AWS Well-Architected 框架

可持续性支柱

可持续性支柱: AWS Well-Architected 框架

Table of Contents

可持续性支柱 – AWS Well-Architected Framework

发布日期：2024 年 6 月 27 日（文档修订）

本白皮书重点介绍 Amazon Web Services（AWS）Well-Architected Framework 的可持续性支柱。该白皮书提供了设计原则、操作指导、最佳实践、潜在权衡和改进计划，可用于满足 AWS 工作负载的可持续性目标。

简介

AWS Well-Architected Framework 能够帮助您理解在 AWS 上构建工作负载时所做决策的利弊。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。该框架提供了一种方法，让您能够根据最佳实践持续衡量架构，从而确定需要改进的方面。拥有架构良好的工作负载可以极大地提高您支持业务成果的能力。

该框架基于六大支柱：

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

本文档重点介绍可持续性支柱，以及可持续性范围内的环境可持续性。本文档的目标读者是技术岗位的人员，例如首席技术官（CTO）、架构师、开发人员和运营团队成员。

阅读本文档后，您将了解可在设计注重可持续性的云架构时使用的 AWS 最新建议和策略。通过采用本白皮书中的实践，您可以构建能够最大限度地提高效率和减少浪费的架构。

云可持续性

可持续性原则能解决业务活动对环境、经济和社会带来的长期影响。联合国世界环境与发展委员会将可持续发展定义为“在不损害子孙后代满足其自身需求的能力的前提下，满足当前需求的发展”。您的企业或组织可能会对环境产生负面影响，例如直接或间接的碳排放、不可回收的废弃物以及对清洁水等共享资源的破坏。

在构建云工作负载时，可持续性实践是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。本文档侧重于环境影响，尤其是能源消耗和效率，因为这些是构架师在直接采取行动来减少资源使用时依据的重要杠杆。

在关注环境影响时，必须了解这些影响通常是如何计算的，以及对组织自身碳排放核算的后续影响。《温室气体核算体系》将碳排放划分为以下范围，也为 AWS 等云提供商在各范围内的相关排放示例：

- 范围 1：组织或其控制下的活动产生的所有直接排放。以数据中心备用发电机燃烧燃料为例。
- 范围 2：购买来为数据中心和其他设施供电的电力产生的间接排放。以商业发电产生的排放为例。
- 范围 3：组织活动产生的且来源无法控制的其余全部间接排放。以 AWS 与数据中心建设以及部署在数据中心的 IT 硬件的制造和运输相关的排放为例。

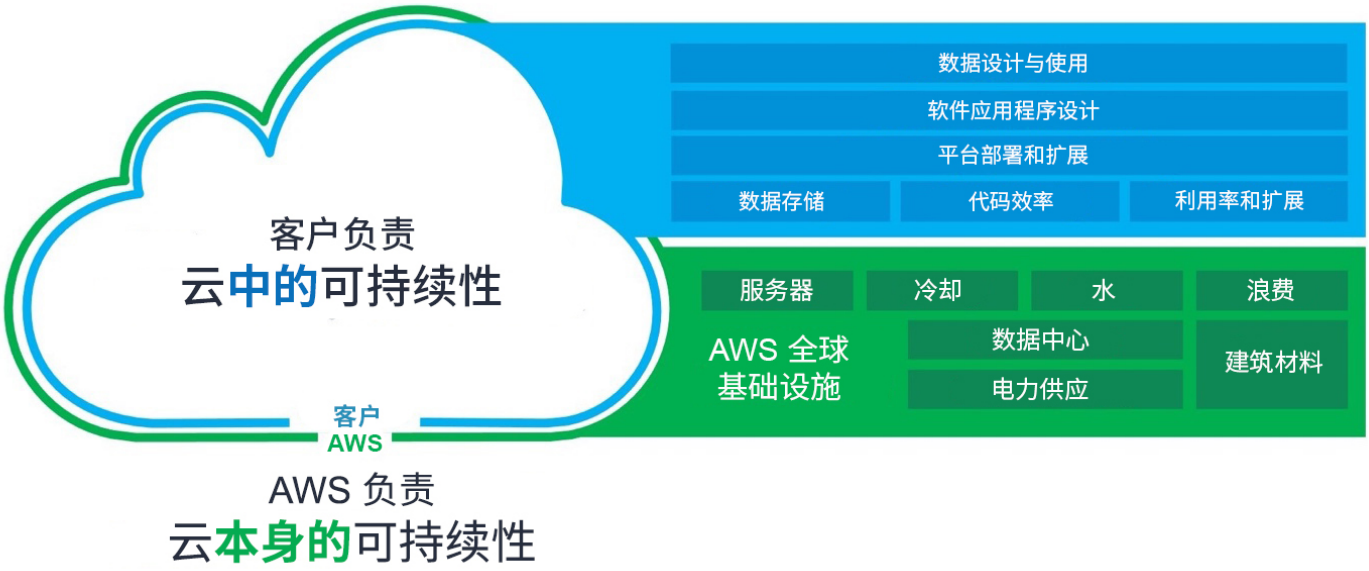
从 AWS 客户的角度来看，运行在 AWS 的工作负载产生的排放被视作间接排放，属于范围 3 排放。部署的每个工作负载产生的排放量只是先前每个范围 AWS 总排放量的一小部分。实际数量因工作负载而异，且受多个因素影响，包括使用的 AWS 服务、这些服务消耗的能源、为运行这些服务的 AWS 数据中心供电的电网的碳强度，以及 AWS 采购的可再生能源。

