独立资源，以及工作负载随着时间推移的要求。使用默认的按需定价模式和其他适用模式比较资源成本。考虑资源或工作负载组件的任何潜在更改。

例如，让我们看一下 AWS 上的这个 Web 应用程序架构。此示例工作负载由多项 AWS 服务组成，例如 Amazon Route 53、AWS WAF、Amazon CloudFront、Amazon EC2 实例、Amazon RDS 实例、负载均衡器、Amazon S3 存储和 Amazon Elastic File System（Amazon EFS）。您需要审查每项服务，并确定使用不同定价模式带来的潜在成本节省机会。其中一些服务可能有资格使用 RI 或 SP，而另一些可能只能使用按需方案。如下图所示，一些 AWS 服务可以使用 RI 或 SP 来承诺。



使用预留实例和节省计划承诺的 AWS 服务

实施步骤

- 实施定价模式：使用分析结果，购买节省计划、预留实例或实施竞价型实例。如果是首次承诺购买，请选择列表中的前 5 项或 10 项建议，然后在接下来的一到两个月内监控并分析结果。AWS Cost Management Console 会指导您完成整个过程。查看控制台中的 RI 或 SP 建议，自定义建议（类型、付款和期限），查看每小时承诺（例如每小时 20 美元），然后添加到购物车。折扣将自动应用于符合条件的使用情况。定期购买少量承诺折扣（例如每 2 周或每月一次）。对可以中断或者无状态的工作负载实施竞价型实例。最后，选择按需型 Amazon EC2 实例并为其余需求分配资源。
- 工作负载审核周期：实施工作负载审核周期，用于专门分析定价模式覆盖范围。工作负载达到所需覆盖范围后，可部分（每几个月）购买额外的承诺折扣，或者随着组织的使用情况变化进行购买。

资源

相关文档：

- Understanding your Savings Plans recommendations
- Accessing Reserved Instance recommendations
- 如何购买预留实例
- 实例购买选项

- 竞价型实例
- 其他 AWS 服务的预留模型
- Savings Plans Supported Services

相关视频：

- Save up to 90% and run production workloads on Spot

相关示例：

- What should you consider before purchasing Savings Plans?
- How can I use Cost Explorer to analyze my spending and usage?

COST07-BP05 在管理账户级别执行定价模式分析

检查账单和成本管理工具，并查看承诺和预留的建议折扣，以便在管理账户级别执行定期分析。

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

定期执行成本建模可帮助您跨多个工作负载进行优化。例如，如果总体上多个工作负载使用按需型实例，则变更的风险较低，并且实施基于承诺的折扣可降低总体成本。建议每两周到一个月定期执行一次分析。这样您可以进行少量调整性采购，因此定价模式的覆盖范围会随着工作负载及其组件的变化而不断变化。

使用 AWS Cost Explorer 建议工具，寻找在管理账户中享受承诺折扣的机会。在计算管理账户级别的建议时，考虑 AWS 组织中具有预留实例（RI）或节省计划（SP）的所有账户的使用情况。它们也是在启用折扣共享的情况下计算的，以便推荐可在不同账户中最大限度实现节省的承诺。

虽然在许多情况下，在管理账户级别购买可以进行优化以最大限度地节省开支，但在某些情况下，您可能会考虑在关联账户级别购买 SP，例如当您希望首先对该特定关联账户中的资源使用情况应用折扣时。成员账户建议在个人账户级别计算，以最大限度地节省每个独立账户的成本。如果您的账户同时拥有 RI 和 SP 承诺，则它们将按以下顺序应用：

1. 区域 RI
2. 标准 RI
3. 可转换 RI

4. 实例节省计划
5. 计算类节省计划

如果您在管理账户级别购买 SP，则将根据从最高到最低的折扣百分比来应用节省额。管理账户级别的 SP 会查看所有关联账户，并在任何可以实现最高折扣的地方应用节省额。如果您希望限制应用节省额的位置，则可以在关联账户级别购买节省计划，而只要该账户在运行符合条件的计算服务，就会首先在其上应用折扣。当账户未运行符合条件的计算服务时，将在同一管理账户下的其他关联账户之间共享折扣。默认情况下，折扣共享处于启用状态，不过您可以根据需要关闭。

在整合账单系列中，节省计划首先应用于所有者账户的使用量，然后应用于其他账户的使用量。只有在启用共享时才会发生这种情况。您的节省计划将首先应用于您的最高节省百分比。如果有多个节省百分比相等的使用量，则节省计划将应用于节省计划费率最低的第一个使用量。节省计划将继续适用，直到没有剩余的使用情况或承诺用量用完为止。任何剩余的使用量按照按需费率收费。您可以在 AWS Cost Management 中刷新节省计划建议，以便随时生成新的节省计划建议。

分析实例的灵活性之后，可以按照建议进行承诺。创建成本模型：使用可能的不同资源选项分析工作负载的短期成本、分析 AWS 定价模式并使其与业务要求保持一致，以发现总拥有成本和成本优化机会。

实施步骤

执行承诺折扣分析：使用账户中的 Cost Explorer 查看节省计划和预留实例建议。确保了解节省计划建议，估算每月支出以及每月节省额。查看管理账户级别的建议，这些建议根据 AWS 组织中启用了 RI 或节省计划折扣共享的所有成员账户的使用情况计算得出，以便各账户实现最大限度的节省。您可以按照 Well-Architected Lab 的说明操作，确认您实施了具有所需折扣和风险的正确建议。

资源

相关文档：

- AWS 如何定价？
- 实例购买选项
- Saving Plan Overview
- Saving Plan recommendations
- Accessing Reserved Instance recommendations
- Understanding your Saving Plans recommendation
- How Savings Plans apply to your AWS usage
- 节省计划与整合账单

- 开启共享的预留实例和节省计划折扣

相关视频：

- Save up to 90% and run production workloads on Spot

相关示例：

- 在购买节省计划之前我应该考虑什么？
- How can I use rolling Savings Plans to reduce commitment risk?
- 何时使用竞价型实例

COST 8. 如何规划数据传输费用？

确保规划并监控数据传输费用，以便制定架构决策，尽可能降低成本。持续以小步迭代的方式进行架构优化可以实现运营成本的大幅降低。

最佳实践

- COST08-BP01 执行数据传输建模
- COST08-BP02 选择组件以便优化数据传输成本
- COST08-BP03 实施服务以便降低数据传输成本

COST08-BP01 执行数据传输建模

收集组织要求，并对工作负载及其每个组件执行数据传输建模。这样可以确定满足当前数据传输要求的最低成本点。

在未建立这种最佳实践的情况下暴露的风险等级：高

实施指导

在云端设计解决方案时，由于习惯于设计使用本地数据中心的架构或者缺乏相应的知识，数据传输费用常常会被忽视。AWS 中的数据传输费用由流量的来源、目的地和数量决定。在设计阶段将这些费用考虑在内可以节省成本。了解工作负载中的哪些环节需要进行数据传输、传输成本及其相关好处，对于准确估算总拥有成本（TCO）非常重要。因此，您可以作出明智的决定来修改或接受架构决策。例如，您可能有一个多可用区配置，可以在可用区之间复制数据。

您可以对在工作负载中传输数据的服务组件进行建模，并确定这是可接受的成本（类似于在两个可用区中支付计算和存储费用），以实现所需的可靠性和韧性。对不同使用水平的成本进行建模。工作负载的使用情况可能随时间而变化，不同的服务可能在不同的水平上更具有成本效益。

在对数据传输进行建模时，需考虑接收了多少数据以及这些数据来自哪里。此外，还需考虑处理了多少数据以及需要多少存储或计算容量。在建模期间，应遵循工作负载架构的联网最佳实践，以优化潜在的数据传输成本。

可利用 AWS 定价计算器 查看特定 AWS 服务和预期数据传输的估计成本。如果您已有工作负载在运行（用于测试目的或在预生产环境中），请使用 AWS Cost Explorer 或 AWS 成本和使用情况报告（CUR）来了解数据传输成本并对其建模。配置概念验证（PoC）或测试您的工作负载，并在实际的模拟负载下运行测试。可以根据不同的工作负载需求对成本进行建模。

实施步骤

- 确定要求：计划在来源和目的地之间传输数据的主要目标和业务要求是什么？ 预期的最终业务成果是什么？ 收集业务要求并定义预期结果。
- 确定来源和目的地：数据传输的数据来源和目的地是什么，例如在 AWS 区域 内部、传输到 AWS 服务或传出到互联网？
  - AWS 区域 内部的数据传输
  - AWS 区域 之间的数据传输
  - 将数据传输到互联网
- 确定数据分类：此数据传输的数据分类是什么？ 这是什么样的数据？ 数据量有多大？ 必须多久传输一次数据？ 数据是否敏感？
- 确定要使用的 AWS 服务或工具：此数据传输使用哪些 AWS 服务？ 是否可以将已经预置的服务用于其他工作负载？
- 计算数据传输成本：结合使用 AWS 定价和您之前创建的数据传输建模，计算工作负载的数据传输成本。计算不同使用水平的数据传输成本，包括工作负载使用情况的增加和减少这两种情况。如果工作负载架构有多个传输选项，则计算每个选项的成本以便进行比较。
- 将成本与成果关联：对于产生的每项数据传输成本，指定其实现的工作负载成果。如果在组件之间传输，可能是为了实现解耦；如果在可用区之间传输，则可能是为了实现冗余。
- 创建数据传输建模：收集所有信息后，为多个应用场景和不同的工作负载创建概念性基础数据传输建模。

资源

相关文档：

- AWS 缓存解决方案
- AWS 定价
- Amazon EC2 定价
- Amazon VPC 定价
- Understanding data transfer charges

相关视频：

- Monitoring and Optimizing Your Data Transfer Costs
- S3 Transfer Acceleration

相关示例：

- Overview of Data Transfer Costs for Common Architectures
- AWS Prescriptive Guidance for Networking

COST08-BP02 选择组件以便优化数据传输成本

选择所有组件然后设计架构，以便降低数据传输成本。其中包括使用广域网（WAN）优化和多可用区（AZ）配置等组件

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

针对数据传输进行构建，可最大限度地降低数据传输成本。这可能涉及使用内容分发网络来定位更靠近用户的数据，或者使用从您的本地设施到 AWS 的专用网络链接。您还可以使用 WAN 优化和应用程序优化来减少组件之间传输的数据量。

向 AWS Cloud 传输数据或在其中传输数据时，必须根据不同的应用场景、数据的性质和可用的网络资源来了解目的地，以便选择正确的 AWS 服务来优化数据传输。AWS 提供一系列针对不同数据迁移要求量身定制的数据传输服务。根据组织内的业务需求，选择合适的数据存储和数据传输选项。

在规划或审查您的工作负载架构时，请考虑以下几点：

- 在 AWS 中使用 VPC 端点：VPC 端点允许您在 VPC 与支持的 AWS 服务之间建立私有连接。这让您可以避免使用公共互联网，而使用公共互联网会产生数据传输成本。

- 使用 NAT 网关：使用 NAT 网关，以便私有子网中的实例可以连接到互联网或 VPC 以外的服务。检查发送流量最多的 NAT 网关后面的资源是否与 NAT 网关位于同一可用区。如果未在同一可用区，则在与资源相同的可用区中创建新的 NAT 网关，从而降低跨可用区传输数据的费用。
- 使用 AWS Direct ConnectAWS Direct Connect 绕过公共互联网，并在您的本地网络和 AWS 之间建立直接的私有连接。与通过互联网传输大量数据相比，这可能更具成本效益且更加一致。
- 避免跨区域边界传输数据：AWS 区域 之间（从一个区域到另一个区域）的数据传输通常会产生费用。采用多区域方法应是一项深思熟虑的决定。有关更多详细信息，请参阅多区域场景。
- 监控数据传输：使用 Amazon CloudWatch 和 VPC 流日志，捕获有关数据传输和网络使用情况的详细信息。分析 VPC 中捕获的网络流量信息，例如进出网络接口的 IP 地址或范围。
- 分析网络使用情况：使用 AWS Cost Explorer、CUDOS 控制面板或 CloudWatch 等计量和报告工具来了解工作负载的数据传输成本。

实施步骤

- 选择用于数据传输的组件：使用 COST08-BP01 执行数据传输建模 中介绍的数据传输建模，关注产生最多数据传输成本之处，或者工作负载使用情况发生变化时产生最多数据传输成本之处。寻找替代架构或其他组件，以消除或减少数据传输的需要（或降低其成本）。

资源

相关最佳实践：

- COST08-BP01 执行数据传输建模
- COST08-BP03 实施服务以便降低数据传输成本

相关文档：

- 云数据迁移
- AWS 缓存解决方案
- 使用 Amazon CloudFront 更快地交付内容

相关示例：

- Overview of Data Transfer Costs for Common Architectures
- AWS Network Optimization Tips

- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format

COST08-BP03 实施服务以便降低数据传输成本

实施服务以减少数据传输。例如，使用边缘站点或内容分发网络（CDN）向终端用户交付内容，在应用程序服务器或数据库前构建缓存层，并使用专用网络连接而不是 VPN 连接到云端。

在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

可以利用多种 AWS 服务来优化网络数据传输使用情况。根据工作负载组件、类型和云架构，这些服务有助于您在云端压缩、缓存、共享和分发流量。

- Amazon CloudFront 是一个全球内容分发网络，可提供低延迟、高传输速度的数据。它在世界各地的边缘站点缓存数据，从而减少资源负担。通过使用 CloudFront，您可以减少向全球大量用户分发内容的管理工作，同时将延迟降到最低。如果您计划随着时间的推移增加使用量，安全防御节省套餐有助于您节省高达 30% 的 CloudFront 使用量。
- AWS Direct Connect 允许您建立到 AWS 的专用网络连接。与基于互联网的连接相比，这可以降低网络成本、增加带宽并提供更一致的网络体验。
- AWS VPN 允许您在专用网络和 AWS 全球网络之间建立安全的专用连接。此网络是小型办公室或业务合作伙伴的理想之选，因其提供简化的连接，并且是完全托管的弹性服务。
- VPC 端点允许通过专用网络在 AWS 服务之间建立连接，可用于减少公共数据传输和 NAT 网关成本。网关 VPC 端点不按小时收费，支持 Amazon S3 和 Amazon DynamoDB。接口 VPC 端点由 AWS PrivateLink 提供，有小时费和每 GB 使用成本。
- 与独立的 NAT 实例相比，NAT 网关提供内置的扩展和管理功能，可降低成本。将 NAT 网关放置在与高流量实例所在的同一可用区中，并考虑对需要访问 Amazon DynamoDB 或 Amazon S3 的实例使用 VPC 端点，以降低数据传输和处理成本。
- 使用具有计算资源的 AWS Snow Family 设备在边缘采集和处理数据。AWS Snow Family 设备（Snowball Edge、Snowball Edge 和 Snowmobile）可让您经济高效地将 PB 级数据离线迁移到 AWS Cloud。

实施步骤

- 实施服务：使用数据传输建模并查看 VPC 流日志，根据服务工作负载类型选择适用的 AWS 网络服务。了解产生最多成本和最多数据流之处。了解 AWS 服务并评测是否存在一种服务，可以减少或消

除传输，特别是网络和内容分发。如有需要重复访问数据或存在大量数据的情况，则应了解缓存服务。

资源

相关文档：

- AWS Direct Connect
- AWS 探索我们的产品
- AWS 缓存解决方案
- Amazon CloudFront
- AWS Snow Family
- Amazon CloudFront 安全防御节省套餐

相关视频：

- Monitoring and Optimizing Your Data Transfer Costs
- AWS Cost Optimization Series: CloudFront
- How can I reduce data transfer charges for my NAT gateway?

相关示例：

- How-to chargeback shared services: An AWS Transit Gateway example
- Understand AWS data transfer details in depth from cost and usage report using Athena query and QuickSight
- Overview of Data Transfer Costs for Common Architectures
- Using AWS Cost Explorer to analyze data transfer costs
- Cost-Optimizing your AWS architectures by utilizing Amazon CloudFront features
- How can I reduce data transfer charges for my NAT gateway?

