使用 Amazon DynamoDB 全局表

# AWS 规范性指导

# AWS 规范性指导: 使用 Amazon DynamoDB 全局表

# Table of Contents

# 使用 Amazon DynamoDB 全局表

Jason Hunter，亚马逊云科技（AWS）

2024 年 3 月（文档历史记录）

全局表基于遍布全球的 Amazon DynamoDB 而构建，可提供完全托管式、多区域和多活动数据库，为大规模扩展的全局应用程序提供快速本地读写性能。全局表会根据您的选择自动复制您的 DynamoDB 表。 AWS 区域全局表使用现有的 DynamoDB API，因此无需更改应用程序。使用全局表无需支付任何预付费用，也没有任何承诺，您只需为实际使用的资源付费。

本指南将介绍如何有效使用 DynamoDB 全局表。它提供了有关全局表的关键事实，解释了该特征的主要用例，介绍了您应考虑的三种不同写入模型的分类法，介绍了您可以实现的四种主要请求路由选择，讨论了撤出处于活动状态的区域或离线区域的方法，解释了如何考虑吞吐能力规划，并提供了部署全局表时需要考虑的事项清单。

本指南适用于更广泛的 AWS 多区域部署背景，如AWS 多区域基础知识白皮书和带视频的数据弹性设计模式所述。 AWS

内容

# 全局表概览

## 关键事实

- 全局表有两个版本：版本 2017.11.29（旧版）（有时称为 v1）和版本 2019.11.21（当前）（有时称为 v2）。本指南仅关注当前版本。
- DynamoDB（不带全局表）是一项区域服务，这意味着它具有高可用性，可以抵御基础设施故障，包括整个可用区的故障。单区域 DynamoDB 表专为实现 99.99% 的可用性而设计。有关更多信息，请参阅 DynamoDB 服务等级协议 (SLA)。
- DynamoDB 全局表在两个或多个区域之间复制其数据。多区域 DynamoDB 表专为 99.999% 的可用性而设计。通过适当的规划，全球表格可以帮助创建能够抵御区域故障的架构。
- 全局表采用主动-主动复制模型。从 DynamoDB 的角度来看，每个区域中的表在接受读取和写入请求方面具有同等地位。收到写入请求后，本地副本表将在后台将写入操作复制到其他参与的远程区域。
- 项目是单独复制的。在单个事务中更新的项目可能无法一起复制。
- 源区域中的每个表分区与其他分区parallel 复制其写入操作。远程区域内的写入操作顺序可能与源区域内发生的写入操作顺序不匹配。有关表分区的更多信息，请参阅博客文章扩缩 DynamoDB：分区、热键和热拆分如何影响性能。
- 新写入的项目通常会在一秒内传播到所有副本表。附近区域的传播速度往往更快。
- 亚马逊CloudWatch为每个区域对提供一个`ReplicationLatency`指标。它的计算方法是查看到达的项目，将它们的到达时间与初始写入时间进行比较，然后计算平均值。时间存储CloudWatch在源区域内。查看平均和最大时间对于确定平均和最坏情况的复制延迟很有用。对于这种延迟，没有 SLA。
- 如果在两个不同的区域中大约同时更新单个项目（在此`ReplicationLatency`窗口中），并且第二次写入操作发生在复制第一个写入操作之前，则可能会出现写入冲突。全局表根据写入操作的时间戳，使用最后写入者获胜机制来解决此类冲突。第一个操作 “输给” 第二个操作。这些冲突未记录在CloudWatch或中AWS CloudTrail。
- 每个项目都有最后一次写入时间戳，保留为一个私有系统属性。l ast writer wins 方法是通过使用条件写入操作来实现的，该操作要求传入项目的时间戳大于现有项目的时间戳。
- 全局表将所有项目复制到所有参与区域。如果您想拥有不同的复制范围，可以创建多个全局表，并为每个表分配不同的参与区域。
- 即使副本区域处于离线状态或`ReplicationLatency`增长，本地区域也接受写入操作。本地表继续尝试将项目复制到远程表，直到每个项目成功为止。

- 万一某个区域完全脱机，稍后恢复联机时，将重试所有待处理的出站和入站复制。无需特殊操作即可使表恢复同步。最后一个作者获胜机制可确保数据最终变得一致。
- 您可以随时向 DynamoDB 表中添加新区域。DynamoDB 处理初始同步和持续复制。您也可以删除区域（甚至是原始区域），这将删除该区域中的本地表。
- DynamoDB 没有全局端点。所有请求都是向区域终端节点发出的，该终端节点访问该区域的本地全局表实例。
- 对 DynamoDB 的调用不应跨区域。最佳做法是让位于一个区域的应用程序仅直接访问其区域的本地 DynamoDB 终端节点。如果在某个区域（在 DynamoDB 层或周围的堆栈中）检测到问题，则应将最终用户流量路由到托管在不同区域的其他应用程序终端节点。全局表确保位于每个区域的应用程序都可以访问相同的数据。

# 使用案例

全局表提供以下常见好处：

- 低延迟的读取操作。您可以将数据副本放在离最终用户更近的地方，以减少读取操作期间的网络延迟。数据与`ReplicationLatency`值一样保持最新状态。
- 低延迟写入操作。最终用户可以写入附近的区域，以减少网络延迟和完成写入操作的时间。必须谨慎路由写入流量，以确保没有冲突。路由技术将在后面的章节中讨论。
- 改善了弹性和灾难恢复。如果某个区域的性能下降或出现全面中断，您可以将其撤离（移除发送到该区域的部分或全部请求），以达到以秒为单位的恢复点目标 (RPO) 和恢复时间目标 (RTO)。使用全局表还可以将 DynamoDB SLA 的每月正常运行时间百分比从 99.99% 提高到 99.999%。
- 无缝的区域迁移。您可以添加一个新区域，然后删除旧区域，以便将部署从一个区域迁移到另一个区域，而无需在数据层停机。

例如，富达投资在 re: Invent 2022 上介绍了他们如何在订单管理系统中使用 DynamoDB 全球表。他们的目标是在本地处理无法达到的规模上实现可靠的低延迟处理，同时保持对可用区和区域故障的适应能力。

# 全局表的写入模式

全局表在表级别始终处于主动-主动状态。但是，您可能希望通过控制您路由写入请求的方式来将它们视为主动-被动。例如，您可能决定将写入请求路由到单个区域，以避免潜在的写入冲突。

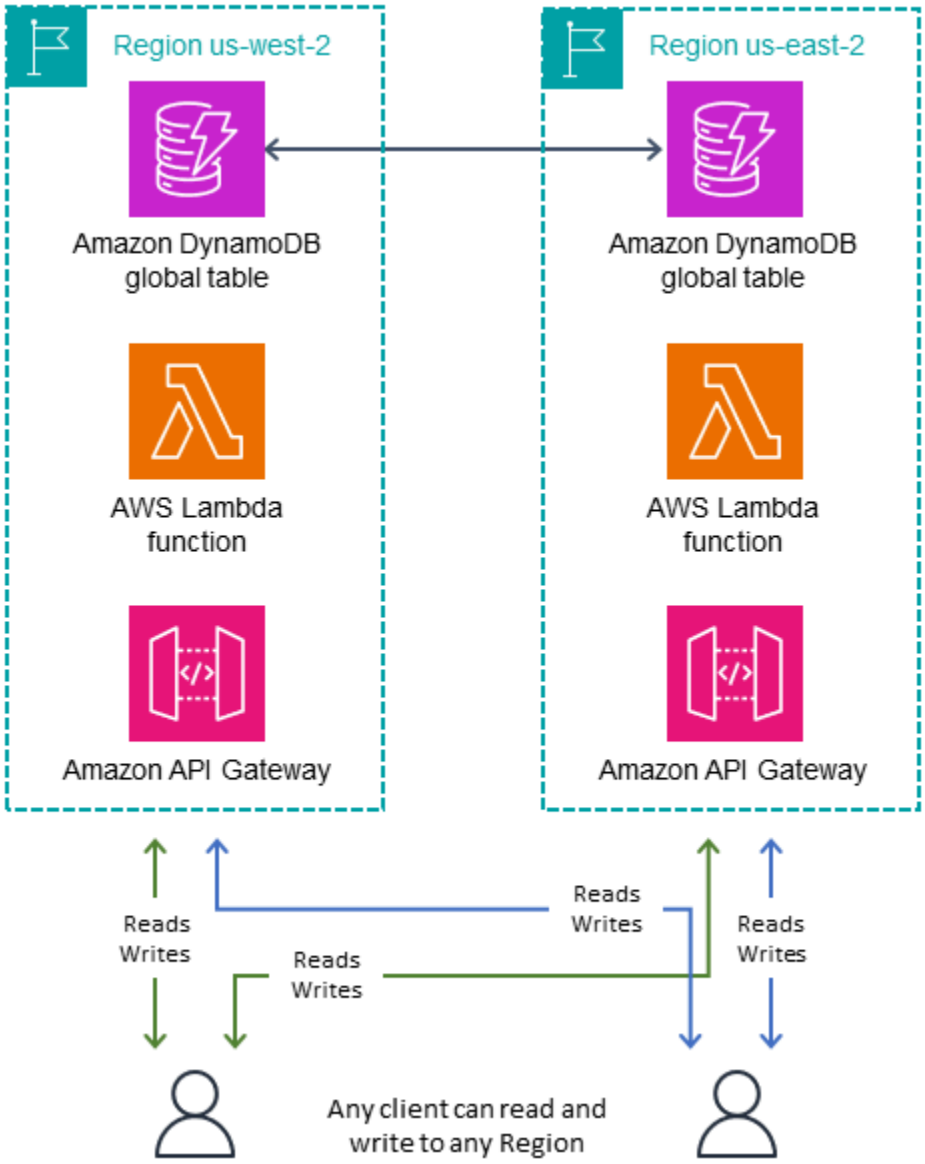
有三种主要的托管写入模式，如接下来的三节所述。您应该考虑哪种写入模式适合您的使用案例。此选择会影响您路由请求、撤离区域和处理灾难恢复的方式。后面部分中的指南取决于应用程序的写入模式。

主题



## 写入任何区域模式（非主模式）

对任何 Regi on 写入模式的写入都完全处于活动状态，不会对写入操作的发生位置施加限制。任何地区都可以随时接受写入请求。这是最简单的模式；但是，它只能用于某些类型的应用程序。当所有写入操作都是等等时，它很合适。Idemp otent 意味着它们是安全可重复的，因此跨区域的并发或重复写入操作不会发生冲突，例如，当用户更新其联系人数据时。它也适用于仅限追加的数据集，在该数据集中，所有写入操作都是确定性主键下的唯一插入，这是一种特殊情况，是等同性的。最后，此模式适用于写入操作冲突风险可以接受的情况。




写入任何区域模式是实现起来最简单的架构。路由更容易，因为任何区域都可以随时成为写入目标。故障转移更容易，因为任何最近的写入操作都可以多次重播到任何辅助区域。在可能的情况下，您应该针对这种写入模式进行设计。

例如，一些视频流媒体服务使用全局表格来跟踪书签、评论、观看状态标志等。这些部署可以使用写入到任何区域模式，只要它们确保每个写入操作都是等等的。如果每次更新（例如，设置新的最新时间码、分配新的评论或设置新的监视状态）都直接分配用户的新状态，而项目的下一个正确值不取决于其当前值，则会出现这种情况。如果偶然将用户的写入请求路由到不同的区域，则最后一次写入操作将持续存在，全局状态将根据最后一次分配确定。此模式下的读取操作最终将变得一致，并因最新`ReplicationLatency`值而延迟。

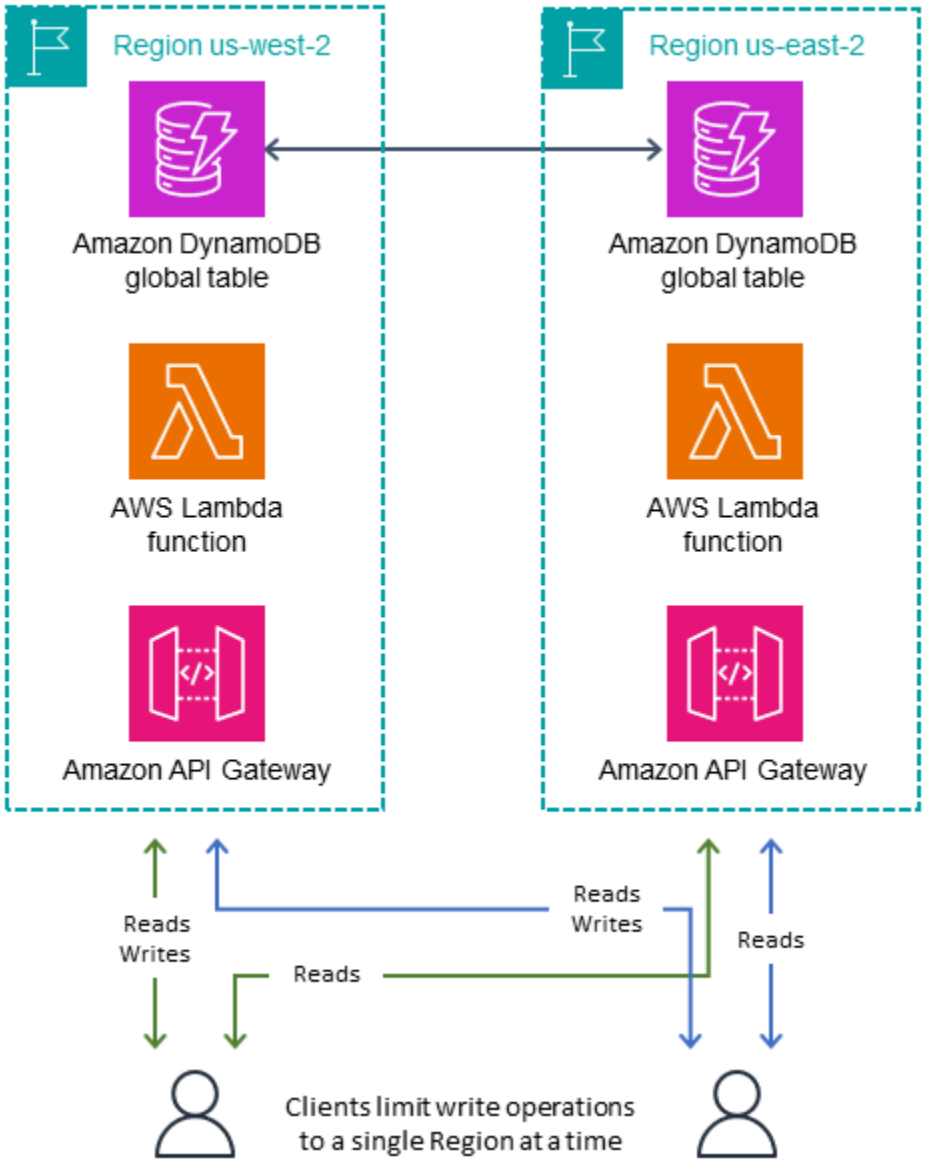
在另一个例子中，一家金融服务公司使用全局表作为系统的一部分，以持续统计每位客户的借记卡购买情况，从而计算该客户的现金返还奖励。新的事务从世界各地流入并转向多个区域。这家公司经过仔细的重新设计，能够使用写入到任何区域模式。最初的设计草图仅为每位客户保留了一RunningBalance件物品。客户操作使用ADD表达式更新了余额，该表达式不是等能的（因为新的正确值取决于当前值），如果在不同区域大约同时对同一个余额进行两次写入操作，则余额将不同步。重新设计使用事件流，其工作原理类似于带有仅限追加工作流程的账本。每个客户操作都会在为该客户维护的项目集合中追加一个新项目。（项目集合是一组共享主键但具有不同排序键的项目。）每个写入操作都是等能插入，使用客户 ID 作为分区键，事务 ID 作为排序键。这种设计使余额的计算变得更加困难，因为它需要先提取项目，然后再Query进行一些客户端数学运算，但是它使所有写入操作都是等的，并显著简化了路由和故障转移。本指南下文中的。