本文档首先介绍了环境可持续性的责任共担模式，然后给出了架构最佳实践，以便您可以通过减少工作负载在 AWS 数据中心运行所需的总资源来尽力降低工作负载的影响。

责任共担模式

环境可持续性是客户与 AWS 的共同责任。

- AWS 负责优化云的可持续性 – 提供高效共享的基础设施、水资源管理以及采购可再生能源。
- 客户负责确保云中的可持续性 – 优化工作负载和资源利用率，尽力减少为运行工作负载而要部署的总资源。



责任共担模式

云的可持续性

与传统的本地提供商相比，云提供商的碳足迹更低，能源效率更高，因为他们投资了高效的电力和冷却技术，运营着节能的服务器，也提高了服务器利用率。云工作负载利用网络、电源、冷却和物理设施等共享资源来减少影响。您可以在更高效的技术出现时将云工作负载迁移到这些技术上，并使用基于云的服务对工作负载进行转型，从而提高可持续性。

资源

- 迁移到 Amazon Web Services 带来的碳减排机会
- AWS 助力可持续性解决方案

云中的可持续性

云中的可持续性是一项持续的工作，侧重于通过从预置的资源中获得最大效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。

通过云实现可持续性

除了尽力降低已部署工作负载的影响外，还可以使用 AWS Cloud 来运行专为应对更广泛的可持续性挑战设计的工作负载。这些挑战示例包括减少碳排放、降低能耗、回收水或减少业务或组织其他领域的浪费。

通过云实现可持续性是指使用 AWS 技术来应对更广泛的可持续性挑战。例如，可以使用 Amazon Monitron 等机器学习服务来检测工业机械中的异常行为。这些检测数据可用于预防性维护，降低设备意外故障导致的环境事故风险，确保机器继续以最高效率运行。

云中的可持续性设计原则

在架构云工作负载时应用这些设计原则，可尽力提高可持续性，最大程度降低影响。

- 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，并为负责人提供投资于可持续性目标所需的资源。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，消除或尽可能减少空闲资源、处理和存储，从而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。减少或消除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。

改进流程

架构改进流程包括了解您拥有什么以及您可以采取哪些改进措施，选择改进目标，测试改进，采用成功的改进，量化您取得的成功并分享您学到的经验，以便可以在其他地方复制，然后重复该流程。