管理需求和供应资源

问题

- COST 9. 如何管理需求和供应资源？

COST 9. 如何管理需求和供应资源？

为了工作负载的性能与支出实现平衡，请确保您支付过费用的所有资源都得到利用，并避免出现实例利用率过低的情况。无论是从运营成本（由于过度使用导致性能下降）还是从浪费 AWS 支出（由于过度预置）的角度衡量，利用率指标过高或过低都会对组织产生负面影响。

最佳实践

- COST09-BP01 对工作负载需求执行分析
- COST09-BP02 实施缓冲区或节流来管理需求
- COST09-BP03 动态供应资源

COST09-BP01 对工作负载需求执行分析

分析工作负载需求随时间的变化。确认分析涵盖季节性趋势，并准确反映整个工作负载生命周期内的运行条件。分析工作应该体现可能带来的好处，例如花费的时间与工作负载成本成正比。

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

分析云计算的工作负载需求涉及了解在云环境中启动的计算任务的模式和特征。这种分析有助于用户优化资源分配、管理成本并验证性能是否达到要求的水平。

了解工作负载的要求。组织要求应指出工作负载对于请求的响应时间。响应时间可用于确定需求是否得到了管理，或者是否应改变资源供应以满足需求。

分析应包括需求的可预测性和可重复性、需求的变化速率以及需求的变化量。对足够长的时间执行分析，以纳入任何季节性变化，例如月末处理或假期高峰。

分析工作应该体现实施扩缩可能带来的好处。查看组件的预期总成本，以及在工作负载生命周期内使用情况和成本的任何增加和减少。

以下是执行云计算的工作负载需求分析时，需要考虑的一些关键方面：

1. 资源利用率和性能指标：分析 AWS 资源在一段时间内的使用情况。确定高峰期和非高峰期使用模式，以优化资源分配和扩缩策略。监控性能指标，如响应时间、延迟、吞吐量和错误率。这些指标有助于评测云基础设施的整体运行状况和效率。
2. 用户和应用程序扩缩行为：了解用户行为及其对工作负载需求的影响。检查用户流量模式有助于增强内容的交付和应用程序的响应能力。分析工作负载如何随着需求的增长而扩展。确定自动扩缩参数的配置是否正确有效，以处理负载波动。

3. 工作负载类型：识别云中运行的不同类型的工作负载，如批处理、实时数据处理、Web 应用程序、数据库或机器学习。每种类型的工作负载都可能有不同的资源需求和性能特征。
4. 服务水平协议（SLA）：将实际绩效与 SLA 进行比较，确保合规性并确定需要改进的方面。

可以使用 Amazon CloudWatch 来收集和跟踪各项指标、监控日志文件、设置警报以及自动应对 AWS 资源的更改。还可以通过使用 Amazon CloudWatch 全面地了解资源利用率、应用程序性能和运行状况。

借助 AWS Trusted Advisor，您可以按照最佳实践预置资源，提高系统性能和可靠性，增强安全性，并寻找节省资金的机会。您也可以关闭非生产实例，并使用 Amazon CloudWatch 和 Auto Scaling 来匹配需求的增加或减少。

最后，可以将 AWS Cost Explorer或 QuickSight 与 AWS 成本和使用情况报告（CUR）文件或应用程序日志一起使用，来对工作负载需求执行高级分析。

总之，通过全面的工作负载需求分析，组织可以就资源预置、扩缩和优化作出明智的决策，从而提高性能、成本效益和用户满意度。

实施步骤

- 分析现有工作负载数据：分析现有工作负载中的数据、以前工作负载版本中的数据或预测使用模式中的数据。使用 Amazon CloudWatch、日志文件和监控数据来深入了解工作负载的使用情况。分析整个工作负载周期，并收集任何季节性变化数据，如月末或年末活动。分析中反映的工作应该体现出工作负载特征。应将最多的精力放在需求变化最大的高价值工作负载上。应将最少的精力放在需求变化最小的低价值工作负载上。
- 预测外部影响：与组织中会影响或更改工作负载需求的团队成员会面。通常涉及的团队包括销售、营销或业务拓展团队。与他们合作，了解其运作周期，以及是否有改变工作负载需求的任何活动。使用这些数据预测工作负载需求。

资源

相关文档：

- Amazon CloudWatch
- AWS Trusted Advisor
- AWS X-Ray
- AWS Auto Scaling

- AWS 实例调度器
- Getting started with Amazon SQS
- AWS Cost Explorer
- QuickSight

相关示例：

- 监控、跟踪和分析以实现成本优化
- Searching and analyzing logs in CloudWatch

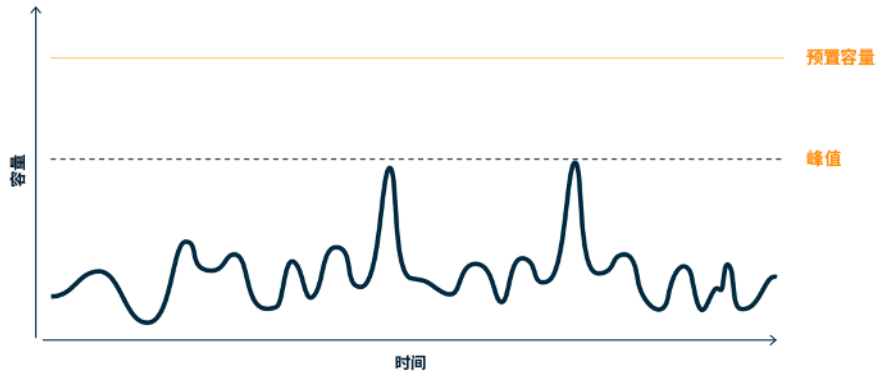
COST09-BP02 实施缓冲区或节流来管理需求

缓冲区和节流可修改工作负载需求，从而避免出现任何峰值情形。在客户端执行重试时实施节流。实施缓冲以存储请求并将处理任务往后推迟一段时间。确认设计节流和缓冲区时客户端能够在所需的时间内收到响应。

在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

在云计算领域，实施缓冲区或节流对于管理需求和减少工作负载所需的预置容量至关重要。为了获得最佳性能，必须衡量总需求，包括峰值、请求变化的速度和必需的响应时间。当客户端能够重新发送请求时，应用节流比较切实可行。相反，对于缺乏重试功能的客户端，理想的方法是实施缓冲区解决方案。此类缓冲区可舒缓请求的涌入，并优化具有不同操作速度的应用程序之间的交互。



需求曲线，有两个不同的峰值，需要高预置容量

假设工作负载的需求曲线如上图所示。此工作负载有两个峰值，为了处理这些峰值，如橙色线所示预置资源容量。因为需要预置容量来处理这两个峰值，所以此工作负载所使用的资源和能源不是由需求曲线

下的区域表示，而是由预置容量线下面的区域表示。展平工作负载需求曲线有助于降低工作负载的预置容量和减少对环境的影响。为了平滑峰值，可以考虑实施节流或缓冲解决方案。

为了更好地理解它们，让我们探讨一下节流和缓冲。

节流：如果需求源具有重试功能，可以实施节流。节流会告诉需求源，如果当前无法处理请求，则应稍后再试。源将等待一段时间，然后重试请求。实施节流的优势是可限制最大资源量和工作负载成本。在 AWS 中，可以使用 Amazon API Gateway 实施节流。

基于缓冲：基于缓冲的方法使用产生器（向队列发送消息的组件）、使用器（从队列接收消息的组件）和队列（保存消息）来存储消息。然后消息将由使用器读取并处理，这样消息就能够以满足使用器业务要求的速率运行。通过使用以缓冲区为中心的方法，产生器发出的消息被存储在队列或流中，随时可供使用器以符合其运营需求的速度进行访问。

在 AWS 中，您可以从多项服务中进行选择，以便实施缓冲方法。Amazon Simple Queue Service（Amazon SQS）是一项托管服务，提供允许单个使用器读取单个消息的队列。Amazon Kinesis 提供允许众多使用器读取相同消息的流。

缓冲和节流可以通过修改工作负载需求来平滑任何峰值。在客户端重试操作时使用节流，并使用缓冲来保存请求以供以后处理。使用基于缓冲区的方法时，请构建工作负载以在所需时间内处理请求，并验证您是否能够处理重复的工作请求。分析总体需求、变化率和所需的响应时间，以使所需节流或缓冲的大小适宜。

实施步骤

- 分析客户端要求：分析客户端请求，确定它们是否能够执行重试。对于无法执行重试的客户端，需要实施缓冲区。分析总体需求、变化率和所需的响应时间，以确定所需的节流或缓冲区大小。
- 实施缓冲或限制：在工作负载中实施缓冲或限制。Amazon Simple Queue Service（Amazon SQS）之类的队列可以为工作负载组件提供缓冲区。Amazon API Gateway 可以为工作负载组件提供节流。

资源

相关最佳实践：

- SUS02-BP06 实施缓冲和节流以展平需求曲线
- REL05-BP02 限制请求

相关文档：

- AWS Auto Scaling
- AWS 实例调度器
- Amazon API Gateway
- Amazon Simple Queue Service
- Getting started with Amazon SQS
- Amazon Kinesis

相关视频：

- Choosing the Right Messaging Service for Your Distributed App

相关示例：

- Managing and monitoring API throttling in your workloads
- Throttling a tiered, multi-tenant REST API at scale using API Gateway
- Enabling Tiering and Throttling in a Multi-Tenant Amazon EKS SaaS Solution Using Amazon API Gateway
- Application integration Using Queues and Messages

COST09-BP03 动态供应资源

资源按计划预置。这种预置可以基于需求（例如通过自动扩缩来实现），也可以基于时间（需求可以预测，基于时间提供资源）。这些方法可以尽可能减少过度预置或预置不足的情况。

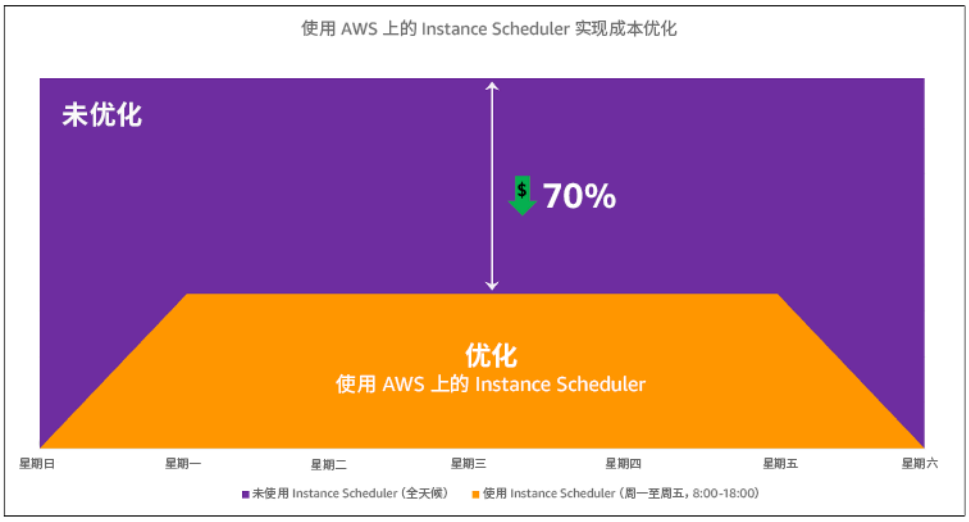
在未建立这种最佳实践的情况下暴露的风险等级：低

实施指导

AWS 客户可以通过多种方式增加可供应用程序使用的资源，并提供资源以满足需求。其中一个选项是使用 AWS 实例调度器，它可以自动启动和停止 Amazon Elastic Compute Cloud（Amazon EC2）及 Amazon Relational Database Service（Amazon RDS）实例。另一个选项是使用 AWS Auto Scaling，该服务让您可以根据应用程序或服务的需求，自动扩缩计算资源。根据需求提供资源，这样您就只需要为使用的资源付费，并且仅在有需要时启动资源，在不需要时终止资源，从而降低成本。

AWS 实例调度器让您可以将 Amazon EC2 和 Amazon RDS 实例配置为在指定的时间停止和启动，这样您就可以通过一致的时间模式满足对相同资源的需求，例如用户在每天早上八点访问 Amazon EC2

实例，晚上六点后就不再需要访问。该解决方案可停止不使用的资源，并在需要时启动它们，帮助降低运营成本。



使用 AWS 实例调度器优化成本。

您还可以通过简单的用户界面（UI）使用 AWS Systems Manager 快速设置功能，轻松地跨账户和区域来为 Amazon EC2 实例配置计划。您可以使用 AWS 实例调度器来计划 Amazon EC2 或 Amazon RDS 实例，也可以停止和启动现有实例。但是，您无法停止和启动属于自动扩缩组（ASG）或管理 Amazon Redshift 或 Amazon OpenSearch Service 等服务的实例。自动扩缩组对组中的实例和何时创建这些实例有自己的计划。

AWS Auto Scaling 有助于您调整容量以维持稳定、可预测的性能，并确保成本最低，以满足不断变化的需求。这是一项用来扩缩应用程序容量的完全托管式免费服务，与 Amazon EC2 实例和竞价型实例集、Amazon ECS、Amazon DynamoDB 与 Amazon Aurora 集成。自动扩缩提供自动资源发现功能，帮助在工作负载中找到可以配置的资源，它具有内置的扩缩策略来优化性能、成本或者在两者之间取得平衡，并提供预测性扩展来协助应对定期出现的峰值。

可以通过多种扩缩选项来扩缩自动扩缩组：

- 始终保持当前实例级别
- 手动缩放
- 按计划扩展
- 根据需求进行扩展
- 使用预测式扩展

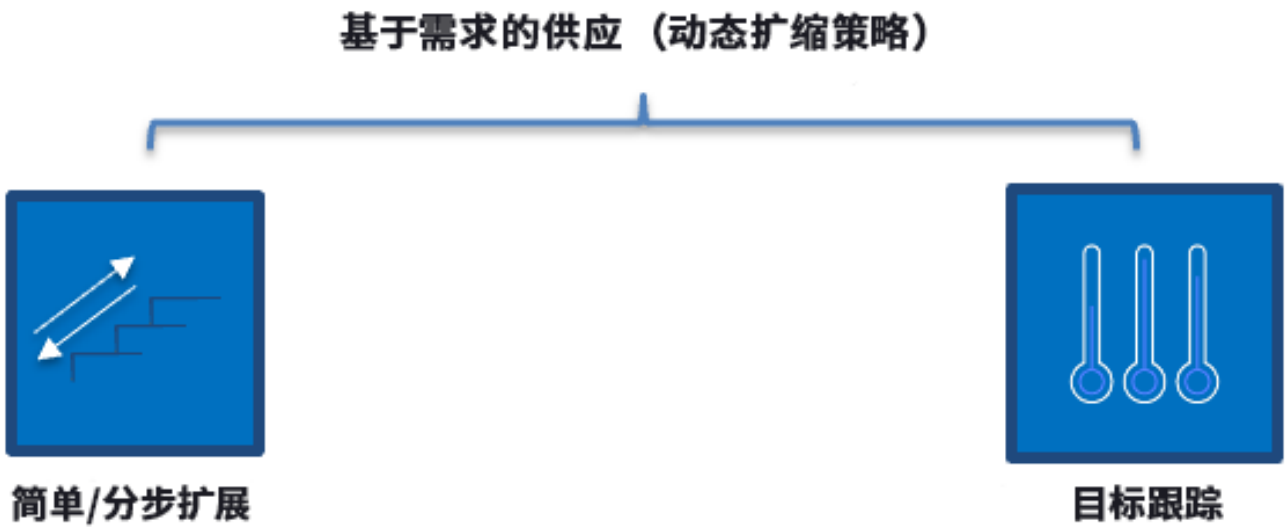
自动扩缩策略各不相同，可以分为动态扩缩策略和计划扩缩策略。动态策略为手动扩缩或动态扩缩，这可以是计划扩缩或者预测性扩缩。您可以针对动态、计划和预测性扩缩使用扩缩策略。还可以使用 Amazon CloudWatch 中的指标和警报来触发工作负载的扩缩事件。我们建议您使用启动模板，确保可以访问最新功能和改进。当您使用启动配置时，并非所有的自动扩缩功能都可用。例如，您无法创建 Auto Scaling 组来同时启动竞价型实例和按需型实例或者指定多个实例类型。您必须使用启动模板来配置这些功能。使用启动模板时，建议您对每个模板进行版本控制。利用启动模板的版本控制，您可以创建全套参数的子集。然后，您可以重复使用它来创建同一启动模板的其他版本。

可以使用 AWS Auto Scaling 或通过 AWS API 或 SDK 在代码中加入扩缩。这样省去了手动更改环境的操作成本，因而工作负载的总体成本得以降低，而且可以更快地执行更改。这还使您的工作负载资源配置随时与您的需求相匹配。为了遵循这一最佳实践，并为组织动态供应资源，您应该了解 AWS Cloud 中的水平和垂直扩缩，以及 Amazon EC2 实例上运行的应用程序的性质。您的云财务管理团队最好与技术团队合作，以遵循这一最佳实践。

弹性负载均衡（ELB）通过在多种资源之间分配需求来帮助您扩缩规模。通过使用 ASG 和弹性负载均衡，可以按照最优方式路由流量来管理传入的请求，这样就可以避免自动扩缩组中某个实例负载过高的情况。请求将以轮询方式，在目标组的所有目标上分配，而不考虑容量或利用率。

典型的指标可以是标准 Amazon 指标，例如 CPU 利用率、网络吞吐量以及弹性负载均衡观察到的请求和响应延迟。如果可能，应该使用指示客户体验的指标，通常是来自工作负载中的应用程序代码的自定义指标。在本文档中，为了详细说明如何动态满足需求，我们将自动扩缩划分为两类：基于需求的供应模型和基于时间的供应模型，并分别深入探讨这两种模型。

基于需求的供应：利用云的弹性来提供资源，根据近实时的需求状态来满足不断变化的需求。对于基于需求的供应，请使用 API 或服务功能，以编程方式改变架构中云资源的数量。这让您能够在架构中扩缩组件，并在需求高峰期间增加资源数量以保持性能，也可以在需求量降低时减少容量以降低成本。

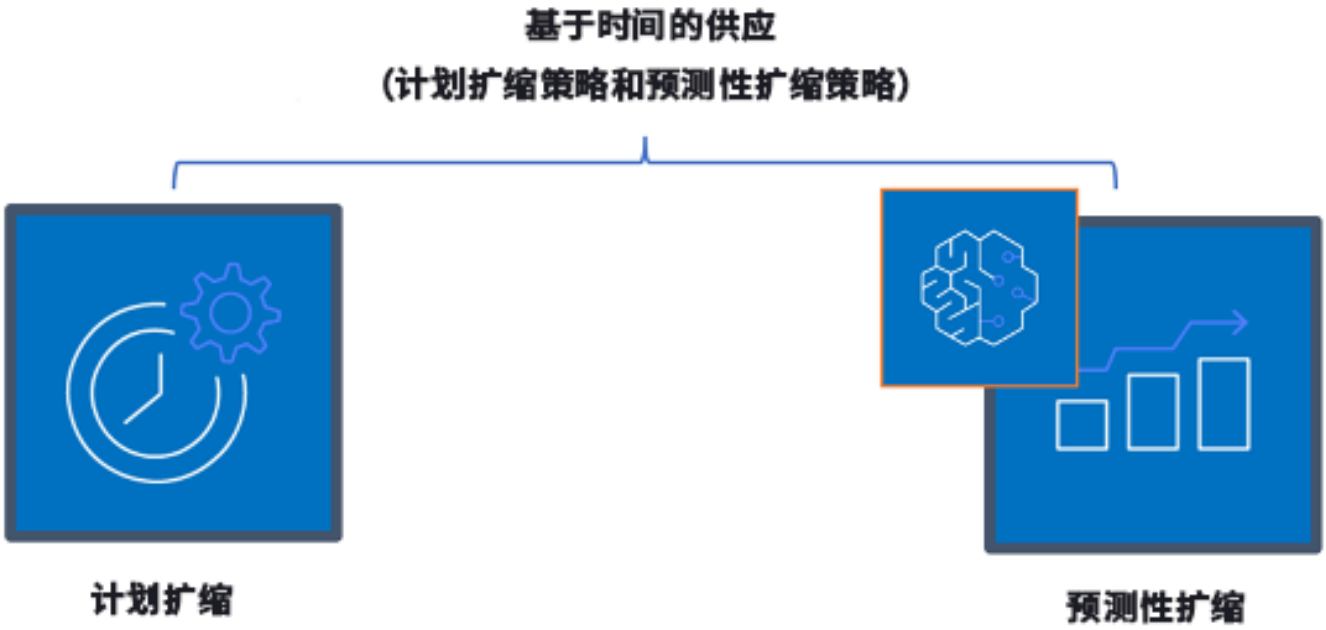


基于需求的动态扩缩策略

- 简单/步骤扩缩：监控指标，按照客户定义的步骤手动添加/删除实例。
- 目标跟踪：类似恒温器的控制机制，可自动添加或删除实例，以维护客户定义的目标指标。

当采用基于需求的方法进行构建时，请注意两个重要事项。首先，了解您必须以多快的速度预置新资源。其次，了解供应和需求之间的差额将发生变化。您必须准备好应对需求变化的速度，并准备好应对资源故障。