第三个例子涉及一家提供在线广告投放服务的公司。该公司认为，为了简化写入任何区域模式的设计，降低数据丢失风险是可以接受的。当他们投放广告时，他们只有几毫秒的时间来检索足够的元数据来确定要展示的广告，然后记录广告曝光量，这样他们就不会很快重复同样的广告。他们使用全局表为全球最终用户提供低延迟读取操作和低延迟写入操作。它们在单个项目中记录了用户的所有广告曝光量，该列表表示为一个不断增长的列表。他们使用一件商品而不是追加到商品集合中，因此他们可以在每次写入操作中删除较早的广告曝光量，而无需为删除操作付费。这种写入操作不是等同的；如果同一个最终用户几乎同时看到来自多个区域的广告，则针对广告曝光量的一项写入操作可能会覆盖另一项写入操作。风险在于用户可能会偶尔看到重复的广告。他们认为这是可以接受的。

# 写入一个区域模式（单主模式）

向一个区域写入模式的写入是主动-被动模式，并将所有表写入操作路由到单个主动区域。（DynamoDB 没有单一活动区域的概念；DynamoDB 之外的层负责管理这个区域。）写入到一个区域模式通过确保写入操作一次仅流向一个区域来避免写入冲突。当你想使用条件表达式或事务时，这种写入模式会有所帮助。除非您知道自己针对的是最新数据，否则这些表达式是不可能的，因此它们需要将所有写入请求发送到拥有最新数据的单个区域。

最终，一致性读取操作可以进入任何副本区域以实现更低的延迟。强一致性读取操作必须进入单个主区域。




如后文所述，有时需要更改活动区域以应对区域故障。一些用户会定期更改当前活跃的区域，例如实施follow-the-sun部署。这使活动区域靠近活动最多的地理位置（通常是白天，因此得名），从而实现最低的读取和写入操作延迟。它还有一个附带好处，那就是每天调用更改区域的代码，并确保在进行任何灾难恢复之前对其进行良好的测试。

被动区域可能会在 DynamoDB 周围保留缩小规模的基础设施，只有当它成为主动区域时，该基础设施才会建成。本指南不涵盖飞行灯和热待机设计。如需更多信息，您可以阅读博客文章灾难恢复 (DR) 架构AWS，第 III 部分：先导灯和热备用。

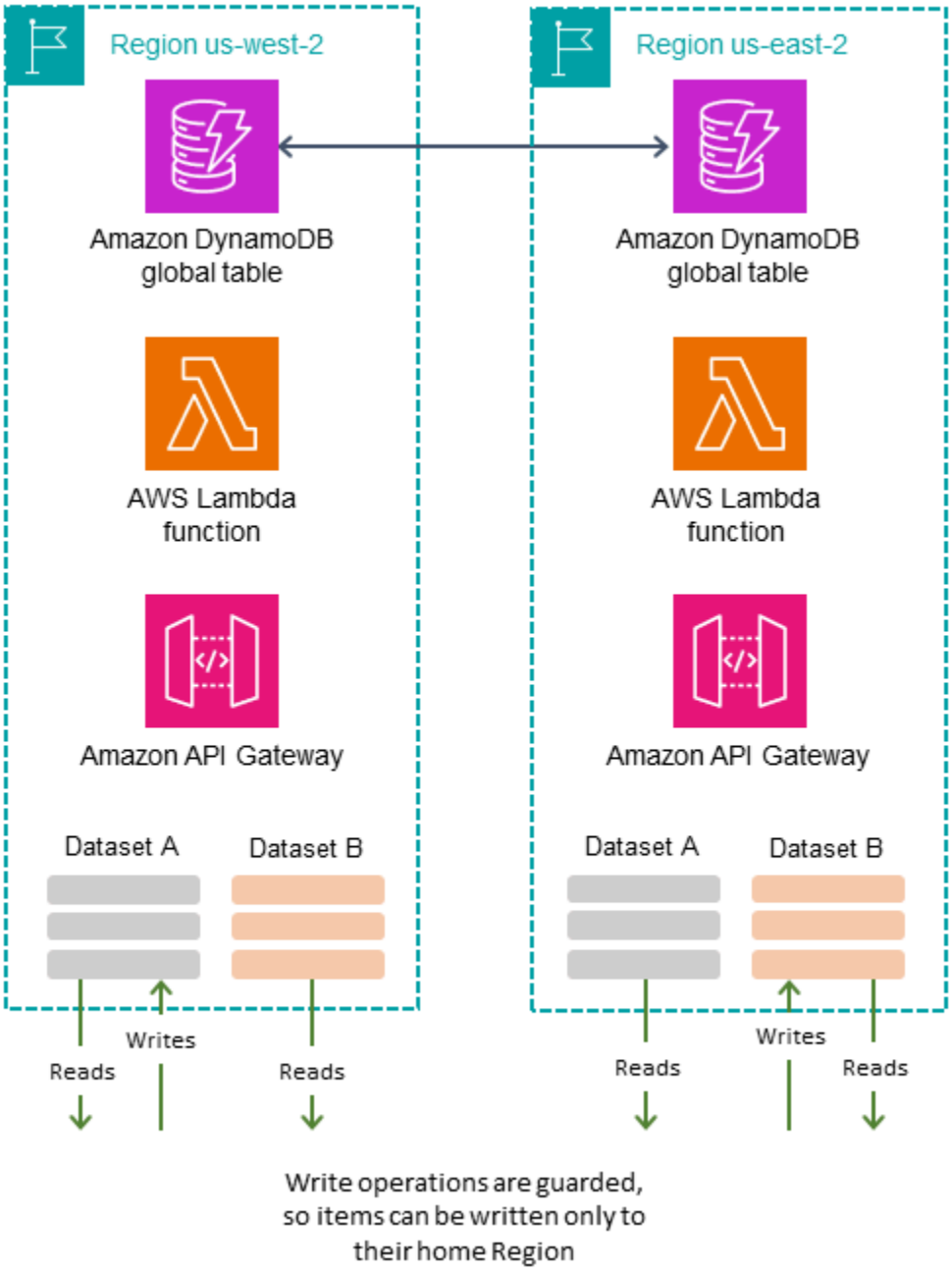
当您使用全局表进行低延迟、全球分布式读取操作时，使用写入到一个区域模式效果很好。例如，一家大型社交媒体公司需要在全球每个地区提供相同的参考数据。他们不经常更新数据，但是当他们更新数据时，他们只写入一个区域，以避免任何潜在的写入冲突。始终允许任何区域进行读取操作。

再举一个例子，以前面讨论的实施每日现金返还计算的金融服务公司为例。他们使用写入任何区域模式来计算余额，但使用一种区域模式写入来跟踪现金返还付款。如果他们想每花费10美元奖励一分钱，就必须Query对前一天的所有交易进行奖励，计算总支出，将现金返还决定写入新表，删除查询过的一组物品以将其标记为已消费，然后将其替换为单一物品，用于存储本应用于第二天计算的所有剩余物品。这项工作需要事务，因此在写入一个区域模式时效果更好。只要工作负载不可能重叠，应用程序就可以混合写入模式，即使在同一张表上也是如此。

## 写入您的区域模式（混合主模式）

写入您的区域写入模式将不同的数据子集分配给不同的主区域，并且仅允许通过其主区域对项目进行写入操作。此模式是主动-被动，但会根据物品分配主动区域。每个区域都是其自己的非重叠数据集的主要数据集，写入操作必须受到保护以确保位置正确。

此模式与写入到一个区域类似，不同之处在于它支持低延迟的写入操作，因为与每个用户相关的数据可以放置在离该用户更近的网络上。它还可以在区域之间更均匀地分布周围的基础架构，并且在故障转移情况下构建基础架构所需的工作更少，因为所有区域的部分基础架构都已处于活动状态。




您可以通过多种方式确定物品的起始区域：

- 固有的：数据的某些方面，例如特殊属性或嵌入在其分区键中的值，使其主区域变得清晰。该技术在博客文章使用区域固定为Amazon DynamoDB 全局表中的项目设置主区域中进行了描述。
- 协商：每个数据集的主区域是通过某种外部方式协商的，例如使用维护分配的单独全球服务进行协商。该任务可能有有限的期限，之后需要重新协商。

- 面向表：您创建的全局表数量与复制区域相同，而不是创建单个复制全局表。每个表的名称都指示其主区域。在标准操作中，所有数据都写入主区域，而其他区域则保留只读副本。在故障转移期间，另一个区域会暂时对该表执行写入任务。

例如，假设你在一家游戏公司工作。您需要为全球所有游戏玩家提供低延迟的读取和写入操作。你将每位玩家分配到离他们最近的区域。该区域负责所有读取和写入操作，确保了高度read-after-write的一致性。但是，当玩家旅行或他们的家乡地区发生故障时，他们的数据的完整副本可以在备用区域中获得，并且可以将玩家分配到不同的主区域。

再举一个例子，假设你在一家视频会议公司工作。每个电话会议的元数据都分配给特定区域。呼叫者可以使用离他们最近的区域以实现最低的延迟。如果出现区域中断，则使用全局表可以快速恢复，因为系统可以将呼叫的处理转移到已经存在数据副本的其他区域。

# 全局表的路由策略

或许全局表部署中最复杂的部分是管理请求路由。请求必须首先从终端用户发送到以某种方式选择和路由的区域。该请求在该区域遇到一些服务堆栈，包括可能由 AWS Lambda 函数、容器或 Amazon Elastic Compute Cloud (AmazonEC2) 节点支持的负载均衡器以及可能包括其他数据库在内的其他服务组成的计算层。该计算层与 DynamoDB 通信。它应该使用该地区的本地终端节点来做到这一点。全局表中的数据会复制到所有其他参与区域，并且每个区域在其 DynamoDB 表周围都有类似的服务堆栈。

全局表为不同区域中的每个堆栈提供具有相同数据的本地副本。如果本地 DynamoDB 表出现问题，您可以考虑在单个区域中设计单个堆栈，并预计会远程调用辅助区域的 DynamoDB 端点。这不是最佳实践。与跨区域关联的延迟可能比本地访问的延迟高 100 倍。一 back-and-forth 连串的 5 个请求在本地执行时可能需要几毫秒，而在穿越地球时则需要几秒钟。最好将终端用户路由到另一个区域进行处理。为了确保弹性，您需要跨多个区域进行复制：计算层和数据层的复制。

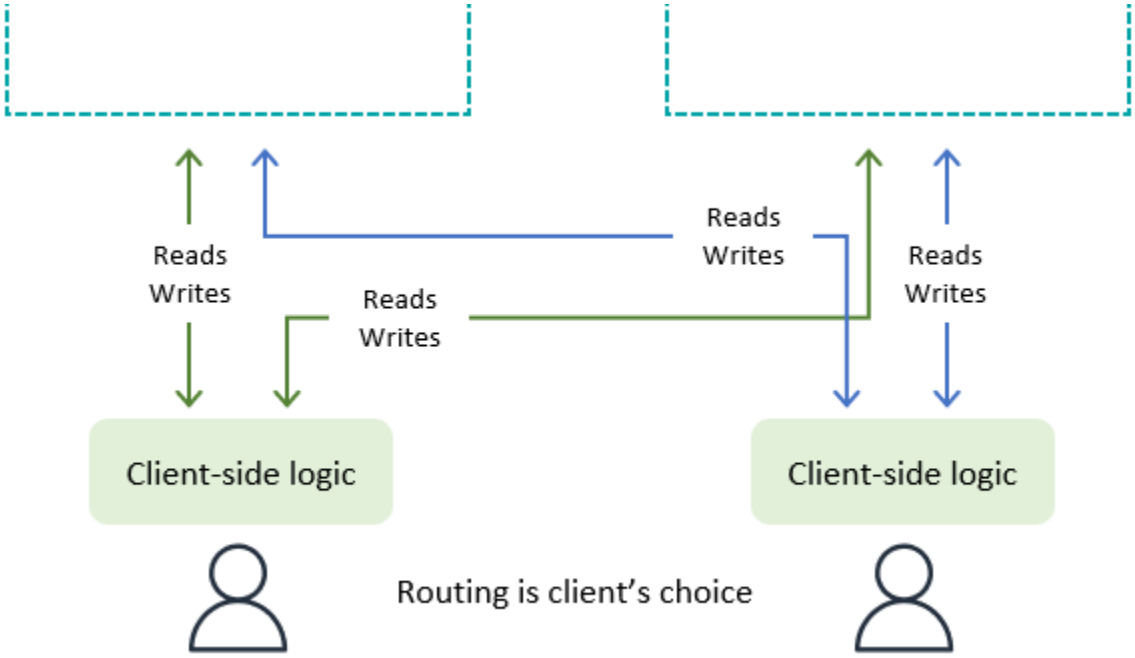
有许多方法可以将最终用户请求路由到某个区域进行处理。正确的选择取决于您的写入模式和故障转移注意事项。本节讨论四个选项：客户端驱动、计算层、Amazon Route 53 和。 AWS Global Accelerator