改进目标可以是：

- 消除浪费、利用率低以及闲置或未使用的资源
- 最大程度发挥所消耗资源的价值

> **Note**
>
> 使用预置的所有资源，以最少的资源完成同样的工作。

在优化的早期阶段，首先解决浪费或低利用率问题，然后转向适合特定工作负载的更有针对性的优化项。

监控资源消耗情况随时间的变化。确定累积变化导致资源消耗效率低下或显著增加的地方。确定要进行哪些改进来应对消耗变化情况，再按优先顺序实施改进。

以下步骤旨在构成一个迭代流程，可针对以可持续性为重点的云工作负载改进进行评估、优先顺序确定、测试和部署。

1. 确定改进目标：根据本文档中确定的可持续性最佳实践审核工作负载，确定需要改进的目标。
2. 评估具体改进：对潜在改进的具体变更、预计成本和业务风险进行评估。
3. 确定改进的优先顺序并制定计划：优先考虑能以最低的成本和风险带来最大改进的变更，并制定测试和实施计划。
4. 测试并验证改进：在测试环境中实施变更，验证变更的改进潜力。
5. 将变更部署到生产环境：在生产环境中实施变更。
6. 衡量结果并复制成功：寻找能在工作负载之间复制成功案例的机会，如遇不可接受的结果，则还原所做的变更。

示例方案

本文档会在后文引用以下示例场景，用以说明改进流程的每个步骤。

贵公司的工作负载在 Amazon EC2 实例上执行复杂的图像处理，将修改后的文件和原始文件存储起来供用户访问。处理活动会占用大量 CPU 资源，输出文件也非常大。

确定改进目标

了解有助于实现可持续性目标的最佳实践。您可以在本文档的后文找到这些最佳实践的详细描述和改进建议。

审核工作负载及其所用资源。确定大型部署和常用资源这样的热点。评估这些热点，从中寻找机会来提高资源的有效利用率，从而减少实现业务成果所需的总资源。

根据最佳实践审核工作负载，确定需要改进的候选项。

将此步骤应用于 示例方案，即可将以下最佳实践确定为可能的改进目标：

- 使用最少的硬件来满足需求
- 使用最能支持数据访问和存储模式的技术

资源

- 优化您的 AWS 基础设施以实现可持续性，第 I 部分：计算
- 优化您的 AWS 基础设施以实现可持续性，第 II 部分：存储
- 优化您的 AWS 基础设施以实现可持续性，第 III 部分：联网

评估具体改进

了解工作负载为完成工作单元而预置的资源。评估潜在改进，并预估这些改进的潜在影响、实施成本以及相关风险。

要衡量一段时间内的改进，首先要了解已在 AWS 中预置的资源以及这些资源的消耗情况。

从全面概述您的 AWS 使用情况开始，再使用 AWS 成本和使用情况报告来帮助确定热点。若采用了 Amazon Athena，此 AWS 示例代码有助于您查看和分析自己的报告。

代理指标

在评估具体变更时，还必须评估哪些指标最能量化该变更对关联资源的影响。这些指标即被称为代理指标。选择最能反映您正在评估的改进类型和改进所针对的资源的代理指标。这些指标可能会随时间发生变化。

为支持工作负载而预置的资源包括计算、存储及网络资源。使用代理指标评估预置资源，了解这些资源的消耗情况。

使用代理指标来衡量为实现业务成果而预置的资源。

资源	代理指标示例	改进目标
计算	vCPU 分钟数	最大程度提高预置资源的利用率
存储	预置的 GB 数	减少预置的总量
网络	传输的 GB 数或传输的数据包数	缩短传输总距离和传输距离

业务指标

选择业务指标，量化业务成果的实现情况。业务指标应反映工作负载提供的价值，例如：并发活跃用户数、已处理的 API 调用数或已完成的事务数。这些指标可能会随时间发生变化。在评估基于财务的业务指标时要谨慎，因为事务值不一致会导致比较无效。