基于时间的供应：基于时间的方法可以协调资源容量，以满足可预测或按时间明确定义的需求。此方法通常不依赖资源的利用水平。基于时间的方法可以确保资源在需要的特定时间可用，并且提供时不会因启动程序和系统或一致性检查而发生延迟。使用基于时间的方法，您可以在繁忙时段提供额外的资源或增加容量。



基于时间的扩缩策略

您可以使用计划性或预测性自动扩缩，实施基于时间的方法。工作负载可以在规定的时间按计划横向扩展或缩减（例如办公时间开始时），从而确保用户就位或需求增加时资源可用。预测性扩缩使用相关模式进行横向扩展，而计划扩缩在预先规定的时间进行横向扩展。您还可以在自动扩缩组中使用基于属性的实例类型选择（ABS）策略，该策略允许您将实例要求表示为一组属性，例如 vCPU、内存和存储。这还允许您在新一代实例类型发布时自动使用它们，并通过 Amazon EC2 竞价型实例访问更广泛的容量。Amazon EC2 Fleet 和 Amazon EC2 Auto Scaling 会选择并启动符合指定属性的实例，无需手动选择实例类型。

可以利用 AWS API 和 SDK 以及 AWS CloudFormation，在需要时自动预置和停用整个环境。此方法非常适合仅在规定的办公时间或时间段运行的开发或测试环境。您可以使用 API 来扩缩环境中的资源大小（垂直扩缩）。例如，可以通过更改实例大小或类纵向扩展生产工作负载。这可以通过停止和启动实例，以及选择不同的实例大小或类来实现。这种技巧也可以应用于其他资源，如 Amazon EBS 弹性卷，您可以在使用时对其进行修改以增加大小、调整性能（IOPS）或更改卷类型。

当采用基于时间的方法进行构建时，请注意两个重要事项。首先，使用模式的一致性如何？ 其次，如果模式发生更改会产生什么影响？ 您可以通过两种方式提高预测的准确性：监控工作负载和使用商业智能。如果您发现使用模式发生重大更改，可以调整时间，以确保提供覆盖范围。

实施步骤

- 配置计划扩缩：对于可预测的需求变化，基于时间的扩缩可以及时提供正确的资源数量。如果资源创建和配置的速度不够快，无法响应需求变化，也可使用这种方法。根据工作负载分析，使用 AWS Auto Scaling 配置计划扩缩。要配置基于时间的计划，可以使用预测性扩缩而不是计划扩缩，根据预期或可预测的负载变化，提前增加自动扩缩组中的 Amazon EC2 实例数量。
- 配置预测性扩缩：使用预测性扩缩，可在流量流的每日和每周模式之前增加自动扩缩组中的 Amazon EC2 实例数量。如果您有定期的流量高峰和需要很长时间才能启动的应用程序，则应该考虑使用预测性扩缩。与本质上属于被动应对的单纯动态扩缩相比，预测性扩缩可帮助您在预计负载到来之前初始化容量，从而更快地扩缩。例如，如果用户在开始上班时开始使用工作负载，而在下班后不再使用，预测性扩缩可以在上班之前增加容量，这就消除了动态扩缩对流量变化作出反应的延迟。
- 配置动态自动扩缩：要根据活动工作负载指标配置扩缩，请使用自动扩缩。使用分析并配置自动扩缩以在正确的资源级别启动，并确认工作负载在所需的时间内横向缩减。您可以启动并自动扩展单个 Auto Scaling 组中的一组按需实例和竞价型实例。除了享受使用竞价型实例的折扣外，您还可以使用预留实例或 Savings Plan 获得常规按需实例定价的折扣费率。以上所有因素的综合作用是帮助您进一步节约 Amazon EC2 实例成本，同时帮助您获得应用程序所需的规模和性能。

资源

相关文档：

- AWS Auto Scaling
- AWS 实例调度器
- 扩展 Auto Scaling 组的大小
- Getting Started with Amazon EC2 Auto Scaling
- Getting started with Amazon SQS
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Predictive scaling for Amazon EC2 Auto Scaling

相关视频：

- Target Tracking Scaling Policies for Auto Scaling
- AWS 实例调度器

相关示例：

- 基于属性选择实例类型用于 Amazon EC2 Fleet 的自动扩缩
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive Scaling with Amazon EC2 Auto Scaling
- 如何将实例调度器和 AWS CloudFormation 配合使用来计划 Amazon EC2 实例？

持续优化

问题

- COST 10. 如何评估新服务?
- COST 11. 如何评估工作量成本？

COST 10. 如何评估新服务?

AWS 不断发布新服务和功能，因此您最好不断审视现有架构决策，确保其始终最具成本效益。

最佳实践

- COST10-BP01 制定工作负载审核流程
- COST10-BP02 定期审核和分析此工作负载

COST10-BP01 制定工作负载审核流程

制定流程，定义工作负载的审核标准和流程。审核工作量应该体现可能带来的好处，例如，核心工作负载或费用占比超过 10% 的工作负载每季度或每六个月审核一次，而费用占比低于 10% 的工作负载每年审核一次。

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

为了让工作负载始终最具成本效益，必须定期对工作负载进行审核，了解是否有机会实施新的服务、功能和组件。为了降低总体成本，审核工作量必须与潜在的节省额成比例。例如，与占总支出 5% 的工作负载相比，应更经常、更彻底地审核占总支出 50% 的工作负载。考虑任何外部因素或波动。如果工作负载服务于特定的地理位置或市场领域，并且您已预测出该领域会出现的变化，则提高审核频率可能会节省成本。审核时要考虑的另一个因素是实施更改的工作量。如果测试和验证变更的成本很高，则审核的频率应该降低。

考虑维护过时和旧式组件及资源的长期成本，以及无法在其中实施新功能的事实。当前的测试和验证成本可能会超过预计的效益。但是，随着时间的推移，工作负载和当前技术之间的差距会增大，进行更改的成本可能会大幅增加，导致成本升高。例如，迁移到新的编程语言的成本当前可能不具成本效益。然而，五年之后，熟练使用该语言的人员的成本可能会增加，并且由于工作负载的扩展，迁移到新语言的系统规模更大，这其中涉及的工作量甚至高于以前。

将工作负载分解成多个组件，分配组件的成本（估算即可），然后在每个组件旁边列出因素（例如工作量和外部市场）。使用这些指示信息来确定每个工作负载的审核频率。例如，您可能觉得 Web 服务器的成本高、变更的工作量小、外部因素多，因而审核频率很高。而中央数据库的成本可能中等、变更的工作量很大、外部因素较少，因而审核频率也为中等。

制定相应流程，在新的服务、设计模式、资源类型和配置推出后，对它们进行评估，以优化工作负载成本。与绩效支柱审核和可靠性支柱审核流程类似，确定、验证、优化和改进活动并确定其优先顺序，然后提出补救措施并将其纳入积压工作中。

实施步骤

- 定义审核频率：定义工作负载及其组件的审核频率。分配时间和资源，以持续改进并保持审核频率，优化工作负载并提高工作负载的效率。应考虑多种因素，且这些因素可能会因组织中的工作负载以及工作负载中的组件而异。常见的因素包括：从收入或品牌角度来讲对组织的重要性、运行工作负载的总成本（包括运营和资源成本）、工作负载的复杂性、实施更改的难易程度、任何软件许可协议，以及更改是否会因惩罚性的许可而显著增加许可成本。可以在功能或技术上定义组件，例如 Web 服务器和数据库，或计算和存储资源。相应地权衡这些因素，并为工作负载及其组件制定一个周期。您可能决定每 18 个月审核一次完整的工作负载，每 6 个月审核一次 Web 服务器，每 12 个月审核一次数据库，每 6 个月审核一次计算资源和短期存储，每 12 个月审核一次长期存储。
- 定义审核的彻底性：定义在工作负载或工作负载组件的审核上投入的工作量。与审核频率类似，这也需要权衡多种因素。评估各种改进机会并确定其优先顺序，以便将精力集中在可以实现最大效益的工作上，同时估计这些活动所需的工作量。如果预期结果不符合目标，并且所需工作量会花费更多成本，则寻求其他行动方案。审核流程中应该分配专用的时间和资源，以便实现持续渐进式改进。例如，您可能决定投入一周的时间对数据库组件进行分析，投入一周的时间对计算资源进行分析，投入四小时的时间进行存储审核。

资源

相关文档：

- AWS 新闻博客
- 云计算类型

- AWS 的新功能

相关示例：

- AWS Support Proactive Services
- 定期审核 SAP 工作负载的工作负载

COST10-BP02 定期审核和分析此工作负载

根据每个明确的流程定期审核现有工作负载，以便确定是采用新服务、替换现有服务还是重新构建工作负载。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

AWS 在不断地增加新功能，以便您可以使用最新技术更快地进行实验和创新。AWS新增功能详细介绍了 AWS 如何做到这一点，并简要概述了 AWS 服务、功能和区域扩展公告的发布时间表。您可以深入了解已经宣布的产品发布，并使用它们来审核和分析现有的工作负载。为了享受新 AWS 服务和功能带来的优势，请对工作负载进行审核，并根据需要实施新服务和功能。这意味着您可能需要替换用于工作负载的现有服务，或对工作负载进行现代化改造，以便采用这些新的 AWS 服务。例如，可以审核工作负载，并使用 Amazon Simple Email Service 替换消息传递组件。这省去了运行和维护实例集的成本，同时能以更低的成本提供所有功能。

为了分析您的工作负载并突显潜在机会，您不仅应考虑新服务，而且还要考虑构建解决方案的新方法。观看 AWS 上的这是我的架构视频，了解其他客户的架构设计、面临的挑战和解决方案。查看 All-In series，了解 AWS 服务的真实应用和客户案例。还可以观看回归基础视频系列，该系列讲解、研究和分解基本的云架构模式最佳实践。另一个来源是如何构建此架构视频，这些视频旨在帮助有宏大想法的人们了解如何使用 AWS 服务将最简可行产品（MVP）变为现实。来自世界各地有着坚定想法的构建者可以利用这种方式从经验丰富的 AWS 解决方案架构师那里获得架构指导。最后，可以查看入门资源材料，其中包含分步教程。

在开始审核流程之前，请遵照企业对工作负载的要求、安全性和数据隐私要求，以便在遵循商定的审核流程时，运用特定的服务或区域和性能要求。

实施步骤

- 定期审核工作负载：使用定义的流程，按照指定的频率执行审核。确认在每个组件上投入适当的工作量。此流程类似于您选择服务进行成本优化的初始设计流程。分析服务及其带来的优势，这一次需考虑进行更改所产生的成本，而不仅仅是长期优势。

- 实施新服务：如果分析结果表明可以实施更改，请先执行工作负载基线，以了解每项产出的当前成本。实施更改，然后执行分析以确认每项产出的新成本。

资源

相关文档：

- AWS 新闻博客
- AWS 的新功能
- AWS 文档
- AWS 入门
- AWS 一般资源

相关视频：

- AWS – 这是我的架构
- AWS – 回归基础
- AWS - All-In series
- 如何构建此架构

COST 11. 如何评估工作量成本？

最佳实践

- COST11-BP01 执行运营自动化

COST11-BP01 执行运营自动化

评估云端的运营成本，专注于量化在管理任务、部署、人为错误风险缓解、合规性以及通过自动化实现的其他操作方面所节省的时间和工作量。评测运营工作所需的时间和相关成本，实现管理任务的自动化，从而最大限度地减少人工操作。

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

通过实现操作的自动化，还可以在部署、管理或运行工作负载时提供一致且可靠的体验，从而降低手动任务的频率、提高效率以及使客户受益。您可以将基础设施资源从手动操作任务中解放出来，让他们参

与更高价值的任务和创新，从而提高业务价值。企业需要行之有效、经过测试的方法来管理云中的工作负载。该解决方案必须安全、快速、经济高效，风险最低且可靠性最高。

首先着眼于总体运营成本，根据所需的工作量确定运营活动的优先级。例如，在云中部署新资源、对现有资源进行优化更改或实施必要的配置需要多长时间？ 考虑运营和管理成本，看看人工行为的总成本。优先考虑管理任务的自动化，以减少人工工作量。

审核工作应该体现可能带来的好处。例如，检查与自动执行任务相比，手动执行任务所花费的时间。优先考虑自动执行重复、高价值、耗时和复杂的活动。从那些高价值或人为错误风险高的活动开始实现自动化通常会更好，因为风险通常会带来不必要的额外运营成本（例如，运营团队加班产生的成本）。

使用 AWS Systems Manager 或 AWS Config 等自动化工具来简化运营、合规性、监控、生命周期和终止流程。借助 AWS 服务、工具和第三方产品，您可以自定义所实施的自动化来满足特定要求。下表显示了您为了自动执行管理和运营，可以通过 AWS 服务实现的一些核心运营职能和能力：

- AWS Audit Manager：持续审核 AWS 使用情况，以简化风险和合规性评测
- AWS Backup：集中管理和自动化数据保护。
- AWS Config：配置计算资源，评估、审核配置和资源清单。
- AWS CloudFormation：使用“基础设施即代码”启动高可用性资源。
- AWS CloudTrail：IT 变更管理、合规性和控制。
- Amazon EventBridge 调度事件并触发 AWS Lambda 采取行动。
- AWS Lambda：通过使用事件触发重复流程，或者使用 AWS EventBridge 按固定计划运行重复流程，实现这些流程的自动化。
- AWS Systems Manager：启动和停止工作负载、修补操作系统、自动配置和持续管理。
- AWS Step Functions：安排作业并实现工作流程自动化。
- AWS Service Catalog：模板消费，基础设施即代码，具有合规性和控制力。

如果您想在使用 AWS 产品和服务时立即采用自动化，并且组织中没有相关技能，请联系 AWS Managed Services（AMS）、AWS 专业服务团队或 AWS 合作伙伴，以提高自动化的采用率并改善您在云端的卓越运营。

AWS Managed Services（AMS）是代表企业客户和合作伙伴运营 AWS 基础设施的服务。该服务提供了一个安全且合规的环境，您可以将工作负载部署到其中。AMS 使用具有自动化功能的企业云运营模型，让您可以满足组织要求，更快地迁移到云中并降低持续的管理成本。

AWS 专业服务团队还可以帮助您实现期望的业务成果并通过 AWS 实现运营自动化。团队成员帮助客户部署自动、稳健、敏捷的 IT 运营，还提供针对云进行优化的治理能力。有关详细的监控示例和推荐的最佳实践，请参阅《卓越运营支柱白皮书》。

实施步骤

- 一次构建，多次部署：使用基础设施即代码（例如 CloudFormation、AWS SDK 或 AWS CLI）一次部署并多次用于类似环境或灾难恢复场景。在部署时进行标记，以便按照其他最佳实践中的规定跟踪使用情况。使用 AWS Launch Wizard 缩短部署许多常用企业工作负载的时间。AWS Launch Wizard 指导您按照 AWS 最佳实践完成企业工作负载的大小调整、配置和部署。还可以使用 Service Catalog，它可以帮助您创建和管理经基础设施即代码批准的模板以在 AWS 上使用，以便任何人都可以发现经批准的自助式云资源。
- 实现持续合规自动化：考虑根据预定义的标准自动评测和补救记录的配置。将 AWS Organizations 与 AWS Config 和 AWS CloudFormation 的功能相结合后，可以高效地大规模管理和自动化数百个成员账户的配置合规性。可以审核配置更改以及 AWS 资源之间的关系，并深入探究资源配置的历史记录。
- 自动执行监控任务 AWS 为您提供了各种工具用来监控服务。您可以配置这些工具来自动执行监控任务。创建并实施监控计划，以收集工作负载中所有部分的监控数据，在发生多点故障时，就能更轻松地进行调试。例如，当系统状态检查、实例状态检查和 Amazon CloudWatch 警报出现问题时，可以使用自动监控工具来观察 Amazon EC2 并向您发送报告。
- 自动执行维护和操作：自动运行例行操作，无需人工干预。使用 AWS 服务和工具，您可以选择实施哪些 AWS 自动化，并根据您的特定要求进行自定义。例如，使用 EC2 Image Builder 构建、测试和部署虚拟机和容器镜像，以在 AWS 或本地使用，或者使用 AWS SSM 修补 EC2 实例。如果无法使用 AWS 服务完成所需操作，或者需要针对筛选资源执行更复杂的操作，请使用 AWS Command Line Interface（AWS CLI）或 AWS SDK 工具自动执行操作。AWS CLI 提供了使用脚本即可自动执行控制和管理 AWS 服务的整个流程的能力，无需使用 AWS Management Console。选择首选的 AWS SDK 与 AWS 服务交互。有关其他代码示例，请参阅 AWS SDK 代码示例存储库。
- 通过自动化创建持续的生命周期：制定和保留成熟的生命周期策略非常重要，这不仅是为了法规或冗余，还是为了成本优化。可以使用 AWS Backup 集中管理和自动保护数据存储（例如，存储桶、卷、数据库和文件系统）中的数据。还可以使用 Amazon Data Lifecycle Manager 来自动创建、保留和删除 EBS 快照和 EBS 支持的 AMI。
- 删除不必要的资源：在沙盒或开发 AWS 账户 中积累未使用的资源是很常见的情况。在一般开发周期内，开发人员将创建和试验各种服务与资源，之后，当他们不再需要这些资源时，不会将其删除。未使用的资源会给组织带来不必要的成本，这些成本有时甚至非常高。删除这些资源可降低这些环境的运营成本。确认您不需要这些数据，如果无法确定，请确认数据已备份。可以使用 AWS