主题



# 客户端驱动的请求路由

通过客户端驱动的请求路由，最终用户客户端（应用程序 JavaScript、带有的网页或其他客户端）会跟踪有效的应用程序终端节点（例如，Amazon API Gateway 终端节点而不是字面上的 DynamoDB 终端节点），并使用自己的嵌入式逻辑来选择要与之通信的区域。它可以根据随机选择、最低观测延迟、观测到的最高带宽测量值或本地执行的运行状况检查进行选择。




作为一项优势，客户端驱动的请求路由可以适应诸如现实世界的公共互联网流量状况之类的因素，以便在发现任何性能下降时切换区域。客户端必须知道所有潜在的端点，但启动新的区域端点并不常见。

通过写入任何区域模式，客户端可以单方面选择其首选端点。如果客户端对一个区域的访问受到妨碍，则客户端可以路由到另一个端点。

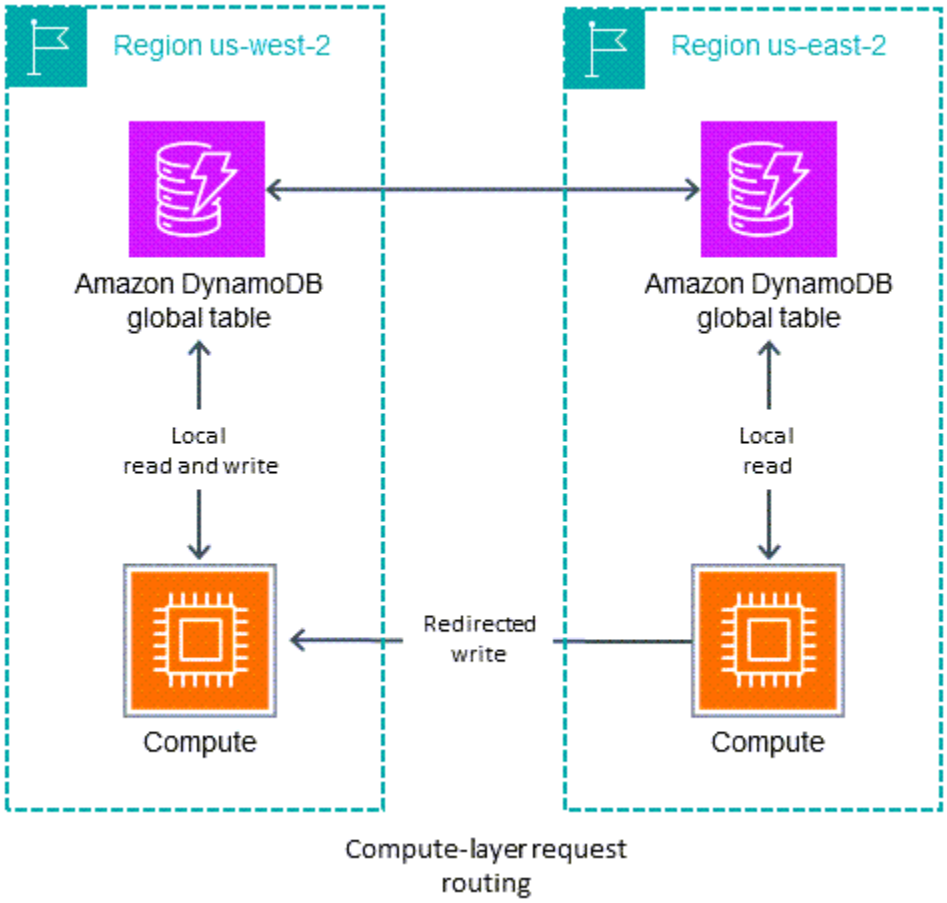
在写入一个区域模式下，客户端需要一种机制来将其写入请求路由到当前处于活动状态的区域。这可能是一种基本机制，例如根据经验测试哪个区域目前正在接受写入请求（注意任何写入拒绝并回退到备用写入请求）。或者它可能是一个复杂的机制，例如使用全局协调器来查询当前的应用程序状态（可能建立在 Amazon Application Recovery Controller (ARCARC) () 路由控制之上，它提供了一个由五个区域、法定驱动的系统来维护全局状态以满足此类需求）。客户端可以决定读取请求是可以发送到任何区域以实现最终一致性，还是必须路由到活动区域以获得强一致性。

在写入您的区域模式下，客户端需要确定其正在使用的数据集的主区域。例如，如果客户端对应一个用户帐户，并且每个用户帐户都位于一个区域，则该客户端可以从全局登录系统请求适当的端点分配以与其凭据一起使用。

例如，一家帮助用户通过网络管理其业务财务的金融服务公司使用带有写入您的 “区域” 模式的全局表。每个用户都必须登录到中央服务。该服务会返回证书以及这些证书将起作用的区域的终端节点。返回的区域基于用户数据集当前所在的位置。凭证在短时间内有效。之后，网页会自动协商新的登录信息，从而有可能将用户的活动重定向到新区域。

# 计算层请求路由

在计算层请求路由中，在计算层中运行的代码决定是在本地处理请求还是将其传递给在另一个区域中运行的自身副本。当您使用写入一个区域模式时，计算层可能会检测到它不是活动区域，并允许本地读取操作，同时将所有写入操作转发到另一个区域。此计算层代码必须了解数据拓扑和路由规则，并根据指定哪些区域对哪些数据处于活动状态的最新设置可靠地强制执行这些规则。区域内的外部软件堆栈不必知道微服务是如何路由读取和写入请求的。在稳健的设计中，接收区域会验证它是否为写入操作的当前主区域。如果不是，则会生成一个错误，表明需要更正全局状态。如果主区域处于更改过程中，则接收区域也可能将写入操作缓冲一段时间。在所有情况下，区域中的计算堆栈仅写入其本地 DynamoDB 端点，但计算堆栈可能会相互通信。

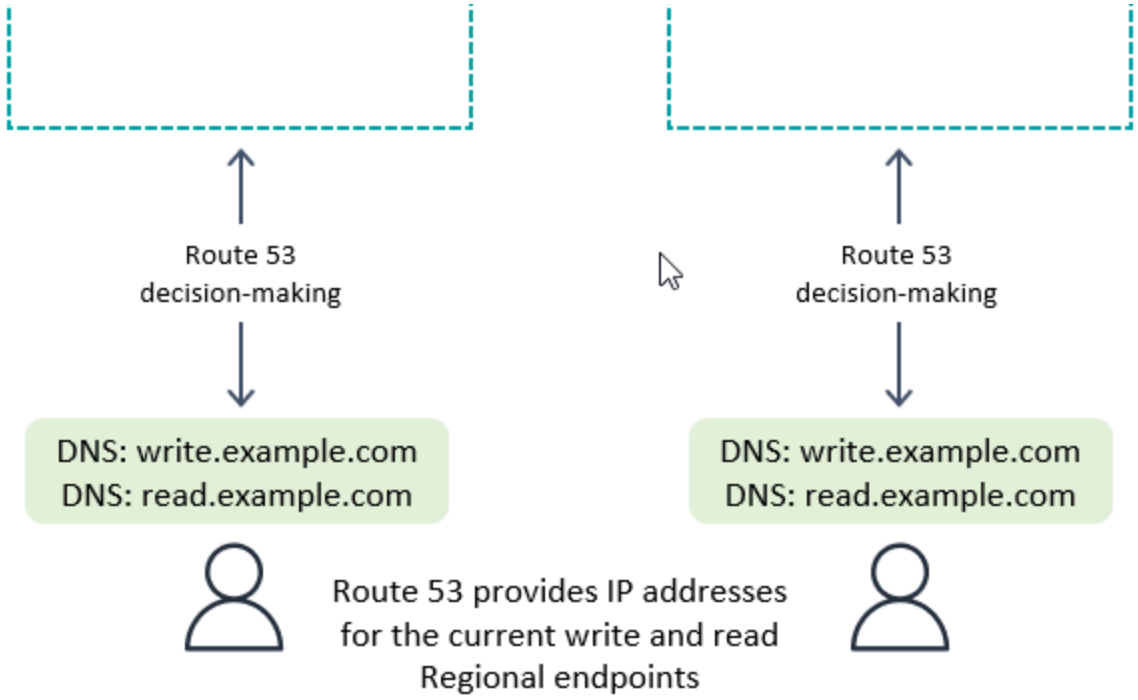



Vanguard Group 使用一个名为全球协调和状态工具 (GOaST) 的系统和一个名为全球多区域库 (GMRlib) 的库来进行此路由流程，如re: Invent 2022上所述。他们使用 follow-the-sun 单一的主模型。GOaST维护全局状态，类似于上一节中讨论的ARC路由控制。它使用全局表来跟踪哪个区域是主区域，以及何时安排下一个主交换机。所有读取和写入操作都要通过GMRlib，与之协调GOaST。GMRlib允许在本地以低延迟执行读取操作。对于写入操作，请GMRlib检查本地区域是否为当前的主区域。如果是，则写入操作将直接完成。如果不是，则将写入任务GMRlib转发到主区域

GMRlib中的。该接收库确认它也将自己视为主区域，如果不是，则会引发错误，这表明全局状态存在传播延迟。这种方法不直接写入远程 DynamoDB 端点，从而提供了验证方面的好处。

## Route 53 请求路由

亚马逊 Route 53 是一种域名服务 (DNS) 技术。在 Route 53 中，客户端通过查找已知DNS域名来请求其终端节点，而 Route 53 会返回与其认为最合适的区域终端节点相对应的 IP 地址。Route 53 有一长串用于确定相应区域的路由策略。它还可以进行故障转移路由，将流量从运行状况检查失败的区域路由出去。




通过写入任何区域模式，或者如果与后端的计算层请求路由结合使用，Route 53 可以完全自由地根据任何复杂的内部规则返回区域，例如在最近的网络或地理位置上选择区域，或者任何其他选择。

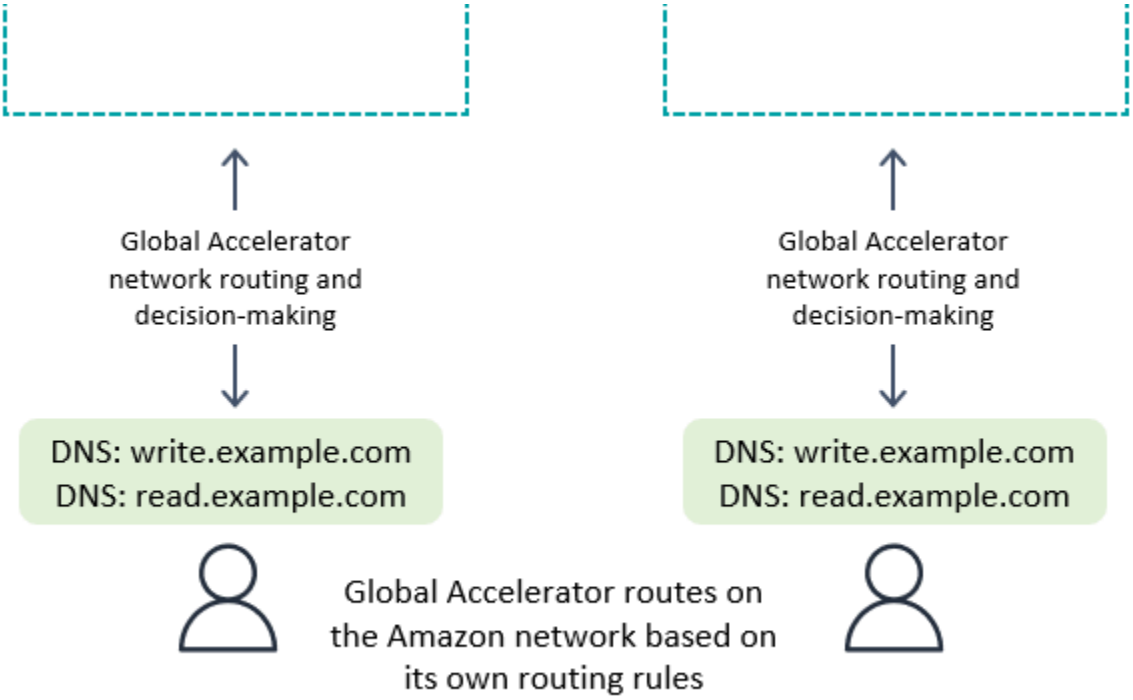
在写入一个区域模式下，您可以将 Route 53 配置为返回当前处于活动状态的区域（使用ARC）。如果客户端想要连接到被动区域（例如，用于读取操作），则可以查找其他DNS名称。

> **Note**
>
> 客户端将 Route 53 的响应中的 IP 地址缓存一段时间，该时间由域名上的生存时间 (TTL) 设置指示。较长的恢复时间目标会TTL延长所有客户端识别新端点的恢复时间目标 (RTO)。60 秒的值通常用于失效转移。并非所有软件都能完全遵守到DNSTTL期要求，并且可能存在多个DNS缓存级别，例如在操作系统、虚拟机和应用程序中。

在写入区域模式下，除非您同时使用计算层请求路由，否则最好避开 Route 53。

## Global Accelerator 请求路由

使用客户端AWS Global Accelerator在 Route 53 中查找众所周知的域名。但是，客户端不会取回与区域端点对应的 IP 地址，而是取回路由到最近 AWS 边缘位置的任播静态 IP 地址。从该边缘站点开始，所有流量都将通过私有 AWS 网络路由到由Global Accelerator中维护的路由规则选择的区域中的某个端点（网络负载均衡器、应用程序负载均衡器、EC2实例或弹性 IP 地址）。与基于 Route 53 规则的路由相比，Global Accelerator 请求路由的延迟较低，因为它减少了公共互联网上的流量。此外，由于 Global Accelerator 不依赖DNSTTL到期时间来更改路由规则，因此它可以更快地调整路由。