关键绩效指标

使用以下公式，将预置的资源除以实现的业务成果，由此确定每个工作单元的预置资源。

$$按工作单元预置的资源 = \frac{预置资源的代理指标}{成果的业务指标}$$

KPI 公式

将每个工作单元的资源用作 KPI。根据预置资源建立基准，作为比较的基础。

资源	KPI 示例	改进目标
计算	每个事务的 vCPU 分钟数	最大程度提高预置资源的利用率

资源	KPI 示例	改进目标
存储	每个事务的 GB 数	减少预置的总量
网络	每个事务传输的 GB 数或每个事务传输的数据包数	缩短传输总距离和传输距离

预估改进

根据预置资源的减少数量（如代理指标所示），以及每个工作单元的预置基准资源的百分比变化情况，预估改进的情况。

资源	KPI 示例	改进目标
计算	每个事务 vCPU 分钟数降幅（%）	最大程度提高利用率
存储	每个事务 GB 数降幅（%）	减少预置的总量
网络	每个事务传输的 GB 数或每个事务传输的数据包数降幅（%）	缩短传输总距离和传输距离

评估改进

根据预期的净效益评估潜在改进。评估实施与维护的时间、成本和工作量以及业务风险，例如意外影响。

有针对性的改进通常表示所消耗资源类型之间的权衡。例如，为了减少计算消耗量，可以将结果存储起来；为了限制数据传输量，可以在将结果发送给客户端之前处理数据。稍后将进一步讨论这些权衡。

在评估工作负载风险时，还应考虑非功能性要求，包括安全性、可靠性、性能效率、成本优化以及改进对运营工作负载的能力的影响。

将此步骤应用于 示例方案，即可评估目标改进，结果如下：

最佳实践	有针对性的改进	潜力	成本	风险
使用最少的硬件来满足需求	实施预测性扩展来缩短利用率低的期限	中	低	低
使用最能支持数据访问和存储模式的技术	实施更有效的压缩机制来减少总存储空间和实现该目标的时间	高	低	低

实施预测性扩展可以减少未充分利用或未使用的实例消耗的 vCPU 小时数。与现有扩展机制相比，预测性扩展机制确有一定优势，消耗的资源量预计降低了 11%。所涉及的成本很低，涵盖了云资源的配置和 Amazon EC2 Auto Scaling 的预测性扩展操作。当需求超出预测时，被动地执行横向扩展会导致性能受限，这便是风险所在。

实施更有效的压缩会产生显著影响，大幅减少所有原始图像和处理后图像的文件大小，生产中的存储需求预计会降低 25%。实施新算法是一种工作量极小的替代方案，涉及的风险很低。

确定改进的优先顺序并制定计划

根据最大的预期影响、最低的成本和可接受的风险，确定已确定改进的优先顺序。

决定最初要重点关注的改进，并将这些改进纳入资源规划和开发路线图中。

将此步骤应用于 示例方案，即可按以下方式确定目标改进的优先顺序：

优先级	改进	潜力	成本	风险
1	实施更有效的压缩机制	高	低	低
2	实施预测性扩展	中	低	低

更新文件压缩机制的高潜力、低成本和风险使其成为贵公司的高价值目标，优先顺序高于实施预测性扩展。在文件压缩完成后，您确定应优先实施具有中等潜在影响、低成本和低风险的预测性扩展。

您可以指派一名团队成员来实施改进后的文件压缩，为积压工作添加预测性扩展功能。

测试并验证改进

以最少的投资进行小规模测试，借此降低大规模工作伴随的风险。

在测试环境中实施工作负载的代表性副本，可限制执行测试和验证工作的成本和风险。执行一组预定义的测试事务、测量预置资源并确定每个工作单元的所用资源，由此建立测试基准。

在测试环境中实现目标改进，再在相同条件下使用相同方法重复执行测试。然后，根据改进情况，衡量预置资源和每个工作单元的所用资源。

计算每个工作单元的预置资源相对于基准资源的百分比变化，确定生产环境中预置资源的预期减少数量。将这些值与预期值进行对比。判定结果是否达到可接受的改进水平。评估所耗额外资源的权衡是否会令改进带来的净效益变得不可接受。

判定改进是否成功，以及是否应投入资源来实施生产变更。如果此时变更被评估为不成功，请重定向资源来测试并验证下一个目标，然后继续推行改进周期。

每个工作单元的预置资源降幅（%）	预置资源减少数量	操作
达到期望	达到期望	继续进行改进
未达到预期	达到期望	继续进行改进
达到期望	未达到预期	寻求替代改进
未达到预期	未达到预期	寻求替代改进