CloudFormation 清理已部署的堆栈，这将自动删除模板中定义的大多数资源。或者，可以使用诸如 aws-nuke 之类的工具实现 AWS 资源删除操作的自动化。

资源

相关文档：

- Modernizing operations in the AWS Cloud
- AWS Services for Automation
- 基础设施和自动化
- AWS Systems Manager Automation
- 自动和手动监控
- AWS automations for SAP administration and operations
- AWS Managed Services
- AWS Professional Services

相关视频：

- Automate Continuous Compliance at Scale in AWS
- AWS Backup Demo: Cross-Account & Cross-Region Backup
- Patching for your Amazon EC2 Instances

相关示例：

- Reinventing automated operations (Part I)
- Reinventing automated operations (Part II)
- Automate deletion of AWS resources by using aws-nuke
- Delete unused Amazon EBS volumes by using AWS Config and AWS SSM
- Automate continuous compliance at scale in AWS
- IT Automations with AWS Lambda

可持续性

可持续性支柱包括在构建云工作负载时，了解所使用服务的影响，量化整个工作负载生命周期的影响，并应用设计原则和最佳实践来减少这些影响。有关具体实施的说明性指导，请参阅《可持续性支柱白皮书》。

最佳实践领域

- 区域选择
- 符合需求
- 软件和架构
- 数据
- 硬件和服务
- 流程和文化

区域选择

问题

- SUS 1 如何为工作负载选择区域？

SUS 1 如何为工作负载选择区域？

为工作负载选择区域会显著影响其 KPI，包括性能、成本和碳足迹。为了有效提高这些 KPI，您应该根据业务要求和可持续性目标为工作负载选择区域。

最佳实践

- SUS01-BP01 根据业务要求和可持续性目标选择区域

SUS01-BP01 根据业务要求和可持续性目标选择区域

根据您的业务需求和可持续性目标为您的工作负载选择一个区域，以优化其 KPI，包括性能、成本和碳足迹。

常见反模式：

- 您可以根据自己所在的位置选择工作负载的区域。
- 将所有工作负载资源整合到一个地理位置中。

建立此最佳实践的好处：将工作负载放置在 Amazon 可再生能源项目或已发布碳强度较低的区域附近，有助于降低云工作负载的碳足迹。

在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 Cloud 是一个不断扩展的区域和入网点（PoP）网络，其全球网络基础设施将它们连接在一起。为工作负载选择区域会显著影响其 KPI，包括性能、成本和碳足迹。为了有效提高这些 KPI，您应该根据业务需求和可持续性目标为工作负载选择区域。

实施步骤

- 将潜在区域列入候选名单：按照以下步骤进行操作，根据您的业务需求（包括合规性、可用功能、成本和延迟）评测工作负载的潜在区域并列入候选名单：
  - 根据您所需的当地法规（例如数据主权），确认这些区域合规。
  - 使用 AWS 区域性服务列表，检查区域是否具有运行工作负载所需的服务和功能。
  - 使用 AWS 定价计算器 计算每个区域的工作负载成本。
  - 测试最终用户位置与每个 AWS 区域 之间的网络延迟。
- 选择区域：选择 Amazon 可再生能源项目附近的区域和其电网公布的碳强度低于其它位置（或区域）的区域。
  - 确定您的相关可持续性准则，以根据温室气体核算协议（基于市场和基于位置的方法）跟踪和比较逐年碳排放量。
  - 根据用于跟踪碳排放的方法选择区域。有关根据可持续性准则选择区域的更多详细信息，请参阅《How to select a Region for your workload based on sustainability goals》。

资源

相关文档：

- 了解碳排放估算
- Amazon 遍布全球
- 可再生能源方法
- What to Consider when Selecting a Region for your Workloads

相关视频：

- AWS re:Invent 2023 - Sustainability innovation in AWS Global Infrastructure
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure

符合需求

问题

- SUS 2 如何将云资源与您的需求相匹配？

SUS 2 如何将云资源与您的需求相匹配？

用户和应用程序使用您的工作负载及其他资源的方式可以帮助您确定改进方面，以实现可持续性目标。扩展基础设施以持续匹配需求，并确认您仅使用了支持用户所需的最少资源。使服务水平与客户需求保持一致。定位资源以限制用户和应用程序使用这些资源所需的网络。删除未使用的资产。为团队成员提供满足其需求的设备，并尽可能降低他们的可持续性影响。

最佳实践

- SUS02-BP01 动态扩展工作负载基础设施
- SUS02-BP02 使 SLA 与可持续性目标保持一致
- SUS02-BP03 停止创建和维护未使用的资产
- SUS02-BP04 根据其联网要求优化工作负载的地理位置
- SUS02-BP05 针对执行的活动优化团队成员资源
- SUS02-BP06 实施缓冲和节流以展平需求曲线

SUS02-BP01 动态扩展工作负载基础设施

利用云的弹性并动态扩展基础设施，以使云资源的供应与需求相匹配，避免在工作负载中过度调配容量。

常见反模式：

- 您没有扩展基础设施以匹配用户负载。

- 您一直在手动扩展基础设施。
- 在扩展事件之后保留增加的容量，而不是缩减容量。

建立此最佳实践的好处：配置和测试工作负载弹性有助于有效地将云资源的供应与需求相匹配，并避免过度调配容量。您可以利用云中的弹性，在需求高峰期间和之后自动扩展容量，以确保您只使用满足业务需求所需的适当数量的资源。

在未建立这种最佳实践的情况下暴露的风险等级：中

实施指导

云让您能够通过各种机制灵活地动态扩展或缩减资源，以便满足不断变化的需求。供应与需求的最佳匹配提供了最低的工作负载环境影响。

需求可以是固定的，也可以是变化的，需要指标和自动化来确保管理不会变成沉重负担。应用程序可以通过修改实例大小来纵向扩展或缩减，通过修改实例数量来横向扩展或缩减，或者组合使用这两种方式。

您可以使用大量不同方法来实现资源的供需匹配。

- 目标跟踪方法：监控您的扩缩指标，并根据需要自动增加或减少容量。
- 预测性扩缩：根据每日和每周的趋势进行扩缩。
- 基于计划的方法：根据可预测的负载变化设置自己的扩缩计划。
- 服务扩缩：选择按设计可以原生扩缩或者将自动扩缩作为一项功能提供的服务（如无服务器）。

确定利用率低或无利用率的时段，缩减资源以消除过剩容量并提高效率。

实施步骤

- 弹性可根据对您拥有的资源的需求来提供这些资源。实例、容器和函数提供了弹性机制，可以与自动扩缩结合使用，也可以作为服务的一项功能。AWS 提供了一系列自动扩缩机制，以确保工作负载可以在低用户负载期间快速轻松地缩减。以下是自动扩缩机制的一些示例：

自动扩缩机制	使用情形
Amazon EC2 Auto Scaling	用于验证您拥有适量的 Amazon EC2 实例，可处理应用程序的用户负载。

自动扩缩机制	使用情形
Application Auto Scaling	用于自动扩展 Amazon EC2 以外的各项 AWS 服务的资源，比如 Lambda 函数或 Amazon Elastic Container Service（Amazon ECS）服务。
Kubernetes Cluster Autoscaler	用于自动扩展 AWS 上的 Kubernetes 集群。

- 扩缩通常与计算服务（如 Amazon EC2 实例或 AWS Lambda 函数）相关。考虑使用非计算服务配置（如 Amazon DynamoDB 读写容量单元或 Amazon Kinesis Data Streams 分片）来满足需求。
- 验证衡量扩展或缩减的指标已根据所部署的工作负载类型进行了验证。如果您正在部署一个视频转码应用程序，预计 CPU 利用率为 100%，但不应将此作为主要指标，如果需要，可以对扩缩策略使用自定义指标（如内存利用率）。要选择正确的指标，请考虑以下关于 Amazon EC2 的指导：
  - 指标应该是有效的利用率指标，并描述实例的繁忙程度。
  - 指标值必须随着自动扩缩组中的实例数按比例增加或减少。
- 对自动扩缩组使用动态扩缩而不是手动扩缩。我们还建议在动态扩缩中使用目标跟踪扩缩策略。
- 确认工作负载部署可以处理横向扩展事件和横向缩减事件。为横向缩减事件创建测试场景，以确认工作负载的行为符合预期，并且不会影响用户体验（如丢失粘滞会话）。您可以使用活动历史记录验证自动扩缩组的扩缩活动。
- 评估工作负载，得出可预测的模式，从而在预期需求会发生预测性的计划内变化时主动扩缩。预测性扩缩可以避免过度预置容量。有关更多详细信息，请参阅 Predictive Scaling with Amazon EC2 Auto Scaling。

资源

相关文档：

- Getting Started with Amazon EC2 Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- 使用 Amazon OpenSearch Service、Amazon Data Firehose 和 Kibana 分析用户行为
- 什么是 Amazon CloudWatch？
- 在 Amazon RDS 上使用性能详情监控数据库负载
- 介绍对 Amazon EC2 Auto Scaling 预测式扩缩的原生支持
- 介绍 Karpenter – 高性能开源 Kubernetes Cluster Autoscaler

- Deep Dive on Amazon ECS Cluster Auto Scaling

相关视频：

- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment
- AWS re:Invent 2022 - Scaling containers from one user to millions
- AWS re:Invent 2023 - Scaling FM inference to hundreds of models with Amazon SageMaker AI
- AWS re:Invent 2023 - Harness the power of Karpenter to scale, optimize & upgrade Kubernetes

相关示例：

- Autoscaling

SUS02-BP02 使 SLA 与可持续性目标保持一致

根据您的可持续性目标审查和优化工作负载服务水平协议（SLA），以便在继续满足业务需求的同时，尽量减少支持您的工作负载所需的资源。

常见反模式：

- 工作负载 SLA 未知或模棱两可。
- 只针对可用性和性能定义您的 SLA。
- 对所有工作负载使用相同设计模式（如多可用区架构）。

建立此最佳实践的好处：使 SLA 与可持续性目标一致，在满足业务需求的同时实现最佳资源使用率。

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

SLA 定义云工作负载的预期服务水平，如响应时间、可用性和数据留存。它们影响云工作负载的架构、资源使用率和环境影响。定期审查 SLA，并做出权衡，显著减少资源使用，以换取可接受的服务水平降低幅度。

实施步骤

- 了解可持续性目标：确定组织的可持续性目标，例如碳减排或提高资源利用率。
- 查看 SLA：评估您的 SLA，以评测这些 SLA 是否支持您的业务需求。如果您超出了 SLA，请做进一步审查。
- 了解权衡：了解工作负载复杂性（例如大量并发用户）、性能（如延迟）以及可持续性影响（如所需资源）之间的权衡。通常，重视其中两个因素会以牺牲第三个因素为代价。
- 调整 SLA：调整 SLA，方法是做出权衡，显著降低可持续性影响，以换取可接受的服务等级降低幅度。
  - 可持续性和可靠性：高可用性工作负载往往会消耗更多资源。
  - 可持续性和性能：使用更多资源来提升性能可能会对环境产生更大影响。
  - 可持续性和安全：过度安全的工作负载可能会对环境产生更大影响。
- 尽可能定义可持续性 SLA：纳入工作负载的可持续性 SLA。例如，将最低利用率级别定义为计算实例的可持续性 SLA。
- 使用高效设计模式：使用优先考虑业务关键功能的设计模式（例如 AWS 上的微服务），并允许非关键功能具有较低的服务等级（例如响应时间或恢复时间目标）。
- 沟通并建立责任制：与所有相关利益相关方共享 SLA，包括您的开发团队和客户。使用报告来跟踪和监控 SLA。分配责任以实现 SLA 的可持续性目标。
- 使用激励和奖励：使用激励或奖励，来鼓励达到或超过与可持续性目标保持一致的 SLA。
- 审查和迭代：定期审查和调整您的 SLA，确保其与不断变化的可持续性和绩效目标保持一致。

资源

相关文档：

- Understand resiliency patterns and trade-offs to architect efficiently in the cloud
- Importance of Service Level Agreement for SaaS Providers

相关视频：

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures

- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

SUS02-BP03 停止创建和维护未使用的资产

停用您的工作负载中未使用的资产，以便减少支持您的需求所需的云资源数量，并最大限度地减少浪费。

常见反模式：

- 您没有分析应用程序以查找冗余或不再需要的资产。
- 您没有移除冗余或不再需要的资产。

建立此最佳实践的好处：移除未使用的资产可释放资源并提高工作负载的整体效率。

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

未使用的资产会消耗存储空间和计算能力等云资源。通过识别和消除这些资产，您可以释放这些资源，从而形成更高效的云架构。定期分析应用程序资产（例如预编制的报告、数据集和静态图像）和资产访问模式，以识别冗余、利用率低下的情况和潜在的淘汰目标。移除这些冗余资产以减少工作负载中的资源浪费。

实施步骤

- 执行清点：执行全面清点，确定工作负载中的所有资产。
- 分析使用情况：使用持续监控功能来确定不再需要的静态资产。
- 移除未使用的资产：制定计划，移除不再需要的资产。
  - 在移除任何资产之前，评估移除它会对架构产生什么影响。
  - 整合生成的重叠资产以消除冗余处理。
  - 更新应用程序，以便不再产生和存储不需要的资产。
- 与第三方进行沟通：指示第三方停止生成和存储代您管理但不再需要的资产。要求整合冗余资产。
- 使用生命周期策略：使用生命周期策略自动删除不必要的数据。
  - 您可以使用 Amazon S3 生命周期在对象的整个生命周期中对其进行管理。
  - 您可以使用 Amazon Data Lifecycle Manager 自动创建、保留和删除 Amazon EBS 快照和 Amazon EBS 支持的 AMI。
- 审核和优化：定期审核工作负载以识别和移除任何未使用的资产。

资源

相关文档：

- 优化您的 AWS 基础设施以实现可持续性，第 II 部分：存储
- 如何终止我的 AWS 账户 中不再需要的活动资源？

相关视频：

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Preserving and maximizing the value of digital media assets using Amazon S3
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

SUS02-BP04 根据其联网要求优化工作负载的地理位置

为工作负载选择可缩短网络流量必须传输的距离的云位置和服务，并减少支持您的工作负载所需的总网络资源。

常见反模式：

- 您根据自己所在的位置选择工作负载的区域。
- 将所有工作负载资源整合到一个地理位置中。
- 所有流量都会流经现有的数据中心。

建立此最佳实践的好处：将工作负载放在接近用户的地方可以提供极低的延迟，同时减少网络中的数据移动并减小对环境的影响。

在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 Cloud 基础设施围绕区域、可用区、置放群组和边缘站点（例如，AWS Outposts 和 AWS Local Zones。这些位置选项负责维护应用程序组件、云服务、边缘网络和本地数据中心之间的连接。

分析您的工作负载中的网络访问模式，以便确定如何使用这些云位置选项和缩短网络流量必须传输的距离。

实施步骤

- 分析工作负载中的网络访问模式，以便确定用户如何使用应用程序。
  - 使用 Amazon CloudWatch 和 AWS CloudTrail 等监控工具收集有关网络活动的数据。
  - 分析数据以确定网络访问模式。
- 请根据以下关键元素，为您的工作负载部署选择区域：
  - 您的可持续性目标：如地区选择中所述。
  - 数据所在位置：对于数据密集型应用程序（如大数据和机器学习），应用程序代码的运行应尽量接近数据。
  - 用户所在位置：对于面向用户的应用程序，选择接近您工作负载用户的一个或多个区域。
  - 其他约束：考虑成本和合规性等约束因素，如 What to Consider when Selecting a Region for your Workloads 中所述。
- 对常用资产使用本地缓存或 AWS 缓存解决方案，以提高性能，减少数据移动并减小对环境的影响。

服务	何时使用
Amazon CloudFront	用于缓存静态内容（如图像、脚本和视频）以及动态内容（如 API 响应或 Web 应用程序）。
Amazon ElastiCache	用于缓存 Web 应用程序的内容。
DynamoDB Accelerator	用于将内存中加速添加到 DynamoDB 表。

- 使用可帮助您在更接近工作负载用户的位置运行代码的服务：

服务	何时使用
Lambda@Edge	用于执行计算密集型操作，当对象不在缓存中时启动这些操作。
Amazon CloudFront Functions	用于处理简单应用场景，如 HTTP(S) 请求或响应操作，这些操作可由短期运行的函数启动。
AWS IoT Greengrass	用于为互联设备运行本地计算、消息收发和数据缓存。

- 使用连接池来允许连接重用并减少所需资源。
- 使用不依赖于持久连接和同步更新的分布式数据存储来保持一致性，从而为区域人口提供服务。
- 用共享的动态容量代替预置的静态网络容量，并与其他订阅用户共享网络容量的可持续性影响。

资源

相关文档：

- 优化您的 AWS 基础设施以实现可持续性，第 III 部分：联网
- Amazon ElastiCache 文档
- What is Amazon CloudFront?
- Amazon CloudFront 的主要功能
- AWS 全球基础设施
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- 置放群组
- AWS Local Zones
- AWS Outposts

相关视频：

- 揭秘 AWS 上的数据传输
- 在新一代 Amazon EC2 实例上扩展网络性能
- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2023 - A migration strategy for edge and on-premises workloads
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020 - AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