通过写入任何区域模式，或者与后端的计算层请求路由结合使用，Global Accelerator 可以无缝运行。客户端连接到最近的边缘站点，不必担心哪个区域会收到请求。

在写入一个区域模式下，全球加速器路由规则必须向当前处于活动状态的区域发送请求。您可以使用运行状况检查，人为地报告任何未被全局系统视为主动区域的区域上的故障。同样DNS，如果读取请求可以来自任何区域，则可以使用备用DNS域名来路由读取请求。

在写入区域模式下，除非您同时使用计算层请求路由，否则最好避免使用全球加速器。

# 全局表的类型

撤离区域是将活动（通常是写入活动，也可能是读取活动）从该区域迁移出去的过程。

## 撤离实时区域

您可能出于多种原因决定撤离实时区域：作为常规业务活动的一部分（例如，如果您使用的是，写入到一个区域模式）follow-the-sun、由于业务决定更改当前活跃区域、响应 DynamoDB 之外的软件堆栈故障，或者您遇到一般问题，例如区域内的延迟时间比平时长。

在写入任何区域模式下，撤出实时区域很简单。您可以使用任何路由系统将流量路由到备用区域，并让已在撤离区域中发生的写入操作照常复制。

在写入到一个区域和写入您的区域模式时，在新的活动区域开始写入操作之前，必须确保对活动区域的所有写入操作均已完全记录、流处理和全局传播，以确保future 写入操作将根据最新版本的数据进行处理。

假设区域 A 处于主动状态，区域 B 处于被动状态（无论是对于整个表，还是对于以区域 A 为主区域的项目）。执行撤离的典型机制是暂停对 A 的写入操作，等待足够长的时间让这些操作完全传播到 B，更新架构堆栈以识别 B 处于主动状态，然后恢复对 B 的写入操作。没有任何指标可以绝对肯定地表明区域 A 已将其数据完全复制到区域 B。如果区域 A 运行状况正常，暂停对区域 A 的写入操作并等待 `ReplicationLatency` 指标的最近最大值的 10 倍，通常足以确定复制完成。如果区域 A 运行状况不佳且显示延迟增加的其他区域，则应为等待时间选择更大的倍数。

## 撤离离线区域

有一个特殊情况需要考虑：如果区域 A 在没有通知的情况下完全离线怎么办？ 这极不可能，但仍应考虑。如果发生这种情况，则区域 A 中尚未传播的任何写入操作都将保留，并在区域 A 恢复在线后进行传播。写入操作不会丢失，但它们的传播会被无限期延迟。

在这种情况下，如何继续操作将由应用程序决定。为了实现业务连续性，写入操作可能需要继续指向新的主区域 B。但是，如果区域 B 中的某个项目收到更新，而针对该项目的写入操作有来自区域 A 的挂起传播，则在以最后写入者为准模型下，该传播将被抑制。区域 B 中的任何更新都可能抑制传入的写入请求。

通过写入任何区域模式，读取和写入操作可以在区域 B 中继续，相信区域 A 中的项目最终会传播到区域 B，并认识到在区域 A 恢复联机之前可能会丢失项目。如果可能，例如使用等同写入操作，则应考

虑重放最近的写入流量（例如，使用上游事件源）以填补任何可能缺失的写入操作的空白，让最后的写入操作获胜的冲突解决方案抑制传入写入操作的最终传播。

在其他写作模式中，你必须考虑在多大程度上可以稍微out-of-date看一下世界然后继续工作。在区域 A 恢复在线之前，将丢失一些持续时间很短的写入操作（由 ReplicationLatency 跟踪）。业务能否向前推进？ 在某些使用案例中可以向前推进，但在另一些使用案例中，如果没有额外的缓解机制，则可能不行。

例如，假设即使在某个地区完全中断之后，您也必须不间断地保持可用的信用余额。您可以将余额拆分为两个不同的项目，一个位于区域 A，一个位于区域 B，然后从可用余额的一半开始。这将使用写入您的区域模式。在每个区域中处理的事务性更新将写入余额的本地副本。如果区域 A 变为完全离线，则仍然可以在区域 B 中继续进行事务处理，写入操作将仅限于区域 B 中持有的余额部分。当余额变低或必须重新计算信贷余额时，像这样拆分余额会带来复杂性，但它确实提供了一个安全业务恢复的示例，即使存在不确定的待处理写入操作。

再举一个例子，假设你正在捕获 Web 表单数据。您可以使用乐观并发控制 (OCC) 为数据项分配版本，并将最新版本作为隐藏字段嵌入到 Web 表单中。在每次提交时，只有当数据库中的版本仍然与构建表单所依据的版本相匹配时，写入操作才会成功。如果版本不匹配，则可以根据数据库中的当前版本刷新（或谨慎合并）Web 表单，然后用户可以再次继续。OCC 模型通常可以防止其他客户端覆盖并生成新版本的数据，但它也可以在失效转移期间提供帮助，此时客户端可能会遇到更旧版本的数据。假设您使用时间戳作为版本。表单最初是在 12:00 针对区域 A 构建的，但是（故障转移后）尝试写入区域 B 并注意到数据库中的最新版本是 11:59。在这种情况下，客户端可以等待 12:00 版本传播到区域 B，然后在该版本之上进行写入；也可以在 11:59 上构建并创建新的 12:01 版本（写入后，该版本将在区域 A 恢复后抑制传入版本）。

第三个例子，一家金融服务公司在 DynamoDB 数据库中保存有关客户账户及其金融交易的数据。如果区域 A 完全中断，他们希望确保与其账户相关的任何写入活动要么在区域 B 中完全可用，要么他们希望将已知的部分账户隔离，直到区域 A 恢复联机。他们没有暂停所有业务，而是决定只对他们确定有未传播交易的一小部分账户暂停业务。为了实现这一点，他们使用了第三个区域，我们称为区域 C。在他们处理区域 A 中的任何写入操作之前，他们在区域 C 中简要地汇总了那些待处理的操作（例如，一个账户的新交易数量）。这个摘要足以让区域 B 确定其视图是否完全是最新的。从在区域 C 中写入操作，到区域 A 接受写入操作并且区域 B 收到写入操作之前，此操作实际上锁定了该账户。除非作为失效转移过程的一部分，否则不会使用区域 C 中的数据，之后，区域 B 可以将其数据与区域 C 进行交叉核对，以检查其账户是否已过期。在区域 A 恢复将部分数据传播到区域 B 之前，这些帐户将被标记为隔离。如果区域 C 出现故障，则可以启动新的区域 D 以供改用。区域 C 中的数据非常短暂，几分钟后，区域 D 将有足够的运行中写入操作的up-to-date记录，因此完全有用。如果区域 B 出现故障，区域 A 可以继续接受与区域 C 合作的写入请求。这家公司愿意接受延迟更高的写入（写入到两个区域：C，然后是 A），并且很幸运有了一个可以简洁汇总账户状态的数据模型。

# 全局表的吞吐能力规划

将流量从一个区域迁移到另一个区域时，需要仔细考虑容量方面的 DynamoDB 表设置。

以下是管理写入容量的一些注意事项：

- 全局表必须处于按需模式，或在启用自动扩缩的情况下进行预调配。
- 如果使用自动扩缩进行预调配，则会跨区域复制写入设置（最小、最大和目标利用率）。尽管自动扩缩设置已同步，但实际的预调配写入容量可能会在区域间独立浮动。
- 您可能会看到不同的预配置写入容量的原因之一是生存时间 (TTL) 功能。在 DynamoDB TTL 中启用时，您可以指定一个属性名称，其值表示项目的过期时间，采用 Unix 纪元时间格式（以秒为单位）。在该时间之后，DynamoDB 可以删除该项目而不会产生写入成本。使用全局表，您可以在任何区域TTL中进行配置，并且该设置会自动复制到与全局表关联的其他区域。当某件商品符合通过TTL规则删除的条件时，可以在任何地区完成该工作。删除操作是在不占用源表上的写入单位的情况下执行的，但是副本表将获得该删除操作的复制写入，并且会产生复制的写入单位成本。
- 如果您使用的是 auto scaling，请确保最大预配置写入容量设置足够高，足以处理所有写入操作以及所有潜在的TTL删除操作。自动扩缩根据每个区域的写入消耗量调整每个区域。按需表没有最大预调配写入容量设置，但表级别的最大写入吞吐量限制指定了按需表将允许的最大持续写入容量。原定设置限制为 40000，但可以调整。我们建议您将其设置得足够高，以处理按需表可能需要的所有TTL写入操作（包括写入操作）。设置全局表时，此值在所有参与区域间必须相同。

以下是管理读取容量的一些注意事项：

- 允许不同区域之间的读取容量管理设置有所不同，因为假设不同的区域可能有独立的读取模式。当您首先向表添加全局副本时，将传播源区域的容量。创建后，您可以调整读取容量设置，这些新设置不会传输到另一端。
- 使用 DynamoDB Auto Scaling 时，请确保最大预调配读取容量设置足够高，足以处理所有区域的所有读取操作。在标准操作期间，读取容量可能会分布在各个区域之间，但在失效转移期间，表应该能够自动适应增加的读取工作负载。按需表没有最大预调配读取容量设置，但表级别的最大读取吞吐量限制指定了按需表将允许的最大持续读取容量。原定设置限制为 40000，但可以调整。我们建议您将其设置得足够高，以处理该表可能需要的所有读取操作（当所有读取操作都路由到此单个区域时）。
- 如果一个区域中的表通常不会接收读取流量，但在失效转移后可能必须吸收大量读取流量，则可以提高表的预调配读取容量，等待表完成更新，然后再次向下预调配表。您可以将表保留为预调配模式，也可以将其切换到按需模式。这会预热表以接受更高级别的读取流量。

ARC无论您是否使用 Route 53 路由请求，都具有就绪性检查，可用于确认 DynamoDB 区域是否具有相似的表设置和账户配额。这些准备情况检查还可以帮助您调整账户级别的配额以使其匹配。

# 全局表的准备清单

在部署全局表时，请使用下面的决策和任务核对清单。

- 确定全局表应涉及多少个区域以及哪些区域。
- 确定应用程序的写入模式。
- 根据您的写作模式规划您的路由策略。
- 根据您的写作模式和路线策略定义您的疏散计划。
- 捕获有关每个区域的运行状况、延迟和错误的指标。有关 DynamoDB 指标的列表，请参阅博客文章监控亚马逊 DynamoDB AWS 以提高运营意识。您还应该使用合成加那利（旨在检测故障的人为请求）以及对客户流量的实时观察。并非所有问题都出现在 DynamoDB 指标中。
- 在 `ReplicationLatency` 中为任何持续增加设置警报。增加可能表示意外配置错误，即全局表在不同区域中具有不同的写入设置，这会导致复制请求失败和延迟增加。这也可能表明存在区域中断。一个很好的例子是，如果最近的平均值超过 180000 毫秒，则生成警报。您可能还会观察到 `ReplicationLatency` 降至 0，这表示复制已停止。
- 为每个全局表分配足够的最大读取和写入设置。
- 确定您要撤离某个地区的条件。如果决定涉及人为判断，请记录所有考虑因素。这项工作应该事先仔细完成，而不是在压力下匆匆了事。
- 为撤离某个区域时必须采取的每项措施制定一份运行手册。通常，全局表所涉及的工作非常少，但是移动堆栈的其余部分可能很复杂。

> **Note**
>
> 对于故障转移程序，最佳做法是仅依赖数据平面操作而不依赖控制平面操作，因为在区域故障期间，某些控制平面操作可能会降级。有关更多信息，请参阅 AWS 博客文章使用 Amazon DynamoDB 全局表构建弹性应用程序：第 4 部分。

- 定期测试运行手册的各个方面，包括区域撤离。未经测试的运行手册是不可靠的。
- 考虑使用AWS Resilience Hub来评估整个应用程序（包括全局表）的弹性。该服务通过其仪表板提供应用程序组合弹性状态的全面视图。
- 考虑使用ARC就绪性检查来评估应用程序的当前配置，并跟踪与最佳实践的任何偏差。
- 在编写与 Route 53 或全球加速器一起使用的运行状况检查时，请进行一组覆盖整个数据库流的调用。如果您将检查限制为仅确认 DynamoDB 终端节点已启动，则无法涵盖许多故障模式，例如

AWS Identity and Access Management IAM () 配置错误、代码部署问题、DynamoDB 之外的堆栈故障、高于平均水平的读取或写入延迟等。

# 全局表常见问题

本节提供了有关 DynamoDB 全局表的常见问题解答。

## 全局表的定价是多少？

- 对传统 DynamoDB 表的写入操作按写入容量单位（WCU，用于预置表）或写入请求单位（WRU，用于按需表）定价。如果您写入一个 5KB 项目，则会产生 5 个单位的费用。对全局表的写入按复制写入容量单位（rWCU，用于预置表）或复制写入请求单位（rWRU，用于按需表）定价。
- rWCU 和 rWRU 包括管理复制所需的流式基础设施的成本。因此，它们的价格比 WCU 和 WRU 高出 50%。跨区域数据传输费用适用。
- 在每个直接写入项目或通过复制写入项目的地区，都会产生 rwCu 和 rwRU 费用。
- 写入全局二级索引（GSI）被视为本地写入操作，使用常规写入单位。
- 目前 rWCU 没有可用的预留容量。对于 GSI 消耗写入单位的表，购买预留容量可能仍有好处。
- 当您将新区域添加到全局表时，DynamoDB 会自动引导新区域，并根据表的大小（GB）向您收取费用，就像表还原一样。还会收取跨区域数据传输费用。

## 全局表支持哪些区域？

全局表支持全部 AWS 区域。

## 如何使用全局表处理 GSI？

在全局表（当前，版本 2019）中，当您在一个区域中创建 GSI 时，它会在其他参与区域中自动创建并自动回填。

## 如何停止复制全局表？

您可以像删除任何其他表一样删除副本表。删除全局表将停止复制到该区域，并删除保留在该区域中的表副本。但是，不能在将表的副本保留为独立实体时停止复制，也不能暂停复制。

## Amazon DynamoDB Streams 如何与全局表交互？

每个全局表都基于其所有写入操作生成一个独立的流，而无论这些写入是从何处开始的。您可以选择在一个区域或在所有区域中（独立）使用 DynamoDB 流。如果您想要处理本地而不是复制的写入操作，