将此步骤应用于 示例方案，即可执行测试来验证是否成功。

在对改进后的压缩算法进行测试后，每个工作单元的预置资源（原始图像和修改后的图像所需的存储空间）的百分比降幅达到了预期，预置存储空间的平均降幅达到 30%，计算负载的增幅可以忽略不计。

与存储空间实现的降幅相比，您确定将改进后的压缩算法应用于生产环境中的现有文件所需的额外计算资源微不足道。您确认所需资源（存储空间 TB 数）在减少数量方面取得了成功，并且改进也获准用于生产部署。

将变更部署到生产

实施经过测试、验证和批准的生产改进。使用有限部署进行实施，确认工作负载的功能，测试在有限部署中预置资源和每个工作单元消耗资源的实际减少情况，并检查变更是否会产生意外后果。测试成功后，继续全面部署。

如果测试失败或者变更造成不可接受的后果，则还原所做的变更。

将此步骤应用于 示例方案，即可执行以下操作。

通过蓝绿部署方法，您可以使用有限部署在生产环境中实施变更。针对新部署实例执行的功能测试已成功。原始图像文件和处理后的图像文件的预置存储空间的平均降幅达到 26%。并无任何证据表明压缩新文件会增加计算负载。

压缩图像文件所花费的时间大幅缩短，这是因为新压缩算法采用了高度优化的代码。

继续全面部署新版本。

衡量结果并复制成功

按以下方式衡量结果并复制成功：

- 衡量每个工作单元的预置资源的初步改进情况和预置资源数量减少情况。
- 将初步估计值和测试结果与生产测量值进行比较。确定可能造成差异的因素，并视情况更新估算和测试方法。
- 确定成功和成功度，并与利益相关方分享结果。
- 如因测试失败或变更造成意料外的负面后果而必须还原所做的变更，请确定成因。在可行之时进行迭代，或者评估可实现变更目标的新方法。
- 利用汲取经验，建立标准，并将成功的改进应用于其他同样可以受益的系统。跨团队和组织收集并共享方法、相关构件以及净效益，以便其他人员可以采用您的标准并复制您的成功。
- 监控每个工作单元的预置资源，跟踪一段时间内的变化情况和总体影响。工作负载的变化或客户使用工作负载的方式都可能会对改进的效果产生影响。如果改进的效果在短期内显著降低，或者随着时间的推移效果累积降低，则重新评估改进机会。
- 量化一段时间内从您的改进中获得的净效益，包括应用您改进的其他团队获得的效益（若有），以此显示您的改进活动带来的投资回报。

将此步骤应用于 示例方案，即可测量以下结果。

在将新的压缩算法部署并应用于现有图像文件后，工作负载显示存储需求的降幅达到了 23%。

测量值与初步估计值（25%）基本一致，与测试值（30%）之间的明显差异则是因为测试中使用的图像文件不能代表生产中存在的图像文件。您可以修改测试图像集，更准确地反映生产中的图像。

该改进被视作完全成功的改进。预置存储空间的总降幅比估计的 25% 少了 2%，但 23% 仍然是可持续性影响方面的巨大改进，同时还实现了同等的成本节省效果。

变更带来的唯一的意外结果是缩短了执行压缩所需的时间，并相应减少了 vCPU 消耗量。之所以会有这些改进，是因为采用了高度优化的代码。

您可以建立一个内部开源项目，在其中共享自己的代码、关联的构件、实施变更的指导以及实施结果。内部开源项目让团队可以轻松地将代码用于其所有持久性文件存储应用场景。您的团队可将改进作为标准。内部开源项目的另一好处是，采用该解决方案的每个人都能从解决方案的改进中受益，并且任何人都可以为项目做出改进。

您可以发布成功案例，在整个组织中共享开源项目。采用该解决方案的每个团队都能以最少的投资复制效益，增加从您的投资中获得的净效益。您将此类资料作为持续的成功案例发布。