相关示例：

- AWS Networking 讲习会
- 针对可持续性设计 – 最大限度地减少跨网络的数据移动

SUS02-BP05 针对执行的活动优化团队成员资源

优化提供给团队成员的资源，在支持其需求的同时最大程度地降低对环境可持续性的影响。

常见反模式：

- 忽略了团队成员使用的设备对云应用程序整体效率的影响。
- 手动管理和更新团队成员使用的资源。

建立此最佳实践的好处：优化团队成员资源可以提高支持云的应用程序的整体效率。

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

了解您的团队成员用来使用服务的资源、它们的预期生命周期，以及财务和可持续性影响。实施战略以优化这些资源。例如，在利用率高的可扩展基础设施上，而不是在利用率不高的强力单用户系统上，执行渲染和编译等复杂的操作。

实施步骤

- 使用节能工作站：为团队成员提供节能工作站和外围设备。在这些设备中使用高效的电源管理功能（例如低功耗模式）来减少其能耗
- 使用虚拟化技术：使用虚拟桌面和应用程序串流来限制升级和设备要求。
- 鼓励远程协作：鼓励团队成员使用 Amazon Chime 和 AWS Wickr 等远程协作工具，以减少差旅需求和相关的碳排放。
- 使用节能软件：通过删除或关闭不必要的功能和流程，为团队成员提供节能软件。
- 管理生命周期：评估流程和系统对您的设备生命周期的影响，并选择在满足业务需求的同时最大限度减少设备更换需求的解决方案。定期维护和更新工作站或软件，维持和提高效率。
- 远程设备管理：对设备实施远程管理以减少所需的商务旅行。
  - AWS Systems Manager Fleet Manager 是一种统一的用户界面（UI）体验，有助于您远程管理在 AWS 上或在本地运行的节点。

资源

相关文档：

- 什么是 Amazon WorkSpaces？
- 适用于 Amazon WorkSpaces 的 Cost Optimizer
- Amazon AppStream 2.0 文档
- NICE DCV

相关视频：

- 管理 AWS 上的 Amazon WorkSpaces 的成本

SUS02-BP06 实施缓冲和节流以展平需求曲线

缓冲和节流可展平需求曲线，并降低工作负载所需的预置容量。

常见反模式：

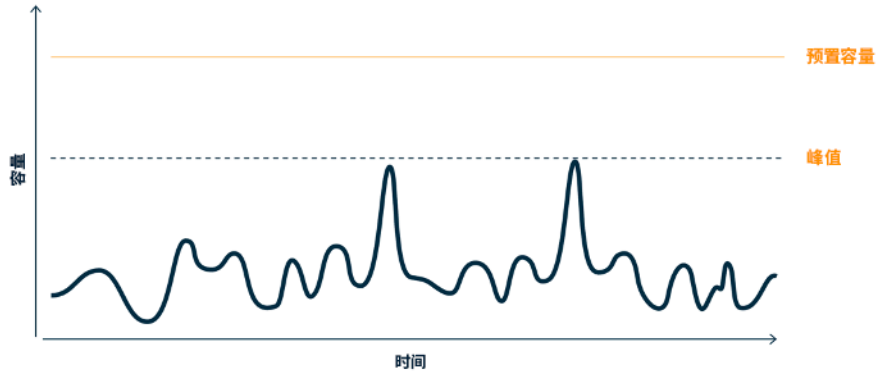
- 在不需要的时候立即处理客户端请求。
- 没有分析客户端请求的要求。

建立此最佳实践的好处：拉平需求曲线可以减少工作负载所需的预置容量。降低预置容量即可减少能源消耗和减少对环境的影响。

在未建立这种最佳实践的情况下暴露的风险等级：低

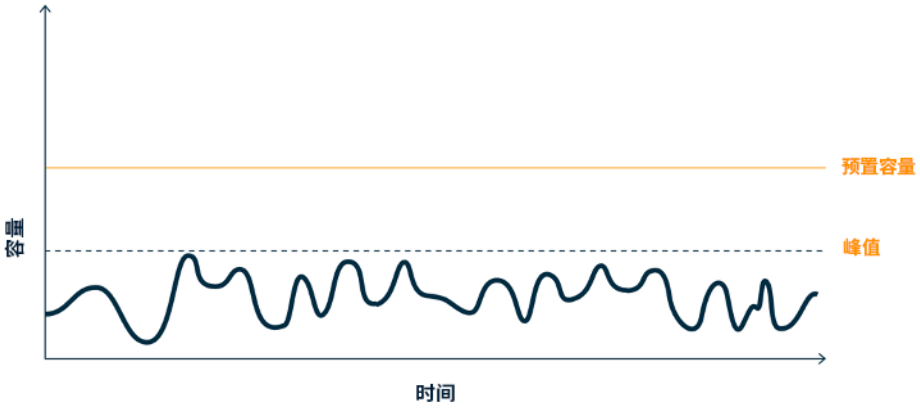
实施指导

展平工作负载需求曲线有助于降低工作负载的预置容量和减少对环境的影响。假设工作负载的需求曲线如下图所示。此工作负载有两个峰值，为了处理这些峰值，如橙色线所示预置资源容量。因为需要预置容量来处理这两个峰值，所以此工作负载所使用的资源和能量不是由需求曲线下的区域表示，而是由预置容量线下面的区域表示。



需求曲线，有两个不同的峰值，需要高预置容量。

您可以使用缓冲和节流来修改需求曲线和弄平峰值，这意味着可以减少预置容量和消耗的能量。在客户端可以执行重试时实施节流。实施缓冲以存储请求并将处理任务往后推迟一段时间。



节流对需求曲线和预置容量的影响。

实施步骤

- 分析客户端请求以确定如何对它们作出响应。要考虑的问题包括：
  - 是否可以异步处理此请求？
  - 客户端是否具有重试能力？
- 如果客户端有重试能力，则您可以实施节流，它会告诉需求源，如果当前无法处理请求，则应稍后再试。
  - 可以使用 Amazon API Gateway 实施节流。

- 对于无法执行重试的客户端，则需要实施缓冲以展平需求曲线。缓冲会延迟请求处理，从而让以不同速率运行的应用程序可以有效通信。基于缓冲的方法使用队列或流来接受来自生产方的消息。然后消息将由使用器读取并处理，这样消息就能够以满足使用器业务要求的速率运行。
  - Amazon Simple Queue Service（Amazon SQS）是一项托管式服务，提供允许单个使用方读取单个消息的队列。
  - Amazon Kinesis 提供允许众多使用器读取相同消息的流。
- 分析总体需求、变化率和所需的响应时间，以使所需节流或缓冲的大小适宜。

资源

相关文档：

- Getting started with Amazon SQS
- Application integration Using Queues and Messages
- Managing and monitoring API throttling in your workloads
- Throttling a tiered, multi-tenant REST API at scale using API Gateway
- Application integration Using Queues and Messages

相关视频：

- AWS re:Invent 2022 - Application integration patterns for microservices
- AWS re:Invent 2023 - Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

软件和架构

问题

- SUS 3 如何利用软件和架构模式来支持可持续性目标？

SUS 3 如何利用软件和架构模式来支持可持续性目标？

实施用于执行负载平滑和保持已部署资源始终如一的高利用率的模式，以最大限度地减少资源消耗。由于用户行为会随着时间的推移而发生变化，组件可能会因缺乏使用而变得空闲。修改模式和架构以整合未充分利用的组件，从而提高整体利用率。停用不再需要的组件。了解工作负载组件的性能，并优化消

耗最多资源的组件。注意客户用来访问您服务的设备，并实施相应的模式以最大限度地减少设备升级需求。

最佳实践

- SUS03-BP01 针对异步和计划作业优化软件和架构
- SUS03-BP02 删除或重构很少或没有使用的工作负载组件
- SUS03-BP03 优化消耗最多时间或资源的代码区域
- SUS03-BP04 优化对设备的影响
- SUS03-BP05 使用最能支持数据访问和存储模式的软件模式和架构

SUS03-BP01 针对异步和计划作业优化软件和架构

使用高效的软件和架构模式（如队列驱动）来保持所部署资源的始终如一的高利用率。

常见反模式：

- 为了应对不可预见的需求高峰，您过度预置云工作负载中的资源。
- 架构不会通过消息传递组件分离异步消息的发送方和接收方。

建立此最佳实践的好处：

- 高效的软件和架构模式可以最大程度地减少工作负载中未使用的资源，并提高整体效率。
- 可以独立于异步消息的接收来扩展处理。
- 通过消息传递组件，可以放宽可用性要求，从而能够用更少的资源来满足这些要求。

在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

使用高效的架构模式，例如事件驱动型架构，这样可以均匀利用组件，并最大限度地减少工作负载中的过度预置。使用高效的架构模式可以最大程度地减少由于需求随时间变化而导致的闲置资源。

了解工作负载组件的要求，并采用可提高资源总体利用率的架构模式。停用不再需要的组件。

实施步骤

- 分析工作负载的需求，以确定如何响应这些需求。

- 对于不需要同步响应的请求或作业，请使用队列驱动型架构和自动扩缩工作线程来最大限度地提高利用率。以下是一些可以考虑采用队列驱动型架构的示例：

排队机制	描述
AWS Batch 作业队列	AWS Batch 作业将提交到作业队列，并一直驻留在队列中，直到可以计划在计算环境中运行。
Amazon Simple Queue Service 和 Amazon EC2 竞价型实例	将 Amazon SQS 实例和竞价型实例配对，构建容错又高效的架构。

- 对于可以随时处理的请求或作业，请使用调度机制批量处理作业以提高效率。以下是 AWS 上的调度机制的一些示例：

调度机制	描述
Amazon EventBridge 调度器	Amazon EventBridge 的一项功能，使您能够大规模创建、运行和管理调度任务。
AWS Glue 基于时间的计划	在 AWS Glue 中为爬网程序和作业定义基于时间的计划。
Amazon Elastic Container Service（Amazon ECS）调度任务	Amazon ECS 支持创建计划任务。计划任务使用 Amazon EventBridge 规则按计划或响应 EventBridge 事件时运行任务。
实例调度器	为您的 Amazon EC2 和 Amazon Relational Database Service 实例配置启动和停止计划。

- 如果在架构中使用轮询和 Webhook 机制，请将它们替换为事件。使用事件驱动型架构来构建高效的工作负载。
- 利用 AWS 上的无服务器架构消除过度预置的基础设施。
- 适当调整架构中各个组件的大小，以防止等待输入的闲置资源。
  - 您可以使用 AWS Cost Explorer 中的合理调整大小建议或 AWS Compute Optimizer 来确定合理调整大小的机会。
  - 有关更多详细信息，请参阅《合理调整大小：预置实例以匹配工作负载》。

资源

相关文档：

- What is Amazon Simple Queue Service?
- 什么是 Amazon MQ？
- 基于 Amazon SQS 进行扩展
- 什么是 AWS Step Functions ？
- 什么是 AWS Lambda ？
- 将 AWS Lambda 与 Amazon SQS 结合使用
- 什么是 Amazon EventBridge？
- 使用 REST API 管理异步工作流程

相关视频：

- AWS re:Invent 2023 - Navigating the journey to serverless event-driven architecture
- AWS re:Invent 2023 - Using serverless for event-driven architecture & domain-driven design
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- 异步消息模式 | AWS 事件

相关示例：

- 带有 AWS Graviton 处理器和 Amazon EC2 竞价型实例的事件驱动型架构

SUS03-BP02 删除或重构很少或没有使用的工作负载组件

移除未使用且不再需要的组件，并重构利用率低的组件，以最大限度减少工作负载中的浪费。

常见反模式：

- 没有定期检查工作负载的各个组件的利用率水平。
- 没有查看和分析来自 AWS 合理调整大小工具（如 AWS Compute Optimizer）的建议。

建立此最佳实践的好处：移除未使用的组建可最大限度减少浪费并提高云工作负载的整体效率。

在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

云工作负载中未使用或未充分利用的组件会消耗不必要的计算、存储或网络资源。移除或重构这些组件，来直接减少浪费并提高云工作负载的整体效率。这是一个迭代改进过程，可以通过需求变化或新云服务的发布来启动。例如，AWS Lambda 函数运行时间的显著减少可能表明需要减少内存大小。此外，随着 AWS 发布新的服务和功能，适用于您的工作负载的最佳服务和架构可能会发生变化。

持续监控工作负载活动并寻找机会来提高单个组件的利用水平。通过删除空闲组件并执行合理调整大小活动，您就可以使用最少的云资源来满足您的业务需求。

实施步骤

- 清点 AWS 资源：创建 AWS 资源的清单。在 AWS 中，您可以开启 AWS 资源探索器 以探索和整理您的 AWS 资源。有关更多详细信息，请参阅 AWS re:Invent 2022 - How to manage resources and applications at scale on AWS。
- 监控利用率：监控和捕获工作负载关键组件的利用率指标（例如 Amazon CloudWatch 指标中的 CPU 利用率、内存利用率或网络吞吐量）。
- 识别未使用的组件：识别架构中未使用或未充分利用的组件。
  - 对于稳定的工作负载，请定期检查 AWS Compute Optimizer 等 AWS 合理调整大小工具，以确定闲置、未使用或未充分利用的组件。
  - 对于临时工作负载，请评估利用率指标以识别空闲、未使用或未充分利用的组件。
- 移除未使用的组件：停用不再需要的组件及关联资产（如 Amazon ECR 映像）。
  - 自动清理 Amazon ECR 中未使用的图片
  - 使用 AWS Config 和 AWS Systems Manager 删除未使用的 Amazon Elastic Block Store（Amazon EBS）卷
- 重构未充分利用的组件：重构未充分利用的组件或将其与其它资源整合，以提高利用效率。例如，您可以在单个 Amazon RDS 数据库实例上预置多个小型数据库，而不必在各个未充分利用的实例上运行数据库。
- 评估改进：了解工作负载为完成工作单元而预置的资源。使用此信息来评估通过移除或重构组件所实现的改进。
  - Measure and track cloud efficiency with sustainability proxy metrics, Part I: What are proxy metrics?
  - Measure and track cloud efficiency with sustainability proxy metrics, Part II: Establish a metrics pipeline

资源

相关文档：

- AWS Trusted Advisor
- 什么是 Amazon CloudWatch？
- 合理调整大小：预置实例以匹配工作负载
- 通过规模优化建议来优化成本

相关视频：

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three

SUS03-BP03 优化消耗最多时间或资源的代码区域

优化在架构的不同组件中运行的代码，以最大限度地减少资源使用和提高性能。

常见反模式：

- 忽略为资源使用优化代码。
- 通常通过增加资源来应对性能问题。
- 代码审核和开发过程不会跟踪性能变化。

建立此最佳实践的好处：使用高效的代码可以最大限度地减少资源使用并提高性能。

在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

至关重要的是检查每个功能区域（包括云架构应用程序的代码）以优化其资源使用和性能。持续监控工作负载在构建环境和生产中的性能，并确定改进资源使用率特别高的代码片段的机会。采用定期审核流程来识别代码中资源使用效率低下的错误或反模式。利用可为您的应用场景产生相同结果的简单和高效算法。

实施步骤

- 使用高效的编程语言：使用高效的操作系统和编程语言来处理工作负载。有关节能编程语言（包括 Rust）的详细信息，请参阅《Sustainability with Rust》。
- 使用 AI 编码伴侣：考虑使用 Amazon Q 开发者版等 AI 编码伴侣来高效编写代码。

- 实现代码审查自动化：在开发工作负载时，采用自动化代码审查流程来提高质量并识别错误和反模式。
  - Automate code reviews with Amazon CodeGuru Reviewer
  - 使用 Amazon CodeGuru 检测并发错误
  - 使用 Amazon CodeGuru 提高 Python 应用程序的代码质量
- 使用代码分析器：使用代码分析器确定使用时间最长或使用资源最多的代码区域作为优化目标。
  - 借助 Amazon CodeGuru Profiler 减少组织的碳排放
  - 使用 Amazon CodeGuru Profiler 了解 Java 应用程序中的内存使用
  - 通过 Amazon CodeGuru Profiler 改进客户体验并降低成本
- 监控和优化：使用持续监控资源来识别资源要求高或配置不佳的组件。
  - 使用可产生相同结果的更简单、更高效算法取代计算密集型算法。
  - 删除排序和格式等不必要的代码。
- 使用代码重构或转换：探索将 Amazon Q 代码转换用于应用程序维护和升级的可能性。
  - 使用 Amazon Q 代码转换升级语言版本
  - AWS re:Invent 2023 - Automate app upgrades & maintenance using Amazon Q Code Transformation

资源

相关文档：

- 什么是 Amazon CodeGuru Profiler？
- FPGA 实例
- 用于在 AWS 上进行构建的 AWS SDK 和工具

相关视频：

- 使用 Amazon CodeGuru Profiler 提高代码效率
- 使用 Amazon CodeGuru 自动提供代码审查和应用程序性能建议

SUS03-BP04 优化对设备的影响

了解您的架构中使用的设备，并使用策略来减少其使用。这可以最大限度地减少云工作负载对环境的整体影响。

常见反模式：

- 忽略客户所用设备对环境的影响。
- 手动管理和更新客户使用的资源。

建立此最佳实践的好处：实施针对客户设备进行了优化的软件模式和功能可以减少云工作负载对环境的总体影响。

在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

实施针对客户设备优化的软件模式和功能可以从几个方面减少对环境的影响。

- 实施向后兼容的新功能可以减少硬件更换次数。
- 优化应用程序以在设备上高效运行，这有助于降低能耗和延长电池寿命（如果它们由电池供电）。
- 针对设备优化应用程序还可以减少网络上的数据传输。