则可以向每个项目添加您自己的区域属性，以确定写入区域。然后，您可以使用 AWS Lambda 事件筛选条件，以便只调用 Lambda 函数来处理本地区域中的写入操作。这有助于执行插入和更新操作，但不能执行删除操作。

## 全局表如何处理事务？

事务操作仅在最初发生写入操作的区域内提供原子性、一致性、隔离性和持久性（ACID）保证。全局表中不支持跨区域的事务。例如，如果您有一个全局表，该表在美国东部（俄亥俄州）和美国西部（俄勒冈州）区域中具有副本，并且在美国东部（俄亥俄州）区域中执行 `TransactWriteItems` 操作，则在复制更改时，可能会在美国西部（俄勒冈州）区域观察到部分完成的事务。更改仅在源区域中提交后才复制到其他区域。

## 全局表如何与 DynamoDB Accelerator（DAX）缓存交互？

全局表通过直接更新 DynamoDB 绕过 DAX，因此 DAX 并不知道它保存的是陈旧数据。DAX 缓存只有在缓存的 TTL 过期时才会刷新。

## 表上的标签会传播吗？

不，标签不会自动传播。

## 我应该备份所有区域中的表，还是只备份一个区域中的表？

答案取决于备份的目的。

- 如果您想确保数据的耐久性，DynamoDB 已经提供了这种保护措施。该服务可确保耐久性。
- 如果您想保留历史记录的快照（例如，为了符合法规要求），备份一个区域中的表就应该足够了。您可以使用 AWS Backup 将备份复制到其他区域。
- 如果您想恢复错误删除或修改的数据，请在一个区域中使用 Dynamo DB point-in-time 恢复 (PITR)。

## 如何使用 AWS CloudFormation 部署全局表？

- CloudFormation 将 DynamoDB 表和全局表表示为两个独立的资源：`AWS::DynamoDB::Table` 和。`AWS::DynamoDB::GlobalTable`一种方法是使用 `GlobalTable` 构造来创建所有可能为全局的表，最初将其保留为独立的表，然后在以后需要时再添加区域。

- 在中 CloudFormation，无论副本数量多少，每个全局表都由单个区域中的单个堆栈控制。部署模板时，作为单个堆栈操作的一部分， CloudFormation 创建和更新所有副本。您不应在多个区域中部署相同的 AWS::DynamoDB::GlobalTable 资源。这样做会导致错误，不受支持。如果在多个区域中部署应用程序模板，则可以使用条件在单个区域中创建 `AWS::DynamoDB::GlobalTable` 资源。或者，您可以选择在独立于应用程序堆栈的堆栈中定义 `AWS::DynamoDB::GlobalTable` 资源，并确保仅将该资源部署到单个区域。
- 如果您有一个常规表，并且想要将其转换为全局表，同时由 CloudFormation以下人员管理：将删除策略设置为`Retain`，从堆栈中移除表，在控制台中将该表转换为全局表，然后将全局表作为新资源导入堆栈。有关更多信息，请参阅AWS GitHub 存储库 amazon-dynamodb-table-to-global-table-cdk。
- 目前不支持跨账户复制。

# 结论和资源

DynamoDB 全局表的控件很少，但仍需要仔细考虑。您必须确定您的写入模式、路由模型和撤离过程。您必须针对每个区域对应用程序进行检测，并准备好调整路由或执行撤离，以维护全局运行状况。奖励是拥有一个具有低延迟读取和写入操作的全球分布式数据集，其可用性为 99.999%。

有关 DynamoDB 全局表的更多信息，请参阅以下资源：

- Amazon DynamoDB 文档
- Amazon Route 53 应用程序恢复控制器
- ARC准备情况检查（AWS 文档）
- Route 53 路由策略（AWS 文档）
- AWS Global Accelerator
- DynamoDB 服务等级协议
- AWS 多区域基础知识（AWS 白皮书）
- 数据弹性设计模式与 AWS（re AWS : Invent 2022 演示文稿）
- 富达投资和Reltio如何利用亚马逊 DynamoDB 实现现代化（re: Invent 2022 演示文稿AWS ）
- 多区域设计模式和最佳实践（re AWS : Invent 2022 演示文稿）
- 开@@ 启灾难恢复 (DR) 架构 AWS，部分III：指示灯和暖待机（AWS 博客文章）
- 使用区域锁定为亚马逊 DynamoDB 全球AWS 表中的项目设置主区域（博客文章）
- 监控 Amazon DynamoDB 以提高运营意识AWS （博客文章）
- 扩展 DynamoDB：分区、热键和分割以提高热量如何影响性能AWS （博客文章）

# 文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 RSS 源。

| 变更 | 说明 | 日期 |
| --- | --- | --- |
| 更新的 AWS Global Accelerator 信息 | 更正了全球加速器请求路由的端点。 | 2024年3月14日 |
| 更新的 AWS 区域 支持信息 | 更新了常见问题以表明全局表格现在支持所有 AWS 区域。 | 2023 年 11 月 15 日 |
| 初次发布 | — | 2023 年 5 月 19 日 |

# AWS 规范性指导词汇表

以下是 AWS 规范性指导提供的策略、指南和模式中的常用术语。若要推荐词条，请使用术语表末尾的提供反馈链接。

## 数字

7 R

将应用程序迁移到云中的 7 种常见迁移策略。这些策略以 Gartner 于 2011 年确定的 5 R 为基础，包括以下内容：

- 重构/重新架构 - 充分利用云原生功能来提高敏捷性、性能和可扩展性，以迁移应用程序并修改其架构。这通常涉及到移植操作系统和数据库。示例：将您的本地 Oracle 数据库迁移到兼容 Amazon Aurora PostgreSQL 的版本。
- 更换平台 - 将应用程序迁移到云中，并进行一定程度的优化，以利用云功能。示例：在中将您的本地 Oracle 数据库迁移到适用于 Oracle 的亚马逊关系数据库服务 (Amazon RDS) AWS Cloud。
- 重新购买 - 转换到其他产品，通常是从传统许可转向 SaaS 模式。示例：将您的客户关系管理 (CRM) 系统迁移到 Salesforce.com。
- 更换主机（直接迁移）- 将应用程序迁移到云中，无需进行任何更改即可利用云功能。示例：在中的 EC2 实例上将您的本地 Oracle 数据库迁移到 Oracle AWS Cloud。
- 重新定位（虚拟机监控器级直接迁移）：将基础设施迁移到云中，无需购买新硬件、重写应用程序或修改现有操作。您可以将服务器从本地平台迁移到同一平台的云服务。示例：将Microsoft Hyper-V应用程序迁移到 AWS。
- 保留（重访）- 将应用程序保留在源环境中。其中可能包括需要进行重大重构的应用程序，并且您希望将工作推迟到以后，以及您希望保留的遗留应用程序，因为迁移它们没有商业上的理由。
- 停用 - 停用或删除源环境中不再需要的应用程序。

## A

ABAC

请参阅基于属性的访问控制。

抽象服务

参见托管服务。

**酸**

参见原子性、一致性、隔离性、耐久性。

主动-主动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步（通过使用双向复制工具或双写操作），两个数据库都在迁移期间处理来自连接应用程序的事务。这种方法支持小批量、可控的迁移，而不需要一次性割接。与主动-被动迁移相比，它更灵活，但需要更多的工作。

主动-被动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步，但在将数据复制到目标数据库时，只有源数据库处理来自连接应用程序的事务。目标数据库在迁移期间不接受任何事务。

聚合函数

一个 SQL 函数，它对一组行进行操作并计算该组的单个返回值。聚合函数的示例包括SUM和MAX。

AI

参见人工智能。

AIOps

参见人工智能操作。

匿名化

永久删除数据集中个人信息的过程。匿名化可以帮助保护个人隐私。匿名化数据不再被视为个人数据。

反模式

一种用于解决反复出现的问题的常用解决方案，而在这类问题中，此解决方案适得其反、无效或不如替代方案有效。

应用程序控制

一种安全方法，仅允许使用经批准的应用程序，以帮助保护系统免受恶意软件的侵害。

应用程序组合

有关组织使用的每个应用程序的详细信息的集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是产品组合发现和分析过程的关键，有助于识别需要进行迁移、现代化和优化的应用程序并确定其优先级。

人工智能（AI）

计算机科学领域致力于使用计算技术执行通常与人类相关的认知功能，例如学习、解决问题和识别模式。有关更多信息，请参阅什么是人工智能？

人工智能运营（AIOps）

使用机器学习技术解决运营问题、减少运营事故和人为干预以及提高服务质量的过程。有关如何在 AWS 迁移策略中使用 AIOps 的更多信息，请参阅运营集成指南。

非对称加密

一种加密算法，使用一对密钥，一个公钥用于加密，一个私钥用于解密。您可以共享公钥，因为它不用于解密，但对私钥的访问应受到严格限制。

原子性、一致性、隔离性、持久性（ACID）

一组软件属性，即使在出现错误、电源故障或其他问题的情况下，也能保证数据库的数据有效性和操作可靠性。

基于属性的访问权限控制（ABAC）

根据用户属性（如部门、工作角色和团队名称）创建精细访问权限的做法。有关更多信息，请参阅 AWS Identity and Access Management (I AM) 文档 AWS中的 AB AC。

权威数据源

存储主要数据版本的位置，被认为是最可靠的信息源。您可以将数据从权威数据源复制到其他位置，以便处理或修改数据，例如对数据进行匿名化、编辑或假名化。

可用区

中的一个不同位置 AWS 区域 ，不受其他可用区域故障的影响，并向同一区域中的其他可用区提供低成本、低延迟的网络连接。

AWS 云采用框架 (AWS CAF)

该框架包含指导方针和最佳实践 AWS ，可帮助组织制定高效且有效的计划，以成功迁移到云端。AWS CAF将指导分为六个重点领域，称为视角：业务、人员、治理、平台、安全和运营。业务、人员和治理角度侧重于业务技能和流程；平台、安全和运营角度侧重于技术技能和流程。例如，人员角度针对的是负责人力资源（HR）、人员配置职能和人员管理的利益相关者。从这个角度来看，AWS CAF 为人员发展、培训和沟通提供了指导，以帮助组织为成功采用云做好准备。有关更多信息，请参阅 AWS CAF 网站和 AWS CAF 白皮书。

AWS 工作负载资格框架 (AWS WQF)

一种评估数据库迁移工作负载、推荐迁移策略和提供工作估算的工具。 AWS WQF 包含在 AWS Schema Conversion Tool (AWS SCT) 中。它用来分析数据库架构和代码对象、应用程序代码、依赖关系和性能特征，并提供评测报告。

# B

坏机器人

旨在破坏个人或组织或对其造成伤害的机器人。

BCP

参见业务连续性计划。

行为图

一段时间内资源行为和交互的统一交互式视图。您可以使用 Amazon Detective 的行为图来检查失败的登录尝试、可疑的 API 调用和类似的操作。有关更多信息，请参阅 Detective 文档中的行为图中的数据。

大端序系统

一个先存储最高有效字节的系统。另请参见字节顺序。

二进制分类

一种预测二进制结果（两个可能的类别之一）的过程。例如，您的 ML 模型可能需要预测诸如“该电子邮件是否为垃圾邮件？”或“这个产品是书还是汽车？”之类的问题

bloom 筛选条件

一种概率性、内存高效的数据结构，用于测试元素是否为集合的成员。

蓝/绿部署

一种部署策略，您可以创建两个独立但完全相同的环境。在一个环境中运行当前的应用程序版本（蓝色），在另一个环境中运行新的应用程序版本（绿色）。此策略可帮助您在影响最小的情况下快速回滚。

自动程序

一种通过互联网运行自动任务并模拟人类活动或互动的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的网络爬虫。其他一些被称为恶意机器人的机器人旨在破坏个人或组织或对其造成伤害。

僵尸网络

被恶意软件感染并受单方（称为机器人牧民或机器人操作员）控制的机器人网络。僵尸网络是最著名的扩展机器人及其影响力的机制。

分支

代码存储库的一个包含区域。在存储库中创建的第一个分支是主分支。您可以从现有分支创建新分支，然后在新分支中开发功能或修复错误。为构建功能而创建的分支通常称为功能分支。当功能可以发布时，将功能分支合并回主分支。有关更多信息，请参阅关于分支（GitHub 文档）。

破碎的玻璃通道

在特殊情况下，通过批准的流程，用户 AWS 账户 可以快速访问他们通常没有访问权限的内容。有关更多信息，请参阅 Well -Architected 指南中的 “实施破碎玻璃程序” 指示 AWS 器。

棕地策略

您环境中的现有基础设施。在为系统架构采用棕地策略时，您需要围绕当前系统和基础设施的限制来设计架构。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

缓冲区缓存

存储最常访问的数据的内存区域。

业务能力

企业如何创造价值（例如，销售、客户服务或营销）。微服务架构和开发决策可以由业务能力驱动。有关更多信息，请参阅在 AWS上运行容器化微服务白皮书中的围绕业务能力进行组织部分。

业务连续性计划（BCP）

一项计划，旨在应对大规模迁移等破坏性事件对运营的潜在影响，并使企业能够快速恢复运营。

# C

CAF

参见AWS 云采用框架。

金丝雀部署

向最终用户缓慢而渐进地发布版本。当你有信心时，你可以部署新版本并全部替换当前版本。

CCoE

参见云卓越中心。

CDC

参见变更数据捕获。

更改数据捕获（CDC）

跟踪数据来源（如数据库表）的更改并记录有关更改的元数据的过程。您可以将 CDC 用于各种目的，例如审计或复制目标系统中的更改以保持同步。

混沌工程

故意引入故障或破坏性事件来测试系统的弹性。您可以使用 AWS Fault Injection Service (AWS FIS) 来执行实验，对您的 AWS 工作负载施加压力并评估其响应。

CI/CD

查看持续集成和持续交付。

分类

一种有助于生成预测的分类流程。分类问题的 ML 模型预测离散值。离散值始终彼此不同。例如，一个模型可能需要评估图像中是否有汽车。

客户端加密

在目标 AWS 服务 收到数据之前，对数据进行本地加密。

云卓越中心（CCoE）

一个多学科团队，负责推动整个组织的云采用工作，包括开发云最佳实践、调动资源、制定迁移时间表、领导组织完成大规模转型。有关更多信息，请参阅 AWS Cloud 企业战略博客上的 CCoE 帖子。