您可以继续监控改进随时间推移产生的影响，也可以根据需要对内部开源项目进行更改。

可持续性作为非功能性要求

在业务要求列表中添加可持续性，可以带来更经济高效的成果。专注于从所使用的资源中获得更多价值，同时减少资源使用量，可以直接节省在 AWS 上花费的成本，因为只需为使用的资源付费。

实现可持续性目标可能不需要对一个或多个其他传统指标（例如正常运行时间、可用性或响应时间）给予同等权衡。可持续性方面可以取得重大进展，服务水平却不会受到明显影响。如遇需要给予细微权衡的情况，通过这些权衡获得的可持续性改善可能会超过服务质量的变化。

鼓励团队成员在制定功能要求时不断尝试可持续性改进。团队在设定目标时还应嵌入代理指标，确保他们在开发工作负载时评估资源耗用强度。

以下示例权衡可用于减少消耗的云资源：

调整结果质量：可以通过近似计算以结果质量（QoR）换得工作负载强度的降低。近似计算的做法可以寻找机会，利用客户需要和实际产出之间的差异。例如，若将数据放在 set 数据结构中，则可删除 SQL 中的 ORDER BY 运算符来免除不必要的处理，在节省资源的同时仍能提供可接受的答案。

调整响应时间：通过在最大程度上降低共同开销，响应时间较长的应答可以减少碳排放。处理临时的、短暂的任务可能产生启动开销。对任务进行分组与批处理，而不是每次在任务出现时支付这些开销。批处理以延长响应时间换得启动实例、下载源代码和运行该流程所需的共同开销的减少。

调整可用性：在 AWS 的助力下，执行几次点击操作即可增加冗余并实现高可用性目标。预置总是会导致利用率降低的闲置资源，便可通过静态稳定性之类的技术来增加冗余。在设定目标时对业务需求进行评估。可用性方面相对细微的权衡可能会大幅提高利用率。例如，静态稳定性架构模式涉及到对闲置的失效转移能力进行预置，便于在组件发生故障以后立即进行加载。放宽可用性要求可以为自动化部署替换资源留出时间，从而消除对闲置在线容量的需求。按需增加失效转移能力可以提高整体利用率，不但不会在正常运营期间对业务产生影响，还能带来降低成本的另一好处。

云中的可持续性最佳实践

优化工作负载放置，并针对需求、软件、数据、硬件和流程优化架构，从而提高能源效率。每一个领域都代表着采用最佳实践来降低云工作负载对可持续性影响的机会，即最大程度提高利用率、尽量减少浪费以及尽力削减为支持工作负载而部署和使用的总资源数。

主题

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

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

- 按照以下步骤进行操作，根据您的业务需求（包括法规遵从性、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，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 Pricing Calculator 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。
- 选择亚马逊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他位置（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

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。
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