了解架构中使用的设备、其预期生命周期以及更换这些组件产生的影响。实施软件模式和功能，有助于最大程度地降低设备能耗、减少客户更换设备和手动升级设备的需求。

实施步骤

- 进行清点：列出您架构中使用的设备。设备可以是移动设备、平板电脑、物联网设备、智能灯，甚至是工厂中的智能设备。
- 使用节能设备：考虑在架构中使用节能设备。在不使用设备时，使用设备上的电源管理配置来进入低功耗模式。
- 运行高效的应用程序：优化在设备上运行的应用程序：
  - 使用策略（例如在后台运行任务）来降低能耗。
  - 在构建有效负载时考虑网络带宽和延迟，并实施有助于您的应用程序在低带宽、高延迟链路上良好运行的功能。
  - 将有效负载和文件转换为设备所需的优化格式。例如，您可以使用 Amazon Elastic Transcoder 或 AWS Elemental MediaConvert 将大型高质量数字媒体文件转换为用户可在移动设备、平板电脑、网络浏览器和联网电视上播放的格式。
  - 在服务器端执行计算密集型活动（例如图像渲染），或使用应用程序串流来改善旧设备上的用户体验。
  - 对输出进行分段和分页，尤其是对于交互式会话，以管理有效负载并限制本地存储要求。

- 吸引供应商：与使用可持续材料的设备供应商合作，提高供应链透明度和环境认证。
- 使用空中下载（OTA）更新：使用自动化空中下载（OTA）机制将更新部署到一个或多个设备。
  - 您可以使用 CI/CD 管道来更新移动应用程序。
  - 您可以使用 AWS IoT Device Management 大规模远程管理连接的设备。
- 使用托管式设备场：要测试新功能和更新，请使用具有代表性硬件集的托管式设备场，并迭代开发以最大限度增加支持的设备数。有关更多详细信息，请参阅SUS06-BP05 使用托管式 Device Farm 进行测试。
- 继续监控和改进：跟踪设备的能源使用情况，以确定需要改进的领域。使用新技术或最佳实践来改善这些设备对环境的影响。

资源

相关文档：

- 什么是 AWS Device Farm ？
- AppStream 2.0 文档
- NICE DCV
- OTA 教程，用于在运行 FreeRTOS 的设备上更新固件
- 优化您的物联网设备以实现环境可持续性

相关视频：

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm

SUS03-BP05 使用最能支持数据访问和存储模式的软件模式和架构

了解数据在工作负载中的使用方式、用户使用数据的方式，以及数据的传输和存储方式。使用最能支持数据访问和存储的软件模式和架构，最大限度地减少支持工作负载所需的计算、网络和存储资源。

常见反模式：

- 假设所有工作负载都具有相似的数据存储和访问模式。
- 假设所有工作负载都位于一个存储层，且只使用该存储层。
- 假设数据访问模式会随着时间的推移保持一致。
- 您的架构支持潜在的高数据访问突发，这会导致资源大部分时间都处于空闲状态。

建立此最佳实践的好处：根据数据访问和存储模式选择并优化架构将有助于降低开发复杂性并提高总体利用率。了解何时使用全局表、数据分区和缓存将帮助您减少运营开销，并根据您的工作负载需求进行扩展。

在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

要提高工作负载的长期可持续性，请使用支持工作负载数据访问和存储特性的架构模式。这些模式有助于您高效地检索和处理数据。例如，可以将 AWS 上的现代数据架构与针对您的独特分析用例进行了优化的专用服务结合使用。这些架构模式可提高数据处理效率和减少资源使用。

实施步骤

- 了解数据特性：分析您的数据特性和访问规律，以便确定云资源的适合配置。要考虑的主要特性包括：
  - 数据类型：结构化、半结构化、非结构化
  - 数据增长：有限、无界
  - 数据持久性：持久、短暂、瞬时
  - 访问模式:读写、频率、峰值或一致
- 使用最佳架构模式：使用最能支持数据访问和存储规律的架构模式。
  - 启用数据持久性的模式
  - Let's Architect! 现代数据架构
  - AWS 上的数据库：为恰当的作业选择恰当的数据库
- 使用专用服务：使用适合用途的技术。
  - 使用可以原生处理压缩数据的技术。
    - Athena 压缩支持文件格式
    - AWS Glue 中的 ETL 输入和输出的格式选项
    - 使用 Amazon Redshift 从 Amazon S3 加载压缩数据文件
  - 使用专用分析服务在您的架构中进行数据处理。有关 AWS 专用分析服务的详细信息，请参阅 AWS re:Invent 2022 - Building modern data architectures on AWS。
  - 使用最能支持您的主导查询模式的数据库引擎。管理您的数据库索引，来实现高效的查询。有关更多详细信息，请参阅《AWS 数据库》和 AWS re:Invent 2022 - Modernize apps with purpose-built databases。
- 尽量减少数据传输：选择可减少架构中所用网络容量的网络协议。

资源

相关文档：

- 使用 Amazon Redshift 从列数据格式复制
- 在 Firehose 中转换输入记录格式
- 通过转换为列格式提高 Amazon Athena 上的查询性能
- 使用 Amazon Aurora 上的“性能洞察”监控数据库负载
- 在 Amazon RDS 上使用性能详情监控数据库负载
- Amazon S3 Intelligent-Tiering 存储类
- 使用 Amazon DynamoDB 构建 CQRS 事件存储

相关视频：

- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge

相关示例：

- AWS Purpose Built Databases 讲习会
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS

数据

问题

- SUS 4 如何利用数据管理策略和模式来支持可持续性目标？

SUS 4 如何利用数据管理策略和模式来支持可持续性目标？

实施数据管理实践以减少支持工作负载所需的预置存储，以及使用存储所需的资源。了解您的数据，并使用能够更有效地支持数据的业务价值及其使用方式的存储技术和配置。当需求减少时，将数据移到更高效、性能更低的存储中，并删除不再需要的数据。

最佳实践

- SUS04-BP01 实施数据分类策略
- SUS04-BP02 使用支持数据访问和存储模式的技术
- SUS04-BP03 使用策略管理数据集的生命周期
- SUS04-BP04 使用弹性和自动化来扩展数据块存储或文件系统
- SUS04-BP05 删除不需要或多余的数据
- SUS04-BP06 使用共享文件系统或存储来访问通用数据
- SUS04-BP07 最大限度地减少跨网络的数据移动
- SUS04-BP08 仅在难以重新创建时备份数据

SUS04-BP01 实施数据分类策略

对数据进行分类，以了解其对业务成果的重要性，并选择合适的节能存储层来存储数据。

常见反模式：

- 您没有识别正在处理或存储的具有类似特征（如敏感性、业务关键性或监管要求）的数据资产。
- 没有实施数据目录来清点数据资产。

建立这种最佳实践的好处：实施数据分类策略让您可以确定能效最高的数据存储层。

在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

数据分类涉及识别在由组织拥有或运营的信息系统中正在处理和存储的数据类型。它还涉及到对数据的重要性以及数据泄露、丢失或滥用的可能影响进行判断。

实施数据分类策略时，要从数据的使用情境进行反推，并创建一个分类方案，该方案考虑到特定数据集对组织运营的重要程度。

实施步骤

- 进行数据清点：对您的工作负载存在的各种数据类型进行清点。
- 进行数据分组：根据给组织带来的风险，确定数据的重要性、机密性、完整性和可用性。使用这些要求将数据分组到您采用的数据分类层之一。例如，请参阅《对数据进行分类并保护初创企业的四个简单步骤》。
- 定义数据分类级别和策略：为每个数据组定义数据分类级别（例如，公开或机密）和处理策略。对数据做相应标记。有关数据分类类别的更多详情，请参阅《数据分类白皮书》。
- 定期检查：定期检查和审核您的环境中是否有未标记和未分类的数据。使用自动化功能来识别这些数据，并对数据进行适当的分类和标记。例如，请参阅《AWS Glue 中的数据目录和爬网程序》。
- 建立数据目录：建立提供审计和治理功能的数据目录。
- 文档：记录每个数据类别的数据分类策略和处理程序。

资源

相关文档：

- 利用 AWS Cloud 支持数据分类
- 来自 AWS Organizations 的标记策略

相关视频：

- AWS re:Invent 2022 - Enabling agility with data governance on AWS
- AWS re:Invent 2023 - Data protection and resilience with AWS storage

SUS04-BP02 使用支持数据访问和存储模式的技术

使用最能支持您的数据访问和存储方式的存储技术，以在支持您的工作负载的同时最大限度地减少预置资源。

常见反模式：

- 假设所有工作负载都具有相似的数据存储和访问模式。
- 假设所有工作负载都位于一个存储层，且只使用该存储层。
- 假设数据访问模式会随着时间的推移保持一致。

建立此最佳实践的好处：根据数据访问和存储模式选择和优化您的存储技术，有助于您减少满足业务需求所需的云资源，并提高云工作负载的整体效率。

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

选择最适合您的访问模式的存储解决方案，或者考虑根据存储解决方案更改访问模式，以便尽可能提高性能和效率。

实施步骤

- 评估数据和访问特性：评估您的数据特性和访问模式，以收集您的存储需求的主要特性。要考虑的主要特性包括：
  - 数据类型：结构化、半结构化、非结构化
  - 数据增长：有限、无界
  - 数据持久性：持久、短暂、瞬时
  - 访问模式:读写、频率、峰值或一致
- 选择适当的存储技术：将数据迁移到支持您的数据特征和访问模式的适当存储技术。下面是 AWS 存储技术的一些示例以及它们的主要特性：

类型	Technology	主要特性
对象存储	Amazon S3	一项对象存储服务，具有无限的可扩展性、高可用性和多种可访问性选项。在 Amazon S3 内外传输和访问对象，可以使用传输加速或接入点等服务来支持您的位置、安全需求和访问模式。
存档存储	Amazon S3 Glacier	Amazon S3 的存储类，用于数据归档。
共享文件系统	Amazon Elastic File System (Amazon EFS)	可装载文件系统，可由多种类型的计算解决方案访问。Amazon EFS 会自动增大和缩小存

类型	Technology	主要特性
		储，并进行性能优化以提供一致的低延迟。
共享文件系统	Amazon FSx	基于最新 AWS 计算解决方案而构建，支持四种常用文件系统：NetApp ONTAP、OpenZFS、Windows File Server 和 Lustre。Amazon FSx 延迟、吞吐量和 IOPS 因文件系统而不同，因此，在为您的工作负载需求选择合适的文件系统时应考虑这些因素。
数据块存储	Amazon Elastic Block Store（Amazon EBS)	可扩展、高性能的数据块存储服务，专为 Amazon Elastic Compute Cloud（Amazon EC2）设计。Amazon EBS 包括用于事务型、IOPS 密集型工作负载的 SSD 支持型存储，以及用于吞吐量密集型工作负载的 HDD 支持型存储。
关系数据库	Amazon Aurora、Amazon RDS、Amazon Redshift	旨在支持 ACID（原子性、一致性、隔离性、持久性）事务，并保持参照完整性和数据强一致性。许多传统应用程序、企业资源规划（ERP）、客户关系管理（CRM）和电子商务系统都使用关系数据库来存储数据。

类型	Technology	主要特性
键值数据库	Amazon DynamoDB	已针对常见的访问模式进行优化，通常用于存储和检索大量数据。键值数据库的典型使用案例包括高流量 Web 应用程序、电子商务系统和游戏应用程序。

- 自动分配存储空间：对于固定大小的存储系统（例如 Amazon EBS 或 Amazon FSx），请监控可用的存储空间，并在达到阈值时自动分配存储空间。您可以利用 Amazon CloudWatch 来收集和分析 Amazon EBS 和 Amazon FSx 的不同指标。
- 选择合适的存储类：为您的数据选择合适的存储类。
  - Amazon S3 可以在对象级别配置存储类。单个存储桶可以包含存储在所有存储类中的对象。
  - 您可以使用 Amazon S3 生命周期策略，在不对应用程序进行任何更改的情况下，于存储类之间自动转换对象或删除数据。通常来说，在考虑这些存储机制时，您必须在资源效率、访问延迟和可靠性之间做出取舍。

资源

相关文档：

- Amazon EBS 卷类型
- Amazon EC2 实例存储
- Amazon S3 Intelligent-Tiering
- Amazon EBS I/O 特性
- 使用 Amazon S3 存储类
- 什么是 Amazon S3 Glacier？

相关视频：

- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2022 - Building modern data architectures on AWS

- AWS re:Invent 2022 - Modernize apps with purpose-built databases
- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023 - Advanced data modeling with Amazon DynamoDB

相关示例：

- Amazon S3 示例
- AWS Purpose Built Databases 讲习会
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS

SUS04-BP03 使用策略管理数据集的生命周期

管理所有数据的生命周期并自动执行删除，以最大限度地减少工作负载所需的总存储。

常见反模式：

- 手动删除数据。
- 不删除任何工作负载数据。
- 不根据数据的保留和访问要求将数据移动到更节能的存储层。

建立此最佳实践的好处：使用数据生命周期策略可确保在工作负载中高效地访问和保留数据。

在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

数据集在其生命周期中通常具有不同的保留和访问要求。例如，应用程序可能需要在有限的时间段内频繁访问某些数据集。之后，这些数据集很少被访问。要随时间推移提高数据存储和计算的效率，请实施生命周期策略，这些策略是定义如何随时间推移来处理数据的规则。

使用生命周期配置规则，您可以指示特定存储服务将数据集转换到更节能的存储层、将其归档或删除。这种做法可最大限度地减少主动数据存储和检索，从而降低能耗。此外，归档或删除过时数据等做法支持法规遵从性和数据治理。

实施步骤

- 使用数据分类：对工作负载中的数据集进行分类。
- 定义处理规则：定义每个数据类的处理过程。
- 启用自动化：设置自动化生命周期策略以强制实施生命周期规则。以下是如何为不同 AWS 存储服务设置自动化生命周期策略的一些示例：

存储服务	如何设置自动化生命周期策略
Amazon S3	您可以使用 Amazon S3 生命周期在对象的整个生命周期中对其进行管理。如果您的访问模式未知、变化或不可预测，则可以使用 Amazon S3 Intelligent-Tiering，它能够监控访问模式并自动将尚未访问的对象移动到成本较低的访问层。您可以利用 Amazon S3 Storage Lens 存储统计管理工具指标来识别生命周期管理中的优化机会和差距。
Amazon Elastic Block Store	您可以使用 Amazon Data Lifecycle Manager 自动创建、保留和删除 Amazon EBS 快照和 Amazon EBS 支持的 AMI。
Amazon Elastic File System	Amazon EFS 生命周期管理会自动为您的文件系统管理文件存储。
Amazon Elastic Container Registry	Amazon ECR 生命周期策略可根据存在期限或计数使映像过期，以此自动清理容器映像。
AWS Elemental MediaStore	您可以使用对象生命周期策略，用于管理对象在 MediaStore 容器中应该存储多长时间。

- 删除未使用的资产：删除未使用的卷、快照和超出保留期的数据。使用原生服务功能（如 Amazon DynamoDB 生存时间或 Amazon CloudWatch log retention）进行删除。
- 聚合和压缩：在适当的情况下根据生命周期规则聚合和压缩数据。

资源

相关文档：

- 使用 Amazon S3 存储类分析优化您的 Amazon S3 生命周期规则
- 使用 AWS Config 规则 评估资源

相关视频：

- AWS re:Invent 2021 - Amazon S3 Lifecycle best practices to optimize your storage spend
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- 利用 Amazon S3 生命周期简化您的数据生命周期并优化存储成本
- 使用 Amazon S3 Storage Lens 存储统计管理工具减少您的存储成本

SUS04-BP04 使用弹性和自动化来扩展数据块存储或文件系统

随着数据的增长，使用弹性和自动化来扩展数据块存储或文件系统，以便最大限度减少总预置存储。

常见反模式：

- 购买大型数据块存储或文件系统以备将来需要。
- 过度预置文件系统的每秒输入和输出操作数（IOPS）。
- 不监控数据卷的利用率。

建立此最佳实践的好处：最大限度地减少存储系统的过度预置可以减少闲置资源并提高工作负载的整体效率。

在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

根据适合工作负载的大小分配、吞吐量和延迟，创建数据块存储和文件系统。随着数据的增长，使用弹性和自动化来扩展数据块存储或文件系统，而无需过度预置这些存储服务。

实施步骤

- 对于固定大小的存储系统（例如 Amazon EBS），请确保您正在监控使用的存储量与总体存储量大小之间的关系，可能的话创建自动化，以便在达到阈值时增加存储大小
- 使用弹性卷和托管式数据块数据服务，随着持久性数据的增长自动分配额外的存储。例如，您可以使用 Amazon EBS 弹性卷来更改卷大小、卷类型或调整 Amazon EBS 卷的性能。

- 为您的文件系统选择适合的存储类、性能模式和吞吐量模式，以满足您的业务需求，不要超过这个需求。
  - Amazon EFS 性能
  - Linux 实例上的 Amazon EBS 卷性能
- 为您的数据卷设置目标利用率水平，并调整超出预期范围的卷大小。
- 合理调整只读卷的大小以适应数据。
- 将数据迁移到对象存储，以避免使用数据块存储上的固定卷大小预置多余容量。
- 定期检查弹性卷和文件系统，终止空闲卷并缩减过度预置的资源，以适应当前数据大小。

资源

相关文档：

- 调整 EBS 卷大小后扩展文件系统
- 使用 Amazon EBS 弹性卷修改卷
- Amazon FSx 文档
- 什么是 Amazon Elastic File System？

相关视频：

- 深入了解 Amazon EBS 弹性卷
- 用于提高性能和节省成本的 Amazon EBS 和快照优化策略
- 使用最佳实践优化 Amazon EFS 的成本和性能

SUS04-BP05 删除不需要或多余的数据

删除不需要或多余的数据，以最大程度地减少存储数据集所需的存储资源。

常见反模式：

- 复制可以轻松获取或重新创建的数据。
- 备份所有数据时不考虑其重要性。
- 只不定期地删除数据、操作事件时删除数据，或者根本不删除数据。
- 无论存储服务的持久性如何，都冗余地存储数据。

- 您在没有任何业务理由的情况下启用 Amazon S3 版本控制。

建立此最佳实践的好处：删除不需要的数据可以减少工作负载所需的存储大小和工作负载对环境的影响。

在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

当您移除不需要和冗余的数据集时，可以降低存储成本和环境足迹。这种做法还可以提高计算效率，因为计算资源只处理重要的数据，而不处理不需要的数据。自动删除不需要的数据。使用技术在文件和数据块级别进行重复数据删除。使用服务功能来实现原生数据复制和冗余。