云计算

通常用于远程数据存储和 IoT 设备管理的云技术。云计算通常与边缘计算技术相关。

云运营模型

在 IT 组织中，一种用于构建、完善和优化一个或多个云环境的运营模型。有关更多信息，请参阅构建您的云运营模型。

云采用阶段

组织迁移到以下阶段时通常会经历四个阶段 AWS Cloud：

- 项目 - 出于概念验证和学习目的，开展一些与云相关的项目
- 基础 - 进行基础投资以扩大云采用率（例如，创建登录区、定义 CCoE、建立运营模型）

- 迁移 - 迁移单个应用程序
- 重塑 - 优化产品和服务，在云中创新

Stephen Orban在 AWS Cloud 企业战略博客的博客文章《云优先之旅和采用阶段》中定义了这些阶段。有关它们与 AWS 迁移策略的关系的信息，请参阅迁移准备指南。

CMDB

参见配置管理数据库。

代码存储库

通过版本控制过程存储和更新源代码和其他资产（如文档、示例和脚本）的位置。常见的云存储库包括 GitHub 或 AWS CodeCommit。每个版本的代码都称为一个分支。在微服务结构中，每个存储库都专门用于一个功能。单个 CI/CD 管道可以使用多个存储库。

冷缓存

一种空的、填充不足或包含过时或不相关数据的缓冲区缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取，这比从缓冲区缓存读取要慢。

冷数据

很少访问的数据，且通常是历史数据。查询此类数据时，通常可以接受慢速查询。将这些数据转移到性能较低且成本更低的存储层或类别可以降低成本。

计算机视觉 (CV)

人工智能领域，使用机器学习来分析和提取数字图像和视频等视觉格式中的信息。例如， AWS Panorama 提供将 CV 添加到本地摄像机网络的设备，而 Amazon 则为 CV SageMaker 提供图像处理算法。

配置偏差

对于工作负载，配置会从预期状态发生变化。这可能会导致工作负载变得不合规，而且通常是渐进的，不是故意的。

配置管理数据库（CMDB）

一种存储库，用于存储和管理有关数据库及其 IT 环境的信息，包括硬件和软件组件及其配置。您通常在迁移的产品组合发现和分析阶段使用来自 CMDB 的数据。

合规性包

一系列 AWS Config 规则和补救措施，您可以汇编这些规则和补救措施，以自定义合规性和安全性检查。您可以使用 YAML 模板将一致性包作为单个实体部署在 AWS 账户 和区域或整个组织中。有关更多信息，请参阅 AWS Config 文档中的一致性包。

持续集成和持续交付（CI/CD）

自动执行软件发布过程的源代码、构建、测试、暂存和生产阶段的过程。CI/CD 通常被描述为管道。CI/CD 可以帮助您实现流程自动化、提高工作效率、改善代码质量并加快交付速度。有关更多信息，请参阅持续交付的优势。CD 也可以表示持续部署。有关更多信息，请参阅持续交付与持续部署。

CV

参见计算机视觉。

# D

静态数据

网络中静止的数据，例如存储中的数据。

数据分类

根据网络中数据的关键性和敏感性对其进行识别和分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它可以帮助您确定对数据的适当保护和保留控制。数据分类是 Well-Architecte AWS d Framework 中安全支柱的一个组成部分。有关详细信息，请参阅数据分类。

数据漂移

生产数据与用来训练机器学习模型的数据之间的有意义差异，或者输入数据随时间推移的有意义变化。数据漂移可能降低机器学习模型预测的整体质量、准确性和公平性。

传输中数据

在网络中主动移动的数据，例如在网络资源之间移动的数据。

数据网格

一种架构框架，可提供分布式、去中心化的数据所有权以及集中式管理和治理。

数据最少化

仅收集并处理绝对必要数据的原则。在中进行数据最小化 AWS Cloud 可以降低隐私风险、成本和分析碳足迹。

数据边界

AWS 环境中的一组预防性防护措施，可帮助确保只有可信身份才能访问来自预期网络的可信资源。有关更多信息，请参阅在上构建数据边界。 AWS

数据预处理

将原始数据转换为 ML 模型易于解析的格式。预处理数据可能意味着删除某些列或行，并处理缺失、不一致或重复的值。

数据溯源

在数据的整个生命周期跟踪其来源和历史的过程，例如数据如何生成、传输和存储。

数据主体

正在收集和处理其数据的个人。

数据仓库

一种支持商业智能（例如分析）的数据管理系统。数据仓库通常包含大量历史数据，通常用于查询和分析。

数据库定义语言（DDL）

在数据库中创建或修改表和对象结构的语句或命令。

数据库操作语言（DML）

在数据库中修改（插入、更新和删除）信息的语句或命令。

DDL

参见数据库定义语言。

深度融合

组合多个深度学习模型进行预测。您可以使用深度融合来获得更准确的预测或估算预测中的不确定性。

深度学习

一个 ML 子字段使用多层人工神经网络来识别输入数据和感兴趣的目标变量之间的映射。

defense-in-depth

一种信息安全方法，经过深思熟虑，在整个计算机网络中分层实施一系列安全机制和控制措施，以保护网络及其中数据的机密性、完整性和可用性。当你采用这种策略时 AWS，你会在 AWS Organizations 结构的不同层面添加多个控件来帮助保护资源。例如，一种 defense-in-depth 方法可以结合多因素身份验证、网络分段和加密。

委托管理员

在中 AWS Organizations，兼容的服务可以注册 AWS 成员帐户来管理组织的帐户并管理该服务的权限。此账户被称为该服务的委托管理员。有关更多信息和兼容服务列表，请参阅 AWS Organizations 文档中使用 AWS Organizations的服务。

部署

使应用程序、新功能或代码修复在目标环境中可用的过程。部署涉及在代码库中实现更改，然后在应用程序的环境中构建和运行该代码库。

开发环境

参见环境。

侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出警报。这些控制是第二道防线，提醒您注意绕过现有预防性控制的安全事件。有关更多信息，请参阅在 AWS上实施安全控制中的侦测性控制。

开发价值流映射 (DVSM)

用于识别对软件开发生命周期中的速度和质量产生不利影响的限制因素并确定其优先级的流程。DVSM 扩展了最初为精益生产实践设计的价值流映射流程。其重点关注在软件开发过程中创造和转移价值所需的步骤和团队。

数字孪生

真实世界系统的虚拟再现，如建筑物、工厂、工业设备或生产线。数字孪生支持预测性维护、远程监控和生产优化。

维度表

在星型架构中，一种较小的表，其中包含事实表中定量数据的数据属性。维度表属性通常是文本字段或行为类似于文本的离散数字。这些属性通常用于查询约束、筛选和结果集标注。

灾难

阻止工作负载或系统在其主要部署位置实现其业务目标的事件。这些事件可能是自然灾害、技术故障或人为操作的结果，例如无意的配置错误或恶意软件攻击。

灾难恢复 (DR)

您用来最大限度地减少灾难造成的停机时间和数据丢失的策略和流程。有关更多信息，请参阅 Well-Architected Fr ame AWS work 中的 “工作负载灾难恢复：云端 AWS 恢复”。

DML

参见数据库操作语言。

领域驱动设计

一种开发复杂软件系统的方法，通过将其组件连接到每个组件所服务的不断发展的领域或核心业务目标。Eric Evans 在其著作领域驱动设计：软件核心复杂性应对之道（Boston: Addison-Wesley Professional, 2003）中介绍了这一概念。有关如何将领域驱动设计与 strangler fig 模式结合使用的信息，请参阅使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET（ASMX）Web 服务现代化。

DR

参见灾难恢复。

漂移检测

跟踪与基准配置的偏差。例如，您可以使用 AWS CloudFormation 来检测系统资源中的偏差，也可以使用 AWS Control Tower 来检测着陆区中可能影响监管要求合规性的变化。

DVSM

参见开发价值流映射。

# E

EDA

参见探索性数据分析。

边缘计算

该技术可提高位于 IoT 网络边缘的智能设备的计算能力。与云计算相比，边缘计算可以减少通信延迟并缩短响应时间。

加密

一种将人类可读的纯文本数据转换为密文的计算过程。

加密密钥

由加密算法生成的随机位的加密字符串。密钥的长度可能有所不同，而且每个密钥都设计为不可预测且唯一。

字节顺序

字节在计算机内存中的存储顺序。大端序系统先存储最高有效字节。小端序系统先存储最低有效字节。

端点

参见服务端点。

端点服务

一种可以在虚拟私有云（VPC）中托管，与其他用户共享的服务。您可以使用其他 AWS 账户 或 AWS Identity and Access Management (IAM) 委托人创建终端节点服务， AWS PrivateLink 并向其授予权限。这些账户或主体可通过创建接口 VPC 端点来私密地连接到您的端点服务。有关更多信息，请参阅 Amazon Virtual Private Cloud（Amazon VPC）文档中的创建端点服务。

企业资源规划 (ERP)

一种自动化和管理企业关键业务流程（例如会计、MES 和项目管理）的系统。

信封加密

用另一个加密密钥对加密密钥进行加密的过程。有关更多信息，请参阅 AWS Key Management Service (AWS KMS) 文档中的信封加密。

environment

正在运行的应用程序的实例。以下是云计算中常见的环境类型：

- 开发环境 — 正在运行的应用程序的实例，只有负责维护应用程序的核心团队才能使用。开发环境用于测试更改，然后再将其提升到上层环境。这类环境有时称为测试环境。
- 下层环境 — 应用程序的所有开发环境，比如用于初始构建和测试的环境。
- 生产环境 — 最终用户可以访问的正在运行的应用程序的实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上层环境 — 除核心开发团队以外的用户可以访问的所有环境。这可能包括生产环境、预生产环境和用户验收测试环境。

epic

在敏捷方法学中，有助于组织工作和确定优先级的功能类别。epics 提供了对需求和实施任务的总体描述。例如， AWS CAF 安全史诗包括身份和访问管理、侦探控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中 epics 的更多信息，请参阅计划实施指南。

ERP

参见企业资源规划。

探索性数据分析（EDA）

分析数据集以了解其主要特征的过程。您收集或汇总数据，并进行初步调查，以发现模式、检测异常并检查假定情况。EDA 通过计算汇总统计数据和创建数据可视化得以执行。

# F

事实表

星形架构中的中心表。它存储有关业务运营的定量数据。通常，事实表包含两种类型的列：包含度量的列和包含维度表外键的列。

失败得很快

一种使用频繁和增量测试来缩短开发生命周期的理念。这是敏捷方法的关键部分。

故障隔离边界

在中 AWS Cloud，诸如可用区 AWS 区域、控制平面或数据平面之类的边界，它限制了故障的影响并有助于提高工作负载的弹性。有关更多信息，请参阅AWS 故障隔离边界。

功能分支

参见分支。

特征

您用来进行预测的输入数据。例如，在制造环境中，特征可能是定期从生产线捕获的图像。

特征重要性

特征对于模型预测的重要性。这通常表示为数值分数，可以通过各种技术进行计算，例如 Shapley 加法解释（SHAP）和积分梯度。有关更多信息，请参阅机器学习模型的可解释性：AWS。

功能转换

为 ML 流程优化数据，包括使用其他来源丰富数据、扩展值或从单个数据字段中提取多组信息。这使得 ML 模型能从数据中获益。例如，如果您将“2021-05-27 00:15:37”日期分解为“2021”、“五月”、“星期四”和“15”，则可以帮助学习与不同数据成分相关的算法学习精细模式。

FGAC

请参阅精细的访问控制。

精细访问控制 (FGAC)

使用多个条件允许或拒绝访问请求。

快闪迁移

一种数据库迁移方法，它使用连续的数据复制，通过更改数据捕获在尽可能短的时间内迁移数据，而不是使用分阶段的方法。目标是将停机时间降至最低。

# G

地理封锁

请参阅地理限制。

地理限制（地理阻止）

在 Amazon 中 CloudFront，一种阻止特定国家/地区的用户访问内容分发的选项。您可以使用允许列表或阻止列表来指定已批准和已禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档中的限制内容的地理分布。

GitFlow 工作流程

一种方法，在这种方法中，下层和上层环境在源代码存储库中使用不同的分支。Gitflow 工作流程被认为是传统的，而基于主干的工作流程是现代的首选方法。

全新策略

在新环境中缺少现有基础设施。在对系统架构采用全新策略时，您可以选择所有新技术，而不受对现有基础设施（也称为棕地）兼容性的限制。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

防护机制

一种高级规则，用于跨组织单位（OU）管理资源、策略和合规性。预防性防护机制会执行策略以确保符合合规性标准。它们是使用服务控制策略和 IAM 权限边界实现的。侦测性防护机制会检测策略违规和合规性问题，并生成警报以进行修复。它们通过使用 AWS Config、、Amazon、 AWS Security Hub GuardDuty AWS Trusted Advisor、Amazon Inspector 和自定义 AWS Lambda 支票来实现。

# H

HA

参见高可用性。

异构数据库迁移

将源数据库迁移到使用不同数据库引擎的目标数据库（例如，从 Oracle 迁移到 Amazon Aurora）。异构迁移通常是重新架构工作的一部分，而转换架构可能是一项复杂的任务。AWS 提供了 AWS SCT 来帮助实现架构转换。

高可用性 (HA)

在遇到挑战或灾难时，工作负载无需干预即可连续运行的能力。HA 系统旨在自动进行故障转移、持续提供良好性能，并以最小的性能影响处理不同负载和故障。

历史数据库现代化

一种用于实现运营技术 (OT) 系统现代化和升级以更好满足制造业需求的方法。历史数据库是一种用于收集和存储工厂中各种来源数据的数据库。

同构数据库迁移

将源数据库迁移到共享同一数据库引擎的目标数据库（例如，从 Microsoft SQL Server 迁移到 Amazon RDS for SQL Server）。同构迁移通常是更换主机或更换平台工作的一部分。您可以使用本机数据库实用程序来迁移架构。