- 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：

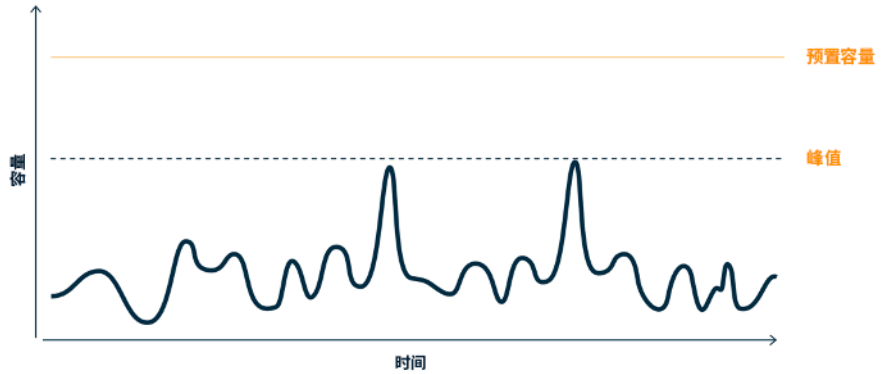
- 在不需要的时候立即处理客户端请求。
- 没有分析客户端请求的要求。

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

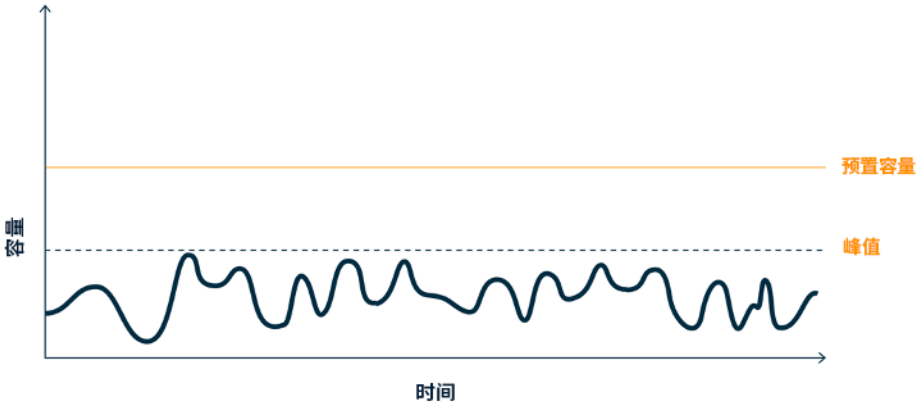
实施指导

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

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。注意客户用来访问服务的设备，据此实施相应的模式，尽量避免升级设备。

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

检查您的工作负载以识别空闲或未使用的组件。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降低内存大小。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

- 清点您的 AWS 资源。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。
- 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或将其与其他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，而不必在未充分利用的单个实例上运行数据库。
- 了解工作负载为完成工作单元而预置的资源。

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

相关示例：

- 优化硬件模式并遵守可持续性 KPI

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 CodeWhisperer 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
- AWS re:Invent 2023 - Best practices for Amazon CodeWhisperer
- 使用 Amazon CodeGuru 自动提供代码审查和应用程序性能建议

相关示例：

- 使用 Amazon CodeGuru 优化代码

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-BP04 使用托管式设备场进行测试。

- 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

使用最符合您的数据特性和访问模式的软件和架构模式。例如，使用AWS 上的现代数据架构，该架构允许您使用针对您的独特分析用例进行优化的专用服务。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

- 分析您的数据特性和访问模式，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
- 使用最能支持数据访问和存储模式的架构模式。
  - 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，确保高效地执行查询。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。
- 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

数据管理

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，并使用最能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。

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

类型	Technology	主要特性
		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。
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。

要在数据集的整个生命周期内高效管理数据集，请配置生命周期策略，这些策略是定义如何处理数据集的规则。

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、将其存档或删除。

实施步骤

- 对工作负载中的数据集进行分类。
- 定义每个数据类的处理过程。
- 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。如果访问模式未知、变化或不可预测，则可以使用 Amazon S3 Intelligent-Tiering，因为此功能可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。

存储服务	如何设置自动化生命周期策略
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。

- 删除未使用的卷、快照和超出保留期的数据。利用本机服务功能（如 Amazon DynamoDB 生存时间或 Amazon CloudWatch 日志保留）进行删除。
- 在适当情况下根据生命周期规则汇总和压缩数据。

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

请勿存储不需要的数据。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。利用服务的本机数据复制和冗余功能。

实施步骤

- 评估是否可以通过使用 AWS Data Exchange 和 Open Data on AWS 中的现有公开数据集来避免存储数据。
- 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。

- 分析数据访问以识别不需要的数据。自动执行生命周期策略。利用本机服务功能（如 Amazon DynamoDB 生存时间或 Amazon S3 生命周期或 Amazon CloudWatch 日志保留）进行删除。
- 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
- 使用可进行增量备份的备份技术。
- 利用 Amazon S3 的持久性和 Amazon EBS 的复制来实现您的持久性目标，而不是自管式技术（例如独立磁盘的冗余阵列（RAID））。
- 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。
- 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。
- 推送新版本的工作负载时，从对象存储和边缘缓存中删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

- 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。

- 仅在需要时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。
- 根据您的使用模式适当删除数据，如 SUS04-BP03 使用策略管理数据集的生命周期 中所述。
- 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

- 在为工作负载选择区域时，请考虑将与数据或用户的距离作为决定因素。
- 按区域对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（如 Parquet 或 ORC），并在通过网络移动数据之前先对其进行压缩。
- 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。