实施步骤

- 评估公开数据集：评估是否可以通过使用 AWS Data Exchange 和 Open Data on AWS 中现有公开可用的数据集来避免存储数据。
- 删除重复数据：使用可以在数据块和对象级别删除重复数据的机制。以下是有关如何删除 AWS 上的重复数据的一些示例：

存储服务	重复数据删除机制
Amazon S3	使用新的 FindMatches 机器学习转换，使用 AWS Lake Formation FindMatches 在数据集中查找匹配的记录（包括没有标识符的记录）。
Amazon FSx	使用 Amazon FSx for Windows 上的重复数据删除。
Amazon Elastic Block Store 快照	快照属于增量 备份，这意味着仅保存设备上在最新快照之后更改的数据块。

- 使用生命周期策略：使用生命周期策略来自动删除不需要的数据。使用原生服务功能（如 Amazon DynamoDB 生存时间、Amazon S3 生命周期或 Amazon CloudWatch log retention）进行删除。
- 使用数据虚拟化：使用 AWS 上的数据虚拟化功能在源头维护数据并避免数据重复。
  - AWS 上的云原生数据虚拟化
  - Optimize Data Pattern Using Amazon Redshift Data Sharing

- 使用增量备份：使用可进行增量备份的备份技术。
- 使用原生持久性：利用 Amazon S3 的持久性和 Amazon EBS 的复制来实现您的持久性目标，而不是使用自行管理的技术 [例如独立磁盘冗余阵列（RAID）]。
- 使用高效的日志记录：集中日志和跟踪数据，对相同的日志条目进行重复数据删除，并在需要时建立调整详细程度的机制。
- 使用高效的缓存：仅在合理的情况下预填充缓存。
- 建立缓存监控和自动化以相应地调整缓存大小。
- 移除旧的版本资产：推送新版本的工作负载时，从对象存储和边缘缓存中移除过时的部署和资产。

资源

相关文档：

- 更改 CloudWatch Logs 中的日志数据留存
- 适用于 Windows File Server 的 Amazon FSx 的重复数据删除
- Amazon FSx for ONTAP 的功能，包括重复数据删除
- 使 Amazon CloudFront 上的文件失效
- 使用 AWS Backup 备份和恢复 Amazon EFS 文件系统
- 什么是 Amazon CloudWatch Logs？
- 在 Amazon RDS 上使用备份
- 使用 AWS Lake Formation 集成数据集并删除其中的重复数据

相关视频：

- Amazon Redshift 数据共享用例

相关示例：

- 如何使用 Amazon Athena 分析我的 Amazon S3 服务器访问日志？

SUS04-BP06 使用共享文件系统或存储来访问通用数据

采用共享文件系统或存储来避免数据重复，并可为工作负载提供更高效的基础设施。

常见反模式：

- 为每个客户端预置存储。
- 未卸下不活动的客户端的数据卷。
- 不提供跨平台和系统的存储访问。

建立此最佳实践的好处：使用共享文件系统或存储可以将数据共享给一个或多个使用者，而无需复制数据。这有助于减少工作负载所需的存储资源。

在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

如果您有多个用户或应用程序访问同一个数据集，则使用共享存储技术很重要，这可以让工作负载高效地使用基础设施。共享存储技术提供一个位置来集中存储和管理数据集并避免数据重复。它还加强了不同系统之间数据的一致性。此外，因为多个计算资源会同时并行访问和处理数据，所以利用共享存储技术可以更高效地使用计算能力。

仅在需要时才从这些共享存储服务中提取数据，并卸下未使用的卷以释放资源。

实施步骤

- 使用共享存储：当数据具有多个使用者时，将数据迁移到共享存储。下面是 AWS 上的共享存储技术的一些示例：

存储选项	何时使用
Amazon EBS 多重挂载	Amazon EBS 多重挂载让您可以将单个预调配 IOPS SSD（io1 或 io2）卷挂载到同一可用区中的多个实例。
Amazon EFS	请参阅《何时选择 Amazon EFS》。
Amazon FSx	请参阅《选择 Amazon FSx 文件系统》。
Amazon S3	不需要文件系统结构而旨在与对象存储一起使用的应用程序可以使用 Amazon S3 作为可大规模扩展、持久、低成本的对象存储解决方案。

- 根据需要提取数据：仅在需要时，才将数据复制到共享文件系统或从共享文件系统提取数据。例如，您可以创建由 Amazon S3 支持的适用于 Lustre 的 Amazon FSx 文件系统，并且只将处理任务所需的数据子集加载到 Amazon FSx。
- 删除不需要的数据：根据您的使用规律适当删除数据，如 SUS04-BP03 使用策略管理数据集的生命周期中所述。
- 分离不活动的客户端：将卷与未积极使用它们的客户端分离。

资源

相关文档：

- 将文件系统链接到 Amazon S3 存储桶
- 在您的无服务器应用程序中使用 AWS Lambda Amazon EFS
- Amazon EFS Intelligent-Tiering 通过不断变化的访问模式来优化工作负载的成本
- 将 Amazon FSx 用于您的本地数据存储库

相关视频：

- 使用 Amazon EFS 优化存储成本
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - File storage for builders and data scientists on Amazon Elastic File System

SUS04-BP07 最大限度地减少跨网络的数据移动

使用共享文件系统或对象存储来访问通用数据，并最大限度地减少支持工作负载数据移动所需的总网络资源。

常见反模式：

- 不管数据用户位于何处，将所有数据存储在同一个 AWS 区域。
- 在网络中移动数据之前不优化数据大小和格式。

建立此最佳实践的好处：优化跨网络的数据移动可以减少工作负载所需的总网络资源，并降低对环境的影响。

在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

在组织中移动数据需要计算、网络和存储资源。使用相应的技术最大程度地减少数据移动并提高工作负载的整体效率。

实施步骤

- 使用邻近性：在 selecting a Region for your workload 时，请考虑将与数据或用户的距离作为一项决定因素。
- 对服务进行分区：对按区域使用的服务进行分区，以便将其特定于区域的数据存储在使用它的区域内。
- 使用高效的文件格式：使用高效的文件格式（如 Parquet 或 ORC），并在通过网络移动数据之前先对其进行压缩。
- 尽量减少数据移动：不移动未使用的数据。一些让您能够避免移动未使用数据的示例：
  - 将 API 响应缩减到仅针对相关数据。
  - 聚合详细数据（不需要记录级别信息）。
  - 请参阅 Well-Architected Lab - Optimize Data Pattern Using Amazon Redshift Data Sharing。
  - 考虑 AWS Lake Formation 中的跨账户数据共享。
- 使用边缘服务：使用有助于您在更接近工作负载用户的位置运行代码的服务。

服务	何时使用
Lambda@Edge	用于计算密集型操作，当对象不在缓存中时会运行这些操作。
CloudFront Functions	用于简单使用场景（如 HTTP(S) 请求/响应操作），这些操作可由短期运行的函数启动。
AWS IoT Greengrass	为互联设备运行本地计算、消息收发和数据缓存。

资源

相关文档：

- 优化您的 AWS 基础设施以实现可持续性，第 III 部分：联网
- AWS 全球基础设施

- Amazon CloudFront 主要功能，包括 CloudFront 全球边缘网络
- 在 Amazon OpenSearch Service 中压缩 HTTP 请求
- 使用 Amazon EMR 进行中间数据压缩
- 从 Amazon S3 将压缩数据文件加载到 Amazon Redshift 中
- 通过 Amazon CloudFront 提供压缩文件

相关视频：

- 揭秘 AWS 上的数据传输

SUS04-BP08 仅在难以重新创建时备份数据

避免备份没有商业价值的数据，尽量减少工作负载的存储资源需求。

常见反模式：

- 没有为数据制定备份策略。
- 备份可以轻松重新创建的数据。

建立此最佳实践的好处：避免备份非关键数据可以减少工作负载所需的存储资源并降低其对环境的影响。

在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

避免备份不必要的数据有助于降低成本和减少工作负载使用的存储资源。仅备份具有商业价值或满足合规性要求所必需的数据。检查备份策略并在恢复方案中排除没有价值的临时存储。

实施步骤

- 对数据进行分类：实施 SUS04-BP01 实施数据分类策略中概述的数据分类策略。
- 设计备份策略：使用数据分类的重要性，并根据恢复时间目标（RTO）和恢复点目标（RPO）来设计备份策略。避免备份非关键数据。
  - 排除可以轻松重新创建的数据。
  - 从备份中排除临时数据。
  - 排除数据的本地副本，除非从公共位置恢复该数据所需的时间会超过您的服务等级协议（SLA）。

- 使用自动备份：使用自动化解决方案或托管式服务来备份业务关键的数据。
  - AWS Backup 是一项完全托管式服务，助您轻松地在云中以及在本地集中管理和自动执行跨 AWS 服务的数据保护。有关如何使用 AWS Backup 创建自动备份的动手实践指导，请参阅 Well-Architected Lab - Testing Backup and Restore of Data。
  - 使用 AWS Backup 自动执行 Amazon EFS 备份并优化备份成本。

资源

相关最佳实践：

- REL09-BP01 识别并备份需要备份的所有数据或从源复制数据
- REL09-BP03 自动执行数据备份
- REL13-BP02 使用定义的恢复策略来实现恢复目标

相关文档：

- 使用 AWS Backup 备份和恢复 Amazon EFS 文件系统
- Amazon EBS 快照
- 使用 Amazon Relational Database Service 上的备份
- APN 合作伙伴：可帮助进行备份的合作伙伴
- AWS Marketplace：可用于备份的产品
- Backing Up Amazon EFS
- Backing Up Amazon FSx for Windows File Server
- Amazon ElastiCache (Redis OSS) 的备份和还原

相关视频：

- AWS re:Invent 2023 - Backup and disaster recovery strategies for increased resilience
- AWS re:Invent 2023 - What's new with AWS Backup
- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS

硬件和服务

问题

- SUS 5 如何选择并使用架构中的云硬件和服务来支持可持续性目标？

SUS 5 如何选择并使用架构中的云硬件和服务来支持可持续性目标？

寻找机会，通过更改硬件管理实践来降低工作负载可持续性影响。最大限度地减少预置和部署所需的硬件数量，并为各项工作负载选择最高效的硬件和服务。

最佳实践

- SUS05-BP01 使用最少的硬件来满足您的需求
- SUS05-BP02 使用影响最小的实例类型
- SUS05-BP03 使用托管服务
- SUS05-BP04 优化基于硬件的计算加速器的使用

SUS05-BP01 使用最少的硬件来满足您的需求

为您的工作负载使用最少的硬件，高效地满足您的业务需求。

常见反模式：

- 不监控资源使用率。
- 架构中有利用率较低的资源。
- 没有检查静态硬件的利用率以确定是否应调整大小。
- 没有根据业务 KPI 为计算基础设施设置硬件利用率目标。

建立此最佳实践的好处：合理调整云资源的大小有助于减少工作负载对环境的影响，节省资金，并维护性能基准。

在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

以最佳方式选择工作负载所需的硬件总数，以提高其整体效率。AWS Cloud 让您能够通过各种机制（例如 AWS Auto Scaling）灵活地动态扩展或缩减资源，以便满足不断变化的需求。它还提供 API 和 SDK，让您可以轻松修改资源。使用这些功能经常更改工作负载实施。此外，按照 AWS 工具中的合理调整大小准则高效地运营您的云资源和满足您的业务需求。

实施步骤

- 选择实例类型：选择最适合您需求的正确实例类型。要了解如何选择 Amazon Elastic Compute Cloud 实例以及如何使用基于属性的实例选择等机制，请参阅以下内容：
  - 如何为我的工作负载选择适当的 Amazon EC2 实例类型？
  - Amazon EC2 Fleet 的基于属性的实例类型选择。
  - 示例：使用基于属性的实例类型选择创建自动扩缩组。
- 扩展：通过小增量扩缩来扩展可变的工作负载。
- 使用多种计算购买选项：在实例灵活性、可扩展性和成本节省与多种计算购买选项之间取得平衡。
  - Amazon EC2 按需型实例最适合实例类型、位置或时间不灵活的新型、有状态和突增工作负载。
  - Amazon EC2 竞价型实例是为容错且灵活的应用程序补充其他选项的好方法。
  - 利用计算类节省计划来处理稳定状态的工作负载，以便在您的需求（例如可用区、区域、实例系列或实例类型）发生变化时提供灵活性。
- 使用实例和可用区的多样性：通过多样化您的实例和可用区，最大限度地提高应用程序可用性并利用多余的容量。
- 合理调整实例的大小：使用来自 AWS 工具的合理调整大小建议来调整工作负载。有关更多信息，请参阅《Optimizing your cost with Rightsizing Recommendations》和《合理调整大小：预置实例以匹配工作负载》
  - 使用 AWS Cost Explorer 中的合理调整大小建议或 AWS Compute Optimizer 来确定合理调整大小的机会。
- 协商服务水平协议（SLA）：协商 SLA，允许暂时减少容量，同时利用自动化功能部署替换资源。

资源

相关文档：

- 优化您的 AWS 基础设施以实现可持续性，第 I 部分：计算
- 基于属性选择实例类型用于 Amazon EC2 Fleet 的自动扩缩
- AWS Compute Optimizer 文档
- 运行 Lambda：性能优化
- 自动扩缩文档

相关视频：

- AWS re:Invent 2023 - What's new with Amazon EC2
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies

- AWS re:Invent 2022 - Optimizing Amazon Elastic Kubernetes Service for performance and cost on AWS
- AWS re:Invent 2023 - Sustainable compute: reducing costs and carbon emissions with AWS

SUS05-BP02 使用影响最小的实例类型

持续监控和使用新实例类型以充分利用能源效率改进。

常见反模式：

- 您只使用一个系列的实例。
- 您只使用 x86 实例。
- 您在 Amazon EC2 Auto Scaling 配置中指定一种实例类型。
- 您使用 AWS 实例的方式与其预期用途不匹配（例如，您将计算优化的实例用于内存密集型工作负载）。
- 您没有定期评估新的实例类型。
- 您不查看 AWS 合理调整大小工具（如 AWS Compute Optimizer）提供的建议。

建立此最佳实践的好处：通过使用节能且大小合适的实例，您可以大大减小工作负载对环境的影响并降低其成本。

在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

在云工作负载中使用高效的实例对于降低资源使用率和成本效益至关重要。持续监控新实例类型的发布并利用能效改进，包括那些旨在支持特定工作负载（例如机器学习训练和推理以及视频转码）的实例类型。

实施步骤

- 学习和探索实例类型：学习和探索可以减小工作负载对环境影响的实例类型。
  - 订阅 AWS 的新功能，随时了解最新 AWS 技术和实例的动态。
  - 了解不同的 AWS 实例类型。
  - 通过观看如下视频，了解基于 AWS Graviton 的实例（这些实例在 Amazon EC2 中每瓦能耗方面提供出色性能）：re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances 和 Deep dive into AWS Graviton3 and Amazon EC2 C7g instances。

- 使用影响最小的实例类型：规划工作负载并将其转换为影响极小的实例类型。
  - 定义一个流程来评估工作负载的新功能或实例。利用云中的敏捷性，快速测试新的实例类型如何改善工作负载的环境可持续性。使用代理指标来衡量完成一个单元的工作需要多少资源。
  - 如有可能，修改工作负载以使用不同数量的 vCPU 和不同数量的内存，以最大限度地增加您的实例类型选项。
  - 考虑将工作负载转换为基于 Graviton 的实例，以提高工作负载的性能效率。有关将工作负载迁移到 AWS Graviton 的更多信息，请参阅《AWS Graviton 使用快速入门和将工作负载过渡到基于 AWS Graviton 的 Amazon Elastic Compute Cloud 实例时的注意事项。
  - 考虑选择 AWS Graviton 选项（在使用 AWS 托管服务时）。
  - 将工作负载迁移到提供对可持续性影响极小的实例且仍满足您的业务要求的区域。
  - 对于机器学习工作负载，请利用特定于工作负载的专用硬件，例如 AWS Trainium、AWS Inferentia 和 Amazon EC2 DL1。AWSInf2 实例等 Inferentia 实例相比同类 Amazon EC2 实例，性能功耗比提升了 50%。
  - 使用 Amazon SageMaker AI Inference Recommender 来合理调整机器学习推理端点的大小。
  - 对于突增工作负载（不经常需要额外容量的工作负载），请使用可突增性能实例。
  - 对于无状态和容错工作负载，请使用 Amazon EC2 竞价型实例用于无状态和容错工作负载，以提高云的整体利用率并减少未使用资源对可持续性的影响。
- 运营和优化：运营和优化您的工作负载实例。
  - 对于临时工作负载，请评估实例 Amazon CloudWatch 指标（例如 `CPUUtilization`），以确定实例是空闲还是未充分利用。
  - 对于稳定工作负载，请定期检查 AWS 合理调整规模工具（如 AWS Compute Optimizer），从而挖掘优化实例和合理调整实例大小的机会。有关更多示例和推荐，请参阅以下实验：
    - Well-Architected Lab – 合理调整大小建议
    - Well-Architected Lab – 使用 Compute Optimizer 合理调整大小
    - Well-Architected Lab – 优化硬件模式并观察可持续性 KPI

资源

相关文档：

- 优化您的 AWS 基础设施以实现可持续性，第 I 部分：计算
- AWS Graviton
- Amazon EC2 DL1

- Amazon EC2 容量预留实例集
- Amazon EC2 竞价型实例集
- 函数：Lambda 函数配置
- Amazon EC2 Fleet 的基于属性的实例类型选择
- 在 AWS 上构建可持续、高效且优化成本的应用程序
- Contino 可持续性控制面板如何助力客户减少碳排放

相关视频：

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 - What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Deep dive into AWS Graviton3 and Amazon EC2 C7g instances
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

相关示例：

- 解决方案：关于在 AWS 上优化深度学习工作负载以实现可持续性的指导

SUS05-BP03 使用托管服务

使用托管服务在云中更高效地运营。

常见反模式：

- 使用利用率低的 Amazon EC2 实例来运行应用程序。
- 内部团队仅管理工作负载，而没有时间专注于创新或简化。
- 为可在托管服务上更高效运行的任务部署和维护技术。