热数据

经常访问的数据，例如实时数据或近期的转化数据。这些数据通常需要高性能存储层或存储类别才能提供快速的查询响应。

修补程序

针对生产环境中关键问题的紧急修复。由于其紧迫性，修补程序通常是在典型的 DevOps 发布工作流程之外进行的。

hypercare 周期

割接之后，迁移团队立即管理和监控云中迁移的应用程序以解决任何问题的时间段。通常，这个周期持续 1-4 天。在 hypercare 周期结束时，迁移团队通常会将应用程序的责任移交给云运营团队。

## I

IaC

参见基础架构即代码。

基于身份的策略

附加到一个或多个 IAM 委托人的策略，用于定义他们在 AWS Cloud 环境中的权限。

空闲应用程序

90 天内平均 CPU 和内存使用率在 5% 到 20% 之间的应用程序。在迁移项目中，通常会停用这些应用程序或将其保留在本地。

IIoT

参见工业物联网。

不可变的基础架构

一种为生产工作负载部署新基础架构，而不是更新、修补或修改现有基础架构的模型。不可变基础架构本质上比可变基础架构更一致、更可靠、更可预测。有关更多信息，请参阅 Well-Architected Framework 中的使用不可变基础架构 AWS 部署最佳实践。

入站（入口）VPC

在 AWS 多账户架构中，一种接受、检查和路由来自应用程序外部的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

增量迁移

一种割接策略，在这种策略中，您可以将应用程序分成小部分进行迁移，而不是一次性完整割接。例如，您最初可能只将几个微服务或用户迁移到新系统。在确认一切正常后，您可以逐步迁移其他微服务或用户，直到停用遗留系统。这种策略降低了大规模迁移带来的风险。

工业 4.0

该术语由克劳斯·施瓦布（Klaus Schwab）于2016年推出，指的是通过连接、实时数据、自动化、分析和人工智能/机器学习的进步实现制造流程的现代化。

基础设施

应用程序环境中包含的所有资源和资产。

基础设施即代码（IaC）

通过一组配置文件预置和管理应用程序基础设施的过程。IaC 旨在帮助您集中管理基础设施、实现资源标准化和快速扩展，使新环境具有可重复性、可靠性和一致性。

工业物联网（IIoT）

在工业领域使用联网的传感器和设备，例如制造业、能源、汽车、医疗保健、生命科学和农业。有关更多信息，请参阅制定工业物联网（IIoT）数字化转型策略。

检查 VPC

在 AWS 多账户架构中，一种集中式 VPC，用于管理 VPC（相同或不同 AWS 区域）、互联网和本地网络之间的网络流量检查。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

物联网（IoT）

由带有嵌入式传感器或处理器的连接物理对象组成的网络，这些传感器或处理器通过互联网或本地通信网络与其他设备和系统进行通信。有关更多信息，请参阅什么是 IoT？

可解释性

它是机器学习模型的一种特征，描述了人类可以理解模型的预测如何取决于其输入的程度。有关更多信息，请参阅使用 AWS 实现机器学习模型的可解释性。

IoT

参见物联网。

IT 信息库（ITIL）

提供 IT 服务并使这些服务符合业务要求的一套最佳实践。ITIL 是 ITSM 的基础。

IT 服务管理（ITSM）

为组织设计、实施、管理和支持 IT 服务的相关活动。有关将云运营与 ITSM 工具集成的信息，请参阅运营集成指南。

ITIL

请参阅 IT 信息库。

ITSM

请参阅 IT 服务管理。

# L

基于标签的访问控制 (LBAC)

强制访问控制 (MAC) 的一种实施方式，其中明确为用户和数据本身分配了安全标签值。用户安全标签和数据安全标签之间的交集决定了用户可以看到哪些行和列。

登录区

landing zone 是一个架构精良的多账户 AWS 环境，具有可扩展性和安全性。这是一个起点，您的组织可以从这里放心地在安全和基础设施环境中快速启动和部署工作负载和应用程序。有关登录区的更多信息，请参阅设置安全且可扩展的多账户 AWS 环境。

大规模迁移

迁移 300 台或更多服务器。

LBAC

参见基于标签的访问控制。

最低权限

授予执行任务所需的最低权限的最佳安全实践。有关更多信息，请参阅 IAM 文档中的应用最低权限许可。

直接迁移

见 7 R。

小端序系统

一个先存储最低有效字节的系统。另请参见字节顺序。

下层环境

参见环境。

# M

机器学习（ML）

一种使用算法和技术进行模式识别和学习的人工智能。ML 对记录的数据（例如物联网（IoT）数据）进行分析和学习，以生成基于模式的统计模型。有关更多信息，请参阅机器学习。

主分支

参见分支。

恶意软件

旨在危害计算机安全或隐私的软件。恶意软件可能会破坏计算机系统、泄露敏感信息或获得未经授权的访问。恶意软件的示例包括病毒、蠕虫、勒索软件、特洛伊木马、间谍软件和键盘记录器。

托管服务

AWS 服务 它 AWS 运行基础设施层、操作系统和平台，您可以访问端点来存储和检索数据。亚马逊简单存储服务 (Amazon S3) Service 和 Amazon DynamoDB 就是托管服务的示例。这些服务也称为抽象服务。

制造执行系统 (MES)

一种软件系统，用于跟踪、监控、记录和控制在车间将原材料转化为成品的生产过程。

MAP

参见迁移加速计划。

机制

一个完整的过程，在此过程中，您可以创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运行过程中自我增强和改进的循环。有关更多信息，请参阅在 Well-Architect AWS ed 框架中构建机制。

成员账户

AWS 账户 除属于组织中的管理账户之外的所有账户 AWS Organizations。一个账户一次只能是一个组织的成员。

MES

参见制造执行系统。

消息队列遥测传输 (MQTT)

一种基于发布/订阅模式的轻量级 machine-to-machine (M2M) 通信协议，适用于资源受限的物联网设备。

微服务

一种小型独立服务，通过明确定义的 API 进行通信，通常由小型独立团队拥有。例如，保险系统可能包括映射到业务能力（如销售或营销）或子域（如购买、理赔或分析）的微服务。微服务的好处包括敏捷、灵活扩展、易于部署、可重复使用的代码和恢复能力。有关更多信息，请参阅使用 AWS 无服务器服务集成微服务。

微服务架构

一种使用独立组件构建应用程序的方法，这些组件将每个应用程序进程作为微服务运行。这些微服务使用轻量级 API 通过明确定义的接口进行通信。该架构中的每个微服务都可以更新、部署和扩展，以满足对应用程序特定功能的需求。有关更多信息，请参阅在上实现微服务。 AWS

迁移加速计划（MAP）

AWS 该计划提供咨询支持、培训和服务，以帮助组织为迁移到云奠定坚实的运营基础，并帮助抵消迁移的初始成本。MAP 提供了一种以系统的方式执行遗留迁移的迁移方法，以及一套用于自动执行和加速常见迁移场景的工具。

大规模迁移

将大部分应用程序组合分波迁移到云中的过程，在每一波中以更快的速度迁移更多应用程序。本阶段使用从早期阶段获得的最佳实践和经验教训，实施由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付简化工作负载的迁移。这是 AWS 迁移策略的第三阶段。

迁移工厂

跨职能团队，通过自动化、敏捷的方法简化工作负载迁移。迁移工厂团队通常包括运营、业务分析师和所有者、迁移工程师、开发 DevOps 人员和冲刺专业人员。20% 到 50% 的企业应用程序组合由可通过工厂方法优化的重复模式组成。有关更多信息，请参阅本内容集中有关迁移工厂的讨论和云迁移工厂指南。

迁移元数据

有关完成迁移所需的应用程序和服务器的信息。每种迁移模式都需要一套不同的迁移元数据。迁移元数据的示例包括目标子网、安全组和 AWS 账户。

迁移模式

一种可重复的迁移任务，详细列出了迁移策略、迁移目标以及所使用的迁移应用程序或服务。示例：使用 AWS 应用程序迁移服务重新托管向 Amazon EC2 的迁移。

迁移组合评测（MPA）

一种在线工具，可提供信息，用于验证迁移到的业务案例。 AWS Cloud MPA 提供了详细的组合评测（服务器规模调整、定价、TCO 比较、迁移成本分析）以及迁移计划（应用程序数据分析和数据收集、应用程序分组、迁移优先级排序和波次规划）。所有 AWS 顾问和 APN 合作伙伴顾问均可免费使用 MPA 工具（需要登录）。

迁移准备情况评测（MRA）

使用 AWS CAF 深入了解组织的云就绪状态、确定优势和劣势以及制定行动计划以缩小已发现差距的过程。有关更多信息，请参阅迁移准备指南。MRA 是 AWS 迁移策略的第一阶段。

迁移策略

用于将工作负载迁移到的方法 AWS Cloud。有关更多信息，请参阅此词汇表中的 7 R 条目和动员组织以加快大规模迁移。

ML

参见机器学习。

现代化

将过时的（原有的或单体）应用程序及其基础设施转变为云中敏捷、弹性和高度可用的系统，以降低成本、提高效率和利用创新。有关更多信息，请参阅中的应用程序现代化策略。 AWS Cloud

现代化准备情况评估

一种评估方式，有助于确定组织应用程序的现代化准备情况；确定收益、风险和依赖关系；确定组织能够在多大程度上支持这些应用程序的未来状态。评估结果是目标架构的蓝图、详细说明现代化进程发展阶段和里程碑的路线图以及解决已发现差距的行动计划。有关更多信息，请参阅中的评估应用程序的现代化准备情况 AWS Cloud。

单体应用程序（单体式）

作为具有紧密耦合进程的单个服务运行的应用程序。单体应用程序有几个缺点。如果某个应用程序功能的需求激增，则必须扩展整个架构。随着代码库的增长，添加或改进单体应用程序的功能也会变得更加复杂。若要解决这些问题，可以使用微服务架构。有关更多信息，请参阅将单体分解为微服务。

MPA

参见迁移组合评估。

MQTT

请参阅消息队列遥测传输。

多分类器

一种帮助为多个类别生成预测（预测两个以上结果之一）的过程。例如，ML 模型可能会询问“这个产品是书、汽车还是手机？”或“此客户最感兴趣什么类别的产品？”

可变基础架构

一种用于更新和修改现有生产工作负载基础架构的模型。为了提高一致性、可靠性和可预测性，Well-Architect AWS ed Framework 建议使用不可变基础设施作为最佳实践。

# O

OAC

请参阅源站访问控制。

OAI

参见源访问身份。

OCM

参见组织变更管理。

离线迁移

一种迁移方法，在这种方法中，源工作负载会在迁移过程中停止运行。这种方法会延长停机时间，通常用于小型非关键工作负载。

OI

参见运营集成。

OLA

参见运营层协议。

在线迁移

一种迁移方法，在这种方法中，源工作负载无需离线即可复制到目标系统。在迁移过程中，连接工作负载的应用程序可以继续运行。这种方法的停机时间为零或最短，通常用于关键生产工作负载。

OPC-UA

参见开放流程通信-统一架构。

开放流程通信-统一架构 (OPC-UA)

一种用于工业自动化的 machine-to-machine （M2M）通信协议。OPC-UA 提供了数据加密、身份验证和授权方案的互操作性标准。

运营级别协议（OLA）

一项协议，阐明了 IT 职能部门承诺相互交付的内容，以支持服务水平协议（SLA）。

运营准备情况审查 (ORR)

一份问题清单和相关的最佳实践，可帮助您理解、评估、预防或缩小事件和可能的故障的范围。有关更多信息，请参阅 Well-Architecte AWS d Frame work 中的运营准备情况评估 (ORR)。

操作技术 (OT)

与物理环境配合使用以控制工业运营、设备和基础设施的硬件和软件系统。在制造业中，OT 和信息技术 (IT) 系统的集成是工业 4.0 转型的重点。

运营整合（OI）

在云中实现运营现代化的过程，包括就绪计划、自动化和集成。有关更多信息，请参阅运营整合指南。

组织跟踪

由 AWS CloudTrail 此创建的跟踪记录组织 AWS 账户 中所有人的所有事件 AWS Organizations。该跟踪是在每个 AWS 账户 中创建的，属于组织的一部分，并跟踪每个账户的活动。有关更多信息，请参阅 CloudTrail文档中的为组织创建跟踪。

组织变革管理（OCM）

一个从人员、文化和领导力角度管理重大、颠覆性业务转型的框架。OCM 通过加快变革采用、解决过渡问题以及推动文化和组织变革，帮助组织为新系统和战略做好准备和过渡。在 AWS 迁移策略中，该框架被称为人员加速，因为云采用项目需要变更的速度。有关更多信息，请参阅 OCM 指南。

来源访问控制（OAC）

在中 CloudFront，一个增强的选项，用于限制访问以保护您的亚马逊简单存储服务 (Amazon S3) 内容。OAC 全部支持所有 S3 存储桶 AWS 区域、使用 AWS KMS (SSE-KMS) 进行服务器端加密，以及对 S3 存储桶的动态PUT和DELETE请求。

来源访问身份（OAI）

在中 CloudFront，一个用于限制访问权限以保护您的 Amazon S3 内容的选项。当您使用 OAI 时，CloudFront 会创建一个 Amazon S3 可以对其进行身份验证的委托人。经过身份验证的委托人只能通过特定 CloudFront 分配访问 S3 存储桶中的内容。另请参阅 OAC，其中提供了更精细和增强的访问控制。

或者

参见运营准备情况审查。

OT

参见运营技术。

出站（出口）VPC

在 AWS 多账户架构中，一种处理从应用程序内部启动的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

# P

权限边界

附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可以拥有的最大权限。有关更多信息，请参阅 IAM 文档中的权限边界。

个人身份信息 (PII)

直接查看其他相关数据或与之配对时可用于合理推断个人身份的信息。PII 的示例包括姓名、地址和联系信息。

PII

查看个人身份信息。

playbook

一套预定义的步骤，用于捕获与迁移相关的工作，例如在云中交付核心运营功能。playbook 可以采用脚本、自动化运行手册的形式，也可以是操作现代化环境所需的流程或步骤的摘要。

PLC

参见可编程逻辑控制器。

PLM

参见产品生命周期管理。

策略

一个对象，可以在中定义权限（参见基于身份的策略）、指定访问条件（参见基于资源的策略）或定义组织中所有账户的最大权限 AWS Organizations （参见服务控制策略）。