服务	何时使用
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

相关示例：

- 针对可持续性设计 – 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

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

- 实施 SUS04-BP01 实施数据分类策略 中概述的数据分类策略。
- 利用数据分类的重要程度，并根据恢复时间目标（RTO）和恢复点目标（RPO）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）。
- 使用自动化解决方案或托管服务来备份关键业务数据。
  - 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

相关示例：

- Well-Architected Lab – 备份数据

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。

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 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
- 将 Amazon Relational Database Service Databases 迁移到 Graviton

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

- 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%。
- 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

- 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：

- 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

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

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 采用可以快速引入可持续性改进的方法
- SUS06-BP02 让您的工作负载保持最新状态
- SUS06-BP03 提高构建环境的利用率
- SUS06-BP04 使用托管式设备场进行测试

SUS06-BP01 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
- 基于 AWS CodeCommit 的可扩展敏捷开发实践

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

相关示例：

- Well-Architected Lab – 将成本和使用情况报告转化为效率报告

SUS06-BP02 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化系统更新流程，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 补丁：AWS Systems Manager

相关示例：

- Well-Architected Lab – 清单和补丁管理
- 实验室：AWS Systems Manager

SUS06-BP03 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-BP04 使用托管式设备场进行测试

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，您可以使用 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。
  - 将 AWS Device Farm 与您的 CI/CD 管道集成，运行跨浏览器的 Selenium 测试
  - 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 设备列表
- 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：

- 针对 Android 的 AWS Device Farm 示例应用程序
- 针对 iOS 的 AWS Device Farm 示例应用程序
- 适用于 AWS Device Farm 的 Appium Web 测试

结论

为了应对政府监管、竞争优势以及客户、员工和投资者需求的变化，越来越多的组织正在设定可持续性目标。首席技术官、架构师、开发人员和运营团队成员都在寻找能够直接帮助实现组织可持续性目标的方法。利用由 AWS 服务提供支持的这些设计原则和最佳实践，您可以做出明智的决策，在安全性、成本、性能、可靠性和卓越运营与 AWS Cloud 工作负载的可持续性成果之间取得平衡。您为减少资源使用量和提高工作负载效率所采取的每一项措施，都有助于降低对环境的影响，更有助于实现组织更广泛的可持续性目标。

贡献者

本文档的贡献者包括：

- Sam Mokhtari，亚马逊云科技高级效率首席解决方案架构师
- Brendan Sisson，Amazon Web Services 首席解决方案架构师
- Margaret O'Toole，Amazon Web Services 可持续性技术负责人
- Steffen Grunwald，Amazon Web Services 首席可持续性解决方案架构师
- Ryan Eccles，Amazon 可持续性业务首席工程师
- Rodney Lester，Amazon Web Services 首席架构师
- Adrian Cockcroft，Amazon Web Services 可持续性架构业务副总裁
- Ian Meyers，Amazon Web Services 解决方案架构业务技术总监

延伸阅读

如需了解其他信息，请参阅：

- AWS Well-Architected
- AWS 架构中心
- 云中的可持续性
- AWS 助力可持续性解决方案
- The Climate Pledge
- 联合国可持续发展目标
- 温室气体核算体系

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
更新了最佳实践指南	对整个支柱做了细微改动。	2024 年 6 月 27 日
更新了风险等级	对最佳实践风险等级进行了细微更新。	2023 年 10 月 3 日
更新了最佳实践指南	根据以下领域的新指导更新了最佳实践：符合需求、软件和架构、数据以及硬件和服务。	2023 年 7 月 13 日
针对新框架进行了更新	为最佳实践更新了规范性指南并增加了新的最佳实践。	2023 年 4 月 10 日
已更新白皮书	为最佳实践更新了新的实施指导。	2022 年 12 月 15 日
已更新白皮书	扩展了最佳实践并增加了改进计划。	2022 年 10 月 20 日
初次发布	发布了可持续性支柱 – AWS Well-Architected Framework。	2021 年 12 月 2 日

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。