建立此最佳实践的好处：

- 使用托管服务将责任转移给 AWS，其拥有对数百万客户的洞察，可以帮助推动新的创新和提高效率。

- 由于使用了多租户控制面板，托管服务将服务的环境影响分散到许多用户。

在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

托管服务将维持已部署硬件的高利用率和可持续性优化的责任转移给 AWS。托管服务还消除了维护服务的运营和管理负担，让您的团队有更多时间专注于创新。

审核您的工作负载，以便确定可由 AWS 托管服务替换的组件。例如，Amazon RDS、Amazon Redshift 和 Amazon ElastiCache 提供托管式数据库服务。Amazon Athena、Amazon EMR 和 Amazon OpenSearch Service 提供托管式分析服务。

实施步骤

1. 清点工作负载：清点工作负载的服务和组件。
2. 识别候选对象：评测和确定可由托管服务替换的组件。以下是一些可以考虑采用托管服务的示例：

任务	在 AWS 上使用什么
托管数据库	使用托管的 Amazon Relational Database Service（Amazon RDS）实例，而不是在 Amazon Elastic Compute Cloud（Amazon EC2）上维护您自己的 Amazon RDS 实例
托管容器工作负载	使用 AWS Fargate，而不是实施自己的容器基础设施。
托管 Web 应用	使用 AWS Amplify 托管作为完全托管式 CI/CD 以及静态网站和服务器端渲染的 Web 应用程序的托管服务。

3. 制定迁移计划：确定依赖关系并制定迁移计划。相应地更新运行手册和行动手册。
   - AWS Application Discovery Service 会自动收集并提供有关应用程序依赖关系和使用情况的详细信息，帮助您在制定迁移计划时做出更明智的决策。
4. 执行测试：迁移到托管服务之前测试服务。
5. 替换自管式服务：使用您的迁移计划将自管式服务替换为托管服务。
6. 监控和调整：迁移完成后持续监控服务，以便根据需要进行调整并优化服务。

资源

相关文档：

- AWS Cloud 产品
- AWS 总拥有成本（TCO）计算器
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service（EKS）
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

相关视频：

- AWS re:Invent 2021 - Cloud operations at scale with AWS Managed Services
- AWS re:Invent 2023 - Best practices for operating on AWS

SUS05-BP04 优化基于硬件的计算加速器的使用

优化加速型计算实例的使用，以减少工作负载的物理基础架构需求。

常见反模式：

- 不监控 GPU 使用情况。
- 将通用实例用于工作负载，而专用实例可以提供更高的性能、更低的成本和更高的性能功耗比。
- 使用基于硬件的计算加速器来完成任务，而使用基于 CPU 的替代方案能更高效地完成任务。

建立此最佳实践的好处：通过优化基于硬件的加速器的使用，您能够减少工作负载对物理基础设施的需求。

在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

如果需要高处理能力，可以受益于使用加速型计算实例，这些实例提供对基于硬件的计算加速器的访问，例如图形处理单元（GPU）和现场可编程门阵列（FPGA）。这些硬件加速器能够比基于 CPU 的替代方案更有效地执行某些功能，例如图形处理或数据模式匹配。许多加速工作负载（如渲染、转码和机器学习）在资源使用方面变化很大。仅在需要时运行此硬件，并在不需要时自动停用它们，以最大限度地减少资源消耗。

实施步骤

- 探索计算加速器：确定可以满足要求的加速型计算实例。
- 使用专用硬件：对于机器学习工作负载，利用特定于工作负载的专用硬件，例如 AWS Trainium、AWS Inferentia 和 Amazon EC2 DL1。AWSInf2 实例等 Inferentia 实例相比同类 Amazon EC2 实例，性能功耗比提升了 50%。
- 监控使用情况指标：收集加速型计算实例的使用情况指标。例如，按照使用 Amazon CloudWatch 收集 NVIDIA GPU 指标所述，使用 CloudWatch 代理收集 GPU 的 `utilization_gpu` 和 `utilization_memory` 等指标。
- 规模适中：优化硬件加速器的代码、网络运营和设置，来确保底层硬件得到充分利用。
  - 优化 GPU 设置
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker AI
- 保持最新：使用最新的高性能库和 GPU 驱动程序。
- 释放不需要的实例：使用自动化功能在不使用 GPU 实例时将其释放。

资源

相关文档：

- 加速计算
- Let's Architect! Architecting with custom chips and accelerators
- 如何为我的工作负载选择适当的 Amazon EC2 实例类型？
- Amazon EC2 VT1 Instances
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

相关视频：

- AWS re:Invent 2021 - How to select Amazon EC2 GPU instances for deep learning
- AWS 在线技术讲座 – 部署经济高效的深度学习推理
- AWS re:Invent 2023 - Cutting-edge AI with AWS and NVIDIA
- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances

- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

流程和文化

问题

- SUS 6 组织流程如何支持可持续性目标？

SUS 6 组织流程如何支持可持续性目标？

寻找机会，通过更改开发、测试和部署实践来降低可持续性影响。

最佳实践

- SUS06-BP01 宣传和传达可持续发展目标
- SUS06-BP02 采用可以快速引入可持续性改进的方法
- SUS06-BP03 让您的工作负载保持最新状态
- SUS06-BP04 提高构建环境的利用率
- SUS06-BP05 使用托管式 Device Farm 进行测试

SUS06-BP01 宣传和传达可持续发展目标

技术是可持续发展的关键推动力。IT 团队在推动有意义的变革来实现组织的可持续发展目标方面，发挥着至关重要的作用。这些团队应清楚地了解公司的可持续发展目标，并努力在公司运营过程中宣传和传达这些优先事项。

常见反模式：

- 您不知道组织的可持续发展目标以及这些目标如何应用于您的团队。
- 您对云工作负载带来的环境影响缺乏足够的认识和培训。
- 您不确定要优先考虑的具体领域。
- 您不让员工和客户参与您的可持续发展计划。

建立此最佳实践的好处：从优化基础设施和系统到使用创新技术，IT 团队可以减少组织的碳排放，并最大限度地减少资源消耗。大力宣传可持续发展目标，可让 IT 团队持续改进，并应对不断变化的可持续发展挑战。此外，这些可持续的优化通常也会转化为成本节约，从而加强业务案例。

在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

IT 团队的主要可持续发展目标应该是优化系统和解决方案来提高资源效率，并最大限度地减少组织的碳足迹和整体环境影响。共享的服务和计划（例如培训计划和运营控制面板）可以支持组织优化 IT 运营，并构建有助于显著减少碳足迹的解决方案。云技术提供了一个机会，不仅可以将物理基础设施和能源采购责任转换为云提供商的共同责任，还可以持续优化基于云的服务的资源效率。

当团队使用云固有的效率和责任共担模式时，可以推动组织显著减少对环境的影响。反过来，这可以为组织的整体可持续发展目标做出贡献，并证明这些团队作为战略合作伙伴在迈向更可持续的未来之旅中的价值。

实施步骤

- 定义目标和目的：为您的 IT 项目设定明确的目标。这包括征求来自不同部门（例如 IT、可持续发展和财务）的负有相应责任的利益相关者的意见。这些团队应定义与贵组织的可持续发展目标相一致的可衡量目标，包括碳减排和资源优化等领域。
- 了解企业的碳核算边界：了解温室气体（GHG）协议等碳核算方法与云中工作负载的关系（有关更多详细信息，请参阅云可持续性）。
- 使用云解决方案进行碳核算：使用 carbon accounting solutions on AWS 等云解决方案，来跟踪运营、投资组合和价值链中 GHG 排放的范围一、二和三。借助这些解决方案，组织可以简化 GHG 排放数据的采集、简化报告并获得见解，从而为其气候策略提供信息。
- 监控 IT 产品组合的碳足迹：跟踪和报告 IT 系统的碳排放。使用 AWS 客户碳足迹工具来跟踪、衡量、审查和预测您使用 AWS 所产生的碳排放。
- 通过代理指标向您的团队传达资源使用情况：通过代理指标跟踪和报告资源使用情况。在云的按需定价模型中，资源使用情况与成本相关，这是一个普遍可以理解的指标。至少应使用成本作为代理指标，来传达每个团队的资源使用情况和改进情况。
  - 在 Cost Explorer 成本管理服务中启用每小时粒度，并创建成本和使用情况报告（CUR）：CUR 为所有 AWS 服务提供每日或每小时的使用情况粒度、费率、成本和使用情况属性。使用 Cloud Intelligence Dashboards 及其可持续性发展代理指标控制面板，作为处理和可视化基于成本和使用情况的数据的起点。有关更多详细信息，请参阅以下内容：
  - Measure and track cloud efficiency with sustainability proxy metrics, Part I: What are proxy metrics?
  - Measure and track cloud efficiency with sustainability proxy metrics, Part II: Establish a metrics pipeline

- 持续优化和评估：使用改进流程来持续优化您的 IT 系统，包括云工作负载，以便提高效率和可持续性。在实施优化策略之前和之后监控碳足迹。利用碳足迹的减少来评测有效性。
- 培养可持续发展文化：使用培训计划（如 AWS Skill Builder）来对员工进行可持续发展教育。让他们参与可持续发展计划。分享并庆祝他们的成功故事。如果他们实现了可持续发展目标，则使用激励措施来奖励他们。

资源

相关文档：

- 了解碳排放估算

相关视频：

- AWS re:Invent 2023 - Accelerate data-driven circular economy initiatives with AWS
- AWS re:Invent 2023 - Sustainability innovation in AWS Global Infrastructure
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure

相关示例：

- Well-Architected Lab – 将成本和使用情况报告转化为效率报告

相关培训：

- Sustainability Transformation on AWS
- SimuLearn - Sustainability Reporting
- Decarbonization with AWS

SUS06-BP02 采用可以快速引入可持续性改进的方法

采用方法和流程来验证潜在的改进、最大限度降低测试成本和带来一些小改进。

常见反模式：

- 仅在项目开始时才完成一次审核应用程序可持续性。
- 工作负载变得过时，因为发布过程过于繁琐，无法为提高资源效率而引入微小的更改。
- 未制定相应的机制来提高工作负载的可持续性。

建立这种最佳实践的好处：通过建立引入和跟踪可持续性改进的流程，您将能够不断采用新的功能和能力，消除问题并提高工作负载的效率。

在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

在将潜在可持续性改进部署到生产环境之前，对其进行测试和验证。在计算改进的潜在未来收益时，考虑测试成本。开发低成本的测试方法，以实现细微的改进。

实施步骤

- 了解并传达组织的可持续性目标：了解组织的可持续性目标，例如碳减排或水资源管理。将这些目标转化为对云工作负载的可持续性要求。将这些要求传达给主要利益相关方。
- 将可持续性要求添加到待办事项中：在开发待办事项中添加可持续性改进要求。
- 迭代和改进：使用迭代改进流程来识别、评估这些改进，确定其优先级，并进行测试和部署。
- 使用最简可行产品（MVP）进行测试：使用最简可行的代表性组件开发和测试潜在的改进，以降低测试的成本和环境影响。
- 简化流程：持续改进和简化您的开发流程。例如，使用持续集成和持续交付（CI/CD）管道测试和部署潜在的改进，自动完成软件交付过程，从而减少工作量和减少手动操作引起的错误。
- 培训和意识：为您的团队成员举办培训计划，教育他们了解可持续性以及他们的活动如何影响组织的可持续性目标。
- 评测和调整：持续评测改进的影响并根据需要作出调整。

资源

相关文档：

- AWS 助力可持续性解决方案

相关视频：

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future

- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure
- AWS re:Invent 2023 - What's new with AWS observability and operations

SUS06-BP03 让您的工作负载保持最新状态

让您的工作负载保持最新状态，采用高效功能、消除问题和提高工作负载的整体效率。

常见反模式：

- 认为当前架构是静态的，将来不会更新。
- 没有任何系统或定期安排来评估更新后的软件和软件包是否与工作负载兼容。

建立此最佳实践的好处：通过建立一个及时更新工作负载的流程，您能够采用新的特性和功能，解决问题，并提高工作负载效率。

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

最新的操作系统、运行时、中间件、库和应用程序可以提高工作负载效率，并简化更高效技术的采用。最新的软件可能还包括更准确地衡量工作负载对可持续性的影响的功能，因为供应商提供的功能是为了满足其自身的可持续性目标。定期更新，以便使用最新的功能和版本让您的工作负载保持最新。

实施步骤

- 定义流程：使用一个流程和计划来评估工作负载的新功能或实例。利用云中的敏捷性，快速测试新功能如何改善工作负载以：
  - 减小对可持续性的影响。
  - 提升性能效率。
  - 为计划改进消除障碍。
  - 提高衡量和管理可持续性影响的能力。
- 进行清点：清点工作负载软件和架构，并确定需要更新的组件。
  - 可以使用 AWS Systems Manager 清单从 Amazon EC2 实例中收集操作系统（OS）、应用程序和实例元数据，并快速了解哪些实例正在运行您的软件策略所需的软件和配置，以及哪些实例需要更新。

- 了解更新程序：了解如何更新工作负载的组件。

工作负载组件	如何更新
系统映像	使用 EC2 Image Builder 管理适用于 Linux 或 Windows 服务器映像的亚马逊系统映像（AMI）的更新。
容器映像	将 Amazon Elastic Container Registry（Amazon ECR）和现有管道结合使用来管理 Amazon Elastic Container Service（Amazon ECS）映像。
AWS Lambda	AWS Lambda 包含版本管理功能。

- 采用自动化：自动化更新可以减少部署新功能的工作量，并减少手动过程引起的错误。
  - 可以使用 CI/CD 自动更新 AMI、容器映像以及其他与云应用程序相关的构件。
  - 您可以使用 AWS Systems Manager 补丁管理器等工具来自动执行系统更新流程，并使用 AWS Systems Manager Maintenance Windows 来安排活动。

资源

相关文档：

- AWS 架构中心
- AWS 的新功能
- AWS 开发人员工具

相关视频：

- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance
- All Things Patch: AWS Systems Manager

SUS06-BP04 提高构建环境的利用率

提高资源利用率，以开发、测试和构建工作负载。

常见反模式：

- 手动预置或终止构建环境。
- 使构建环境保持独立于测试、构建或发布活动运行（例如，在开发团队成员的工作时间之外运行环境）。
- 为构建环境过度预置资源。

建立此最佳实践的好处：通过提高构建环境的利用率，您可以提高云工作负载的整体效率，同时将资源分配给构建者以进行高效的开发、测试和构建。

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

使用自动化和基础设施即代码功能，在需要时启动构建环境，并在不使用时将其关闭。一种常见模式是安排与开发团队成员的工作时间相吻合的可用时段。您的测试环境与生产配置非常相似。但是，寻找机会使用具有容量爆增的实例类型、Amazon EC2 竞价型实例、自动扩展数据库服务、容器和无服务器技术，以使开发和测试容量与使用容量保持一致。限制数据量，使之刚好满足测试要求。如果在测试中使用生产数据，请探索共享生产数据，而无需四处移动数据的可能性。

实施步骤

- 使用基础设施即代码：使用基础设施即代码来预置构建环境。
- 使用自动化：使用自动化功能来管理开发和测试环境的生命周期，并最大限度地提高构建资源的效率。
- 实现利用率最大化：使用策略来最大程度地利用开发和测试环境。
  - 使用最小可行代表性环境来开发和测试潜在的改进。
  - 如果可能，请使用无服务器技术。
  - 使用按需型实例来补充您的开发人员设备。
  - 使用具有容量爆增的实例类型、竞价型实例和其他技术，使构建容量与使用容量保持一致。
  - 采用原生云服务来实现安全的实例 Shell 访问，而不是部署堡垒主机群。
  - 根据构建作业自动扩展构建资源。

资源

相关文档：

- AWS Systems Manager Session Manager
- Amazon EC2 具爆发能力的实例
- 什么是 AWS CloudFormation ？
- 什么是 AWS CodeBuild ？
- AWS 实例调度器

相关视频：

- AWS re:Invent 2023 - Continuous integration and delivery for AWS

SUS06-BP05 使用托管式 Device Farm 进行测试

使用托管式设备场在一组具有代表性的硬件上高效地测试新功能。

常见反模式：

- 在各个物理设备上手动测试和部署应用程序。
- 未在真实的物理设备上使用应用测试服务进行测试以及与应用（例如，Android、iOS 和 Web 应用）互动。

建立此最佳实践的好处：使用托管式设备场测试支持云的应用程序有许多好处：

- 包括可在各种设备上测试应用程序的更高效功能。
- 无需使用内部基础设施进行测试。
- 提供多种设备类型（包括不太常用的较旧硬件），从而不需要进行不必要的设备升级。

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

使用托管式设备场有助于简化在一组有代表性的硬件上测试新功能的过程。托管式设备场提供多种设备类型，包括不太常用的较旧硬件，并避免不必要的设备升级对客户可持续性的影响。

实施步骤

- 定义测试要求：定义您的测试要求和计划（例如，测试类型、操作系统和测试时间表）。
  - 您可以使用 Amazon CloudWatch RUM 来收集和分析客户端数据，并制定您的测试计划。
- 选择托管式设备场：选择可以支持您的测试要求的托管式设备场。例如，可以使用 AWS Device Farm 来测试和了解您的更改对一组具有代表性的硬件的影响。
- 使用自动化：使用自动化和持续集成/持续部署（CI/CD）来安排和运行测试。
  - Integrating AWS Device Farm with your CI/CD pipeline to run cross-browser Selenium tests
  - 使用 AWS DevOps 和移动服务构建和测试 iOS 和 iPadOS 应用程序
- 审核和调整：持续审核测试结果，必要时进行改进。

资源

相关文档：

- AWS Device Farm device list
- 查看 CloudWatch RUM 控制面板

相关视频：

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm
- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM

相关示例：

- AWS Device Farm Sample App for Android
- AWS Device Farm Sample App for iOS
- Appium Web tests for AWS Device Farm

版权声明

AWS 术语表

有关最新的 AWS 术语，请参阅 AWS 词汇表 参考中的 AWS 词汇表。