多语言持久性

根据数据访问模式和其他要求，独立选择微服务的数据存储技术。如果您的微服务采用相同的数据存储技术，它们可能会遇到实现难题或性能不佳。如果微服务使用最适合其需求的数据存储，则可以更轻松地实现微服务，并获得更好的性能和可扩展性。有关更多信息，请参阅在微服务中实现数据持久性。

组合评测

一个发现、分析和确定应用程序组合优先级以规划迁移的过程。有关更多信息，请参阅评估迁移准备情况。

谓词

返回`true`或的查询条件`false`，通常位于子`WHERE`句中。

谓词下推

一种数据库查询优化技术，可在传输前筛选查询中的数据。这减少了必须从关系数据库检索和处理的数据量，并提高了查询性能。

预防性控制

一种安全控制，旨在防止事件发生。这些控制是第一道防线，帮助防止未经授权的访问或对网络的意外更改。有关更多信息，请参阅在 AWS上实施安全控制中的预防性控制。

主体

中 AWS 可以执行操作和访问资源的实体。此实体通常是 IAM 角色的根用户或用户。 AWS 账户有关更多信息，请参阅 IAM 文档中角色术语和概念中的主体。

隐私设计

一种贯穿整个工程化过程考虑隐私的系统工程方法。

私有托管区

私有托管区就是一个容器，其中包含的信息说明您希望 Amazon Route 53 如何响应一个或多个 VPC 中的某个域及其子域的 DNS 查询。有关更多信息，请参阅 Route 53 文档中的私有托管区的使用。

主动控制

一种安全控制措施，旨在防止部署不合规的资源。这些控件会在资源置备之前对其进行扫描。如果资源与控件不兼容，则不会对其进行配置。有关更多信息，请参阅 AWS Control Tower 文档中的控制参考指南，并参见在上实施安全控制中的主动控制 AWS。

产品生命周期管理 (PLM)

在产品的整个生命周期中，从设计、开发和上市，到成长和成熟，再到衰落和移除，对产品进行数据和流程的管理。

生产环境

参见环境。

可编程逻辑控制器 (PLC)

在制造业中，一种高度可靠、适应性强的计算机，用于监控机器并实现制造过程自动化。

假名化

用占位符值替换数据集中个人标识符的过程。假名化可以帮助保护个人隐私。假名化数据仍被视为个人数据。

发布/订阅（发布/订阅）

一种支持微服务间异步通信的模式，以提高可扩展性和响应能力。例如，在基于微服务的 MES 中，微服务可以将事件消息发布到其他微服务可以订阅的频道。系统可以在不更改发布服务的情况下添加新的微服务。

# Q

查询计划

一系列步骤，例如指令，用于访问 SQL 关系数据库系统中的数据。

查询计划回归

当数据库服务优化程序选择的最佳计划不如数据库环境发生特定变化之前时。这可能是由统计数据、约束、环境设置、查询参数绑定更改和数据库引擎更新造成的。

# R

RACI 矩阵

参见 “负责任、负责、咨询、知情”（RACI）。

勒索软件

一种恶意软件，旨在阻止对计算机系统或数据的访问，直到付款为止。

RASCI 矩阵

参见 “负责任、负责、咨询、知情”（RACI）。

RCAC

请参阅行和列访问控制。

只读副本

用于只读目的的数据库副本。您可以将查询路由到只读副本，以减轻主数据库的负载。

重新架构师

见 7 R。

恢复点目标 (RPO)

自上一个数据恢复点以来可接受的最长时间。这决定了从上一个恢复点到服务中断之间可接受的数据丢失情况。

恢复时间目标 (RTO)

服务中断和服务恢复之间可接受的最大延迟。

重构

见 7 R。

区域

地理区域内的 AWS 资源集合。每一个 AWS 区域 都相互隔离，彼此独立，以提供容错、稳定性和弹性。有关更多信息，请参阅指定 AWS 区域 您的账户可以使用的账户。

回归

一种预测数值的 ML 技术。例如，要解决“这套房子的售价是多少？”的问题 ML 模型可以使用线性回归模型，根据房屋的已知事实（如建筑面积）来预测房屋的销售价格。

重新托管

见 7 R。

版本

在部署过程中，推动生产环境变更的行为。

搬迁

见 7 R。

更换平台

见 7 R。

回购

见 7 R。

故障恢复能力

应用程序抵御中断或从中断中恢复的能力。在中规划弹性时，高可用性和灾难恢复是常见的考虑因素。 AWS Cloud有关更多信息，请参阅AWS Cloud 弹性。

基于资源的策略

一种附加到资源的策略，例如 AmazonS3 存储桶、端点或加密密钥。此类策略指定了允许哪些主体访问、支持的操作以及必须满足的任何其他条件。

责任、问责、咨询和知情（RACI）矩阵

定义参与迁移活动和云运营的所有各方的角色和责任的矩阵。矩阵名称源自矩阵中定义的责任类型：负责 (R)、问责 (A)、咨询 (C) 和知情 (I)。支持 (S) 类型是可选的。如果包括支持，则该矩阵称为 RASCI 矩阵，如果将其排除在外，则称为 RACI 矩阵。

响应性控制

一种安全控制，旨在推动对不良事件或偏离安全基线的情况进行修复。有关更多信息，请参阅在AWS上实施安全控制中的响应性控制。

保留

见 7 R。

退休

见 7 R。

旋转

定期更新密钥以使攻击者更难访问凭据的过程。

行列访问控制 (RCAC)

使用已定义访问规则的基本、灵活的 SQL 表达式。RCAC 由行权限和列掩码组成。

RPO

参见恢复点目标。

RTO

参见恢复时间目标。

运行手册

执行特定任务所需的一套手动或自动程序。它们通常是为了简化重复性操作或高错误率的程序而设计的。

# S

SAML 2.0

许多身份提供商 (IdPs) 使用的开放标准。此功能支持联合单点登录 (SSO)，因此用户无需在 IAM 中为组织中的所有人创建用户即可登录 AWS Management Console 或调用 AWS API 操作。有关基于 SAML 2.0 的联合身份验证的更多信息，请参阅 IAM 文档中的关于基于 SAML 2.0 的联合身份验证。

SCADA

参见监督控制和数据采集。

SCP

参见服务控制政策。

secret

在中 AWS Secrets Manager，您以加密形式存储的机密或受限信息，例如密码或用户凭证。它由密钥值及其元数据组成。密钥值可以是二进制、单个字符串或多个字符串。有关更多信息，请参阅 Secret s Manager 密钥中有什么？ 在 Secrets Manager 文档中。

安全控制

一种技术或管理防护机制，可防止、检测或降低威胁行为体利用安全漏洞的能力。安全控制主要有四种类型：预防性、侦测、响应式和主动式。

安全加固

缩小攻击面，使其更能抵御攻击的过程。这可能包括删除不再需要的资源、实施授予最低权限的最佳安全实践或停用配置文件中不必要的功能等操作。

安全信息和事件管理（SIEM）系统

结合了安全信息管理（SIM）和安全事件管理（SEM）系统的工具和服务。SIEM 系统会收集、监控和分析来自服务器、网络、设备和其他来源的数据，以检测威胁和安全漏洞，并生成警报。

安全响应自动化

一种预定义和编程的操作，旨在自动响应或修复安全事件。这些自动化可作为侦探或响应式安全控制措施，帮助您实施 AWS 安全最佳实践。自动响应操作的示例包括修改 VPC 安全组、修补 Amazon EC2 实例或轮换证书。

服务器端加密

在目的地对数据进行加密，由接收数据 AWS 服务 的人加密。

服务控制策略（SCP）

一种策略，用于集中控制 AWS Organizations的组织中所有账户的权限。SCP 为管理员可以委托给用户或角色的操作定义了防护机制或设定了限制。您可以将 SCP 用作允许列表或拒绝列表，指定允许或禁止哪些服务或操作。有关更多信息，请参阅 AWS Organizations 文档中的服务控制策略。

服务端点

的入口点的 URL AWS 服务。您可以使用端点，通过编程方式连接到目标服务。有关更多信息，请参阅 AWS 一般参考 中的 AWS 服务 端点。

服务水平协议（SLA）

一份协议，阐明了 IT 团队承诺向客户交付的内容，比如服务正常运行时间和性能。

服务级别指示器 (SLI)

对服务性能方面的衡量，例如其错误率、可用性或吞吐量。

服务级别目标 (SLO)

代表服务运行状况的目标指标，由服务级别指标衡量。

责任共担模式

描述您在云安全与合规方面共同承担 AWS 的责任的模型。 AWS 负责云的安全，而您则负责云中的安全。有关更多信息，请参阅责任共担模式。

暹粒

参见安全信息和事件管理系统。

单点故障 (SPOF)

应用程序的单个关键组件出现故障，可能会中断系统。

SLA

参见服务级别协议。

SLI

参见服务级别指标。

SLO

参见服务级别目标。

split-and-seed 模型

一种扩展和加速现代化项目的模式。随着新功能和产品发布的定义，核心团队会拆分以创建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的工作效率，支持快速创新。有关更多信息，请参阅中的分阶段实现应用程序现代化的方法。 AWS Cloud

恶作剧

参见单点故障。

星型架构

一种数据库组织结构，它使用一个大型事实表来存储交易数据或测量数据，并使用一个或多个较小的维度表来存储数据属性。此结构专为在数据仓库中使用或用于商业智能目的而设计。

strangler fig 模式

一种通过逐步重写和替换系统功能直至可以停用原有的系统来实现单体系统现代化的方法。这种模式用无花果藤作为类比，这种藤蔓成长为一棵树，最终战胜并取代了宿主。该模式是由 Martin Fowler 提出的，作为重写单体系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET（ASMX）Web 服务现代化。

子网

您的 VPC 内的一个 IP 地址范围。子网必须位于单个可用区中。

监控和数据采集 (SCADA)

在制造业中，一种使用硬件和软件来监控有形资产和生产操作的系统。

对称加密

一种加密算法，它使用相同的密钥来加密和解密数据。

综合测试

以模拟用户交互的方式测试系统，以检测潜在问题或监控性能。你可以使用 Amazon S CloudWatch ynthetics 来创建这些测试。

# T

标签

键值对，充当用于组织资源的元数据。 AWS 标签可帮助您管理、识别、组织、搜索和筛选资源。有关更多信息，请参阅标记您的 AWS 资源。

目标变量

您在监督式 ML 中尝试预测的值。这也被称为结果变量。例如，在制造环境中，目标变量可能是产品缺陷。

任务列表

一种通过运行手册用于跟踪进度的工具。任务列表包含运行手册的概述和要完成的常规任务列表。对于每项常规任务，它包括预计所需时间、所有者和进度。

测试环境

参见环境。

训练

为您的 ML 模型提供学习数据。训练数据必须包含正确答案。学习算法在训练数据中查找将输入数据属性映射到目标（您希望预测的答案）的模式。然后输出捕获这些模式的 ML 模型。然后，您可以使用 ML 模型对不知道目标的新数据进行预测。

中转网关

中转网关是网络中转中心，您可用它来互连 VPC 和本地网络。有关更多信息，请参阅 AWS Transit Gateway 文档中的什么是公交网关。

基于中继的工作流程

一种方法，开发人员在功能分支中本地构建和测试功能，然后将这些更改合并到主分支中。然后，按顺序将主分支构建到开发、预生产和生产环境。

可信访问权限

向您指定的服务授予权限，该服务可以代表您在其账户中执行任务。 AWS Organizations 当需要服务相关的角色时，受信任的服务会在每个账户中创建一个角色，为您执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的AWS Organizations 与其他 AWS 服务一起使用。

优化

更改训练过程的各个方面，以提高 ML 模型的准确性。例如，您可以通过生成标签集、添加标签，并在不同的设置下多次重复这些步骤来优化模型，从而训练 ML 模型。

双披萨团队

一个小 DevOps 团队，你可以用两个披萨来喂食。双披萨团队的规模可确保在软件开发过程中充分协作。

# U

不确定性

这一概念指的是不精确、不完整或未知的信息，这些信息可能会破坏预测式 ML 模型的可靠性。不确定性有两种类型：认知不确定性是由有限的、不完整的数据造成的，而偶然不确定性是由数据中固有的噪声和随机性导致的。有关更多信息，请参阅量化深度学习系统中的不确定性指南。

无差别任务

也称为繁重工作，即创建和运行应用程序所必需的工作，但不能为最终用户提供直接价值或竞争优势。无差别任务的示例包括采购、维护和容量规划。

上层环境

参见环境。

# V

vacuum 操作

一种数据库维护操作，包括在增量更新后进行清理，以回收存储空间并提高性能。

版本控制

跟踪更改的过程和工具，例如存储库中源代码的更改。

VPC 对等连接

两个 VPC 之间的连接，允许您使用私有 IP 地址路由流量。有关更多信息，请参阅 Amazon VPC 文档中的什么是 VPC 对等连接。

漏洞

损害系统安全的软件缺陷或硬件缺陷。

# W

热缓存

一种包含经常访问的当前相关数据的缓冲区缓存。数据库实例可以从缓冲区缓存读取，这比从主内存或磁盘读取要快。

暖数据

不常访问的数据。查询此类数据时，通常可以接受中速查询。

窗口函数

一个 SQL 函数，用于对一组以某种方式与当前记录相关的行进行计算。窗口函数对于处理任务很有用，例如计算移动平均线或根据当前行的相对位置访问行的值。

工作负载

一系列资源和代码，它们可以提供商业价值，如面向客户的应用程序或后端过程。

工作流

迁移项目中负责一组特定任务的职能小组。每个工作流都是独立的，但支持项目中的其他工作流。例如，组合工作流负责确定应用程序的优先级、波次规划和收集迁移元数据。组合工作流将这些资产交付给迁移工作流，然后迁移服务器和应用程序。

蠕虫

参见一次写入，多读。

WQF

请参阅 AWS 工作负载资格框架。

一次写入，多次读取 (WORM)

一种存储模型，它可以一次写入数据并防止数据被删除或修改。授权用户可以根据需要多次读取数据，但他们无法对其进行更改。这种数据存储基础架构被认为是不可变的。

# Z

零日漏洞利用

一种利用未修补漏洞的攻击，通常是恶意软件。

零日漏洞

生产系统中不可避免的缺陷或漏洞。威胁主体可能利用这种类型的漏洞攻击系统。开发人员经常因攻击而意识到该漏洞。

僵尸应用程序

平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常会停用这些应用程序。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。